Deep Discrete Hashing for Label Distribution Learning

Zhen Zhang, Lei Zhu, *Senior Member, IEEE*, Yaping Li, and Yang Xu

***Abstract*—Label Distribution Learning (LDL) can better describe the real-world data by learning a set of label distributions instead of discrete binary labels. Particularly, hashing-based LDL has achieved promising performance due to its desirable advantages of fast similarity computation and extremely low storage cost. However, existing hashing-based LDL methods are still shallow learning methods, which cannot deeply capture the implicit data semantics, and meanwhile fail to fully model the semantic data relations. In this letter, we propose an effective and efficient Deep Discrete Hashing for Label Distribution Learning (DDH-LDL) method, which develops the first deep hashing framework for LDL. Specifically, DDH-LDL captures implicit semantic information by multi-layer non-linear transformation, and simultaneously preserves the modeled semantic relations of instances into hash codes via semantic message aggregation on Graph Convolutional Network (GCN). Furthermore, we elaborately design a discrete optimization module that is seamlessly integrated into our proposed deep hashing framework to reduce the binary quantization errors. Experiments on several widely tested datasets verify the superiority of the proposed method on both learning accuracy and efficiency.**

***Index Terms*—Label distribution learning, deep hashing, discrete optimization.**

I. Introduction

DIFFERENT from multi-label learning [1]–[5], Label Distribution Learning (LDL) [6] is a new label learning paradigm, which assumes the labels of the instances are composed of a group of real-value distributions. Thus, the labels obtained by LDL can represent the data more accurately and can better characterize the data form of the real world.

Although LDL has desirable advantages [7], most existing methods suffer from an important problem that the training time increases rapidly with the increase of training data. To alleviate this problem, significant efforts have been made in the literature. Notably, Adaptation Algorithm of k-Nearest Neighbor (AA-kNN) is proposed in [6] to predict the label distributions by nearest neighbor search. Hence, AA-kNN is not scalable since it relies heavily on the quality and the dimensions of the instance features. Recently, hashing technology [8]–[11] has been developed with the advantage of high similarity computation efficiency and extremely low storage cost. Since hashing technology is suitable for large-scale LDL, several hashing-based LDL methods have been proposed, such as Binary Coding based Label Distribution Learning (BC-LDL) [12] and Discrete Binary Coding based Label Distribution Learning (DBC-LDL) [13]. They basically follow the three-step learning paradigm: 1) Project the instance features into the Hamming space. 2) Search the k most similar instances in the database. 3) Calculate the label distribution of the query instance according to the search results. It has been reported that BC-LDL [12] and DBC-LDL [13] have obtained promising label learning results and significant efficiency improvements. However, existing hashing-based LDL methods still achieve sub-optimal performance due to two problems: 1) They are shallow learning methods that conduct the learning process on hand-crafted features and thus cannot deeply capture the implicit data semantics. 2) They simply construct a similarity matrix with labels to preserve the semantic relations of instances, which cannot fully model the intrinsic semantic relations of instances.

To solve the above problems, in this letter, we propose an effective and efficient deep hashing-based LDL method, dubbed as Deep Discrete Hashing for LDL (DDH-LDL). Specifically, our learning framework captures implicit data semantics via multiple layers of nonlinear feature transformation. Simultaneously, we capture the intrinsic semantic relations of instances with message aggregation on Graph Convolutional Network (GCN) [14]–[17], as accurately modeling data semantic relations is important for similarity search accuracy and thus the quality of prediction labels. Moreover, we elaborately design a discrete optimization module [18] in our learning framework to seamlessly preserve the extracted semantics into binary hash codes by minimizing binary quantization errors. The overall architecture of our model is demonstrated in Fig. 1. The main contributions of this work are summarized as follows:

- Different from existing hashing-based LDL methods on shallow models, we develop a deep hashing-based LDL model, which deeply captures the implicit data semantics and simultaneously models the semantic relations of instances. To the best of our knowledge, there is still no similar work.
- We propose a discrete optimization module in the deep hashing framework for LDL, which effectively reduces the

Manuscript received January 25, 2022; revised March 4, 2022; accepted March 4, 2022. Date of publication March 10, 2022; date of current version March 30, 2022. This work was supported in part by the National Natural Science Foundation of China under Grants 62172263 and 62002209, in part by the Natural Science Foundation of Shandong, China, under Grants ZR2020YQ47, ZR2020QF111, and ZR2019QF002, and in part by the Youth Innovation Project of Shandong Universities, China, under Grant 2019KJN040. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jianxin Li. *(Corresponding author: Yang Xu.)*

Zhen Zhang, Lei Zhu, and Yang Xu are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China (e-mail: zhangz2333@163.com; leizhu0608@gmail.com; zzmylq@gmail.com).

