Progressive Learning Algorithm for Efficient Person Re-Identification

Zhen Li[1,2] , Hanyang Shao[1], Liang Niu[3], Nian Xue[3,*]

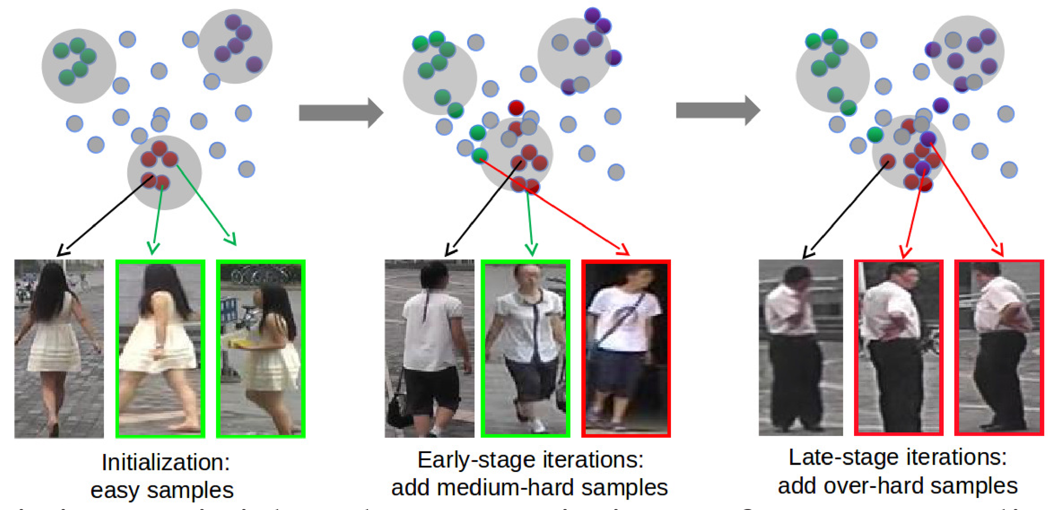
Shanghai Grandhonor Information Technology[1]; Nanjing University of Aeronautics and Astronautic[2]; New York University[3]

Problem Definition and Contribution

Goal: Develop a novel learning strategy to find efficient feature embeddings while maintaining the balance of accuracy and model complexity.

Motivations:

- Existing triplet loss methods select only the hard identity examples will confuse the model weights and might result in failure in associating the easy identity examples to the triplet anchor.



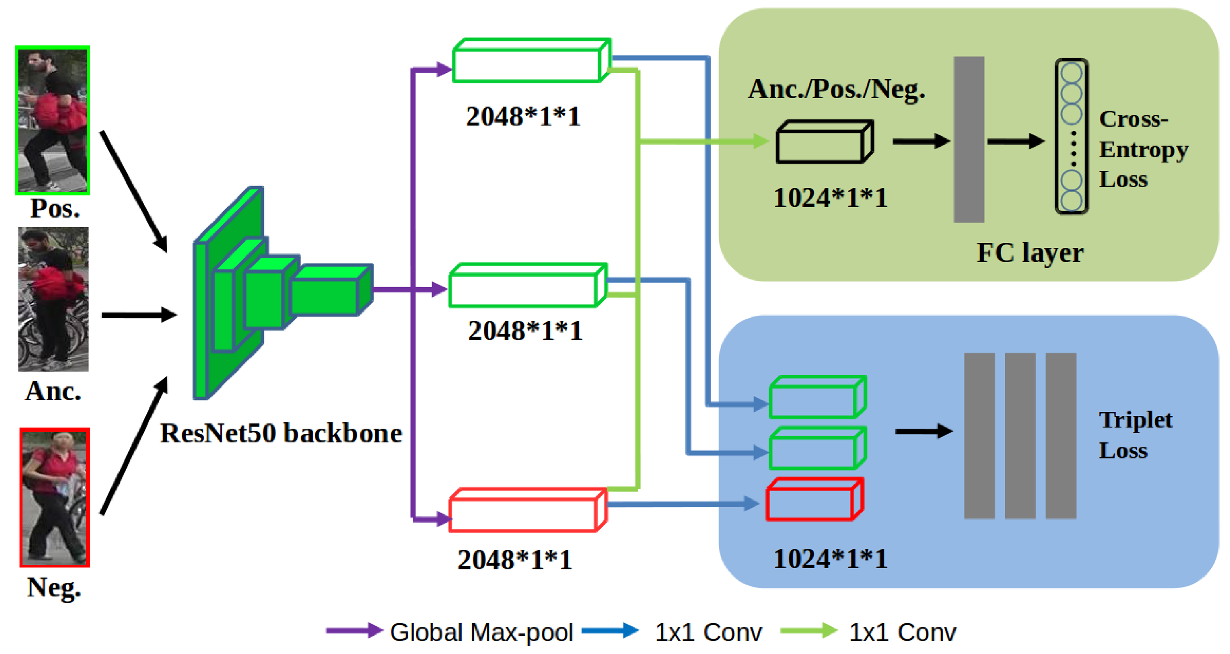
- Training mini-batches consisting of easy, medium-hard and over-hard examples. A red box indicates a person that is different from the anchor sample, while a green box indicates the same person. The hard examples are valuable to train a high precision model but may lead to confusion in the optimization.

