基于MindSpore的细粒度哈希学习

作者：沈阳

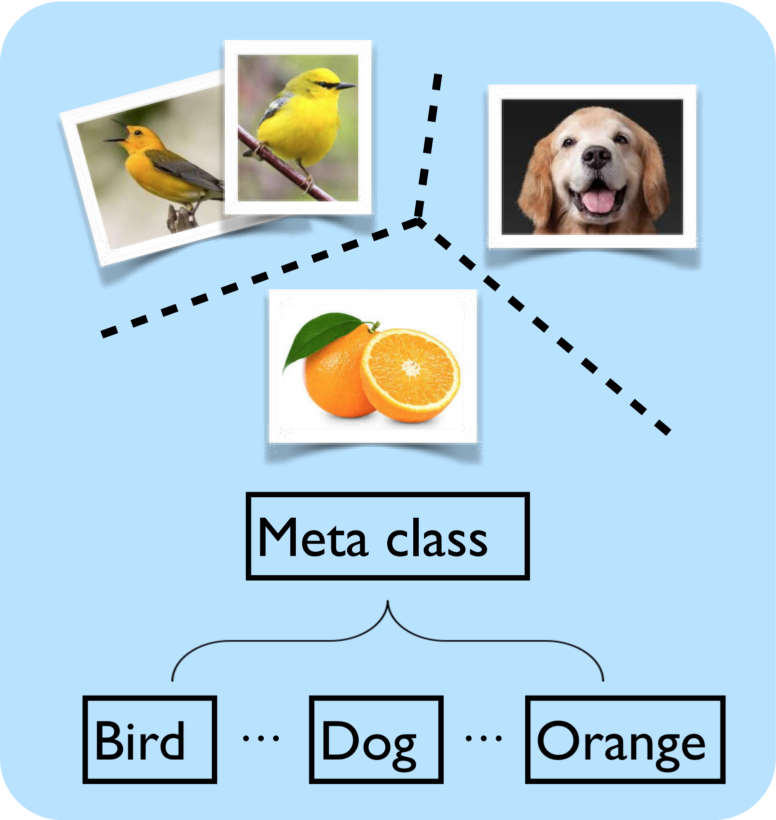
目录

contents

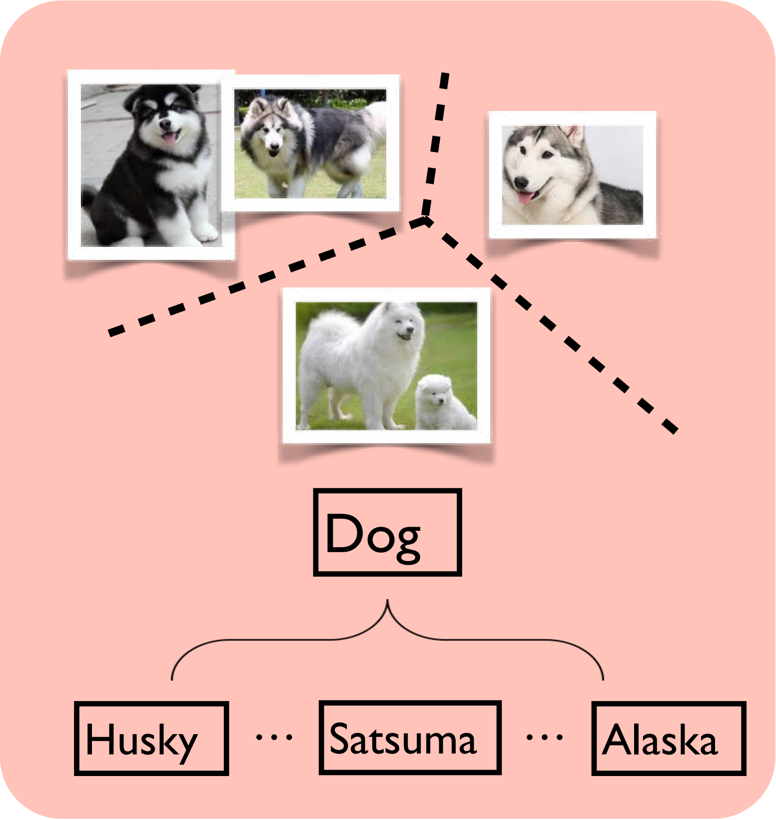
MindSpore

Introduction

Fine-grained images *vs.* generic images



Traditional image tasks
(Coarse-grained)

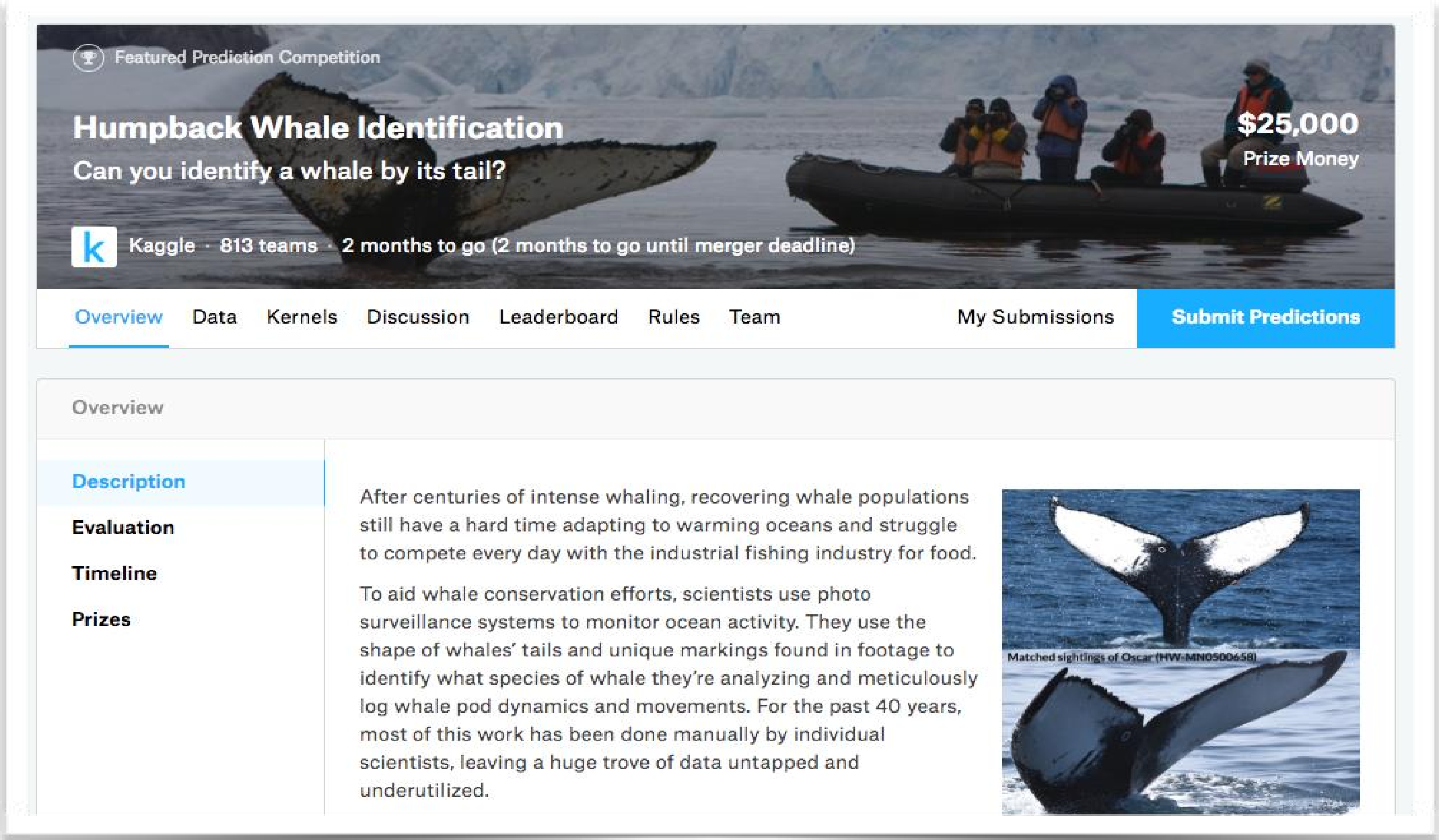


Fine-grained image tasks

Introduction (con't)

