Final Report: Handwritten Signature Verification

Ananya Singh
singh745@purdue.edu

Tao Sun
sun955@purdue.edu

1 TOPIC

The topic that we chose for our final project was Application of Existing Data Mining Algorithms. Our specific task is signature verification, i.e. determining whether two images of signature samples were written by the same person. Determining whether a signature is genuine or not is a subset of handwriting analysis. The reason this is interesting and innovative is due to its applicability in areas such as criminal justice, forensics, legal, forgery, historical document analysis and art authentication, which can save resources and time for various agencies and professionals. Our code can be found here: https://github.com/ananya-singhh/signature-verification.

2 LITERATURE REVIEW

With security risks surrounding forgery, identity theft, and similar areas, the task of handwritten signature verification is prevalent to providing sound solutions to these issues. Our main goal with this paper is to compare approaches to identify handwritten signatures and accurately determine whether or not they are forged.

Prior to our experiment, this task has been examined by other researchers who have looked into a number of other approaches. At a simplified level, Navid et al utilized simple models and Convolutional Neural Networks (CNNs) to determine the validity of signatures on the CEDAR Signature dataset Navid et al. [2]. The method proposed in their research, which hovered around 88% accuracy, involves thorough preprocessing of data and transitioning to grayscale before converting to a bitmap. This is then applied to a CNN of 3 layers with size (32,64) along with max pooling layers, which should work to extract features and result in a classification based on similarity.

A slightly more developed approach involves Shaikh et al's use of attention-based Siamese Neural Networks (SNN) in conjunction with a feature extractor called Inception-Resnet-v2 Shaikh et al. [3]. This strategy is best suited for multimodal inputs, especially fit for taking two input signatures and comparing them. The researchers were able to extract the features of signatures on the CEDAR Signature and AND datasets through a stem block and apply soft attention to reduce our focus on background pixels that have no use. This helped them achieve a staggering 96% accuracy.

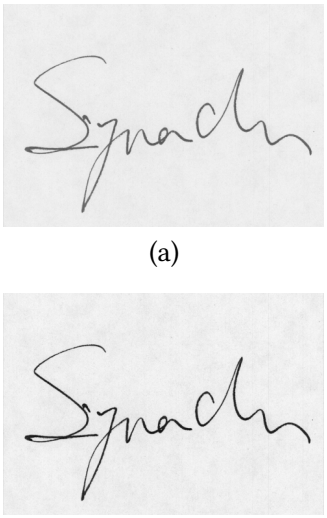
Another consideration to be taken is understanding the task to be a similarity comparison task, and proceed to use kNNs which Amato and Falchi undertook during their research in image comparisons on their self-constructed dataset of landmarks. With the use of Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF), local features can be extracted to perform kNN on the feature space. This gives an accurate measure of matches for image similarity, resulting in around an 85% accuracy.

3 DATASET

3.1 Data Collection / Search

We have chosen to work with the CEDAR dataset, created by the Center of Excellence for Document Analysis and Recognition at the University at Buffalo, New York. It is a database of off-line signatures for signature verification. There are 1,320 genuine signatures created by 55 individuals who contributed 24 signatures each. There are also 1,320 forged signatures contributed by other individuals, 24 forgeries per original writer. Each signature was scanned at 300 dpi gray-scale and binarized using a gray-scale histogram.

Since our task consists of having two images of signature samples as input, we create pairs of images from the CEDAR dataset for each writer. We do so by joining each genuine sample to another genuine sample of the same writer to be the positive examples, and joining each forged sample to a genuine sample for the same writer to be the negative examples. This gives us a total of 62,040 example pairs ($= 55 \cdot (24 \cdot 23 + 24 \cdot 24)$).



(a)

(b)

Figure 1: (a) Signature in RGB (b) Signature in Grayscale

3.2 Data Pre-processing / Augmentation

The dataset already had two steps of image pre-processing: salt pepper noise removal and slant normalization.

