HELLENIC REPUBLIC
National and Kapodistrian
University of Athens
EST. 1837

Metric Learning: A Deep Dive

Master's Thesis Presentation

Bill Psomas
Supervisor: Yannis Avrithis

22 October 2020

01
INTRODUCTION
Definition, Motivation, Related Work

02
BACKGROUND
Metric Learning, Neural Networks, Deep Metric Learning

03
EXPERIMENTAL SETUP
Datasets, Networks, Evaluation, Implementation Details, Issues

04
EXPERIMENTAL RESULTS
Results, Discussion

05
OUR SETUP
Cross Validation, Fixed Validation

06
OUR METHOD
Definition, Formulation, Visualization, Results

INTRODUCTION

Definition, Motivation, Challenges, Related Work

Similarity vs. Dissimilarity

Tesla Model S
Sedan 2012

- Color: red
- Angle: up front right

Toyota Corolla
Sedan 2012

- Color: red
- Angle: up front right

Tesla Model S
Sedan 2012

- Color: white
- Angle: down front left

Image Credit: Krause et al. 2011. 3D Object Representations for Fine-Grained Categorization

How to choose this similarity function?

Handcrafted Solution

- Combining appropriate features **by hand**

Metric Learning

- Learn **task-specific** similarity functions and **automate** this process

Deep Metric Learning

- Use Convolution Neural Networks to extract **features** and learn a semantic **embedding**

Metric Learning

"Learning a similarity function that **increases** the **similarity** between **similar** objects and **decreases** the **similarity** between **dissimilar** ones."

Deep Metric Learning

- The default setup is introduced by Song et al. in Deep Metric Learning via Lifted Structure Feature Embedding
- Convolutional Neural Network is trained having available **image annotations** for each image and using a **loss** function that should have the Metric Learning **properties**.
- **Half** of the classes of the dataset are used for **training**, while the **other half** half for **testing**.
- Former losses: **Contrastive**, **Triplet**

Visualization of the **embedding space** on the test split of CARS196 using the **LiftedStructure** loss

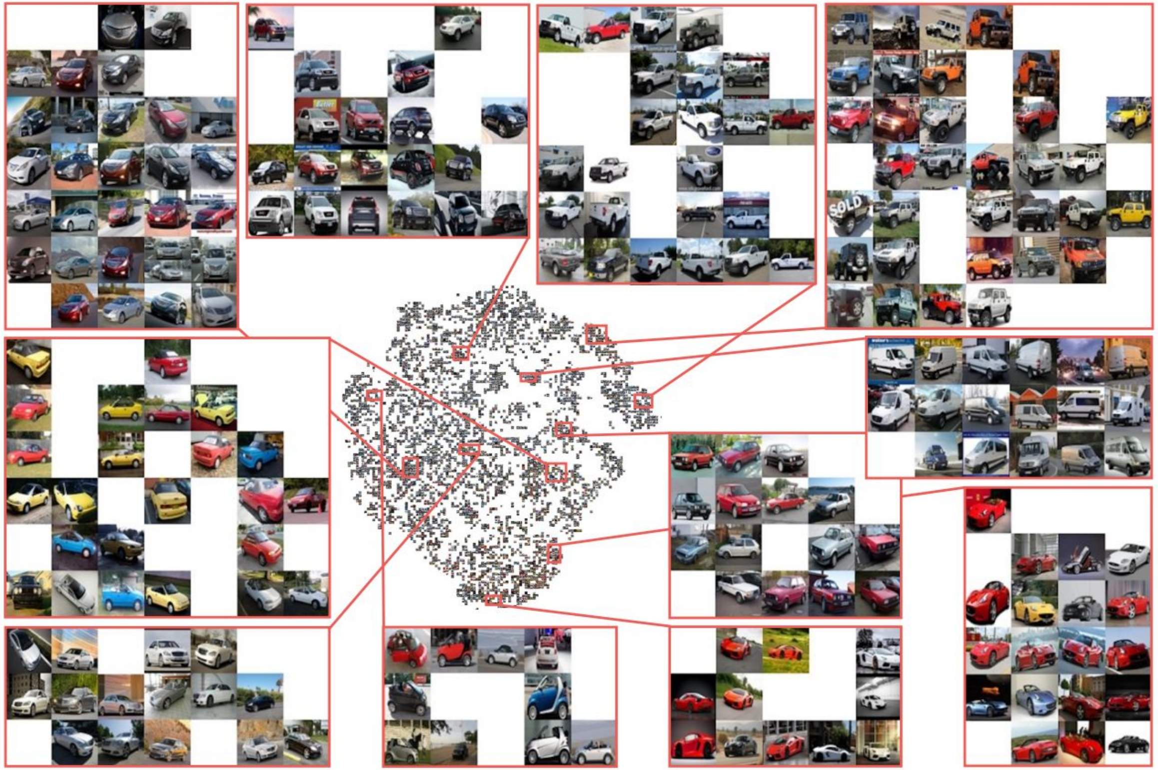


Image Credit: Song et al. CVPR 2016. Deep Metric Learning via Lifted Structured Feature Embedding

BACKGROUND

Metric Learning, Neural Networks, Deep Metric Learning

Metric Learning

$$s(x, y) \rightarrow s'(x, y) = s(f(x), f(y))$$

Linear Metric Learning

- Mapping f is **linear**

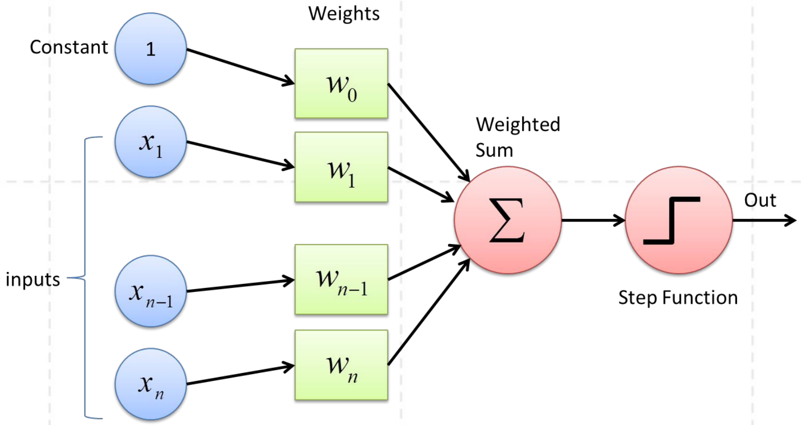
Nonlinear Metric Learning

- Mapping f is **nonlinear**
- Can be done extending linear methods via **kernelization**

Neural Networks

1. Perceptron
2. Multilayer Perceptrons (MLPs)
3. Convolutional Neural Networks (CNNs)

Perceptron



$$y = f(x; w) = sgn(w^\top x)$$

where:

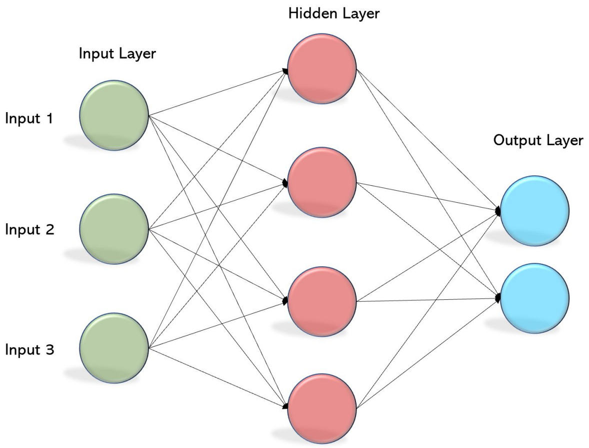
$x \in \mathbb{R}^d$ is the **input**

$w \in \mathbb{R}^d$ is a **weight** vector

$$sgn(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$ is the **step** function

Rosenblatt. 1962. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms
Image credit: Mahdid, Perceptron algorithm from scratch in Python

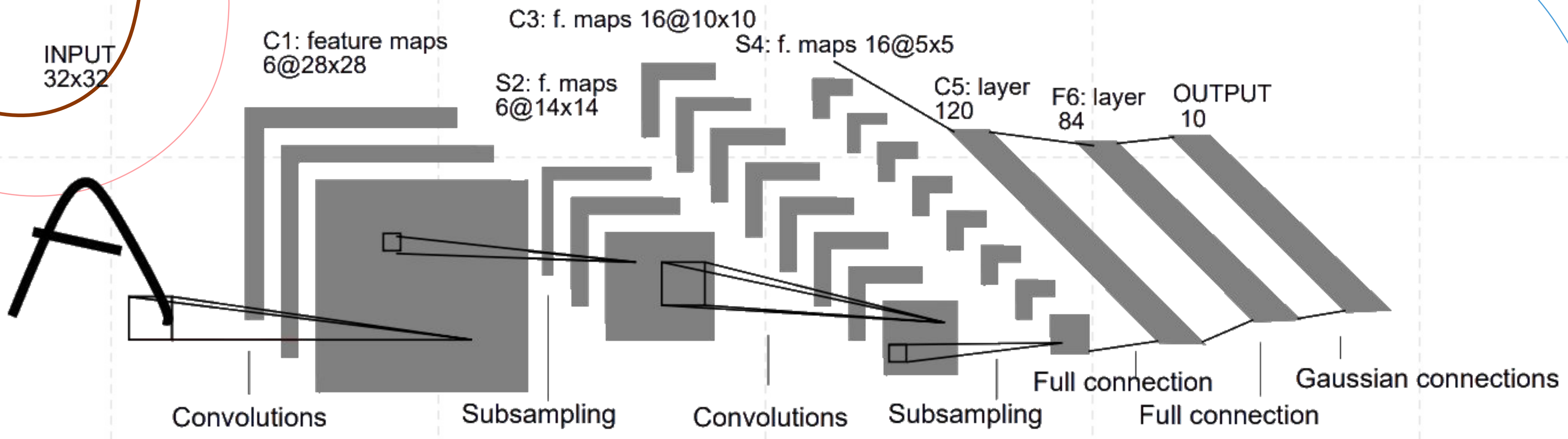
MultiLayer Perceptrons



- Efficient **nonlinear function approximators**
- MultiLayer Perceptron defines a **mapping** $f(x;\theta)$ and learns the value of **parameters** θ that result in the best **approximation** of a function $f^*(x)$
- Then naive MultiLayer Perceptron of **figure** can be formulated as: $f(x) = f^{(2)}(f^{(1)}(x))$, in which the functions are connected in **chains** and represent respectively the first and second layer it
- **Activation functions**: step, sigmoid, hyperbolic tangent, **rectified linear unit** (ReLU)