Key Contributions:

- A novel learning approach is proposed to find efficient feature embeddings while maintaining the balance of accuracy and model complexity.
- A novel method is developed to explore the hard examples and build a discriminant feature embedding yet compact enough for large-scale applications.
- A novel Bayesian approach is used to progressively learn the triplet loss from simple to hard samples.

Framework

Overview of the proposed ReID network architecture. "Anc.", "Pos." and "Neg." represent anchor image, positive images that belong to the same identity and negative images that belong to different identities, respectively.



Algorithm

- **Input**: A fixed-size mini-batch consisting of $P = 16$ randomly selected identities and $K = 8$ randomly selected images per identity from the training set.
- **Output**: The optimal hyperparameter $\boldsymbol{w}^*$ along with the well trained CNN.
- **Initialization**: Randomly initialize N sets of hyperparameters $\mathbb{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_N\}$ where $\boldsymbol{w}_i = (\lambda_i, m_i, k_i, p_i)$, $\lambda_i \in [0, 2]$, $m_i \in [-0.1, 0.3]$, $k_i \in [1, 8]$, $p_i \in [1, 16]$ for $i = 1, \cdots, N$.
- **Repeat**

  for each hyperparameter $i = 1$ to N **do**

  Exploration: Backpropagate CNN in 20 epochs and evaluate the loss $\mathcal{L}$ according to Eq. 1 and Eq. 2, and evaluate the Bayesian objective $f(\boldsymbol{w}_i)$.

  Restoration: CNN weights are restored to that before 20 epochs of exploration.

  end for

  Exploitation: Based on $f(\mathbb{W})$, obtain a new improved candidate $\boldsymbol{w}'$ and update Gaussian process according to Eq. 3 and Eq. 4, and add $\boldsymbol{w}'$ to $\mathbb{W}$.

  Backpropagate to update CNN weights for 300 epochs based on the new hyperparameter $\hat{\boldsymbol{w}}$ and the feed-forward loss $\mathcal{L}$;

  Save the model with lowest loss $\mathcal{L}$ for the current hyperparameter $\hat{\boldsymbol{w}}$;
- **Until** maximum epochs ($M = 3,000$) reached

Equations

Loss function:

$$\mathcal{L}_{GBH}^{k,p}(\boldsymbol{\theta}; X) = \sum_{l=1}^{P} \sum_{\substack{a,b \\ y_a = y_b = l}} \ln\left(1 + e^{m + T_{k,p}(a,b,n)}\right) \quad (1)$$

$$\mathcal{L}^{k,p}(\boldsymbol{\theta}; X) = \mathcal{L}_{softmax}(\boldsymbol{\theta}; X) + \lambda \mathcal{L}_{GBH}^{k,p}(\boldsymbol{\theta}; X) \quad (2)$$

.

$$\begin{cases} \tilde{f}(\hat{\boldsymbol{w}}) \sim \mathcal{GP}(\mu(\hat{\boldsymbol{w}}) + \triangle\mu, \mathcal{K}(\hat{\boldsymbol{w}}) - \triangle\mathcal{K}) \\ \triangle\mu = \mathcal{K}(\hat{\boldsymbol{w}}, \mathbb{W}) \mathcal{K}(\mathbb{W})^{-1} (f(\mathbb{W}) - \mu(\mathbb{W})) \\ \triangle\mathcal{K} = \mathcal{K}(\hat{\boldsymbol{w}}, \mathbb{W}) \mathcal{K}(\mathbb{W})^{-1} \mathcal{K}(\mathbb{W}, \hat{\boldsymbol{w}}) \end{cases} \quad (3)$$

$$\mathcal{EI}(\hat{\boldsymbol{w}}) = (\mathcal{K}(\hat{\boldsymbol{w}}) - \triangle\mathcal{K})^{\frac{1}{2}} (Z\Phi(Z) + \phi(Z)) \quad (4)$$

