**Siamese Triple Ranking Convolution Network in Signature Forgery Detection**

**Solution Design**

Signature forgery can be broadly of two types:

1. Blind Forgery: Where the forger has no idea what the signature to be forged looks like. This is easy to detect by machine because it is usually not very close to the appearance of a genuine signature.
2. Skilled Forgery: Either simulation or tracing, in which the forger has a sample of the signature to be forged. In this case, detecting fraud requires more sophisticated tools to differentiate minute but critical details between genuine and forged signatures.

We have developed a framework to detect fraud signatures using Image processing and Deep convolutional Siamese networks wherein a *deep triplet ranking network* is used to calculate the image embeddings. This is coupled with generalized linear model architecture with logistic loss functions and cross validation to arrive at the final model to label images as *authentic* or *forged*.

The best part about this framework is once the model is trained, we require just one base image to determine whether another signature image is genuine or not with one shot learning.

**Tech Stack**

* Python
* Keras
* Tensorflow
* OpenCV

**Data Used**

* Training: ICDAR 2009 Signature Verification Competition (SigComp2009)
* Test: ICDAR 2011 Signature Verification Competition (SigComp2011): Dutch Dataset

**Accuracy Attained**

**Execution Code**

All the code and environment for execution is in the [github link attached](https://github.com/ojaswinich/signature_verification). Please follow the instructions present in the [README.md](https://github.com/ojaswinich/signature_verification/blob/master/README.md) document in github.

**Detailed Algorithm: Siamese Triplet Convolution Network**

**Step 1: Dataset Preparation & Preprocessing**

In the first step, we contrive the dataset in a triplet formation. The data set used is from SigComp2009 which contains 5 unique genuine images each of 12 people and 5 forged images of each of these 12 people by 28 people in the training set. In the test set, the data contains 12 unique genuine images each of 80 people and 6 forged images of each of these 80 people by around 80 people. Since the training data by itself is too small (just 12 unique genuine signature patterns), we take both the training and test set from SigComp2009 as our training set.

We create all combinations of *triplets* from this data: this is done by taking an *anchor image* (genuine signature of a person) and placing it in conjunction with both a *positive sample* (another genuine signature of the same person) and a *negative sample* (a forged signature by someone else of the same person). A total of ---- such triplet combinations is obtained. We resize all the images and convert them to arrays to be passed in the CNN triplet model.

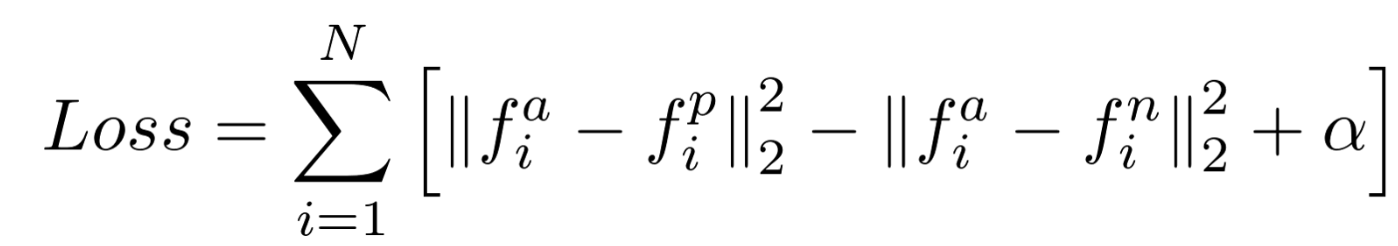
**Step 2: CNN Network Architecture & Embeddings**

Once we construct the training data in a triplet formation, we train the model based on convolutional Siamese network. Siamese convolution networks are twin networks with shared weights, which can be trained to learn the feature embeddings where similar observations are placed in proximity and dissimilar are placed apart.

----

Add fig1. Network architecture photo

We use the *Adam Optimizer* with *mean absolute error* for back propagation to get to the final encodings. The image embeddings so obtained are such that the dissimilarity between the anchor image and positive image must be low and the dissimilarity between the anchor image and the negative image must be high for every triplet. This kind of architecture ensures that even small differences in signatures can be captured in order to flag a skilled forgery as shown in Fig.2. The optimal hyperparameter combination that we obtained was ---



where,

*f(a) refers to the image encoding of the anchor a*

*f(p) refers to the image encoding of the positive p*

*f(n) refers to the image encoding of the negative n*

*alpha is a constant used to make sure that the network does not try to optimize towards f(a) - f(p) = f(a) - f(n) = 0*

