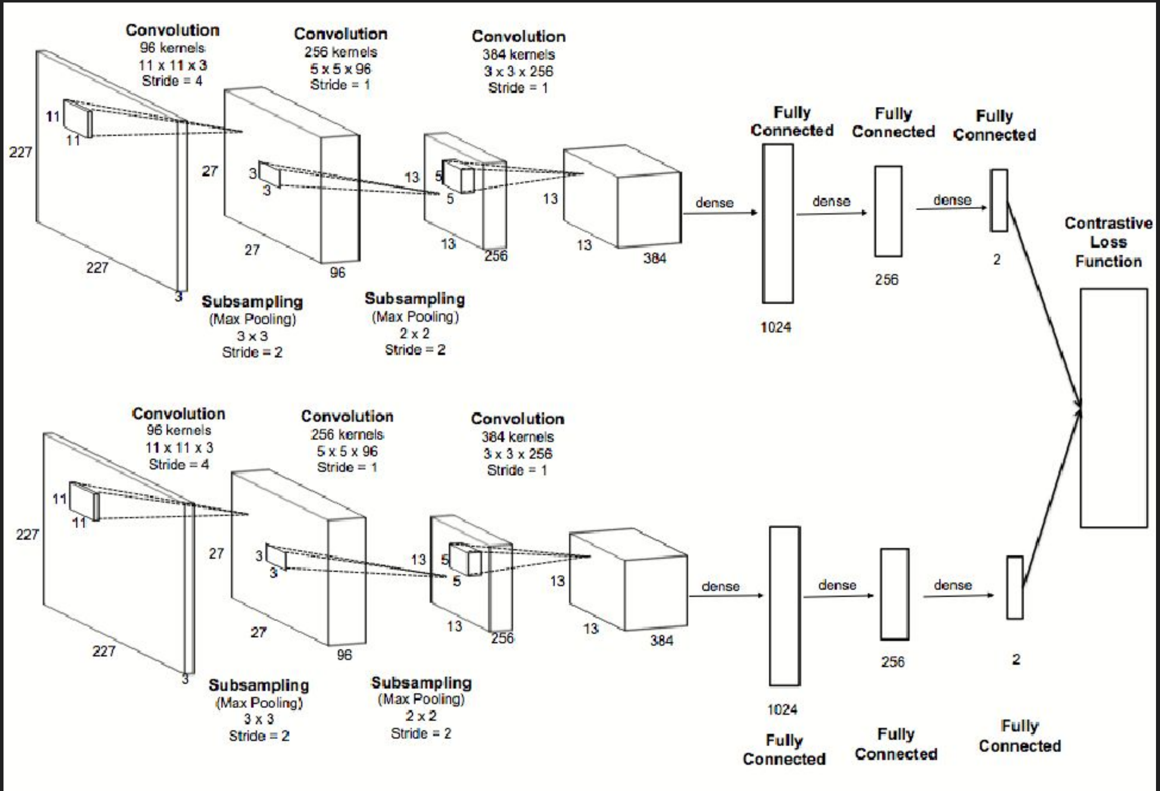
Offline Signature Verification Using Siamese-CNNs

Shakuntala Mitra
Springboard Data Science

What is Offline Signature Verification?

Can we distinguish a forged signature from a genuine one from only images of the signatures? This is one of the most difficult tasks for forensic document examiners!

Background: Siamese CNN Architecture

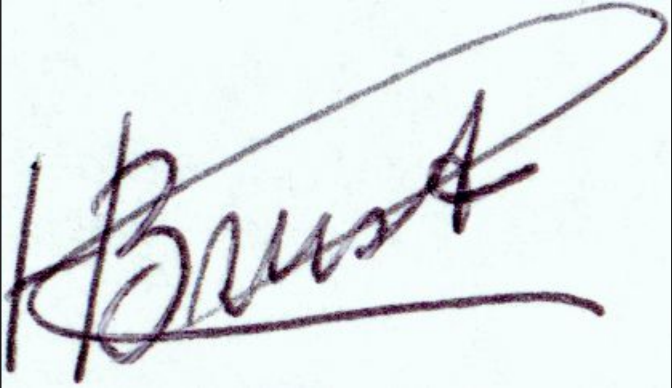


- Input is a **pair of images**
- Two identical CNNs work in parallel to extract features, each outputting an **embedding**
- The loss function compares the distances between the embeddings and then back-propagates

