**This document will use the VGG13 network as an example to introduce the training process of icCNN for single-category and multi-category classification tasks. We also illustrate how to generate and show the feature maps.**

**1.Preparing Datasets**

After downloading the code from the Github, we need modify the corresponding dataset path parameters in **vgg\_ori\_train.py**, **vgg\_iccnn\_train.py** and **vgg\_iccnn\_multi\_train.py**, respectively. The modified paths are shown as following:

cub\_file = '../../datasets/frac\_dataset'  
voc\_file = '../../datasets/VOCdevkit/VOC2010/voc2010\_crop'  
celeb\_file = '../../datasets/celeba'

where cub\_file, voc\_file, celeb\_file represent the **CUB**, **PascalVOC** and **CelebA** datasets, respectively.

**If you do not know how to download the dataset voc\_crop, or other datasets, please download from here: https://pan.baidu.com/s/1RlCfjiJCdGxI8fGCpeClKQ?pwd=87ps**

We can use the **tree** command to check the files:

iMoonKeyBoy: pwd

/home/~/Data/PycharmWorkspace/datasets/VOCdevkit/VOC2010/voc2010\_crop

iMoonKeyBoy: tree -L 1

.

├── airplane\_aug\_info.txt

├── airplane\_info.txt

├── all\_crop

├── animal

├── animal\_info.txt

├── animal\_iou.txt

├── bird\_info.txt

├── cat\_info.txt

├── cow\_info.txt

├── dog\_info.txt

├── generate\_info.py

├── helen\_dataset

├── helen\_info.txt

├── horse\_info.txt

├── huohua\_dataset

├── image\_info.txt

├── sheep\_info.txt

├── ship\_aug\_info.txt

└── ship\_info.txt

**2. Training Step**

**2.1 For single-category classification**

**2.1.1. Train the original model**

**Firstly, we need set reasonable hyper-parameters in vgg\_ori\_train.py.** Since we train the single-category model, we need set the **hyper-parameters in vgg\_ori\_train.py** as following:

IS\_TRAIN = 1 # 1 denotes run the training process, while 0 denotes this py file just run the feature extraction process. So here we need set this to 1.  
IS\_MULTI = 0 # 1 denotes run the multi-category classification, while 0 denotes single-category classification. So here we need set this to 0.  
  
  
LAYERS = '13' # The type of VGG model. Here we select Vgg13 model.

DATANAME = bird' # Choose one category for our single-category classification process. Here we select ‘bird’ as demo. If we need train the multi-category model, please set this to ‘voc\_multi’

# Other hyper-parameters: those parameters we set as default  
NUM\_CLASSES = 6 if IS\_MULTI else 2  
if DATANAME == 'celeb':  
 NUM\_CLASSES = 80

BATCHSIZE = 1  
LR = 0.000001  
EPOCH = 200

save\_path = './icCNN/basic\_fmap/vgg/' # for get\_feature  
acc\_path = './icCNN/basic\_fmap/vgg/acc/'

log\_path = './icCNN/vgg/'

dataset = '%s\_vgg\_%s\_ori' % (LAYERS, DATANAME)

log\_path = log\_path + dataset + '/'  
pretrain\_model = None # Here, since we do not need any pre-trained model, we set this to None.

After setting all above hyper-parameters, we can use the following command to train the original model.

python3 train\_all.py -type ori -is\_multi 0 -model vgg

After training process, the trained model will be saved in **log\_path.**

**2.1.2. Generate feature maps of the original model.**

As we discussed above, we need change the hyper-parameters in **vgg\_ori\_train.py** to generation process, rather than training process**:**

IS\_TRAIN = 0 # feature map generation process

# Set the model path for generation:

dataset = '%s\_vgg\_%s\_iccnn' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = log\_path + 'model\_200.pth' #using pre-trained model

And using the following command to generate the feature maps.

python3 train\_all.py -type ori -is\_multi 0 -model vgg

After this command, it will achieve the feature maps and the accuracy file in **save\_path** and **acc\_path**.

