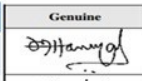
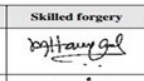
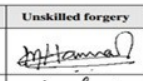
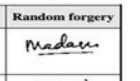
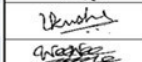
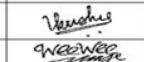
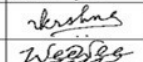
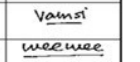
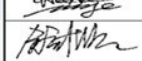
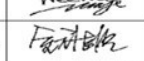
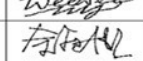
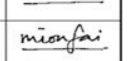
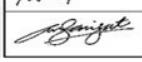
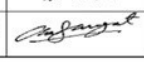
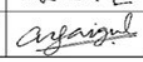
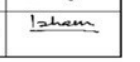
Offline Signature Verification with Siamese CNNs

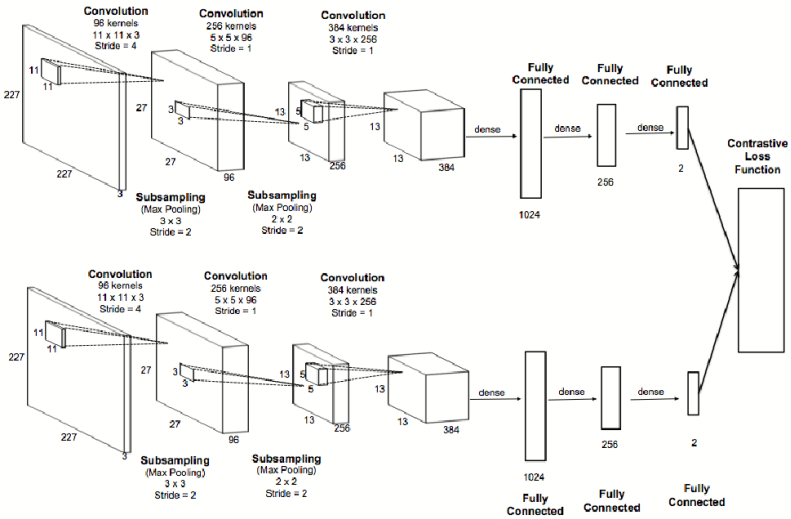
Abstract:

Offline signature verification is one of the most difficult tasks for forensic document examiners. Since the signatures are being scrutinized for fraud using only static images of the completed signature, a lot of crucial information for making these comparisons is lost. Additionally, a person's signature can vary from time to time due to factors such as emotional state, age, illness, and more. Forensic document examiners need methods that can generalize well to new data and detect skilled signature forgeries. This project applies Siamese CNNs with contrastive loss to distinguish forged signatures from genuine ones using the ICDAR 2011 Signature Dataset.

Customer	Genuine	Skilled forgery	Unskilled forgery	Random forgery
Person-1				
Person-2				
Person-3				
Person-4				
Person-5				

Deep Learning Architecture:

This project uses a supervised metric-based learning approach with Siamese Convolutional Neural Networks (Siamese CNNs). Siamese CNNs (as shown below) are two CNNs that run in parallel to each other to learn feature embeddings for images. The distance between the two embeddings were compared using contrastive loss and then those features were reused for one-shot learning to make predictions on the test data.



Data:

I used the offline Dutch user images from the ICDAR 2011 Signature dataset. This is a classic signature verification dataset and is used for the SigComp competition. The training set has 10 reference writers, each of whom has 12 genuine signatures and 12 skilled forgeries made for them. The test set has 54 reference writers, each with 12 genuine and 12 corresponding skilled forgeries. This gives a total of 362 training set signatures and 1932 test set signatures.

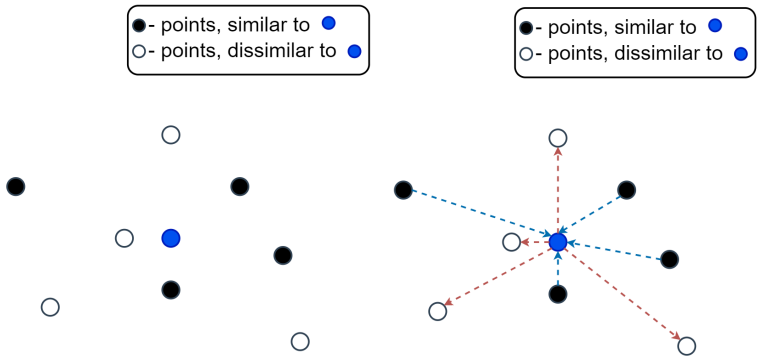
Upon doing some initial exploratory data analysis, I discovered that a lot of the images had varying sizes, varying background and pen colors, varying intensity (how hard the pen was pushed to paper, shown by darkness of writing), and were all very closely cropped to the signatures. There are also differences in the lighting and contrast of the pictures.

