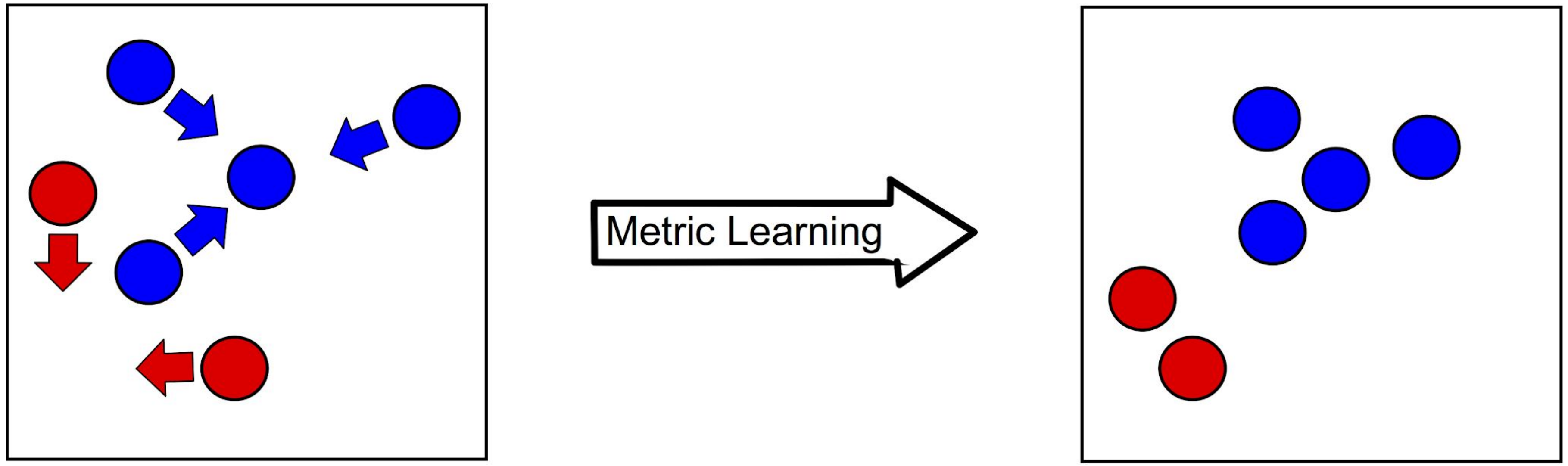
A Metric Learning Reality Check

Dmitry Gradoboev, 171.

Metric Learning



Embedding losses

Contrastive loss: $L_{contrastive} = [d_p - m_{pos}]_+ + [m_{neg} - d_n]_+$

Triplet margin loss: $L_{triplet} = [d_{ap} - d_{an} + m]_+$

Modifications: **The angular loss** is the triplet loss where the margin is based on the angles formed by the triplet vectors.

The margin loss is the contrastive loss where $m_{pos} = \beta - \alpha$ and $m_{neg} = \beta + \alpha$, where α is fixed, and β is learnable.

The lifted structure loss is the contrastive loss but with LogSumExp applied to all negative pairs.

The N-Pairs loss applies the softmax function to each positive pair relative to all other pairs.

The tuplet margin loss combines LogSumExp with an implicit pair weighting method.

The circle loss weights each pair’s similarity by its deviation from a pre-determined optimal similarity value.

FastAP attempts to optimize for average precision within each batch, using a soft histogram binning technique.

Classification losses

Classification losses are based on the inclusion of a weight matrix, where each column corresponds to a particular class. In most cases, training consists of matrix multiplying the weights with embedding vectors to obtain logits, and then applying a loss function to the logits.

The normalized softmax loss, which is identical to cross entropy, but with the columns of the weight matrix L2 normalized.

ProxyNCA is a variation of this, where cross entropy is applied to the Euclidean distances, rather than the cosine similarities, between embeddings and the weight matrix.

SphereFace, **CosFace**, and **ArcFace** apply multiplicative-angular, additive-cosine, and additive-angular margins in the softmax expression, respectively.

The SoftTriple loss takes a different approach, by expanding the weight matrix to have multiple columns per class, theoretically providing more flexibility for modeling class variances.

Pair and triplet mining

Mining is the process of finding the best pairs or triplets to train on.

There are two broad approaches to mining: **offline** and **online**.

Offline mining is performed before batch construction, so that each batch is made to contain the most informative samples.

