In [1]:
```
#from google.colab import drive
#drive.mount('/content/drive')

# List of dependencies from the 'environment.yaml' file
dependencies = [
    'absl-py==0.11.0',
    'aiohttp==3.7.3',
    'argon2-cffi==20.1.0',
    'attrs==20.3.0',
    'boto==2.49.0',
    'boto3==1.9.66',
    'botocore==1.12.67',
    'bleach==3.2.1',
    'brotlipy==0.7.0',
    'cachetools==4.2.0',
    'certifi==2021.10.8',
    'cffi==1.14.4',
    'chardet==3.0.4',
    'click==7.1.2',
    'cloudpickle==1.6.0',
    'cryptography==3.3.1',
    'decorator==4.4.2',
    'defusedxml==0.6.0',
    'docutils==0.16',
    'filelock==3.0.12',
    'google-api-core==1.22.2',
    'google-auth==1.23.0',
    'google-auth-oauthlib==0.4.1',
    'google-cloud-core==1.4.4',
    'google-cloud-storage==1.33.0',
    'googleapis-common-protos==1.52.0',
    'grpcio==1.31.0',
    'gdown==3.12.2',
    'idna==2.10',
    'imageio==2.9.0',
    'importlib-metadata==2.0.0',
    'ipykernel==5.3.4',
    'ipython==5.8.0',
    'ipython-genutils==0.2.0',
    'ipywidgets==7.5.1',
    'jinja2==2.11.2',
    'jmespath==0.10.0',
    'joblib==0.17.0',
    'jsonschema==3.2.0',
    'jupyter-client==6.1.7',
    'jupyter-core==4.7.0',
    'kiwisolver==1.3.0',
    'markdown==3.3.3',
    'matplotlib==3.3.2',
    'mistune==0.8.4',
    'nbclient==0.5.1',
    'nbconvert==6.0.7',
    'nbformat==5.0.8',
    'nest-asyncio==1.4.3',
```

```
    'notebook==6.1.4',
    'numpy==1.19.2',
    'oauthlib==3.1.0',
    'opencv-python==4.5.1.48',
    'packaging==20.7',
    'pandas==1.1.5',
    'prometheus-client==0.9.0',
    'protobuf==3.13.0',
    'pyasn1==0.4.8',
    'pyasn1-modules==0.2.8',
    'pycparser==2.20',
    'pygments==2.7.3',
    'pyopenssl==20.0.0',
    'pyparsing==2.4.7',
    'pyrsistent==0.17.3',
    'pysocks==1.7.1',
    'python-dateutil==2.8.1',
    'pytz==2020.4',
    'pyyaml==5.3.1',
    'pyzmq==20.0.0',
    'requests==2.25.0',
    'requests-oauthlib==1.3.0',
    'rsa==4.6',
    'scikit-image==0.17.2',
    'scikit-learn==0.23.2',
    'scipy==1.5.2',
    'send2trash==1.5.0',
    'setuptools==51.0.0',
    'six==1.15.0',
    'sqlite==3.33.0',
    'tensorboard==2.3.0',
    'terminado==0.9.1',
    'testpath==0.4.4',
    'tornado==6.1',
    'tqdm==4.51.0',
    'traitlets==5.0.5',
    'typing-extensions==3.7.4.3',
    'urllib3==1.24.3',
    'wcwidth==0.2.5',
    'webencodings==0.5.1',
    'werkzeug==1.0.1',
    'wheel==0.36.1',
    'widgetsnbextension==3.5.1',
    'zipp==3.4.0',
    'dataclasses==0.6',
    'einops==0.4.0',
    'future==0.18.2',
    'vit-pytorch==0.26.7'
]

# Install all dependencies at once using pip
!pip install {' '.join(dependencies)}

!pip install tensorboardX

# Setup File path for dataset
```

```
!cp /content/drive/MyDrive/ECE570-Project/dogs.zip /content/
!unzip /content/dogs.zip -d /content/
```

```
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_877.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_8798.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_8871.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9065.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9067.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_9110.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_925.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9272.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9302.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_9348.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9396.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_9455.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9675.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_9686.png
 inflating: /content/dogs/val_pre/n02115641-dingo/n02115641_970.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9763.png
extracting: /content/dogs/val_pre/n02115641-dingo/n02115641_9977.png
```

In [2]:

```python
# Conv_4

import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np

class ConvBlock(nn.Module):

    def __init__(self,input_channel,output_channel):
        super().__init__()

        self.layers = nn.Sequential(
            nn.Conv2d(input_channel,output_channel,kernel_size=3,padding=1),
            nn.BatchNorm2d(output_channel))

    def forward(self,inp):
        return self.layers(inp)


class BackBone(nn.Module):

    def __init__(self,num_channel=64):
        super().__init__()

        self.layers = nn.Sequential(
            ConvBlock(3,num_channel),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            ConvBlock(num_channel,num_channel),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            ConvBlock(num_channel,num_channel),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            ConvBlock(num_channel,num_channel),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2))
```

```
    def forward(self,inp):

        return self.layers(inp)
```

In [3]:
```
# FSRM (Feature Sequence Representation Module)

import torch
import torch.nn as nn
import torch.nn.functional as F


class Attention(nn.Module):
    """
    Feature Self-reconstruction Module
    """
    def __init__(self, dim, num_heads=8, attention_dropout=0.1):
        super().__init__()
        self.num_heads = num_heads
        head_dim = dim // self.num_heads
        self.scale = head_dim ** -0.5
        self.qkv = nn.Linear(dim, dim * 3, bias=False)
        self.attn_drop = nn.Dropout(attention_dropout)

        self.proj = nn.Linear(dim, dim)
        self.proj_drop = nn.Dropout(attention_dropout)

    def forward(self, x):
        B, N, C = x.shape
        qkv = self.qkv(x).reshape(B, N, 3, self.num_heads, C // self.num_heads).per
        q, k, v = qkv[0], qkv[1], qkv[2]

        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = attn.softmax(dim=-1)
        attn = self.attn_drop(attn)
        x = (attn @ v).transpose(1, 2).reshape(B, N, C)
        x = self.proj(x)
        x = self.proj_drop(x)
        return x


class TransformerEncoderLayer(nn.Module):
    """
    Inspired by torch.nn.TransformerEncoderLayer and
    rwightman's timm package.
    """
    def __init__(self, d_model, nhead, dropout=0.1, attention_dropout=0.1):
        super(TransformerEncoderLayer, self).__init__()

        self.self_attn = Attention(dim=d_model, num_heads=nhead, attention_dropout=
        self.pre_norm = nn.LayerNorm(d_model)
        self.linear1  = nn.Linear(d_model, d_model)
        self.linear2  = nn.Linear(d_model, d_model)
        self.norm1    = nn.LayerNorm(d_model)
        self.dropout1 = nn.Dropout(dropout)
        self.activation = F.gelu
```

```
    def forward(self, src: torch.Tensor, *args, **kwargs) -> torch.Tensor:
        src = src + self.self_attn(self.pre_norm(src))
        src = self.norm1(src)
        src2 = self.linear2(self.dropout1(self.activation(self.linear1(src))))
        src = src + self.dropout1(src2)

        return src



class Transformer(nn.Module):
    def __init__(self,
                 sequence_length=None,
                 embedding_dim=64,
                 num_layers=1,
                 num_heads=1,
                 attention_dropout=0.,
                 mlp_dropout_rate=0.,
                 positional_embedding='sine',
                 *args, **kwargs):
        super().__init__()
        positional_embedding = positional_embedding if \
            positional_embedding in ['sine', 'learnable', 'none'] else 'sine'
        self.embedding_dim = embedding_dim
        self.sequence_length = sequence_length

        assert sequence_length is not None or positional_embedding == 'none', \
            f"Positional embedding is set to {positional_embedding} and" \
            f" the sequence length was not specified."