Image credit: Mohanty, Multi layer perceptron models on real world banking data

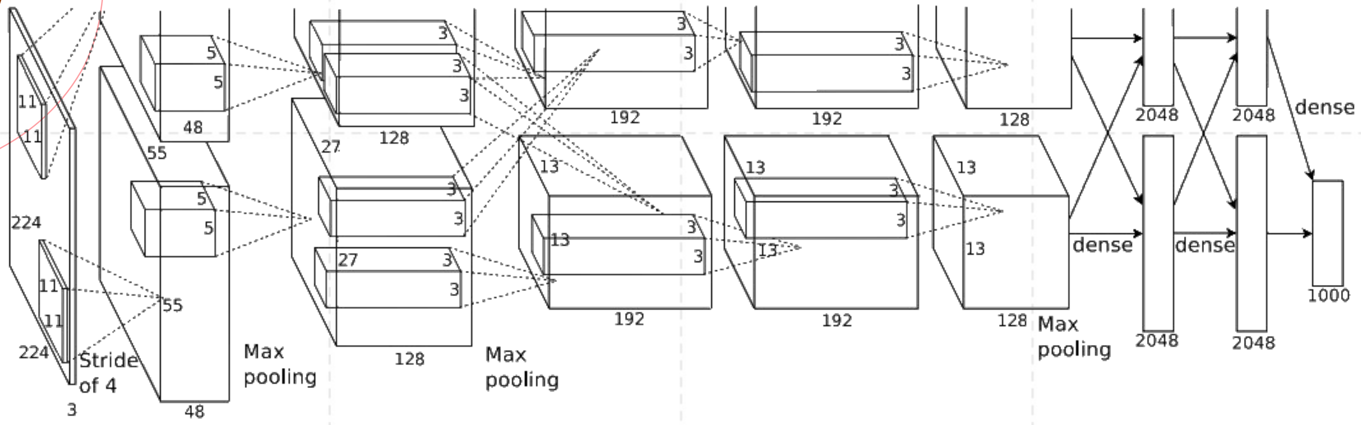
LeNet



- 2 convolutional and 3 fully connected layers
- Convolutional layer consists of: convolutions, activation function, pooling
- **Convolution**: sliding a kernel (or equivalently a filter) over an image
- **Pooling**: replaces the output of a location with a summary statistic of the nearby outputs

LeCun et al. Proceedings of the IEEE 1998. Gradient-based learning applied to document recognition

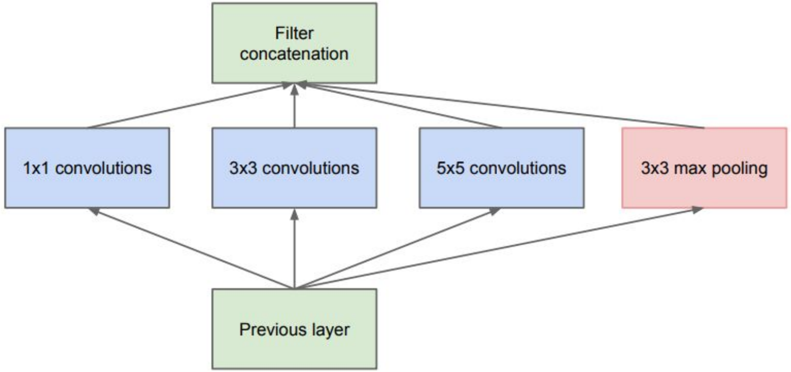
AlexNet

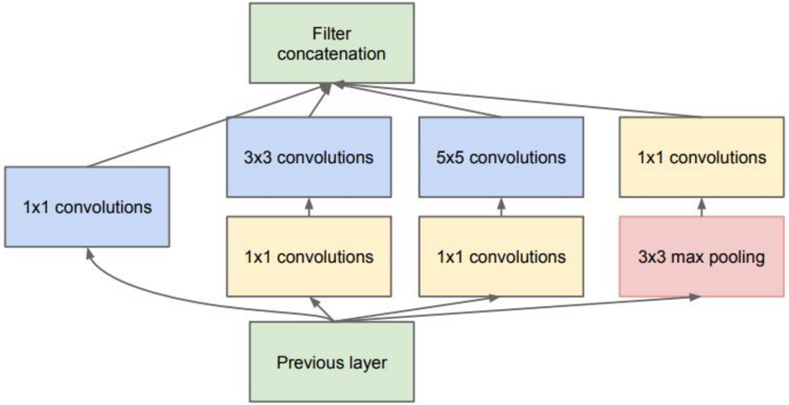


- 5 convolutional and 3 fully connected layers
- The first to use the **ReLU** as an activation function
- Winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012, **outperforming** all its competitors by more than 10%
- Probably the **beginning** of **Deep Learning**

Krizhevsky et al. NIPS 2012. ImageNet Classification with Deep Convolutional Neural Networks

GoogLeNet (Inception v1)





Naive Inception module: simple feature-wise concatenation of three different convolutions and one max pooling

Inception module: 1x1 kernels are used as bottlenecks for dimensionality reduction

- 22 layers
- **Inception** module: 25 times **less** parameters than **AlexNet**
- Winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2014

Szegedy et al. CVPR 2015. Going Deeper With Convolutions

BNInception (Inception v2)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

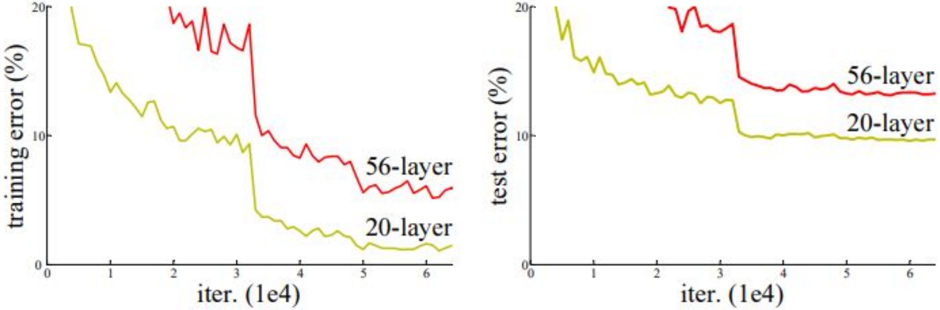
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

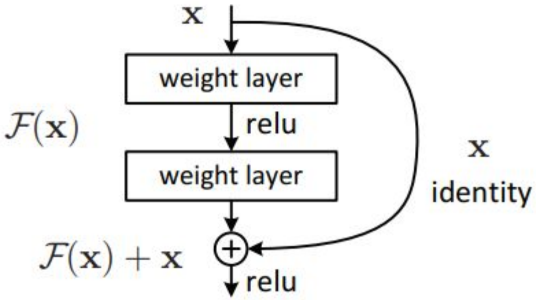
- Same architecture as GoogLeNet, but:
- Makes use of **batch normalization** transform
- BN layer can be added to any Network to manipulate any set of activation functions

Szegedy et al. CVPR 2016. Rethinking the Inception Architecture for Computer Vision

ResNets



Training and test error of a 20-layer and 56-layer Network.
Increasing depth leads to **worse** performance.



The **residual** block.