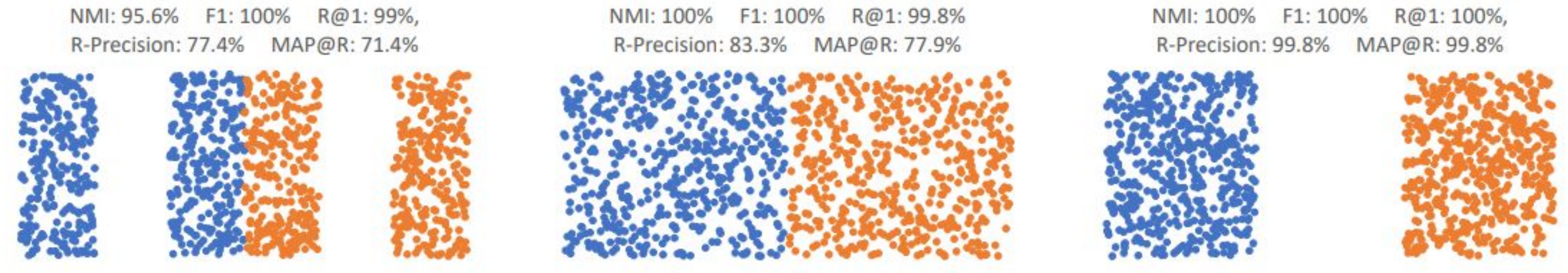
In contrast, **online mining** finds hard pairs or triplets within each randomly sampled batch.

Unfair comparisons

1. Network architecture.
2. Dimensionality of the embedding space.
3. Image augmentations.
4. Choice of optimizer.
5. BatchNorm parameters frozen during training.
6. Confidence intervals.

Weakness of commonly used accuracy metrics

To report accuracy, most metric learning papers use Recall@K, Normalized Mutual Information (NMI), and the F1 score.



Training with test set feedback

Training set - 50% of classes

Test set - remainder 50% of classes

There is no validation set, and model selection and hyperparameter tuning are done with direct feedback from the test set.

Training with test set feedback leads to overfitting on the test set.

Fair comparisons and reproducibility

All experiments are run using PyTorch with the following settings:

– The trunk model is **an ImageNet pretrained BN-Inception network**, with output embedding size of 128. BatchNorm parameters are frozen during training, to reduce overfitting.

– **The batch size** is set to 32. Batches are constructed by first randomly sampling C classes, and then randomly sampling M images for each of the C classes. We set C = 8 and M = 4 for embedding losses, and C = 32 and M = 1 for classification losses.

– During training, images are augmented using **the random resized cropping strategy**. Specifically, we first resize each image so that its shorter side has length 256, then make a random crop that has a size between 40 and 256, and aspect ratio between 3/4 and 4/3. This crop is then resized to 227x227, and flipped horizontally with 50% probability. During evaluation, images are resized to 256 and then center cropped to 227.

– All network parameters are optimized using **RMSprop with learning rate 1e-6**. We chose RMSprop because it converges faster than SGD, and seems to generalize better than Adam, based on a small set of experiments. For loss functions that include their own learnable weights (e.g. ArcFace), we use RMSprop but leave the learning rate as a hyperparameter to be optimized.

– Embeddings are **L2 normalized** before computing the loss, and during evaluation.

Informative accuracy metrics

R-Precision is defined as follows: For each query, let R be the total number of references that are the same class as the query. Find the R nearest references to the query, and let r be the number of those nearest references that are the same class as the query. The score for the query is (r / R).

Mean Average Precision for a single query:

$$MAP@R = \frac{1}{R}\sum_{i=1}^{R} P(i)$$

$$P(i) = \begin{cases} \text{precision at } i, & \text{if the ith retrieval is correct} \\ 0, & \text{otherwise} \end{cases}$$

Retrieval results	Recall@1	R-Precision	MAP@R
10 results, of which only the 1st is correct	100	10	10
10 results, of which the 1st and 10th are correct	100	20	12
10 results, of which the 1st and 2nd are correct	100	20	20
10 results, of which all 10 are correct	100	100	100

Hyperparameter search via cross validation

To find the best loss function hyperparameters, we run 50 iterations of bayesian optimization. Each iteration consists of 4-fold cross validation:

– **The first half of classes are used for cross validation**, and the 4 partitions are created deterministically: the first 0-12.5% of classes make up the first partition, the next 12.5-25% of classes make up the second partition, and so on.