Yaping Li is with the School of Information Engineering, Shandong Management University, Jinan 250357, China (e-mail: 69154334@qq.com).

Digital Object Identifier 10.1109/LSP.2022.3158229

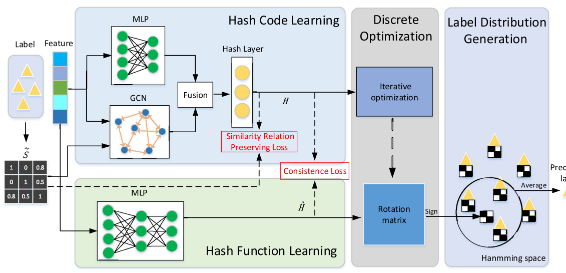

Fig. 1. The basic learning framework of the proposed method. The figure is better viewed with pdf magnification.

quantization errors, improves the quality of hash codes, and further improves the search performance and the accuracy of predicted labels.

II Related Work

1) Label Distribution Learning: LDL aims to learn label distributions instead of discrete binary labels to describe the instances. Improved Iterative Scaling-Learning from Label Distribution (IIS-LLD) [19] is the first method for LDL, which learns label distributions by maximizing a likelihood of a maximum entropy model. Conditional Probability Neural Network (CPNN) [19] exploits a three-layer neural network to learn the label distributions. Label Distribution Support Vector Regressor (LDSVR) [20] is proposed to fit a sigmoid function to each module of the label distribution simultaneously by a multi-output support vector machine. The concept of label distribution learning is formally proposed in [6] and six LDL methods are proposed in [6]. Particularly, Adaptation Algorithm of k-Nearest Neighbor (AA-kNN) [6] first finds the k most similar samples of the query sample in the training set on feature space, then the mean of the label distributions of the k similar samples is calculated as the label distribution of the query sample. AA-kNN is performed without the training process and it improves the learning scalability.

2) Hashing for Label Distribution Learning: Hashing [8] aims to learn a set of hash functions to project instance features into the Hamming space and measure the similarities of instances through Hamming distance. For the detailed introduction about hashing, please refer to the survey [8].

In order to accelerate the similarity search process of AA-kNN mentioned above, hashing-based methods, including Binary Coding based Label Distribution Learning (BC-LDL) [12] and Discrete Binary Coding based Label Distribution Learning (DBC-LDL) [13] are proposed to first transform the original features into the Hamming space, then the label distribution of unknown instance is predicted according to top-k similar instances in Hamming space. BC-LDL is the first method that introduces hashing to LDL, which is supervised by a pair-wise semantic matrix at the stage of hash code learning and adopts a sequential learning strategy to learn the hash codes and functions. Further, DBC-LDL learns hash codes and functions by joint label and pair-wise semantic supervision, and adopts an iterative strategy to optimize the objective model.

Different from the shallow methods BC-LDL and DBC-LDL, in this letter, we propose a new deep hashing-based LDL method to fully capture the implicit data semantics and simultaneously model the semantic correlations of instances. To the best of our knowledge, there is still no similar work.

III. The Proposed Method

Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ denote the dataset, where n represents the number of instances and d is the feature dimension of each instance. Let $\boldsymbol{B} \in \{-1,+1\}^{n\times r}$ denote the hash codes[1], where r is the hash code length. Let $\boldsymbol{D} = [\boldsymbol{d}_1; \boldsymbol{d}_2; \ldots; \boldsymbol{d}_n] \in (0,1)^{n\times c}$ denote the labels, where c is the number of instance categories. $\|\cdot\|_F$ denotes Frobenius norm. $\mathtt{sgn}(\cdot)$ means the sign function which binarizes the input to -1 or 1.

1) Deep Hash Learning: In our method, the hash code learning network mainly contains three modules: Multi-Layer Perception (MLP) [21] module, GCN module, and Fusion module [22], [23]. For the MLP module, we evacuate the implicit identity semantics of instances. The GCN module is designed to capture the intrinsic relations between instances. Then we design a Fusion module to fuse the embedded representations from two modules into a unified one and support the subsequent binary coding.

The structure of the MLP module is formulated as follows:

$$\boldsymbol{F}_{mlp}^{(l)} = \sigma_{mlp}(\boldsymbol{F}_{mlp}^{(l-1)}\boldsymbol{W}_{mlp}^{(l)} + \boldsymbol{z}^{(l)}),\ s.t.\ l \geq 1, \tag{1}$$

where $\boldsymbol{F}_{mlp}^{(l)}$ denotes the output of l-th layer network, σ_{mlp} denotes the activation function, $\boldsymbol{W}_{mlp}$ denotes the parameter matrix, z denotes the bias term, and $\boldsymbol{F}_{mlp}^{(0)}$ is the input $\boldsymbol{X}$.

