Research Reproducibility Paper: Learning Neural Networks for Multi-label Medical Image Retrieval Using Hamming Distance Fabricated with Jaccard Similarity Coefficient

Asim Manna[0000−0001−7617−9762] and Debdoot Sheet[0000−0001−9046−149X]

Indian Institute of Technology Kharagpur, 721302, West Bengal, India
{asimmanna17@kgpian,debdoot@ee}.iitkgp.ac.in

Abstract. This is the research reproducibility paper for the ICPR 2024 paper "Learning Neural Networks for Multi-label Medical Image Retrieval Using Hamming Distance Fabricated with Jaccard Similarity Coefficient." This work provides an overview of the problem, highlighting the motivation behind multi-label image retrieval in the medical domain. It delves into the algorithmic framework, offering a detailed explanation of the proposed solution, including illustrative examples. Additionally, we outline the implementation guidelines for replicating the approach. A thorough experimental evaluation emphasizes the employed loss functions, parameter analysis, and an ablation study. Furthermore, the limitations of the method are discussed in the last.

Keywords: Content-based medical image retrieval · Deep neural hashing network · Jaccard coefficient · Hamming distance · Pairwise similarity

1 Introduction

In multi-label medical imaging, a pair of images may exhibit comorbid pathologies as well as distinct pathologies [2,7,3]. Therefore, the Hamming distance (HD) [4,6,5] between the generated hash codes of a multi-label image pair should account for the proportion of shared labels relative to the total possible labels. Our approach generates hash codes for multi-label medical image retrieval by integrating both HD and the Jaccard Similarity Coefficient (JSC) [1]. In Section 2, the main objective is mathematically defined. The key goals of the proposed method are: (i) to ensure that the Hamming distance (HD) between hash codes accurately reflects the number of matching pathologies, thus improving the precision of representation and retrieval, and (ii) to generate distinctive features for different combinations of pathologies within images. To achieve these objectives during training, we utilize two loss functions: adaptive Hamming distance loss (AHDL) and pairwise multi-label classification loss (PMCL). Section 3 outlines the entire learning process using these loss functions, including examples and the corresponding PyTorch

code for the proposed AHDL. Our source code is available at Github [1]. We address reproducibility from two perspectives: method reproducibility and results reproducibility. Method reproducibility refers to the ability to replicate the proposed approach using the provided implementation details, ensuring that others can independently verify the findings with the same approach and dataset. Results reproducibility, in contrast, emphasizes the consistency of outcomes, ensuring that applying the described method yields the same results as reported in the main paper. Both aspects are discussed in detail in Section 5. The two associated scale parameters, λ_1 and λ_2, correspond to the two loss functions. In Section 6, we examine the influence of the method's parameters and the impact of each individual loss. Finally, we present the limitations of this work in Section 7, followed by the conclusion in Section 8.

2 Objective

Consider a training set of images represented as $\mathbf{X}_{\mathfrak{T}} = \{\mathbf{x}_1^{\mathfrak{T}}, \mathbf{x}_2^{\mathfrak{T}}, \ldots, \mathbf{x}_i^{\mathfrak{T}}, \ldots, \mathbf{x}_{U_1}^{\mathfrak{T}}\}$. $\mathbf{y}_i^{\mathfrak{T}} \in \{0,1\}^L$ represents the label set of image $\mathbf{x}_i^{\mathfrak{T}} \in \mathbb{R}^{M \times N}$, where L denotes the number of possible labels in the dataset. Consider a non-linear hash function $F : \mathbb{R}^{M \times N} \mapsto \{-1,1\}^K$ such that each $\mathbf{x}_i^{\mathfrak{T}} \in \mathbb{R}^{M \times N}$ is an image to be hashed into a K-length binary hash code $\mathbf{b}_i^{\mathfrak{T}} \in \{-1,1\}^K$.

Let, the number of all possible labels and shared labels between an image pair $\mathbf{x}_i^{\mathfrak{T}}, \mathbf{x}_j^{\mathfrak{T}}$ are denoted as $n_{ij}^{(1)} (= |\mathbf{y}_i^{\mathfrak{T}} \cup \mathbf{y}_j^{\mathfrak{T}}| \neq 0), n_{ij}^{(2)} (= |\mathbf{y}_i^{\mathfrak{T}} \cap \mathbf{y}_j^{\mathfrak{T}}|)$ respectively. Our aims to learn $F(\cdot)$ following a supervised learning approach such that $d_H(\mathbf{b}_i^{\mathfrak{T}}, \mathbf{b}_j^{\mathfrak{T}}) \leq d_H(\mathbf{b}_i^{\mathfrak{T}}, \mathbf{b}_k^{\mathfrak{T}})$ if and only if $\frac{n_{ij}^{(2)}}{n_{ij}^{(1)}} \geq \frac{n_{ik}^{(2)}}{n_{ik}^{(1)}}$, where $\mathbf{b}_i^{\mathfrak{T}} = F(\mathbf{x}_i^{\mathfrak{T}})$ and $d_H(\cdot)$ represents the HD between two hash codes of length K. $\frac{n_{ij}^{(2)}}{n_{ij}^{(1)}}$ denotes the JSC between the label sets. The idea is that if the number of common labels between a pair of images is more, then the HD between a pair of generated hash codes should be less.