        if positional_embedding != 'none':
            if positional_embedding == 'learnable':
                self.positional_emb = nn.Parameter(torch.zeros(1, sequence_length,
                                                   requires_grad=True)
                nn.init.trunc_normal_(self.positional_emb, std=0.2)
            else:
                self.positional_emb = nn.Parameter(self.sinusoidal_embedding(sequen
                                                   requires_grad=False)
        else:
            self.positional_emb = None

        self.blocks = nn.ModuleList([
            TransformerEncoderLayer(d_model=embedding_dim, nhead=num_heads,
                                    dropout=mlp_dropout_rate,
                                    attention_dropout=attention_dropout)
            for i in range(num_layers)])

        self.norm = nn.LayerNorm(embedding_dim)

        self.apply(self.init_weight)

    def forward(self, x):
        if self.positional_emb is not None:
            x += self.positional_emb
```

```
        for blk in self.blocks:
            x = blk(x)
        x = self.norm(x)
        return x

    @staticmethod
    def init_weight(m):
        if isinstance(m, nn.Linear):
            nn.init.trunc_normal_(m.weight, std=.02)
            if isinstance(m, nn.Linear) and m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.LayerNorm):
            nn.init.constant_(m.bias, 0)
            nn.init.constant_(m.weight, 1.0)

    @staticmethod
    def sinusoidal_embedding(n_channels, dim):
        pe = torch.FloatTensor([[p / (10000 ** (2 * (i // 2) / dim)) for i in range
                                 for p in range(n_channels)]])
        pe[:, 0::2] = torch.sin(pe[:, 0::2])
        pe[:, 1::2] = torch.cos(pe[:, 1::2])
        return pe.unsqueeze(0)


# FSRM Main model
class FSRM(nn.Module):
    def __init__(self,
                 sequence_length=25,
                 embedding_dim=64,
                 *args, **kwargs):
        super(FSRM, self).__init__()
        self.transformer = Transformer(sequence_length=sequence_length, embedding_d
        self.flattener = nn.Flatten(2, 3)

    def forward(self, x):
        x = self.flattener(x).transpose(-2, -1)
        x = self.transformer(x)
        return x
```

In [4]:

```
#FMRM (Feature Matching and Refinement Module)

import torch.nn as nn
import torch
import torch.nn.functional as F
import math


class FMRM(nn.Module):
    """
    Feature Mutual Reconstruction Module
    """
    def __init__(self, hidden_size, inner_size=None, num_patch=25, drop_prob=0.):
        super(FMRM, self).__init__()

        self.hidden_size = hidden_size
        self.inner_size = inner_size if inner_size is not None else hidden_size//8
```

```
        self.num_patch = num_patch

        dim_per_head = inner_size
        self.num_heads = 1
        inner_dim = self.inner_size * self.num_heads
        self.to_qkv = nn.Sequential(
            nn.Linear(self.hidden_size, inner_dim * 3, bias=False),
            )

        self.dropout = nn.Dropout(drop_prob)


    def compute_distances(self, query_a, key_a, value_a, query_b, key_b, value_b, f

        # 1) feature reconstruction
        value_a = value_a.unsqueeze(0)
        value_b = value_b.unsqueeze(1)

        n_way = value_a.size(1)
        n_query = value_b.size(0)
        s_patch = value_a.size(3)

        # Reconstructed features B
        att_scores = torch.matmul(query_b.unsqueeze(1), key_a.unsqueeze(0).transpos
        att_probs = nn.Softmax(dim=-1)(att_scores / math.sqrt(self.inner_size))
        att_probs = self.dropout(att_probs)
        # (N_query x N_way x 1 x HW x N-shot*HW) x (1 x N_way x 1 x N-shot*HW x C)
        reconstructed_features_b = torch.matmul(att_probs, value_a)

        # Reconstructed features A
        att_scores = torch.matmul(query_a.unsqueeze(0), key_b.unsqueeze(1).transpos
        att_probs = nn.Softmax(dim=-1)(att_scores / math.sqrt(self.inner_size))
        att_probs = self.dropout(att_probs)

        # (N_query x N_way x 1 x N-shot*HW x HW) x (N_query x 1 x 1 x HW x C) -> (N
        reconstructed_features_a = torch.matmul(att_probs, value_b)

        assert reconstructed_features_a.size(-1) == self.inner_size
        assert reconstructed_features_b.size(-1) == self.inner_size
        assert value_a.size(-1) == self.inner_size
        assert value_b.size(-1) == self.inner_size

        # 2) compute the Euclide distance
        sq_similarity = -torch.sum((value_a.view(value_a.size(0), value_a.size(1),
        qs_similarity = -torch.sum((value_b.view(value_b.size(0), value_b.size(1),

        return sq_similarity, qs_similarity

    def forward(self, features_a, features_b):
        # projection of features a
        features_a = features_a.view(features_a.size(0), features_a.size(1), -1).pe

        b_a, l_a, d_a = features_a.shape

        '''i. QKV projection'''
        # (b,l,dim_all_heads x 3)
```

```
        qkv_a = self.to_qkv(features_a)
        # (3,b,num_heads,l,dim_per_head)
        qkv_a = qkv_a.view(b_a, l_a, 3, self.num_heads, -1).permute(2, 0, 3, 1, 4).
        # 3 x (1,b,num_heads,l,dim_per_head)
        query_a, key_a, value_a = qkv_a.chunk(3)
        query_a, key_a, value_a = query_a.squeeze(0), key_a.squeeze(0), value_a.squ



        # projection of features b
        features_b = features_b.view(features_b.size(0), features_b.size(1), -1).pe

        b_b, l_b, d_b = features_b.shape

        '''i. QKV projection'''
        # (b,l,dim_all_heads x 3)
        qkv_b = self.to_qkv(features_b)
        # (3,b,num_heads,l,dim_per_head)
        qkv_b = qkv_b.view(b_b, l_b, 3, self.num_heads, -1).permute(2, 0, 3, 1, 4).
        # 3 x (1,b,num_heads,l,dim_per_head)
        query_b, key_b, value_b = qkv_b.chunk(3)
        query_b, key_b, value_b = query_b.squeeze(0), key_b.squeeze(0), value_b.squ

        # compute the total spatial similarity
        distances = self.compute_distances(query_a, key_a, value_a, query_b, key_b,

        return distances
```

In [5]:

```
# BiFRN

import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision.models as torch_models
import numpy as np


class BiFRN(nn.Module):

    def __init__(self,way=None,shots=None,resnet=False):

        super().__init__()

        # Pull out the features using ResNet or Conv-4
        self.resolution = 5*5
        if resnet:
            self.num_channel = 640
            self.feature_extractor = ResNet.resnet12()
            self.dim = self.num_channel*5*5

        else:
            self.num_channel = 64
            self.feature_extractor = BackBone(self.num_channel)
            self.dim = self.num_channel*5*5

        # Create an FSRM instance
```

```
        self.fsrm = FSRM(
                sequence_length=self.resolution,
                embedding_dim=self.num_channel,
                num_layers=1,
                num_heads=1,
                mlp_dropout_rate=0.,
                attention_dropout=0.,
                positional_embedding='sine')

        # Create an FMRM instance
        self.fmrm = FMRM(hidden_size=self.num_channel,
                         inner_size=self.num_channel,
                         num_patch=self.resolution,
                         drop_prob=0.1
                         )

        self.shots = shots
        self.way = way
        self.resnet = resnet

        # Setup the scale and weights for learning
        self.scale = nn.Parameter(torch.FloatTensor([1.0]),requires_grad=True)
        self.w1 = nn.Parameter(torch.FloatTensor([0.5]),requires_grad=True)
        self.w2 = nn.Parameter(torch.FloatTensor([0.5]),requires_grad=True)


    def get_feature_vector(self,inp):

        batch_size = inp.size(0)
        feature_map = self.feature_extractor(inp)
        feature_map = self.fsrm(feature_map).transpose(1, 2).view(batch_size, self.