In the GCN module, we calculate the Cosine similarity of labels as the similarities between instances, that is, $\boldsymbol{S} = \mathtt{Cos}(\boldsymbol{D}, \boldsymbol{D})$. To better adapt to the characteristics of LDL, we only focus on the similar paired instances, and ignore the paired instances with low similarity.

$$\tilde{\boldsymbol{S}} = \begin{cases} 0, & \text{if } \boldsymbol{S}_{ij} < \beta, \\ \boldsymbol{S}_{ij}, & \text{else} \end{cases}, \tag{2}$$

where S_{ij} represents the similarity between the i-th instance and the j-th instance, β is a threshold parameter. The message passing process of the GCN module is formulated as follows:

$$\boldsymbol{F}_{gcn}^{(l)} = \sigma_{gcn}(\tilde{\boldsymbol{S}}\boldsymbol{F}_{gcn}^{(l-1)}\boldsymbol{W}_{gcn}^{(l)}),\ s.t.\ l \geq 1, \tag{3}$$

the definitions of variables in (3) are similar to (1).

The process of our deep hash learning can be expressed as follows:

$$\begin{aligned} \boldsymbol{F}_{mlp} &= \mathtt{MLP}(\boldsymbol{X}; \boldsymbol{\Theta}_{mlp}),\ \boldsymbol{F}_{gcn} = \mathtt{GCN}(\boldsymbol{X}, \tilde{\boldsymbol{S}}; \boldsymbol{\Theta}_{gcn}), \\ \boldsymbol{F} &= \mathtt{Fusion}(\boldsymbol{F}_{mlp}, \boldsymbol{F}_{gcn}; \boldsymbol{\Theta}_{fs}), \\ \boldsymbol{H} &= \mathtt{HashLayer}(\boldsymbol{F}; \boldsymbol{\Theta}_{hl}), \end{aligned} \tag{4}$$

where $\boldsymbol{F}$ denotes the fused features, $\boldsymbol{H}$ denotes the relaxed representations of hash codes, $\boldsymbol{\Theta}_{mlp}$, $\boldsymbol{\Theta}_{gcn}$, $\boldsymbol{\Theta}_{fs}$, $\boldsymbol{\Theta}_{hl}$ denote the parameters of the MLP module, GCN module, Fusion module and Hash Layer, respectively.

In addition, we design a similarity relation preserving loss to preserve the similarities between the instances into the hash

[1]It can be transformed into binary codes by simple operation $(\mathtt{sgn}(\boldsymbol{B})+1)/2$

codes. The hash code learning is performed by minimizing the following loss:

$$\mathcal{L}_{code} = \| \texttt{Cos}(\boldsymbol{H}, \boldsymbol{H}) - \tilde{\boldsymbol{S}} \|_F^2. \tag{5}$$

We take MLP as the backbone of hash function to project the original features into the relaxed binary representations. The hash function can be expressed as $\hat{\boldsymbol{H}} = \texttt{MLP}_H(\boldsymbol{X}; \boldsymbol{\Theta}_{hf})$, where $\boldsymbol{\Theta}_{hf}$ denotes the parameters of the hash function. To make the hash function learning network fit the hash code learning network, we minimize the differences between the outputs, $\hat{\boldsymbol{H}}, \boldsymbol{H}$, of these two networks. The loss function for hash function learning is formulated as follows:

$$\mathcal{L}_{func} = \| \hat{\boldsymbol{H}} - \boldsymbol{H} \|_F^2. \tag{6}$$

Thus, the overall objective function for deep hash learning is formulated as follows:

$$\min_{\substack{\boldsymbol{\Theta}_{mlp}, \boldsymbol{\Theta}_{gcn}, \\ \boldsymbol{\Theta}_{fs}, \boldsymbol{\Theta}_{hl}, \boldsymbol{\Theta}_{hf}}} \mathcal{L} = \mathcal{L}_{code} + \mathcal{L}_{func}. \tag{7}$$

We employ Adam optimizer [24] to optimize the hash code and function learning network. In the test stage, the hash codes can be obtained by binarizing $\hat{\boldsymbol{H}}\boldsymbol{R}$: $\boldsymbol{B} = \texttt{sgn}(\hat{\boldsymbol{H}}\boldsymbol{R})$, where $\boldsymbol{R}$ is a rotating matrix, we will introduce it in the next subsection.

