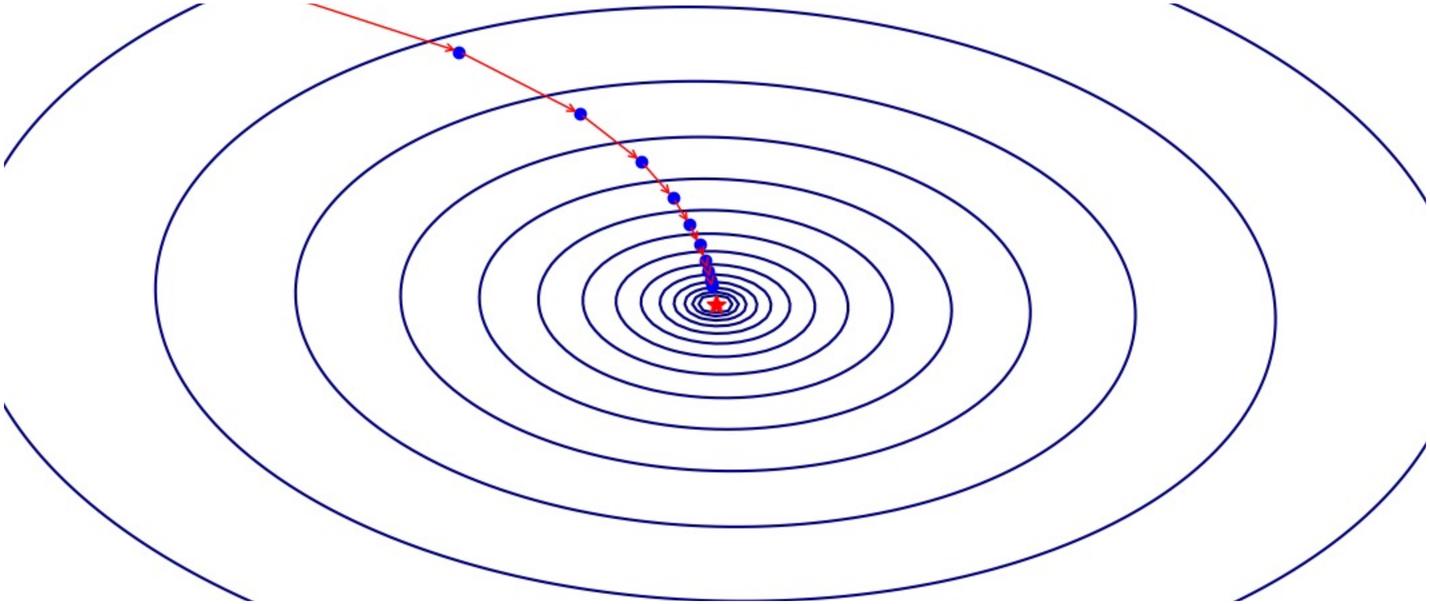


Advanced Topics in Deep Learning

Summer Semester 2024

5. Similarity Learning

06.05.2024

Prof. Dr. Vasileios Belagiannis

Chair of Multimedia Communications and Signal Processing

Friedrich-Alexander-Universität
Technische Fakultät

Course Topics

1. Interpretability.
2. Attention and Transformers.
3. Self-supervised Learning.
4. **Similarity Learning.**
5. Generative Models.
6. Model Compression.
7. Transfer learning, domain adaptation, few-shot learning.
8. Uncertainty Estimation.
9. Geometric Deep Learning.
10. Recap and Q&A.

- The exam will be written.
- We will have an exam preparation test.

Acknowledgements

- Special thanks Arij Bouazizi, Julia Hornauer, Julian Wiederer, Adrian Holzbock and Youssef Dawoud for contributing to the lecture preparation.

Recap

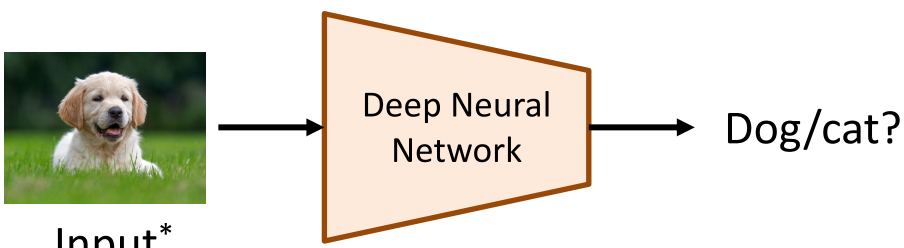
- Generative modelling.
- Contrastive learning loss functions.
- Self-supervised contrastive learning.
- Negative-free contrastive learning.
- Momentum contrastive learning.

Today's Agenda and Objectives

- Similarity learning
- Deep metric learning
- Triplet loss
- Quadruplet loss
- Hierarchical Triplet loss