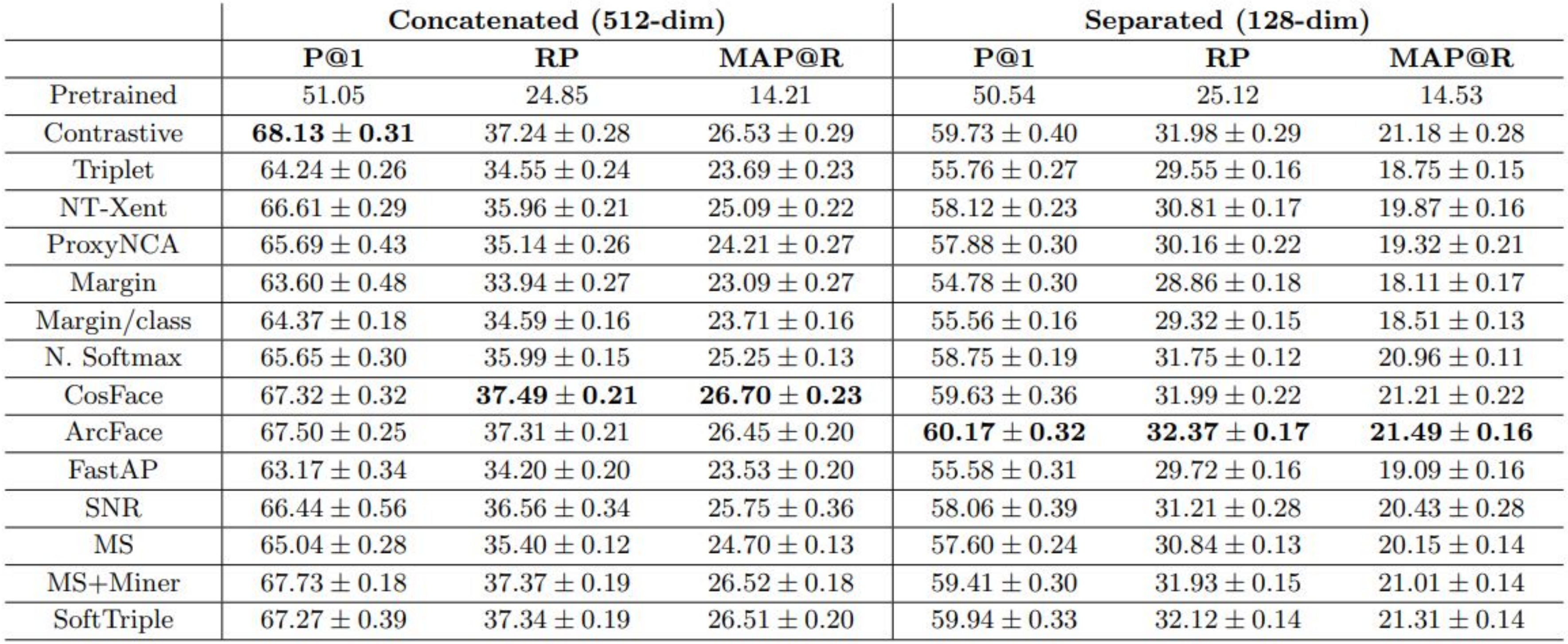
– **The second half of classes are used as the test set**. This is the same setting that metric learning papers have used for years, and we use it so that results can be compared more easily to past papers.

Compute accuracy using two methods:

1. **Concatenated (512-dim)**: For each sample in the test set, we concatenate the 128-dim embeddings of the 4 models to get 512-dim embeddings, and then L2 normalize. We then report the accuracy of these embeddings.