2) Discrete Optimization Module: To reduce the binary quantization loss, we design a discrete optimization module [25], [26] to directly learn binary hash codes $\boldsymbol{B}$. Firstly, we rotate $\boldsymbol{H}$ to make it closer to the vertex of the hypercube without changing the relative paired similarities of $\boldsymbol{H}$. Secondly, we perform the label embedding $\boldsymbol{D}$ as a supervised semantic-guided hash code learning process via a linear mapping, and calculate the optimal hash codes by iterative optimization until convergence. We formulate this module as:

$$\min_{\substack{\boldsymbol{P}, \boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I}, \\ \boldsymbol{B}\in\{-1,1\}^{n\times r}}} \| \boldsymbol{B} - \boldsymbol{H}\boldsymbol{R} \|_F^2 + \alpha \| \boldsymbol{B} - \boldsymbol{D}\boldsymbol{P} \|_F^2, \tag{8}$$

where $\boldsymbol{R}$ is the rotation matrix, and α is the balance parameter, and $\boldsymbol{P}$ is a mapping matrix from label embedding to the hash codes. In this letter, we propose an iterative optimization strategy to solve this optimization problem.

Step P: By fixing $\boldsymbol{R}, \boldsymbol{B}$, update $\boldsymbol{P}$. The formula of optimizing $\boldsymbol{P}$ can be written as:

$$\min_{\boldsymbol{P}} \alpha \texttt{tr}(\boldsymbol{P}^\top \boldsymbol{D}^\top \boldsymbol{D}\boldsymbol{P} - 2\boldsymbol{B}^\top \boldsymbol{D}\boldsymbol{P}), \tag{9}$$

where $\texttt{tr}(\cdot)$ denotes the trace operator. By calculating the derivative of (9) w.r.t $\boldsymbol{P}$ and setting it to $\mathbf{0}$, we can obtain that $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{B}$.

Step R: By fixing $\boldsymbol{P}, \boldsymbol{B}$, update $\boldsymbol{R}$. We first compute the SVD of the matrix $\boldsymbol{B}^\top\boldsymbol{H}$:

$$\boldsymbol{U}, \Omega, \boldsymbol{V} = \texttt{svd}(\boldsymbol{B}^\top\boldsymbol{H}), \tag{10}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ denote left and right singular matrix of $\boldsymbol{B}^\top\boldsymbol{H}$, respectively [27]. Then, we can obtain the solution of $\boldsymbol{R}$ as $\boldsymbol{R} = \boldsymbol{V}^\top\boldsymbol{U}^\top$.

Step B: By fixing $\boldsymbol{P}, \boldsymbol{R}$, the formula for optimizing $\boldsymbol{B}$ can be written as:

$$\min_{\boldsymbol{B}\in\{-1,1\}^{n\times r}} -\texttt{tr}(\boldsymbol{B}^\top(\boldsymbol{H}\boldsymbol{R} + \alpha\boldsymbol{D}\boldsymbol{P})). \tag{11}$$

TABLE I
STATISTICS OF THE EXPERIMENTAL DATASETS

ID	Dataset	Train num	Query num	Feature dim	Label dim
1	Movie [20]	6,980	775	1,869	5
2	RAF-ML [28]	4,418	490	200	6
3	fbp5500 [29]	4,950	550	512	5
4	Flickr-LDL [30]	10,035	1,115	256	8
5	SCUT-FBP [31]	1,350	150	300	5

Then, we can obtain that $\boldsymbol{B} = \texttt{sgn}(\boldsymbol{H}\boldsymbol{R} + \alpha\boldsymbol{D}\boldsymbol{P})$.

3) Label Distribution Generation: For a new query set, we first obtain the query hash codes through hash functions. Then, we retrieve the k most similar instances from the retrieval set in the Hamming space for each instance in the query set. Finally, for each instance in the query set, we take the average value of the label distributions of the k most similar instances as the predicted label distribution. The calculation process of the predicted label distribution is formulated as follows:

$$\boldsymbol{d}_q = \frac{1}{k} \sum \boldsymbol{d}_i, i \in \{1, 2, \ldots, k\}_{nearest}. \tag{12}$$

IV. Experiment

1) Datasets: We conduct our experiments on five widely tested datasets: Movie [20], RAF-ML [28], fbp5500 [29], Flickr-LDL [30], and SCUT-FBP [31]. For each dataset, we randomly choose 90% of the instances as the training set, and set the remaining instances as the query set. The basic statistics of these datasets are summarized in Table I.

