ICS 35.040
CCS L 71

团　体　标　准

T/AI XXX.XX—XXXX

信息技术 视觉特征编码 第 3 部分：深度学习特征

Information Technology – Visual Feature Coding

Part 3: Deep Learning Feature

点击此处添加与国际标准一致性程度的标识

（征求意见稿）

（在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上）

XXXX-XX-XX 发布　　XXXX-XX-XX 实施

中关村视听产业技术创新联盟　发布

目　　次

前　言

本文件按照 GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规则起草。

本文件是T/AI XX.XX-XXXX《信息技术 视觉特征编码》的第3部分。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由新一代人工智能产业技术创新战略联盟AI标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本文件起草单位：鹏城实验室，北京大学，青岛海信网络科技股份有限公司，青岛新一代人工智能技术研究院，青岛图灵科技有限公司，浙江邦盛科技股份有限公司

本文件起草人：段凌宇，刘美含，白燕，楼燚航，王雯雯，冯栋，高峰，陈卓，杨文瀚，王新宇，陈伟，赵海英，崔晓冉。

引　言

《信息技术 视觉特征编码》拟由6个部分构成。

——第1部分：系统；

——第2部分：手工设计特征；

——第3部分：深度学习特征；

——第4部分：深度特征图；

——第5部分：语义分割图；

——第6部分：结构点序列。

本文件的发布机构提请注意，声明符合本文件时，可能涉及到XX、XX中如下X项相关专利的使用。专利名称如下：

CNxxxxx.1，xxxxxx；

本文件的发布机构对于该专利的真实性、有效性和范围无任何立场。

该专利持有人已向本文件的发布机构保证，他愿意同任何申请人在合理且无歧视的条款和条件下，就专利授权许可进行谈判。该专利持有人的声明已在本文件的发布机构备案，相关信息可以通过以下联系方式获得：

联 系 人：段凌宇

通讯地址：北京市海淀区颐和园路5号 北京大学 数字媒体研究所

邮政编码：100098

电子邮件：lingyu@pku.edu.cn

电话：

传真：

网址：https://cs.pku.edu.cn/info/1089/1654.htm

请注意除上述专利外，本文件的某些内容仍可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

信息技术 视觉特征编码
第 3 部分：深度学习特征

1 范围

本文件规定了从深度网络中提取特征的二值化压缩编码方法及其编码过程, 提供了面向图像的细粒度检索的检索框架。基于哈希学习的编码方法可以应用于各种卷积网络提取出的深度特征。

本文件适用于图像和视频数据中目标或场景的搜索、识别等应用。

2 规范性引用文件

本文件不包含规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1

深度网络 deep neural network

多层神经网络。

3.2

深度特征 deep feature

经过深度卷积网络提取后的具有语义信息的浮点数向量。

3.3

交叉熵损失 cross-entropy loss

基于最大似然函数对数化的监督约束

3.4

三元组损失 triplet loss

基于三元组的增大同一类别事物（如车、人、人脸）中不同对象的区分度的监督约束。

3.5

码书 codebook

聚类算法中簇的质心集合

3.6

码字 codeword

聚类算法中一个簇的质心

3.7

超参 hypterparameter

超参数是在开始学习过程之前设置值的参数

4 约定

4.1 概述

本部分中使用的数学运算符和优先级参照C语言。但对整型除法和算术移位操作进行了特定定义。除特别说明外，约定编号和计数从0开始。

4.2 算术运算符

算术运算符定义见表1。

表 1 算术运算符定义

算术运算符	定义
+	加法运算
-	减法运算（二元运算符）或取反（一元前缀运算符）
×	乘法运算
a^b	幂运算，表示a的b次幂。也可表示上标
/	整除运算，沿向0的取值方向截断。例如，7/4和-7/-4截断至1，-7/4和7/-4截断至-1
÷	除法运算，不做截断或四舍五入
$\frac{a}{b}$	除法运算，不做截断或四舍五入
$\sum_{i=a}^{b} f(i)$	自变量i取由a到b（含b）的所有整数值时，函数$f(i)$的累加和
$a \% b$	模运算，a除以b的余数，其中a与b都是正整数
Sqrt	开平方运算
Arctg	反正切运算
Arcsin	反正弦运算
Arccos	反余弦运算
Abs	取绝对值
Max	取较大值
Min	取较小值
tan	正切运算