**2.1.3. Train the corresponding ICCNN model**

The training process is based on the model that achieved in original training process. Therefore, we need set the hyper-parameters **pretrain\_model** in **vgg\_iccnn\_train.py** to pre-trained model**.** The total parameters are shown as following:

IS\_TRAIN = 1 # setting training process  
  
LAYERS = '13'  
DATANAME = 'bird'  
NUM\_CLASSES = 80 if DATANAME == 'celeb' else 2  
  
cub\_file = '../../datasets/frac\_dataset'  
voc\_file = '../../datasets/VOCdevkit/VOC2010/voc2010\_crop'  
celeb\_file = '../../datasets/CelebA'  
  
save\_path = './icCNN/basic\_fmap/vgg/'  
acc\_path = './icCNN/basic\_fmap/vgg/acc/'  
  
log\_path = './icCNN/vgg/'   
dataset = '%s\_vgg\_%s\_ori' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = log\_path + 'model\_200.pth'# The model that we generated based on vgg\_ori\_train.py file

BATCHSIZE = 1  
LR = 0.00001  
EPOCH = 2500

center\_num = 5 # denotes the number of convolutional kernel groups  
lam = 0.1 # denotes the weight before similarity loss  
T = 2 # denotes the T round for one clustering

F\_MAP\_SIZE = 196 # denotes the size of the feature map   
STOP\_CLUSTERING = 200 # denotes the epoch position to stop clustering

# CHANNEL\_NUM denotes the number of channels of the feature map  
if LAYERS == '13':  
 CHANNEL\_NUM = 512  
elif LAYERS == '16':  
 CHANNEL\_NUM = 512

And using the following command to train the corresponding ICCNN model.

python3 train\_all.py -type iccnn -is\_multi 0 -model vgg

**2.1.4. Generate feature maps of the single-category iccnn model.**

Similarly, we need change the hyper-parameters as following in **vgg\_iccnn\_train.py:**

IS\_TRAIN = 0

# Set the model path for generation:

dataset = '%s\_vgg\_%s\_iccnn' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = log\_path + 'model\_2500.pth' # using the pre-trained model.

And using the following command to generate the feature maps.

python3 train\_all.py -type iccnn -is\_multi 0 -model vgg

After this command, it will achieve the feature maps and the accuracy file in **save\_path** and **acc\_path**.

**2.2 For multi-category classification**

**2.2.1. Train the original model**

**Similarly, we need set reasonable hyper-parameters in vgg\_ori\_train.py.** Here, we need to train the multi-category model:

IS\_TRAIN = 1   
IS\_MULTI = 1   
  
  
LAYERS = '13'

DATANAME = 'voc\_multi'

# Other hyper-parameters: those parameters we set as default  
NUM\_CLASSES = 6 if IS\_MULTI else 2  
if DATANAME == 'celeb':  
 NUM\_CLASSES = 80

BATCHSIZE = 1  
LR = 0.000001  
EPOCH = 200

save\_path = './icCNN/basic\_fmap/vgg/' # for get\_feature  
acc\_path = './icCNN/basic\_fmap/vgg/acc/'

log\_path = './icCNN/vgg/'

dataset = '%s\_vgg\_%s\_iccnn' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = None

After setting all above hyper-parameters, we can use the following command to train the original model.

python3 train\_all.py -type ori -is\_multi 1 -model vgg

After training process, the trained model will be saved in **log\_path.**

**2.2.2. Generate feature maps for original multi-category model.**

Similarly, we need change the hyper-parameters as following in **vgg\_ori\_train.py:**

IS\_TRAIN = 0

IS\_MULTI = 1

# Set the model path for generation:

dataset = '%s\_vgg\_%s\_iccnn' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = log\_path + 'model\_200.pth' #using the pre-trained model.

And using the following command to generate the feature maps.

python3 train\_all.py -type ori -is\_multi 1 -model vgg

After this command, it will achieve the feature maps and the accuracy file in **save\_path** and **acc\_path**.