2) Compared Baselines: We compare the proposed method with six state-of-the-art methods, including four traditional LDL methods: Improved Iterative Scaling-Learning from Label Distribution (IIS-LLD) [19], Conditional Probability Neural Network (CPNN) [19], Label Distribution Support Vector Regressor (LDSVR) [20] and Adaptation Algorithm of k-Nearest Neighbor (AA-kNN) [6], and two shallow hashing-based LDL methods: Binary Coding based Label Distribution Learning (BC-LDL) [12] and Discrete Binary Coding based Label Distribution Learning (DBC-LDL) [13]. In experiments, we directly use the source codes of IIS-LLD, CPNN, LDSVR and AA-KNN to perform experiments, and carefully reproduce the codes of BC-LDL and DBC-LDL according to the original letters. The parameters of all baselines are set to the values recommended by the authors.

3) Evaluation Metrics: We adopt the same evaluation metrics with the previous works [12], [13] to evaluate the compared baselines. Among them, Chebyshev distance (Cheb), Clark distance (Clark), Canberra metric (Canber), Kullback-Leibler divergence (K-L) are used to measure the distance between two distributions. For these metrics, the smaller value indicates better performance. Cosine coefficient (Cosine) and Intersection similarity (Intersec) measure the similarity between two distributions. For these metrics, the larger value indicates better performance. Furthermore, the ranks of six measure values are also given in the parentheses right after the corresponding measure values. Then we utilize three different rank metrics to evaluate the performance of all methods in our experiments, i.e., Accuracy Rank (Acc. Rank), Time Rank, and Average Rank (Avg. Rank).