3 Method

This section discusses the method to achieve the above objective, with supporting examples. The pseudo code for the overall training pipeline to learn the hash code from images is presented Algorithm 1, respectively. An example of how to compute the ground truth HD using the given JSC between an image pair label set is provided in Table 1. Finally, the PyTorch code for AHDL is presented in Code 1.1.

[1] `https://github.com/asimmanna17/RRPR2024`

Algorithm 1: The pseudo code for the overall training pipeline of our method to learn the hash code from images

Input: Train image pair $(\mathbf{x}_i^{\mathfrak{T}}, \mathbf{x}_j^{\mathfrak{T}})$, label set pair $(\mathbf{y}_i^{\mathfrak{T}}, \mathbf{y}_j^{\mathfrak{T}})$, and Train set $\mathbf{X}_{\mathfrak{T}}$, scale parameters λ_1 and λ_2

Output: Binary hash code $(\mathbf{b}_i^{\mathfrak{T}}, \mathbf{b}_j^{\mathfrak{T}})$

Initialize: Model $\texttt{net}_\texttt{e}(\cdot)$, $\texttt{net}_\texttt{c}(\cdot), \texttt{fc}_\texttt{h}(\cdot)$, and AHDL $J_1 = 0$, PMCL $J_2 = 0, \#epochs = T$

for *each epoch=1:T* **do**

  for $(\mathbf{x}_i^{\mathfrak{T}}, \mathbf{x}_j^{\mathfrak{T}}) \in \mathbf{X}_{\mathfrak{T}} \times \mathbf{X}_{\mathfrak{T}}$ **do**

    $\mathbf{z}_i, \mathbf{z}_j \leftarrow \texttt{net}_\texttt{e}(\mathbf{x}_i^{\mathfrak{T}}), \texttt{net}_\texttt{e}(\mathbf{x}_j^{\mathfrak{T}})$;

    $\hat{\mathbf{y}}_i^{\mathfrak{T}}, \hat{\mathbf{y}}_j^{\mathfrak{T}} \leftarrow \texttt{net}_\texttt{c}(\mathbf{z}_j), \texttt{net}_\texttt{c}(\mathbf{z}_j)$; `/* Predicted labels */`

    $\mathbf{h}_i, \mathbf{h}_j \leftarrow Tanh(\texttt{fc}_\texttt{h}(\mathbf{z}_i)), Tanh(\texttt{fc}_\texttt{h}(\mathbf{z}_j))$; `/* Continuous hash codes */`

    $d_H((\mathbf{h}_i, \mathbf{h}_j) \leftarrow \frac{K}{2}(1 - \cos(\mathbf{h}_i, \mathbf{h}_j))$; `/* Predicted HD */`

    $n_{ij}^{(1)} \leftarrow |\mathbf{y}_i^{\mathfrak{T}} \cup \mathbf{y}_j^{\mathfrak{T}}|$; `/* Total possible labels */`

    $n_{ij}^{(2)} \leftarrow |\mathbf{y}_i^{\mathfrak{T}} \cap \mathbf{y}_j^{\mathfrak{T}}|$; `/* Shared labels */`

    $L_{HD}^{(n_{ij}^{(1)})}(\mathbf{h}_i, \mathbf{h}_j) \leftarrow \left[K, \left\lfloor \frac{(n_{ij}^{(1)}-1)K}{n_{ij}^{(1)}} \right\rfloor, \left\lfloor \frac{(n_{ij}^{(1)}-2)K}{n_{ij}^{(1)}} \right\rfloor, \ldots, 0\right]$;

    `/* Descending order list depends on the value of` $n_{ij}^{(1)}$ `*/`

    $D_H^{(n_{ij}^{(1)}, n_{ij}^{(2)})}(\mathbf{h}_i, \mathbf{h}_j) \leftarrow L_{HD}^{(n_{ij}^{(1)})}(\mathbf{h}_i, \mathbf{h}_j)[n_{ij}^{(2)}]$; `/* Groundthruth HD depends on the value of` $n_{ij}^{(1)}, n_{ij}^{(2)}$ `*/`

