FaceNet: A Unified Embedding for Face Recognition and Clustering

- [[📚 1503.03832.pdf]]
- **Basic Idea**:
  - To be able to directly train a DCNN to optimise the embedding itself, rather than an intermediate bottleneck layer (directly optimises the loss for the task at hand i.e., face verification).
  - The networks without 2D or 3D alignment of face images (only scale and translation is fixed).
  - 🟡 **P3** Namely, we strive for an embedding f (x), from an image x into a feature space Rd, such that the squared distance between all faces, independent of imaging conditions, of the same identity is small, whereas the squared distance between a pair of face images from different identities is large.
- **Different problems related to Faces in deep learning**:
  - *Face verification*: Given two images, is it the same person.
  - *Face recognition*: Who is this person.
  - *Face clustering*: Find common people among these faces.
  - 🟡 **P1** Once this embedding has been produced, then the aforementioned tasks become straight-forward: face verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be achieved using off-theshelf techniques such as k-means or agglomerative clustering.
- **Triplet Loss**
  - Notation: $f(x) \in \mathbb{R}^d$ embedding of an image x into a d-dimensional space.
  - Constrain the embedding to live on the d-dimensional hypersphere, i.e., $||f(x)||_2 = 1$.
  - Motivation: We want to ensure that an image x_i^a (anchor) of a specific person is closer to all other images x_p^i (positive) of the same person than it is to any image x_n^i (negative) of any other person. Thus we want,

$$||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2$$
$$\forall f(x_i^a), f(x_i^p), f(x_i^n) \in \mathcal{T}$$

    where α is a margin that is enforced between positive and negative pairs. $\mathcal{T}$ is the set of all possible triplets in the training set and has cardinality N.
  - The loss to minimise is:

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

    where $[x]_+ = max\{0, x\}$.
  - **Utility of margin**:
    - At least, α is the distance between positive and negative examples (ensuring discriminability rather than just separability). The magnitude of α controls the amount of discriminative power of the embedding.
    - The role of the hinge function $[m + \cdot]_+$ is to avoid correcting "already correct" triplet because the triplets which already satisfy will have loss L = 0. But for discriminability we also want the embeddings from the same class to be pulled close to each other, so this L should not be that easily satisfied i.e. m should have sufficiently high value.
- **Triplet Selection**
  - 🟡 **P3** Generating all possible triplets would result in many triplets that are easily satisfied (i.e. fulfill the constraint in Eq. (1)). These triplets would not contribute to the training and result in slower convergence, as they would still be passed through the network. It is crucial to select hard triplets, that are active and can therefore contribute to improving the model.
  - This means that given x_i^a, we want to select an x_i^p (hard positive) such that $argmax_{x_i^p}||f(x_i^a) - f(x_i^p)||_2^2$ and similarly x_i^n (hard negative) such that $argmin_{x_i^n}||f(x_i^a) - f(x_i^n)||_2^2$.

- 🟡 **P3** It is infeasible to compute the argmin and argmax across the whole training set. Additionally, it might lead to poor training, as mislabelled and poorly imaged faces would dominate the hard positives and negatives. There are two obvious choices that avoid this issue:
  - 🟡 **P3** Generate triplets offline every n steps, using the most recent network checkpoint and computing the argmin and argmax on a subset of the data.
  - 🟡 **P3** Generate triplets online. This can be done by selecting the hard positive/negative exemplars from within a mini-batch.
- Use online triplet generation. This requires two things:
  - Large batch-size so that argmin and argmax that are calculated are meaningful. Notice the trade-off: Smaller batch-sizes are required for faster convergence whereas larger batch-sizes required for contrastive training.
  - Minimal number of exemplars of one identity is present in each mini-batch so as to have meaningful representation of the anchor-positive distances (in this paper, they choose 40 images per identity per mini-batch and also add randomly sampled negative samples).
- They use all anchor-positive pairs in a mini-batch for loss calculation instead of just using hardest positive examples.
- Selecting the hardest negatives can in practice lead to bad local minima early on in training, specifically it can result in a collapsed model (i.e. $f(x) = 0$). In order to mitigate this, it helps to select x_i^n such that

$$||f(x_i^a) - f(x_i^p)||_2^2 < ||f(x_i^a) - f(x_i^n)||_2^2$$

  These negative examples are called semi-hard because they are further away from the anchor than the the positive exemplar, but still hard because the squared distance is close to the anchor-positive distance. Ideally we would want no negative exemplar to be within α margin of the anchor. So, basically these semi-hard negative exemplars are all those negative exemplars which lie inside the margin α (for all other negative exemplar which satisfy the above condition but lie outside the margin will have loss L=0).
- **Implementation**:
  - batch_size: 1800 exemplars
  - α : 0.2
- 🟡 **P3**

Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

- **Harmonic Embedding**
  - Set of embeddings which are generated by different models but are still compatible with each other.
  - **Use-case**: 🟡 **P9** This compatibility greatly simplifies upgrade paths. E.g. in an scenario where embedding v1 was computed across a large set of images and a new embedding model v2 is being rolled out, this compatibility ensures a smooth transition without the need to worry about version incompatibilities.
  - **Harmonic Triplet Loss**:

- 🟡 P9

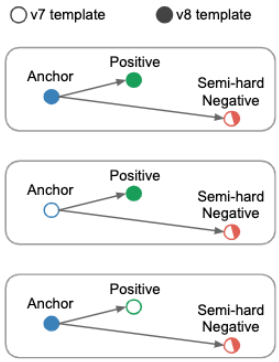

Figure 9. **Learning the Harmonic Embedding.** In order to learn a *harmonic* embedding, we generate triplets that mix the v1 embeddings with the v2 embeddings that are being trained. The semi-hard negatives are selected from the whole set of both v1 and v2 embeddings.

- 🟡 P10

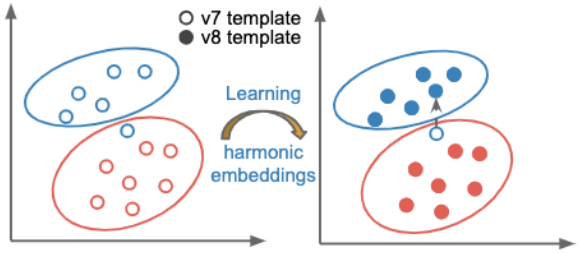

Figure 10. **Harmonic Embedding Space.** This visualisation sketches a possible interpretation of how *harmonic* embeddings are able to improve verification accuracy while maintaining compatibility to less accurate embeddings. In this scenario there is one misclassified face, whose embedding is perturbed to the "correct" location in v2.

- **Extensions: Deep Face Recognition by Parkhi et al. 2015**
  - Able to achieve the same performance as FaceNet but with considerably less data and less number of unique identities.
  - Instead of directly training the network with the triplet-loss, they first train the network using a classifier and then throw away the last layer. The training with a triplet-loss learns a new projection head whose output will be considered the embedding for the face.
  - **Sampling of triplets**: This is similar to the FaceNet (but more verbose). An epoch here contains all the possible positive pairs (a, p), where image a is considered the anchor and p its paired positive example. Choosing good triplets is crucial and should strike a balance between selecting informative (i.e. challenging) examples and swamping training with examples that are too hard. This is achieved by extending each pair $(a.p)$ to a triplet (a, p, n) by sampling the image n at random, but only between the ones that violate the triplet loss margin. The latter is a form of hard-negative mining, but it is not as aggressive as (and much cheaper than) choosing the maximally violating example, as often done in structured output learning.
  - **Tricks of the trade**: The input to all networks is a face image of size 224 × 224 with the average face image (computed from the training set) subtracted - this is critical for the stability of the optimisation algorithm. Reason: If we are using ReLU activation, we want some portions of the inputs to be negative as well. Otherwise, there will be no non-linearity.
  - **Metrics**: In addition to the verification accuracy Acc., they use the *Equal Error Rate (EER)* as an evaluation metric, defined as the error rate at the ROC operating point where the false positive and false negative rates are equal. The advantage on Acc. is that it is (i.e., EER) independent of the distance threshold τ. This is because there will be mostly one threshold where the false negatives and false positives rates would the same, whereas the accuracy is highly dependent on the threshold selection.

Unlinked References