Supervised Learning – Classification

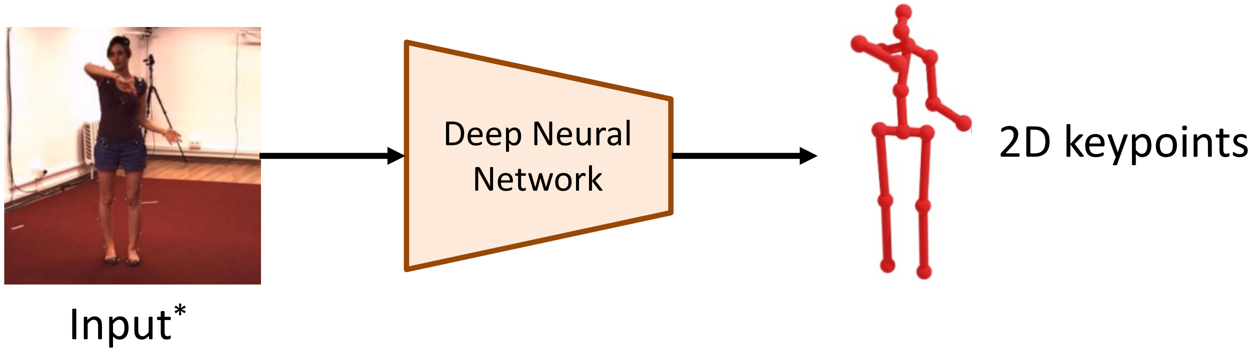
- The approximation function $f: R^n \rightarrow 1, \dots, K$ learns to map the input sample to K categories, represented by a set of n-dimensional features, to a categorical output.
- The numeric code $y = f(x)$ is the category prediction for the input x where the output can be probabilistic.
- We normally rely on the cross-entropy loss and binary cross-entropy to define the loss function.
- The loss functions <u>directly</u> affects the learn feature representation.



Input*

Supervised Learning – Regression

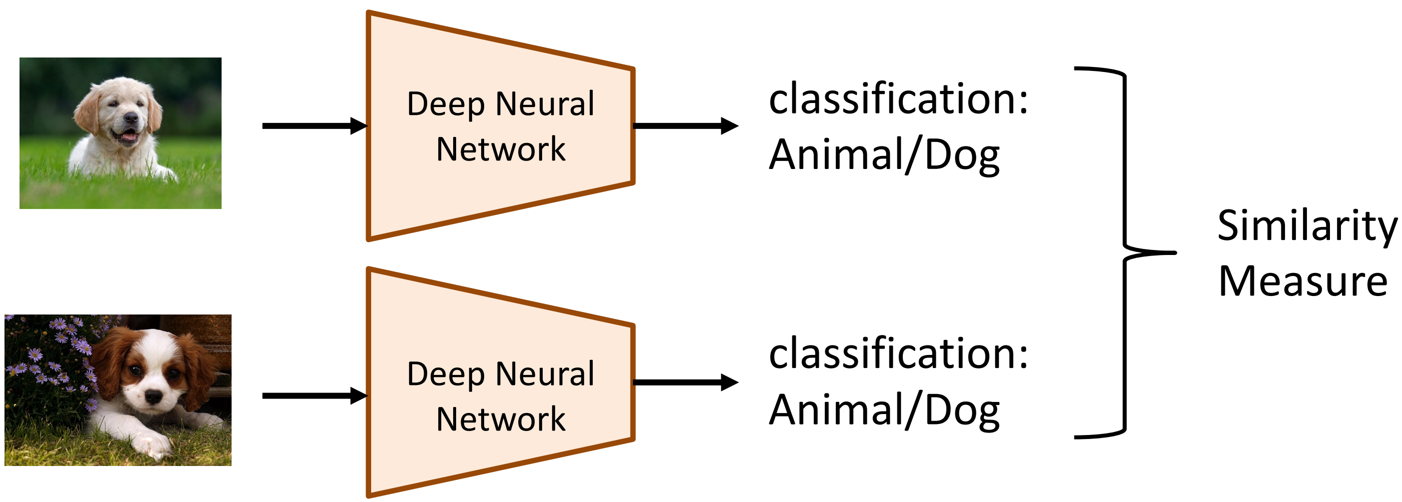
- For regression, the mapping $f: R^n \rightarrow R^n$ learns to predict numerical value(s).
- The common loss function is the mean-squared error.
- Similar to classification, the learned feature representation is based on the loss function.



*Ionescu, Catalin, et al. "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments." IEEE transactions on pattern analysis and machine intelligence 36.7 (2013): 1325-1339.

Similarity Learning

- Similarity learning involves modelling the similarities between two objects (e.g. people, animals). Our focus is on <u>image</u> data.
- Given a similarity function, images with similar context are projected into the same neighbourhood in the latent space. Images with different contexts are far apart in the latent space.
- It is also known as <u>distance metric learning</u>.



Similarity Learning (Cont.)

- Deep neural networks are used to learn feature embeddings that fit a distance (or similarity) function.
- For instance, the Euclidean distance can be used as loss function for deep metric learning.
- The learned representation can be used to determine whether two input samples belong to the same class or not.

Caltech-UCSD Bird Species

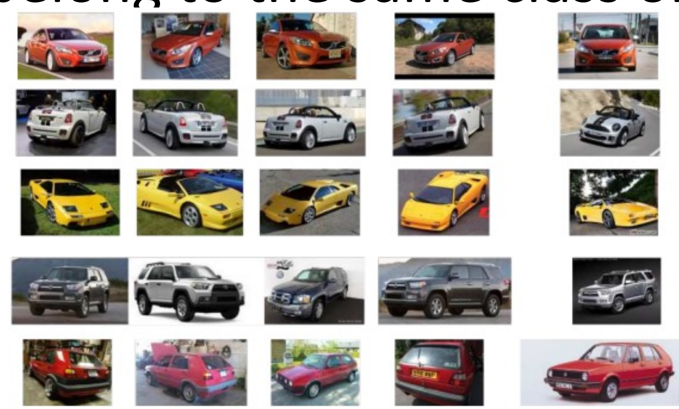


Image Retrieval (Image from [1]

[1] Sanakoyeu, A., Tschernezki, V., Buchler, U., & Ommer, B. (2019). Divide and conquer the embedding space for metric learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 471-480).

