Deep Metric Learning for Ancient Coin Identification

Jackson Greer

Faculty Advisors: Dr. Sprague and Dr. Forsyth

4/25/2025

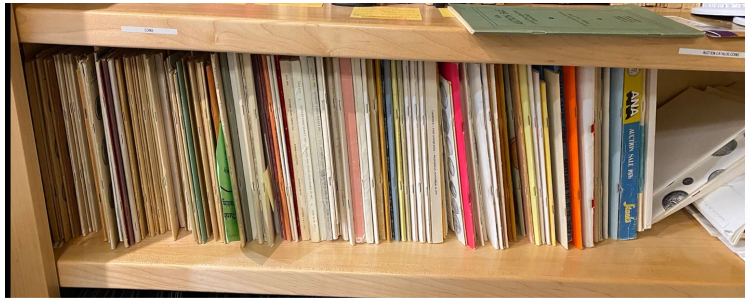
JMU JAMES MADISON UNIVERSITY.

The Problem: Provenance

- Establishing a history of prior ownership for a coin is difficult
- Important to establish a paper trail of legality that proves a coin was obtained from its country legally.
- It requires a professional to search through catalogs and build a auction record
- This type of time commitment is only made for valuable or unique objects
- What if we could automatically match a coin to its scan in an auction catalog?

The Sawhill Collection at JMU

- Collection of 450 Ancient Roman and Greek coins donated by Drs. John and Bessie Sawhill in 1976.
- No paper trail establishes where coins were purchased. But we do have his catalogs and notes.
- Can ML save time by matching coins to catalogs?



(a) Tetradrachm of Alexander the Great from 323 BCE (b) Athenian Tetradrachm from 449 - 413 BCE

(c) Gold Aureus of Julius Caesar from 46 BCE (d) Titus Sestertius Judea Capta from 80 CE

Identification Problem: Which Coins (if any) are an Exact Match?

Fig. Reverse-motif variability—symbol (red), object (blue), legend (brown). Source: Anwar et al., *Deep Ancient Roman Republican Coin Classification via Feature Fusion & Attention* (Pattern Recognition 2021).

Previous Research

- Previous research involving ancient coins is interested in motif or portrait class, with limited work in exact coin identification.
- We are interested in instance-level retrieval: given a coin photo, we locate its exact match in a reference catalog

The Approach: Metric Learning or Classification?

- Metric learning focuses on directly learning **embeddings**, compact sets of numbers that capture the key features of the data while classification focuses on assigning inputs to predefined class labels.
- It can generalize to new/unseen classes using learned distances; classification is typically restricted to known classes seen during training.
- Metric learning approaches have been successful in face recognition, we are building on that work

Goal: Generalize

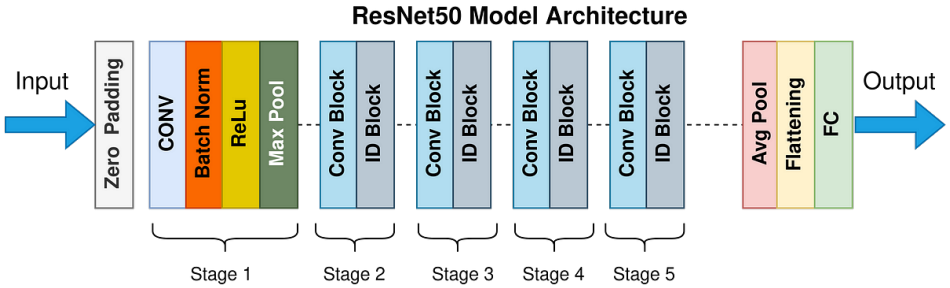
- Face identification and coin identification have shared goals
- We want to enforce a certain distance where we can rule out a new sample belonging to a class
- In order to train a model to do this, you need many photos of the same face (data points belonging to the same class)
- How do we do this in the domain of ancient coins? There aren’t large datasets of various images of the same coin.

Making the dataset with image augmentations.