4.3 逻辑运算符

逻辑运算符定义见表2。

表 2 逻辑运算符定义

逻辑运算符	定义
a&&*b*	*a*和*b*之间的与逻辑运算
a || *b*	*a*和*b*之间的或逻辑运算
!	逻辑非运算

4.4 关系运算符

关系运算符定义见表3。

表 3 关系运算符定义

关系运算符	定义
>	大于
>=	大于或等于
<	小于
<=	小于或等于
==	等于
!=	不等于

4.5 位运算符

位运算符定义见表4。

表 4 位运算符定义

位运算符	定义
&	与运算
|	或运算

~	取反运算
a>>*b*	将*a*以2的补码整数表示的形式向右移*b*位。仅当*b*取正数时定义此运算
a<<*b*	将*a*以2的补码整数表示的形式向左移*b*位。仅当*b*取正数时定义此运算

4.6 赋值

赋值运算定义见表5。

表 5 赋值运算定义

赋值运算	定义
=	赋值运算符
++	递增，*x*++相当于*x*= *x*+ 1。当用于数组下标时，在自加运算前先求变量值
--	递减，*x*--相当于*x*=*x* – 1。当用于数组下标时，在自减运算前先求变量值
+=	自加指定值，例如*x* += 3相当于*x* = x + 3，*x* += (-3)相当于*x* = *x* + (-3)
-=	自减指定值，例如*x*-= 3相当于*x* = *x*- 3,，*x*-= (-3)相当于*x* = *x*- (-3)

4.7 位流语法、解析过程和解码过程的描述方法

位流语法描述方法类似C语言。位流的语法元素使用粗体字表示，每个语法元素通过名字（用下划线分割的英文字母组，所有字母都是小写）、语法和语义来描述。语法表和正文中语法元素的值用常规字体表示。

某些情况下，可在语法表中应用从语法元素导出的其他变量值，这样的变量在语法表或正文中用不带下划线的小写字母和大写字母混合命名。大写字母开头的变量用于解码当前以及相关的语法结构，也可用于解码后续的语法结构。小写字母开头的变量只在它们所在的小节内使用。

语法元素值的助记符和变量值的助记符与它们的值之间的关系在正文中说明。在某些情况下，二者等同使用。助记符由一个或多个使用下划线分隔的字母组表示，每个字母组以大写字母开始，也可包括多个大写字母。

位串的长度是4的整数倍时，可使用十六进制符号表示。十六进制的前缀是“0x”，例如“0x1a”表示位串“0001 1010”。

条件语句中0表示FALSE，非0表示TRUE。

语法表描述了所有符合本部分的位流语法的超集，附加的语法限制在相关条中说明。

表6给出了描述语法的伪代码例子。当语法元素出现时，表示从位流中读一个数据单元。

表 6　语法描述的伪代码

伪代码	描述符
/*语句是一个语法元素的描述符，或者说明语法元素的存在、类型和数值，下面给出两个例子。*/	
syntax_element	ue(v)
conditioning statement	
/*花括号括起来的语句组是复合语句，在功能上视作单个语句。*/	
{	
Statement	
…	
}	
/*“while”语句测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/	
while (condition)	
statement	
/*“do … while”语句先执行循环体一次，然后测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/	
do	
statement	
while (condition)	
/*“if … else”语句首先测试condition，如果为TRUE，则执行primarY语句，否则执行alternative语句。如果alternative语句不需要执行，结构的“else”部分和相关的alternative语句可忽略。*/	
if (condition)	
primarY statement	

else	
alternative statement	
/*“for”语句首先执行initial语句，然后测试condition，如果conditon为TRUE，则重复执行primarY语句和subsequent语句直到condition不为TRUE。*/	
for (initial statement; condition; subsequent statement)	
primarY statement	

解析过程和解码过程用文字和类似 C 语言的伪代码描述。

4.8 函数

以下函数用于语法描述。函数由函数名及左右圆括号内的参数构成。函数也可没有参数。

split_vectors()

进行乘积量化时，拆分高维输入特征向量为多个低维特征向量集合，具体过程的定义见章节6.2 。

search_codebook()

进行乘积量化时，在量化码本中查找最佳匹配的编码向量，具体过程的定义见章节6.2 。

vstack()