MindSpore

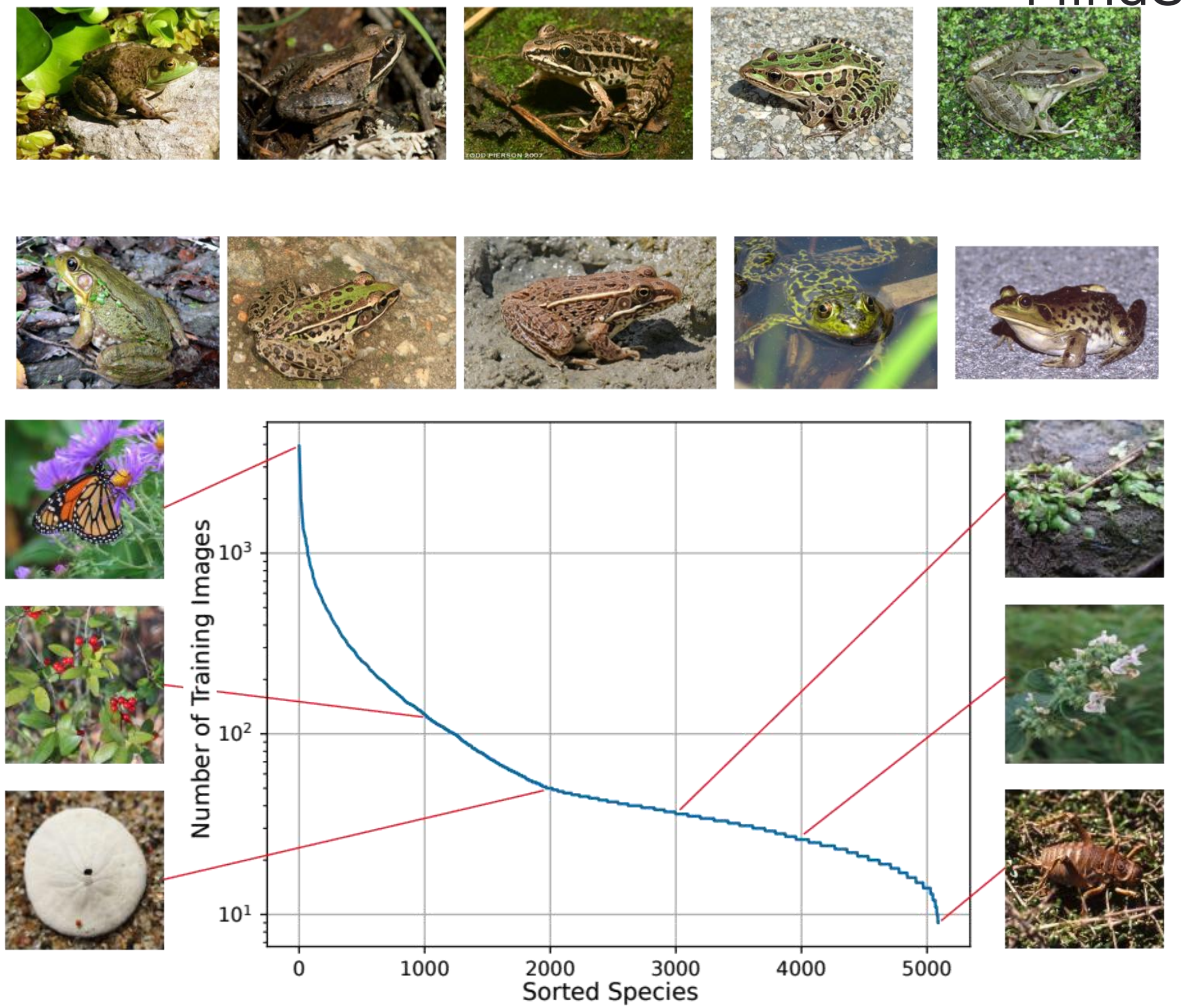
Various real-world applications



https://www.kaggle.com/c/humpback-whale-identification

Introduction (con't)

Various real-world applications



https://www.kaggle.com/c/inaturalist-challenge-at-fgvc-2017

Introduction (con't)

Challenge of Fine-Grained Image Analysis (FGIA)

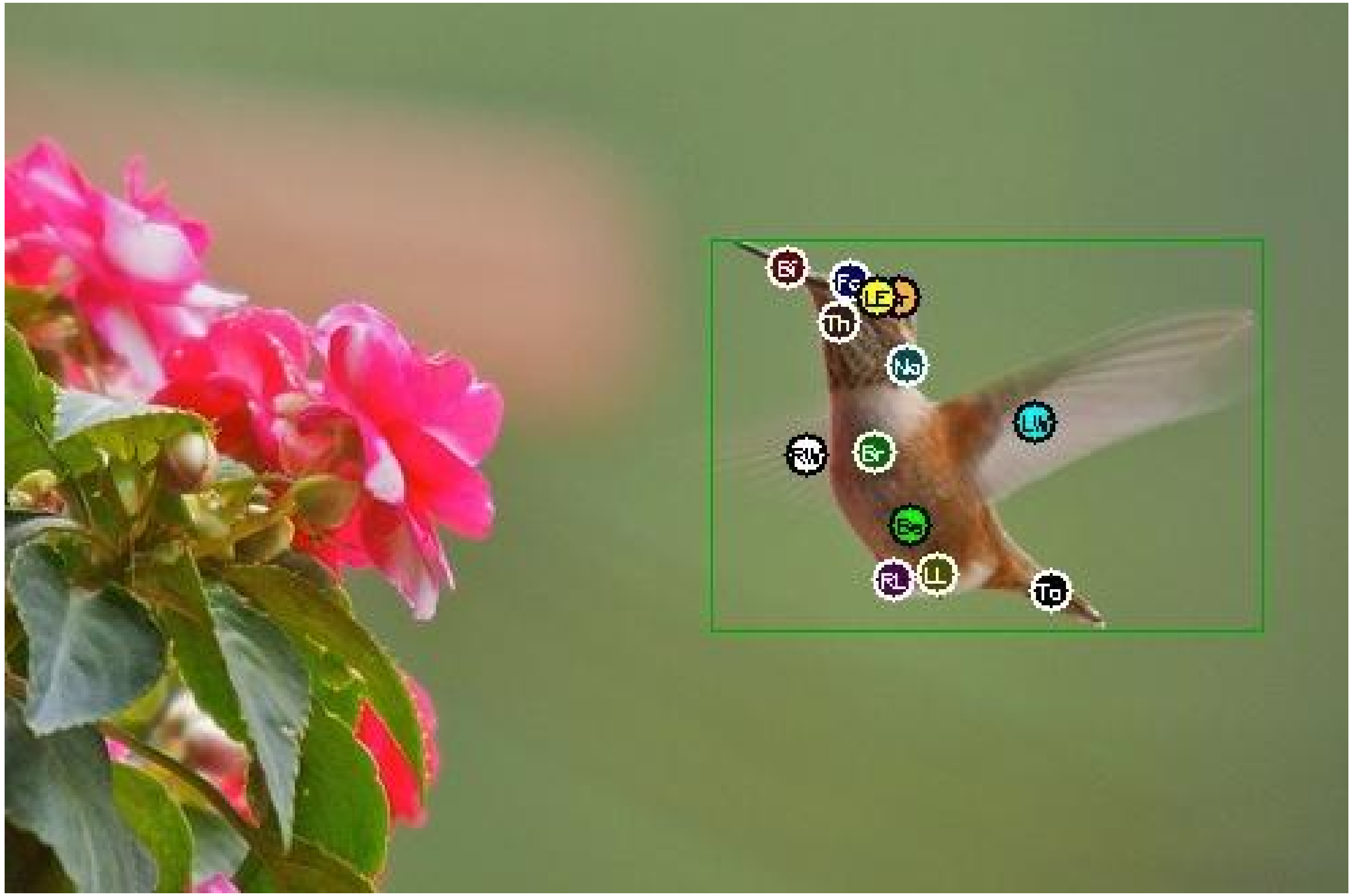
Small inter-class variance
Large intra-class variance

