大作业说明

1. 数据集介绍

1.1 ImageNet-1K 数据集介绍

ImageNet数据是CV领域非常出名的数据集，ISLVRC竞赛使用的数据集是轻量版的ImageNet数据集。在很多CV领域的论文，都会使用这个数据集对自己的模型进行测试。ImageNet是一个计算机视觉系统识别项目，是目前世界上图像识别最大的数据库。是美国斯坦福的计算机科学家，模拟人类的识别系统建立的。能够从图片中识别物体，超过1400万的图像URL被ImageNet手动注释，以指示图片中的对象;在至少一百万张图像中，还提供了边界框。ImageNet包含2万多个类别;一个典型的类别，如“气球”或“草莓”，每个类包含数百张图像。在一些论文中，有的人会将这个数据叫成ImageNet 1K或者ISLVRC2012，两者是一样的。“1 K”代表的是1000个类别。

1.2 ImageNet-1K-16shot数据集介绍

- 本次Project中，会用到ImageNet 1K 中的少量样本，即训练集包括1000个类别，每个类别只提供至多16个样本，总样本数目为16000，测试集还是和原来ImageNet 1K的测试集一样。
- ImageNet-1K-16shot包含两个文件夹，为train 和 val，文件夹的结构如下所示：

```
ImageNet-1K-16shot/
|- - train/ # contains 1,000 folders like , n01443537, etc.
|-- val/ # contains the same folders as train
|-- classnames.txt / # contains 1,000 lines like n01440764 tench, etc.
|-- label2class.json / # contains the relationship between label and class name.
```

- 本次Project使用train文件夹里面的数据当作训练集，val文件夹里面的数据当作测试集，包含总计50,000张图像；如果有需要用到类别名字，可以使用classname.txt进行名字转换。
- 数据集下载链接：ImageNet-1K-16shot (https://zjueducn-my.sharepoint.com/:u:/g/personal/lms_zju_edu_cn/Efyj9dbGQcVPu_BMNtveWfgB2SrncMIi9u8_bNpO5xYOWQ?e=etGF1J) 密码: zju_summer_07

In [1]:
```
import os
import json
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
root_path= '/data/base-to-novel/' # Your root folder

data_path = os.path.join(root_path, 'ImageNet-1K-16shot')
print(os.listdir(data_path))
```

```
['label2class.json', 'val', 'classnames.txt', 'train']
```