We recognized the importance and need of more pre-processing methods. To achieve this, we employed a variety of techniques such as:

- Converting the images from RGB to grayscale to reduce the image size for reduced data size, reduced complexity and more efficient training. Examples are shown in Figure 1.
- Cropping each image to the closest bounding box around the signature as shown in Figure 2 by finding the topmost, bottom-most, rightmost and left-most non-white pixel.
- Resizing images to have constant dimensions (256 x 512) since the models require constant input sizes. We did so by finding the average aspect ratio (≈ 0.5) of all the images to decide the aspect ratio for the resize. We recognize this might cause loss of information through image distortion and lose the original visual features of the signature, but since we are resizing them all to the same shape, we expected genuine signatures to be distorted in the same way.

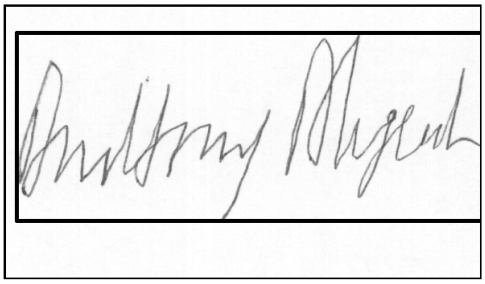


Figure 2: Visualization of Bounding Box to Extract the Signature. Bounding box of the whole image is also shown to compare with the bounding box of the signature.

- Normalizing the images to the range $[0, 1]$ for numerical stability, faster convergence, and easier training.
- We didn't apply any data augmentations since the task of handwriting verification doesn't have the classical potential variations in real-world scenarios where the model will be applied.

3.3 Data Exploration

We explored the data in many ways both before and after the data preprocessing. Here is what we found:

- Before conducting any preprocessing, we took a deeper look at the data visually by selecting random samples to take a manual look. We observed that the signatures weren't centered similarly and there was uneven whitespace around them. One such example is the image in Figure 2 where we can see that the signature isn't centered and has uneven whitespace surrounding it.
- Another thing we noticed was they were of uneven heights and widths as seen in the scatterplot in Figure 3, even though most of them seem to have similar dimension due to the big cluster in the middle. However, we do see some outliers for the forged signatures.
- We quickly realized that all this would pose a problem during classification and could make it harder or even impossible for models to learn to classify accurately. As a result, we decided to resize them to have constant dimensions using the methodology described in the previous section. After resizing the images, we took another look at

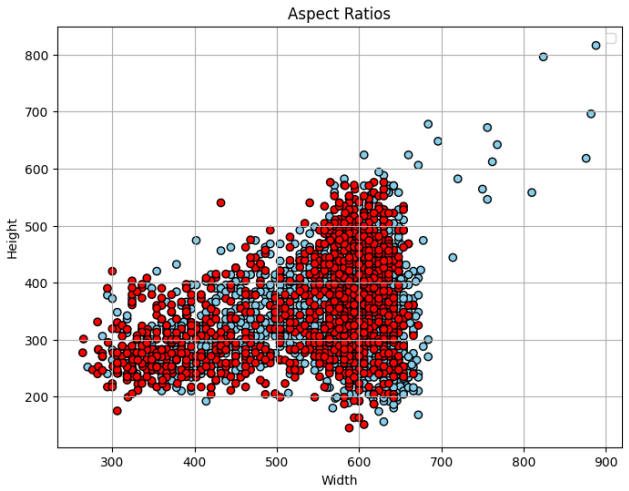


Figure 3: Blue points represent forged signatures and the red points represent genuine signatures.