. Bayesian optimization objective function:

$$f(\boldsymbol{w}) = \left| \frac{\bar{\mathcal{L}}_t^{k,p}(\boldsymbol{\theta}; \boldsymbol{w}; X) - \bar{\mathcal{L}}_{t'}^{k,p}(\boldsymbol{\theta}; \boldsymbol{w}; X)}{\bar{\mathcal{L}}_t^{k,p}(\boldsymbol{\theta}; \boldsymbol{w}; X)} - \mathcal{ED} \right| \quad (5)$$

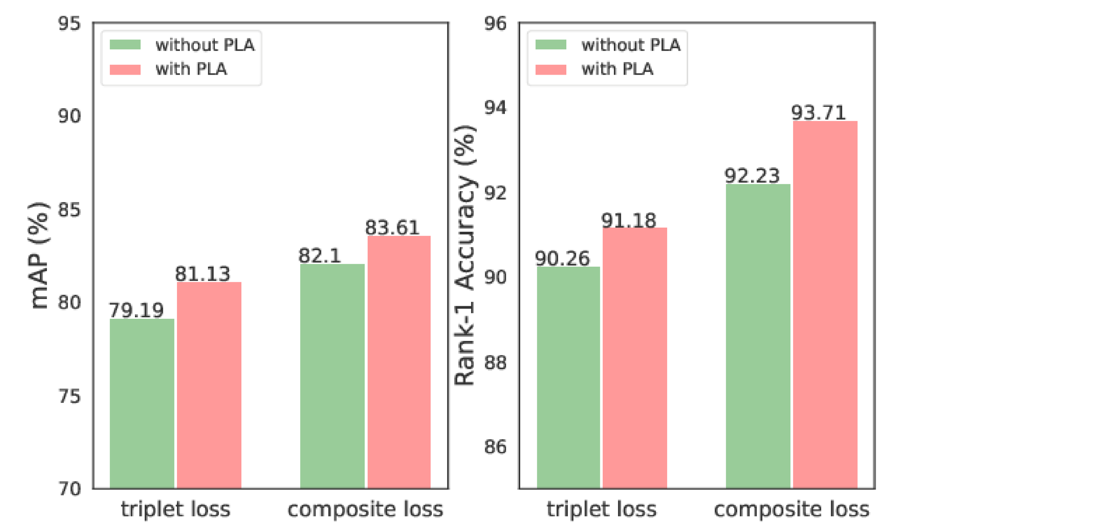
Experiments & Results

Dataset:

ReID Benchmark datasets used in our experiments.

Dataset	Market1501	DukeMTMC	CUHK03(D/L)
Identities	1,501	1,812	1,360
Bboxes	32,668	36,411	13,164
Camera	6	8	6
Train images	12,936	16,522	7,365/7,368
Train ids	751	702	767
Query images	3,368	2,228	1,400
Query ids	750	702	700
Gallery images	19,732	17,661	5,332

PLA consistently improves both triplet loss and composite loss on Market-1501 dataset.