Deep Metric Learning

- Given the training set $\mathcal{X}$ and the corresponding label set $\mathcal{Y}$, the goal is to train a deep neural network, parametrized by θ, producing the latent representation $f_\theta: \mathcal{X} \rightarrow \mathbb{R}^n$ of n dimensions, based on the distance function $D: \mathbb{R}^n \rightarrow \mathbb{R}$.
- For the input samples $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ with labels $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$, the distance $D(f_\theta(\boldsymbol{x}_1), f_\theta(\boldsymbol{x}_2))$ has small value if $\boldsymbol{y}_1 = \boldsymbol{y}_2$ and large value otherwise.
- The selection of the <u>distance function</u> can be done in advanced and keep it fixed during training, e.g. Euclidean distance.
- The neural network <u>architecture</u> plays an important role to the embedding learning.
- The idea is similar to <u>contrastive</u> learning.

Distance Function

- *How do we choose the distance function?*
- A metric, or distance function, is a mathematical function used to calculate the distance between a pair of elements in a set.
- There are two main types of metrics:
  - Pre-defined metrics.
  - Learned metrics.
- Predefined metrics: Metrics which are can be defined without the knowledge of data.
  - Example: Euclidian Distance: $f(x - y) = (x - y)^T (x - y)$.
- Learned Metrics: Metrics which can only be specified with the knowledge of the data.
  - E.g., Mahalanobis Distance: $f(x - y) = (x - y)^T M (x - y)$, with M being a matrix estimated from the data.

Distance Function (Cont.)

- The similarity measure is a task-relevant metric. Some common distance functions are:
  - Numerical data: Euclidean distance, cosine similarity.
  - Categorical data: Hamming distance, Jaccard distance.
- There are different loss functions to learn a metric. The most common are:
  - Triplet loss.
  - Quadruplet loss.
  - Hierarchical triplet loss.
- The triplet loss is the <u>most</u> popular function for metric learning.

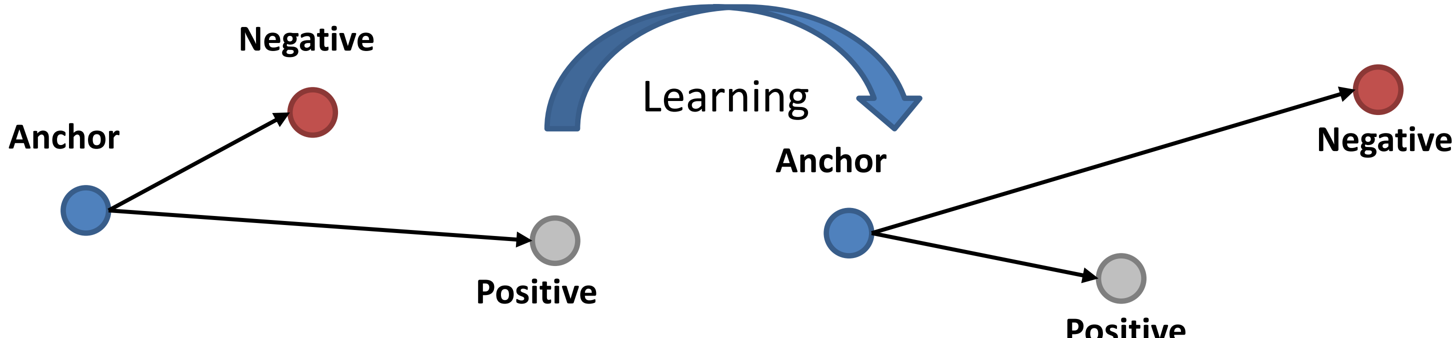
Triplet Loss

- Let the triplet x_a, x_p, x_n be input samples from a given dataset and y_a, y_p, y_n are their corresponding labels, so that $y_a = y_p$ and $y_a \neq y_n$.
- For example, x_a, x_p are image of the same person and x_n the image of another person.
- x_a is called **anchor** sample.
- x_p is called **positive** sample.
- x_n is called **negative** sample.
- The goal of the triplet loss is to learn the embedding space where the embedding of x_a and x_p are closer than the x_a and x_n.
- For the face example, we aim to learn embeddings (latent representations) for the faces of our database.

*Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." CVPR2015.

Triplet Loss (Cont.)

