Long-Tail Hashing: Model, Optimization and Application

Yong Chen[1,2], Yuqing Hou[3], Shu Leng[4], Qing Zhang[3], Zhouchen Lin[1,2,*], Dell Zhang[5,6]

[1]Key Lab. Of Machine Perception (MoE), School of EECS, Peking University, Beijing, China

[2]Pazhou Lab, Guangzhou, China [3]Meituan, Beijing, China

[4]Department of Automation, Tsinghua University, Beijing, China

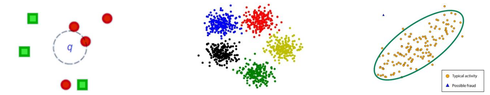
[5]Blue Prism AI Labs, London, UK [6]Birkbeck, University of London, UK

Abstract

Hashing, which represents data items as compact binary codes, has been becoming a more and more popular technique, e.g., for large-scale image retrieval, owing to its super fast search speed as well as its extremely economical memory consumption. However, existing hashing methods all try to learn binary codes from artificially balanced datasets which are not commonly available in real-world scenarios. In this paper, we propose Long-Tail Hashing Network (LTHNet), a novel two-stage deep hashing approach that addresses the problem of learning to hash for more realistic datasets where the data labels roughly exhibit a long-tail distribution. Specifically, the first stage is to learn relaxed embeddings of the given dataset with its long-tail characteristic taken into account via an end-to-end deep neural network; the second stage is to binarize those obtained embeddings. A critical part of LTHNet is its dynamic meta-embedding module extended with a determinantal point process which can adaptively realize visual knowledge transfer between head and tail classes, and thus enrich image representations for hashing. Our experiments have shown that LTHNet achieves dramatic performance improvements over all state-of-the-art competitors on long-tail datasets, with no or little sacrifice on balanced datasets. Further analyses reveal that while to our surprise directly manipulating class weights in the loss function has little effect, the extended dynamic meta-embedding module, the usage of cross-entropy loss instead of square loss, and the relatively small batch-size for training all contribute to LTHNet's success.

Introduction

1. Nearest Neighbor Search (NNS) is underlying many Machine Learning, Data Mining, Information Retrieval problems.



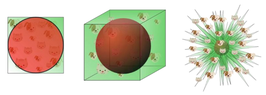
K-NN Classifier & IR | K-Means Clustering | Anomaly Detection

2. When NNS meets Big Data?

(1) Curse of dimensionality;

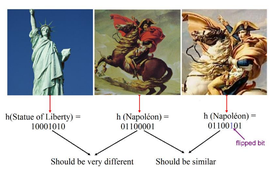
(2) Storage cost;

(3) Query speed;

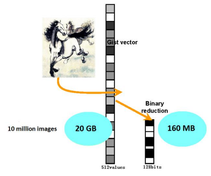


As the dimensionality increases, a larger percentage of the training data resides in the corners of the feature space.

3. Learning to Hash: A Memory-Economical and Computation-Fast Representation Learning.



(1) Similarity Preserved Hashing

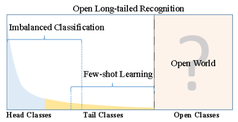


(2) Reduce Dimensionality and Storage Cost

(3) By using hash-code to construct index, we can achieve constant or sub-linear search time complexity.

Motivations & Challenges

1. The Visual World is Long-Tail Distributed.

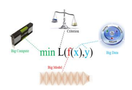


A dataset X is called a long-tail dataset if the sizes of its sorted classes follow the Zipf's law, i.e.,

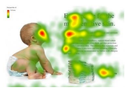
$$s_i = s_1 \times i^{-\mu},$$

where μ is a parameter controlling the degree of data imbalance that is measured by the imbalance factor (IF for short) s_1/s_C.

2. There appears Diverse Technologies and Theories.



Deep Learning: Representation and Generalization



Attention is All your need: Memory Mechanism

Transfer Learning: Across head/tail Classes, Domains

3. Challenges.

Is there any Unified paradigm to Achieve good representations?