进行乘积量化时，将所有编码向量组合成一个完整的编码向量，具体过程的定义见章节6.2 。

4.9 保留、禁止和标记位

本部分定义的位流语法中，某些语法元素的值被标注为“保留”（reserved）或“禁止”（forbidden）。

“保留”定义了一些特定语法元素值用于将来对本部分的扩展。这些值不应出现在符合本部分的位流中。

“禁止”定义了一些特定语法元素值，这些值不应出现在符合本部分的位流中。

“标记位”（marker_bit）指该位的值应为‘1’。

位流中的“保留位”（reserved_bits）表明保留了一些语法单元用于将来对本部分的扩展，解码处理应忽略这些位。“保留位”不应出现从任意字节对齐位置开始的21个以上连续的‘0’。

5 语法和语义

5.1 深度学习特征编码语法

希学习特征编码语法见表7。

表 7 哈希学习特征编码语法

哈希学习特征编码语法	描述符
HashEnc {	
feat_len	u(16)

feat	vector<f(32)> (feat_len)
nbits	u(8)
weight1	vector<vector<f(32)>> (nbits*feat_len)
hash_code	vector<bool> (nbits)
for(m=0; m<nbits;m++){	
for(n=0; n<feat_len; n++){	
hash_code[n] += weight1[m][n] * feat[n]	
}	
}	
hash_code = hash_code > 0 ? 1:0	
}	

乘积量化特征编码语法见表8。

表 8　乘积量化特征编码语法

乘积量化特征编码语法	描述符
PQEnc {	
feat_len	u(16)
feat	vector<f(32)> (feat_len)
codeword_len	u(16)
codebook_len	u(16)
nbits	u(16)
group	u(16)
codebook	vector<vector<f(32)>> (nbits/group*codebook_len)
pq_code	vector<bool> (nbits)

subvectors = split_vector(feat, group)	vector<vector<f(32)>> (group*nbits/group)
codeword_indices	vector<vector< bool >> (nbits/group* $2^{codeword\_len}$)
for(j=0; j<group; j++){	
subvector_j = subvectors[j]	vector<f(32)> (nbits/group)
codebook_j = codebooks[j]	vector<f(32)> (nbits/group)
codeword_j = search_codebook(subvector_j, codebook_j)	vector<bool> (nbits/group)
codeword_indices.append(codeword_j)	
}	
pq_code = vstack(codeword_indices)	vector< bool > (nbits)
}	

其中，函数split_vector(), search_codebook()和vstack()的定义见章节1.1。

5.2 深度学习特征编码语义

feat_len
深度学习特征长度。
feat
定长深度学习特征。
nbits
目标压缩二值码长度。
group
乘积量化编码时的分组数。
weight1
哈希学习编码中用于将输入定长深度学习特征映射到目标比特数长度的映射矩阵。
hash_code
表示哈希学习码流。
pq_code
表示乘积量化码流。
codebook
表示乘积量化码书。
codeword_len
表示乘积量化码字的编码索引长度。
codebook_len

表示乘积量化码书中的码字数量。

subvectors

表示深度学习特征按组拆分后的特征集合。

codeword_indices

映射后的所有编码索引集合。

6 深度学习特征编码

6.1 概述

深度学习特征编码由两个码流组成，分别为深度哈希学习编码产生的码流hash_code以及矢量量化编码产生的码流pq_code。对于从一张图像中抽取的若干特征，输入的深度学习特征feat应为从深度卷积网络模型中提取出来的定长的32位浮点数特征向量。其中深度卷积网络模型作为非规范化部分。