        return feature_map


    def get_neg_l2_dist(self,inp,way,shot,query_shot):

        feature_map = self.get_feature_vector(inp)
        support = feature_map[:way*shot].view(way, shot, *feature_map.size()[1:]).p
        query = feature_map[way*shot:]

        sq_similarity, qs_similarity = self.fmrm(support, query)

        l2_dist = self.w1*sq_similarity + self.w2*qs_similarity

        return l2_dist




    def meta_test(self,inp,way,shot,query_shot):

        neg_l2_dist = self.get_neg_l2_dist(inp=inp,
                                           way=way,
                                           shot=shot,
                                           query_shot=query_shot)
```

```
        _,max_index = torch.max(neg_l2_dist,1)

        return max_index


    def forward(self,inp):

        logits = self.get_neg_l2_dist(inp=inp,
                                      way=self.way,
                                      shot=self.shots[0],
                                      query_shot=self.shots[1])
        logits = logits/self.dim*self.scale

        log_prediction = F.log_softmax(logits,dim=1)

        return log_prediction
```

In [6]:
```
# Path Manager
import os
import sys
import torch
import torch.optim as optim
import logging
import numpy as np
import argparse
from tqdm import tqdm

class Path_Manager:

    def __init__(self,fewshot_path,args):

        self.train = os.path.join(fewshot_path,'train')

        if args.pre:
            self.test = os.path.join(fewshot_path,'test_pre')
            self.val = os.path.join(fewshot_path,'val_pre') if not args.no_val else

        else:
            self.test = os.path.join(fewshot_path,'test')
            self.val = os.path.join(fewshot_path,'val') if not args.no_val else sel

        # if not os.path.exists(self.train):
        #     raise FileNotFoundError(f"Training directory not found: {self.train}"

        # if not os.path.exists(self.test):
        #     raise FileNotFoundError(f"Testing directory not found: {self.test}")

        # if not os.path.exists(self.val):
        #     raise FileNotFoundError(f"Validation directory not found: {self.val}"
```

In [16]:
```
import os
import math
import sys
import torch
```

```
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import numpy as np
import logging
import argparse
import yaml
#sys.path.append('..')
from tensorboardX import SummaryWriter
from torch.nn import NLLLoss
from copy import deepcopy
from PIL import Image
from functools import partial
from argparse import Namespace
from tqdm import tqdm
from collections import defaultdict
from joblib import Parallel, delayed

#FUNCTIONS
#******************************************************************************
# IMAGE LOADER FUNCTION
# Use to transform the image
#******************************************************************************
def image_loader(path,is_training,transform_type,pre):
  # Load image
  p = Image.open(path)
  p = p.convert('RGB')

  final_transform = get_transform(is_training=is_training,transform_type=transform_

  p = final_transform(p)

  return p

#******************************************************************************

#******************************************************************************
# TRANSFORM FUNCTION
# Use to get the final transforms of images
#******************************************************************************
def get_transform(is_training=None,transform_type=None,pre=None):

  if is_training and pre:
      raise Exception('is_training and pre cannot be specified as True at the same

  if transform_type and pre:
      raise Exception('transform_type and pre cannot be specified as True at the sa

  mean=[0.485,0.456,0.406]
  std=[0.229,0.224,0.225]

  normalize = transforms.Compose([transforms.ToTensor(),
                                  transforms.Normalize(mean=mean,std=std)
                                  ])
```

```
    if is_training:

        if transform_type == 0:
            size_transform = transforms.RandomResizedCrop(84)
        elif transform_type == 1:
            size_transform = transforms.RandomCrop(84,padding=8)
        else:
            raise Exception('transform_type must be specified during training!')

        train_transform = transforms.Compose([size_transform,
                                              transforms.ColorJitter(brightness=0.4,con
                                              transforms.RandomHorizontalFlip(),
                                              normalize
                                              ])
        return train_transform

    elif pre:
        return normalize

    else:

        if transform_type == 0:
            size_transform = transforms.Compose([transforms.Resize(92),
                                                 transforms.CenterCrop(84)])
        elif transform_type == 1:
            size_transform = transforms.Compose([transforms.Resize([92,92]),
                                                 transforms.CenterCrop(84)])
        elif transform_type == 2:
            # for tiered-imagenet and (tiered) meta-inat where val/test images are al
            return normalize

        else:
            raise Exception('transform_type must be specified during inference if not

        eval_transform = transforms.Compose([size_transform,normalize])
        return eval_transform
#*********************************************************************************

#*********************************************************************************
# TRAIN 1 EPOCH FUNCTION
# Use to train the model for 1 epoch
#*********************************************************************************
def default_train(train_loader,model,optimizer,writer,iter_counter):

    # Setup training variables
    way = model.way
    query_shot = model.shots[-1]
    target = torch.LongTensor([i//query_shot for i in range(query_shot*way)]).cuda(
    criterion = NLLLoss().cuda()

    # Log initial values
    lr = optimizer.param_groups[0]['lr']

    writer.add_scalar('lr',lr,iter_counter)
    writer.add_scalar('W1',model.w1.item(),iter_counter)
```

```
    writer.add_scalar('W2',model.w2.item(),iter_counter)
    writer.add_scalar('scale',model.scale.item(),iter_counter)

    avg_loss = 0
    avg_acc = 0

    # Loop batches
    for i, (inp,_) in enumerate(train_loader):

        iter_counter += 1

        # Forward pass
        inp = inp.cuda()
        log_prediction = model(inp)

        loss = criterion(log_prediction,target)

        # Backpropogation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        loss_value = loss.item()

        # Compute accuracy
        _,max_index = torch.max(log_prediction,1)
        acc = 100*torch.sum(torch.eq(max_index,target)).item()/query_shot/way

        avg_acc += acc
        avg_loss += loss_value

    avg_acc = avg_acc/(i+1)
    avg_loss = avg_loss/(i+1)

    writer.add_scalar('proto_loss',avg_loss,iter_counter)
    writer.add_scalar('train_acc',avg_acc,iter_counter)

    return iter_counter,avg_acc
#**************************************************************************

#**************************************************************************
# FILENAME LOGGER FUNCTION
# Configures a logger to log the execution during training (will help for compariso
#**************************************************************************
def get_logger(filename):

    formatter = logging.Formatter(
        "[%(asctime)s] %(message)s",datefmt='%m/%d %I:%M:%S')
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)

    fh = logging.FileHandler(filename,"w")
    fh.setFormatter(formatter)
    logger.addHandler(fh)

    sh = logging.StreamHandler()
```

```
    sh.setFormatter(formatter)
    logger.addHandler(sh)

    return logger
#*******************************************************************************

#*******************************************************************************
# TRAINING FUNCTION
# Train the model
#*******************************************************************************
def train(model):

    args = trainer_args
    train_func = trainer_train_func
    writer = trainer_writer
    save_path = trainer_save_path
    logger = trainer_logger

    optimizer,scheduler = get_opt(model,args)

    val_shot = args.train_shot
    test_way = args.test_way

    best_val_acc = 0
    best_epoch = 0

    model.train()
    model.cuda()

    iter_counter = 0

    if args.decay_epoch is not None:
        total_epoch = args.epoch
    else:
        total_epoch = args.epoch*args.stage

    logger.info("start training!")

    for e in tqdm(range(total_epoch)):

        iter_counter,train_acc = train_func(model=model,
                                            optimizer=optimizer,
                                            writer=writer,
                                            iter_counter=iter_counter
                                            )

        if (e+1)%args.val_epoch==0:

            logger.info("")
            logger.info("epoch %d/%d, iter %d:" % (e+1,total_epoch,iter_counter))
            logger.info("train_acc: %.3f" % (train_acc))

            model.eval()
            with torch.no_grad():
              val_acc,val_interval = meta_test(data_path=pm.val,
                                               model=model,
```

```
                                            way=test_way,
                                            shot=val_shot,
                                            pre=args.pre,
                                            transform_type=args.test_transform_
                                            query_shot=args.test_query_shot,
                                            trial=args.val_trial,
                                            return_list=False
                                            )
              writer.add_scalar('val_%d-way-%d-shot_acc'%(test_way,val_shot),val_ac

            logger.info('val_%d-way-%d-shot_acc: %.3f\t%.3f'%(test_way,val_shot,val

            if val_acc > best_val_acc:
                best_val_acc = val_acc
                best_epoch = e+1
                if not args.no_val:
                    torch.save(model.state_dict(),save_path)
                logger.info('BEST!')