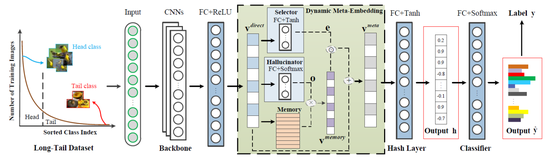
Preliminary Explorations

1. Long-Tail Hashing.

Given a long-tail dataset $X = \{(x_i, l_i)\}(i=1,\cdots,N)$, the problem of long-tail hashing is to learn a set of hash functions $\{h_j(\cdot)\}$ based on it so that

$$H(x_n) = [h_1(x_n), \ldots, h_q(x_n)]^T = b_n \in \{-1,+1\}^q.$$

2. Solution: Long-Tail Hashing Network.



3. Mathematics: Dynamic Meta-Embedding & Classifier & Loss Function.

Dymamic Memory:

$$\mathcal{M} \triangleq \{\mathbf{m}_j\}_{j=1}^{(k+1)C} = \cup_{i=1}^{C}(\{\mathbf{c}(i)\} \cup \mathrm{DPP}_k(i)),$$

Meta-Embedding:

$$\mathbf{v}^{meta} = \mathbf{v}^{direct} + \mathbf{e} \odot \mathbf{v}^{memory},$$

LTHNet: Inputs & Outputs:

$$[\mathbf{v}^{direct}, \mathbf{h}, \hat{\mathbf{y}}] = \mathrm{LTHNet}(\mathbf{x}; \mathcal{M}, \theta),$$

Generalized Class-balanced Loss:

$$\mathrm{L}_{CB}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{E_{n_\mathbf{y}}}\mathrm{L}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1-\beta}{1-\beta^{n_\mathbf{y}}}\mathrm{L}(\hat{\mathbf{y}}, \mathbf{y}),$$

4. Learning Algorithm.

Algorithm 1: Long-Tail Hashing Network (LTHNet)

/* A deep neural network for learning to hash from long-tail data */

Input: the training dataset $\mathfrak{X} = \{(x_n, l_n)\}_{n=1}^{N}$, the number of classes C, the maximum number of epochs *MaxEpoch*, and the hyperparameter β and k;

Initialize LTHNet parameters θ;

while *not MaxEpoch* **do**

  /* Memory: update $\mathcal{M} = \{\mathbf{m}_j\}_{j=1}^{(k+1)C}$ */

  for $n = 1$ *to* N **do**

    $[\mathbf{v}_n^{direct}, \sim, \sim] = \mathrm{LTHNet}(\mathbf{x}_n; \mathcal{M}, \theta)$;

  end

  Compute the centroid for each class via Eq. (3) and retrieve k-more diverse and similar samples for each centroid via Eq. (4), and the memory is updated as $\mathcal{M} = \{\mathbf{m}_j\}_{j=1}^{(k+1)C}$;

  /* LTHNet training: update θ */

  for x *in Dataloader*($\mathfrak{X}$) **do**

    $[\sim, \hat{\mathbf{y}}] = \mathrm{LTHNet}(\mathbf{x}; \mathcal{M}, \theta)$;

    $\mathrm{L}_{CB}(\hat{\mathbf{y}}, \mathbf{y})$;

    $\theta = \mathrm{RMSprop}(\mathrm{L}_{CB}, \theta)$;

  end

end

/* Out-of-samples (x_{oos}) Hashing */

$[\sim, b_{OOS}, \sim] = \mathrm{LTHNet}(x_{OOS}; \mathcal{M}, \theta)$;

$b_{OOS} = \mathrm{sgn}(b_{OOS})$;

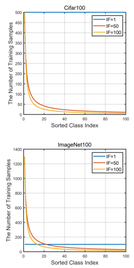
--- do while
Memory: Update M
LTHNet training \theta
--- end
Out-of-Sample Hashing

Experiments & Results

1. Long-Tail Datasets.

Statistics of long-tail benchmarks with various IFs.

