Preparation

```
!git clone https://github.com/mlvlab/ProMetaR.git
%cd ProMetaR/

!git clone https://github.com/KaiyangZhou/Dassl.pytorch.git
%cd Dassl.pytorch/

# Install dependencies
!pip install -r requirements.txt
!cp -r dassl ../
# Install this library (no need to re-build if the source code is modified)
# !python setup.py develop
%cd ..

!pip install -r requirements.txt

%mkdir outputs
%mkdir data

%cd data
%mkdir eurosat
!wget http://madm.dfki.de/files/sentinel/EuroSAT.zip EuroSAT.zip

!unzip -o EuroSAT.zip -d eurosat/
%cd eurosat
!gdown 1Ip7yaCWFi0eaOFUGga0lUdVi_DDQth1o

%cd ../../

import os.path as osp
from collections import OrderedDict
import math
import torch
import torch.nn as nn
from torch.nn import functional as F
from torch.cuda.amp import GradScaler, autocast
from PIL import Image
import torchvision.transforms as transforms
import torch
from clip import clip
from clip.simple_tokenizer import SimpleTokenizer as _Tokenizer
import time
from tqdm import tqdm
import datetime
import argparse
from dassl.utils import setup_logger, set_random_seed, collect_env_info
from dassl.config import get_cfg_default
from dassl.engine import build_trainer
from dassl.engine import TRAINER_REGISTRY, TrainerX
from dassl.metrics import compute_accuracy
from dassl.utils import load_pretrained_weights, load_checkpoint
from dassl.optim import build_optimizer, build_lr_scheduler

# custom
import datasets.oxford_pets
import datasets.oxford_flowers
import datasets.fgvc_aircraft
import datasets.dtd
import datasets.eurosat
import datasets.stanford_cars
import datasets.food101
import datasets.sun397
import datasets.caltech101
import datasets.ucf101
import datasets.imagenet
import datasets.imagenet_sketch
import datasets.imagenetv2
import datasets.imagenet_a
import datasets.imagenet_r

def print_args(args, cfg):
    print("***************")
    print("** Arguments **")
    print("***************")
    optkeys = list(args.__dict__.keys())
    optkeys.sort()
    for key in optkeys:
        print("{}: {}".format(key, args.__dict__[key]))
```

```
        print("************")
        print("** Config **")
        print("************")
        print(cfg)

    def reset_cfg(cfg, args):
        if args.root:
            cfg.DATASET.ROOT = args.root
        if args.output_dir:
            cfg.OUTPUT_DIR = args.output_dir
        if args.seed:
            cfg.SEED = args.seed
        if args.trainer:
            cfg.TRAINER.NAME = args.trainer
        cfg.DATASET.NUM_SHOTS = 16
        cfg.DATASET.SUBSAMPLE_CLASSES = args.subsample_classes
        cfg.DATALOADER.TRAIN_X.BATCH_SIZE = args.train_batch_size
        cfg.OPTIM.MAX_EPOCH = args.epoch

    def extend_cfg(cfg):
        """
        Add new config variables.
        """
        from yacs.config import CfgNode as CN
        cfg.TRAINER.COOP = CN()
        cfg.TRAINER.COOP.N_CTX = 16  # number of context vectors
        cfg.TRAINER.COOP.CSC = False  # class-specific context
        cfg.TRAINER.COOP.CTX_INIT = ""  # initialization words
        cfg.TRAINER.COOP.PREC = "fp16"  # fp16, fp32, amp
        cfg.TRAINER.COOP.CLASS_TOKEN_POSITION = "end"  # 'middle' or 'end' or 'front'
        cfg.TRAINER.COCOOP = CN()
        cfg.TRAINER.COCOOP.N_CTX = 4  # number of context vectors
        cfg.TRAINER.COCOOP.CTX_INIT = "a photo of a"  # initialization words
        cfg.TRAINER.COCOOP.PREC = "fp16"  # fp16, fp32, amp
        cfg.TRAINER.PROMETAR = CN()
        cfg.TRAINER.PROMETAR.N_CTX_VISION = 4  # number of context vectors at the vision branch
        cfg.TRAINER.PROMETAR.N_CTX_TEXT = 4  # number of context vectors at the language branch
        cfg.TRAINER.PROMETAR.CTX_INIT = "a photo of a"  # initialization words
        cfg.TRAINER.PROMETAR.PREC = "fp16"  # fp16, fp32, amp
        cfg.TRAINER.PROMETAR.PROMPT_DEPTH_VISION = 9  # Max 12, minimum 0, for 0 it will be using shallow IVLP prompting (J=1)
        cfg.TRAINER.PROMETAR.PROMPT_DEPTH_TEXT = 9  # Max 12, minimum 0, for 0 it will be using shallow IVLP prompting (J=1)
        cfg.DATASET.SUBSAMPLE_CLASSES = "all"  # all, base or new
        cfg.TRAINER.PROMETAR.ADAPT_LR = 0.0005
        cfg.TRAINER.PROMETAR.LR_RATIO = 0.0005
        cfg.TRAINER.PROMETAR.FAST_ADAPTATION = False
        cfg.TRAINER.PROMETAR.MIXUP_ALPHA = 0.5
        cfg.TRAINER.PROMETAR.MIXUP_BETA = 0.5
        cfg.TRAINER.PROMETAR.DIM_RATE=8
        cfg.OPTIM_VNET = CN()
        cfg.OPTIM_VNET.NAME = "adam"
        cfg.OPTIM_VNET.LR = 0.0003
        cfg.OPTIM_VNET.WEIGHT_DECAY = 5e-4
        cfg.OPTIM_VNET.MOMENTUM = 0.9
        cfg.OPTIM_VNET.SGD_DAMPNING = 0
        cfg.OPTIM_VNET.SGD_NESTEROV = False
        cfg.OPTIM_VNET.RMSPROP_ALPHA = 0.99
        cfg.OPTIM_VNET.ADAM_BETA1 = 0.9
        cfg.OPTIM_VNET.ADAM_BETA2 = 0.999
        cfg.OPTIM_VNET.STAGED_LR = False
        cfg.OPTIM_VNET.NEW_LAYERS = ()
        cfg.OPTIM_VNET.BASE_LR_MULT = 0.1
        # Learning rate scheduler
        cfg.OPTIM_VNET.LR_SCHEDULER = "single_step"
        # -1 or 0 means the stepsize is equal to max_epoch