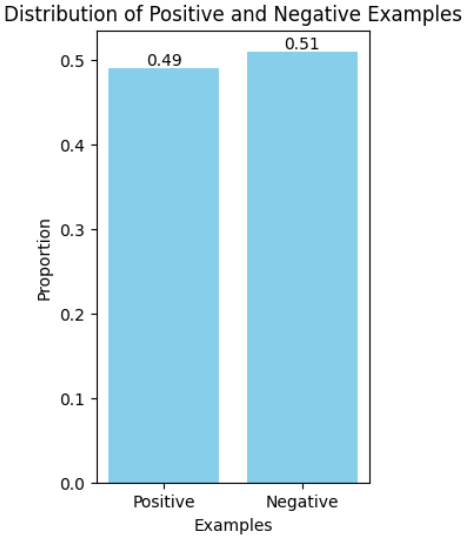


Figure 4

the data manually to verify that the resizing was done correctly.

- And after generating the data, we decided to take a look at the distribution of labels of the dataset to gain more insight about any potential problems we would face. As seen in Figure 7, we have a decently balanced dataset, the negative class not being too much more prevalent than the positive class. There are around 30360 positive and 31680 negative examples.

4 METHODS

After the dataset preprocessing and exploration, we divided the data into train and validation/test sets (80/20 split by writer ID). Then, we selected three models of focus from the papers we reviewed in the earlier section. We explored and implemented each of these models and then trained them on the training set and validated/tested them on the validation/test set. Finally, we optimized their performance using hyperparameter tuning to find the best performing model.

4.1 k-Nearest Neighbor

The k-Nearest Neighbor model is a classification model that utilizes features of inputs to make comparisons with other data points within the dataset. By taking in an input, we use a distance measurement (Euclidean) to find the nearest k data points with respect to the input. Then, we look at their classes from our training set and choose the majority out of these classes to classify the new input.

The main issue that we were faced with when designing the model and preparing the dataset was that for kNNs, we can only compare data with respect to single points. Thus, it would be a much more difficult task to determine the similarity between a pair of images, given that we would have to find a way to utilize both images in a kNN model.

The solution that we propose is to consider the differences of images and take the difference of our test images as a single input for our model space. For our training procedure, we paired each image to the 23 other images of the authors real handwriting, as well as to the 24 images of attempted forgeries at the authors handwriting, and then stored the differences of the pairs. However, we realized that the order of the images shouldn't matter, so we squared the differences to remove negative values from consideration, Conveniently, this also aids us during comparison since a lot of the images are similar to each other, so amplifying the difference is helpful for our task. Then whenever we needed to test, we would take the pair of input images, find the squared difference, and then classify it as "Real" or "Forged" based on our kNN's result over our learned data.

To address hyperparameter tuning, we looked at different values of k to train with and performed testing

afterwards to measure our accuracy. The best peforming k-value would then be taken to train the final model.

4.2 Convolutional Neural Network

Convolutional Neural Networks are neural networks that are designed to learn the features of images and visual input. They utilize various layers that have specific properties in feature extraction including Convolutional layers, Pooling layers, and Fully Connected Layers.

We first utilized Convolutional layers to apply filters that slide across the image performing mutiplication of the pixels in order to emphasize certain predetermined features. This was followed by a Pooling layer which downsamples the image while keeping important features. This works similarly to Convolutional layers in the sense that we apply a "filter" across the image again, however this filter takes a specific value (for example the max value) of the pixels in the filter and discards the rest, which reduces the size of the image. Finally, we passed our neurons through Fully Connected layers, which essentially condenses them to a specific dimension, usually the number of classes that can be classified. In our case, it ended up being two classes (Real or Forged).

While running the model, we saw that it was very easy for the model to overfit, so we introduced a few strategies to avoid this. First, we added Dropout layers to ignore a fraction of neurons for different passes during training. This makes it so that the model gets random training points, making it harder to overfit. To add on, we also added L2 regularization, which introduces a penalty term to the loss function, making it more likely to learn simpler weights rather than large weight values. This reduces the complexity of the model.

Our CNN had a similar issue to the kNN model, which was that comparing a pair of images would be a difficult task, so we replicated our preprocessed and paired data (from the kNN) to perform this task on single inputs.. To address hyperparameter tuning, we trained on different epochs, batch sizes, and learning rates to measure the best performance.