MindSpore

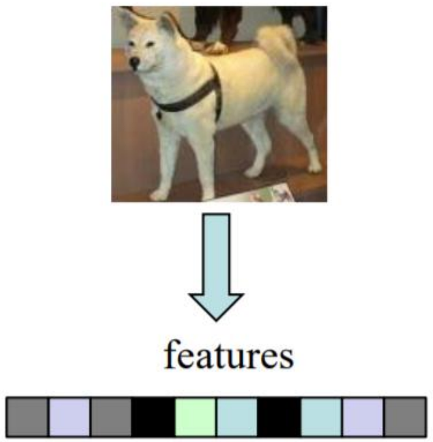
Figures are courtesy of [X. He and Y. Peng, CVPR 2017]

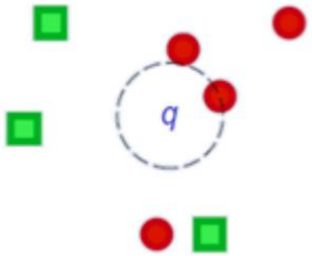
Introduction (con't)

The key of fine-grained image analysis

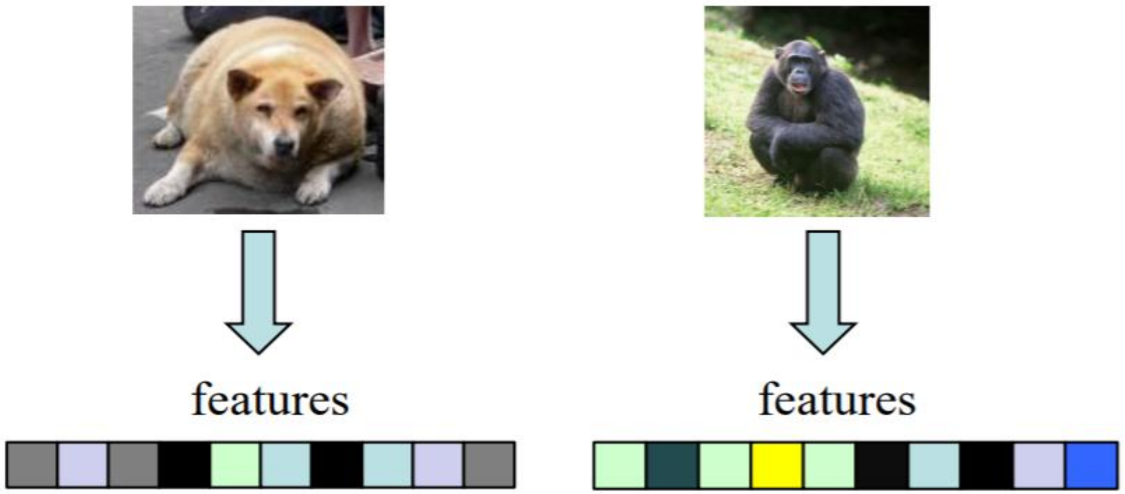


Why hashing





Nearest neighbors search

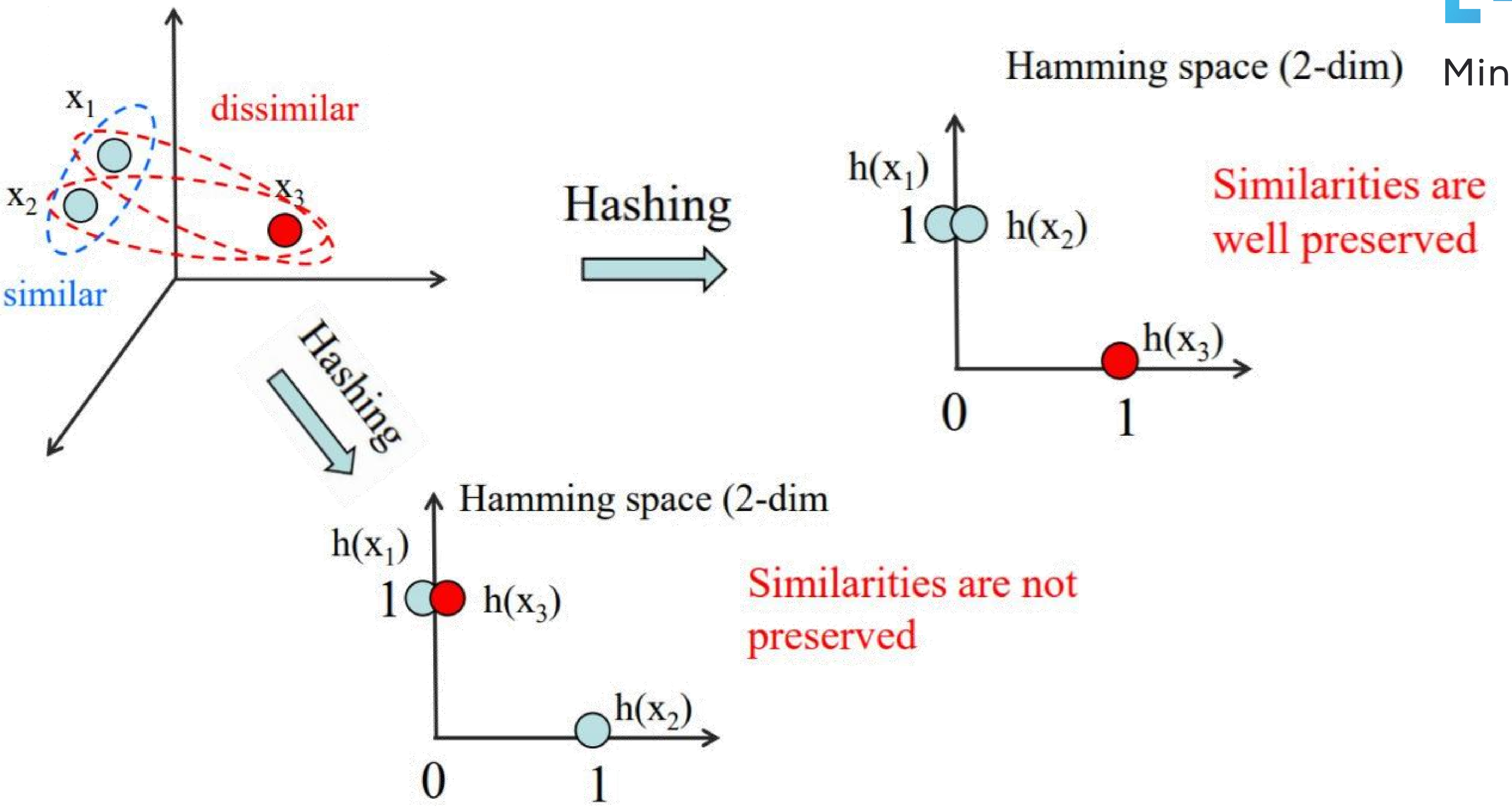


Challenge： how to efficiently search over millions or billions of data?