        cfg.OPTIM_VNET.STEPSIZE = (-1, )
        cfg.OPTIM_VNET.GAMMA = 0.1
        cfg.OPTIM_VNET.MAX_EPOCH = 10
        # Set WARMUP_EPOCH larger than 0 to activate warmup training
        cfg.OPTIM_VNET.WARMUP_EPOCH = -1
        # Either linear or constant
        cfg.OPTIM_VNET.WARMUP_TYPE = "linear"
        # Constant learning rate when type=constant
        cfg.OPTIM_VNET.WARMUP_CONS_LR = 1e-5
        # Minimum learning rate when type=linear
        cfg.OPTIM_VNET.WARMUP_MIN_LR = 1e-5
        # Recount epoch for the next scheduler (last_epoch=-1)
        # Otherwise last_epoch=warmup_epoch
        cfg.OPTIM_VNET.WARMUP_RECOUNT = True

    def setup_cfg(args):
        cfg = get_cfg_default()
        extend_cfg(cfg)
```

```
    # 1. From the dataset config file
    if args.dataset_config_file:
        cfg.merge_from_file(args.dataset_config_file)
    # 2. From the method config file
    if args.config_file:
        cfg.merge_from_file(args.config_file)
    # 3. From input arguments
    reset_cfg(cfg, args)
    cfg.freeze()
    return cfg

_tokenizer = _Tokenizer()

def load_clip_to_cpu(cfg): # Load CLIP
    backbone_name = cfg.MODEL.BACKBONE.NAME
    url = clip._MODELS[backbone_name]
    model_path = clip._download(url)

    try:
        # loading JIT archive
        model = torch.jit.load(model_path, map_location="cpu").eval()
        state_dict = None

    except RuntimeError:
        state_dict = torch.load(model_path, map_location="cpu")

    if cfg.TRAINER.NAME == "":
      design_trainer = "CoOp"
    else:
      design_trainer = cfg.TRAINER.NAME
    design_details = {"trainer": design_trainer,
                      "vision_depth": 0,
                      "language_depth": 0, "vision_ctx": 0,
                      "language_ctx": 0}
    model = clip.build_model(state_dict or model.state_dict(), design_details)

    return model

from dassl.config import get_cfg_default
cfg = get_cfg_default()
cfg.MODEL.BACKBONE.NAME = "ViT-B/16" # Set the vision encoder backbone of CLIP to ViT.
clip_model = load_clip_to_cpu(cfg)



class TextEncoder(nn.Module):
    def __init__(self, clip_model): # 초기화 하는 함수
        super().__init__()
        self.transformer = clip_model.transformer
        self.positional_embedding = clip_model.positional_embedding
        self.ln_final = clip_model.ln_final
        self.text_projection = clip_model.text_projection
        self.dtype = clip_model.dtype

    def forward(self, prompts, tokenized_prompts): # 모델 호출
        x = prompts + self.positional_embedding.type(self.dtype)
        x = x.permute(1, 0, 2)  # NLD -> LND
        x = self.transformer(x)
        x = x.permute(1, 0, 2)  # LND -> NLD
        x = self.ln_final(x).type(self.dtype)

        # x.shape = [batch_size, n_ctx, transformer.width]
        # take features from the eot embedding (eot_token is the highest number in each sequence)
        x = x[torch.arange(x.shape[0]), tokenized_prompts.argmax(dim=-1)] @ self.text_projection

        return x


@TRAINER_REGISTRY.register(force=True)
class CoCoOp(TrainerX):
    def check_cfg(self, cfg):
        assert cfg.TRAINER.COCOOP.PREC in ["fp16", "fp32", "amp"]

    def build_model(self):
        cfg = self.cfg
        classnames = self.dm.dataset.classnames
        print(f"Loading CLIP (backbone: {cfg.MODEL.BACKBONE.NAME})")
        clip_model = load_clip_to_cpu(cfg)

        if cfg.TRAINER.COCOOP.PREC == "fp32" or cfg.TRAINER.COCOOP.PREC == "amp":
            # CLIP's default precision is fp16
            clip_model.float()
```

```
        print("Building custom CLIP")
        self.model = CoCoOpCustomCLIP(cfg, classnames, clip_model)

        print("Turning off gradients in both the image and the text encoder")
        name_to_update = "prompt_learner"

        for name, param in self.model.named_parameters():
            if name_to_update not in name:
                param.requires_grad_(False)

        # Double check
        enabled = set()
        for name, param in self.model.named_parameters():
            if param.requires_grad:
                enabled.add(name)
        print(f"Parameters to be updated: {enabled}")

        if cfg.MODEL.INIT_WEIGHTS:
            load_pretrained_weights(self.model.prompt_learner, cfg.MODEL.INIT_WEIGHTS)

        self.model.to(self.device)
        # NOTE: only give prompt_learner to the optimizer
        self.optim = build_optimizer(self.model.prompt_learner, cfg.OPTIM)
        self.sched = build_lr_scheduler(self.optim, cfg.OPTIM)
        self.register_model("prompt_learner", self.model.prompt_learner, self.optim, self.sched)

        self.scaler = GradScaler() if cfg.TRAINER.COCOOP.PREC == "amp" else None

        # Note that multi-gpu training could be slow because CLIP's size is
        # big, which slows down the copy operation in DataParallel
        device_count = torch.cuda.device_count()
        if device_count > 1:
            print(f"Multiple GPUs detected (n_gpus={device_count}), use all of them!")