4.3 Siamese Neural Network

A Siamese Neural Network (SNN) is a class of neural network architectures that contain two or more identical sub-networks. “Identical” here means they have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks and it’s used to find similarities between inputs by comparing feature vectors. Essentially, they learn a similarity function and output embeddings such that the distance between similar inputs is minimized, while that between different inputs is maximized. They only need a few images to get better predictions. The ability to learn from very little data has made them more popular in recent years.

We implemented this model using PyTorch. Its input is a pair of images. The basic architecture is the same as the ResNet model architecture, modifying the first layer to accept single channel Grayscale images as input instead of 3 channels of RGB and the last layer to output an n-dimensional embedding rather than a single value for classification. It outputs two n-dimensional embedding vectors for each of the input images. For inference, we applied cosine similarity on the two output embeddings for each pair of images along with a threshold to determine the labels.

For training, we used Contrastive Loss, Adam Optimizer ($\beta = (0.9, 0.999)$) with L2 regularization ($\lambda = 1e-5$) to prevent overfitting, a Learning Rate scheduler ($\gamma = 0.1$), and batching of 32 for Gradient Descent. We loaded in pre-trained weights from ResNet and fine-tuned the model to save on the computational resources required to train Siamese Neural Networks, since we used our personal machines, and achieved great results doing so with just a single epoch.

As for hyperparameter tuning, we used grid search over three main parameters for model optimization. The varying hyperparameter options were the following:

- Learning rates: [1e-5, 1e-4, 1e-3]
- Output embedding sizes: [128, 256, 512]
- Cosine Similarity Thresholds: [0.4, 0.5, 0.6]

4.4 A Comparison of Assumptions and Formulations

Given these pieces of research, a comparison could be made between the different approaches. Primarily, an important distinction is that kNNs and CNNs usually expect a single instance input for classification, while SNNs are trained to perform similarity analysis given a pair of inputs. Thus, it would typically be difficult to verify handwritten signatures through the former two methods. A work-around could be to convert the

pair of images into a single value by taking the differences between image pixel values and training based on the similarity of differences. Regardless, these are clearly limitations of the methods. As an improvement, Siamese Neural Networks are designed to require less information to train yet still produce adequate results in pair-wise image comparisons, or even better results as shown in the results of the previous papers. With this being said, SNNs are very complex models and sacrifice training time in order to perform its functions, bringing their own limitations. Thus, while they seem ideal, their scalability falls short in comparison with other approaches. Additionally, it's also helpful to note that all three have the limitation that the input sizes must be constant across all methods.

Before diving into our findings, it's important to understand the assumptions made with each scenario. Primarily, kNNs perform under locality assumption, where similar data points are taken to be closely related in comparison with data with feature vectors that differ more. This applies to CNNs as well; however, with pooling layers and convolution of images, CNNs additionally assume the triviality of fine details of images in order to help with computation. Finally, SNNs work under the assumption that data is pairwise comparable, as well as embeddable in order to create a common feature space based on the embeddings. With this in mind, we proceeded with our experiments maintaining these assumptions as close as possible.