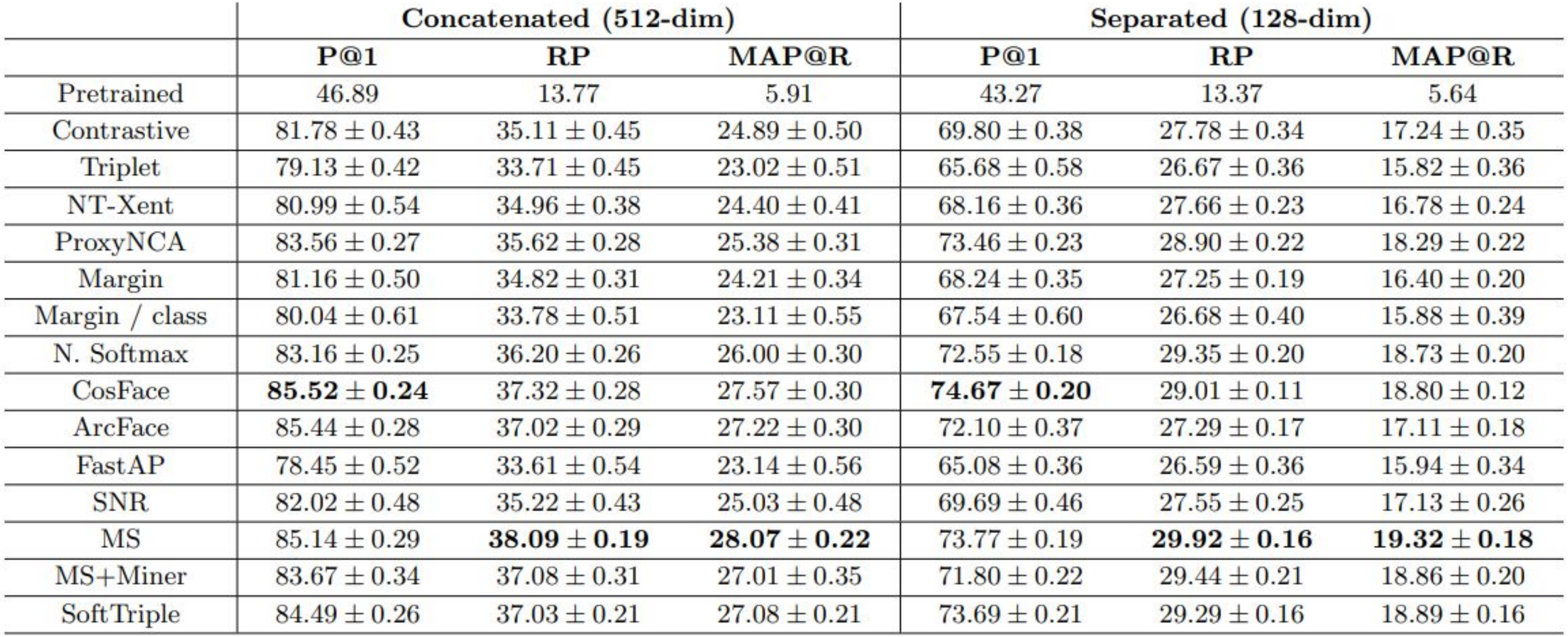
2. **Separated (128-dim)**: For each sample in the test set, we compute the accuracy of the 128-dim embeddings separately, and therefore obtain 4 different accuracies, one for each model's embeddings. We then report the average of these accuracies.