            model.train()

        scheduler.step()

    logger.info('training finished!')
    if args.no_val:
        torch.save(model.state_dict(),save_path)

    logger.info('------------------------')
    logger.info(('the best epoch is %d/%d') % (best_epoch,total_epoch))
    logger.info(('the best %d-way %d-shot val acc is %.3f') % (test_way,val_shot,be
#**************************************************************************

#**************************************************************************
# EVALUATE FUNCTION
# Evaluate the model
#**************************************************************************
def evaluate(model):

  logger = trainer_logger
  args = trainer_args

  logger.info('------------------------')
  logger.info('evaluating on test set:')
  print('\n *******Starting Evaluate******** \n')
  with torch.no_grad():

      model.load_state_dict(torch.load(trainer_save_path))
      model.eval()
      for shot in args.test_shot:
          mean,interval = meta_test(data_path=pm.test,
                                    model=model,
                                    way=args.test_way,
                                    shot=args.train_shot,
                                    pre=args.pre,
                                    transform_type=args.test_transform_type,
                                    query_shot=args.test_query_shot,
```

```
                                    trial=args.val_trial)

          logger.info('%d-way-%d-shot acc: %.2f\t%.2f'%(args.test_way,shot,mean,int
#**************************************************************************

#**************************************************************************
# OPIMIZER and SCHEDULER FUNCTION
# Backpropogation
#**************************************************************************
def get_opt(model,args):

    if args.opt == 'adam':
        optimizer = optim.Adam(model.parameters(),
                               lr=args.lr,
                               weight_decay=args.weight_decay
                               )
    elif args.opt == 'sgd':
        optimizer = optim.SGD(model.parameters(),
                              lr=args.lr,
                              momentum=0.9,
                              weight_decay=args.weight_decay,
                              nesterov=args.nesterov
                              )

    if args.decay_epoch is not None:
        scheduler = optim.lr_scheduler.MultiStepLR(optimizer,
                                                   milestones=args.decay_epoch,
                                                   gamma=args.gamma
                                                   )

    else:
        scheduler = optim.lr_scheduler.StepLR(optimizer,
                                              step_size=args.epoch,
                                              gamma=args.gamma
                                              )

    return optimizer,scheduler
#**************************************************************************

#**************************************************************************
# TESTING FOR EVALUATION FUNCTION
# Evaluates the Model
#**************************************************************************
def meta_test(data_path,model,way,shot,pre,transform_type,query_shot=16,trial=10000
  # SETTING UP THE DATALOADER
  # Setup training vars
  train_way = args.train_way                        # Num of classes in each episode
  shots = [args.train_shot, args.train_query_shot]  # Num of images per class

  # Get Dataset
  dataset = datasets.ImageFolder(data_path,
                                 loader = lambda x: image_loader(path=x,is_training
                                 )

  # Setup data for episode-based sampling during training (the sampler)
  class2id = defaultdict(list)
```

```
  for i,(image_data,class_id) in enumerate(dataset):
    # if class_id not in class2id:
    #   class2id[class_id]=[]
    class2id[class_id].append(i)

  temp_class2id = deepcopy(class2id)
  list_class_id = list(temp_class2id.keys())

  # Create a list to store id_lists for each trial
  all_id_lists = []

  for i in range(trial):

    id_list = []

    np.random.shuffle(list_class_id)
    picked_class = list_class_id[:way]

    for cat in picked_class:
        np.random.shuffle(class2id[cat])

    for cat in picked_class:
        id_list.extend(class2id[cat][:shot])
    for cat in picked_class:
        id_list.extend(class2id[cat][shot:(shot+query_shot)])

    # Append the id_list for the current trial
    all_id_lists.append(id_list)

  # Setup the dataloader for eval
  eval_loader = torch.utils.data.DataLoader(dataset,
                                            batch_sampler = all_id_lists,
                                            num_workers = 3,
                                            pin_memory = False
                                            )

  target = torch.LongTensor([i//query_shot for i in range(query_shot*way)]).cuda()

  acc_list = []

  for i, (inp,_) in tqdm(enumerate(eval_loader)):
    inp = inp.cuda()
    max_index = model.meta_test(inp,way=way,shot=shot,query_shot=query_shot)

    acc = 100*torch.sum(torch.eq(max_index,target)).item()/query_shot/way
    acc_list.append(acc)

  if return_list:
    return np.array(acc_list)
  else:
    mean = np.mean(acc_list)
    interval = 1.96*np.sqrt(np.var(acc_list)/len(acc_list))
    return mean,interval
#********************************************************************************
```

```
# MAIN BLOCK
# Setup all of the arguments
args = Namespace(
    opt='sgd',            # The optimizer ('adam' or 'sgd')
    lr=0.1,               # Learning rate for training
    gamma=0.01,            # Scalar to reduce the learning rate by at specific epoc
    epoch=20,              # Number of epochs before reducing the learning rate
    stage=3,              # Number of learning rate stages during training
    weight_decay=0.0005,  # Weight decay for regularizing the optimizer
    gpu=0,                # The GPU device to use
    seed=42,              # Random seed for reproducibility
    val_epoch=5,         # Frequency of validation
    resnet=False,         # ACTION: is ResNet-12 the backbone?
    nesterov=False,       # ACTION: Use Nesterov momentum with SGD?
    batch_size=0,         # Batch size for pre-training
    decay_epoch=None,     # Specific epochs at which to decay the learning rate
    pre=False,            # ACTION: Use pre-resized 84x84 images for validation and
    no_val=False,         # ACTION: Skip validation, only saving the model at the e
    train_way=20,         # Number of classes used during training episodes
    test_way=5,           # Number of classes used during testing episodes
    train_shot=5,         # Number of support images per class during training
    test_shot=[1, 5],     # Number of support images per class for testing
    train_query_shot=15,  # Number of query images per class during training
    test_query_shot=16,   # Number of query images per class during testing
    train_transform_type=0, # Type of transformation applied during training
    test_transform_type=0,  # Type of transformation applied during testing
    val_trial=50,         # Number of meta-testing episodes to use during validatio
    detailed_name=True      # ACTION: Include training details in the model name fo
)

# Setup the file path
data_path = "/content/dogs"

# data_path = os.path.abspath('/content/drive/MyDrive/ECE570-Project/')
# fewshot_path = "/content/drive/MyDrive/ECE570-Project/dogs"

pm = Path_Manager(fewshot_path=data_path,args=args)

print('data_path: %s' % (data_path))
print('pm.train: %s' % (pm.train))
print('pm.val: %s' % (pm.val))
print('pm.test: %s' % (pm.test))

# SETTING UP THE DATALOADER
# Setup training vars
train_way = args.train_way                        # Num of classes in each episode
shots = [args.train_shot, args.train_query_shot]  # Num of images per class

# Get Dataset
dataset = datasets.ImageFolder(pm.train,
                               loader = lambda x: image_loader(path=x,is_training=T
                               )

print('dataset length: %d' % (len(dataset)))

# Setup data for episode-based sampling during training (the sampler)
```

```
class2id = defaultdict(list)

for i,(image_data,class_id) in enumerate(dataset):

    if i%2000==0:
      print('images added: %d' % (i))
    elif i>12140:
      print('images added: %d' % (i))
    # if class_id not in class2id:
    #   class2id[class_id]=[]
    class2id[class_id].append(i)

temp_class2id = deepcopy(class2id)

# Parallel shuffle each class list
def shuffle_class_ids(c_id, temp_class2id):
    np.random.shuffle(temp_class2id[c_id])

Parallel(n_jobs=-1)(delayed(shuffle_class_ids)(c_id, temp_class2id) for c_id in tem

# Accumulate id_list across iterations
all_id_lists = []

while len(temp_class2id) >= train_way:

    id_list = []

    list_class_id = list(temp_class2id.keys())

    pcount = np.array([len(temp_class2id[c_id]) for c_id in list_class_id])

    batch_class_id = np.random.choice(list_class_id,
                                      size=train_way,
                                      replace=False,
                                      p=pcount/sum(pcount)
                                      )

    for shot in shots:
        for c_id in batch_class_id:
            for _ in range(shot):
                id_list.append(temp_class2id[c_id].pop(0))

    for c_id in batch_class_id:
        if len(temp_class2id[c_id])<sum(shots):
            temp_class2id.pop(c_id)

    # Append the current id_list to the accumulator
    all_id_lists.append(id_list)

# Setup the dataloader
train_loader = torch.utils.data.DataLoader(dataset,
                                           batch_sampler = all_id_lists,
                                           num_workers = 3,
                                           pin_memory = False
                                           )
```

```
# Setup the model: BiFRN
model = BiFRN(way=train_way,
              shots=[args.train_shot, args.train_query_shot],
              resnet=args.resnet
              )

# TRAINING
# Lock the train_loader for train function to keep the code clean
train_func = partial(default_train,train_loader=train_loader)

# Setup the training variables
seed = args.seed
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
np.random.seed(seed)
torch.cuda.set_device(args.gpu)

if args.resnet:
    name = 'ResNet-12'
else:
    name = 'Conv-4'

if args.detailed_name:
    if args.decay_epoch is not None:
        temp = ''
        for i in args.decay_epoch:
            temp += ('_'+str(i))

        suffix = '%s-lr_%.0e-gamma_%.0e-epoch_%d-drop%s-decay_%.0e-way_%d' % (args.