TABLE II
EXPERIMENTAL RESULTS ON FIVE DATASETS, THE BEST RESULT IN EACH COLUMN IS MARKED WITH BOLD

	Method	Accuracy							Time Cost (s)			Avg. rank
		Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑	Acc. Rank	Train time	Test time	Sum (Time Rank)	
Movie	BC-LDL [12]	0.1229(3)	0.5865(3)	1.1030(3)	0.1030(3)	0.9366(3)	0.8267(3)	3.0	429.58	0.89	430.47(7)	5.0
	DBC-LDL [13]	0.1326(4)	0.6047(4)	1.1781(4)	0.1423(4)	0.9099(4)	0.8052(4)	4.0	36.18	1.05	37.23(3)	3.5
	IIS-LLD [19]	0.1615(6)	0.6215(5)	1.2010(5)	0.1470(5)	0.8978(6)	0.7904(6)	5.5	308.58	0.03	308.61(6)	5.8
	CPNN [19]	0.1691(7)	0.7060(7)	1.3742(7)	0.2004(7)	0.8688(7)	0.7628(7)	7.0	263.46	0.14	263.60(5)	6.0
	LDSVR [20]	0.1101(2)	0.5117(2)	0.9874(2)	0.0895(2)	0.9427(2)	0.8435(2)	2.0	18.08	0.43	18.51(2)	2.0
	AA-kNN [6]	0.1421(5)	0.6592(6)	1.2732(6)	0.1559(6)	0.9021(5)	0.7906(5)	5.5	0.00	97.28	97.28(4)	4.8
	DDH-LDL	**0.1083(1)**	**0.5005(1)**	**0.9560(1)**	**0.0859(1)**	**0.9449(1)**	**0.8483(1)**	**1.0**	13.76	0.36	**14.12(1)**	**1.0**
	DDH-LDL-I	0.1192	0.5604	1.0730	0.1005	0.9363	0.8299	-	-	-	-	-
	DDH-LDL-II	0.1091	0.5059	0.9685	0.0881	0.9438	0.8467	-	-	-	-	-
	Method	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑	Acc. Rank	Train time	Test time	Sum (Time Rank)	Avg. rank
RAF-ML	BC-LDL [12]	0.2083(3)	1.5226(3)	3.0223(3)	0.2876(3)	0.8879(3)	0.7493(3)	3.0	24.42	0.81	25.23(5)	4.0
	DBC-LDL [13]	0.1928(2)	**1.5181(1)**	3.0126(2)	0.2714(2)	0.8959(2)	0.7650(2)	1.8	8.15	1.16	9.31(3)	2.4
	IIS-LLD [19]	0.3847(7)	1.6485(6)	3.5876(6)	0.7299(7)	0.6629(7)	0.5054(7)	6.7	73.72	0.01	73.73(6)	6.4
	CPNN [19]	0.3604(6)	1.6360(5)	3.5124(5)	0.6546(6)	0.7020(6)	0.5460(6)	5.7	129.07	0.18	129.25(7)	6.4
	LDSVR [20]	0.2470(4)	1.7763(7)	3.8073(7)	0.4842(5)	0.8727(4)	0.7300(4)	5.2	12.42	0.06	12.48(4)	4.6
	AA-kNN [6]	0.2691(5)	1.5454(4)	3.1498(4)	0.4307(4)	0.8171(5)	0.6723(5)	4.5	0.00	6.23	6.23(2)	3.3
	DDH-LDL	**0.1927(1)**	1.5190(2)	**2.9966(1)**	**0.2647(1)**	**0.8980(1)**	**0.7671(1)**	**1.2**	4.86	0.16	**5.02(1)**	**1.1**
	DDH-LDL-I	0.2165	1.5428	3.1034	0.3098	0.8751	0.7361	-	-	-	-	-
	DDH-LDL-II	0.1888	1.5290	3.0302	0.2690	0.8969	0.7691	-	-	-	-	-
	Method	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑	Acc. Rank	Train time	Test time	Sum (Time Rank)	Avg. rank
fbp5500	BC-LDL [12]	0.2123(7)	1.1864(3)	2.0937(4)	0.2287(6)	0.8857(7)	0.7551(6)	5.5	56.59	0.91	57.50(5)	5.3
	DBC-LDL [13]	0.1408(2)	1.2015(4)	2.0204(3)	0.1180(2)	0.9507(2)	0.8448(2)	2.5	10.18	2.11	12.29(3)	2.8
	IIS-LLD [19]	0.2118(6)	1.3315(7)	2.4097(7)	0.2262(5)	0.9086(6)	0.7511(7)	6.3	99.00	0.01	99.01(6)	6.2
	CPNN [19]	0.1614(4)	1.3033(5)	2.2741(5)	0.1406(3)	0.9397(4)	0.8213(5)	4.3	171.42	0.07	171.49(7)	5.7
	LDSVR [20]	0.1646(5)	1.3066(6)	2.3452(6)	0.2476(7)	0.9419(3)	0.8284(3)	5.0	19.13	0.10	19.23(4)	4.5
	AA-kNN [6]	0.1560(3)	**1.0129(1)**	**1.6936(1)**	0.1637(4)	0.9375(5)	0.8258(4)	3.0	0.00	10.97	10.97(2)	2.5
	DDH-LDL	**0.1405(1)**	1.1844(2)	1.9815(2)	**0.1155(1)**	**0.9511(1)**	**0.8450(1)**	**1.3**	9.35	0.19	**9.54(1)**	**1.2**
	DDH-LDL-I	0.1683	1.2570	2.1868	0.1740	0.9248	0.8086	-	-	-	-	-
	DDH-LDL-II	0.1441	1.1942	2.0127	0.1177	0.9490	0.8411	-	-	-	-	-
	Method	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑	Acc. Rank	Train time	Test time	Sum (Time Rank)	Avg. rank
Flickr-LDL	BC-LDL [12]	0.2422(2)	2.1837(3)	5.4009(3)	**0.4803(1)**	**0.8451(1)**	0.6679(2)	2.0	14.60	3.14	17.74(2)	2.0
	DBC-LDL [13]	0.2436(3)	2.2034(4)	5.4843(4)	0.5026(3)	0.8407(3)	0.6676(3)	3.3	27.05	3.32	30.37(4)	3.7
	IIS-LLD [19]	0.3411(5)	2.2122(6)	5.4952(5)	0.7569(4)	0.7338(6)	0.5558(6)	5.3	73.00	0.01	73.01(6)	5.7
	CPNN [19]	0.3545(6)	2.2107(5)	5.5457(6)	0.7698(5)	0.7244(7)	0.5352(7)	6.0	281.07	0.13	281.20(7)	6.5
	LDSVR [20]	0.4064(7)	2.3845(7)	6.3624(7)	1.7603(7)	0.7568(5)	0.5732(5)	6.3	36.87	0.19	37.06(5)	5.7
	AA-kNN [6]	0.2558(4)	**1.8745(1)**	**4.1554(1)**	0.9558(6)	0.8190(4)	0.6590(4)	3.3	0.00	21.43	21.43(3)	3.2
	DDH-LDL	**0.2415(1)**	2.1761(2)	5.3641(2)	0.4916(2)	0.8413(2)	**0.6705(1)**	**1.7**	8.18	0.65	**8.83(1)**	**1.4**
	DDH-LDL-I	0.2554	2.1832	5.3965	0.5207	0.8310	0.6526	-	-	-	-	-
	DDH-LDL-II	0.2415	2.1812	5.3886	0.4899	0.8414	0.6689	-	-	-	-	-
	Method	Cheb↓	Clark↓	Canber↓	K-L↓	Cosine↑	Intersec↑	Acc. Rank	Train time	Test time	Sum (Time Rank)	Avg. rank
SCUT-FBP	BC-LDL [12]	0.2811(5)	1.2571(6)	2.3706(6)	0.4560(3)	0.7672(5)	0.6382(6)	5.2	15.75	0.05	15.80(5)	5.1
	DBC-LDL [13]	0.2447(3)	1.2368(4)	2.2916(4)	0.4671(4)	0.8081(3)	0.6940(3)	3.5	1.23	0.07	1.30(3)	3.3
	IIS-LLD [19]	0.2559(4)	1.2450(5)	2.3624(5)	0.4748(5)	0.7920(4)	0.6732(4)	4.5	68.88	0.01	68.89(6)	5.3
	CPNN [19]	0.2350(2)	1.1942(3)	2.1968(2)	**0.3611**(1)	0.8345(2)	0.7050(2)	2.0	69.6	0.01	69.61(7)	4.5
	LDSVR [20]	0.3870(7)	1.6296(7)	3.2853(7)	1.8460(7)	0.7145(7)	0.5688(7)	7.0	1.31	0.01	1.32(4)	5.5
	AA-kNN [6]	0.2867(6)	1.1924(2)	2.2383(3)	0.7654(6)	0.7525(6)	0.6519(5)	4.7	0.00	0.43	**0.43(1)**	2.9
	DDH-LDL	**0.2222(1)**	**1.1734(1)**	**2.1520(1)**	0.3791(2)	**0.8365(1)**	**0.7277(1)**	**1.2**	0.92	0.01	0.93(2)	**1.6**
	DDH-LDL-I	0.2465	1.2635	2.3692	0.5291	0.7970	0.6912	-	-	-	-	-
	DDH-LDL-II	0.2373	1.2206	2.2652	0.4510	0.8109	0.7011	-	-	-	-	-