• e.g., if each image is represented by a 512-dim GIST vector, one needs 20G memory to store 10 million images.

Figures are courtesy of [R. Xia, Y. Pan, et al., AAAI 2014]

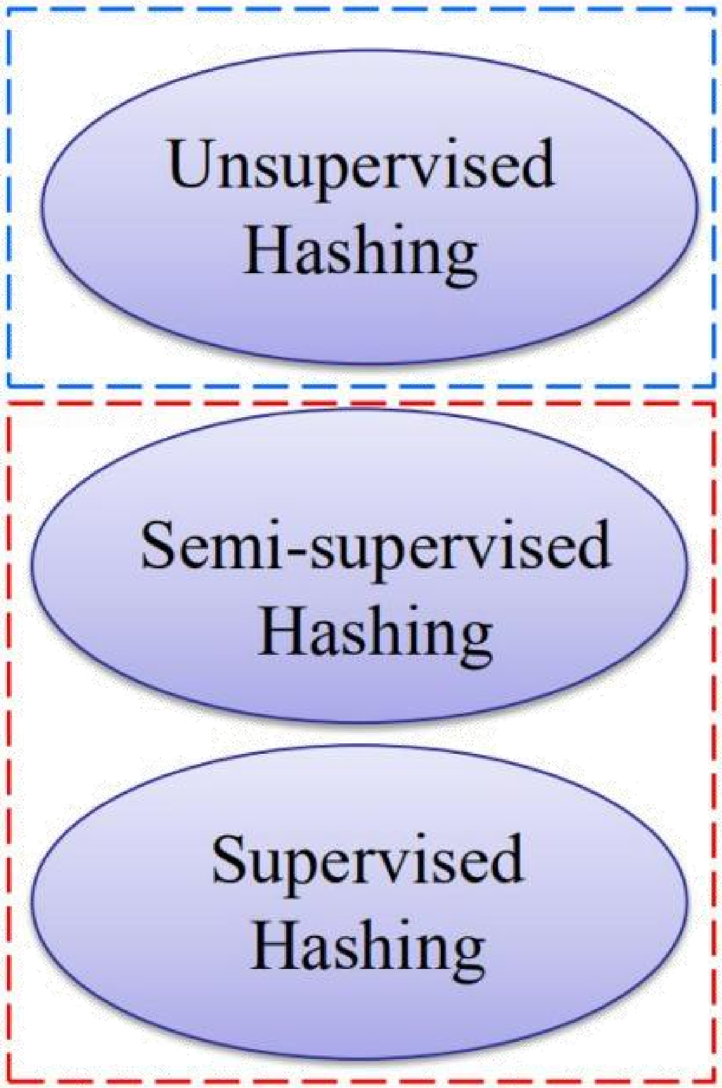
Hashing


x1
x2
x3
dissimilar
similar
Hashing
Hamming space (2-dim)
h(x1)
1
h(x2)
h(x3)
0
1
Similarities are well preserved
Hashing
Hamming space (2-dim
h(x1)
1
h(x3)
h(x2)
0
1
Similarities are not preserved

Hashing (con't)

Long codes are needed to preserve similarities.

Learn compact codes by using label information.

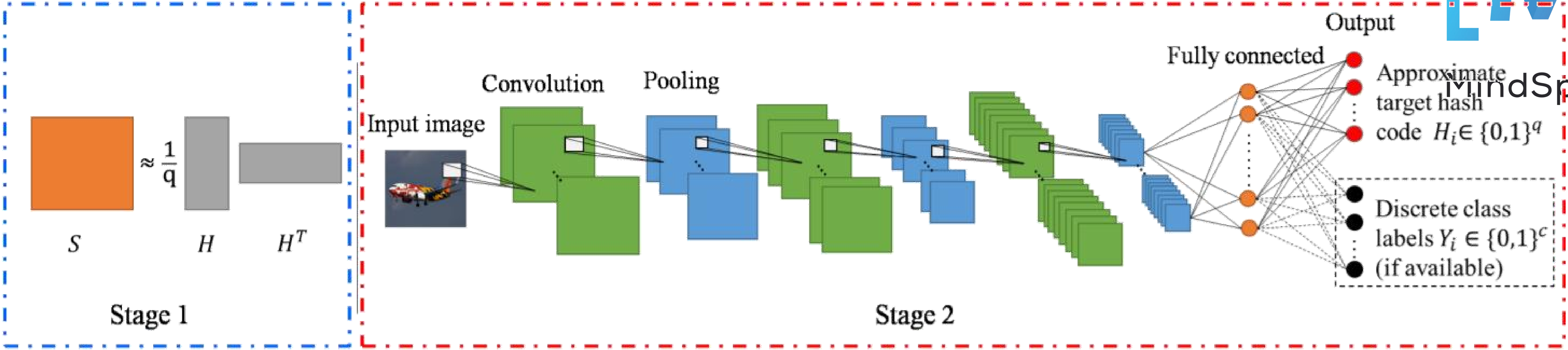


LSH[Gionis et al. VLDB,1999]
KLSH[Kulis and Grauman. PAMI,2012]
SH[Weiss and Torralba. NIPS,2008]
ITQ[Gong and Lazebnik. CVPR,2011]

SSH[Wang et al. CVPR,2010]

MLH[Norouzi and Blei. ICML,2011]
BRE[Kulis and Darrell. NIPS,2009]
KSH[Liu et al. CVPR,2012]
TSH[Lin et al. ICCV,2013]

Deep Hashing



1. **Learn approximate hash codes for the training samples**, i.e., the pairwise similarity matrix S is decomposed into a product $S = \frac{1}{q}HH^T$ where the ith row in **H** is the approximate hash code for the ith training image

$$\min_{H} \|S - \frac{1}{q}HH^T\|_F^2 \quad s.t.\ H \in [-1,1]^{n \times q}.\ S_{ij} = \begin{cases} +1, & I_i,\ I_j \text{ are semantically similar} \\ -1, & I_i,\ I_j \text{ are semantically dissimilar.} \end{cases}$$

2. **By using the learnt H and the raw image pixels as input, learn image representation and hash functions via *deep convolutional neural networks*.**

Figures are courtesy of [R. Xia, Y. Pan, et al., AAAI 2014]