**2.2.3. Train the corresponding ICCNN model**

Similarly, we need set the hyper-parameters **pretrain\_model** in **vgg\_iccnn\_multi\_train.py** to pre-trained model**.** The total parameters are shown as following:

IS\_TRAIN = 1

IS\_MULTI = 1

LAYERS = '13'  
DATANAME = 'voc\_multi'

NUM\_CLASSES = 80 if DATANAME == 'celeb' else 2  
  
cub\_file = '../../datasets/frac\_dataset'  
voc\_file = '../../datasets/VOCdevkit/VOC2010/voc2010\_crop'  
celeb\_file = '../../datasets/CelebA'  
  
save\_path = './icCNN/basic\_fmap/vgg/'  
acc\_path = './icCNN/basic\_fmap/vgg/acc/'

log\_path = './icCNN/vgg/'   
dataset = '%s\_vgg\_%s\_ori' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = log\_path + 'model\_200.pth'# The model that we generated based on vgg\_ori\_train.py file with IS\_MULTI = 1

BATCHSIZE = 1   
LR = 0.00001  
EPOCH = 1000

center\_num = 16 # denotes the number of convolutional kernel groups  
lam1 = 0.1 # denotes the weight before the similarity loss  
lam2 = 0.1 # denotes the weight before the multi-category loss  
  
T = 2 # denotes the T round for one clustering  
F\_MAP\_SIZE = 196 # denotes the size of the feature map  
STOP\_CLUSTERING = 200 # denotes the epoch position to stop clustering

# denotes the number of channels of the feature map.   
if LAYERS == '13':  
 CHANNEL\_NUM = 512  
elif LAYERS == '16':  
 CHANNEL\_NUM = 512  
elif LAYERS == '19':  
 CHANNEL\_NUM = 512

And using the following command to train the corresponding multi-category ICCNN model

python3 train\_all.py -type iccnn -is\_multi 1 -model vgg

**2.2.4. Generate feature maps for multi-category iccnn model.**

Similarly, we need change the hyper-parameters as following in **vgg\_iccnn\_multi\_train.py:**

IS\_TRAIN = 0

IS\_MULTI = 1

# Set the model path for generation:

dataset = '%s\_vgg\_%s\_iccnn' % (LAYERS, DATANAME)  
log\_path = log\_path + dataset + '/'  
pretrain\_model = log\_path + 'model\_1000.pth' # using the model in 2.2.3

And using the following command to generate the feature maps.

python3 train\_all.py -type ori -is\_multi 1 -model vgg

After this command, it will achieve the feature maps and the accuracy file in **save\_path** and **acc\_path**.

**3.Show Feature maps**

parser.add\_argument('-Th', type=int, default=0.2) # denotes which pix will be showed in red circle, We can set it as default.  
parser.add\_argument('-factor', type=int, default=0.5). # denote the transparency of the red circle, We can set it as default.

parser.add\_argument('-show\_num', type=int, default=10) # denote the number of filters in each group to show  
parser.add\_argument('-model', default="resnet", type=str) # model name   
parser.add\_argument('-animal', default="bird",type=str) # the type to show, you can find the information in the following array: animal.  
parser.add\_argument('-fmap\_path', type=str) # the path for npz file of feature maps.   
parser.add\_argument('-loss\_path' type=str) # used to choose filters  
parser.add\_argument('-folder\_name', default=None, type=str)# the folder for save

Also, we should prepare the Images that we want to show. It should extract some JPG image from test\_dataset, and rename those images from **vocbird\_0.jpg** to vocbird\_n.jpg, where *n* is the number you want to test.

After that, using the following command to show the feature maps.

python3 draw\_fmap.py \

-model vgg -animal cub \

-folder\_name 16\_vgg\_cub\_iccnn\_e-6 \

-fmap\_path ./npz/vgg16\_cub/16\_vgg\_cub\_iccnn\_e-6.npz \

-loss\_path ./npz/vgg16\_cub/loss\_1000.npz

After this command, it will achieve the feature maps with mask in **folder\_name**.