In [7]:
```
# 查看训练集的数据和图像
train_path = os.path.join(data_path, 'train')
test_path = os.path.join(data_path, 'val')
train_folders = os.listdir(train_path)
print('Train folders: ', train_folders)

# 文件夹名字有对应的类别名字，可以从classnames.txt 中读取，存成字典的格式
with open(os.path.join(data_path, 'classnames.txt'), 'r') as f:
    data = f.readlines()
    data = [d.strip().replace('/', 'and').split(" ") for d in data]
    print(data)
    folders_to_names = {d[0]:' '.join(d[1:]) for d in data}
    print(folders_to_names)

# label2class json 文件读取
with open(os.path.join(data_path, 'label2class.json'), 'r') as f:
    json_data = json.load(f)
    print(json_data)
```

```
Train folders:  ['n02965783', 'n02037110', 'n02500267', 'n03384352', 'n03742115', 'n02095570', 'n01694178', 'n03920288', 'n04409515', 'n
03201208', 'n03937543', 'n02129604', 'n03680355', 'n03291819', 'n03075370', 'n03944341', 'n02128385', 'n01843065', 'n03095699', 'n034433
71', 'n07754684', 'n02111277', 'n03666591', 'n02444819', 'n03803284', 'n03085013', 'n02106382', 'n01644900', 'n02129165', 'n04209239',
'n02966193', 'n01665541', 'n04487081', 'n04239074', 'n04033995', 'n03792972', 'n02006656', 'n02412080', 'n02100877', 'n13054560', 'n0297
4003', 'n01819313', 'n03837869', 'n02089867', 'n03977966', 'n02999410', 'n04552348', 'n02099267', 'n02168699', 'n02123045', 'n02281406',
'n04154565', 'n03179701', 'n01773797', 'n03028079', 'n02002556', 'n02840245', 'n02422106', 'n02138441', 'n02088632', 'n04404412', 'n0417
9913', 'n02100735', 'n02326432', 'n03250847', 'n04366367', 'n04517823', 'n04332243', 'n02108000', 'n01773549', 'n02786058', 'n04008634',
'n03124170', 'n02098413', 'n03733281', 'n04067472', 'n02137549', 'n04069434', 'n02422699', 'n02115641', 'n01847000', 'n07695742', 'n0403
3901', 'n02107574', 'n04336792', 'n03902125', 'n03461385', 'n02018795', 'n02916936', 'n02123597', 'n04612504', 'n03804744', 'n04023962',
'n03770679', 'n01729322', 'n02655020', 'n01592084', 'n02167151', 'n02113712', 'n03929660', 'n01614925', 'n02497673', 'n03126707', 'n0435
5933', 'n04127249', 'n01582220', 'n02814533', 'n03998194', 'n07880968', 'n02233338', 'n03017168', 'n04604644', 'n03887697', 'n02493509',
'n02704792', 'n03733131', 'n04263257', 'n03272562', 'n07760859', 'n03710637', 'n02823428', 'n02012849', 'n01753488', 'n03146219', 'n0922
9709', 'n01693334', 'n01632777', 'n06874185', 'n02105505', 'n03133878', 'n03777568', 'n04344873', 'n02109961', 'n02837789', 'n02097474',
'n04465501', 'n04254120', 'n02791124', 'n01871265', 'n01622779', 'n04589890', 'n04542943', 'n02794156', 'n04613696', 'n02018207', 'n0213
0308', 'n01978455', 'n03127925', 'n04285008', 'n02917067', 'n04482393', 'n03476991', 'n04208210', 'n03888257', 'n03018349', 'n03452741',
'n01824575', 'n02102040', 'n03717622', 'n03450230', 'n03967562', 'n03445777', 'n02486261', 'n02110063', 'n01644373', 'n01744401', 'n0179
5545', 'n04086273', 'n13044778', 'n04456115', 'n07718747', 'n09835506', 'n02033041', 'n02398521', 'n01796340', 'n04041544', 'n02504013',
'n02134084', 'n02109047', 'n02814860', 'n02106662', 'n03825788', 'n01755581', 'n04443257', 'n01616318', 'n04389033', 'n01440764', 'n0298
8304', 'n01943899', 'n02172182', 'n09246464', 'n13052670', 'n04350905', 'n01695060', 'n02095314', 'n01748264', 'n03743016', 'n09421951',
```

2. CLIP 模型介绍

2.1 CLIP (Contrastive Language-Image Pre-Training)

- CLIP（对比语言-图像预训练）是一种在各种（图像、文本）对上训练的神经网络。通过大规模图像-文本对的预训练，CLIP有着很强大的下游任务泛化能力，可以在下游任务不提供任何样本进行训练的情况下实现零样本推理。
- 本次 Project 提供 CLIP 进行零样本推理的代码教程，以及在ImageNet-1K-16shot数据集上的零样本性能。
- CLIP参考网址 (https://github.com/openai/CLIP/tree/main)

In [8]:
```
import numpy as np
import torch
from pkg_resources import packaging

print("Torch version:", torch.__version__)
```

```
Torch version: 1.12.0+cu113
```

2.2 CLIP 模型导入

在2.2部分，会介绍如何导入CLIP模型以及计算图像文本相似度

Loading the model

`clip.available_models()` 列出可用 CLIP 模型的名称， `RN50` 指的ResNet。

In [9]:
```
import clip
clip.available_models()
```

Out[9]:
```
['RN50', 'RN101', 'RN50x4', 'RN50x16', 'ViT-B/32', 'ViT-B/16']
```

In [10]:
```
model, preprocess = clip.load("RN50")

# 可以设定为 cpu的格式在自己电脑运行
model.cuda().eval()

input_resolution = model.visual.input_resolution
context_length = model.context_length
vocab_size = model.vocab_size

print("Model parameters:", f"{np.sum([int(np.prod(p.shape)) for p in model.parameters()]):,}")
print("Input resolution:", input_resolution)
print("Context length:", context_length)
print("Vocab size:", vocab_size)
```

```
Model parameters: 102,007,137
Input resolution: 224
Context length: 77
Vocab size: 49408
```