Preprocessing:

For the current version, I kept the preprocessing steps very simple. The first thing that I did was to resize all of the images to be 105 pixels x 105 pixels. Making sure all of the images are the same size is crucial to designing the neural nets, because the inputs for each neural net have to be 1) identical to each other and 2) 2D arrays of an expected size. I also wanted to make sure that any color variations in the training dataset images would not be a problem, so I converted all the images to grayscale.

Feature Vector Extraction:

The Siamese network was constructed to take inputs of image pairs and calculate embeddings for each image in the pair. Since there are natural variations in genuine signatures and we need a model that can not only tell apart forgeries, but can generalize well to never before seen signatures, we need to be able to extract **discriminative features**. Discriminative features can be used to distinguish between two different people's signatures. This now becomes a metric learning problem, as the deep learning model is not looking for a hyperplane to separate images, but is trying to reorganize the input space in a way that brings similar samples together and distances dissimilar samples.



Defining the Loss Function:

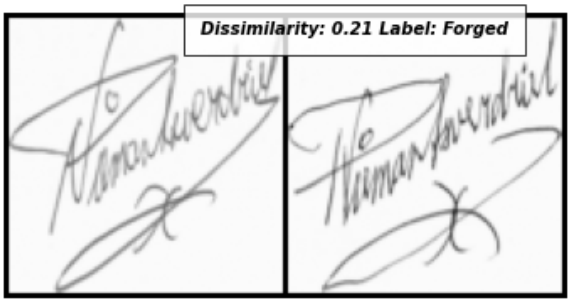
During the training process, the image pair embeddings were compared using the contrastive loss function that Yann LeCun proposed in 2005.

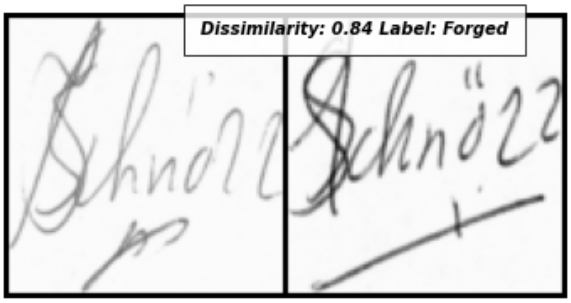
$$L(W, Y, \vec{X_1}, \vec{X_2}) = \\ (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

The function says that the loss is equal to the sum of the terms for the "dissimilar" cases (left-hand side) and "similar" cases (right-hand side). The contrastive loss function uses Euclidean distance (this can be any distance metric, though) to compare the embeddings. Similar samples are penalized from being distant from each other and dissimilar samples must stay outside the "margin". Any dissimilar samples that are closer than the set margin distance are penalized by the model.