- **Motivation:** increasing Network depth does not work by simply stacking more layers, as there is the notorious problem of **vanishing gradients**
- **Idea:** identity shortcut connections that skip one or more layers. These are the **residual blocks**.
- An ensemble of ResNets was the winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2015

He et al. ICCV 2016. Deep Residual Learning for Image Recognition

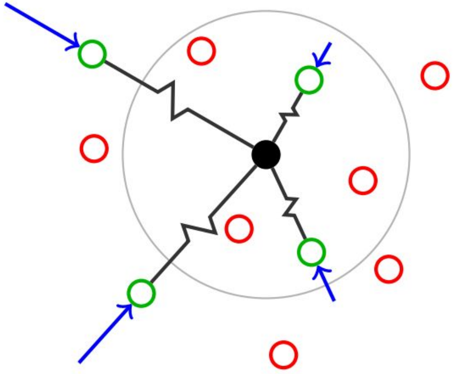
Deep Metric Learning

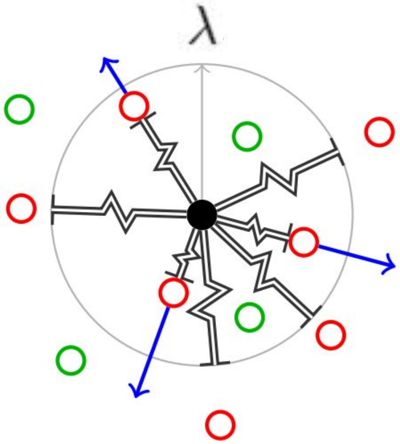
- CNN learns the **nonlinear mapping** from each input to a **lower dimensional** and semantically powerful **embedding**
- This is done by **minimizing** a **loss** function that:
  - **pushes** embeddings of images of the **same class closer**
  - **pulls** embeddings of images of **different classes apart**
- Loss functions can be split into:
  - **Embedding** loss functions (pair-based, triplet-based, in general tuple-based)
  - **Classification** loss functions (proxy-based)

Deep Metric Learning

- Let $x_i \in \mathbb{R}^d$ be a real-value instance **vector,** $X \in \mathbb{R}^{m \times d}$ the corresponding instance **matrix** and $y \in \{1, 2, ..., C\}^m$ a **label** vector for the m training **samples** respectively, where C are the **classes** and d the embedding dimension
- An input x_i is projected in a l-dimensional space by $f(\cdot;\theta) : \mathbb{R}^d \rightarrow S^l$, where f is a Neural Network parametrized by θ
- The **similarity** of two samples is defined as the dot product $S_{ij} =< f(x_i;\theta), f(x_j;\theta) >$ resulting in a $m \times m$ similarity matrix S whose element at (i,j) is S_{ij}
- For classification loss functions: let $\{w_1, ..., w_C\} \in \mathbb{R}^{d \times C}$ be a **weight** vector corresponding to **proxies**

Contrastive





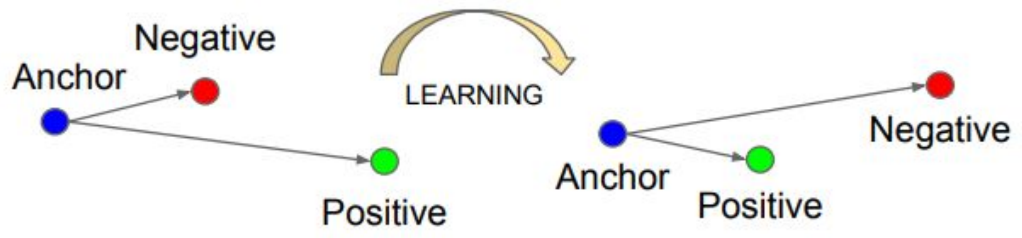
- Designed to **encourage**:
  - **positive pairs** to be as **close** as possible
  - **negative pairs** to be **apart** from each other over a **margin** λ :

$$\mathcal{L}_{Contrastive} = (1 - \mathcal{I}_{ij})[S_{ij} - \lambda]_+ - \mathcal{I}_{ij} S_{ij}$$

where $\mathcal{I}_{ij} = 1$ indicates a positive pair, while $\mathcal{I}_{ij} = 0$ indicates a negative one.

Chopra et al. CVPR 2005. Learning a similarity metric discriminatively, with application to face verification
Hadsell et al. CVPR 2006. Dimensionality Reduction by Learning an Invariant Mapping

Triplet



- Designed to ensure that an input vector x_i^a called an **anchor** is:
  - more **similar** to all other positives x_i^p
  - **than** to any other negative x_i^n
- Thus, the **Triplet constraint**:

$$S_{ap} > S_{an} + \lambda, \forall (x_i^a, x_i^p, x_i^n) \in \mathcal{T}$$

where S_{ap} and S_{an} denote the similarity of a positive pair and a negative pair with an anchor respectively, λ is a margin enforced between positives and negatives and $\mathcal{T}$ is the set of all possible triplet is the training set

- The **Triplet loss** is:

$$\mathcal{L} = [S_{an} - S_{ap} + \lambda]_+$$

Kilian et al. NIPS 2006. Distance metric learning for large margin nearest neighbor classification
Schroff et al. CVPR 2015. FaceNet: A unified embedding for face recognition and clustering
Hoffer et al. SIMBAD 2015. Deep metric learning using triplet network

Triplet

- **Issue:** Generating **all** the possible triplets would result in many triplets that **easily fulfil** the Triplet constraint and thus do not contribute in training, as their gradients are really **small** or even **zero**
- **Solution:** Mining is the process of finding informative pairs:
  - **Hard**, selecting:
    - hard positives, such that: $\underset{x_i^p}{\arg\min} < f(x_i^a), f(x_i^p) >$
    - hard negatives, such that: $\underset{x_i^n}{\arg\max} < f(x_i^a), f(x_i^n) >$
  - **Semi-hard**, selecting: $n_{ap} = \underset{n: S_{ap} > S_{an}}{\arg\max} S_{an},$
- **Mining:**
  - **Online:** selecting samples from within the batch
  - **Offline:** selecting samples from the whole training in order to construct the batch

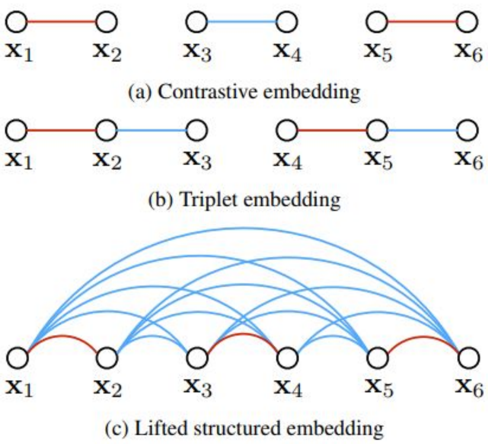
Kilian et al. NIPS 2006. Distance metric learning for large margin nearest neighbor classification
Schroff et al. CVPR 2015. FaceNet: A unified embedding for face recognition and clustering
Hoffer et al. SIMBAD 2015. Deep metric learning using triplet network

LiftedStructure

- Takes full advantage of **each sample** within the **batch** by "lifting the **vector** of pairwise distances to the **matrix** of pairwise distances".
- **LiftedStructure** loss:

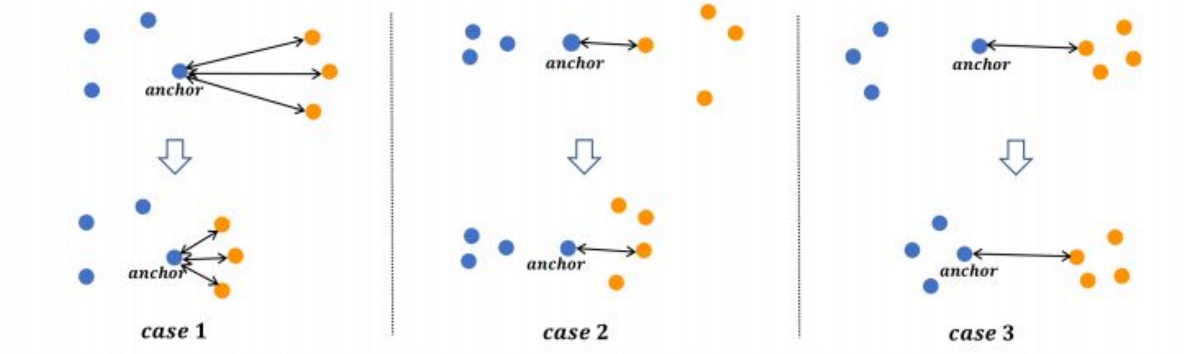
$$\mathcal{L}_{LiftedStructure} = \sum_{i=1}^{m} \left[log \sum_{y_k = y_i} e^{\lambda - S_{ik}} + log \sum_{y_k \neq y_i} e^{S_{ik}} \right]_+$$

where λ is a fixed margin.
- **Issue:** Randomly selected negative pairs might carry limited information
- **Solution**: Online hard mining.



Song et al. CVPR 2016. Deep Metric Learning via Lifted Structured Feature Embedding

MultiSimilarity



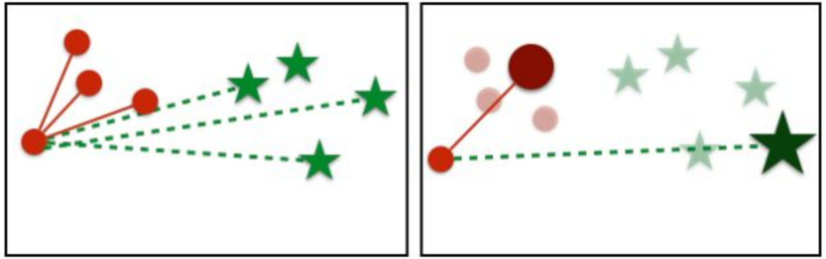
- Defines **three** different types of **similarity**:
  - **S**: Self-similarity
  - **N**: Negative relative similarity
  - **P**: Positive relative similarity
- Introduces a loss function taking advantage of all types of similarity:

$$\mathcal{L}_{\text{MultiSimilarity}} = \frac{1}{m}\sum_{i=1}^{m}\left\{\frac{1}{\alpha}\log\left[1+\sum_{k\in\mathcal{P}_i} e^{-\alpha(S_{ik}-\lambda)}\right] + \frac{1}{\beta}\log\left[1+\sum_{k\in\mathcal{N}_i} e^{\beta(S_{ik}-\lambda)}\right]\right\},$$

where α, β, λ are hyperparameters, $\mathcal{P}_i$ and $\mathcal{N}_i$ the sets of positives and negatives respectively

Wang et al. CVPR 2019. Multi-Similarity Loss With General Pair Weighting for Deep Metric Learning