    $J_1 \leftarrow J_1 + \log\left(\cosh\left(\frac{D_H^{(n_{ij}^{(1)}, n_{ij}^{(2)})}(\mathbf{h}_i, \mathbf{h}_j) - d_H(\mathbf{h}_i, \mathbf{h}_j)}{K}\right)\right)$; `/* AHDL */`

    $J_2 \leftarrow J_2 + \texttt{BCEWithLogitsLoss}(\hat{\mathbf{y}}_\texttt{i}^{\mathfrak{T}}, \mathbf{y}_\texttt{i}^{\mathfrak{T}}) + \texttt{BCEWithLogitsLoss}(\hat{\mathbf{y}}_\texttt{j}^{\mathfrak{T}}, \mathbf{y}_\texttt{j}^{\mathfrak{T}})$; `/* PMCL */`

  end

  $J = \lambda_1 J_1 + \lambda_2 J_2$;

  Update the weights of $\texttt{net}_\texttt{e}(\cdot)$, $\texttt{net}_\texttt{c}(\cdot), \texttt{fc}_\texttt{h}(\cdot)$ by minimizing J;

end

$\mathbf{b}_i^{\mathfrak{T}}, \mathbf{b}_j^{\mathfrak{T}} = sign(\mathbf{h}_i), sign(\mathbf{h}_j)$

```python
# Code for Adaptive Hamming distance loss
import torch
import torch.nn.functional as F
hash_code_length = 16
numClasses = 13
y_i = torch.tensor([[1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1.]])
y_j = torch.tensor([[0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1.]])
sum_label = y_i +y_j
union_label = (sum_label >= 1).sum(dim=1, keepdim=False) #n^(1)
intersection_label = (sum_label >= 2).sum(dim=1, keepdim=False)#n^(2)
l_hd = []
for i in range(numClasses+1):  # Loop from i=0 to i=13
    if i==0:
        l_hd.append([hash_code_length])
    else:
        sublist = sorted([int(j * hash_code_length / i) for j in range(i + 1)], reverse=True)
        l_hd.append(sublist)
g_distH = l_hd[union_label][intersection_label]
h_i = 2 * torch.rand(1, hash_code_length) - 1
h_j = 2 * torch.rand(1, hash_code_length) - 1
cos = F.cosine_similarity(h_i, h_j, dim=1, eps=1e-6)
cos_distH = F.relu((1-cos)*hash_code_length/2)
adaptive_HD_dist_loss = (torch.div(cos_distH-g_distH, hash_code_length)).cosh().log().sum()
```

Listing 1.1: Example of a PyTorch code for AHDL for an image pair along with their label sets. Hash pairs and label pairs have been selected randomly here.

Table 1: An example of computing groudthruth HD for $L = 4$ and $K = 16$.

$\mathbf{y}_i^{\mathfrak{T}}$	$\mathbf{y}_j^{\mathfrak{T}}$	$n_{ij}^{(1)}$	$n_{ij}^{(2)}$	$L_{HD}^{(n_{ij}^{(1)})}(\mathbf{h}_i, \mathbf{h}_j)$	$D_H^{(n_{ij}^{(1)}, n_{ij}^{(2)})}(\mathbf{h}_i, \mathbf{h}_j)$
$\{1,0,1,1\}$	$\{0,1,0,0\}$	4	0	$[16,12,8,4,0]$	16
$\{1,1,1,1\}$	$\{1,0,0,0\}$		1		12
$\{1,1,1,1\}$	$\{1,1,0,0\}$		2		8
$\{1,1,1,1\}$	$\{1,1,1,0\}$		3		4
$\{1,1,1,1\}$	$\{1,1,1,1\}$		4		0
$\{1,1,0,0\}$	$\{0,0,1,0\}$	3	0	$[16,10,5,0]$	16
$\{1,1,1,0\}$	$\{1,0,0,0\}$		1		10
$\{1,1,1,0\}$	$\{1,1,0,0\}$		2		5
$\{1,1,1,0\}$	$\{1,1,1,0\}$		3		0
$\{1,0,0,0\}$	$\{0,1,0,0\}$	2	0	$[16,8,0]$	16
$\{1,1,0,0\}$	$\{1,0,0,0\}$		1		8
$\{1,1,0,0\}$	$\{1,1,0,0\}$		2		0
$\{1,0,0,0\}$	$\{1,0,0,0\}$	1	1	$[16,0]$	0