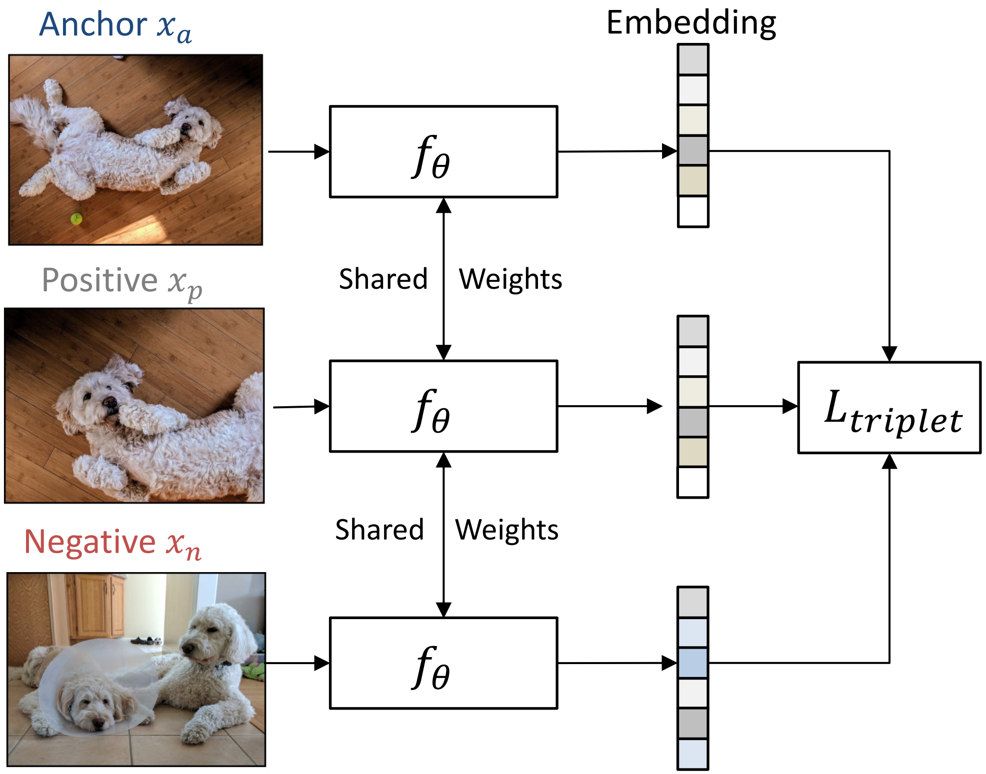
- Overall* the distance of the positive sample to the anchor is minimised, while the the distance of the negative sample to the anchor is maximized.
- The loss function is defined as:
  - $\mathcal{L}_{triplet} = \max\left(0, D\big(f_\theta(x_a), f_\theta(x_p)\big) - D(f_\theta(x_a), f_\theta(x_n)) + \alpha\right)$.
  - The distance function is the Euclidean distance $D = ||p - q||_2^2$.
  - f_θ represents a deep neural network that extract features.
  - α is a margin (hyper-parameter) that is enforced between positive and negative pairs.



*Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." CVPR2015.

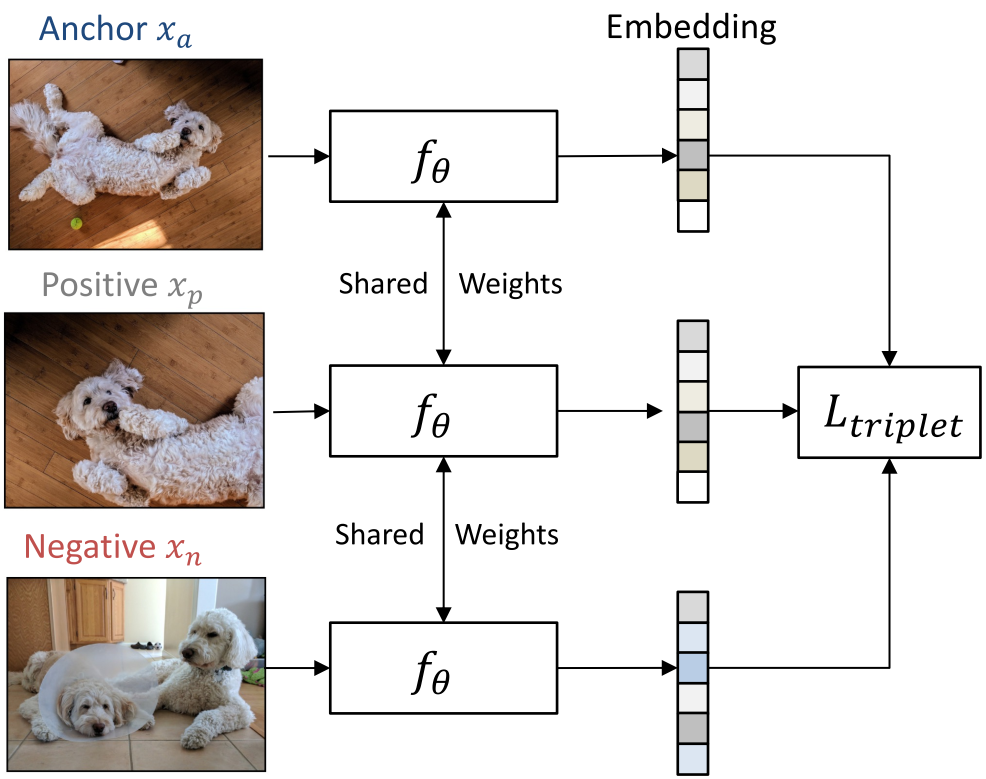
Triplet Loss (Cont.)

- The anchor x_a corresponds to the dog.
- The positive x_p corresponds to the same dog from another view.
- The negative x_p corresponds to some different dog (s).
- Note that we chose a <u>difficult</u> negative here.
- Note that $f_\theta(\cdot)$ is <u>normalized</u> so that the distance function returns results in the range of $[0,1]$.



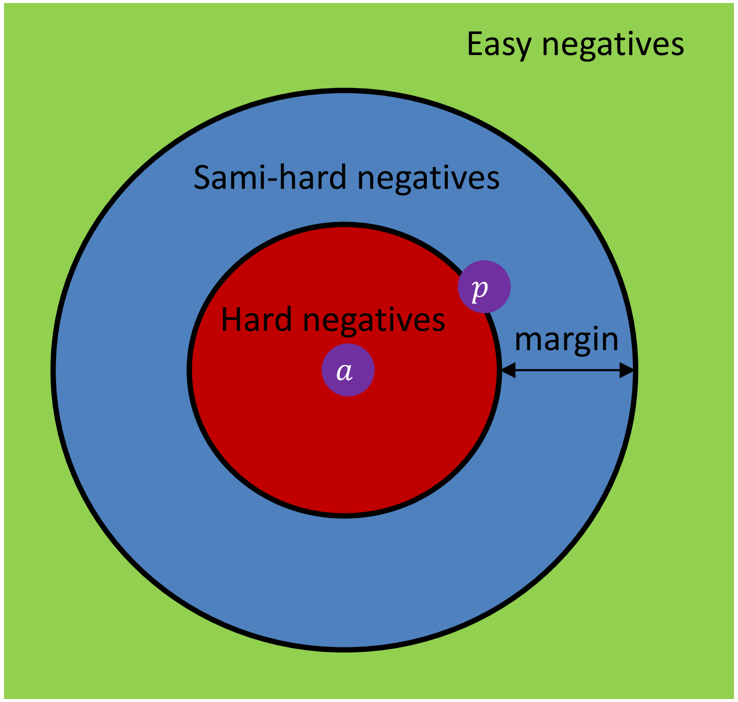
Triplet Loss (Cont.)