输出的哈希学习编码码流以及矢量量化编码码流均为长度为目标压缩比特数nbits的二值码。

6.2 哈希学习编码流程

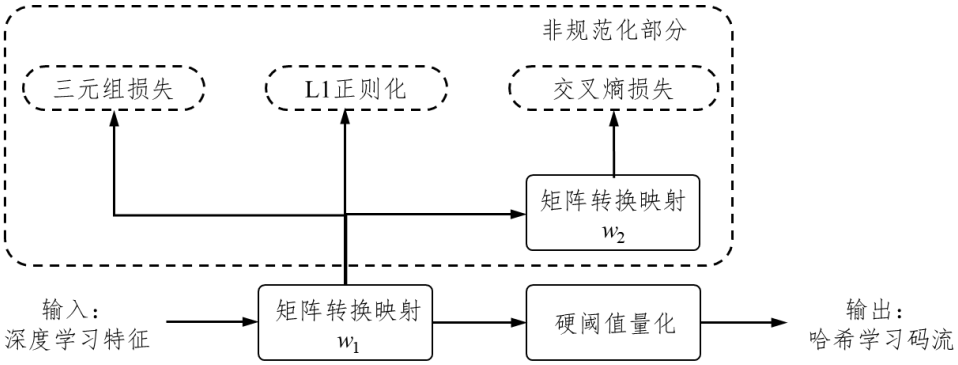

图1 深度哈希编码训练与推理流程

哈希学习编码存在两个步骤：矩阵转换映射以及硬阈值量化。

其中矩阵转换映射的矩阵即为深度哈希学习网络中全连接层的权重参数w_1和w_2。

在这里不规范化权重参数w_2以及训练迭代过程，详见附录B。规范化权重参数w_1，以及训练时的目标函数以及硬阈值量化部分。

w_1将输入深度特征feat中的向量$x \in R^d$映射到目标压缩比特数长度nbits，其中d为feat_len，并得到hash_code $\hat{x} \in R^d$，见式（1）：

$$\hat{x} = w_1^T x, \tag{1}$$

式中：

w_1——连接层中的权重矩阵。

硬阈值量化可形式化表示，见式（2）：

$$K(\hat{x}_i) = \begin{cases} 1, \hat{x}_i > 0, \\ 0, \hat{x}_i \le 0. \end{cases} \tag{2}$$

对$\hat{x}$的每一位都进行硬阈值量化，即可得到哈希编码码流。

6.3 乘积量化编码流程

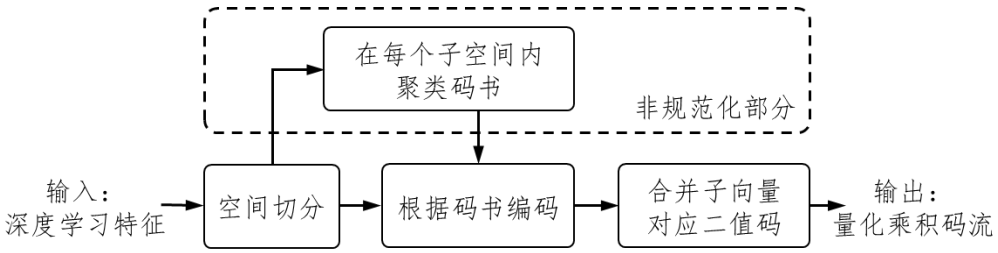

图2 乘积量化编码流程

空间切分函数**split_vectors()**将输入的深度学习特征向量$x \in R^{\text{feat_len}}$沿维度均等切分成group份。该函数过程用Python语言伪代码见表9：

表 9 空间切分函数 split_vectors()伪代码描述

```
def split_vectors(vectors, group, d):
    num_vectors = len(vectors)
    subvector_size = num_dimensions // group
    subvectors = []
    for i in range(num_vectors):
        vector_i = vectors[i]
        subvectors_i = []
        for j in range(num_subvectors):
            start_index = j * subvector_size
            end_index = (j+1) * subvector_size
            subvector_ij = vector_i[start_index:end_index]
            subvectors_i.append(subvector_ij)
        subvectors.append(subvectors_i)
    return subvectors
```

本标准中规范group=nbits/8，nbits为目标压缩长度。切分完成后，对于输入深度学习特征向量x，可得到$\left[x_0, x_1, \ldots, x_{\text{group}-1}\right]$，其中子向量$x_i$为nbits/group长度的浮点数向量，其中feat_len为输入深度学习特征长度。

对训练集中的特征向量空间切分完成后(乘积量化编码的训练集应与哈希学习编码采用的训练集一致），需要对每个空间单独训练一本码书，共group本码书，即codebook。本标准规范每个码书包含codebook_len个码字$\left\{\mu_i, 0 \le i \le 2^{\text{codeword_len}} - 1\right\}$，这些码字对应的编号为codeword_len=8位二进制数表示，每个码字应为长度为nbits/group的浮点数向量，且满足在每个子空间内码书中的码字与切分后的训练集对应子空间的向量集合均方误差和小于一定阈值，这里阈值为非规范部分。