            args.lr,args.gamma,args.epoch,temp,args.weight_decay,args.train_way)
    else:
        suffix = '%s-lr_%.0e-gamma_%.0e-epoch_%d-stage_%d-decay_%.0e-way_%d' % (arg
            args.lr,args.gamma,args.epoch,args.stage,args.weight_decay,args.train_w

    name = "%s-%s"%(name,suffix)

# Log the parameters and settings
trainer_logger = get_logger('%s.log' % (name))
trainer_save_path = 'model_%s.pth' % (name)
trainer_writer = SummaryWriter('log_%s' % (name))

trainer_logger.info('display all the hyper-parameters in args:')
for arg in vars(args):
    value = getattr(args,arg)
    if value is not None:
        trainer_logger.info('%s: %s' % (str(arg),str(value)))
trainer_logger.info('------------------------')
trainer_args = args
trainer_train_func = train_func
trainer_pm = pm.train

# Train and evaluate the model
train(model)
evaluate(model)
```

```
data_path: /content/dogs
pm.train: /content/dogs/train
pm.val: /content/dogs/val
pm.test: /content/dogs/test
dataset length: 12145
images added: 0
images added: 2000
images added: 4000
images added: 6000
images added: 8000
images added: 10000
images added: 12000
images added: 12141
images added: 12142
images added: 12143
images added: 12144
```

```
INFO:root:display all the hyper-parameters in args:
[11/10 11:26:34] display all the hyper-parameters in args:
[11/10 11:26:34] display all the hyper-parameters in args:
[11/10 11:26:34] display all the hyper-parameters in args:
[11/10 11:26:34] display all the hyper-parameters in args:
[11/10 11:26:34] display all the hyper-parameters in args:
INFO:root:opt: sgd
[11/10 11:26:34] opt: sgd
[11/10 11:26:34] opt: sgd
[11/10 11:26:34] opt: sgd
[11/10 11:26:34] opt: sgd
[11/10 11:26:34] opt: sgd
INFO:root:lr: 0.1
[11/10 11:26:34] lr: 0.1
[11/10 11:26:34] lr: 0.1
[11/10 11:26:34] lr: 0.1
[11/10 11:26:34] lr: 0.1
[11/10 11:26:34] lr: 0.1
INFO:root:gamma: 0.01
[11/10 11:26:34] gamma: 0.01
[11/10 11:26:34] gamma: 0.01
[11/10 11:26:34] gamma: 0.01
[11/10 11:26:34] gamma: 0.01
[11/10 11:26:34] gamma: 0.01
INFO:root:epoch: 20
[11/10 11:26:34] epoch: 20
[11/10 11:26:34] epoch: 20
[11/10 11:26:34] epoch: 20
[11/10 11:26:34] epoch: 20
[11/10 11:26:34] epoch: 20
INFO:root:stage: 3
[11/10 11:26:34] stage: 3
[11/10 11:26:34] stage: 3
[11/10 11:26:34] stage: 3
[11/10 11:26:34] stage: 3
[11/10 11:26:34] stage: 3
INFO:root:weight_decay: 0.0005
[11/10 11:26:34] weight_decay: 0.0005
[11/10 11:26:34] weight_decay: 0.0005
[11/10 11:26:34] weight_decay: 0.0005
[11/10 11:26:34] weight_decay: 0.0005
[11/10 11:26:34] weight_decay: 0.0005
INFO:root:gpu: 0
[11/10 11:26:34] gpu: 0
[11/10 11:26:34] gpu: 0
[11/10 11:26:34] gpu: 0
[11/10 11:26:34] gpu: 0
[11/10 11:26:34] gpu: 0
INFO:root:seed: 42
[11/10 11:26:34] seed: 42
[11/10 11:26:34] seed: 42
[11/10 11:26:34] seed: 42
[11/10 11:26:34] seed: 42
[11/10 11:26:34] seed: 42
INFO:root:val_epoch: 5
[11/10 11:26:34] val_epoch: 5
```

```
[11/10 11:26:34] val_epoch: 5
[11/10 11:26:34] val_epoch: 5
[11/10 11:26:34] val_epoch: 5
[11/10 11:26:34] val_epoch: 5
INFO:root:resnet: False
[11/10 11:26:34] resnet: False
[11/10 11:26:34] resnet: False
[11/10 11:26:34] resnet: False
[11/10 11:26:34] resnet: False
[11/10 11:26:34] resnet: False
INFO:root:nesterov: False
[11/10 11:26:34] nesterov: False
[11/10 11:26:34] nesterov: False
[11/10 11:26:34] nesterov: False
[11/10 11:26:34] nesterov: False
[11/10 11:26:34] nesterov: False
INFO:root:batch_size: 0
[11/10 11:26:34] batch_size: 0
[11/10 11:26:34] batch_size: 0
[11/10 11:26:34] batch_size: 0
[11/10 11:26:34] batch_size: 0
[11/10 11:26:34] batch_size: 0
INFO:root:pre: False
[11/10 11:26:34] pre: False
[11/10 11:26:34] pre: False
[11/10 11:26:34] pre: False
[11/10 11:26:34] pre: False
[11/10 11:26:34] pre: False
INFO:root:no_val: False
[11/10 11:26:34] no_val: False
[11/10 11:26:34] no_val: False
[11/10 11:26:34] no_val: False
[11/10 11:26:34] no_val: False
[11/10 11:26:34] no_val: False
INFO:root:train_way: 20
[11/10 11:26:34] train_way: 20
[11/10 11:26:34] train_way: 20
[11/10 11:26:34] train_way: 20
[11/10 11:26:34] train_way: 20
[11/10 11:26:34] train_way: 20
INFO:root:test_way: 5
[11/10 11:26:34] test_way: 5
[11/10 11:26:34] test_way: 5
[11/10 11:26:34] test_way: 5
[11/10 11:26:34] test_way: 5
[11/10 11:26:34] test_way: 5
INFO:root:train_shot: 5
[11/10 11:26:34] train_shot: 5
[11/10 11:26:34] train_shot: 5
[11/10 11:26:34] train_shot: 5
[11/10 11:26:34] train_shot: 5
[11/10 11:26:34] train_shot: 5
INFO:root:test_shot: [1, 5]
[11/10 11:26:34] test_shot: [1, 5]
[11/10 11:26:34] test_shot: [1, 5]
[11/10 11:26:34] test_shot: [1, 5]
```

```
[11/10 11:26:34] test_shot: [1, 5]
[11/10 11:26:34] test_shot: [1, 5]
INFO:root:train_query_shot: 15
[11/10 11:26:34] train_query_shot: 15
[11/10 11:26:34] train_query_shot: 15
[11/10 11:26:34] train_query_shot: 15
[11/10 11:26:34] train_query_shot: 15
[11/10 11:26:34] train_query_shot: 15
INFO:root:test_query_shot: 16
[11/10 11:26:34] test_query_shot: 16
[11/10 11:26:34] test_query_shot: 16
[11/10 11:26:34] test_query_shot: 16
[11/10 11:26:34] test_query_shot: 16
[11/10 11:26:34] test_query_shot: 16
INFO:root:train_transform_type: 0
[11/10 11:26:34] train_transform_type: 0
[11/10 11:26:34] train_transform_type: 0
[11/10 11:26:34] train_transform_type: 0
[11/10 11:26:34] train_transform_type: 0
[11/10 11:26:34] train_transform_type: 0
INFO:root:test_transform_type: 0
[11/10 11:26:34] test_transform_type: 0
[11/10 11:26:34] test_transform_type: 0
[11/10 11:26:34] test_transform_type: 0
[11/10 11:26:34] test_transform_type: 0
[11/10 11:26:34] test_transform_type: 0
INFO:root:val_trial: 50
[11/10 11:26:34] val_trial: 50
[11/10 11:26:34] val_trial: 50
[11/10 11:26:34] val_trial: 50
[11/10 11:26:34] val_trial: 50
[11/10 11:26:34] val_trial: 50
INFO:root:detailed_name: True
[11/10 11:26:34] detailed_name: True
[11/10 11:26:34] detailed_name: True
[11/10 11:26:34] detailed_name: True
[11/10 11:26:34] detailed_name: True
[11/10 11:26:34] detailed_name: True
INFO:root:------------------------
[11/10 11:26:34] ------------------------
[11/10 11:26:34] ------------------------
[11/10 11:26:34] ------------------------
[11/10 11:26:34] ------------------------
[11/10 11:26:34] ------------------------
INFO:root:start training!