- If $D\left(f_\theta(x_a), f_\theta(x_p)\right)$ tends to be 0 & $D(f_\theta(x_a), f_\theta(x_n)) > D\left(f_\theta(x_a), f_\theta(x_p)\right) + \alpha$, then x_n becomes an easy negative.
- An easy example may not be useful for creating meaningful, i.e., large, gradients.
- Learning a robust representation is clearly based on the selection of the positive and negative samples.
- *How do we select representative positive and negative samples?*



Negative triplets

- We can define three types of negatives.
  - Easy Triplets: have a loss of 0 because $D\big(f_\theta(x_a), f_\theta(x_p)\big) + \alpha < D(f_\theta(x_a), f_\theta(x_n))$.
  - Hard Triplets: x_n is closer x_a than x_p, i.e. $D(f_\theta(x_a), f_\theta(x_n)) < D\big(f_\theta(x_a), f_\theta(x_p)\big)$.
  - Semi-hard Triplets: x_n is not closer x_a than x_p, but which still have positive loss $D\big(f_\theta(x_a), f_\theta(x_p)\big) < D(f_\theta(x_a), f_\theta(x_n)) + \alpha$.
- A standard approach is to select a random semi-hard negative for every pair of anchor and positive samples.



Source: https://omoindrot.github.io/triplet-loss

Triplet Mining

- The triplet formation can be performed either online or offline.
- Offline triplets:
  - This approach involves creating the triplets before starting the neural network epoch training.
  - The embeddings are extracted from f_θ for all training samples and then triplets are sampled.
  - Only the hard or semi-hard triplets are selected for creating mini-batches.
  - For all mini-batches, back-propagation and gradient descent are used to update the deep neural network f_θ.
  - This is a computationally demanding approach since we make use of the full dataset to look for hard or semi-hard triplets. In addition, each mini-batch formation is more expensive than standard supervised learning because of the triplet computations (3 times more computations).

Triplet Mining (Cont.)

- Online triplets:
  - We can load samples on the mini-batch and compute the triplets on the fly during training.
  - All triplets will not be necessarily valid. We need 2 positives and 1 negative per triplet.
  - Online triplets are easy to implement and make training faster.
  - The embeddings, gradients and parameter update is performed on the same way with the offline triplets.

Triplet Loss Challenges

- The number of possible triplets is $O\ (n^3)$.
  - Training with the triplet loss can cost significant time to train and can lead to inferior performance.
  - There will be multiple correlated samples.
  - Overall, the training is highly redundant and less informative because of the random sampling.
- Because of the challenges, the optimisation can easily get stuck a some local minimum.
- There are haven proposed related loss functions to offer faster convergence:
  - Quadruplet loss.
  - Hierarchical triplet loss.

Quadruplet Loss

- The quadruplet* Loss replaces the Euclidean distance with a learned metric.
- The learned <u>distance</u> metric function $g(x_i, x_j)$ can model more relationships between the samples instead of a fixed metric.
- The triplet loss can be now reformulated as:
  - $\mathcal{L}_{triplet} = \max\left(0, g\big(f_\theta(x_a), f_\theta(x_p)\big)^2 - g(f_\theta(x_a), f_\theta(x_n))^2 + \alpha\right).$
  - $g(x_i, x_j)$ represents the distance between two samples, i.e., high values for the functions correspond to dissimilar samples.
- The metric $g(x_i, x_j)$ can be formulated as a fully-connected layer followed by a two-dimensional output (class-1: dissimilarity, class-2: similarity).
  - One of these outputs can express the dissimilarity probability of the two samples. We use it as input to the triplet loss above.

*Chen, Weihua, et al. "Beyond triplet loss: a deep quadruplet network for person re-identification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Quadruplet Loss (Cont.)