4 Used Database

The dataset is sourced from the publicly available NIH Chest X-ray database [2], which contains 112,120 frontal-view X-ray images from 30,805 unique patients [8]. Each image is labeled with one or more of 14 common thoracic pathologies identified in the associated radiological reports. From this dataset, we selected 51,480 images representing the 13 most frequent pathologies, including Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural thickening, Cardiomegaly, Nodule, and Mass. These images are organized into three distinct sets: a training set with 38,610 images, a gallery set with 10,296 images, and a query set with 2,574 images. All images are stored in '.npy' format. The training set is used during training, while the gallery and query sets are used during inference. The dataset is available at Mendeley [3]. After downloading and extracting the 'Dataset.zip' file, three image subfolders are provided: 'train', 'gallery', and 'query'.

5 Implement Guidelines

In this section, we provide a detailed implementation guide for our algorithm, outlining the steps for end-to-end training to generate hash codes from images and measure retrieval performance. The experiments are conducted on a server equipped with 2× Intel Xeon 4110 CPUs, 12 × 8 GB DDR4 ECC Reg. RAM, 2 × 4 TB HDD, 4× Nvidia GTX 1080Ti GPUs, each with 11 GB DDR5 RAM, and Ubuntu 20.04 LTS operating system. The algorithms are implemented using Python 3.9 with PyTorch 1.10 and CUDA 11.2. The source code is available at: `https://github.com/asimmanna17/RRPR2024`. It can be used for two purposes: Method reproducibility and result reproducibility.

5.1 Method reproducibility

Method reproducibility refers to the ability to replicate the proposed approach based on the provided implementation details. In this case, the algorithm's code is demonstrated using a small subset of the dataset found in the '*./Dataset*' directory. Anyone can replicate the code with different datasets, enabling independent verification of the results using the same method. Furthermore, ensure that the '*./Dataset*' directory contains three subfolders.

5.2 Result reproducibility

Results reproducibility focuses on the consistency of outcomes, ensuring that the application of the described method yields the same results as reported in the main paper. To reproduce the results shown in Table 2 and Table 3

[2] `https://www.kaggle.com/datasets/nih-chest-xrays/data`
[3] `https://data.mendeley.com/datasets/c5x35tmj5v/1`

(main paper), the full dataset provided above must be used and saved in the appropriate directory, i.e., './Dataset'. For training, the '*train.py*' script should be executed using the images from '*train*' folder. However, the training process can be skipped since the pre-trained model has already been uploaded at here [4]. Save the models '*./Datastore/Models/*' directory. The models are saved with the names '*JaccHash_16.pkl*' for a hash code length of 16, '*JaccHash_32.pkl*' for a hash code length of 32, '*JaccHash_48.pkl*' for a hash code length of 48, and '*JaccHash_64.pkl*' for a hash code length of 64. Once the trained models are available, the inference results can be reproduced by running '*evaluation.py*' with the '*gallery*' and '*query*' folders by given specific hash code length. Ensure that all data paths are correctly linked to the code.
For Table 4 (main paper), the notebook 'demo.ipynb' can be used.

6 Experimental Analysis

In this section, we present an analysis of the impact of both employed loss functions through loss analysis and an ablation study.

Table 2: Performance of proposed method with $K = 48$ for different values of λ_1, λ_2.

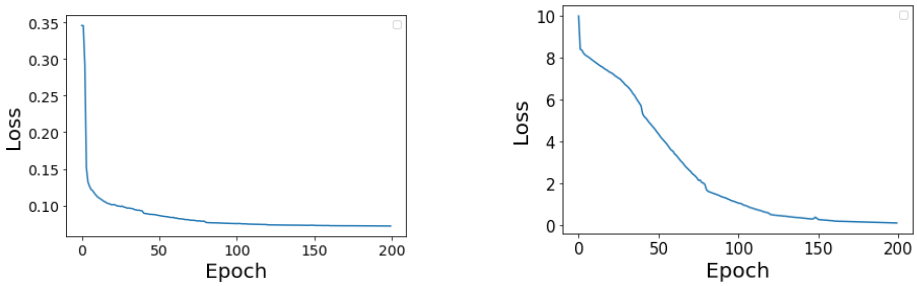
λ_1	λ_2	$nDCG$@100	ACG@100	$wMAP$
0.5	1.0	0.6378	0.3941	0.4661
1.0	1.5	**0.6426**	**0.4028**	**0.4767**
1.5	2.0	0.6388	0.3973	0.4671
2.0	2.5	0.6376	0.3938	0.4651
2.5	3.0	0.6408	0.3998	0.4753
3.0	3.5	0.6374	0.3937	0.4643
3.5	4.0	0.6373	0.3948	0.4650
4.0	4.5	0.6403	0.3984	0.4691
4.5	5.0	0.6367	0.3928	0.4615