码书的训练采用聚类算法，这里聚类算法为非规范化部分，详见附录C。

之后，对每个子向量x_i，可以利用最小均方误差找到距离最近的码字，见式（3）：

$$K^i = \arg\min_k \left\| x_i - \mu_k \right\|_2^2 . \tag{3}$$

search_codebook()函数过程用Python语言伪代码见表10：

表 10 search_codebook()函数伪代码描述

```
function search_codebook(subvector, codebook):
    best_distance = infinity
    best_codeword = -1
    for i in range(num_codewords):
        codeword_i = codebook[i]
        distance_i = compute_distance(subvector, codeword_i)
        if distance_i < best_distance:
            best_distance = distance_i
            best_codeword = i
    return best_codeword
```

最后，用码字编号的二进制作为该子向量对应的二值码，并得到$\left[K_2^0, K_2^1, \ldots, K_2^{\text{group}-1}\right]$，其中下标2表示2进制表示。使用vstack()函数将每个子向量对应二值码进行连接后，即为对应输入深度学习特征的乘积量化码流pq_code。vstack()函数过程用Python语言伪代码表示见表11：

表 11 vstack()函数伪代码描述

```
def vstack(arrays):
    # Determine the number of rows in the output array
    num_rows = 0
    for array in arrays:
        num_rows += array.shape[0]

    # Create the output array
    output = np.zeros((num_rows, num_columns))

    # Copy the input arrays into the output array
    row = 0
    for array in arrays:
        output[row : row + array.shape[0], :] = array
        row += array.shape[0]

    return output
```

6.4 长度可调整深度学习特征编码

本章节为章节6.1和章节6.2的非规范化拓展，考虑到实际应用的不同需求对深度学习特征的建模能力和计算复杂度具有不同要求。因此，本标准针对具体应用场景进行适当的参数调整和拓展。

通过调整编码长度nbits构建不同表征能力的哈希学习编码模型。较大的nbits可以提高特征的表达能力，较长的哈希编码长度可以减少哈希冲突的可能性，提高哈希函数的准确率，但过长的哈希编码长度会导致较高的计算复杂度。

通过调整编码长度nbits (由codeword_len、codebook_len和group共同决定)构建不同表征能力的量化乘积学习编码模型，较大的codeword_len、codebook_len和较大的group，导致编码长度的增加，量化后的向量能够表示的数值范围就越大，因此可以获得更高的准确率。但是，编码长度越长，需要存储和计算的信息也就越多，导致计算复杂度的增加和存储空间的增加。

此外，通过联合不同长度的量化乘积编码结果，构建多级量化乘积编码：

$$\text{pq\_code\_hier}=[\text{pq\_code}_1,\text{pq\_code}_2,\ldots,\text{pq\_code}_l], \tag{4}$$

$\text{pq_code}_1,\text{pq_code}_2,\ldots,\text{pq_code}_l$是长度分别为$\text{nbits}_1,\text{nbits}_2,\ldots,\text{nbits}_l$的量化乘积编码结果，并且$\text{nbits}_1<\text{nbits}_2<\ldots<\text{nbits}_l$，从而该编码从低位到高位的组合逐渐具有更强的表达能力。在实际应用时，可以使用pq_code_hier 以获取不同表达粒度联合表达的综合优势，也可以根据实际应用场景的性能/复杂度的需求取舍，动态选取部分表征，实现契合场景应用的高效表征。

附 录 A
（资料性）
深度哈希编码检索流程

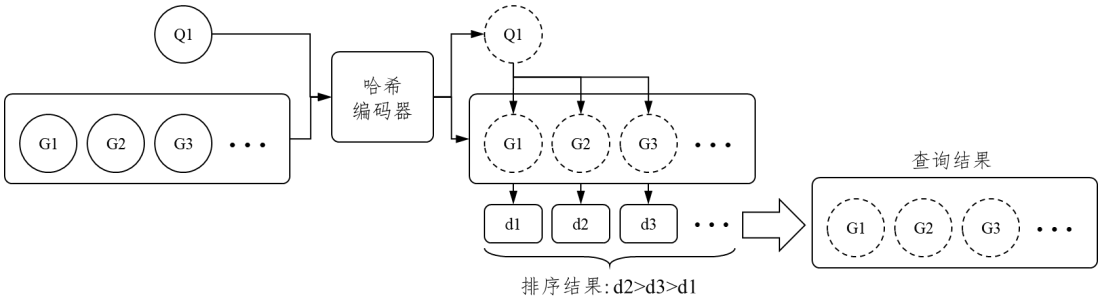

图 A.1 深度哈希编码检索流程图

给定查询集（Query）Q以及待查询集（Gallery）G，首先用深度哈希编码对这两个深度特征集合进行二值化编码。之后，见图A.1，对查询集中的每一个查询变量计算其与待查询集中所有待查询变量的相似度（距离）。这里用汉明距离计算两个二值码的相似度，汉明距离越小，相似度越高，其中汉明距离的公式表达见式（A.1）：