ProxyNCA

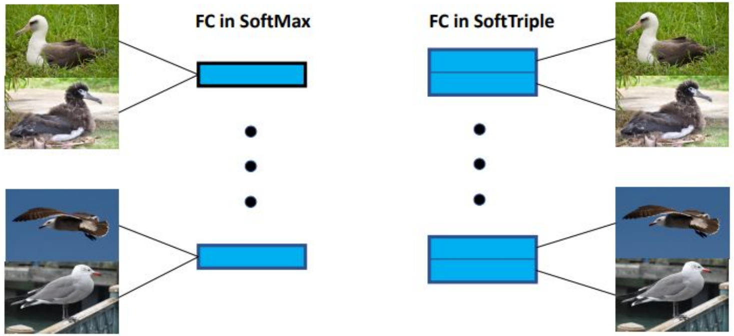


- **Issue:** when using embedding losses, only a specific subset of all possible tuples are taken into consideration
- **Solution:** use of proxies that serve as a concise representation for each semantic concept
- Proxies are equal to the number of **classes**
- Proxy-based **Triplet** loss consisting of: anchor, learnable positive proxy, learnable negative proxy

$$\mathcal{L}_{ProxyNCA} = -log \frac{e^{w_{y_i}^T x_i}}{\sum_{j \neq y_i} e^{w_j^T x_i}},$$

Movshovitz et al. ICCV 2017. No fuss distance metric learning using proxies

SoftTriple



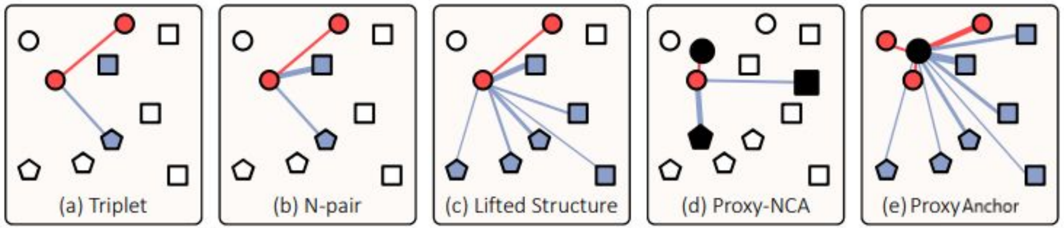
- **Motivation:** a class in a real-world data can consist of multiple local clusters and thus a single proxy might not be able to **capture** the inherent **structure** of the data
- **Idea:** a proxy-based (softmax-like) Triplet loss that uses **multiple** proxies and thus is more capable of modeling the **intra-class variability**

$$\mathcal{L}_{SoftTriple} = -log\frac{e^{\alpha(w_{y_i}^T x_i - \lambda)}}{e^{\alpha(w_{y_i}^T x_i - \lambda)} + \sum_{j \neq y_i} e^{\alpha w_j^T x_i}},$$

where λ is a margin and α is a scaling factor

Qian et al. ICCV 2019. SoftTriple Loss: Deep Metric Learning Without Triplet Sampling

ProxyAnchor



- **Motivation:** a loss function that **combines** the good points of **embedding** and **classification** loss functions, while **correcting** their defects
- **Idea:** A proxy-based loss that associates each **proxy** with **all samples** in a batch
- Thus:
  - as a proxy-based loss: fast and stable **convergence**, no tuple sampling, robust against noisy labels and outliers
  - while also utilizing **data-to-data** relations

$$\mathcal{L}_{\text{ProxyAnchor}} = \frac{1}{|W+|} \sum_{w \in W+} \log \left(1 + \sum_{x \in X_w^+} e^{-\alpha(w^T x - \lambda)} \right) + \frac{1}{|W|} \sum_{w \in W} \log \left(1 + \sum_{x \in X_w^-} e^{\alpha(w^T x + \lambda)} \right),$$

where $\lambda > 0$ is a margin, $\alpha > 0$ is a scaling factor, W indicates the set of all proxies, W^+ denotes the set of positive proxies in the batch, X_w^+ and X_w^- the set of positive and negative embedding vectors of w

Kim et al. CVPR 2020. Proxy Anchor Loss for Deep Metric Learning

EXPERIMENTAL SETUP

Datasets, Networks, Evaluation, Implementation Details, Issues

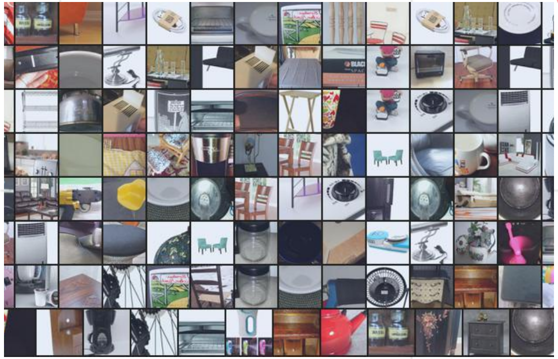
Datasets

CUB200-2011

- Birds
- 200 classes
- 11788 images
- ~59 images/class

CARS196

- Cars
- 196 classes
- 16185 images
- ~82 images/class



SOP

- Online products
- 22634 classes
- 120023 images
- ~5 images/class

Wah et al. 2011. The Caltech-UCSD Birds-200-2011 Dataset
Krause et al. 2011. 3D Object Representations for Fine-Grained Categorization
Song et al. CVPR 2016. Deep Metric Learning via Lifted Structured Feature Embedding

Networks

Loss Function	Network	Embedding Size
Contrastive	2-layer CNN, 5-layer CNN	2, 50
Triplet	22-layer CNN, GoogLeNet	128
LiftedStructure	GoogLeNet	64
NPair	GoogLeNet	64, 512
ProxyNCA	BNInception	64
Margin	ResNet50	128
ArcFace	ResNet50, ResNet100	512
MultiSimilarity	BNInception	64, 512
SoftTriple	BNInception	64, 512
ProxyAnchor	BNInception	512

Evaluation

- **Recall@k** metric:
  - Compute the **embeddings** of every image in the **test** set
  - Each of these retrieves **k nearest neighbors** from the test set
    - Receives score **1** if an image of the same class is retrieved among the k
    - Otherwise receives score **0**
- Recall@k **averages** this score over **all** images of the test set

Implementation Details

Extensive experiments on:

- all 3 **datasets**:
  - CUB200-2011
  - CARS196
  - SOP
- most common **Networks**:
  - GoogLeNet
  - BNInception
  - ResNet50
- 4 different **embedding sizes**:
  - 64
  - 128
  - 512
  - 1024
- 10 different **loss functions**:
  - Contrastive
  - Triplet
  - LiftedStructure
  - NPair
  - ProxyNCA
  - Margin
  - ArcFace
  - MultiSimilarity
  - SoftTriple
  - ProxyAnchor

Implementation Details

Extensive experiments:

- Under the **same conditions** (so that no method is favored):
  - **epochs**: 100
  - **optimizer**: AdamW variant of Adam
  - **scheduler:** StepLR
  - **hyperparameters**:
    - of **losses** like margins, scales, etc. are **taken** from **papers**
    - of **optimization** like learning rate and scheduling **taken** from papers **once available**, **else** from a small **search** around the **default values**
  - **batch size**:
    - **100** for ResNet50
    - **180** for GoogLeNet and BNInception
  - **mining**: as proposed in the respective paper
  - **sampling**: as proposed in the respective paper
  - **evaluation:** Recall@k, which shows the retrieval quality
- Using either **NVIDIA V100** or the **NVIDIA GeForce RTX 2080 Ti**

Implementation Details

Loss Function	Hyperparameter	Value
Contrastive	margin λ	0.5
Triplet	margin λ	0.1
LiftedStructure	margin λ	0.5
NPair	l_2	0.02
ProxyNCA	proxy lr	0.00001
Margin	margin λ	0.5
	beta	1.2
	beta lr	0.00005
ArcFace	margin λ	28.6
	scale s	64
	weights lr	0.0001
MultiSimilarity	margin λ	0.5
	scale α	2
	scale β	50
	epsilon	0.1
SoftTriple	margin λ	0.1
	scale α	20
	weights lr	0.0001
	gamma	10
	tau	0.2
ProxyAnchor	margin λ	0.1
	scale α	32

Loss Function	Mining Method
Contrastive	-
Triplet	semi-hard
LiftedStructure	hard
NPair	-
ProxyNCA	-
Margin	distance weighted
ArcFace	-
MultiSimilarity	hard
SoftTriple	-
ProxyAnchor	-

Loss Function	Sampling Method
Contrastive	random
Triplet	random
LiftedStructure	balanced
NPair	random
ProxyNCA	random
Margin	random
ArcFace	random
MultiSimilarity	balanced
SoftTriple	random
ProxyAnchor	random

Experiment	Learning Rate	Step Size	Gamma
CUB200-2011 ResNet50	0.0001	5	0.1
CUB200-2011 BNInception	0.0001	10	0.1
CUB200-2011 GoogLeNet	0.0001	10	0.1
CARS196 ResNet50	0.0001	10	0.1
CARS196 BNInception	0.0001	20	0.1
CARS196 GoogLeNet	0.0001	20	0.1
SOP ResNet50	0.0006	10	0.25
SOP BNInception	0.0006	20	0.25
SOP GoogLeNet	0.0006	20	0.25

Issues

Why do we conduct these experiments?