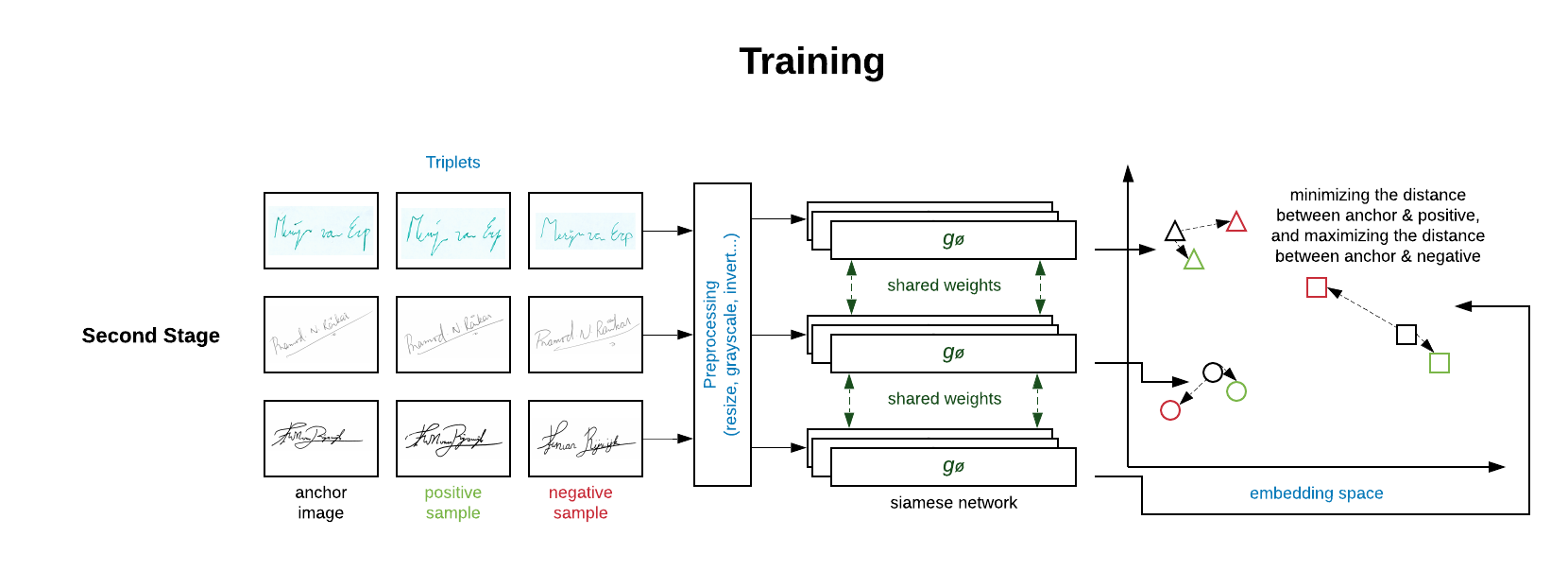


Fig.2: Deep Triplet Ranking CNN Network

**Step 3: Logistic Regression Model & Cross Validation**

Once we attain the final image embeddings for all the training images through the triplet loss architecture, we train a *generalized linear model* with *logistic loss function* to get the final model that declares any signature as genuine or forged against a base image signature. For training this logistic function, we arrange the images in a *pairwise* manner where each observation is *pair* of images, either both of a person’s genuine signature, or one of person’s genuine signature, and the other as person’s forged signature. These will have labels (class) genuine or fraud assigned to them respectively. A total of ---such pairs is obtained.

Since in the data, there is a lot more genuine-fraud image combinations than genuine-genuine image combinations, there is a major class imbalance problem which needs to be dealt with otherwise the model will be skewed to return fraud as the resultant class. We deal with this by under-sampling the fraud class, i.e., we take only a random subset of the genuine-fraud image combinations to make the classes in the data more balanced.

The logistic model is then trained on the subset of dataset pairs to determine that given a base signature image, whether the other image matches the base (genuine) or not (forged). We use cross validation to get to the final logistic model taking the corresponding differences between the embeddings of each of the pairs (128 length difference vector of embeddings) as the feature set and the class labels (genuine/ fraud) as the dependent variable *y*.

*Loss: g(sigmoid(Σ(w\* X(xa , xb) +b)),y)*

*where,*

*g() is the Logistic Loss Function*

*y is the response class (0/1)*

*xa , xb are the image embeddings of corresponding images in the image pair each of dimension m.*

*X(xa , xb) represents the entire set of difference based features computed from the embeddings in space Rm*

*w and b represent the final weights obtained from generalized linear logistic loss function where the response will be the images are same (0) or forged (1)(y).*

This ends the one-time training process for signature verification. We save the image encodings obtained in step 2 and the weights of trained logistic model obtained in step 3 as the final model for the real time verification process.

**Step 4: Final Framework Architecture**

Once the training process is completed, all we need is the trained model outputs (encodings and logistic model weights). Next, we created a database framework to save the original signatures of every new customer that the bank acquires against a unique ID. We pass these *base* images through the encodings we obtained in the training process and get the corresponding image embeddings. This is in the format of a vector of length 128. We precompute these embeddings and save them against the individuals’ unique ID in the database.