Fine-grained hashing

Fine-grained image retrieval

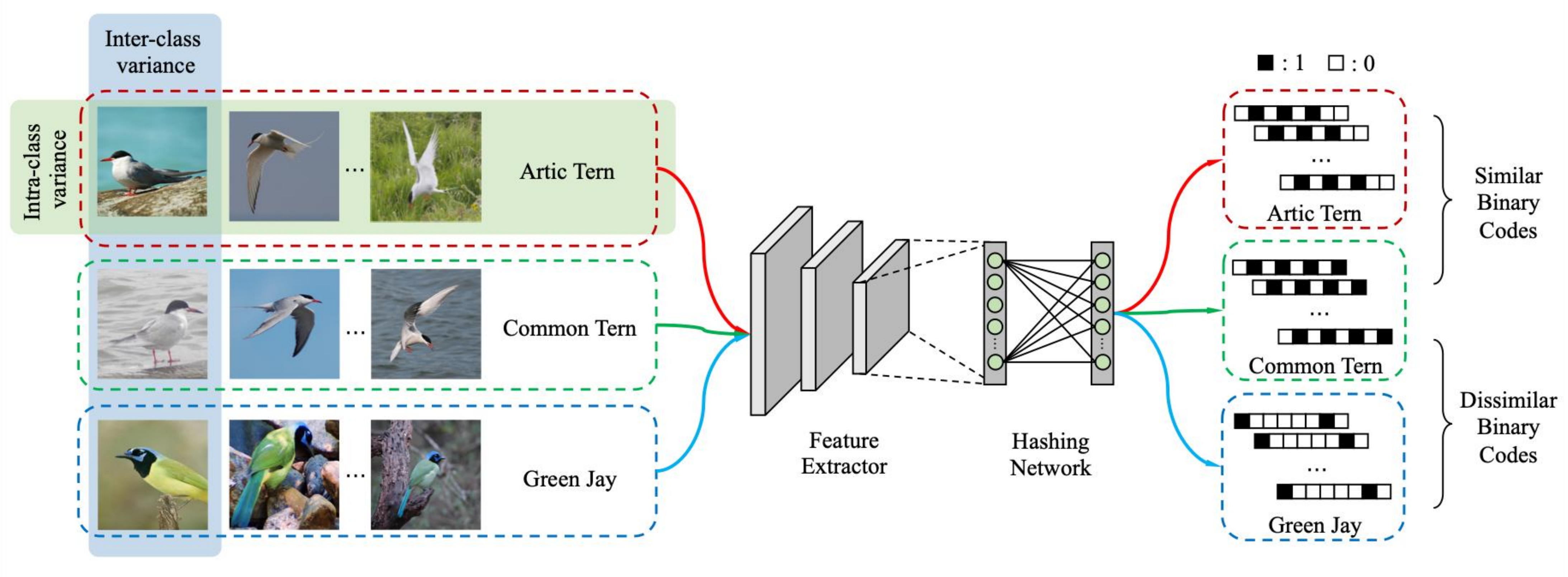
Illustrations of fine-grained image retrieval. Two examples ("Mallard" and "Rolls-Royce Phantom Sedan 2012") from the *CUB200-2011* and *Cars* datasets, respectively.

Figures are courtesy of [Wei et al., IEEE TIP 2017]

Fine-grained hashing (con't)

MindSpore

Large-scale fine-grained retrieval requires efficiency …

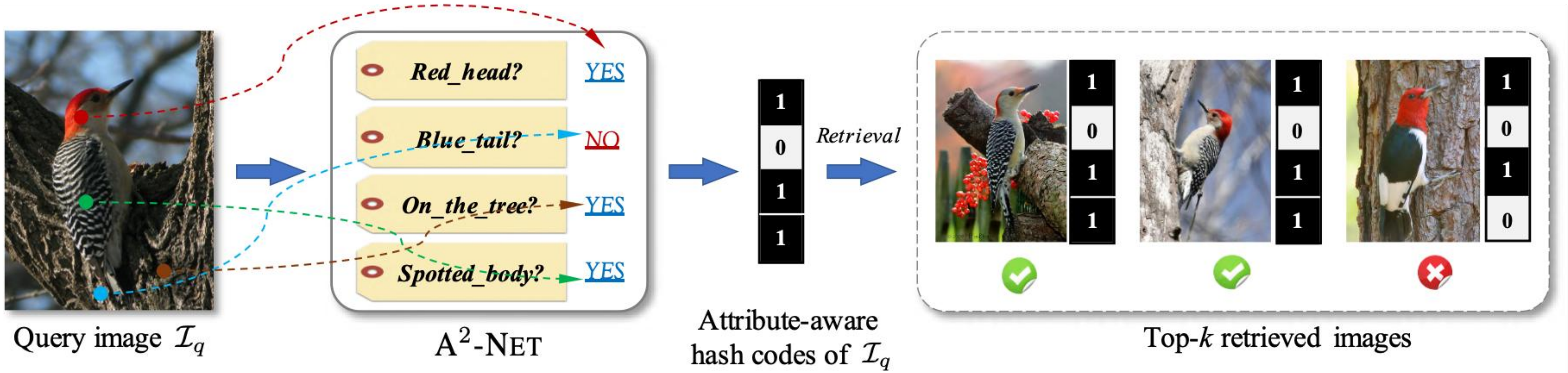


Figures are courtesy of [Cui et al., ECCV 2020]

Our works (NIPS 2021)

MindSpore

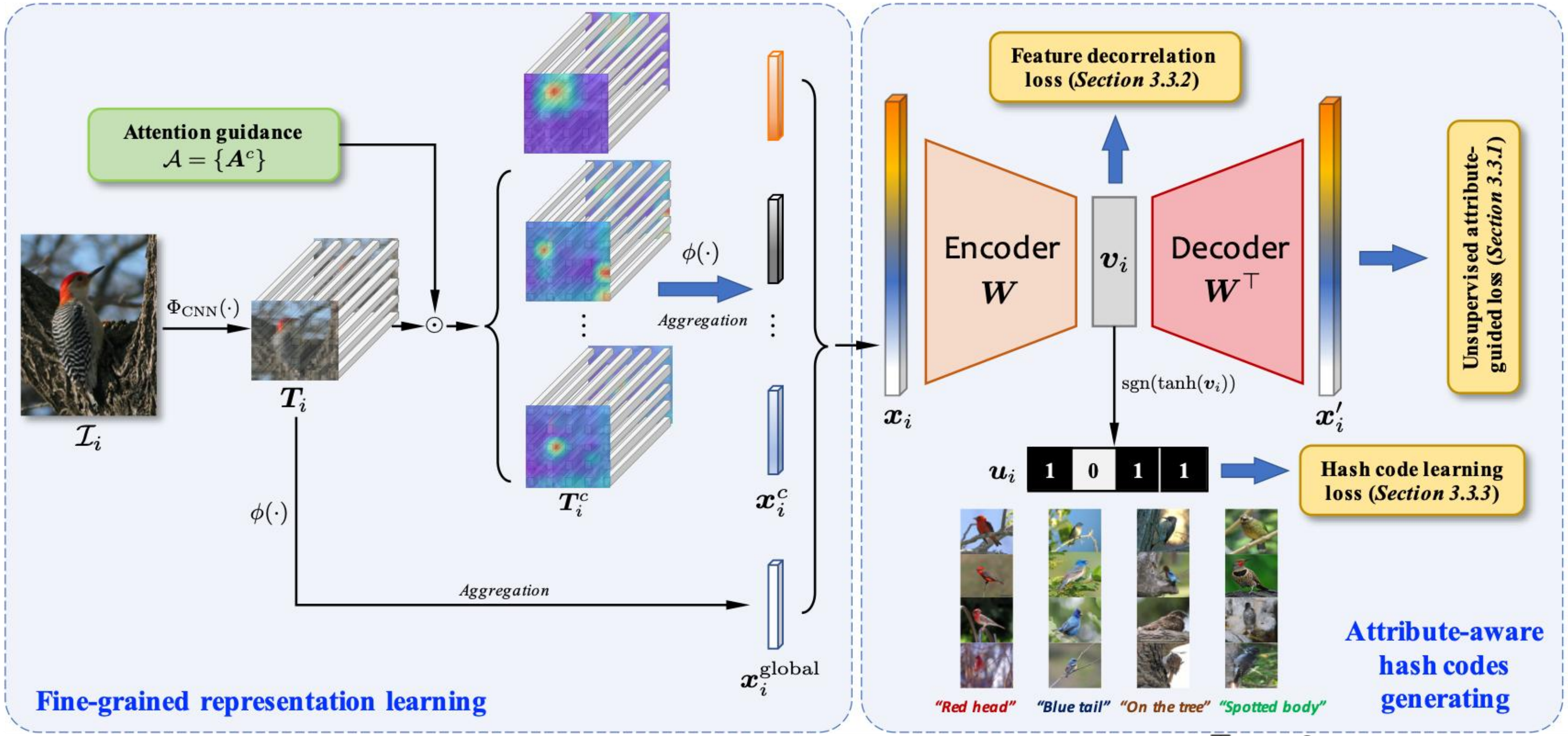
Attribute-aware correspondences are expected …