Cifar100	μ	$n_{\max}$	$n_{\min}$	n_{db}	n_{query}	n_{train}
IF=1	0.000	500	500	50k	10,000	50,000
IF=50	0.830	500	10	50k	10,000	3,732
IF=100	0.990	500	5	50k	10,000	2,598
ImageNet100	μ	$n_{\max}$	$n_{\min}$	n_{db}	n_{query}	n_{train}
IF=1	0.000	100	100	130k	5,000	10,000
IF=50	0.845	1300	26	130k	5,000	9,437
IF=100	0.990	1300	13	130k	5,000	6,834



2. Results on Cifar100 & ImageNet100.

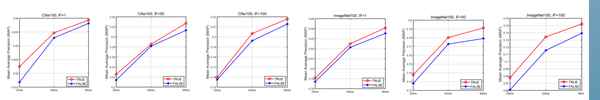
Cifar100

Settings /MAP/ Methods	IF=1			IF=50			IF=100		
	32 bits	64 bits	96 bits	32 bits	64 bits	96 bits	32 bits	64 bits	96 bits
LSH	0.0333	0.0458	0.0608	0.0333	0.0467	0.0615	0.0307	0.0480	0.0585
PCAH	0.0569	0.0612	0.0605	0.0532	0.0617	0.0627	0.0519	0.0608	0.0625
ITQ	0.0806	0.0976	0.1026	0.0709	0.0858	0.0903	0.0677	0.0824	0.0896
KNNH	0.0844	0.1012	0.1088	0.0703	0.0840	0.0906	0.0689	0.0810	0.0871
SDH	0.1950	0.2472	0.2739	0.1115	0.1363	0.1460	0.1006	0.1182	0.1258
COSDISH	0.1262	0.1921	0.2143	0.0695	0.0875	0.1000	0.0583	0.0724	0.0809
FastHash	0.2239	0.3161	0.3636	0.0787	0.1061	0.1211	0.0714	0.0903	0.1001
FSSH	0.1849	0.2207	0.2671	0.1101	0.1384	0.1512	0.0957	0.1146	0.1274
SCDH	0.2415	0.3003	0.3316	0.1282	0.1536	0.1661	0.1138	0.1335	0.1415
DPSH	0.3113	0.4506	0.4957	0.1069	0.1407	0.1634	0.0978	0.1216	0.1383
HashNet	0.4380	0.5719	0.6311	0.1726	0.1950	0.2079	0.1444	0.1559	0.1631
DSDH	0.5398	0.6100	0.6407	0.1119	0.1000	0.0999	0.0940	0.0872	0.0807
CSQ	0.7711	0.7984	0.7821	0.2221	0.2745	0.2669	0.1716	0.1992	0.1658
LTHNet_{sig} (k=0)	0.8191	0.8321	0.8362	0.2220	0.2144	0.2330	0.1624	0.1546	0.1508
LTHNet_{sig} (k=3)	0.8232	0.8360	0.8390	0.2432	0.2794	0.3116	0.1750	0.2079	0.2264
LTHNet (k=0)	0.8195	0.8336	0.8400	0.2427	0.3028	0.3309	0.1752	0.2240	0.2415
LTHNet (k=3)	**0.8268**	**0.8416**	**0.8490**	**0.2687**	**0.3354**	**0.3484**	**0.1819**	**0.2376**	**0.2620**