            self.model = nn.DataParallel(self.model)

    def before_train(self):
        directory = self.cfg.OUTPUT_DIR
        if self.cfg.RESUME:
            directory = self.cfg.RESUME
        self.start_epoch = self.resume_model_if_exist(directory)

        # Remember the starting time (for computing the elapsed time)
        self.time_start = time.time()


    def forward_backward(self, batch):
        image, label = self.parse_batch_train(batch)

        model = self.model
        optim = self.optim
        scaler = self.scaler

        prec = self.cfg.TRAINER.COCOOP.PREC
        loss = model(image, label) # Input image 모델 통과
        optim.zero_grad()
        loss.backward() # Backward (역전파)
        optim.step() # 모델 parameter update

        loss_summary = {"loss": loss.item()}

        if (self.batch_idx + 1) == self.num_batches:
            self.update_lr()

        return loss_summary

    def parse_batch_train(self, batch):
        input = batch["img"]
        label = batch["label"]
        input = input.to(self.device)
        label = label.to(self.device)
        return input, label

    def load_model(self, directory, epoch=None):
        if not directory:
            print("Note that load_model() is skipped as no pretrained model is given")
            return

        names = self.get_model_names()

        # By default, the best model is loaded
        model_file = "model-best.pth.tar"
```

```
        if epoch is not None:
            model_file = "model.pth.tar-" + str(epoch)

        for name in names:
            model_path = osp.join(directory, name, model_file)

            if not osp.exists(model_path):
                raise FileNotFoundError('Model not found at "{}"'.format(model_path))

            checkpoint = load_checkpoint(model_path)
            state_dict = checkpoint["state_dict"]
            epoch = checkpoint["epoch"]

            # Ignore fixed token vectors
            if "token_prefix" in state_dict:
                del state_dict["token_prefix"]

            if "token_suffix" in state_dict:
                del state_dict["token_suffix"]

            print("Loading weights to {} " 'from "{}" (epoch = {})'.format(name, model_path, epoch))
            # set strict=False
            self._models[name].load_state_dict(state_dict, strict=False)

    def after_train(self):
      print("Finish training")

      do_test = not self.cfg.TEST.NO_TEST
      if do_test:
          if self.cfg.TEST.FINAL_MODEL == "best_val":
              print("Deploy the model with the best val performance")
              self.load_model(self.output_dir)
          else:
              print("Deploy the last-epoch model")
          acc = self.test()

      # Show elapsed time
      elapsed = round(time.time() - self.time_start)
      elapsed = str(datetime.timedelta(seconds=elapsed))
      print(f"Elapsed: {elapsed}")

      # Close writer
      self.close_writer()
      return acc

    def train(self):
        """Generic training loops."""
        self.before_train()
        for self.epoch in range(self.start_epoch, self.max_epoch):
            self.before_epoch()
            self.run_epoch()
            self.after_epoch()
        acc = self.after_train()
        return acc

parser = argparse.ArgumentParser()
parser.add_argument("--root", type=str, default="data/", help="path to dataset")
parser.add_argument("--output-dir", type=str, default="outputs/cocoop3", help="output directory")
parser.add_argument(
    "--seed", type=int, default=1, help="only positive value enables a fixed seed"
)
parser.add_argument(
    "--config-file", type=str, default="configs/trainers/ProMetaR/vit_b16_c2_ep10_batch4_4+4ctx.yaml", help="path to config
)
parser.add_argument(
    "--dataset-config-file",
    type=str,
    default="configs/datasets/eurosat.yaml",
    help="path to config file for dataset setup",
)
parser.add_argument("--trainer", type=str, default="CoOp", help="name of trainer")
parser.add_argument("--eval-only", action="store_true", help="evaluation only")
parser.add_argument(
    "--model-dir",
    type=str,
    default="",
    help="load model from this directory for eval-only mode",
)
parser.add_argument("--train-batch-size", type=int, default=4)
parser.add_argument("--epoch", type=int, default=10)
parser.add_argument("--subsample-classes", type=str, default="base")
```

```
parser.add_argument(
    "--load-epoch", type=int, default=0, help="load model weights at this epoch for evaluation"
)
args = parser.parse_args([])

def main(args):
    cfg = setup_cfg(args)
    if cfg.SEED >= 0:
        set_random_seed(cfg.SEED)

    if torch.cuda.is_available() and cfg.USE_CUDA:
        torch.backends.cudnn.benchmark = True

    trainer = build_trainer(cfg)
    if args.eval_only:
        trainer.load_model(args.model_dir, epoch=args.load_epoch)
        acc = trainer.test()
        return acc

    acc = trainer.train()
    return acc
```

```
Requirement already satisfied: google-auth==2.17.0 in /usr/local/lib/python3.10/dist-packages (from google-auth[aiohttp]
Requirement already satisfied: google-auth-httplib2>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from gsutil->lear
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth==2.17
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth==2.17.