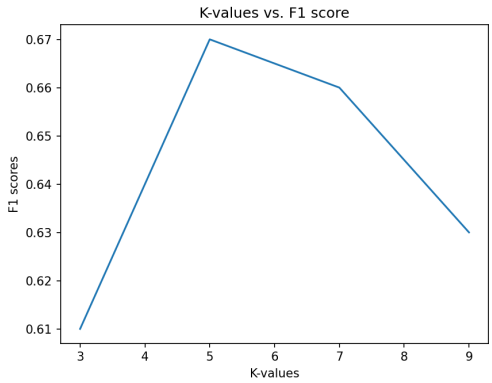


Figure 5

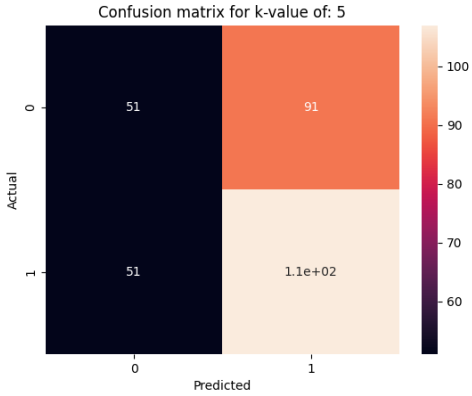


Figure 6

5 PERFORMANCE

5.1 k-Nearest Neighbor

After running our model on our pre-processed data and performing parameter tuning, we chose our best performing models to summarize our results. For the kNN model, we managed to achieve a 70% accuracy performing on test data. This was chosen after running our data on the highest fitting k-value, which we ran with respect to accuracy. This helped us conclude that the highest accuracy was seen at a k-value of 5 (Figure 5).

In comparison to Amato and Falchi's findings [1], their use of local feature extraction techniques resulted in an accuracy of 85%. This difference could be caused by varying training methods, where our kNN measured similarity between differences in images, while Amato and Falchi's model measured the similarity in extracted features. Despite the difference in accuracies, both methods are effective ways of evaluating the data. In addition to accuracy, we also measured the F1-score of the model (a much better measurement for datasets of imbalanced classes) and got a score of 0.61. To visualize the imbalance, we visualize a confusion matrix that displays all of the true positives, false positives, true negatives, and false negatives (Figure 6).

5.2 Convolutional Neural Network

Following this, we examined the accuracy of our CNN model and saw that it was around 85% , which is very similar to the findings of Navid et. al. [2]. In their research, they found that their model performed at an

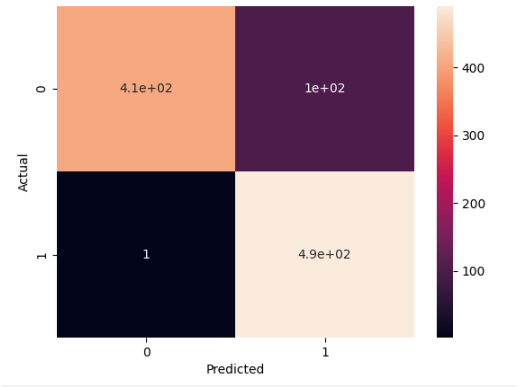


Figure 7

OES	T	F1	A
128	0.4	0.9871 / 1.0	0.9872 / 1.0
128	0.5	0.9926 / 1.0	0.9928 / 1.0
128	0.6	0.9970 / 1.0	0.9970 / 1.0
256	0.4	0.9984 / 0.9976	0.9985 / 0.9976
256	0.5	0.9964 / 0.998	0.9965 / 0.998
256	0.6	0.9989 / 1.0	0.9989 / 1.0
512	0.4	0.9985 / 0.998	0.9985 / 0.998
512	0.5	0.9991 / 0.998	0.9991 / 0.998
512	0.6	0.9851 / 1.0	0.9852 / 1.0

Table 1: Table for Learning Rate of 1e-5. Output Embedding Size (Column 1), Threshold (Column 2), F1 Score (Train/Validation) (Column 3), Accuracy (Train/Validation) (Column 4)

accuracy of 88%. This could jut be as a result of differences in preprocessed data or another minor difference, but the overall performance is essentially the same. After numerous trials and testing, we found the best parameters for this model to be a learning rate of 0.001, training on one epoch, and testing with a batch size of 32. Resulting from this, we calculated an F1-score of 0.91, a significant improvement from the kNN approach. As we take a look at the confusion matrix for this model, we notice that the classifications yield more true positives and true negatives, which explains the heightened performance of the CNNs (Figure 7).