[11/10 11:26:34] start training!
[11/10 11:26:34] start training!
[11/10 11:26:34] start training!
[11/10 11:26:34] start training!
[11/10 11:26:34] start training!
  7%|▉          | 4/60 [00:53<12:22, 13.26s/it]INFO:root:
[11/10 11:27:40]
[11/10 11:27:40]
[11/10 11:27:40]
[11/10 11:27:40]
[11/10 11:27:40]
```

```
INFO:root:epoch 5/60, iter 140:
[11/10 11:27:40] epoch 5/60, iter 140:
[11/10 11:27:40] epoch 5/60, iter 140:
[11/10 11:27:40] epoch 5/60, iter 140:
[11/10 11:27:40] epoch 5/60, iter 140:
[11/10 11:27:40] epoch 5/60, iter 140:
INFO:root:train_acc: 11.655
[11/10 11:27:40] train_acc: 11.655
[11/10 11:27:40] train_acc: 11.655
[11/10 11:27:40] train_acc: 11.655
[11/10 11:27:40] train_acc: 11.655
[11/10 11:27:40] train_acc: 11.655

0it [00:00, ?it/s]
1it [00:00,  2.90it/s]
4it [00:00,  6.70it/s]
7it [00:00,  7.66it/s]
10it [00:01,  7.62it/s]
13it [00:01,  7.97it/s]
16it [00:02,  8.61it/s]
19it [00:02,  8.85it/s]
22it [00:02,  9.10it/s]
25it [00:03,  9.06it/s]
28it [00:03,  9.23it/s]
31it [00:03,  9.34it/s]
34it [00:03,  9.23it/s]
37it [00:04,  9.54it/s]
40it [00:04,  8.98it/s]
43it [00:04,  8.98it/s]
46it [00:05,  9.21it/s]
50it [00:05,  8.91it/s]
INFO:root:val_5-way-5-shot_acc: 40.400  2.626
[11/10 11:27:56] val_5-way-5-shot_acc: 40.400   2.626
[11/10 11:27:56] val_5-way-5-shot_acc: 40.400   2.626
[11/10 11:27:56] val_5-way-5-shot_acc: 40.400   2.626
[11/10 11:27:56] val_5-way-5-shot_acc: 40.400   2.626
[11/10 11:27:56] val_5-way-5-shot_acc: 40.400   2.626
INFO:root:BEST!
[11/10 11:27:56] BEST!
[11/10 11:27:56] BEST!
[11/10 11:27:56] BEST!
[11/10 11:27:56] BEST!
[11/10 11:27:56] BEST!
 15%|          | 9/60 [02:14<12:12, 14.36s/it]INFO:root:
[11/10 11:29:02]
[11/10 11:29:02]
[11/10 11:29:02]
[11/10 11:29:02]
[11/10 11:29:02]
INFO:root:epoch 10/60, iter 280:
[11/10 11:29:02] epoch 10/60, iter 280:
[11/10 11:29:02] epoch 10/60, iter 280:
[11/10 11:29:02] epoch 10/60, iter 280:
[11/10 11:29:02] epoch 10/60, iter 280:
[11/10 11:29:02] epoch 10/60, iter 280:
INFO:root:train_acc: 14.000
```

```
[11/10 11:29:02] train_acc: 14.000
[11/10 11:29:02] train_acc: 14.000
[11/10 11:29:02] train_acc: 14.000
[11/10 11:29:02] train_acc: 14.000
[11/10 11:29:02] train_acc: 14.000

0it [00:00, ?it/s]
1it [00:00,  2.52it/s]
4it [00:00,  5.76it/s]
7it [00:01,  6.95it/s]
10it [00:01,  7.95it/s]
13it [00:01,  8.44it/s]
16it [00:02,  9.06it/s]
19it [00:02,  9.26it/s]
22it [00:02,  9.37it/s]
25it [00:02,  9.43it/s]
28it [00:03,  9.29it/s]
31it [00:03,  9.40it/s]
34it [00:03,  8.90it/s]
37it [00:04,  8.92it/s]
40it [00:04,  9.01it/s]
43it [00:04,  8.90it/s]
46it [00:05,  9.16it/s]
50it [00:05,  8.82it/s]
INFO:root:val_5-way-5-shot_acc: 42.475  2.098
[11/10 11:29:17] val_5-way-5-shot_acc: 42.475   2.098
[11/10 11:29:17] val_5-way-5-shot_acc: 42.475   2.098
[11/10 11:29:17] val_5-way-5-shot_acc: 42.475   2.098
[11/10 11:29:17] val_5-way-5-shot_acc: 42.475   2.098
[11/10 11:29:17] val_5-way-5-shot_acc: 42.475   2.098
INFO:root:BEST!
[11/10 11:29:17] BEST!
[11/10 11:29:17] BEST!
[11/10 11:29:17] BEST!
[11/10 11:29:17] BEST!
[11/10 11:29:17] BEST!
 23%|██▎       | 14/60 [03:36<11:12, 14.62s/it]INFO:root:
[11/10 11:30:24]
[11/10 11:30:24]
[11/10 11:30:24]
[11/10 11:30:24]
[11/10 11:30:24]
INFO:root:epoch 15/60, iter 420:
[11/10 11:30:24] epoch 15/60, iter 420:
[11/10 11:30:24] epoch 15/60, iter 420:
[11/10 11:30:24] epoch 15/60, iter 420:
[11/10 11:30:24] epoch 15/60, iter 420:
[11/10 11:30:24] epoch 15/60, iter 420:
INFO:root:train_acc: 16.262
[11/10 11:30:24] train_acc: 16.262
[11/10 11:30:24] train_acc: 16.262
[11/10 11:30:24] train_acc: 16.262
[11/10 11:30:24] train_acc: 16.262
[11/10 11:30:24] train_acc: 16.262

0it [00:00, ?it/s]
```

```
1it [00:00,  2.90it/s]
4it [00:00,  6.77it/s]
7it [00:00,  7.86it/s]
10it [00:01,  8.78it/s]
13it [00:01,  8.67it/s]
16it [00:01,  9.06it/s]
19it [00:02,  9.07it/s]
22it [00:02,  9.39it/s]
25it [00:02,  9.30it/s]
27it [00:02, 10.42it/s]
29it [00:03, 10.14it/s]
31it [00:03,  8.98it/s]
33it [00:03,  9.66it/s]
35it [00:03,  9.50it/s]
37it [00:04,  8.29it/s]
39it [00:04,  9.75it/s]
41it [00:04,  8.43it/s]
43it [00:04,  7.67it/s]
46it [00:05,  8.09it/s]
50it [00:05,  8.82it/s]
INFO:root:val_5-way-5-shot_acc: 43.075  2.205
[11/10 11:30:39] val_5-way-5-shot_acc: 43.075   2.205
[11/10 11:30:39] val_5-way-5-shot_acc: 43.075   2.205
[11/10 11:30:39] val_5-way-5-shot_acc: 43.075   2.205
[11/10 11:30:39] val_5-way-5-shot_acc: 43.075   2.205
[11/10 11:30:39] val_5-way-5-shot_acc: 43.075   2.205
INFO:root:BEST!