Image Preprocessing

我们调整输入图像的大小并对它们进行中心裁剪，以符合模型期望的图像分辨率。 在此之前，我们将使用数据集平均值和标准差对像素强度进行标准化。

`clip.load()` 的第二个返回值 preprocess 包含执行此预处理的 torchvision Transform。

In [11]:
```
preprocess
```

Out[11]:
```
Compose(
    Resize(size=224, interpolation=bicubic, max_size=None, antialias=None)
    CenterCrop(size=(224, 224))
    <function _transform.<locals>.<lambda> at 0x7f7cdef00af0>
    ToTensor()
    Normalize(mean=(0.48145466, 0.4578275, 0.40821073), std=(0.26862954, 0.26130258, 0.27577711))
)
```

Text Preprocessing

我们使用不区分大小写的分词器，可以使用clip.tokenize()调用它。 默认情况下，输出被填充为 77 个令牌长，这是 CLIP 模型所期望的文本输入。

In [12]:
```
clip.tokenize("Hello World!")
```

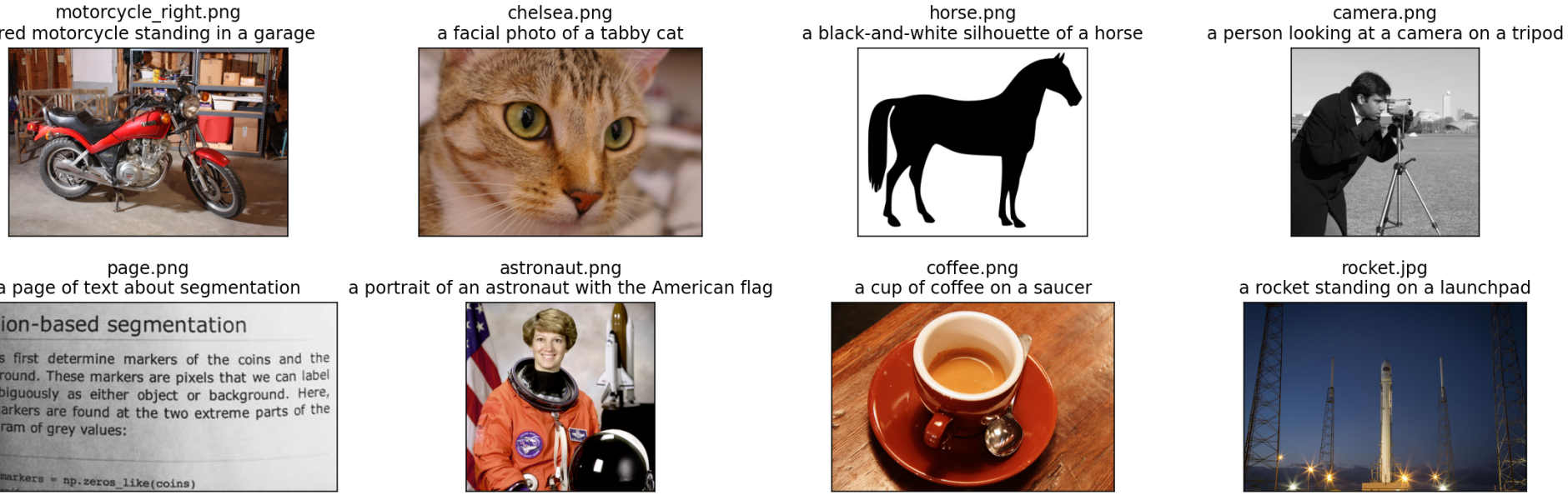
Out[12]:
```
tensor([[49406,  3306,  1002,   256, 49407,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0]])
```