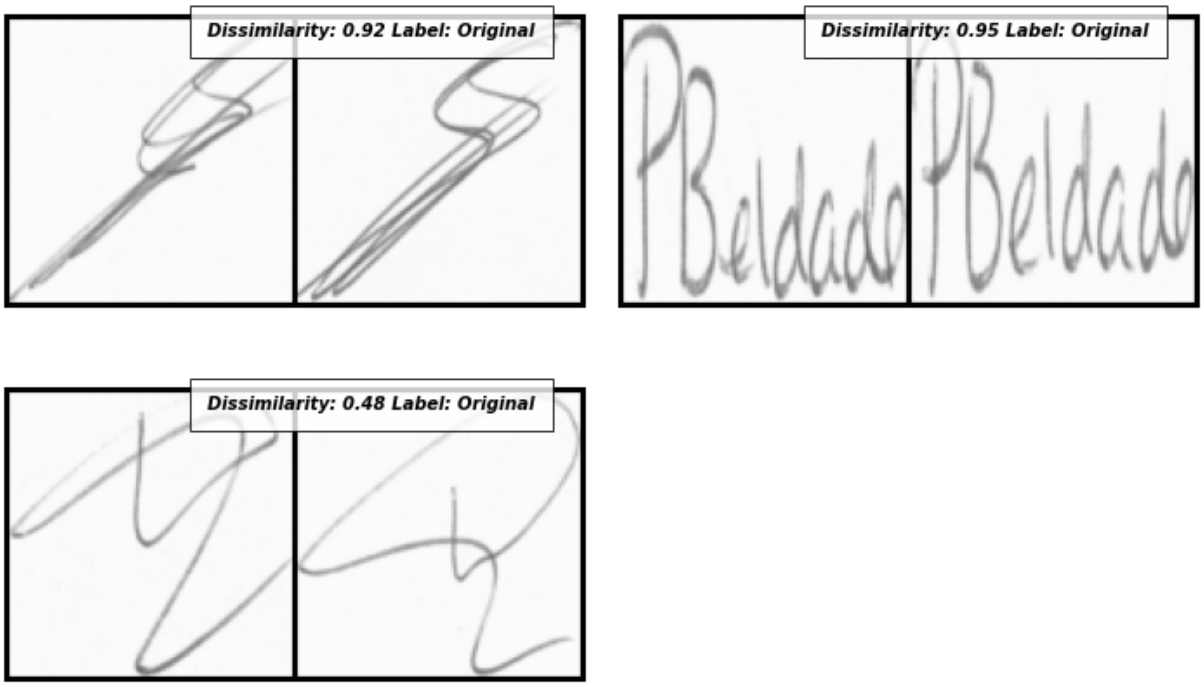
Calculating Similarity Scores:

To make these comparisons of "similar" or "dissimilar", there must be some method of calculating similarity scores. The similarity scores for this project were calculated using the Euclidean distance.





Some of the forgeries had higher dissimilarity to the originals than others. This could illustrate the difference between skilled and unskilled forgeries. In the example on the right, there are also a few extra strokes in the forged signature, so that could also play a part in why the dissimilarity is so high.



Most commonly, the genuine-genuine signature pairs had lower dissimilarity than the genuine-forged signature pairs. However, there are plenty of cases where the signatures are both genuine, but have very high dissimilarity between them, and this can be due to many different factors. These cases must be accounted for in the model creation and optimization process. Since there are many mixed cases, contrastive loss may not be the best option for the loss function or possibly Euclidean distance is not the best distance metric to use for this problem.

Optimization:

Root Mean Square Propagation, which is a gradient descent optimization algorithm, was used to optimize the model during training. The great feature of this optimizer is that if there is a parameter with a disproportionately large cost to the loss function, this optimizer penalizes that parameter in a way that prevents the algorithm from adapting to changes in that parameter too quickly compared to changes in the other parameters. One downside is that the learning rate and other hyperparameters need to be manually set beforehand, and the out of the box learning rate doesn't work well for all cases.

Parameters used for RMSProp: lr=1e-4, alpha=0.99, eps=1e-8, weight_decay=0.0005, momentum=0.9

Testing with One-Shot Learning:

The goal of one-shot learning is to have the model learn concepts from either one or few training examples. Thus, the pre-trained model was loaded and used on the test dataset to extract

feature vectors and make predictions based on the dissimilarity. The resulting model accuracy was 51%, which is to be expected due to the rather simple image preprocessing steps. Performance can be significantly improved with some changes to the preprocessing pipeline and further optimization.

Discussion:

After training the model, the prediction accuracy was only 51%, which is only 1% better than a naive classifier's performance. My main idea to improve model performance is to apply more image pre-processing techniques (normalization, invert grayscale images, salt and pepper noise removal) to the training dataset, as well as to artificially increase the size of the training dataset by applying image augmentations (ex. Flip, rotate, adjust contrast, adding noise). The ICDAR 2011 site also states that the 2009 data can also be used for training models, so I would use both the 2009 and 2011 data together in future iterations. Other improvements include using Adam optimizer instead of RMSProp, testing the performance using triplet loss instead of contrastive loss, and simply increasing the number of training epochs (currently 10 epochs).