- We have access to a large collection of images of Roman Republican Coins (29,531 from RRCD & RRC-60)
- No more than one image of the exact same coin, categorized by coin **type**, but we want to produce more images of the same class.
- We use data augmentation techniques to generate modified versions of our existing coin images

Model Architecture

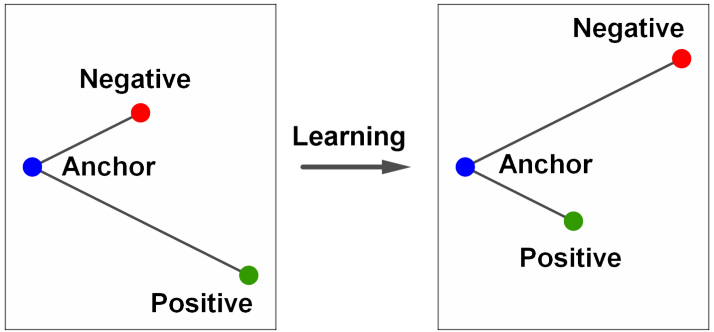
- We use ResNet50, a pre trained deep convolutional neural network (CNN)
- We input the images of coins to produce the embeddings



Creating Test Data

- How do we assess the performance of the model if we don't have any real life examples?
- Capture images from Sawhill Collection at JMU. 450 Ancient Roman and Greek coins donated by Drs. John and Bessie Sawhill in 1976
- Nine photos per coin: three different lighting variations and three different camera angles.

Multiple photos of Silver Denarius of Julius Caesar (R-11)



Triplet Loss

- A machine learning loss function that takes a triplet of training points
- Each triplet contains three data points: an anchor, a positive (an augmented image of the same coin), and a negative (similar coin of a different class)
- Triplets are found using Triplet "mining", which focuses on the smart selection of triplets for optimization
- Finding good triplets can be costly

Triplet Loss Formula

$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

Where

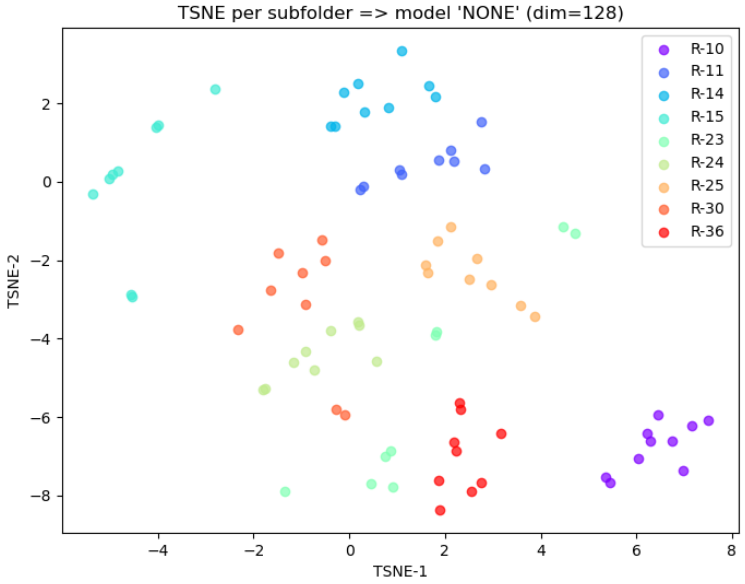
- *f(x)* accepts an input of *x* and generates a 128-dimensional vector *w*
- *i* represents the *i*'th input
- The superscript *a* denotes an anchor image, *p* is a positive image, and *n* is a negative image
- **α** refers to the margin