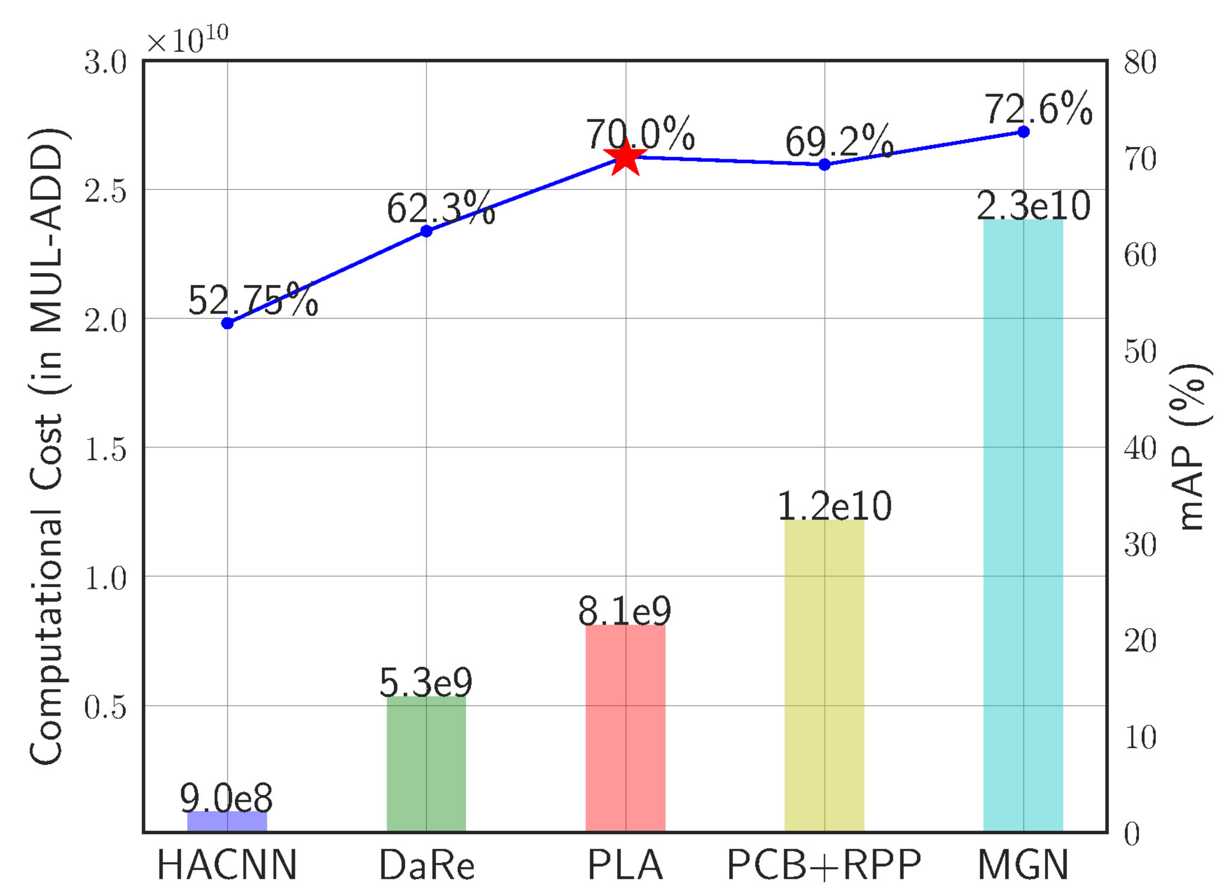


Precision Test Results:

Comparing PLA with different global and part models on all datasets. "RK" stands for reranking.

Category	Methods	Market1501(SQ)		Market1501(MQ)		CUHK03(D)		CUHK03(L)		DukeMTMC	
		mAP	Rank-1	mAP	Rank-1	mAP	Rank-1	mAP	Rank-1	mAP	Rank-1
part	HA-CNN	75.7	91.2	82.8	93.8	38.6	41.7	41.0	44.4	63.8	80.5
	Deep-Person	79.6	92.3	85.1	94.5	-	-	-	-	64.8	80.9
	PCB	77.4	92.3	-	-	54.2	61.3	-	-	66.1	81.7
	PCB+RPP	81.6	93.8	-	-	57.5	63.7	-	-	69.2	83.3
	Aligned-ReID	82.3	92.6	-	-	-	-	-	-	-	-
	MGN	**86.9**	**95.7**	**90.7**	**96.9**	**66.0**	**66.8**	**67.4**	**68.0**	**78.4**	**88.7**
global	SVDNet	62.1	82.3	-	-	37.2	41.5	37.8	40.9	56.8	76.7
	TriNet	69.1	84.9	76.4	90.5	-	-	-	-	-	-
	GP-reid	81.2	92.2	82.8	93.8	-	-	-	-	**72.8**	**85.2**
	DaRe	74.2	88.5	-	-	58.1	61.6	60.2	64.5	63.0	79.1
	PLA	**83.6**	**93.7**	**88.4**	**95.2**	**63.2**	**67.2**	**67.5**	**71.5**	72.5	84.3
RK	Trinet	81.1	86.7	87.2	91.8	-	-	-	-	-	-
	DaRe	85.9	90.8	-	-	71.2	69.8	73.7	72.9	79.6	84.4
	MGN	**94.2**	**96.6**	**95.9**	**97.1**	-	-	-	-	-	-
	PLA	89.4	94.7	92.9	95.7	**77.2**	**75.5**	**81.0**	**79.6**	**80.1**	**87.0**

Accuracy vs. computation cost (number of Mul-Add):



Accuracy vs. inference memory (MB):