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth==2.17.0->googl
Requirement already satisfied: aiohttp<4.0.0dev,>=3.6.2 in /usr/local/lib/python3.10/dist-packages (from google-auth[aio
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->learn
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->learn2learn==0.2.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->learn2learn
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->learn2learn
Requirement already satisfied: osqp>=0.6.2 in /usr/local/lib/python3.10/dist-packages (from cvxpy>=1.1.0->qpth>=0.0.15->
Requirement already satisfied: ecos>=2 in /usr/local/lib/python3.10/dist-packages (from cvxpy>=1.1.0->qpth>=0.0.15->lear
Requirement already satisfied: clarabel>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from cvxpy>=1.1.0->qpth>=0.0.
Requirement already satisfied: scs>=3.2.4.post1 in /usr/local/lib/python3.10/dist-packages (from cvxpy>=1.1.0->qpth>=0.0
Requirement already satisfied: boto>=2.29.1 in /usr/local/lib/python3.10/dist-packages (from gcs-oauth2-boto-plugin>=3.2
Requirement already satisfied: oauth2client>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from gcs-oauth2-boto-plug
Requirement already satisfied: pyasn1>=0.1.3 in /usr/local/lib/python3.10/dist-packages (from rsa<5,>=3.1.4->google-auth
Requirement already satisfied: pyu2f in /usr/local/lib/python3.10/dist-packages (from google-reauth>=0.1.0->gsutil->lear
Requirement already satisfied: cryptography<44,>=41.0.5 in /usr/local/lib/python3.10/dist-packages (from pyOpenSSL<=24.2
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.1.0->le
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev,>=3.6.
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev,>=3.6.2->
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev,>=3.6
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev,>=3
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev,>=3.6.
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0dev,>=3.6
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography<44,>=41.0.5->pyO
Requirement already satisfied: qdldl in /usr/local/lib/python3.10/dist-packages (from osqp>=0.6.2->cvxpy>=1.1.0->qpth>=0
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography<44,>=
mkdir: cannot create directory ‘outputs’: File exists
mkdir: cannot create directory ‘data’: File exists
/content/ProMetaR/data
mkdir: cannot create directory ‘eurosat’: File exists
--2024-12-10 04:21:33--  http://madm.dfki.de/files/sentinel/EuroSAT.zip
Resolving madm.dfki.de (madm.dfki.de)... 131.246.195.183
Connecting to madm.dfki.de (madm.dfki.de)|131.246.195.183|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 94280567 (90M) [application/zip]
Saving to: ‘EuroSAT.zip.1’

EuroSAT.zip.1         1%[                    ]   1.03M   122KB/s    eta 10m 43s^C
Archive:  EuroSAT.zip
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of EuroSAT.zip or
        EuroSAT.zip.zip, and cannot find EuroSAT.zip.ZIP, period.
/content/ProMetaR/data/eurosat
Downloading...
From: https://drive.google.com/uc?id=1Ip7yaCWFi0eaOFUGga0lUdVi_DDQth1o
To: /content/ProMetaR/data/eurosat/split_zhou_EuroSAT.json
100% 3.01M/3.01M [00:00<00:00, 131MB/s]
/content/ProMetaR
100%|████████████████████████████████████████| 351M/351M [00:02<00:00, 117MiB/s]
```

```
!ls data/eurosat/2750
```

```
ls: cannot access 'data/eurosat/2750': No such file or directory
```

```
# 1. 먼저 현재 eurosat 디렉토리를 정리
!rm -rf data/eurosat/*
```

```
# 2. 작업 디렉토리 이동
%cd data/eurosat

# 3. EuroSAT 데이터셋 다운로드 (직접적인 다운로드 URL 사용)
!wget http://madm.dfki.de/files/sentinel/EuroSAT.zip

# 4. 압축 해제
!unzip -o EuroSAT.zip

# 5. split 파일 다운로드
!gdown 1Ip7yaCWFi0eaOFUGga0lUdVi_DDQth1o

# 6. 원래 디렉토리로 돌아가기
%cd ../../

# 7. 확인
!ls data/eurosat/2750
```

```
  inflating: 2750/PermanentCrop/PermanentCrop_1398.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_163.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_970.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_502.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2472.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1567.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1915.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2013.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_828.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1106.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1670.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1211.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2304.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_273.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1088.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_612.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1438.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_164.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1059.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_505.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_977.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2475.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1912.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1560.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2014.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1101.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1677.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_19.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1216.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2303.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1753.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1332.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1495.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2227.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_118.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1444.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1836.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2130.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1782.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_579.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1025.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2409.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_853.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_421.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_386.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_2068.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_882.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_357.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_1.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_65.jpg
  inflating: 2750/PermanentCrop/PermanentCrop_736.jpg
Downloading...