6.1 Hyperparameter and loss analysis

During training, two scaling hyperparameters, λ_1 and λ_2, are employed to control the contributions of AHDL and PMCL, respectively. The results for different combinations of λ_1 and λ_2 are presented in Table 2, showing how these hyperparameters impact the model's performance. The training loss curves for both AHDL and PMCL are depicted in Figure 1, illustrating how the losses evolve over the course of 200 epochs. These curves provide insight into the

[4] `https://iitkgpacin-my.sharepoint.com/:f:/g/personal/asimmanna17_kgpian_iitkgp_ac_in/EnpMHJhxq21IofZky0V_gTYBbSkzb0r_aDBi6c_A-0E6ug?e=ilrbxm`

model's convergence behavior and the stability of the training process for each loss function. Additionally, the impact of these loss functions is further explored through an ablation study, with the results summarized in Table 3. This study highlights the individual contributions of each loss function, providing a clearer understanding of their roles in the overall performance of the model.



(a) AHDL (b) PMCL

Fig. 1: The training loss progression over 200 epochs for both losses.

Table 3: Ablation study for $K = 48$.

Employed loss	$nDCG@100$	$ACG@100$	$wMAP$
AHDL	0.6083	0.3322	0.3954
AHDL + PMCL	0.6426	0.4028	0.4767

7 Limitation

The Hamming distance (HD) between any pair of hashes ranges from 0 to K. Given that $0 \leq n_{ij}^{(2)} \leq n_{ij}^{(1)} \leq L$, the number of possible values for $n_{ij}^{(2)}$ is $(n_{ij}^{(1)}+1)$. Approximately $(n_{ij}^{(1)}+1)$ equidistant points are chosen from the interval $[0, K]$ using the floor function. These HD values are then stored in a descending order list, denoted as $L_{HD}^{(n_{ij}^{(1)})}(\mathbf{h}_i, \mathbf{h}_j)$ as shown in line 10 of Alogorithm 1. As the number of labels, L, increases, $n_{ij}^{(1)}$ can also increase, which means that the spacing between two points in $L_{HD}^{(n_{ij}^{(1)})}(\mathbf{h}_i, \mathbf{h}_j)$ becomes smaller. For large values of L, consecutive points in the list become very close to each other, making it difficult for the model to distinguish between the true HD values corresponding to different values of $\frac{n_{ij}^{(2)}}{n_{ij}^{(1)}}$.

8 Conclusion

In this companion paper, we outline and discuss the motivation behind the challenges of multi-label medical image retrieval and the design of a one-to-one mapping between HD and JSC. We provide details on training, implementation guidelines, and analyses of loss and parameters, all of which contribute to enhancing reproducibility. The code for this study is publicly available, benefiting the medical image retrieval research community. Lastly, we address the limitations of our work and outline our plans to address these limitations and propose solutions in future research.

References

1. Bag, S., Kumar, S.K., Tiwari, M.K.: An efficient recommendation generation using relevant jaccard similarity. Inf. Sciences **483**, 53–64 (2019)
2. Guo, X., Duan, J., Gichoya, J., Trivedi, H., Purkayastha, S., Sharma, A., Banerjee, I.: Multi-label medical image retrieval via learning multi-class similarity. Available at SSRN 4149616 (2022)
3. Hou, D., Zhao, Z., Hu, S.: Multi-label learning with visual-semantic embedded knowledge graph for diagnosis of radiology imaging. IEEE Access **9**, 15720–15730 (2021)
4. Luo, X., Wang, H., Wu, D., Chen, C., Deng, M., Huang, J., Hua, X.S.: A survey on deep hashing methods. ACM Trans. Knowl. Discovery Data **17**(1), 1–50 (2023)
5. Manna, A., Sista, R., Sheet, D.: Deep neural hashing for content-based medical image retrieval: A survey (2024)
6. Rodrigues, J., Cristo, M., Colonna, J.G.: Deep hashing for multi-label image retrieval: a survey. Artif. Intell. Rev. **53**(7), 5261–5307 (2020)
7. Sorower, M.S.: A literature survey on algorithms for multi-label learning. Oregon State University, Corvallis **18**(1), 25 (2010)
8. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., Summers, R.M.: Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: Proc. Conf. Comput. Vision Pattern Rec. pp. 2097–2106 (2017)