5.3 Siamese Neural Network

For the Siamese Neural Network, there were 27 total experiment runs based on the different hyper-parameter choices listed in the previous section. Tables 1, 2, 3 summarize them in terms of F1 score and accuracy. As we can see from the tables, the performance of the models were quite similar and quite high across most hyper-parameter combinations. After these numerous trials and testing, we found the best parameters for this model to be the output embedding size of 128 and threshold of 0.4 as they had the highest train and validation accuracies and F1 scores for each of the learning rates. The confusion matrix for this model can be seen in 8. Since values of 1.0 in F1 score and 100% Accuracy seemed out of the ordinary, we made sure that none of the image pairs were common in either of the sets. Since there were around 12k images in the validation/test set, it

OES	T	F1	A
128	0.4	0.9975 / 1.0	0.9975 / 1.0
128	0.5	0.9937 / 0.984	0.9938 / 0.985
128	0.6	0.9989 / 0.939	0.999 / 0.940
256	0.4	0.999 / 0.9982	0.999 / 0.9983
256	0.5	0.9982 / 0.9938	0.9982 / 0.9939
256	0.6	0.9996 / 0.9924	0.9996 / 0.9925
512	0.4	0.9991 / 0.9924	0.9992 / 0.9925
512	0.5	0.9992 / 0.9762	0.9993 / 0.9762
512	0.6	0.9882 / 0.9885	0.9884 / 0.9886

Table 2: Table for Learning Rate of 1e-4. Output Embedding Size (Column 1), Threshold (Column 2), F1 Score (Train/Validation) (Column 3), Accuracy (Train/Validation) (Column 4)

seems that the model's performance isn't just a fluke and the dataset is probably easy!

When compared to the works of Shaikh et al [3], it seems our approach did better than their 96% accuracy, probably because of our use of pretrained weights and only using the CEDAR dataset. It also performs way better than the kNN and CNN approaches earlier, which makes sense since SNNs are formulated to model tasks just like this one.

6 INSIGHTS

(1) Based on the Siamese Neural Network results, it seems that this dataset was quite easy for the model. We are interested in looking at ways to

OES	T	F1	A
128	0.4	0.9923 / 1.0	0.9925 / 1.0
128	0.5	0.9962 / 1.0	0.9963 / 1.0
128	0.6	0.9983 / 1.0	0.9983 / 1.0
256	0.4	0.9841 / 0.9901	0.9843 / 0.9902
256	0.5	0.9781 / 0.7431	0.9783 / 0.6821
256	0.6	0.9967 / 0.9831	0.9968 / 0.9835
512	0.4	0.988 / 0.6987	0.9881 / 0.5782
512	0.5	0.984 / 0.9909	0.9843 / 0.991
512	0.6	0.9962 / 0.9981	0.9963 / 0.9981

Table 3: Table for Learning Rate of 1e-3. Output Embedding Size (Column 1), Threshold (Column 2), F1 Score (Train/Validation) (Column 3), Accuracy (Train/Validation) (Column 4)

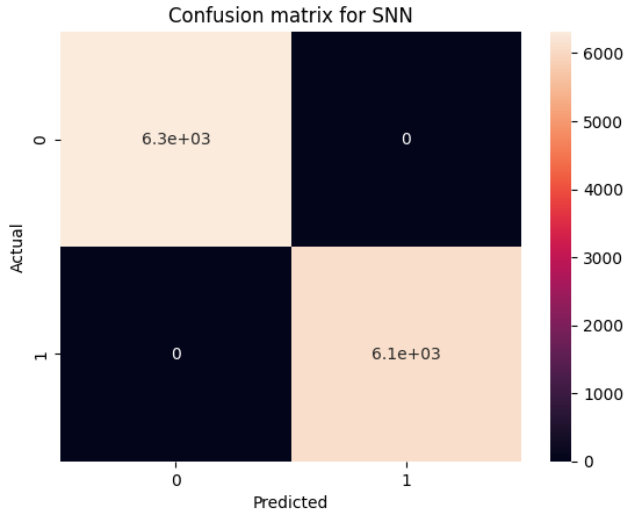


Figure 8: Confusion Matrix for Highest Performing SNN

make the task harder, such as finding more datasets (in different languages as well) and combining them or moving onto a harder and more general task of handwriting analysis.