From: https://drive.google.com/uc?id=1Ip7yaCWFi0eaOFUGga0lUdVi_DDQth1o
To: /content/ProMetaR/data/eurosat/split_zhou_EuroSAT.json
100% 3.01M/3.01M [00:00<00:00, 221MB/s]
/content/ProMetaR
AnnualCrop  HerbaceousVegetation  Industrial  PermanentCrop  River
Forest      Highway               Pasture     Residential    SeaLake
```

Use another architecture for CoCoOp's Meta Net - attention

Key changes made to the meta network:

Original architecture:

- Simple feed-forward network with two linear layers and ReLU

New attention-based architecture:

- Input projection layer: Linear(vis_dim → hidden_dim)
- Multi-head Self-Attention:
    - 8 attention heads
    - Query, Key, Value projections
    - Scaled dot-product attention
- Layer Normalization layers
- Output projection: Two-layer MLP to ctx_dim

Benefits of attention mechanism:

- Dynamic feature interaction
- Ability to capture long-range dependencies
- Content-dependent feature transformation
- Enhanced representation power

Additional components:

- Dropout layers for regularization
- Layer normalization for training stability
- Scaled attention for better gradient flow

This modification allows the model to learn more complex, context-aware transformations of the visual features through self-attention mechanisms, potentially capturing richer feature relationships than the original feed-forward architecture.

```
import torch.nn as nn

class CoCoOpPromptLearner(nn.Module):
    def __init__(self, cfg, classnames, clip_model):
        super().__init__()
        n_cls = len(classnames)
        n_ctx = cfg.TRAINER.COCOOP.N_CTX
        ctx_init = cfg.TRAINER.COCOOP.CTX_INIT
        dtype = clip_model.dtype
        ctx_dim = clip_model.ln_final.weight.shape[0]
        vis_dim = clip_model.visual.output_dim
        clip_imsize = clip_model.visual.input_resolution
        cfg_imsize = cfg.INPUT.SIZE[0]
        assert cfg_imsize == clip_imsize, f"cfg_imsize ({cfg_imsize}) must equal to clip_imsize ({clip_imsize})"

        if ctx_init:
            # use given words to initialize context vectors
            ctx_init = ctx_init.replace("_", " ")
            n_ctx = len(ctx_init.split(" "))
            prompt = clip.tokenize(ctx_init)
            with torch.no_grad():
                embedding = clip_model.token_embedding(prompt).type(dtype)
            ctx_vectors = embedding[0, 1: 1 + n_ctx, :]
            prompt_prefix = ctx_init
        else:
            # random initialization
            ctx_vectors = torch.empty(n_ctx, ctx_dim, dtype=dtype)
            nn.init.normal_(ctx_vectors, std=0.02)
            prompt_prefix = " ".join(["X"] * n_ctx)

        print(f'Initial context: "{prompt_prefix}"')
        print(f"Number of context words (tokens): {n_ctx}")

        self.ctx = nn.Parameter(ctx_vectors)  # Wrap the initialized prompts above as parameters to make them trainable.

        ### Tokenize ###
        classnames = [name.replace("_", " ") for name in classnames]  # 예) "Forest"
        name_lens = [len(_tokenizer.encode(name)) for name in classnames]
        prompts = [prompt_prefix + " " + name + "." for name in classnames] # 예) "A photo of Forest."

        tokenized_prompts = torch.cat([clip.tokenize(p) for p in prompts]) # 예) [49406, 320, 1125, 539...]



        # ####################################
        # ########## Define Meta Net ##########
        # self.meta_net = nn.Sequential(OrderedDict([
        #     #("linear1", "fill in here"(vis_dim, vis_dim // 16)),
        #     ("linear1", nn.Linear(vis_dim, vis_dim // 16)),
        #     ("relu", nn.ReLU(inplace=True)),
        #     ("linear2", nn.Linear(vis_dim // 16, ctx_dim))
        # ]))
        # ####################################
```

```
        #####################################
        ########## Define Meta Net ##########
        # Define attention parameters
        self.hidden_dim = vis_dim // 16  # 기존 hidden_dim 크기 유지

        # Input projection
        self.input_proj = nn.Linear(vis_dim, self.hidden_dim)

        # Multi-head Attention
        num_heads = 8
        self.num_heads = num_heads
        self.scale = (self.hidden_dim // num_heads) ** -0.5

        self.q_proj = nn.Linear(self.hidden_dim, self.hidden_dim)
        self.k_proj = nn.Linear(self.hidden_dim, self.hidden_dim)
        self.v_proj = nn.Linear(self.hidden_dim, self.hidden_dim)

        self.attn_dropout = nn.Dropout(0.1)

        # Output projection
        self.output_proj = nn.Sequential(
            nn.Linear(self.hidden_dim, self.hidden_dim),
            nn.ReLU(inplace=True),
            nn.Linear(self.hidden_dim, ctx_dim)
        )

        # Layer Normalization
        self.norm1 = nn.LayerNorm(self.hidden_dim)
        self.norm2 = nn.LayerNorm(self.hidden_dim)
        #####################################

        # if cfg.TRAINER.COCOOP.PREC == "fp16":
        #     self.meta_net.half()

        if cfg.TRAINER.COCOOP.PREC == "fp16":
              self.input_proj.half()
              self.q_proj.half()
              self.k_proj.half()
              self.v_proj.half()
              self.output_proj.half()
              self.norm1.half()
              self.norm2.half()

        with torch.no_grad():