A² -NET: Learning Attribute-Aware Hash Codes for Large-Scale Fine-Grained Image Retrieval

Our works (con't)

A²-Net (Attribute-Aware hashing Network)



$$\min_{\boldsymbol{W},\boldsymbol{\Theta}} \mathcal{L}(\mathcal{I}) = \|\boldsymbol{X} - \boldsymbol{W}^\top \boldsymbol{V}'\|_F^2 + \lambda\|\boldsymbol{W}\boldsymbol{X} - \boldsymbol{V}'\|_F^2 + \alpha\|\boldsymbol{V}'\boldsymbol{V}'^\top - n\boldsymbol{I}\|_F^2$$

$$+ \beta \sum_{i\in\Omega}\sum_{j\in\Gamma}\left(\tanh(\boldsymbol{W}\cdot F(\mathcal{I}_i;\Theta))^\top \boldsymbol{z}_j - kS_{ij}\right)^2 .$$

A² -NET: Learning Attribute-Aware Hash Codes for Large-Scale Fine-Grained Image Retrieval

Our works (con't)

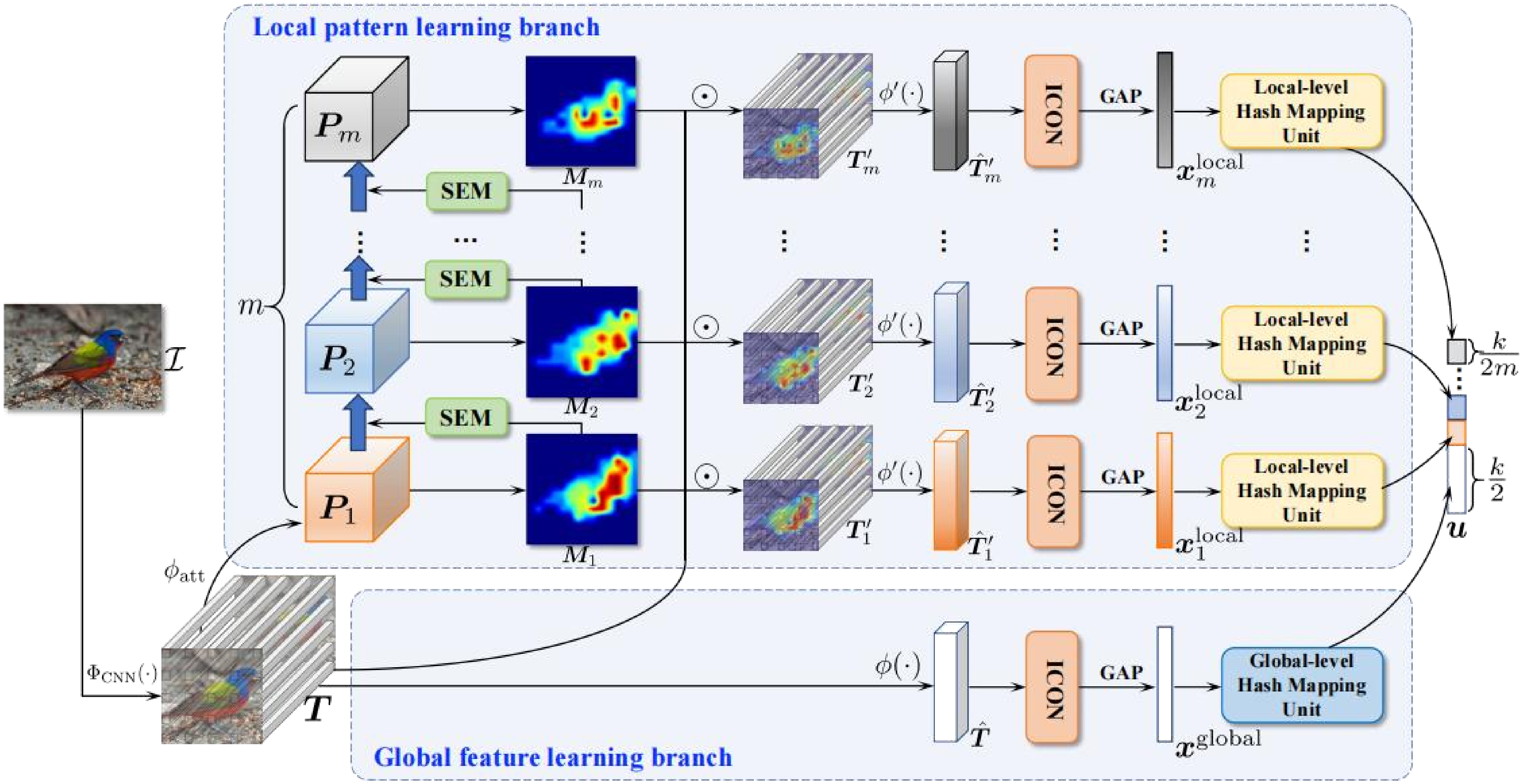
MindSpore

Quality demonstrations of the learned hash codes

Attribute-aware!

A^2 -NET: Learning Attribute-Aware Hash Codes for Large-Scale Fine-Grained Image Retrieval

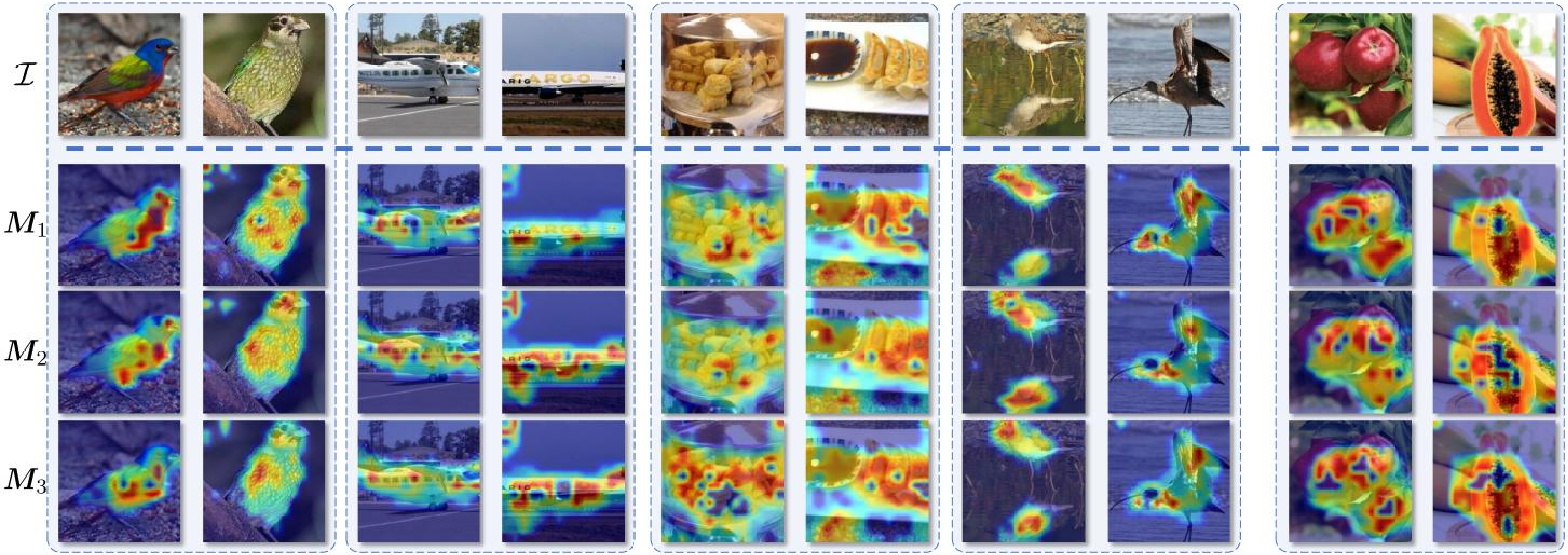
Our works (ECCV 2022)