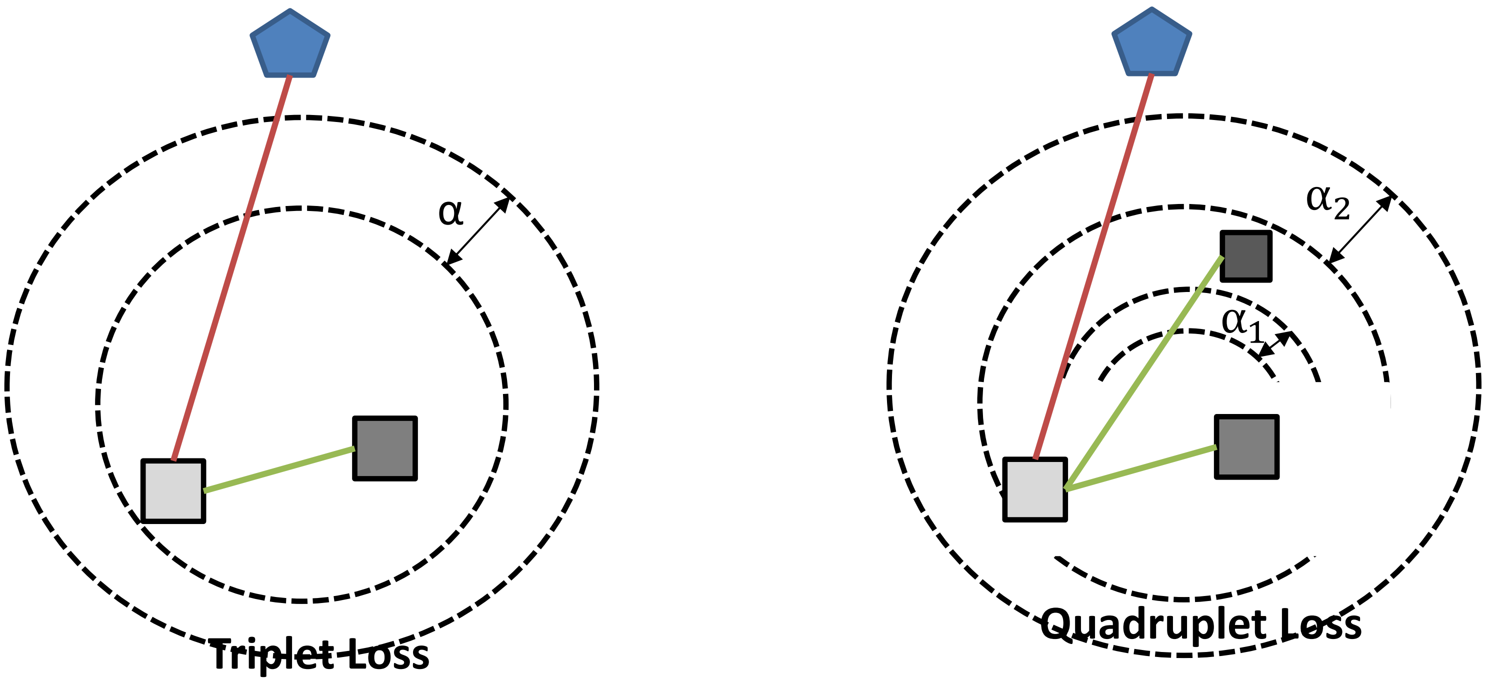
- The triplet loss is based <u>only</u> on the <u>relative</u> distances between positive and negative pairs with respect to the anchor sample.
- The quadruplet loss imposes additional <u>constraints</u> by considering additionally a pair that is formed only from negative samples.
  - Note that the second negative pair does not contain the <u>anchor</u> image, instead only two negative samples.
- Unlike the triplet loss, which does not consider the class variation of the features $f_\theta(\cdot)$, the quadruplet loss is designed to encourage greater <u>interclass</u> variation, while encouraging smaller <u>intraclass</u> variation.

Quadruplet Loss (Cont.)

- Let x_a, x_p, x_n, x_s be the samples and $y_a = y_p, y_a \neq y_n, y_a \neq y_s, y_s \neq y_n$their corresponding labels.
- x_a is the anchor sample.
- x_p is the positive sample.
- x_n is the negative sample.
- x_s is another negative sample.
- The quadruplet loss is computed as:
  - $\mathcal{L}_{quadruplet} = \max\left(0, g\left(f_\theta(x_a), f_\theta(x_p)\right)^2 - g(f_\theta(x_a), f_\theta(x_n))^2 + \alpha_1\right) + \max\left(0, g\left(f_\theta(x_a), f_\theta(x_p)\right)^2 - g(f_\theta(x_s), f_\theta(x_n))^2 + \alpha_2\right).$
  - α_1, α_2 are two different margins.
  - $\mathcal{L}_{quadruplet} = \mathcal{L}_{triplet} + \max\left(0, g\left(f_\theta(x_a), f_\theta(x_p)\right)^2 - g(f_\theta(x_s), f_\theta(x_n))^2 + \alpha_2\right).$

Quadruplet Loss (Cont.)

- The first term of the loss is the triplet loss.
- The second term considers the minimum inter-class distance to be larger than the maximum intra-class distance regardless of the anchor sample.



Quadruplet Loss Margins

- The second terms is <u>auxiliary</u> and thus it should not have the same impact as the first one.
- To make the impact <u>smoother</u> α_1, α_2 are two margin thresholds that have the same impact with using two different weights.
- The first term should have a <u>large</u> margin $\alpha_1$because the anchor is used on both pairs.
- The second term can have a <u>small</u> margin α_2 because it is auxiliary term.

Quadruplet Loss Margins (Cont.)

- The margin value is important because it affects the number of hard samples to be used for training.
- However, the margin threshold cannot be too high, as this will produce too many hard samples and cause overfitting.
- The <u>adaptive margin threshold</u> can help for selecting the hard samples.
- The quadruplet loss computes the margin as the average distance of the positive and negative sample distributions. This can be written as:
  - $a_{1,2} = w(\mu_n - \mu_p) = w\left(\frac{1}{N_n}\sum_{a,n}^{N} g(f_\theta(x_a), f_\theta(x_n))^2 - \frac{1}{N_p}\sum_{a,p}^{N} g\big(f_\theta(x_a), f_\theta(x_p)\big)^2\right).$
  - μ_n, μ_p are the mean values of the positive and negative sample distributions.
  - N_n, N_p are the number of negative and positive pairs.
  - The weight w is set to 1 for a_1 and 0.5 for a_2.
  - The computations are performed on the mini-batch level.