            embedding = clip_model.token_embedding(tokenized_prompts).type(dtype)

        # These token vectors will be saved when in save_model(),
        # but they should be ignored in load_model() as we want to use
        # those computed using the current class names
        self.register_buffer("token_prefix", embedding[:, :1, :])  # SOS
        self.register_buffer("token_suffix", embedding[:, 1 + n_ctx:, :])  # CLS, EOS
        self.n_cls = n_cls
        self.n_ctx = n_ctx
        self.tokenized_prompts = tokenized_prompts  # torch.Tensor
        self.name_lens = name_lens

    def construct_prompts(self, ctx, prefix, suffix, label=None):
        # dim0 is either batch_size (during training) or n_cls (during testing)
        # ctx: context tokens, with shape of (dim0, n_ctx, ctx_dim)
        # prefix: the sos token, with shape of (n_cls, 1, ctx_dim)
        # suffix: remaining tokens, with shape of (n_cls, *, ctx_dim)

        if label is not None:
            prefix = prefix[label]
            suffix = suffix[label]

        prompts = torch.cat(
            [
                prefix,  # (dim0, 1, dim)
                ctx,  # (dim0, n_ctx, dim)
                suffix,  # (dim0, *, dim)
            ],
            dim=1,
        )

        return prompts

    def forward(self, im_features):
        prefix = self.token_prefix
        suffix = self.token_suffix
        ctx = self.ctx  # (n_ctx, ctx_dim)
```

```
        # ############################################
        # #Image feature is given as input to meta network # (batch, ctx_dim)
        # bias = self.meta_net(im_features)  # (batch, ctx_dim)
        # bias = bias.unsqueeze(1)  # (batch, 1, ctx_dim)
        # ctx = ctx.unsqueeze(0)  # (1, n_ctx, ctx_dim)
        # #ctx_shifted = ctx + " Add meta token to context token"  # (batch, n_ctx, ctx_dim)
        # ctx_shifted = ctx + bias  # (batch, n_ctx, ctx_dim)
        # ############################################
        # ############################################

        ############################################
        # Project input features
        x = self.input_proj(im_features)

        # Add sequence dimension for attention
        x = x.unsqueeze(1)  # [B, 1, hidden_dim]

        # Multi-head attention
        B = x.shape[0]
        q = self.q_proj(x).view(B, -1, self.num_heads, self.hidden_dim // self.num_heads).transpose(1, 2)
        k = self.k_proj(x).view(B, -1, self.num_heads, self.hidden_dim // self.num_heads).transpose(1, 2)
        v = self.v_proj(x).view(B, -1, self.num_heads, self.hidden_dim // self.num_heads).transpose(1, 2)

        # Scaled dot-product attention
        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = attn.softmax(dim=-1)
        attn = self.attn_dropout(attn)

        # Apply attention to values
        x = (attn @ v).transpose(1, 2).reshape(B, -1, self.hidden_dim)
        x = self.norm1(x)

        # Output projection with normalization
        x = self.norm2(x)
        bias = self.output_proj(x)

        # Reshape to match original expectations
        bias = bias.squeeze(1)  # Remove sequence dimension
        bias = bias.unsqueeze(1)  # Add context dimension
        ctx = ctx.unsqueeze(0)  # (1, n_ctx, ctx_dim)
        ctx_shifted = ctx + bias  # (batch, n_ctx, ctx_dim)
        ############################################


        # Use instance-conditioned context tokens for all classes
        prompts = []
        for ctx_shifted_i in ctx_shifted:
            ctx_i = ctx_shifted_i.unsqueeze(0).expand(self.n_cls, -1, -1)
            pts_i = self.construct_prompts(ctx_i, prefix, suffix)  # (n_cls, n_tkn, ctx_dim)
            prompts.append(pts_i)
        prompts = torch.stack(prompts)

        return prompts


class CoCoOpCustomCLIP(nn.Module):
    def __init__(self, cfg, classnames, clip_model):
        super().__init__()
        self.prompt_learner = CoCoOpPromptLearner(cfg, classnames, clip_model)
        self.tokenized_prompts = self.prompt_learner.tokenized_prompts
        self.image_encoder = clip_model.visual
        self.text_encoder = TextEncoder(clip_model)
        self.logit_scale = clip_model.logit_scale
        self.dtype = clip_model.dtype

    def forward(self, image, label=None):
        tokenized_prompts = self.tokenized_prompts
        logit_scale = self.logit_scale.exp()

        image_features = self.image_encoder(image.type(self.dtype))
        image_features = image_features / image_features.norm(dim=-1, keepdim=True)


        ############################################
        prompts = self.prompt_learner(image_features)
        ############################################


        logits = []
        for pts_i, imf_i in zip(prompts, image_features):
```

```
            text_features = self.text_encoder(pts_i, tokenized_prompts)
            text_features = text_features / text_features.norm(dim=-1, keepdim=True)
            l_i = logit_scale * imf_i @ text_features.t()
            logits.append(l_i)
        logits = torch.stack(logits)

        if self.prompt_learner.training:
            return F.cross_entropy(logits, label)

        return logits
```

Training

- Train modified version of CoCoOp (attention) on the EuroSAT dataset.