4) Implementation Details: Our hash code learning network includes an MLP module and a GCN module. The MLP module consists of three fully connected layers and the GCN module contains one graph convolutional layer. The fusion module consists of two one-layer fully connected layer module, which follows the GCN and MLP modules respectively, the outputs of these two modules are added element-wise as the fused features. The hash function network consists of three fully connected layers. To learn more comprehensive information, we set the batch size in the training process as the size of the training set. Our method adopts the same configuration as BC-LDL [12] and DBC-LDL [13]: set the length of hash codes to 128, k to 30 and the training set as the retrieval set.

5) Experimental Results and Discussions: The comparison results are shown in Table II. In this table, we can find that our method achieves the best accuracy performance on five datasets. That is mainly because our method can deeply capture implicit data semantics, effectively model the semantic relations between instances, and reduce binary quantization errors with discrete optimization. In addition to accuracy performance, our method also achieves better performance on learning efficiency, especially on larger datasets, such as Movie, RAF-ML, fbp5500, and Flickr-LDL. This is because the network we adopted is lightweight and less parameters are required to optimize. Besides, we use a larger batch to train the network directly and further improve the efficiency of our method. Hence, our proposed method can converge rapidly and achieve high efficiency. On the whole, the average ranking of our method is the best on five datasets.

Besides, we design two variants to evaluate the effects of our designed deep architecture. DDH-LDL-I removes the GCN module, which means that only the MLP module is used as the backbone network of hash code learning. DDH-LDL-II removes the discrete optimization module, which means that the output $\hat{\boldsymbol{H}}$ of the hash function is directly binarized without rotating. The experimental results in Table II show that the performance of these variants decreases when we remove GCN or discrete optimization module. The results prove the effects of these modules.

V. CONCLUSION

In this letter, we propose a simple but effective Deep Discrete Hashing for Label Distribution Learning (DDH-LDL) network, which is the first deep hashing framework for label distribution learning. The proposed model captures implicit data semantics with deep networks and seamlessly embeds the extracted semantics into binary hash codes, so that more accurate label distribution can be predicted. Moreover, a discrete optimization module is designed in the deep hashing framework to reduce the binary quantization errors and further improve the performance. Experiments validate the superiority of the proposed approach.

ACKNOWLEDGMENT

Our source codes and testing datasets can be obtained at https://github.com/whisper233/DDH-LDL.

References

[1] X. Shen, W. Liu, I. Tsang, Q.-S. Sun, and Y.-S. Ong, "Compact multi-label learning," in *Proc. AAAI Conf. Artif. Intell.* 2018, pp. 4066–4073.

[2] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Multilabel prediction via cross-view search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4324–4338, Sep. 2017.