SEMICON: A Learning-to-hash Solution for Large-scale Fine-grained Image Retrieval

Our works (con't)

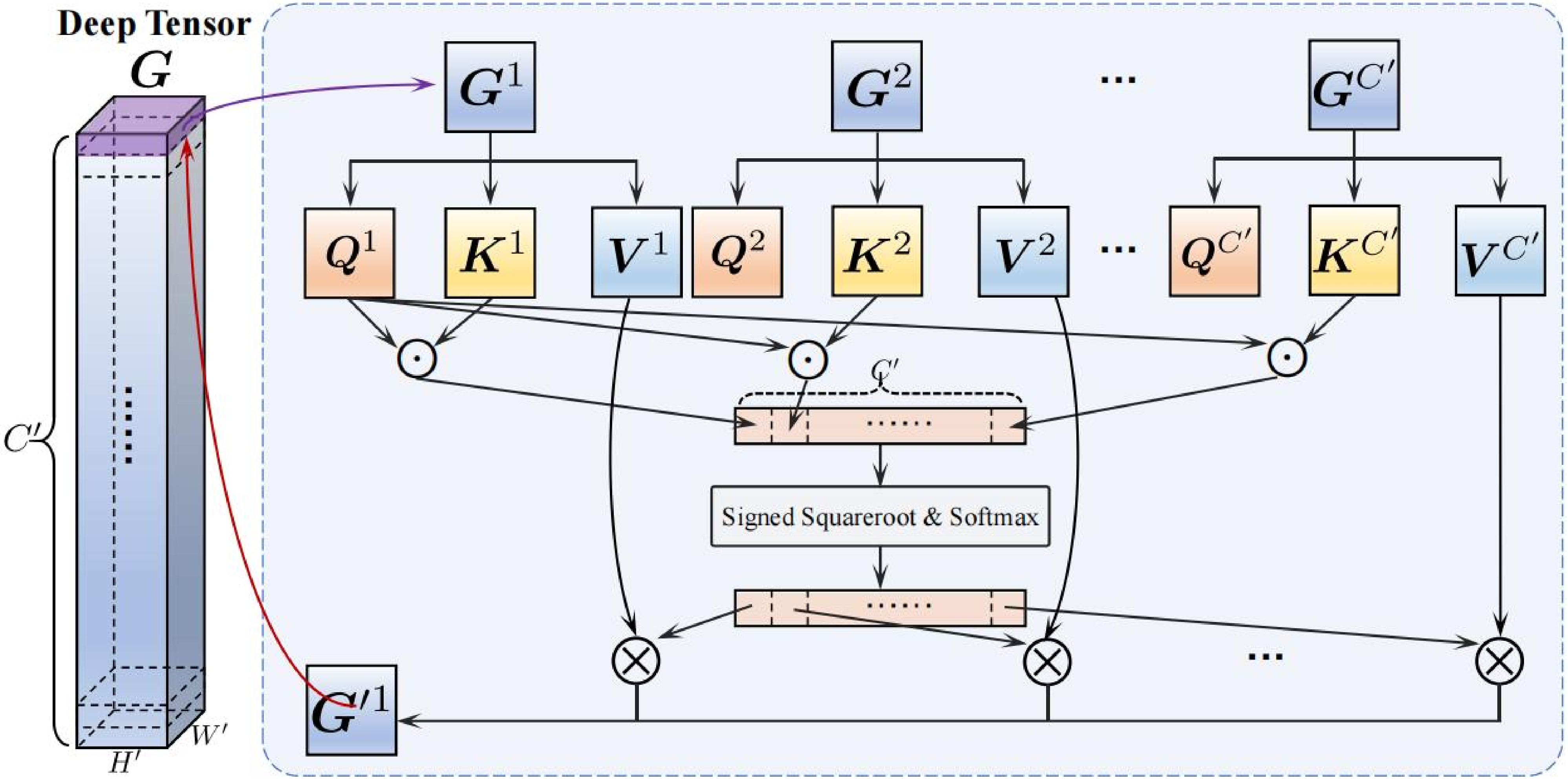
MindSpore



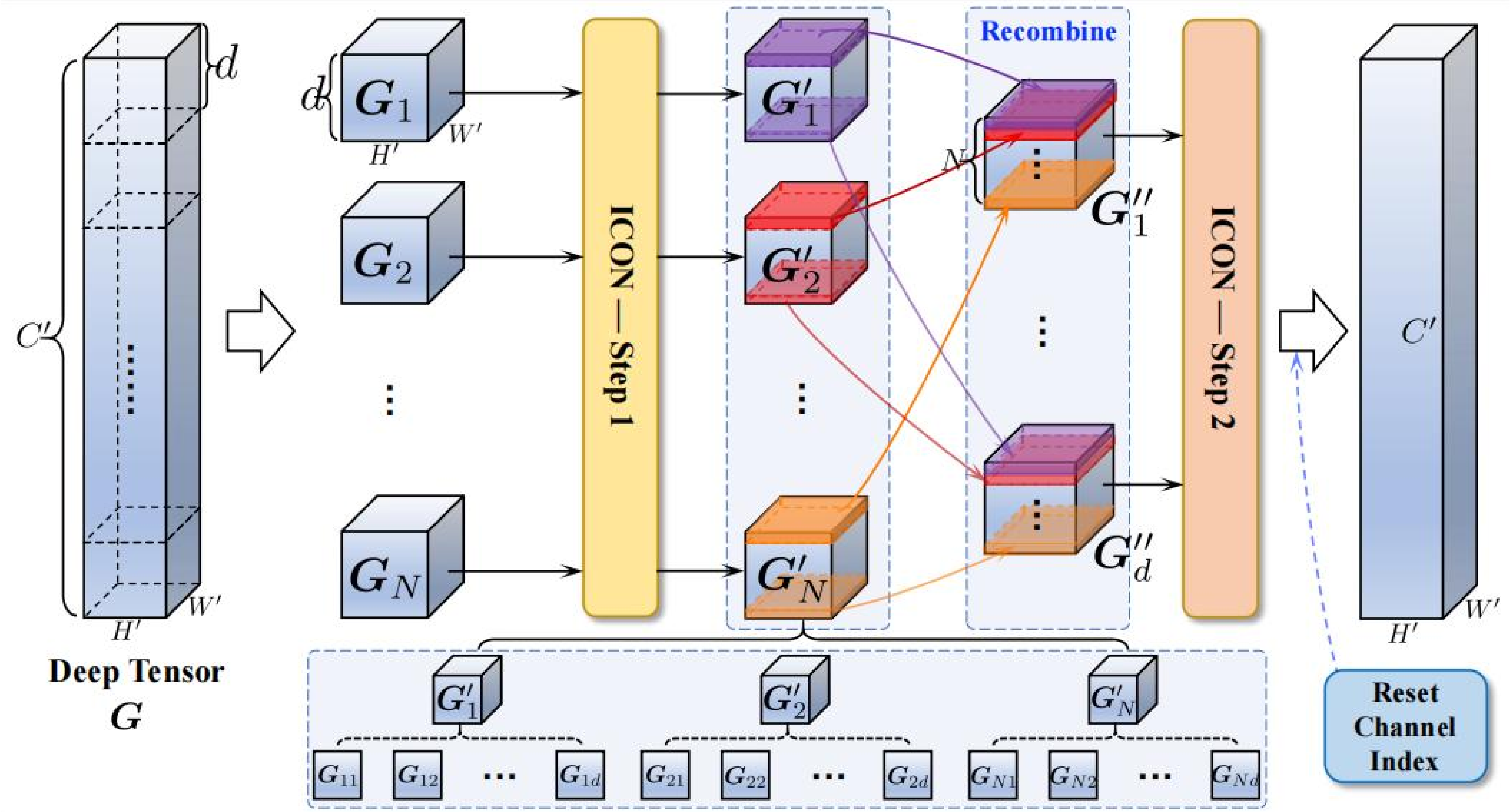
(a) *CUB200-2011* (b) *Aircraft* (c) *Food101* (d) *NABirds* (e) *VegFru*