Setting up input images and texts

在这个例子中，我们将向模型提供 8 个示例图像及其文本描述，并比较相应特征之间的相似度。分词器不区分大小写，我们可以自由地给出任何合适的文本描述。

In [14]:
```
import os
import skimage
import IPython.display
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

from collections import OrderedDict
import torch

%matplotlib inline
%config InlineBackend.figure_format = 'retina'

# images in skimage to use and their textual descriptions
descriptions = {
    "page": "a page of text about segmentation",
    "chelsea": "a facial photo of a tabby cat",
    "astronaut": "a portrait of an astronaut with the American flag",
    "rocket": "a rocket standing on a launchpad",
    "motorcycle_right": "a red motorcycle standing in a garage",
    "camera": "a person looking at a camera on a tripod",
    "horse": "a black-and-white silhouette of a horse",
    "coffee": "a cup of coffee on a saucer"
}

original_images = []
images = []
texts = []
plt.figure(figsize=(16, 5))

for filename in [filename for filename in os.listdir(skimage.data_dir) if filename.endswith(".png") or filename.endswith(".jpg")]:
    name = os.path.splitext(filename)[0]
    if name not in descriptions:
        continue

    image = Image.open(os.path.join(skimage.data_dir, filename)).convert("RGB")

    plt.subplot(2, 4, len(images) + 1)
    plt.imshow(image)
    plt.title(f"{filename}\n{descriptions[name]}")
    plt.xticks([])
    plt.yticks([])

    original_images.append(image)
    images.append(preprocess(image))
    texts.append(descriptions[name])
```

```
plt.tight_layout()
```



Building features

首先，我们对图像进行标准化处理和对每个文本输入进行标记，然后运行模型的forward过程来获取图像和文本特征。

In [15]:
```
image_input = torch.tensor(np.stack(images)).cuda()
text_tokens = clip.tokenize(["This is " + desc for desc in texts]).cuda()
```

In [16]:
```
with torch.no_grad():
    image_features = model.encode_image(image_input).float()
    text_features = model.encode_text(text_tokens).float()
```

Calculating cosine similarity

计算图像和文本特征之间的相似度。

In [17]:
```
image_features /= image_features.norm(dim=-1, keepdim=True)
text_features /= text_features.norm(dim=-1, keepdim=True)
similarity = text_features.cpu().numpy() @ image_features.cpu().numpy().T

count = len(descriptions)

plt.figure(figsize=(20, 14))
plt.imshow(similarity, vmin=0.1, vmax=0.3)
# plt.colorbar()
plt.yticks(range(count), texts, fontsize=18)
plt.xticks([])
for i, image in enumerate(original_images):
    plt.imshow(image, extent=(i - 0.5, i + 0.5, -1.6, -0.6), origin="lower")
for x in range(similarity.shape[1]):
    for y in range(similarity.shape[0]):
        plt.text(x, y, f"{similarity[y, x]:.2f}", ha="center", va="center", size=12)

for side in ["left", "top", "right", "bottom"]:
  plt.gca().spines[side].set_visible(False)

plt.xlim([-0.5, count - 0.5])
plt.ylim([count + 0.5, -2])

plt.title("Cosine similarity between text and image features", size=20)
```

Out[17]:
```
Text(0.5, 1.0, 'Cosine similarity between text and image features')
```

Cosine similarity between text and image features

a red motorcycle standing in a garage	0.27	0.09	0.11	0.05	0.05	0.10	0.08	0.05
a facial photo of a tabby cat	0.07	0.26	0.12	0.12	0.13	0.15	0.14	0.09
a black-and-white silhouette of a horse	0.06	0.08	0.30	0.16	0.15	0.08	0.11	0.08
a person looking at a camera on a tripod	0.08	0.17	0.14	0.24	0.13	0.13	0.12	0.13
a page of text about segmentation	0.08	0.15	0.14	0.12	0.28	0.10	0.14	0.10
a portrait of an astronaut with the American flag	0.07	0.10	0.11	0.14	0.08	0.23	0.09	0.15
a cup of coffee on a saucer	0.08	0.14	0.11	0.11	0.12	0.11	0.24	0.07
a rocket standing on a launchpad	0.12	0.15	0.12	0.18	0.11	0.17	0.10	0.24