[3] X. Shen, W. Liu, Y. Luo, Y. Ong, and I. W. Tsang, "Deep discrete prototype multilabel learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2675–2681.

[4] X. Chang, H. Shen, S. Wang, J. Liu, and X. Li, "Semi-supervised feature analysis for multimedia annotation by mining label correlation," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2014, pp. 74–85.

[5] X. Chang, F. Nie, Y. Yang, and H. Huang, "A convex formulation for semi-supervised multi-label feature selection," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 1171–1177.

[6] X. Geng, "Label distribution learning," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1734–1748, Jul. 2016.

[7] J. Wang and X. Geng, "Label distribution learning by exploiting label distribution manifold," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Aug. 16, 2021, doi: 10.1109/TNNLS.2021.3103178.

[8] J. Wang, T. Zhang, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.

[9] X. Luo, Y. Wu, and X.-S. Xu, "Scalable supervised discrete hashing for large-scale search," in *Proc. World Wide Web Conf.*, 2018, pp. 1603–1612.

[10] C. Zheng, L. Zhu, S. Zhang, and H. Zhang, "Efficient parameter-free adaptive multi-modal hashing," *IEEE Signal Process. Lett.*, vol. 27, pp. 1270–1274, 2020.

[11] L. Xie, J. Shen, and L. Zhu, "Online cross-modal hashing for web image retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 294–300.

[12] K. Wang and X. Geng, "Binary coding based label distribution learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2783–2789.

[13] K. Wang and X. Geng, "Discrete binary coding based label distribution learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 3733–3739.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[15] Z. Li, X. Wang, J. Li, and Q. Zhang, "Deep attributed network representation learning of complex coupling and interaction," *Knowl.-Based Syst.*, vol. 212, 2021, Art. no. 106618.

[16] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, and A. Mian, "Bi-CLKT: Bi-graph contrastive learning based knowledge tracing," *Knowl.-Based Syst.*, vol. 241, 2022, Art. no. 108274.

[17] X. Song, J. Li, Y. Tang, T. Zhao, Y. Chen, and Z. Guan, "JKT: A joint graph convolutional network based deep knowledge tracing," *Inf. Sci.*, vol. 580, pp. 510–523, 2021.

[18] H. Cui, L. Zhu, J. Li, Y. Yang, and L. Nie, "Scalable deep hashing for large-scale social image retrieval," *IEEE Trans. Image Process.*, vol. 29, pp. 1271–1284, 2020.

[19] X. Geng, C. Yin, and Z.-H. Zhou, "Facial age estimation by learning from label distributions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 10, pp. 2401–2412, Oct. 2013.

[20] X. Geng and P. Hou, "Pre-release prediction of crowd opinion on movies by label distribution learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3511–3517.

[21] D. E. Rumelhart *et al.*, "A general framework for parallel distributed processing," *Parallel Distrib. Process.: Explorations Microstructure Cogn.*, vol. 1, no. 26, pp. 45–76, 1986.

[22] L. Xie, J. Shen, J. Han, L. Zhu, and L. Shao, "Dynamic multi-view hashing for online image retrieval," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3133–3139.

[23] L. Zhu, X. Lu, Z. Cheng, J. Li, and H. Zhang, "Deep collaborative multi-view hashing for large-scale image search," *IEEE Trans. Image Process.*, vol. 29, pp. 4643–4655, 2020.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[25] X. Lu, L. Zhu, Z. Cheng, L. Nie, and H. Zhang, "Online multi-modal hashing with dynamic query-adaption," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev.*, 2019, pp. 715–724.

[26] L. Zhu, Z. Huang, Z. Li, L. Xie, and H. T. Shen, "Exploring auxiliary context: Discrete semantic transfer hashing for scalable image retrieval," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5264–5276, Nov. 2018.

[27] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[28] S. Li and W. Deng, "Blended emotion in-the-wild: Multi-label facial expression recognition using crowdsourced annotations and deep locality feature learning," *Int. J. Comput. Vis*, vol. 127, no. 6, pp. 884–906, 2019.

[29] L. Liang, L. Lin, L. Jin, D. Xie, and M. Li, "SCUT-FBP5500: A diverse benchmark dataset for multi-paradigm facial beauty prediction," in *Proc. Int. Conf. Pattern Recognit.*, 2018, pp. 1598–1603.

[30] J. Yang, M. Sun, and X. Sun, "Learning visual sentiment distributions via augmented conditional probability neural network," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 224–230.

[31] D. Xie, L. Liang, L. Jin, J. Xu, and M. Li, "SCUT-FBP: A benchmark dataset for facial beauty perception," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 1821–1826.