$$d\left(\hat{x},\hat{y}\right)=\sum \hat{x}_i \oplus \hat{y}_i, \tag{A.1}$$

式中：

⊕——异或操作。

对距离进行排序即可得到对应查询变量的深度哈希编码相似度查询结果。

A.1 乘积量化检索流程

与深度哈希编码的检索流程类似，给定查询集（Query）Q以及待查询集（Gallery）G并通过乘积量化编码得到对应的二值码后，对查询集中的每一个查询变量计算其与待查询集中变量的相似度（距离）。

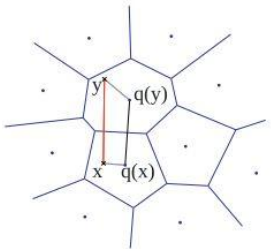

图 A.2 对乘积量化码进行对称距离计算

为了提高计算速度，建议采用对称距离计算方式（Symmetric Distance Computation），见图（A.2）。对称距离计算需要预存查询表，也就是每个子空间中两两码字之间的浮点距离$d\left(c_{m,i},c_{m,j}\right)^2$，根据距离公式检索，见式（A.2）：

$$d(x,y)=d\left(q(x),q(y)\right)=\sqrt{\sum_m d\left(q_m(x),q_m(y)\right)^2}, \tag{A.2}$$

以 O(1)复杂度计算变量 x, y 之间的距离。

A.2 两阶段检索框架检索流程

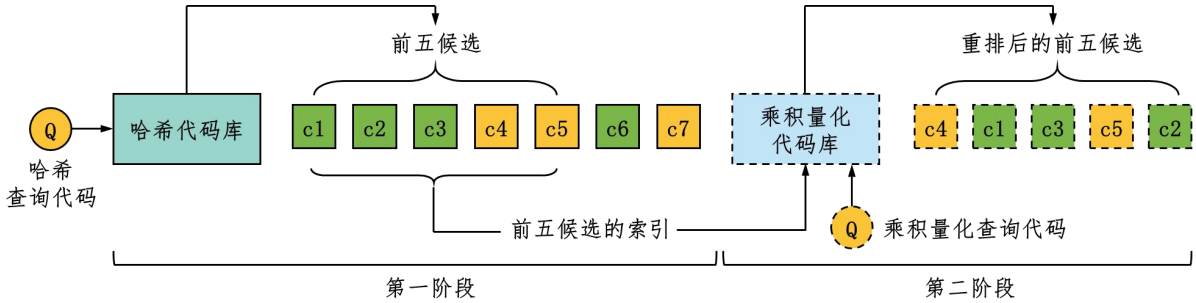

图 A.3 两阶段检索框架检索流程示意图

对深度哈希量化编码器以及乘积量化编码器进行训练后，即可利用两阶段检索框架进行检索，见图A.3。给定查询变量 Q、rerank 范围 N 以及待查询集 Gallery，利用两阶段检索得到最后检索结果，python伪代码见表 A.1：

表 A.1 两阶段检索伪代码

candidates_hash := 哈希编码检索(Q, Gallery)
candidates_pq := candidates_hash[:N]
return 乘积量化编码检索(Q, candidates_pq) + candidates_hash[N:]

见图A.3，第一层（1st LEVEL）即为深度哈希量化编码的检索流程，可得到初步搜索结果候选集candidates_hash。从该候选集中选出top N作为乘积量化部分的新待查询集candidates_pq，并执行乘积量化的检索步骤即可得到最后的查询结果。