- **Unfair comparisons** concerning:
  - Networks
  - embedding sizes
  - details omitted (BN freeze, GAP + GMP, crop type)
- Lack of **validation** set
- **Benchmark** and **Ablation Study**

EXPERIMENTAL RESULTS

Results, Discussion

Results

CUB200-2011 ResNet50

- Performance:
  - Worst: Triplet, **NPair**
  - Best: **ProxyAnchor**, SoftTriple, MultiSimilarity
  - Better than expected: **Contrastive**
- Unfair comparison confirmed:
  - In paper (R@1):
    - Margin: 63.60% (R)
    - LiftedStructure: **43.57%** (G)
    - Triplet: **42.60%** (G)
  - In our results (R@1)
    - Margin: 63.00% (R)
    - LiftedStructure: **60.16%** (R)
    - Triplet: **60.48%** (R)

R: ResNet50, G: GoogLeNet

(a) embedding size = 64.

	R@1	R@2	R@4	R@8
Contrastive	60.28	71.49	80.77	87.07
Triplet	57.56	69.62	80.22	87.44
LiftedStructure	58.36	70.41	79.25	87.20
NPair	57.28	68.54	78.92	87.29
ProxyNCA	60.25	71.51	80.71	87.68
Margin	59.66	71.10	81.06	88.40
ArcFace	58.32	69.23	78.38	85.84
MultiSimilarity	60.84	72.15	81.67	88.86
SoftTriple	61.28	73.11	82.58	89.37
ProxyAnchor	**62.93**	**74.00**	**83.13**	**89.62**

(b) embedding size = 128.

	R@1	R@2	R@4	R@8
Contrastive	62.64	73.66	82.55	89.03
Triplet	60.48	72.13	82.11	89.03
LiftedStructure	60.16	72.35	81.88	88,44
NPair	58.91	70.66	79.98	87.74
ProxyNCA	62.76	73.13	82.17	88.50
Margin	63.00	74.00	83.59	90.41
ArcFace	61.33	71.84	80.13	87.36
MultiSimilarity	63.96	74.85	83.63	90.31
SoftTriple	64.16	75.59	84.01	90.21
ProxyAnchor	**66.71**	**76.79**	**85.18**	**90.63**

(c) embedding size = 512.

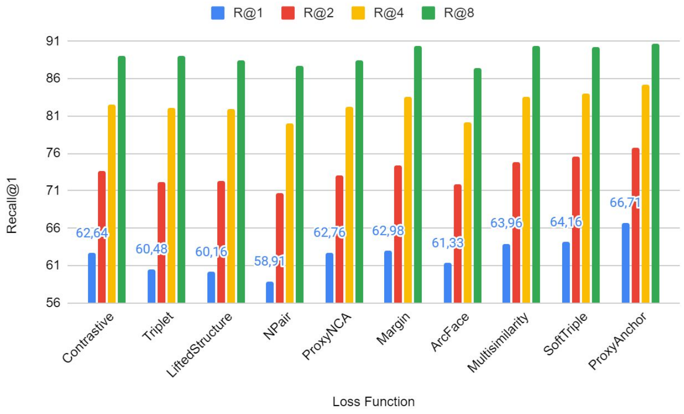
	R@1	R@2	R@4	R@8
Contrastive	64.87	75.41	83.27	89.67
Triplet	63.52	75.62	84.38	90.50
LiftedStructure	65.92	75.81	84.50	90.41
NPair	61.36	72.81	82.08	89.01
ProxyNCA	65.22	75.55	83.76	89.60
Margin	64.99	76.15	84.60	90.46
ArcFace	64.40	74.68	83.20	89.60
MultiSimilarity	68.69	78.56	86.75	92.08
SoftTriple	67.27	77.73	86.19	92.00
ProxyAnchor	**69.48**	**79.27**	**86.95**	**92.37**

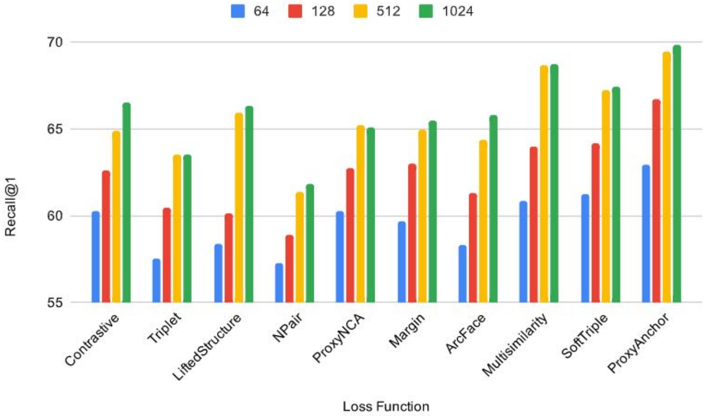
(d) embedding size = 1024.

	R@1	R@2	R@4	R@8
Contrastive	66.51	76.50	85.15	90.73
Triplet	63.55	75.35	84.03	90.36
LiftedStructure	66.34	76.67	84.47	90.36
NPair	61.83	72.60	82.07	89.01
ProxyNCA	65.12	74.78	83.56	89.60
Margin	65.48	76.54	84.53	91.15
ArcFace	65.82	76.71	84.18	89.70
MultiSimilarity	68.72	79.17	**87.15**	92.29
SoftTriple	67.42	78.16	86.02	91.64
ProxyAnchor	**69.82**	**79.86**	87.12	**92.69**

Results

CUB200-2011 ResNet50





- Chronological order
- Embedding size = 128
- Lack of improvement visible

- Sizes 512 and 1024 almost the same retrieval quality

Results

CUB200-2011 BNInception

- Performance:
  - Worst: Triplet, NPair
  - Best: **ProxyAnchor**, SoftTriple, MultiSimilarity
  - Better than expected: Contrastive, **LiftedStructure**, **SoftTriple**
- SoftTriple:
  - In paper (R@1): **65.40%**
  - In our results (R@1): **66.76%**
- Unfair comparison confirmed:
  - In paper (R@1):
    - ProxyNCA: 49.21% (BN)
    - LiftedStructure: **43.57%** (G)
  - In our results (R@1)
    - ProxyNCA: 56.98% (BN)
    - LiftedStructure: **58.29%** (BN)

R: ResNet50, G: GoogLeNet, BN:BNInception

(a) embedding size = 64.

	R@1	R@2	R@4	R@8
Contrastive	58.88	69.70	78.53	86.12
Triplet	55.82	67.13	77.11	83.95
LiftedStructure	58.29	68.96	79.43	87.22
NPair	54.17	65.98	76.87	83.80
ProxyNCA	56.98	67.10	77.08	85.14
Margin	56.80	68.08	78.00	85.24
ArcFace	55.77	67.92	77.92	85.50
MultiSimilarity	57.24	69.31	79.49	86.92
SoftTriple	58.07	69.42	79.42	87.39
ProxyAnchor	**61.06**	**72.67**	**82.05**	**88.67**

(b) embedding size = 128.

	R@1	R@2	R@4	R@8
Contrastive	61.24	71.79	80.40	87.62
Triplet	58.56	70.12	79.10	86.45
LiftedStructure	61.60	73.36	81.97	88.61
NPair	56.90	69.02	78.02	84.98
ProxyNCA	60.15	71.08	81.15	85.80
Margin	60.80	71.45	81.90	86.24
ArcFace	59.94	71.08	80.57	87.63
MultiSimilarity	61.92	73.28	82.99	89.21
SoftTriple	63.44	74.29	83.27	**89.96**
ProxyAnchor	**63.88**	**74.51**	**83.86**	89.92

(c) embedding size = 512.

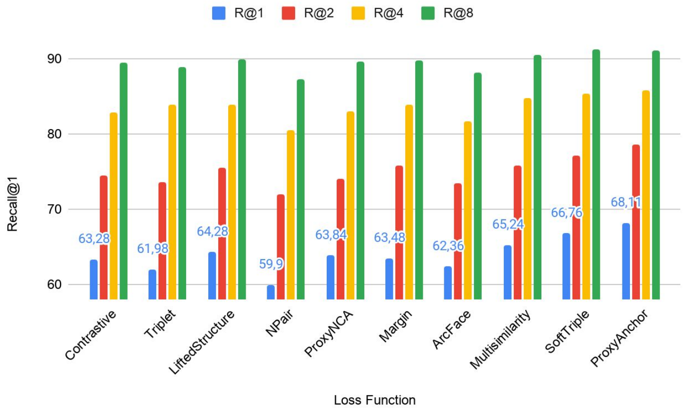
	R@1	R@2	R@4	R@8
Contrastive	63.28	74.51	82.83	89.50
Triplet	61.98	73.59	83.80	88.87
LiftedStructure	64.28	75.47	83.91	89.89
NPair	59.90	71.98	80.47	87.25
ProxyNCA	63.84	74.02	82.98	89.54
Margin	63.48	75.86	83.90	89.78
ArcFace	62.36	73.48	81.67	88.08
MultiSimilarity	65.24	75.76	84.69	90.48
SoftTriple	66.76	77.09	85.36	**91.21**
ProxyAnchor	**68.11**	**78.63**	**85.77**	91.12

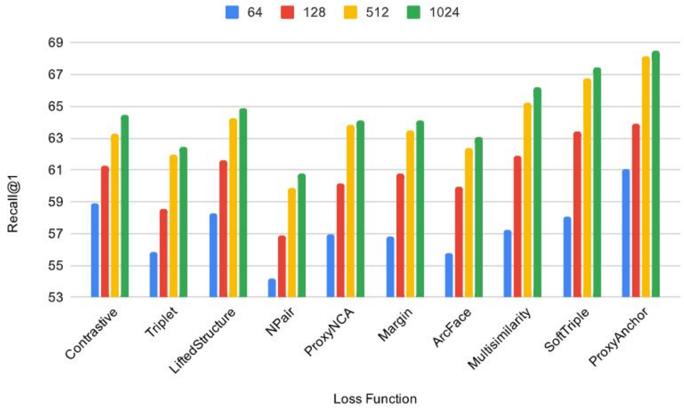
(d) embedding size = 1024.