Person Re-Identification

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- The quadruplet loss was originally proposed for person re-Identification.

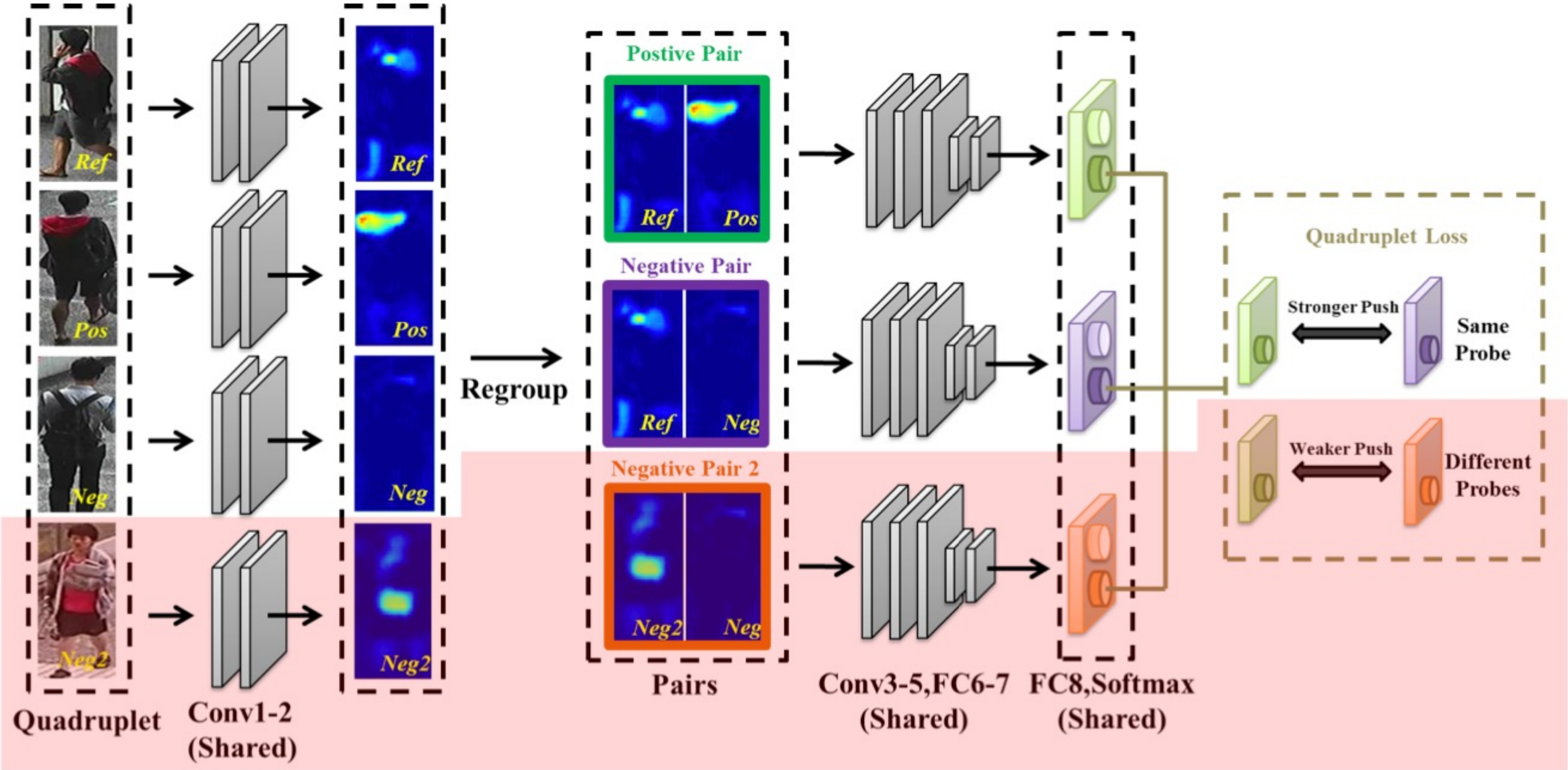


Image Source: https://arxiv.org/pdf/1704.01719.pdf

Hierarchical Triplet Loss

- The hierarchical triplet (HTL) loss is a variation* where the training set is organized into a hierarchy of classes.
- Each class is associated with a set of child classes that are more specific than the parent class.