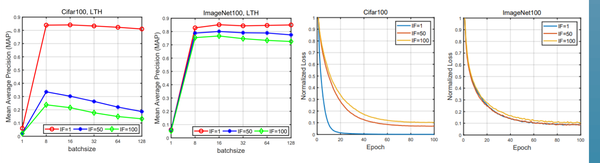
ImageNet100

Settings /MAP/ Methods	IF=1			IF=50			IF=100		
	32 bits	64 bits	96 bits	32 bits	64 bits	96 bits	32 bits	64 bits	96 bits
LSH	0.0613	0.1066	0.1557	0.0606	0.1121	0.1475	0.0556	0.1097	0.1510
PCAH	0.1478	0.2004	0.2042	0.1306	0.1817	0.1919	0.1280	0.1788	0.1947
ITQ	0.1965	0.2687	0.2907	0.1803	0.2458	0.2731	0.1719	0.2371	0.2667
KNNH	0.1996	0.2778	0.3007	0.1830	0.2537	0.2798	0.1766	0.2411	0.2666
SDH	0.4416	0.5108	0.5385	0.3553	0.4188	0.4414	0.3126	0.3733	0.3975
COSDISH	0.2875	0.4040	0.4559	0.2072	0.2900	0.3320	0.1763	0.2395	0.2731
FastHash	0.3178	0.4295	0.4744	0.2462	0.3274	0.3741	0.1932	0.2703	0.3100
FSSH	0.4746	0.5184	0.5528	0.3681	0.4533	0.4702	0.3312	0.4017	0.4314
SCDH	0.4894	0.5598	0.5854	0.3937	0.4726	0.4954	0.3601	0.4194	0.4422
DPSH	0.4887	0.6055	0.6514	0.2186	0.3125	0.3791	0.1788	0.2832	0.3468
HashNet	0.4410	0.6006	0.6421	0.3465	0.4034	0.4240	0.3101	0.3770	0.3800
DSDH	0.6554	0.7015	0.7231	0.2568	0.2617	0.2744	0.1841	0.2134	0.2429
CSQ	**0.8507**	**0.8733**	**0.8657**	0.6629	0.7022	0.6823	0.5989	0.5620	0.5495
LTHNet_{sig} (k=0)	0.7338	0.7713	0.8130	0.5523	0.5154	0.5530	0.3924	0.3490	0.4132
LTHNet_{sig} (k=3)	0.7896	0.8259	0.8465	0.7133	0.7491	0.7753	0.6333	0.6587	0.6958
LTHNet (k=0)	0.7924	0.8267	0.8382	0.7369	0.7804	0.7920	0.6771	0.7350	0.7528
LTHNet (k=3)	0.8142	0.8453	0.8592	**0.7612**	**0.8007**	**0.8157**	**0.7146**	**0.7665**	**0.7828**

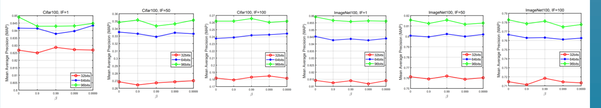
3. Dynamic Meta-Embedding: TRUE or FALSE?



4. How Batch-size affects the MAP results? & Convergence Curves.

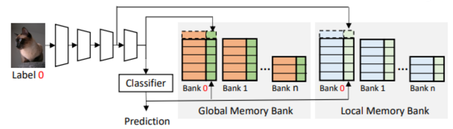


5. How β (class re-weight) affects the MAP results?



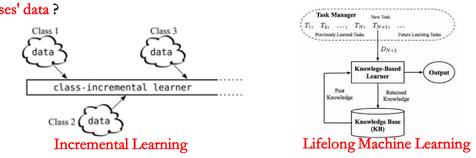
Take-Away Messages

(1) Deep Learning + Memory Mechanism;



(2) Comprehensive Tricks: Model + Loss + Training;

(3) What if a retrieval system consistently encounters Accumulated data including New Classes' data ?



References (Excerpts)

1. Learning to Hash.

(1) Li Yuan, Tao Wang, Xiaopeng Zhang, Francis E. H. Tay, Zequn Jie, Wei Liu, Jiashi Feng: *Central Similarity Quantization for Efficient Image and Video Retrieval*. CVPR 2020: 3080-3089.

(2) Qi Li, Zhenan Sun, Ran He, Tieniu Tan: *A General Framework for Deep Supervised Discrete Hashing*. Int. J. Comput. Vis. 128(8): 2204-2222 (2020).

(3) Yong Chen, Zhibao Tian, Hui Zhang, Jun Wang, Dell Zhang: *Strongly Constrain-ed Discrete Hashing*. IEEE Trans. Image Process. 29: 3596-3611 (2020).

2. Long-Tailed Visual Recognition.

(4) Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, Stella X. Yu: *Large-Scale Long-Tailed Recognition in an Open World*. CVPR 2019: 2537-2546.

(5) Boyan Zhou, Quan Cui, Xiu-Shen Wei, Zhao-Min Chen: *BBN: Bilateral-Branch Network With Cumulative Learning for Long-Tailed Visual Recognition*. CVPR 2020: 9716-9725.

(6) Linchao Zhu, Yi Yang: *Inflated Episodic Memory With Region Self-Attention for Long-Tailed Visual Recognition*. CVPR 2020: 4343-4352.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Zhouchen Lin is supported by the Key-Area Research and Development Program of Guangdong Province (Grant No. 2019B121204008), National Natural Science Foundation of China (Grant No. 61625301 & 61731018), Major Scientific Research Project of Zhejiang Lab (Grant No. 2019KB0AC01 & 2019KB0AB02), Beijing Academy of Artificial Intelligence, and Qualcomm. .