[11/10 11:30:39] BEST!
[11/10 11:30:39] BEST!
[11/10 11:30:39] BEST!
[11/10 11:30:39] BEST!
[11/10 11:30:39] BEST!
 32%|███▏      | 19/60 [04:58<10:00, 14.65s/it]INFO:root:
[11/10 11:31:46]
[11/10 11:31:46]
[11/10 11:31:46]
[11/10 11:31:46]
[11/10 11:31:46]
INFO:root:epoch 20/60, iter 560:
[11/10 11:31:46] epoch 20/60, iter 560:
[11/10 11:31:46] epoch 20/60, iter 560:
[11/10 11:31:46] epoch 20/60, iter 560:
[11/10 11:31:46] epoch 20/60, iter 560:
[11/10 11:31:46] epoch 20/60, iter 560:
INFO:root:train_acc: 17.560
[11/10 11:31:46] train_acc: 17.560
[11/10 11:31:46] train_acc: 17.560
[11/10 11:31:46] train_acc: 17.560
[11/10 11:31:46] train_acc: 17.560
[11/10 11:31:46] train_acc: 17.560

0it [00:00, ?it/s]
1it [00:00,  2.87it/s]
4it [00:00,  5.87it/s]
7it [00:01,  7.00it/s]
10it [00:01,  7.71it/s]
```

```
13it [00:01,  8.23it/s]
16it [00:02,  8.48it/s]
19it [00:02,  8.88it/s]
22it [00:02,  9.22it/s]
25it [00:03,  9.09it/s]
28it [00:03,  9.32it/s]
31it [00:03,  9.22it/s]
34it [00:03,  9.47it/s]
37it [00:04,  9.47it/s]
40it [00:04,  9.56it/s]
43it [00:04,  9.49it/s]
46it [00:05,  9.51it/s]
50it [00:05,  8.84it/s]
INFO:root:val_5-way-5-shot_acc: 47.325  2.721
[11/10 11:32:01] val_5-way-5-shot_acc: 47.325   2.721
[11/10 11:32:01] val_5-way-5-shot_acc: 47.325   2.721
[11/10 11:32:01] val_5-way-5-shot_acc: 47.325   2.721
[11/10 11:32:01] val_5-way-5-shot_acc: 47.325   2.721
[11/10 11:32:01] val_5-way-5-shot_acc: 47.325   2.721
INFO:root:BEST!
[11/10 11:32:01] BEST!
[11/10 11:32:01] BEST!
[11/10 11:32:01] BEST!
[11/10 11:32:01] BEST!
[11/10 11:32:01] BEST!
 40%|████      | 24/60 [06:20<08:45, 14.61s/it]INFO:root:
[11/10 11:33:08]
[11/10 11:33:08]
[11/10 11:33:08]
[11/10 11:33:08]
[11/10 11:33:08]
INFO:root:epoch 25/60, iter 700:
[11/10 11:33:08] epoch 25/60, iter 700:
[11/10 11:33:08] epoch 25/60, iter 700:
[11/10 11:33:08] epoch 25/60, iter 700:
[11/10 11:33:08] epoch 25/60, iter 700:
[11/10 11:33:08] epoch 25/60, iter 700:
INFO:root:train_acc: 19.226
[11/10 11:33:08] train_acc: 19.226
[11/10 11:33:08] train_acc: 19.226
[11/10 11:33:08] train_acc: 19.226
[11/10 11:33:08] train_acc: 19.226
[11/10 11:33:08] train_acc: 19.226

0it [00:00, ?it/s]
1it [00:00,  2.41it/s]
4it [00:00,  5.97it/s]
7it [00:01,  7.29it/s]
10it [00:01,  7.99it/s]
13it [00:01,  8.27it/s]
16it [00:02,  8.51it/s]
19it [00:02,  8.08it/s]
22it [00:02,  8.31it/s]
25it [00:03,  8.50it/s]
28it [00:03,  8.29it/s]
31it [00:03,  8.69it/s]
```

```
34it [00:04,  8.67it/s]
37it [00:04,  8.85it/s]
40it [00:04,  9.12it/s]
43it [00:05,  9.01it/s]
46it [00:05,  9.18it/s]
50it [00:05,  8.46it/s]
INFO:root:val_5-way-5-shot_acc: 51.325  2.373
[11/10 11:33:24] val_5-way-5-shot_acc: 51.325   2.373
[11/10 11:33:24] val_5-way-5-shot_acc: 51.325   2.373
[11/10 11:33:24] val_5-way-5-shot_acc: 51.325   2.373
[11/10 11:33:24] val_5-way-5-shot_acc: 51.325   2.373
[11/10 11:33:24] val_5-way-5-shot_acc: 51.325   2.373
INFO:root:BEST!
[11/10 11:33:24] BEST!
[11/10 11:33:24] BEST!
[11/10 11:33:24] BEST!
[11/10 11:33:24] BEST!
[11/10 11:33:24] BEST!
 48%|████▊     | 29/60 [07:42<07:33, 14.64s/it]INFO:root:
[11/10 11:34:30]
[11/10 11:34:30]
[11/10 11:34:30]
[11/10 11:34:30]
[11/10 11:34:30]
INFO:root:epoch 30/60, iter 840:
[11/10 11:34:30] epoch 30/60, iter 840:
[11/10 11:34:30] epoch 30/60, iter 840:
[11/10 11:34:30] epoch 30/60, iter 840:
[11/10 11:34:30] epoch 30/60, iter 840:
[11/10 11:34:30] epoch 30/60, iter 840:
INFO:root:train_acc: 19.893
[11/10 11:34:30] train_acc: 19.893
[11/10 11:34:30] train_acc: 19.893
[11/10 11:34:30] train_acc: 19.893
[11/10 11:34:30] train_acc: 19.893
[11/10 11:34:30] train_acc: 19.893

0it [00:00, ?it/s]
1it [00:00,  2.81it/s]
4it [00:00,  6.68it/s]
7it [00:00,  7.73it/s]
10it [00:01,  8.42it/s]
13it [00:01,  8.20it/s]
16it [00:02,  8.37it/s]
19it [00:02,  8.69it/s]
22it [00:02,  8.88it/s]
25it [00:02,  9.04it/s]
28it [00:03,  9.09it/s]
31it [00:03,  9.30it/s]
34it [00:03,  9.52it/s]
37it [00:04,  9.32it/s]
40it [00:04,  9.63it/s]
43it [00:04,  9.58it/s]
46it [00:05,  9.28it/s]
50it [00:05,  8.94it/s]
INFO:root:val_5-way-5-shot_acc: 51.125  2.652
```

```
[11/10 11:34:46] val_5-way-5-shot_acc: 51.125   2.652
[11/10 11:34:46] val_5-way-5-shot_acc: 51.125   2.652
[11/10 11:34:46] val_5-way-5-shot_acc: 51.125   2.652
[11/10 11:34:46] val_5-way-5-shot_acc: 51.125   2.652
[11/10 11:34:46] val_5-way-5-shot_acc: 51.125   2.652
 57%|[redacted]     | 34/60 [09:04<06:21, 14.67s/it]INFO:root:
[11/10 11:35:52]
[11/10 11:35:52]
[11/10 11:35:52]
[11/10 11:35:52]
[11/10 11:35:52]
INFO:root:epoch 35/60, iter 980:
[11/10 11:35:52] epoch 35/60, iter 980:
[11/10 11:35:52] epoch 35/60, iter 980:
[11/10 11:35:52] epoch 35/60, iter 980:
[11/10 11:35:52] epoch 35/60, iter 980:
[11/10 11:35:52] epoch 35/60, iter 980:
INFO:root:train_acc: 19.512
[11/10 11:35:52] train_acc: 19.512
[11/10 11:35:52] train_acc: 19.512
[11/10 11:35:52] train_acc: 19.512
[11/10 11:35:52] train_acc: 19.512
[11/10 11:35:52] train_acc: 19.512

0it [00:00, ?it/s]