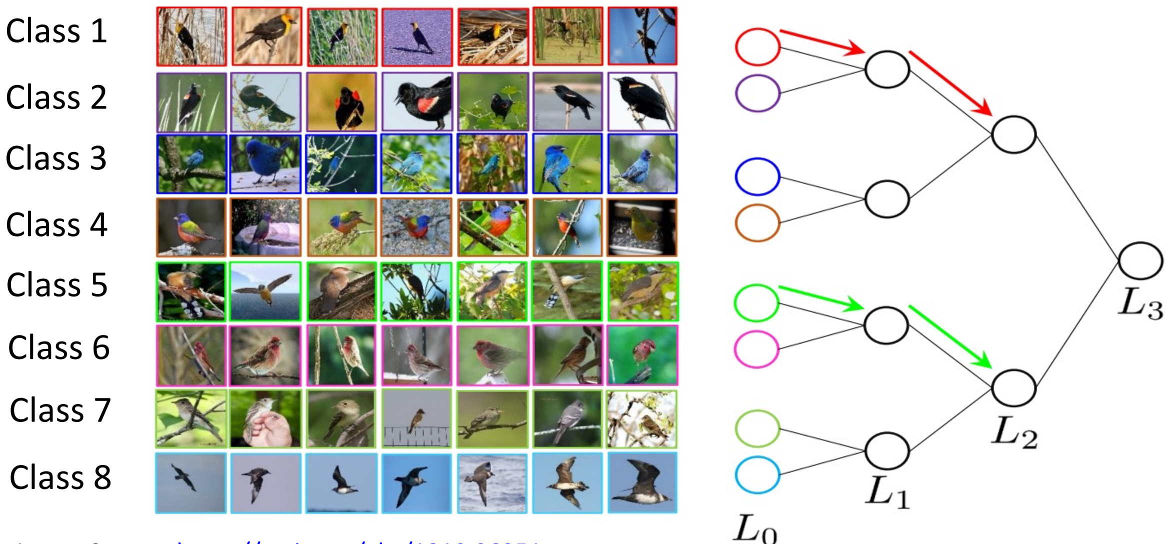


Image Source: https://arxiv.org/abs/1810.06951

*Ge, W. (2018). Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 269-285).

Hierarchical Tree

- Consider the deep neural network f_θ that is pre-trained with the standard triplet loss $\mathcal{L}_{triplet}$.
- Prior to constructing the hierarchy of classes as a tree, the distance matrix of classes $\mathcal{H}$ is calculated as:
  - $d(p,q) = \frac{1}{n_p n_q} \sum_{i \in p, j \in q} \left\| f_\theta(x_i) - f_\theta(x_j) \right\|^2.$
  - n_p defines the cardinality of class p and n_p defines the cardinality of class q.
- Based on $\mathcal{H}$ a hierarchy tree is then constructed.
  - The tree leaves are as many as the number of object classes. Each class corresponds to a leaf at the level 0 (L0).
  - Using $\mathcal{H}$, the tree grows by recursively merging the leave nodes at each level.
  - The growth stops after reaching the the level L.
  - The average inner distance is used a threshold to merge the nodes at the level 0. This is defined as: $d_0 = \frac{1}{C} \sum_{i=1}^{C} \frac{1}{n_c^2 - n_c} \sum_{i \in c, j \in c} \left\| f_\theta(x_i) - f_\theta(x_j) \right\|^2$ with n_c the number of samples of the c class.
  - At the l-th level, the threshold is defined based on d_0 as $d_l = \frac{l(4-d_0)}{L}$ with L total levels.
  - Two classes with distance smaller then d_l are merged to one node.
  - Finally, we have a tree with a single top node and C leave nodes.

Hierarchical Triplet Loss

- First, we randomly select l' nodes at the $0th$ level. This ensures the class diversity in the mini-batch.
- Next, based on the distance in feature space, $m - 1$ nearest classes at the $0th$ level of the hierarchical tree are selected for each of the l' nodes.
- This is done to encourage the model to learn features that capture the difference and similarities between the visual similar classes.
- Finally, t images per class are randomly collected. The total number of images per mini-batch then is $n = l'mt$.
- Based on the collected n, the triplets are formed.

Hierarchical Triplet Loss (Cont.)

- Based on the mini-batch collected triplets, the hierarchical triplet loss can be written as:
  - $\mathcal{L}_M = \frac{1}{2Z_M} \sum_{T^z \in T^M} \left[\left\| x_a{}^z - x_p{}^z \right\| - \| x_a{}^z - x_n{}^z \| + a_z \right]$.
  - T^M are the mini-batch triplets.
  - $T^z = (x_a, x_p, x_n)$.
  - Z_M is the number of triplets.
  - a_z is the dynamic margin.
- The dynamic margin is computed as:
  - $a_z = \beta + d_{\mathcal{H}(y_a, y_n)} - s_{y_a}$.
  - β is a constants that encourages the classes to be further apart than in previous iterations.
  - $\mathcal{H}(y_a, y_n)$ indicate the tree level where the two classes are merged into a single node in the next level.
  - $d_{\mathcal{H}(y_a, y_n)}$ is the threshold for merging the two classes.
  - $s_{y_a} = \frac{1}{n_{y_a}^2 - n_{y_a}} \sum_{i,j \in y_a} \left\| f_\theta(x_i) - f_\theta(x_j) \right\|^2$ is the average distance between samples of the class y_a.
  - In this way, x_a is encouraged to push the nearby points with different semantics far from itself. In addition, it contributes to the gradients of very distant samples by calculating the dynamic margin.

Hierarchical Triplet Loss Algorithm

- The complete algorithm.

Algorithm 1: Training with hierarchical triplet loss

Input: Training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{k=1}^{N}$. Network $\phi(\cdot, \boldsymbol{\theta})$ is initialized with a pretrained ImageNet model. The hierarchical class tree $\mathcal{H}$ is built according to the features of the initialized model. The margin α_z for any pair of classes is set to 0.2 at the beginning.

Output: The learnable parameters θ of the neural network $\phi(\cdot, \boldsymbol{\theta})$.

```
while not converge do
    t ← t + 1 ;
    Sample anchors randomly and their neighborhoods according to H ;
    Compute the violate margin for different pairs of image classes by
      searching through the hierarchical tree H ;
    Compute the hierarchical triplet loss in a mini-batch L_M;
    Backpropagate the gradients produced at the loss layer and update
      the learnable parameters ;
    At each epoch, update the hierarchical tree H with current model.
```

Image Source: https://arxiv.org/abs/1810.06951

*Ge, W. (2018). Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 269-285).

Study Material

- *Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." CVPR2015.*
- Chen, Weihua, et al. "Beyond triplet loss: a deep quadruplet network for person re-identification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
- Ge, W. (2018). Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 269-285).
- [*https://omoindrot.github.io/triplet-loss*](https://omoindrot.github.io/triplet-loss)

Next Lecture

Generative Models