Accuracy on CUB200



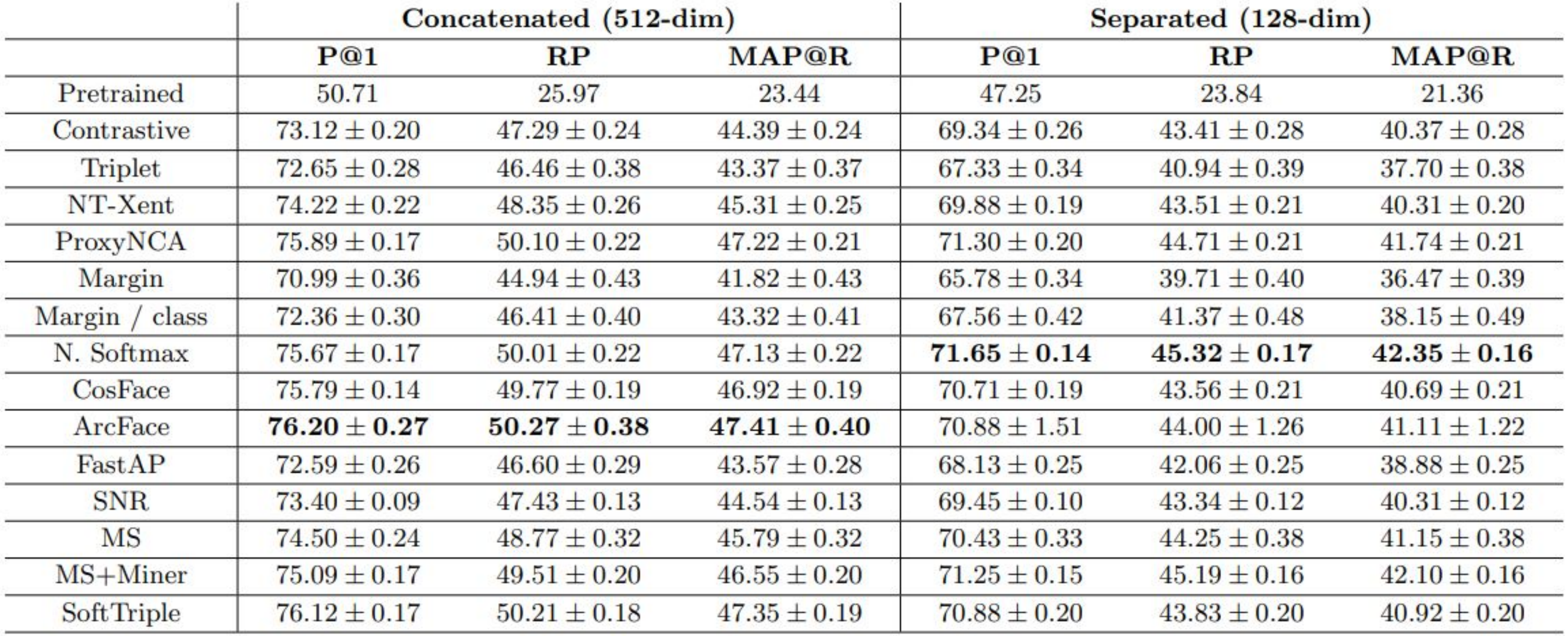
	Concatenated (512-dim)			Separated (128-dim)		
	P@1	**RP**	**MAP@R**	**P@1**	**RP**	**MAP@R**
Pretrained	51.05	24.85	14.21	50.54	25.12	14.53
Contrastive	**68.13 ± 0.31**	37.24 ± 0.28	26.53 ± 0.29	59.73 ± 0.40	31.98 ± 0.29	21.18 ± 0.28
Triplet	64.24 ± 0.26	34.55 ± 0.24	23.69 ± 0.23	55.76 ± 0.27	29.55 ± 0.16	18.75 ± 0.15
NT-Xent	66.61 ± 0.29	35.96 ± 0.21	25.09 ± 0.22	58.12 ± 0.23	30.81 ± 0.17	19.87 ± 0.16
ProxyNCA	65.69 ± 0.43	35.14 ± 0.26	24.21 ± 0.27	57.88 ± 0.30	30.16 ± 0.22	19.32 ± 0.21
Margin	63.60 ± 0.48	33.94 ± 0.27	23.09 ± 0.27	54.78 ± 0.30	28.86 ± 0.18	18.11 ± 0.17
Margin/class	64.37 ± 0.18	34.59 ± 0.16	23.71 ± 0.16	55.56 ± 0.16	29.32 ± 0.15	18.51 ± 0.13
N. Softmax	65.65 ± 0.30	35.99 ± 0.15	25.25 ± 0.13	58.75 ± 0.19	31.75 ± 0.12	20.96 ± 0.11
CosFace	67.32 ± 0.32	**37.49 ± 0.21**	**26.70 ± 0.23**	59.63 ± 0.36	31.99 ± 0.22	21.21 ± 0.22
ArcFace	67.50 ± 0.25	37.31 ± 0.21	26.45 ± 0.20	**60.17 ± 0.32**	**32.37 ± 0.17**	**21.49 ± 0.16**
FastAP	63.17 ± 0.34	34.20 ± 0.20	23.53 ± 0.20	55.58 ± 0.31	29.72 ± 0.16	19.09 ± 0.16
SNR	66.44 ± 0.56	36.56 ± 0.34	25.75 ± 0.36	58.06 ± 0.39	31.21 ± 0.28	20.43 ± 0.28
MS	65.04 ± 0.28	35.40 ± 0.12	24.70 ± 0.13	57.60 ± 0.24	30.84 ± 0.13	20.15 ± 0.14
MS+Miner	67.73 ± 0.18	37.37 ± 0.19	26.52 ± 0.18	59.41 ± 0.30	31.93 ± 0.15	21.01 ± 0.14
SoftTriple	67.27 ± 0.39	37.34 ± 0.19	26.51 ± 0.20	59.94 ± 0.33	32.12 ± 0.14	21.31 ± 0.14