We replace with cosine distance

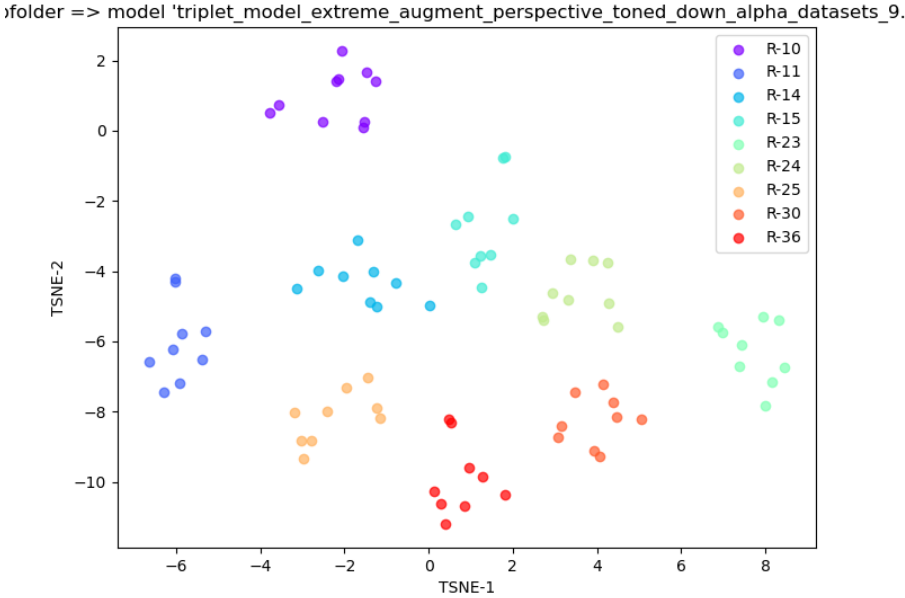
$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sum_{i=1}^{n} B_i^2}.$$

distance = 1 - similarity(A, B)