```
# Train on the Base Classes Train split and evaluate accuracy on the Base Classes Test split.
args.trainer = "CoCoOp"
args.train_batch_size = 4
args.epoch = 100
args.output_dir = "outputs/cocoop"

args.subsample_classes = "base"
args.eval_only = False
cocoop_base_acc = main(args)
```

```
epoch [55/100] batch [20/20] time 0.171 (0.167) data 0.000 (0.016) loss 0.1671 (0.1557) lr 1.0933e-03 eta 0:02:30
epoch [56/100] batch [20/20] time 0.101 (0.148) data 0.000 (0.035) loss 0.2837 (0.2411) lr 1.0545e-03 eta 0:02:10
epoch [57/100] batch [20/20] time 0.103 (0.136) data 0.000 (0.018) loss 0.1342 (0.1692) lr 1.0158e-03 eta 0:01:56
epoch [58/100] batch [20/20] time 0.104 (0.136) data 0.000 (0.022) loss 0.4031 (0.2366) lr 9.7732e-04 eta 0:01:54
epoch [59/100] batch [20/20] time 0.166 (0.172) data 0.000 (0.018) loss 0.6001 (0.1920) lr 9.3914e-04 eta 0:02:21
epoch [60/100] batch [20/20] time 0.109 (0.151) data 0.000 (0.028) loss 0.2893 (0.1639) lr 9.0126e-04 eta 0:02:00
epoch [61/100] batch [20/20] time 0.116 (0.136) data 0.000 (0.018) loss 0.4861 (0.1726) lr 8.6373e-04 eta 0:01:46
epoch [62/100] batch [20/20] time 0.106 (0.136) data 0.000 (0.020) loss 0.1142 (0.1630) lr 8.2658e-04 eta 0:01:43
epoch [63/100] batch [20/20] time 0.167 (0.159) data 0.000 (0.019) loss 0.1274 (0.1640) lr 7.8984e-04 eta 0:01:57
epoch [64/100] batch [20/20] time 0.157 (0.218) data 0.000 (0.038) loss 0.0599 (0.1769) lr 7.5357e-04 eta 0:02:36
epoch [65/100] batch [20/20] time 0.104 (0.141) data 0.000 (0.021) loss 0.0748 (0.1576) lr 7.1778e-04 eta 0:01:38
epoch [66/100] batch [20/20] time 0.104 (0.136) data 0.000 (0.017) loss 0.0853 (0.1692) lr 6.8251e-04 eta 0:01:32
epoch [67/100] batch [20/20] time 0.103 (0.139) data 0.000 (0.023) loss 0.0605 (0.1874) lr 6.4781e-04 eta 0:01:31
epoch [68/100] batch [20/20] time 0.162 (0.175) data 0.000 (0.017) loss 0.0459 (0.2167) lr 6.1370e-04 eta 0:01:52
epoch [69/100] batch [20/20] time 0.113 (0.151) data 0.000 (0.037) loss 0.0652 (0.2556) lr 5.8022e-04 eta 0:01:33
epoch [70/100] batch [20/20] time 0.104 (0.136) data 0.000 (0.019) loss 0.0228 (0.1478) lr 5.4740e-04 eta 0:01:21
epoch [71/100] batch [20/20] time 0.109 (0.142) data 0.000 (0.017) loss 0.6313 (0.1949) lr 5.1527e-04 eta 0:01:22
epoch [72/100] batch [20/20] time 0.160 (0.171) data 0.000 (0.018) loss 0.1803 (0.1820) lr 4.8387e-04 eta 0:01:35
epoch [73/100] batch [20/20] time 0.109 (0.150) data 0.000 (0.031) loss 0.1229 (0.2125) lr 4.5322e-04 eta 0:01:21
epoch [74/100] batch [20/20] time 0.102 (0.142) data 0.000 (0.025) loss 0.0265 (0.2105) lr 4.2336e-04 eta 0:01:13
epoch [75/100] batch [20/20] time 0.110 (0.142) data 0.000 (0.018) loss 0.4802 (0.1531) lr 3.9432e-04 eta 0:01:11
epoch [76/100] batch [20/20] time 0.176 (0.183) data 0.000 (0.020) loss 0.0651 (0.1312) lr 3.6612e-04 eta 0:01:27
epoch [77/100] batch [20/20] time 0.105 (0.158) data 0.000 (0.026) loss 0.1317 (0.1197) lr 3.3879e-04 eta 0:01:12
epoch [78/100] batch [20/20] time 0.115 (0.150) data 0.000 (0.018) loss 0.4038 (0.2846) lr 3.1236e-04 eta 0:01:05
epoch [79/100] batch [20/20] time 0.112 (0.151) data 0.000 (0.022) loss 0.0757 (0.1938) lr 2.8686e-04 eta 0:01:03
epoch [80/100] batch [20/20] time 0.166 (0.205) data 0.000 (0.019) loss 0.4578 (0.2208) lr 2.6231e-04 eta 0:01:21
epoch [81/100] batch [20/20] time 0.115 (0.154) data 0.000 (0.024) loss 0.0412 (0.1638) lr 2.3873e-04 eta 0:00:58
epoch [82/100] batch [20/20] time 0.117 (0.151) data 0.000 (0.022) loss 0.0385 (0.1642) lr 2.1615e-04 eta 0:00:54
epoch [83/100] batch [20/20] time 0.115 (0.152) data 0.000 (0.022) loss 0.2583 (0.2050) lr 1.9459e-04 eta 0:00:51
epoch [84/100] batch [20/20] time 0.192 (0.219) data 0.000 (0.033) loss 0.0464 (0.3121) lr 1.7407e-04 eta 0:01:10
epoch [85/100] batch [20/20] time 0.113 (0.153) data 0.000 (0.021) loss 0.4912 (0.1886) lr 1.5462e-04 eta 0:00:45
epoch [86/100] batch [20/20] time 0.115 (0.163) data 0.000 (0.018) loss 0.1860 (0.2096) lr 1.3624e-04 eta 0:00:45