2.3 CLIP 在ImageNet-1K-16shot的零样本推理

在这一个部分中，我们会介绍如何利用CLIP进行某个数据集的零样本推理，以Project提供的ImageNet-1K-16shot数据集为例子。

- 准备数据集

In [42]:
```
import torch
import torchvision
import torchvision.datasets as datasets

import clip

from tqdm import tqdm

model, preprocess = clip.load("RN50")

test_dataset = datasets.ImageFolder(test_path, transform=preprocess)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=64, shuffle=False)

num_classes = len(test_dataset.classes)
print("class counts:", num_classes)
```

```
class counts: 1000
```

In [36]:
```
with open(os.path.join(data_path, 'label2class.json'), 'r') as f:
    json_data = json.load(f)
    print(json_data)
label2class = json_data['label2class']
print(label2class)
classnames = [v for k, v in label2class.items()]
print(classnames)
```


```
{'label2class': {'0': 'tench', '1': 'goldfish', '2': 'great white shark', '3': 'tiger shark', '4': 'hammerhead shark', '5': 'electric ray', '6': 'stingray', '7': 'rooster', '8': 'hen', '9': 'ostrich', '10': 'brambling', '11': 'goldfinch', '12': 'house finch', '13': 'junco', '14': 'indigo bunting', '15': 'American robin', '16': 'bulbul', '17': 'jay', '18': 'magpie', '19': 'chickadee', '20': 'American dipper', '21': 'kite (bird of prey)', '22': 'bald eagle', '23': 'vulture', '24': 'great grey owl', '25': 'fire salamander', '26': 'smooth newt', '27': 'newt', '28': 'spotted salamander', '29': 'axolotl', '30': 'American bullfrog', '31': 'tree frog', '32': 'tailed frog', '33': 'loggerhead sea turtle', '34': 'leatherback sea turtle', '35': 'mud turtle', '36': 'terrapin', '37': 'box turtle', '38': 'banded gecko', '39': 'green iguana', '40': 'Carolina anole', '41': 'desert grassland whiptail lizard', '42': 'agama', '43': 'frilled-necked lizard', '44': 'alligator lizard', '45': 'Gila monster', '46': 'European green lizard', '47': 'chameleon', '48': 'Komodo dragon', '49': 'Nile crocodile', '50': 'American alligator', '51': 'triceratops', '52': 'worm snake', '53': 'ring-necked snake', '54': 'eastern hog-nosed snake', '55': 'smooth green snake', '56': 'kingsnake', '57': 'garter snake', '58': 'water snake', '59': 'vine snake', '60': 'night snake', '61': 'boa constrictor', '62': 'African rock python', '63': 'Indian cobra', '64': 'green mamba', '65': 'sea snake', '66': 'Saharan horned viper', '67': 'eastern diamondback rattlesnake', '68': 'sidewinder rattlesnake', '69': 'trilobite', '70': 'harvestman', '71': 'scorpion', '72': 'yellow garden spider', '73': 'barn spider', '74': 'European garden spider', '75': 'southern black widow', '76': 'tarantula', '77': 'wolf spider', '78': 'tick', '79': 'centipede', '80': 'black grouse', '81': 'ptarmigan', '82': 'ruffed grouse', '83': 'prairie grouse', '84': 'peafowl', '85': 'quail', '86': 'partridge', '87': 'african grey parrot', '88': 'macaw', '89': 'sulphur-crested cockatoo', '90': 'lorikeet', '91': 'coucal', '92': 'bee eater', '93': 'hornbill', '94': 'hummingbird', '95': 'jacamar', '96': 'toucan', '97': 'duck', '98': 'red-breasted merganser', '99': 'goose', '100': 'black swan', '101': 'tusker', '102': 'echidna', '103': 'platypus', '104': 'wallaby', '105': 'koala', '106': 'wombat', '107': 'jellyfish', '108': 'sea anemone', '109': 'brain coral', '110': 'flatworm', '111': 'nematode', '112': 'conch', '113': 'snail', '114': 'slug', '115': 'sea slug', '116': 'chiton', '117': 'chambered nautilus', '118': 'Dungeness cr
```