Triplet Loss Results On Test Data

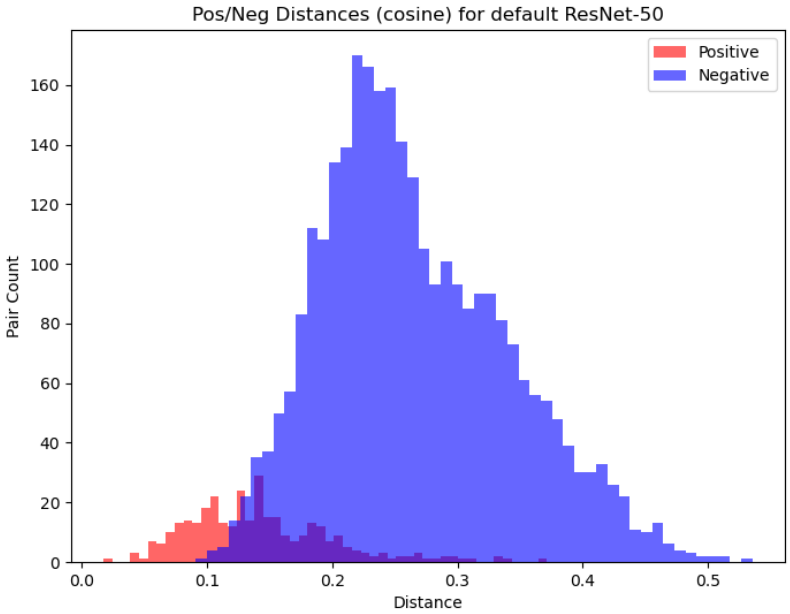


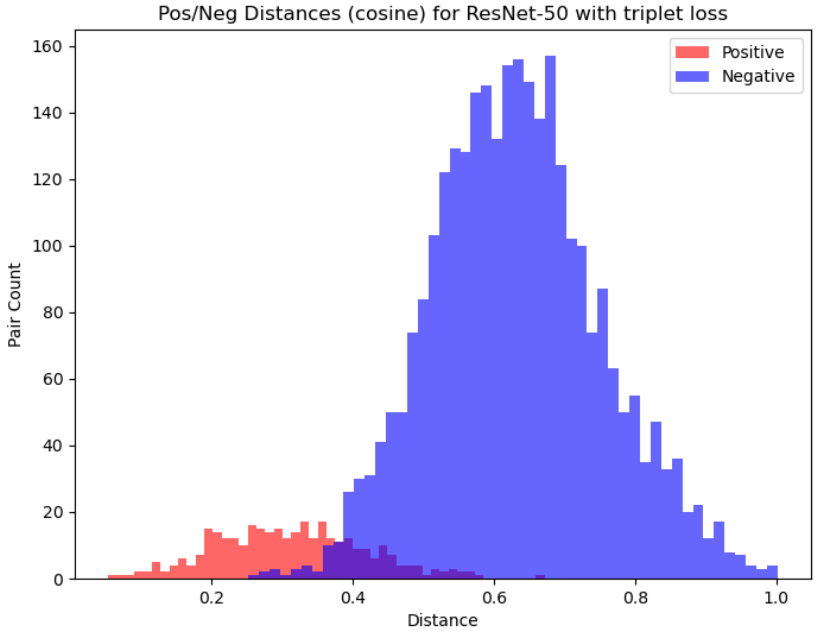
Before



After

Triplet Loss Results On Test Data





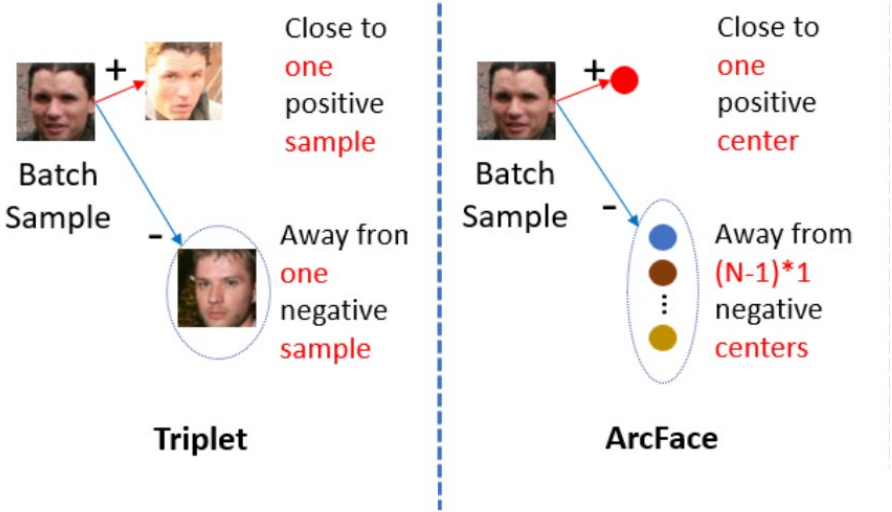


Fig. Comparison of Triplet and ArcFace loss. Source: Deng et al., *ArcFace: Additive Angular Margin Loss for Deep Face Recognition* (Pattern Recognition 2018).

Additive Angular Margin Loss (ArcFace)

- Alternative to Triplet Loss, biggest upside: no triplet mining!
- Enforces a fixed angular margin between classes on a hypersphere
- Each embedding must sit at least that angle away from different-class centers

ArcFace Loss Formula

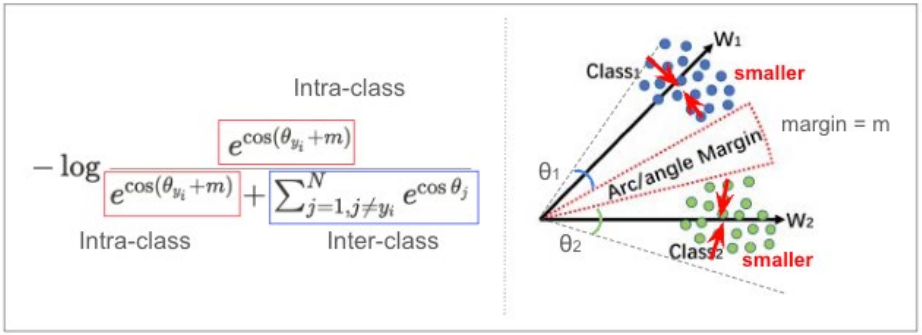


Image illustrating arc/angle margin in the arcface loss function by Yuki Shizuya in "ArcFace — Architecture and Practical example: How to calculate the face similarity between images."