Data: The Signatures

- ICDAR 2011 Signature Dataset
- Training set: 10 reference writers, 12 genuine and 12 skilled forgeries of each signature
- Images have light blue background, grey or black pen strokes, different sizes

Genuine



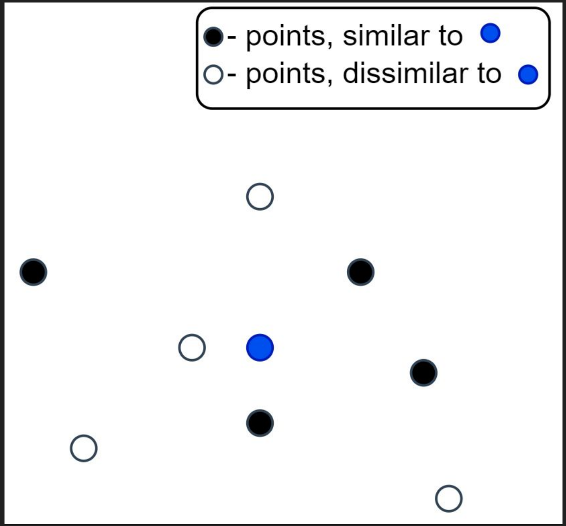
Forgery

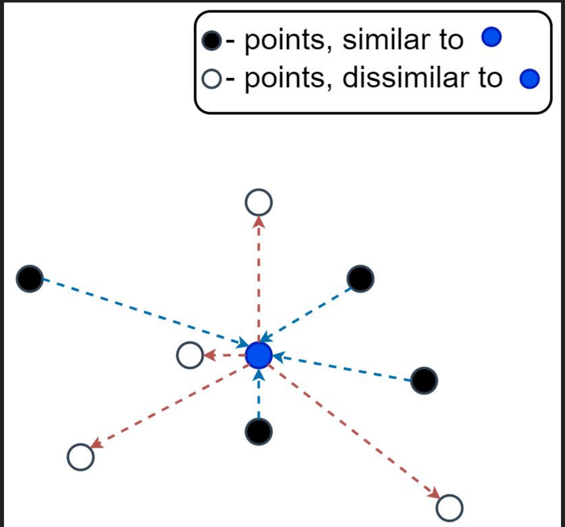
Preprocessing

- Images were resized to 105x105 pixels, converted to grayscale, and converted to tensors

Feature Vector Extraction

- Metric learning problem : want model to generalize to unseen data
- Need to extract **discriminative features**!!
- Bring similar images together, push dissimilar images away





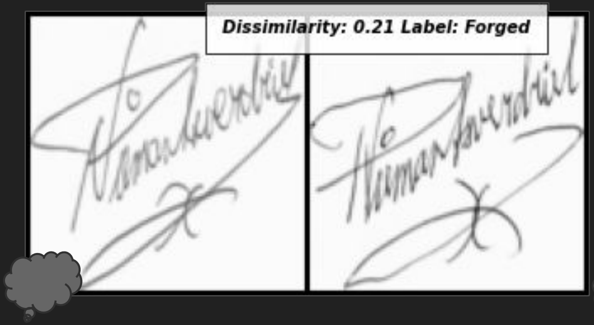
Defining the Loss Function

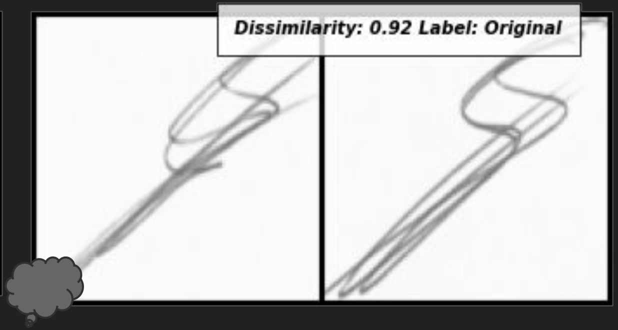
- Contrastive Loss: (from Yann Lecun)