(2) We gained insights into how to optimize data mining models and that techniques that work well is grid search, optimizers, and learning schedulers. We also learned about regularization and dropout layers that avoid overfitting.

(3) Siamese Neural Networks are great at tasks that can be modeled and reduced to learning a similarity function between two or more inputs.

(4) For kNNs and CNNs, taking the difference between the input images and using that as input worked better than expected for this task, once again possibly pointing to how easy the dataset probably is. It also gave us insights into how we can approach such tasks with models that do not classically work with the types of inputs the task requires.

(5) This project also emphasized it is harder to identify good forgeries and quite easy to distinguish between bad ones, which is to be expected and was seen in the types of mistakes the models were making.

7 EVALUATION

To evaluate, we will summarize the findings from our experiment first. For each of the models of kNN, CNN, and SNN, we finalized our outcomes as 70%, 85%, and 100% respectively. The F1-scores correlate in a similar fashion as well, with kNNs achieving 0.61, CNNs at 0.91, and 1.0 for Siamese. This compares well with the three findings that were explored during literature review, with the kNN performing decently, but paling in comparison to CNN and SNN models. Looking at confusion matrices, we gain a better understanding at why or how the models received the score that they did. For kNNs, we discovered that there were significantly more positives (true or false) than negatives. This could indicate that the model is less sensitive to differences and overclassifies "Real" pairings, which explains its underperformance. On the other hand, CNNs and SNNs have a much higher categorization of true positives and true negatives, which shows that they do not portray the same issue of lessened sensitivity, and perform much higher than the kNN.

Our original goal for this final project set by our proposal was to look at and compare three different methods of performing the task of signature verification and analyzing their procedures as well as results. All of these objectives have been shown and supported by both our findings as well as the reviewed works.

7.1 A Qualitative Approach to Evaluation for Siamese Neural Networks

We also decided to do some qualitative evaluation for Siamese Neutral Networks since it was our best performing model. Here are some things we noticed:

- Most of the models (that didn't perform perfectly) mainly had false positives that contributed to their lack in accuracy. If there were both false positives and false negatives, which was mainly in the case of the learning rate of $1e-3$ probably because the model didn't learn enough due to it, the rate of false positives was either close to or much higher than false negatives. This is bad for this particular task since false positives would mean mistaking a forged signature for a genuine one.
- For the false positives, when taking a manual look at the images, we noticed that it was because of how close that forged sample was to the genuine images, which makes sense.
- For the false negatives, when taking a manual look at the images, we noticed some level of variance in the original writer's signatures in comparison to the other writers, which also logically aligns with the model mistakes.
- As a result of this, we noticed the model making a larger proportion of mistakes for writers with a little more variance in their signatures.

8 CHALLENGES

We were met with our first challenge with the original, more broad task of handwriting analysis. The dataset we were looking at called the IAM dataset was a collection of image and writer pairs of handwriting samples of random words and sentences. The challenge with that was that the images were vastly different in dimensions due to the inevitable differences in lengths of words, phrases, and sentences the images were partitioned into. We faced difficulties in trying to figure out how to represent the input as resizing would distort the handwriting, but using some sort of embeddings could lose important physical features that might be necessary to classify the pairs of images. Moreover, the data we had didn't necessary have the same piece of text handwritten by another writer, which would additionally make it harder for the model to learn. For example, if we had pairs of the same word / phrase / sentence by many writers or even longer pieces of texts where we had more letters or words in common between them for the model to be able to "compare", the task would be much easier. So when during our preliminary literature review, we found the task of handwritten signature verification, we decided that alleviated most of the challenges that came with the prior dataset and task.