Where:

- $\boldsymbol{\theta_y}$**:** the angle between the feature vector and the correct class center
- $\boldsymbol{\theta_j}$: the angle between the feature vector and other (wrong) class centers
- ***m*****:** additive angular margin
- ***s*****:** scaling factor (controls how “sharp” the decision boundary is)
- ***y*****:** the ground truth class index

The whole thing sits inside a softmax function, which turns logits into probabilities

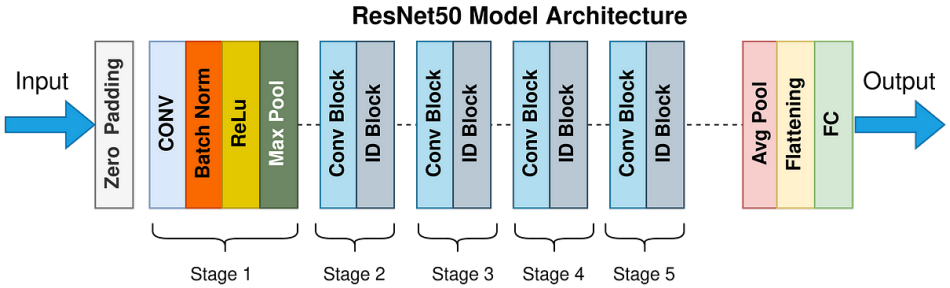
ArcFace Nearest Neighbors

Before

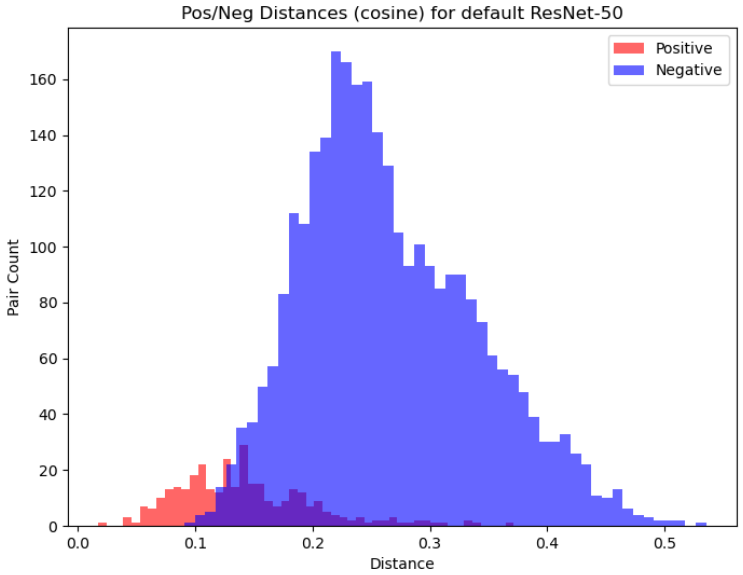
After

Model Architecture

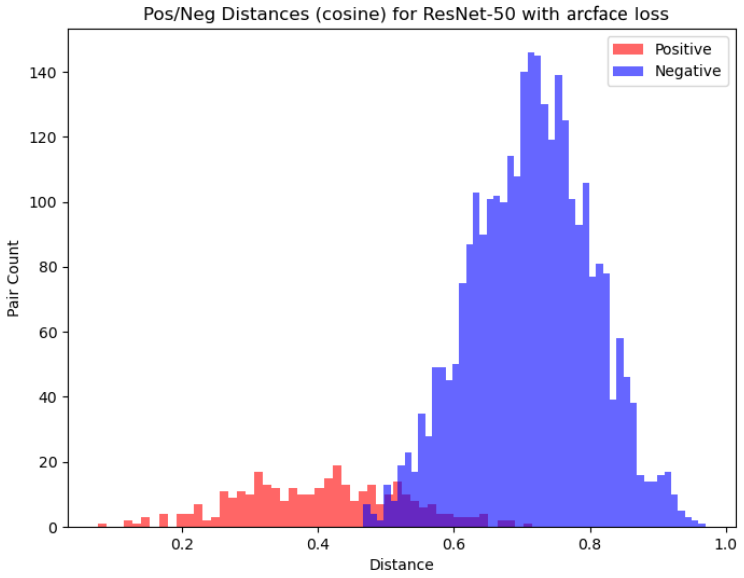
- We use ResNet50, a pre trained deep convolutional neural network (CNN)
- We input the images of coins to produce the embeddings