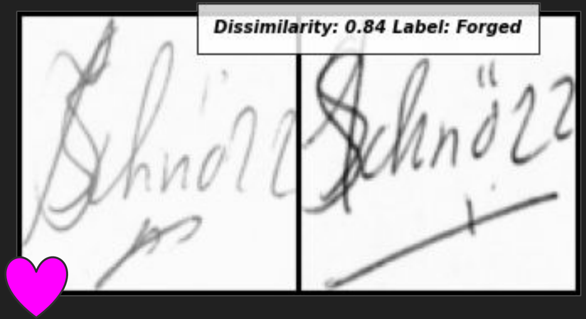
$$L(W, Y, \vec{X}_1, \vec{X}_2) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

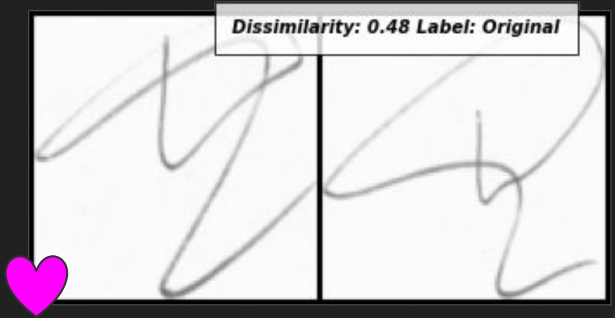
- Penalizes "similar" samples from being distant from each other
  - Distance used was Euclidean distance, but can be another metric
- "Margin" = minimum distance that "dissimilar" samples must keep from each other

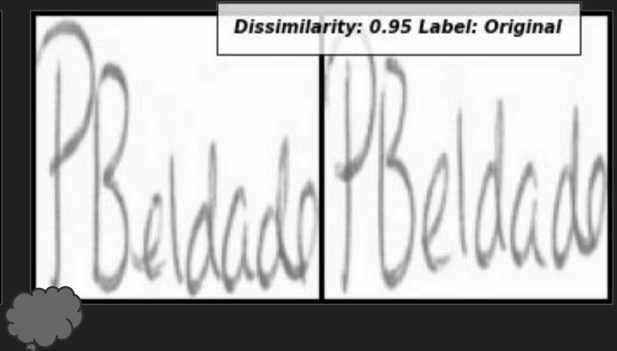
Similarity Score Calculation











Current model seems to be having problems with the more “complex” signatures. Looks like the more strokes there are and the closer together strokes are, the easier it is for the model to make mistakes.

RMSProp Optimizer

- Root Mean Square Propagation
- Gradient descent optimization algorithm
- Prevents algorithm from adapting to changes in one parameter too quickly compared to changes in other parameters
- Parameters used: lr=1e-4, alpha=0.99, eps=1e-8, weight_decay=0.0005, momentum=0.9

Testing with One-Shot Learning

- Learn from one or few training examples
- Extract feature vectors of input images from trained model => calculate dissimilarity between images => make prediction
- Result: 51% accuracy :(

Discussion: How to improve model performance?

- The loss function: Try triplet loss instead of contrastive loss?
- The optimizer: Try Adam instead of RMSProp
- More image preprocessing: Normalize, invert the black and white images, Salt pepper noise removal and slant normalization, filtering methods, etc.
- Increase the number of training epochs
- Image augmentations to increase training data diversity (ex. flip, rotate)

References:

1. https://medium.com/@subham.tiwari186/siamese-neural-network-for-one-shot-image-recognition-paper-analysis-44cf7f0c66cb
2. https://arxiv.org/pdf/1707.02131.pdf
3. https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e
4. https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
5. http://cs231n.stanford.edu/reports/2017/pdfs/801.pdf