Now, whenever we acquire a new signature image against one of the unique ID’s that needs to be accessed to determine whether it is genuine or fraud, the framework passes that image through the encoding once again to get its image embeddings.

This new image embedding is then compared to its corresponding embedding of the *base* image of that individual to determine whether it is genuine or forged. This is done by taking the difference vector of the two embeddings and passing it through the logistic model to get the final prediction. If the resultant probability from the logistic model is low, then the framework declares the new image as a genuine, otherwise it is considered a forgery.

The framework flowchart is depicted below in Fig.3 and the test workflow is depicted in Fig4.

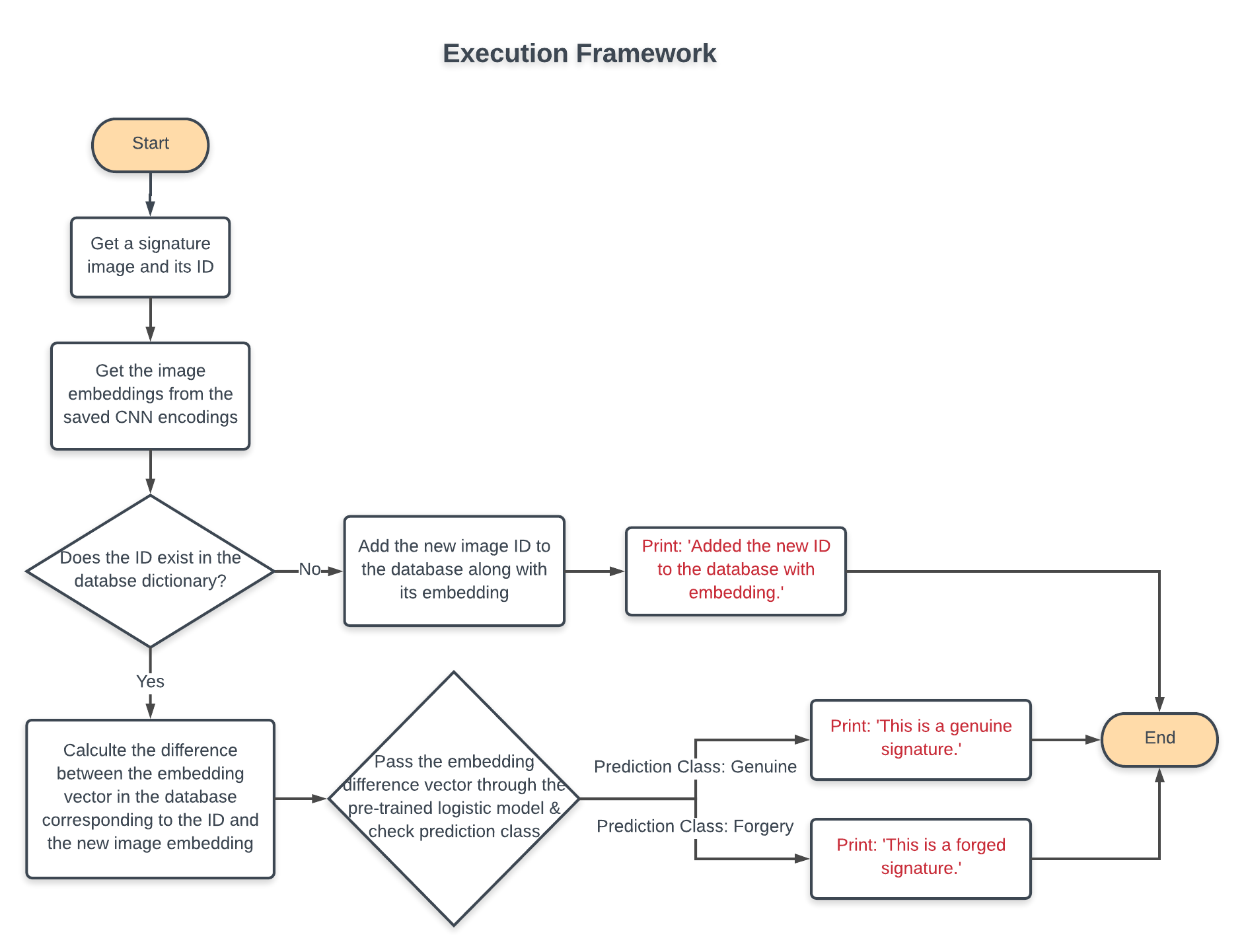


Fig.3: Execution Framework Flowchart

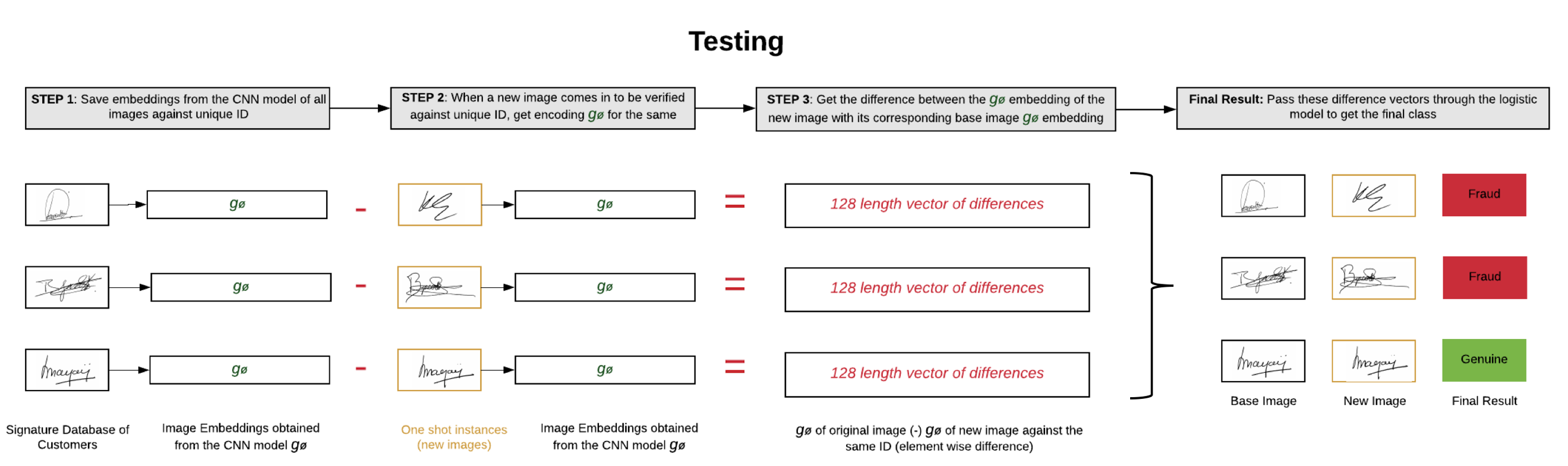


Fig.4: Testing process of determining whether signature is genuine or fraud

**Superiority of Triplet Loss over other Architectures**

The differences between the images of a genuine signature and its skilled forgery is at times very minute and is challenging to detect by even a trained eye. Deep Triplet Ranking CNN is a very powerful loss function used in the industry for face recognition (facenet). We created our custom CNN based on this loss wherein the image embeddings are created in such a way that the dissimilarity between the anchor image and positive image must be low and the dissimilarity between the anchor image and the negative image must be high for every triplet. This kind of architecture ensures that even small differences in signatures can be captured in order to flag a skilled forgery effectively.

**Deviations made from the proposal in Stage 1**

1. Hierarchical Framework: In the first stage of the competition we proposed a hierarchical framework to first detect blind forgeries and then subsequently detect skilled forgeries through two separately trained CNN models. We abandoned this approach in favor of a single CNN model that detects both skilled and blind forgeries in one-shot because the single CNN we trained along with the logistic model gave optimistic accuracies of >98% on unseen test data of both skilled and blind forgeries.