Accuracy on Cars196



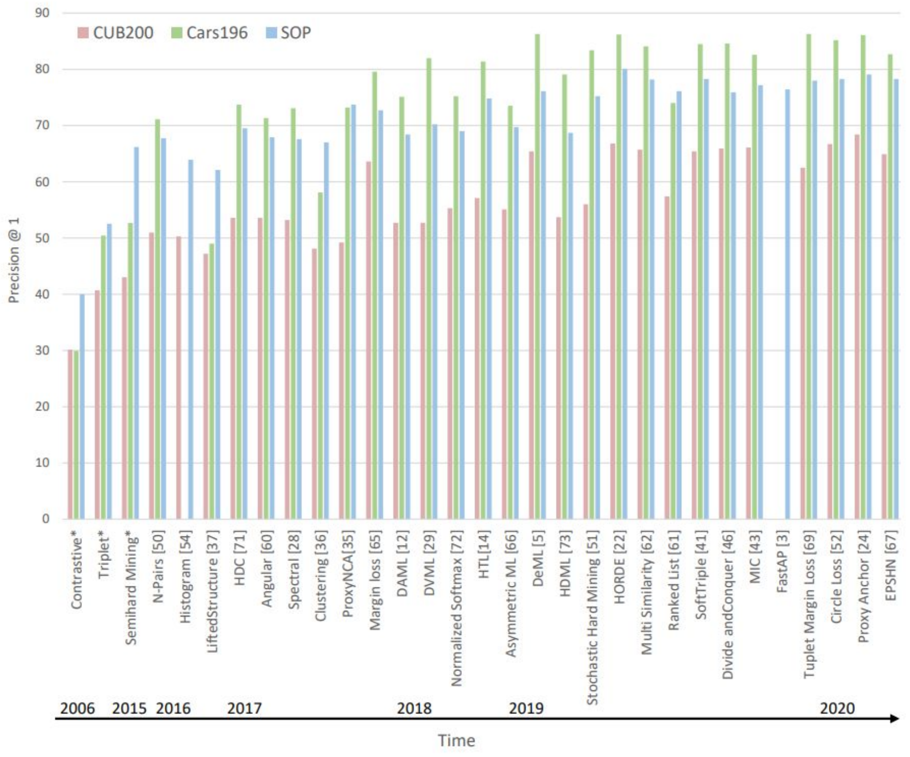
	Concatenated (512-dim)			Separated (128-dim)		
	P@1	**RP**	**MAP@R**	**P@1**	**RP**	**MAP@R**
Pretrained	46.89	13.77	5.91	43.27	13.37	5.64
Contrastive	81.78 ± 0.43	35.11 ± 0.45	24.89 ± 0.50	69.80 ± 0.38	27.78 ± 0.34	17.24 ± 0.35
Triplet	79.13 ± 0.42	33.71 ± 0.45	23.02 ± 0.51	65.68 ± 0.58	26.67 ± 0.36	15.82 ± 0.36
NT-Xent	80.99 ± 0.54	34.96 ± 0.38	24.40 ± 0.41	68.16 ± 0.36	27.66 ± 0.23	16.78 ± 0.24
ProxyNCA	83.56 ± 0.27	35.62 ± 0.28	25.38 ± 0.31	73.46 ± 0.23	28.90 ± 0.22	18.29 ± 0.22
Margin	81.16 ± 0.50	34.82 ± 0.31	24.21 ± 0.34	68.24 ± 0.35	27.25 ± 0.19	16.40 ± 0.20
Margin / class	80.04 ± 0.61	33.78 ± 0.51	23.11 ± 0.55	67.54 ± 0.60	26.68 ± 0.40	15.88 ± 0.39
N. Softmax	83.16 ± 0.25	36.20 ± 0.26	26.00 ± 0.30	72.55 ± 0.18	29.35 ± 0.20	18.73 ± 0.20
CosFace	$\mathbf{85.52 \pm 0.24}$	37.32 ± 0.28	27.57 ± 0.30	$\mathbf{74.67 \pm 0.20}$	29.01 ± 0.11	18.80 ± 0.12
ArcFace	85.44 ± 0.28	37.02 ± 0.29	27.22 ± 0.30	72.10 ± 0.37	27.29 ± 0.17	17.11 ± 0.18
FastAP	78.45 ± 0.52	33.61 ± 0.54	23.14 ± 0.56	65.08 ± 0.36	26.59 ± 0.36	15.94 ± 0.34
SNR	82.02 ± 0.48	35.22 ± 0.43	25.03 ± 0.48	69.69 ± 0.46	27.55 ± 0.25	17.13 ± 0.26
MS	85.14 ± 0.29	$\mathbf{38.09 \pm 0.19}$	$\mathbf{28.07 \pm 0.22}$	73.77 ± 0.19	$\mathbf{29.92 \pm 0.16}$	$\mathbf{19.32 \pm 0.18}$
MS+Miner	83.67 ± 0.34	37.08 ± 0.31	27.01 ± 0.35	71.80 ± 0.22	29.44 ± 0.21	18.86 ± 0.20
SoftTriple	84.49 ± 0.26	37.03 ± 0.21	27.08 ± 0.21	73.69 ± 0.21	29.29 ± 0.16	18.89 ± 0.16