附 录 B
（规范性）
哈希学习

B.1 哈希学习网络结构

网络结构由两个全连接层串联而成。第一个全连接层用于将输入深度特征$x \in R^d$映射到目标压缩比特数长度b，计算得到$\hat{x} \in R^b$，见式（B.1）：

$$\hat{x} = w_1^T x. \tag{B.1}$$

另外一个全连接层用于得到类别打分，仅用于训练阶段。其权重为$b \times c$的矩阵w_2，其中用于将上一层全连接层的输出$\hat{x} \in R^b$映射为类别打分$s \in R^c$，其中c为目标应用场景下大规模数据的总类别数，见式（B.2）：

$$s = w_2^T w_1^T x. \tag{B.2}$$

B.2 哈希学习目标函数

最小化目标函数包含：交叉熵损失、三元组损失、以及稀疏正则化。

B.2.1 交叉熵损失

给定具有c个类别N个样本的数据集的数据集，交叉熵损失的计算方式见式（B.3）：

$$loss_{CE} = \sum_i^N -\log\left(\frac{\exp\left(s\left[y_i\right]\right)}{\sum_{j=1}^c \exp\left(s\left[j\right]\right)}\right), \tag{B.3}$$

式中：

y_i——第i个样本的类别标签。

B.2.2 三元组损失

为了利用三元组损失，训练时需要对喂给网络的训练数据进行采样。该采样方法为非规范部分，详见附录C。在采样后得到三元组，其中x+是与x同一类别的样本，x-是与x不同类别的样本。该三元组在经过第一个全连接层后可得到$\langle \hat{x}, \hat{x}+, \hat{x}- \rangle$，因此三元组损失计算见式（B.4）：

$$loss_{triplet} = \sum_i^N \left[\left\| \hat{x} - \left(\hat{x}+\right) \right\|_2^2 - \left\| \hat{x} - \left(\hat{x}-\right) \right\|_2^2 + \alpha \right]_+, \tag{B.4}$$

式中：

α——超参，定义了负样本距离与正样本距离的软间隔，在这里不做规范。

B.2.3 稀疏正则化

L1正则化的表达见式（B.5）：

$$L_{norm} = \sum_i^b \left| \hat{x}\left[i\right] \right|. \tag{B.5}$$

附 录 C
（资料性）
乘积量化码书的训练

空间切分完成后需要为每个子空间单独训练一本码书以及对应的量化器$q(x)$。

这里在每个子空间内用k-means算法对码书进行迭代修正以达到整体的均方误差最小。k-means算法训练伪代码见表C.1。

表C.1 k-means算法训练伪代码

创建k个点作为起始码字（质心）
Do
将子空间中的每个点指派到对应码字（质心）
重新计算每个簇的码字（质心）
while not 簇变化小于阈值

阈值可自行定义，这里不做规范。

C.1 哈希学习编码训练时所需要的三元组采样

为了利用三元组损失，训练时需要通过采样得到深度学习特征三元组$\langle x, x+, x- \rangle$，其中$x+$是与$x$同一类别的样本，$x-$是与$x$不同类别的样本。在批处理中，建议采用难样本挖掘的方法进行三元组采样：对于每一个训练批（batch），随机挑选P个类别，每个类别中挑选K个样本，即每个训练批中含有$P \times K$个深度特征。对于该训练批中每一个深度特征x，定义该训练批中与x有着相同类别的深度特征集为A，剩下不同类别的深度特征集为B。

计算x与A中每一个深度特征的欧式距离，选取与x距离最远的样本作为$x-$；计算x与B中每一个深度特征的欧式距离，选取与x距离最近的样本作为$x-$。

C.2 哈希学习编码全连接层训练

为了加快训练速度并减小内存压力，训练时采用分批处理训练数据的方式，根据目标压缩比特数b以及总类别数c两个超参确定深度哈希编码网络的两个全连接层后并给定训练集T后，训练基本逻辑为：在训练时，可以根据total_loss的下降情况判断模型是否收敛，也可以通过测试集的mAP等指标观察模型的收敛情况以调整学习率等超参。伪代码表示见表C.2。

表C.2 哈希学习编码全连接层训练伪代码

Do
for batch, label in T:
batch_F := 第一个全连接层(batch)
batch_S := 第二个全连接层(batch_F)
triplet_loss := 三元组损失(batch_F)

softmax_loss := 交叉熵损失(batch_S, label)
reg_loss := 稀疏正则项(batch_F)
total_loss := triplet_loss + softmax_loss + reg_loss
根据total_loss计算model梯度
根据计算好的梯度进行更新连个线性层的矩阵权重参数
while not 收敛