2. Bayesian optimization: We also proposed a Bayesian optimization and Gaussian Process in the first stage which would choose the model hyperparameters using a Bayesian Process. This is a novel approach that can save a lot on the time complexity of the grid search with selecting the best hyperparameter at each iteration. We did not implement this in the code solely because of time restrictions since the *threshold alpha* we chose in the triplet loss function already gave us a good cross validation accuracy.

**Challenges Faced**

1. Data Limitations: We used the data from SigComp2009 competition which contains unique genuine signatures of a total of less than 100 people. This is a major limitation since training CNN on so few unique signature patterns can lead to overfitting. Although more datasets relevant to genuine and fraud signatures exist (like SigComp2011 containing Dutch & Chinese signatures and BHSig260 containing Hindi & Bengali Signatures), we did not use them for our training process owing to time constraints since the data structure for both is messy and it would have taken a significant amount of time to get them in an organized pattern to form pairs/ triplets.
2. Imbalanced Data: The dataset used were skewed towards forged signatures and they did not have enough genuine signatures corresponding to each person. This led to a data imbalance problem in the training step of logistic regression. We used under-sampling of the forged class to deal with this problem.
3. Compute Restrictions: Training of a Deep CNN architecture to get to the final image encoding takes significant computation time and resources. Since we were dealing with both time constraint and lack of GPU compute, we built a custom CNN architecture trained on CPUs that balanced between compute requirements and accuracy attained.