Accuracy on SOP

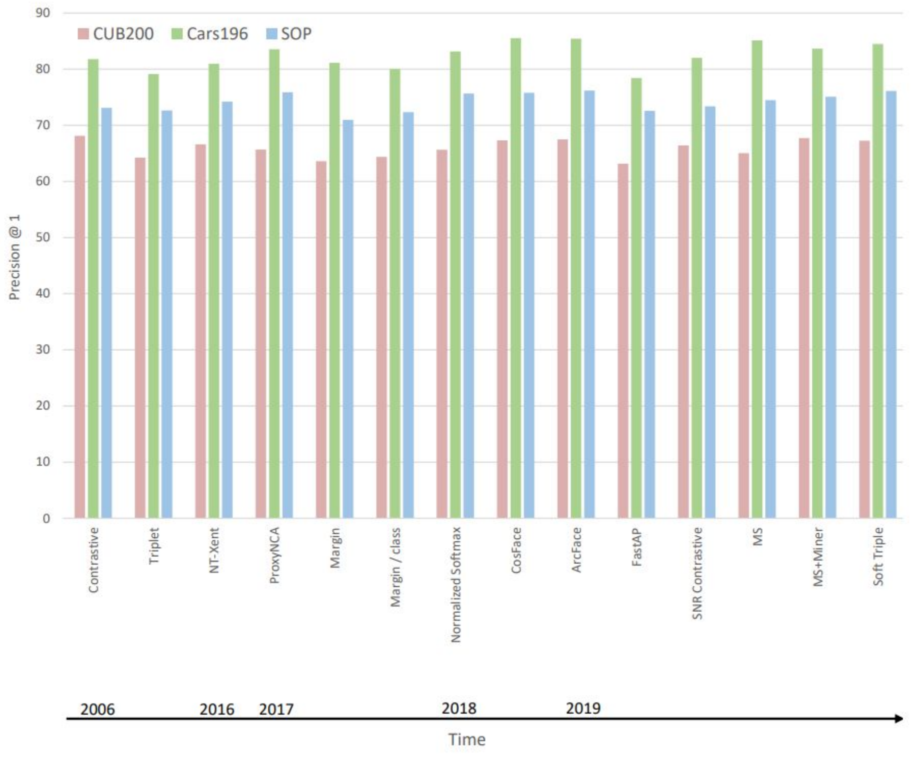


	Concatenated (512-dim)			Separated (128-dim)		
	P@1	**RP**	**MAP@R**	**P@1**	**RP**	**MAP@R**
Pretrained	50.71	25.97	23.44	47.25	23.84	21.36
Contrastive	73.12 ± 0.20	47.29 ± 0.24	44.39 ± 0.24	69.34 ± 0.26	43.41 ± 0.28	40.37 ± 0.28
Triplet	72.65 ± 0.28	46.46 ± 0.38	43.37 ± 0.37	67.33 ± 0.34	40.94 ± 0.39	37.70 ± 0.38
NT-Xent	74.22 ± 0.22	48.35 ± 0.26	45.31 ± 0.25	69.88 ± 0.19	43.51 ± 0.21	40.31 ± 0.20
ProxyNCA	75.89 ± 0.17	50.10 ± 0.22	47.22 ± 0.21	71.30 ± 0.20	44.71 ± 0.21	41.74 ± 0.21
Margin	70.99 ± 0.36	44.94 ± 0.43	41.82 ± 0.43	65.78 ± 0.34	39.71 ± 0.40	36.47 ± 0.39
Margin / class	72.36 ± 0.30	46.41 ± 0.40	43.32 ± 0.41	67.56 ± 0.42	41.37 ± 0.48	38.15 ± 0.49
N. Softmax	75.67 ± 0.17	50.01 ± 0.22	47.13 ± 0.22	$\mathbf{71.65 \pm 0.14}$	$\mathbf{45.32 \pm 0.17}$	$\mathbf{42.35 \pm 0.16}$
CosFace	75.79 ± 0.14	49.77 ± 0.19	46.92 ± 0.19	70.71 ± 0.19	43.56 ± 0.21	40.69 ± 0.21
ArcFace	$\mathbf{76.20 \pm 0.27}$	$\mathbf{50.27 \pm 0.38}$	$\mathbf{47.41 \pm 0.40}$	70.88 ± 1.51	44.00 ± 1.26	41.11 ± 1.22
FastAP	72.59 ± 0.26	46.60 ± 0.29	43.57 ± 0.28	68.13 ± 0.25	42.06 ± 0.25	38.88 ± 0.25
SNR	73.40 ± 0.09	47.43 ± 0.13	44.54 ± 0.13	69.45 ± 0.10	43.34 ± 0.12	40.31 ± 0.12
MS	74.50 ± 0.24	48.77 ± 0.32	45.79 ± 0.32	70.43 ± 0.33	44.25 ± 0.38	41.15 ± 0.38
MS+Miner	75.09 ± 0.17	49.51 ± 0.20	46.55 ± 0.20	71.25 ± 0.15	45.19 ± 0.16	42.10 ± 0.16
SoftTriple	76.12 ± 0.17	50.21 ± 0.18	47.35 ± 0.19	70.88 ± 0.20	43.83 ± 0.20	40.92 ± 0.20