epoch [87/100] batch [20/20] time 0.115 (0.151) data 0.000 (0.020) loss 0.0730 (0.1450) lr 1.1897e-04 eta 0:00:39
epoch [88/100] batch [20/20] time 0.169 (0.219) data 0.000 (0.039) loss 0.6846 (0.2311) lr 1.0281e-04 eta 0:00:52
epoch [89/100] batch [20/20] time 0.115 (0.154) data 0.000 (0.024) loss 0.2366 (0.1328) lr 8.7779e-05 eta 0:00:33
epoch [90/100] batch [20/20] time 0.114 (0.150) data 0.000 (0.025) loss 0.1218 (0.1274) lr 7.3899e-05 eta 0:00:30
epoch [91/100] batch [20/20] time 0.114 (0.151) data 0.000 (0.018) loss 0.1830 (0.1055) lr 6.1179e-05 eta 0:00:27
epoch [92/100] batch [20/20] time 0.166 (0.210) data 0.000 (0.027) loss 1.1846 (0.2205) lr 4.9633e-05 eta 0:00:33
epoch [93/100] batch [20/20] time 0.116 (0.152) data 0.000 (0.025) loss 0.0054 (0.2143) lr 3.9271e-05 eta 0:00:21
epoch [94/100] batch [20/20] time 0.115 (0.156) data 0.000 (0.017) loss 0.1220 (0.1181) lr 3.0104e-05 eta 0:00:18
epoch [95/100] batch [20/20] time 0.112 (0.150) data 0.000 (0.018) loss 0.4741 (0.1931) lr 2.2141e-05 eta 0:00:15
epoch [96/100] batch [20/20] time 0.169 (0.214) data 0.000 (0.033) loss 0.1070 (0.1846) lr 1.5390e-05 eta 0:00:17
epoch [97/100] batch [20/20] time 0.113 (0.153) data 0.000 (0.026) loss 0.3340 (0.1823) lr 9.8566e-06 eta 0:00:09
epoch [98/100] batch [20/20] time 0.128 (0.151) data 0.000 (0.020) loss 0.1276 (0.1977) lr 5.5475e-06 eta 0:00:06
epoch [99/100] batch [20/20] time 0.115 (0.151) data 0.000 (0.019) loss 0.4463 (0.1528) lr 2.4666e-06 eta 0:00:03
epoch [100/100] batch [20/20] time 0.175 (0.218) data 0.000 (0.029) loss 0.0516 (0.1780) lr 6.1680e-07 eta 0:00:00
Checkpoint saved to outputs/cocoop/prompt_learner/model.pth.tar-100
Finish training
Deploy the last-epoch model
Evaluate on the *test* set
100%|██████████| 42/42 [01:06<00:00,  1.58s/it]=> result
* total: 4,200
* correct: 3,846
* accuracy: 91.6%
* error: 8.4%
* macro_f1: 91.6%
Elapsed: 0:06:57
```

```
# Accuracy on the New Classes.
args.model_dir = "outputs/cocoop"
args.output_dir = "outputs/cocoop/new_classes"
args.subsample_classes = "new"
args.load_epoch = 100
args.eval_only = True
coop_novel_acc = main(args)
```

```
Loading trainer: CoCoOp
Loading dataset: EuroSAT
Reading split from /content/ProMetaR/data/eurosat/split_zhou_EuroSAT.json
Loading preprocessed few-shot data from /content/ProMetaR/data/eurosat/split_fewshot/shot_16-seed_1.pkl
SUBSAMPLE NEW CLASSES!
Building transform_train
+ random resized crop (size=(224, 224), scale=(0.08, 1.0))
+ random flip
+ to torch tensor of range [0, 1]
+ normalization (mean=[0.48145466, 0.4578275, 0.40821073], std=[0.26862954, 0.26130258, 0.27577711])
Building transform_test
+ resize the smaller edge to 224
+ 224x224 center crop
+ to torch tensor of range [0, 1]
+ normalization (mean=[0.48145466, 0.4578275, 0.40821073], std=[0.26862954, 0.26130258, 0.27577711])
---------  -------
Dataset    EuroSAT
# classes  5
# train_x  80
# val      20
# test     3,900
---------  -------
Loading CLIP (backbone: ViT-B/16)
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:617: UserWarning: This DataLoader will create 8 w
  warnings.warn(
Building custom CLIP
Initial context: "a photo of a"
Number of context words (tokens): 4
Turning off gradients in both the image and the text encoder
Parameters to be updated: {'prompt_learner.output_proj.0.bias', 'prompt_learner.ctx', 'prompt_learner.q_proj.bias', 'pro
Loading evaluator: Classification
Loading weights to prompt_learner from "outputs/cocoop/prompt_learner/model.pth.tar-100" (epoch = 100)
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:62: UserWarning: The verbose parameter is deprecated
  warnings.warn(
/content/ProMetaR/dassl/utils/torchtools.py:102: FutureWarning: You are using `torch.load` with `weights_only=False` (th
  checkpoint = torch.load(fpath, map_location=map_location)
Evaluate on the *test* set
100%|██████████| 39/39 [01:05<00:00,  1.67s/it]=> result
* total: 3,900
* correct: 1,877
* accuracy: 48.1%
* error: 51.9%
* macro_f1: 46.4%
```