In [39]:
```
def clip_classifier(classnames, clip_model, template):
    clip_model.eval()
    with torch.no_grad():
        clip_weights = []
        for classname in classnames:
            # Tokenize the prompts
            texts = template.format(classname)
            texts = clip.tokenize(texts).cuda()
            # prompt ensemble for ImageNet
            class_embedding = clip_model.encode_text(texts)
            class_embedding /= class_embedding.norm(dim=-1, keepdim=True)
            clip_weights.append(class_embedding)
        clip_weights = torch.concat(clip_weights, dim=0).cuda()
    return clip_weights


template = 'a photo of a {}.'
text_classifier = clip_classifier(classnames, model, template)
```

In [50]:
```
top1_accuracy = 0.0
top5_accuracy = 0.0
total_samples = 0

model.cuda().eval()
with torch.no_grad():
    for images, labels in tqdm(test_loader):
        images = images.cuda()
        image_features = model.encode_image(images)
        image_features = image_features / image_features.norm(dim=-1, keepdim=True)
        logits = image_features @ text_classifier.t()
        probs = logits.softmax(dim=-1)
        _, predicted_labels = probs.topk(5, dim=-1)
        predicted_labels = predicted_labels.cpu()
        total_samples += labels.size(0)
        top1_accuracy += (predicted_labels[:, 0] == labels).sum().item()
        top5_accuracy += (predicted_labels == labels.view(-1, 1)).sum().item()

top1_accuracy /= total_samples
top5_accuracy /= total_samples
```

```
100%|██████████| 782/782 [07:03<00:00,  1.84it/s]

29056.0
50000
Top-1 Accuracy: 58.11%
Top-5 Accuracy: 85.15%
```

In [52]:
```
print("Top-1 Accuracy: {:.2f}%".format(top1_accuracy * 100))
print("Top-5 Accuracy: {:.2f}%".format(top5_accuracy * 100))
```

```
Top-1 Accuracy: 58.11%
Top-5 Accuracy: 85.15%
```

3. 任务说明

3.1 任务目标

少样本学习在机器学习领域具有重大意义和挑战性，是否拥有从少量样本中学习和概括的能力，是将人工智能和人类智能进行区分的明显分界点，因为人类可以仅通过一个或几个示例就可以轻松地建立对新事物的认知，而机器学习算法通常需要成千上万个有监督样本来保证其泛化能力。本次大作业在给定具有挑战性的ImageNet-1K-16shot数据集（即每个类只有包含少量训练样本）的基础上，旨在用少量的样本来提升模型性能，即增强模型在ImageNet-1K-16shot测试集上的分类性能。

注明：可以通过以下渠道进行思考解决这个任务：1. 通过上述的预训练大模型比如CLIP来提升性能；2. 通过设计特殊的数据处理方式来提升性能；3. 通过设计特殊的网络结构来提升性能。

八仙过海，各显神通

3.2 任务要求

4. 相关参考文献

- 如何利用少量的样本来进行模型能力的提升，可以参考下面的论文

- Learning Transferable Visual Models From Natural Language Supervision. (ICML 2021)
- Learning to Prompt for Vision-Language Models. (IJCV 2022)
- P>M>F pipeline for few-shot learning (CVPR2022)

任务要求：在ImageNet上实现小样本分类。

1）设计可学习参数的模型，可以但不能仅使用懒惰学习模型，如KNN；

2）测试模型分别在每个类只提供1个样本，5个样本，16个样本情况下的性能；

3）性能要求：1-Shot （每个类提供一个样本）超越零样本分类结果。说明：可以和零样本分类结合。

4）实验分析。验证模型的有效性，性能，可视化，定量和定性分析。