SEMICON:A Learning-to-hash Solution for Large-scale Fine-grained Image Retrieval

Our works (con't)



Our works (con't)



SEMICON: A Learning-to-hash Solution for Large-scale Fine-grained Image Retrieval

Code with MindSpore

dSpore

```
class CUB200(object):
    def __init__(self, root, train=True):
        self.root = root
        self.train = train
        image_path = os.path.join(root, 'images')
        id2name = np.genfromtxt(os.path.join(root, 'images.txt'), dtype=str)
        id2train = np.genfromtxt(os.path.join(root, 'train_test_split.txt'), dtype=int)
        self.images = []
        self.labels = []
        target = 1 if train else 0
        for i in range(id2name.shape[0]):
            if id2train[i, 1] != target:
                continue
            label = int(id2name[i, 1][:3]) - 1
            self.images.append(os.path.join(image_path, id2name[i, 1]))
            self.labels.append(label)

    @staticmethod
    def smallest_max_size(x, max_size):
        def py3round(number):
            if abs(round(number) - number) == 0.5:
                return int(2.0 * round(number / 2.0))
            return int(round(number))

        h, w = x.shape[:2]
        scale = max_size / float(min(h, w))
        h, w = tuple(py3round(dim * scale) for dim in (h, w))
        x = cv2.resize(x, (h, w), interpolation=cv2.INTER_LINEAR)
        return x
```

MindSpore 1.3

mindspore.dataset.vision.c_transforms.AutoContrast

mindspore.dataset.vision.c_transforms.BoundingBoxAugment

mindspore.dataset.vision.c_transforms.CenterCrop

mindspore.dataset.vision.c_transforms.ConvertColor

mindspore.dataset.vision.c_transforms.Crop

mindspore.dataset.vision.c_transforms.CutMixBatch

mindspore.dataset.vision.c_transforms.CutOut

mindspore.dataset.vision.c_transforms.Decode

mindspore.dataset.vision.c_transforms.Equalize

mindspore.dataset.vision.c_transforms.GaussianBlur

MindSpore 1.7

Code with MindSpore (con’t)

以ECCV 2022的工作为例，采用MindSpore搭建模型

Code with MindSpore (con't)

MindSpore

```python
class ChannelTransformer(nn.Cell):
    def __init__(self, dim, num_heads):
        super().__init__()
        self.num_heads = num_heads
        head_dim = dim // num_heads
        self.scale = head_dim ** -0.5
        self.head_dim = head_dim
        self.norm = nn.BatchNorm2d(dim)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(axis=-1)
        self.qkv = nn.Conv2d(dim, dim * 3, 1, group=num_heads)
        self.qkv2 = nn.Conv2d(dim, dim * 3, 1, group=head_dim)

    def construct(self, x):
        B, C, H, W = x.shape
        qkv = self.qkv(x).reshape(B, 3, self.num_heads, self.head_dim, H * W).transpose(1,0,2,3,4)
        q, k, v = qkv[0], qkv[1], qkv[2]
        attn = ms.numpy.matmul(q, k.transpose(0,1,3,2)) * self.scale
        attn = ms.numpy.sign(attn) * ms.numpy.sqrt(ms.numpy.abs(attn) + 1e-5)
        attn = self.softmax(attn)
        x = (ms.numpy.matmul(attn, v).reshape(B, C, H, W) + x).reshape(B, self.num_heads, self.head_dim,
        y = self.norm(x)
        x = self.relu(y)
        qkv2 = self.qkv2(x).reshape(B, 3, self.head_dim, self.num_heads, H * W).transpose(1,0,2,3,4)
        q, k, v = qkv2[0], qkv2[1], qkv2[2]
        attn = (ms.numpy.matmul(q, k.transpose(0,1,3,2))) * (self.num_heads ** -0.5)
        attn = ms.numpy.sign(attn) * ms.numpy.sqrt(ms.numpy.abs(attn) + 1e-5)
        attn = self.softmax(attn)
        x = ms.numpy.matmul(attn, v).reshape(B, self.head_dim, self.num_heads, H, W).transpose(0,2,1,3,4
        return x
```

MindSpore

```python
class ChannelTransformer(nn.Module):
    def __init__(self, dim, num_heads):
        super().__init__()
        self.num_heads = num_heads
        head_dim = dim // num_heads
        self.scale = head_dim ** -0.5
        self.head_dim = head_dim
        self.norm = nn.BatchNorm2d(dim)
        self.relu = nn.ReLU(inplace=True)
        self.qkv = nn.Conv2d(dim, dim * 3, 1, groups=num_heads)
        self.qkv2 = nn.Conv2d(dim, dim * 3, 1, groups=head_dim)

    def forward(self, x):
        B, C, H, W = x.shape
        qkv = self.qkv(x).reshape(B, 3, self.num_heads, self.head_dim, H * W).transpose(0, 1)
        q, k, v = qkv[0], qkv[1], qkv[2]
        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = torch.sign(attn) * torch.sqrt(torch.abs(attn) + 1e-5)
        attn = attn.softmax(dim=-1)
        x = ((attn @ v).reshape(B, C, H, W) + x).reshape(B, self.num_heads, self.head_dim, H, W).transp
        y = self.norm(x)
        x = self.relu(y)
        qkv2 = self.qkv2(x).reshape(B, 3, self.head_dim, self.num_heads, H * W).transpose(0, 1)
        q, k, v = qkv2[0], qkv2[1], qkv2[2]
        attn = (q @ k.transpose(-2, -1)) * (self.num_heads ** -0.5)
        attn = torch.sign(attn) * torch.sqrt(torch.abs(attn) + 1e-5)
        attn = attn.softmax(dim=-1)
        x = (attn @ v).reshape(B, self.head_dim, self.num_heads, H, W).transpose(1, 2).reshape(B, C, H,
        return x
```

PyTorch

THANK YOU