Our second challenge with the newly selected task was figuring out how to have images of constant sizes since most models require that and also addressing the problems we mentioned in the previous sections related to signature alignment and whitespaces. We wanted to do this in a way that the original signature is not distorted, but we settled on cropping the images to a respective bounding box around the signature and resize them to an averaged aspect ratio.

Other challenges we faced we figuring out what types of preprocessing and data augmentation techniques to apply and implementing/training/optimizing the CNNs and Siamese Networks since Computer Vision is a unfamiliar domain for my team. We learned everything from scratch and applied our knowledge.

Overall, the entire project was quite challenging and was a reach goal for our team due to the lack of knowledge and experience we had with Computer Vision tasks. This combined with the fact that this task wasn't a normal classification problem and instead worked with a pair of inputs added to this fact.

9 FUTURE WORKS

(1) Something that interests our team is applying these models (especially the Siamese Neural Network) on more different datasets (with different languages and scripts!) and combine them to make the task harder.
(2) We also think we should look into more types of models useful for this task, or design our own models and algorithms that would work well for this problem formulation.
(3) Since we didn't try apply any data augmentations as a preprocessing technique, we would like to explore such preprocessing techniques to see if we can further increase the generalizability of the models.

(4) As always, if we had unlimited resources and time we would try hyper-parameter tuning on even more parameters for each of the models like number of layers, hidden units, number of channels, image sizes, and more.
(5) We would like to conduct more thorough qualitative evaluation on all the models for comparison by looking at the specific mistakes and trying to identify any patterns between errors like writers that were more incorrectly classified and find the reasons and room for improvement.
(6) We would also like to explore a more general and harder task: given two images of handwriting samples determine whether or not they were written by the same person.

10 MEMBER CONTRIBUTIONS

Since we only have two members, we divided the tasks equally among ourselves. We had no issues working as a group, and are very satisfied with the division of labor and project results.

10.1 Ananya

I helped choose the topic and narrow it down to a more reasonable task from general handwriting analysis to signature verification. I also browsed to look for a dataset to use for the project, and I found the CEDAR dataset after looking through many options. Then, I generated the dataset to use for our project and found useful preprocessing techniques that are applied in Computer Vision tasks and preprocessed the entire data accordingly. I also conducted all the data exploration. On the models side, I looked through multiple research papers, and I found the existence of Siamese Networks and their usefulness for such similarity tasks. Then, I took on the role of implementing, training, tuning and evaluating them (quantitatively and qualitatively). I played an active role in completing all the project tasks such as the Project Proposal, Midterm Report, Presentation and finally this Final Report.

10.2 Tao

My contributions first consisted of helping choose the topic and analyzing the dataset to assist in determining how it should be preprocessed to best optimize our models. Additionally, I searched through research papers to determine which models to compare and decided to go with the kNN and CNN models which I ended up creating and training. A lot of my time went in to optimizing the models for performance as well as for efficiency and studying their sensitivity to different parameters. This helped me understand which would most effectively improve or impair the models. Finally, I also played an active role in completing all of the components for this project including the Project Proposal, Midterm Report, Presentation, and Final Report.

11 REFERENCES

[1] Giuseppe Amato and Fabrizio Falchi. "KNN Based Image Classification Relying on Local Feature Similarity". In: *Proceedings of the Third International Conference on SImilarity Search and APplications*. SISAP '10. Istanbul, Turkey: Association for Computing Machinery, 2010, pp. 101–108. ISBN: 9781450304207. DOI: 10.1145/1862344.1862360. URL: https://doi.org/10.1145/1862344.1862360.

[2] Shayekh Mohiuddin Ahmed Navid et al. "Signature Verification Using Convolutional Neural Network". In: *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*. 2019, pp. 35–39. DOI: 10.1109/RAAICON48939.2019.19.

[3] Mohammad Abuzar Shaikh et al. "Attention based Writer Independent Verification". In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, pp. 373–379. DOI: 10.1109/ICFHR2020.2020.00074.