Papers versus reality

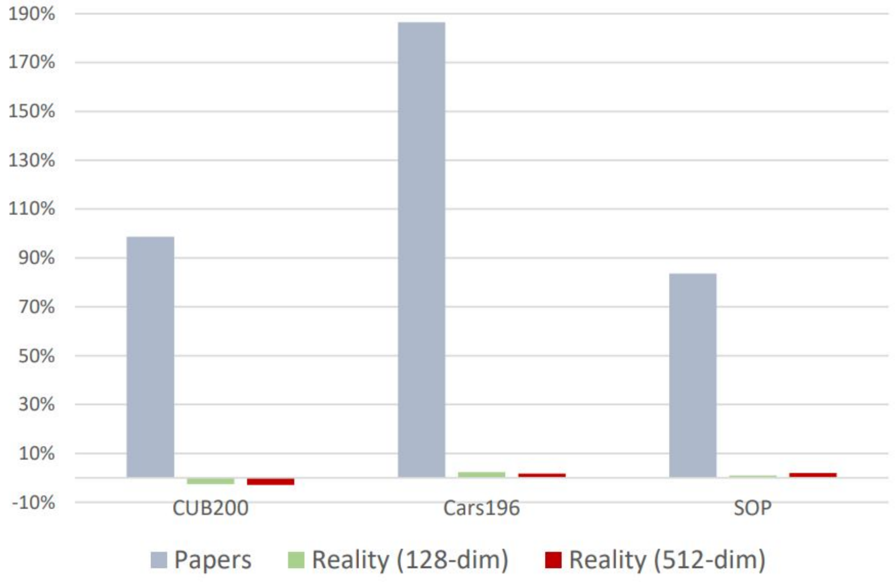


(a) The trend according to papers

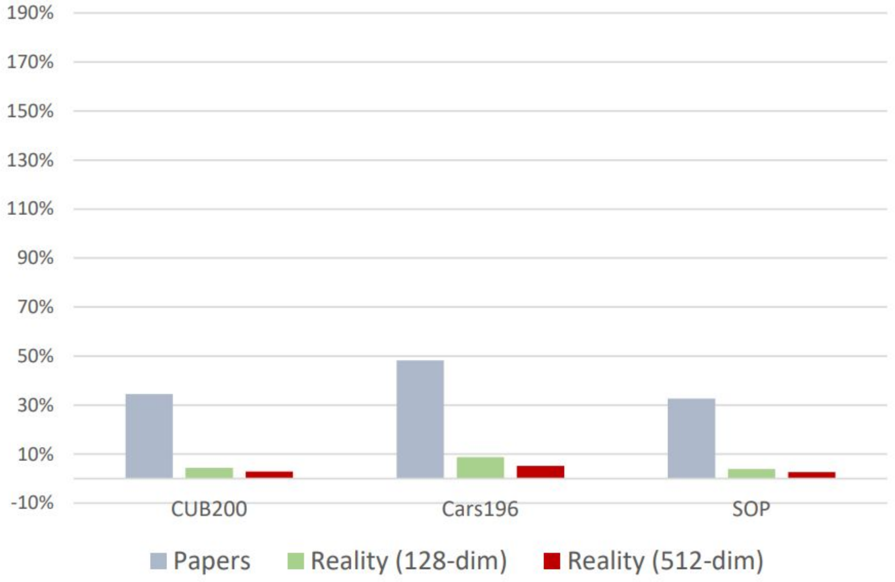


(b) The trend according to reality

Papers versus reality



(a) Relative improvement over the contrastive loss



(b) Relative improvement over the triplet loss

Conclusion

In this paper, authors uncovered several flaws in the current metric learning literature, namely:

– Unfair comparisons caused by changes in network architecture, embedding size, image augmentation method, and optimizers.

– The use of accuracy metrics that are either misleading, or do not a provide a complete picture of the embedding space.

– Training without a validation set, i.e. with test set feedback.

Questions

1. What problem do metric learning methods solve? What types of loss functions are used in these methods? Give some examples of modifications of these functions.
2. What is the mining for pair and triplet loss functions? What are the two broad approaches to mining?
3. What flaws were made in the author's opinion, which violates the objective evaluation of the methods in comparison with the previous algorithms?

References

https://arxiv.org/abs/2003.08505 - K.Musgrave, S.Belongie, S.Lim. A Metric Learning Reality Check. 2020