ArcFace Results On Test Data

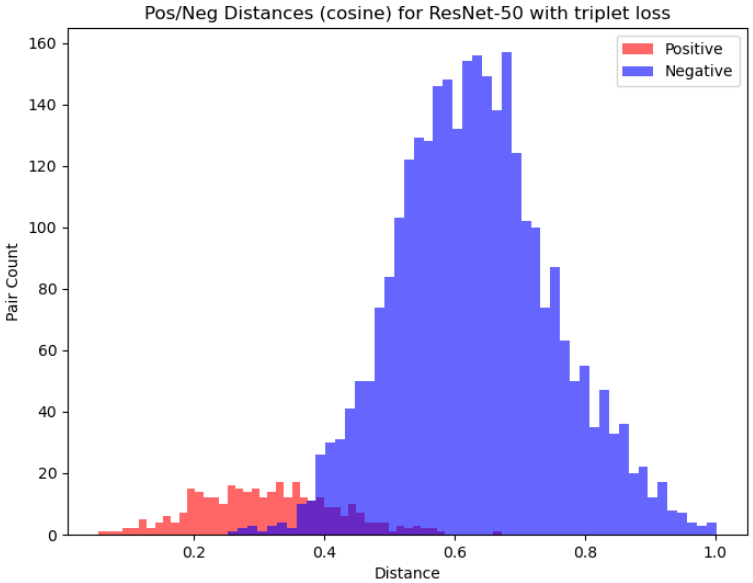


Before

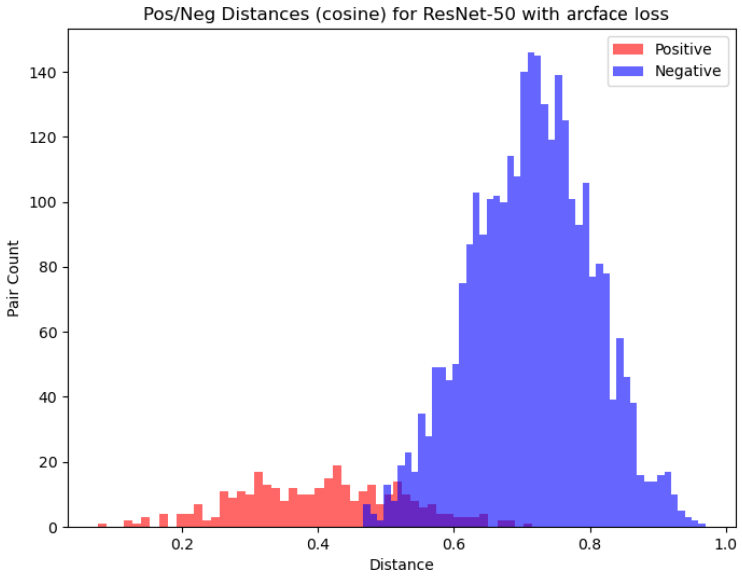


After

Triplet Loss vs ArcFace



Triplet Loss



Arcface

Lessons Learned

- Informative validation metrics like cosine similarity of unmatched and augmented pairs help us track success per epoch.
- Loading images from disk is one of the biggest slowdowns in the training, it can be helpful to load all of the data into memory first
- Resizing images before augmentation takes a lot of load off the CPU

Project Continuation

- Data preprocessing, RRCD and RRC-60 have duplicate images
  - This is partially fixed by looking for exact duplicates (same sig)
  - Issue of removing backgrounds from the images for augmentation, can be done using Segment Anything (SAM) but very slow
- Sufficient separation of the distribution of positive and negative pairs on the test data
  - More epochs, more data, better augmentations?
- Putting the model to work, quality test data with GUI
  - Each data point should have an image & detailed info on coin
- More systematic testing of real world data

Thank you!

Any questions?

JMU JAMES MADISON UNIVERSITY.