	R@1	R@2	R@4	R@8
Contrastive	64.43	74.48	82.88	89.47
Triplet	62.45	73.80	83.10	89.20
LiftedStructure	64.86	75.68	84.00	90.01
NPair	60.76	71.89	81.67	88.40
ProxyNCA	64.10	74.40	82.80	89.14
Margin	64.08	75.40	83.01	89.90
ArcFace	63.07	73.94	83.04	88.93
MultiSimilarity	66.22	77.62	85.40	90.94
SoftTriple	67.44	78.11	**85.91**	91.28
ProxyAnchor	**68.47**	**78.41**	85.75	**91.36**

Results

CUB200-2011 BNInception





- Embedding size = 512
- Impressive performance: Contrastive, SoftTriple

- Size 1024 improves the retrieval quality by little

Results

CUB200-2011 GoogLeNet

- Performance:
  - Worst: Triplet, NPair, **ProxyNCA**
  - Best: ProxyAnchor
  - Worse than before: **MultiSimilarity**, **SoftTriple**
  - Better than expected: Contrastive, LiftedStructure
  - Better than before: **ArcFace** (ranks second using sizes of 512 and 1024)

(a) embedding size = 64.

	R@1	R@2	R@4	R@8
Contrastive	56.36	67.94	78.41	86.19
Triplet	52.12	63.69	75.08	84.37
LiftedStructure	55.18	67.76	77.51	86.02
NPair	48.76	60.38	71.78	81.36
ProxyNCA	51.01	61.93	73.07	82.56
Margin	54.27	66.48	77.13	85.69
ArcFace	52.92	63.52	74.31	82.77
MultiSimilarity	53.47	65.69	76.33	85.07
SoftTriple	55.00	67.51	77.95	85.96
ProxyAnchor	**58.07**	**69.23**	**79.37**	**87.05**

(b) embedding size = 128.

	R@1	R@2	R@4	R@8
Contrastive	57.73	69.04	79.51	87.29
Triplet	55.06	67.22	77.80	85.62
LiftedStructure	58.07	69.78	79.56	87.14
NPair	50.30	61.19	72.99	81.79
ProxyNCA	55.77	66.88	76.90	85.16
Margin	57.88	69.54	79.29	86.99
ArcFace	56.94	68.01	77.97	85.67
MultiSimilarity	55.66	68.37	78.87	86.95
SoftTriple	57.00	68.91	79.61	87.61
ProxyAnchor	**60.23**	**71.89**	**82.26**	**88.86**

(c) embedding size = 512.

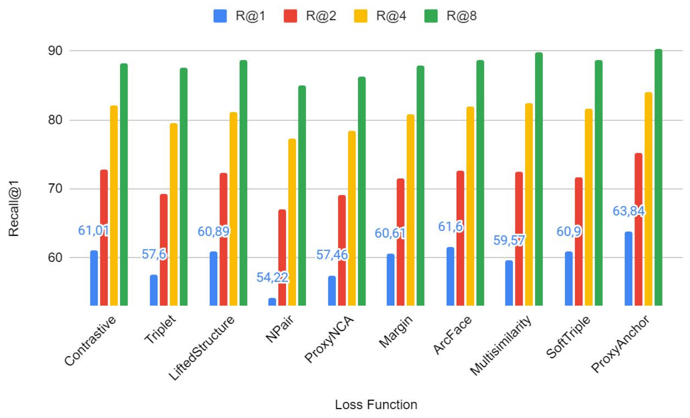
	R@1	R@2	R@4	R@8
Contrastive	61.01	72.79	82.07	88.20
Triplet	57.60	69.35	79.60	87.56
LiftedStructure	60.89	72.37	81.20	88.67
NPair	54.22	67.10	77.29	85.05
ProxyNCA	57.46	69.09	78.40	86.30
Margin	60.61	71.51	80.77	87.90
ArcFace	61.60	72.67	81.95	88.62
MultiSimilarity	59.57	72.42	82.42	89.76
SoftTriple	60.90	71.62	81.67	88.71
ProxyAnchor	**63.84**	**75.25**	**84.05**	**90.29**

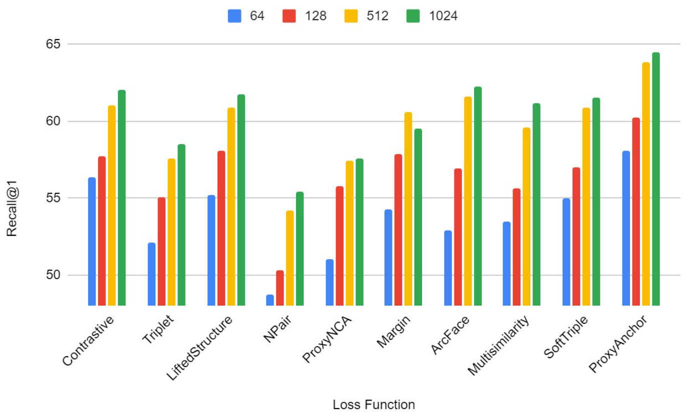
(d) embedding size = 1024.

	R@1	R@2	R@4	R@8
Contrastive	62.05	73.14	82.14	88.78
Triplet	58.54	69.68	80.22	87.84
LiftedStructure	61.77	72.74	82.19	89.08
NPair	55.40	67.58	77.86	85.75
ProxyNCA	57.60	69.02	78.61	86.34
Margin	59.55	71.49	80.96	88.08
ArcFace	62.24	73.57	82.38	88.42
MultiSimilarity	61.16	72.92	82.51	89.18
SoftTriple	61.55	73.09	82.49	89.61
ProxyAnchor	**64.47**	**75.96**	**84.61**	**90.63**

Results

CUB200-2011 GoogLeNet





- Embedding size = 512
- Impressive performance: ArcFace, Contrastive, LiftedStructure

- Size 1024 improves significantly the retrieval quality

Results

CARS196 BNInception

- Performance:
  - Worst: Triplet, NPair
  - Best: **ProxyAnchor**, SoftTriple, MultiSimilarity
  - Ranked in the middle: LiftedStructure, ProxyNCA, Margin
  - Better than expected: **Contrastive**
  - Better as the embedding size increases: **ArcFace**
- Unfair comparison confirmed:
  - In paper (R@1):
    - ProxyNCA: 73.22% (BN)
    - LiftedStructure: **52.98%** (G)
  - In our results (R@1)
    - ProxyNCA: 72.52% (BN)
    - LiftedStructure: **73.53%** (BN)

G: GoogLeNet, BN:BNInception

(a) embedding size = 64.

	R@1	R@2	R@4	R@8
Contrastive	73.39	81.98	88.14	92.61
Triplet	70.02	79.12	85.98	91.01
LiftedStructure	73.53	82.51	88.40	92.81
NPair	68.54	78.21	84.90	89.87
ProxyNCA	72.52	81.20	86.05	91.20
Margin	72.94	81.48	87.09	91.68
ArcFace	69.33	78.82	85.62	90.74
MultiSimilarity	76.25	84.60	90.30	94.50
SoftTriple	77.70	86.11	91.33	95.02
ProxyAnchor	**79.79**	**87.27**	**92.44**	**95.52**

(b) embedding size = 128.

	R@1	R@2	R@4	R@8
Contrastive	75.52	84.12	89.35	93.17
Triplet	72.48	81.80	87.90	92.02
LiftedStructure	77.68	85.27	90.47	94.12
NPair	70.56	80.18	86.50	90.46
ProxyNCA	76.10	84.98	90.03	94.24
Margin	78.12	86.03	91.24	94.45
ArcFace	75.19	83.34	88.86	92.71
MultiSimilarity	80.69	87.75	92.29	95.53
SoftTriple	81.44	89.08	**93.68**	**96.35**
ProxyAnchor	**83.11**	**89.53**	93.46	95.99

(c) embedding size = 512.

	R@1	R@2	R@4	R@8
Contrastive	79.09	86.36	91.69	95.06
Triplet	77.02	84.12	89.79	93.56
LiftedStructure	79.82	86.79	91.86	94.92
NPair	73.25	81.86	86.58	90.45
ProxyNCA	81.02	86.97	92.47	95.12
Margin	81.98	87.75	91.75	94.85
ArcFace	79.42	86.77	91.71	94.70
MultiSimilarity	83.75	89.84	93.75	96.53
SoftTriple	85.29	91.10	**94.78**	**97.10**
ProxyAnchor	**86.21**	**91.71**	94.70	96.95

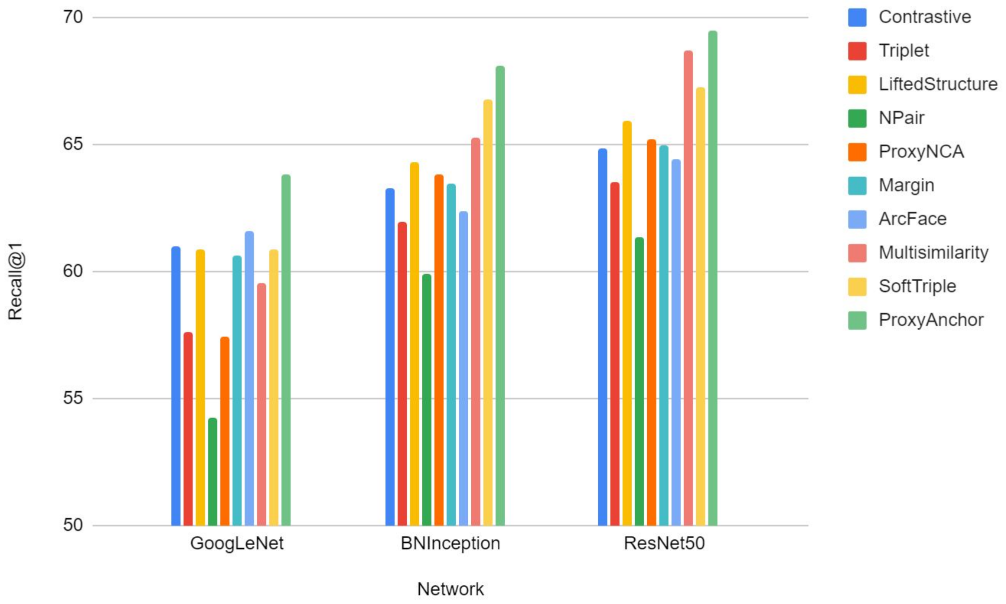
(d) embedding size = 1024.

	R@1	R@2	R@4	R@8
Contrastive	78.86	86.37	91.72	94.93
Triplet	77.40	84.23	89.98	93.47
LiftedStructure	79.46	86.70	91.39	95.00
NPair	74.28	81.98	86.79	90.63
ProxyNCA	81.90	87.70	91.66	94.45
Margin	81.78	87.60	91.78	94.90
ArcFace	79.74	86.57	91.24	94.50
MultiSimilarity	84.38	90.64	94.34	96.64
SoftTriple	86.20	**91.88**	**95.41**	**97.40**
ProxyAnchor	**86.41**	91.70	94.90	97.12

Discussion

About Networks

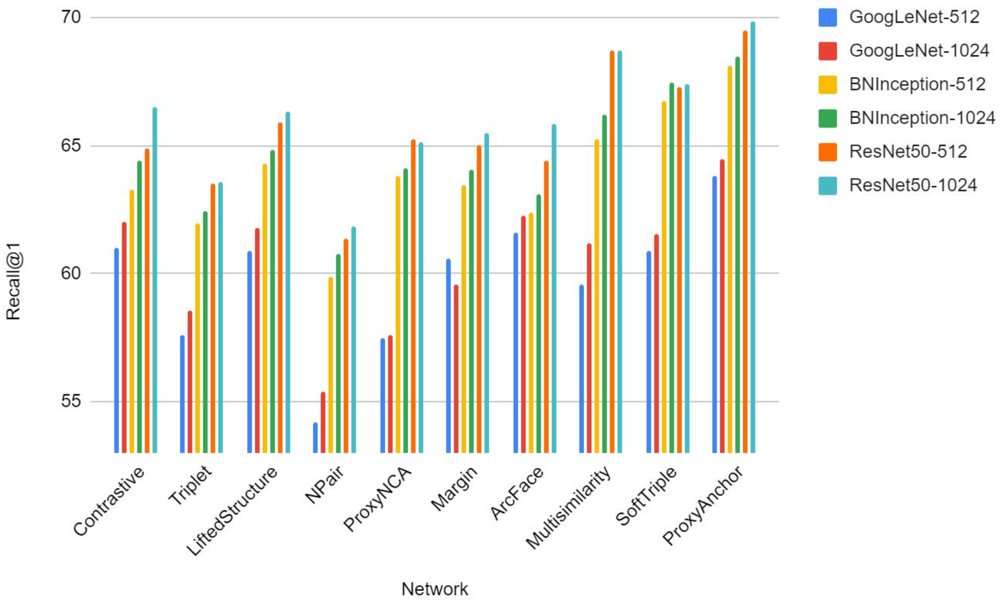
- ResNet50's representations are more **powerful**
- A loss function using ResNet **cannot** be **compared** with one using one of the other Networks
- If that happens, the **superiority** would probably be due to the **Network**, rather than due to the **loss**



Discussion

About embeddings

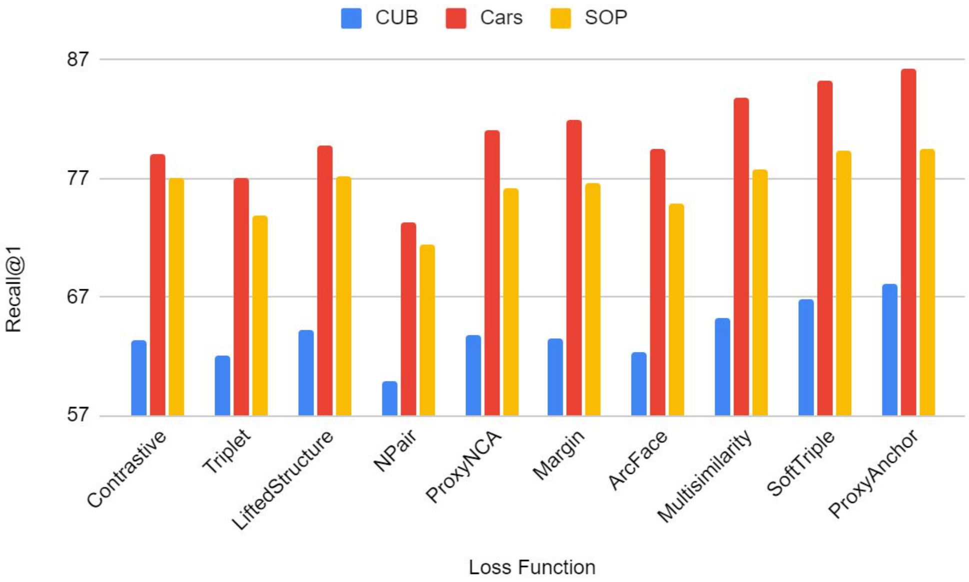
- Cannot really draw a **clear** conclusion
- GoogLeNet seems to **improve** performance when **512** →**1024**
- BNInception and ResNet50 **not** always
- Taking into account the **computational** cost: **512** the optimal



Discussion

About Datasets

- CUB200-2011 is the **smallest** one, with ~59 images/class
- CARS196 is slightly bigger with ~82 images/class
- SOP is huge with **22.5k** classes, **120k** images and ~**5** images/class
- However, CUB's retrieval **scores** are the **lowest**
- Reason: **intraclass variance** (birds in different poses and ages)



Discussion

About Loss Functions

- Embedding losses (pair-based, triplet-based, tuple-based):
    - Able to capture **data-to-data** relations
    - **Sensitive** to noisy labels and outliers
    - Can sometimes **easily** fulfil their constraints →**mining** needed
    - Converge **slowly**
- Classification losses (proxy-based):
    - Fast, reliable convergence
    - **Less** hyperparameter finetuning
    - **Robust** again noisy labels and outliers

Discussion

Tournament of Loss Functions

- A quantitative process on CUB200-2011 that will help us draw more conclusions:
  - Collect the **ranking** of each loss in **each** experiment
  - **Total** experiments=**12**=(**4** different embedding sizes **x 3** different Networks)
  - Ranking examples: ProxyAnchor=1, NPair=10
  - Sum of rankings → **Total Rankings**
  - Divide by 12 → **Average Ranking**
  - Calculate the **Standard Deviation** of each loss

Discussion

Tournament of Loss Functions

- **Winner**→ Proxy Anchor:
  - Use of Log-Sum-Exp
  - Use of proxies
  - Association of proxies with samples in batch
- **Runner Up**→SoftTriple:
  - Multiple proxies
  - Able to capture inherent structure of data
- **Third**→MultiSimilarity:
  - Use of Log-Sum-Exp
  - Data-to-data relations
- **Fourth**→Contrastive:
  - Exploits our batch size
  - Simple but effective
- **Last**→Triplet & NPair:
  - Problematic convergence
  - Sophisticated mining needed

Loss Function	Total Rankings	Average Ranking	Standard Deviation
ProxyAnchor	12	1	0
SoftTriple	38	3.16	1.19
MultiSimilarity	51	4.25	2.05
Contrastive	52	4.33	1.72
LiftedStructure	54	4.5	1.89
Margin	70	5.83	1.27
ArcFace	79	6.58	2.31
ProxyNCA	81	6.75	1.66
Triplet	103	8.85	0.51
NPair	120	10	0

Discussion

About Setup

- **Minor changes** in hyperparameters→**affect** the **performance** more than expected
- Difficulties in **finetuning**
- Not sure if hyperparameters are surely the **optimal** ones
- Lack of **validation** set→not a good tactic, **generalization** to be questioned

OUR SETUP

Cross Validation, Fixed Validation

Our Setup

Cross Validation

- **10-fold** CV
- Keep the classes of the test set the **same**
- Training classes of default setup→**9/10** Training, **1/10** Validation
- **Random selection**, as consecutive classes sometimes are semantically similar
- By the end of CV→all the classes will have been included **once** in validation

- At each **epoch**→ report R@1 on **validation set**
- By the end of **one fold**→save and load the model with the **best R@1 on validation set** for **testing**
- By the **end of CV**→compute the **average and std** of the R@1 scores of the 10 models

- Experiments using:
  - BNInception with a 512-dimensional embedding
  - CUB200-2011
  - ProxyAnchor, SoftTriple, MultiSimilarity

Our Setup

Cross Validation

Loss Function	R@1
MultiSimilarity	63.61 $\pm$0.59
SoftTriple	64.09 $\pm$0.48
ProxyAnchor	66.32 $\pm$0.44

CV R@1 scores

Loss Function	R@1
MultiSimilarity	65.24
SoftTriple	66.76
ProxyAnchor	68.11

Default Setup R@1 scores

- Hyperparameter searching proved really expensive→**not made**
- Consider that fact of training **10** models instead of **1**
- CV R@1 scores are lower because **90** classes are used

Our Setup

Fixed Validation

- **Idea:** train only 1 model, but split the classes in order to have a validation set
- **Problem:** What's the best **split ratio**?
- Answer: **90/10**

Split Ratio (Training Classes/ Validation Classes)	Best R@1 on Validation Set	R@1 on Test Set
70/30	86.13	61.28
80/20	92.53	62.74
90/10	91.49	**64.38**
95/5	93.31	62.92

Experiments using MultiSimilarity in different split schemes on CUB200-2011

Our Setup

Fixed Validation

- Experiments using:
  - BNInception with a 512-dimensional embedding
  - CUB200-2011
  - ProxyAnchor, SoftTriple, MultiSimilarity
- Exhaustive hyperparameter **grid-like** searching:
  - Define a **range of search** for each hyperparameter
  - Define a **search step**
  - Train until the impact of the value is visible ~10 epochs

Loss Function	Hyperparameter	Range of Search	Search Step	Optimal Value
MultiSimilarity	margin λ	[0,1]	0.1	0.8
	scale α	(0,100]	2	18
	scale β	(0,100]	2	76
	epsilon	[0,1]	0.1	0.4
SoftTriple	margin λ	[0,1]	0.1	0.4
	scale α	(0,100]	2	78
	weights lr	[0.00001, 0.0001]	0.00001	0.00005
	gamma	(0,100]	10	58
	tau	[0,1]	0.1	0.4
ProxyAnchor	margin λ	[0,1]	0.1	0.1
	scale α	(0,100]	2	32

Our Setup

Fixed Validation

Loss Function	R@1
MultiSimilarity	65.61
SoftTriple	66.12
ProxyAnchor	66.56

Our Fixed Validation R@1 scores

Loss Function	R@1
MultiSimilarity	65.24
SoftTriple	66.76
ProxyAnchor	68.11

Our Default Setup R@1 scores

Loss Function	R@1
MultiSimilarity	65.40
SoftTriple	65.40
ProxyAnchor	68.40

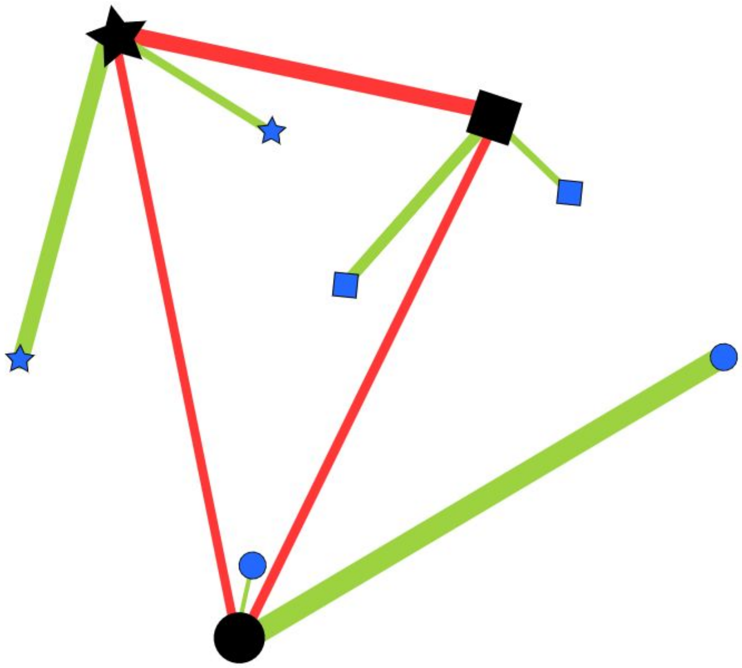
Authors Default Setup R@1 scores

- ProxyAnchor is the only out of 3 that had already optimal hyperparameters
- MultiSimilarity and SoftTriple **slightly improve** their performance
- **Not expected:** training is done using 90 classes
- **Speculation:** authors **avoid** to conduct extensive finetuning - they know finetuning on test set is **not** a good practice
- Propose Fixed Validation as the **new default** setup of Deep Metric Learning

OUR METHOD

Definition, Formulation, Visualization, Results

Our Method



- Different shapes→different classes
- Black nodes→proxies
- Blue nodes→samples
- Green edges→positive associations
- Red edges→negative associations

- **Thickness** of edges is **analogous** to **gradients**
- Gradients are determined by **relative hardness:**
    - Positives: the farther the greater
    - Negatives: the closer the greater

Our Method

- Assign **one** proxy to **each** class
- **Samples** of the batch are associated with **positive** proxies
- **Proxies** themselves are treated as **negatives** that should be **pushed** away
- Two different **variations**
- The second one utilizes a **trick** in order to exploit more **data-to-data** relations: the similarity **between proxies** is computed by taking into consideration the **samples of the batch** too

$$\mathcal{L}_{\text{OurLoss}_1} = \frac{1}{|W+|} \sum_{w \in W+} \log\left(1 + \sum_{x \in X_w^+} e^{-\alpha(w^T x - \lambda)}\right) + \frac{1}{|W|} \sum_{w \in W} \log\left(1 + \sum_{w^- \in W-} e^{\alpha(w^T w^- + \lambda)}\right),$$

$$\mathcal{L}_{\text{OurLoss}_2} = \frac{1}{|W+|} \sum_{w \in W+} \log\left(1 + \sum_{x \in X_w^+} e^{-\alpha(w^T x - \lambda)}\right) + \frac{1}{|W|} \sum_{w \in W} \log\left(1 + \sum_{w^- \in W-} e^{\alpha\left(\sum_{x \in X}(w^T x)(x^T w^-) + \lambda\right)}\right)$$

where $\lambda > 0$ is a margin, $\alpha > 0$ is a scaling factor, $W = W^+ + W^-$ indicates the set of all proxies, $X = X_w^+ + X_w^-$ indicates the batch of embedding vectors and w^- a negative proxy to w

Our Method

- Experiments using the second variation of OurLoss and the BNInception with a 512-dimensional embadding on all 3 datasets

	R@1	R@2	R@4	R@8
Contrastive	63.28	74.51	82.83	89.50
Triplet	61.98	73.59	83.80	88.87
LiftedStructure	64.28	75.47	83.91	89.89
NPair	59.90	71.98	80.47	87.25
ProxyNCA	63.84	74.02	82.98	89.54
Margin	63.48	75.86	83.90	89.78
ArcFace	62.36	73.48	81.67	88.08
MultiSimilarity	65.24	75.76	84.69	90.48
SoftTriple	66.76	77.09	85.36	**91.21**
OurLoss	65.42	75.89	84.99	90.52
ProxyAnchor	**68.11**	**78.63**	**85.77**	91.12

CUB200-2011

	R@1	R@2	R@4	R@8
Contrastive	79.09	86.36	91.69	95.06
Triplet	77.02	84.12	89.79	93.56
LiftedStructure	79.82	86.79	91.86	94.92
NPair	73.25	81.86	86.58	90.45
ProxyNCA	81.02	86.97	92.47	95.12
Margin	81.98	87.75	91.75	94.85
ArcFace	79.42	86.77	91.71	94.70
MultiSimilarity	83.75	89.84	93.75	96.53
SoftTriple	85.29	91.10	**94.78**	**97.10**
OurLoss	84.12	90.12	94.00	96.97
ProxyAnchor	**86.21**	**91.71**	94.70	96.95

CARS196

	R@1	R@10	R@100	R@1000
Contrastive	77.10	89.01	95.23	97.85
Triplet	73.85	84.93	90.11	92.45
LiftedStructure	77.14	89.62	95.73	98.75
NPair	71.45	82.88	87.99	90.58
ProxyNCA	76.15	88.02	93.14	95.47
Margin	76.54	87.98	92.51	94.89
ArcFace	74.91	85.29	90.81	93.39
MultiSimilarity	77.73	89.88	95.77	98.69
SoftTriple	79.29	**90.70**	95.85	98.53
OurLoss	77.92	90.01	95.89	98.99
ProxyAnchor	**79.42**	90.66	**96.05**	**98.62**

SOP

Conclusions

- Success of CNNs: Metric Learning→**Deep Metric Learning**
- **Issues** related to Deep Metric Learning: unfair comparisons, lack of validation
- Conduct extensive **experiments**→draw important **conclusions** about:
  - Loss Functions
  - Networks
  - Embeddings
  - Datasets
  - Setup
- Propose:
  - **Fixed Validation** as the new **default setup** of Deep Metric Learning
- Introduce:
  - New **loss function** that is in between classification and embedding ones and its performance is almost on a par with the state-of-the-art

Future Work

- Extensive experiments using our **Fixed Validation** setup
- Redesign our loss to capture even more **data-to-data** relations
- Experiment with ideas like **offline mining** for batch construction, **memory, multiple proxies** per class

Thank you!


OCÉANO
ATLÁNTICO
CORRIENTE ECUATORIAL DEL NORTE
Mar de los Sargazos
CUENCA ATLÁNTICA DEL NOROESTE
CORRIENTE DEL GOLFO
Is. Bermudas (R.U.)
CUENCA DE LA GUAYANA
CORRIENTE DE LAS CANARIAS
OCCIDENTALES
BAHAMAS
INDIAS
MARANHÃO
PIAUÍ
CEARÁ