1it [00:00,  2.82it/s]
4it [00:00,  6.67it/s]
7it [00:00,  7.97it/s]
10it [00:01,  8.73it/s]
13it [00:01,  9.08it/s]
15it [00:01, 10.40it/s]
17it [00:02,  8.68it/s]
19it [00:02,  8.87it/s]
20it [00:02,  8.73it/s]
22it [00:02,  9.26it/s]
23it [00:02,  8.85it/s]
25it [00:02,  8.29it/s]
28it [00:03,  8.77it/s]
31it [00:03,  8.86it/s]
34it [00:03,  9.17it/s]
37it [00:04,  9.10it/s]
40it [00:04,  9.12it/s]
43it [00:04,  8.95it/s]
46it [00:05,  9.30it/s]
47it [00:05,  9.11it/s]
49it [00:05,  9.44it/s]
50it [00:05,  8.80it/s]
INFO:root:val_5-way-5-shot_acc: 50.350  2.784
[11/10 11:36:08] val_5-way-5-shot_acc: 50.350   2.784
[11/10 11:36:08] val_5-way-5-shot_acc: 50.350   2.784
[11/10 11:36:08] val_5-way-5-shot_acc: 50.350   2.784
[11/10 11:36:08] val_5-way-5-shot_acc: 50.350   2.784
[11/10 11:36:08] val_5-way-5-shot_acc: 50.350   2.784
 65%|[redacted]    | 39/60 [10:26<05:07, 14.62s/it]INFO:root:
[11/10 11:37:14]
[11/10 11:37:14]
```

```
[11/10 11:37:14]
[11/10 11:37:14]
[11/10 11:37:14]
INFO:root:epoch 40/60, iter 1120:
[11/10 11:37:14] epoch 40/60, iter 1120:
[11/10 11:37:14] epoch 40/60, iter 1120:
[11/10 11:37:14] epoch 40/60, iter 1120:
[11/10 11:37:14] epoch 40/60, iter 1120:
[11/10 11:37:14] epoch 40/60, iter 1120:
INFO:root:train_acc: 20.381
[11/10 11:37:14] train_acc: 20.381
[11/10 11:37:14] train_acc: 20.381
[11/10 11:37:14] train_acc: 20.381
[11/10 11:37:14] train_acc: 20.381
[11/10 11:37:14] train_acc: 20.381

0it [00:00, ?it/s]
1it [00:00,  2.98it/s]
4it [00:00,  6.76it/s]
7it [00:01,  7.35it/s]
10it [00:01,  8.06it/s]
13it [00:01,  8.51it/s]
16it [00:01,  8.97it/s]
18it [00:02, 10.42it/s]
20it [00:02,  8.81it/s]
22it [00:02,  8.56it/s]
25it [00:02,  8.79it/s]
28it [00:03,  9.21it/s]
29it [00:03,  8.71it/s]
31it [00:03,  9.49it/s]
32it [00:03,  8.56it/s]
34it [00:03,  9.82it/s]
36it [00:04,  9.65it/s]
37it [00:04,  9.06it/s]
38it [00:04,  8.41it/s]
40it [00:04,  9.53it/s]
41it [00:04,  8.34it/s]
43it [00:04,  9.56it/s]
44it [00:05,  8.54it/s]
46it [00:05,  9.79it/s]
47it [00:05,  8.82it/s]
49it [00:05,  9.28it/s]
50it [00:05,  8.72it/s]
INFO:root:val_5-way-5-shot_acc: 55.200  2.672
[11/10 11:37:30] val_5-way-5-shot_acc: 55.200   2.672
[11/10 11:37:30] val_5-way-5-shot_acc: 55.200   2.672
[11/10 11:37:30] val_5-way-5-shot_acc: 55.200   2.672
[11/10 11:37:30] val_5-way-5-shot_acc: 55.200   2.672
[11/10 11:37:30] val_5-way-5-shot_acc: 55.200   2.672
INFO:root:BEST!
[11/10 11:37:30] BEST!
[11/10 11:37:30] BEST!
[11/10 11:37:30] BEST!
[11/10 11:37:30] BEST!
[11/10 11:37:30] BEST!
 73%|███████   | 44/60 [11:49<03:55, 14.70s/it]INFO:root:
```

```
[11/10 11:38:37]
[11/10 11:38:37]
[11/10 11:38:37]
[11/10 11:38:37]
[11/10 11:38:37]
INFO:root:epoch 45/60, iter 1260:
[11/10 11:38:37] epoch 45/60, iter 1260:
[11/10 11:38:37] epoch 45/60, iter 1260:
[11/10 11:38:37] epoch 45/60, iter 1260:
[11/10 11:38:37] epoch 45/60, iter 1260:
[11/10 11:38:37] epoch 45/60, iter 1260:
INFO:root:train_acc: 20.167
[11/10 11:38:37] train_acc: 20.167
[11/10 11:38:37] train_acc: 20.167
[11/10 11:38:37] train_acc: 20.167
[11/10 11:38:37] train_acc: 20.167
[11/10 11:38:37] train_acc: 20.167

0it [00:00, ?it/s]
1it [00:00,  2.68it/s]
4it [00:00,  6.49it/s]
7it [00:00,  7.83it/s]
10it [00:01,  7.84it/s]
13it [00:01,  8.35it/s]
16it [00:02,  8.80it/s]
19it [00:02,  8.88it/s]
22it [00:02,  9.27it/s]
25it [00:02,  9.31it/s]
28it [00:03,  9.39it/s]
31it [00:03,  9.23it/s]
34it [00:03,  9.27it/s]
37it [00:04,  9.21it/s]
40it [00:04,  9.42it/s]
43it [00:04,  9.43it/s]
46it [00:05,  9.57it/s]
50it [00:05,  9.03it/s]
INFO:root:val_5-way-5-shot_acc: 51.225  2.835
[11/10 11:38:52] val_5-way-5-shot_acc: 51.225   2.835
[11/10 11:38:52] val_5-way-5-shot_acc: 51.225   2.835
[11/10 11:38:52] val_5-way-5-shot_acc: 51.225   2.835
[11/10 11:38:52] val_5-way-5-shot_acc: 51.225   2.835
[11/10 11:38:52] val_5-way-5-shot_acc: 51.225   2.835
 82%|████████   | 49/60 [13:11<02:40, 14.60s/it]INFO:root:
[11/10 11:39:58]
[11/10 11:39:58]
[11/10 11:39:58]
[11/10 11:39:58]
[11/10 11:39:58]
INFO:root:epoch 50/60, iter 1400:
[11/10 11:39:58] epoch 50/60, iter 1400:
[11/10 11:39:58] epoch 50/60, iter 1400:
[11/10 11:39:58] epoch 50/60, iter 1400:
[11/10 11:39:58] epoch 50/60, iter 1400:
[11/10 11:39:58] epoch 50/60, iter 1400:
INFO:root:train_acc: 20.536
[11/10 11:39:58] train_acc: 20.536
```

```
[11/10 11:39:58] train_acc: 20.536
[11/10 11:39:58] train_acc: 20.536
[11/10 11:39:58] train_acc: 20.536
[11/10 11:39:58] train_acc: 20.536

0it [00:00, ?it/s]
1it [00:00,  2.91it/s]
4it [00:00,  6.62it/s]
7it [00:00,  8.03it/s]
8it [00:01,  8.11it/s]
10it [00:01,  8.61it/s]
11it [00:01,  8.12it/s]
13it [00:01,  9.12it/s]
14it [00:01,  8.70it/s]
16it [00:01,  8.48it/s]
17it [00:02,  8.59it/s]
19it [00:02,  8.64it/s]
20it [00:02,  8.61it/s]
22it [00:02,  9.29it/s]
23it [00:02,  7.93it/s]
25it [00:02,  9.76it/s]
27it [00:03,  9.96it/s]
29it [00:03,  8.46it/s]
31it [00:03,  8.36it/s]
33it [00:03, 10.18it/s]
35it [00:04,  8.47it/s]
37it [00:04,  8.15it/s]
40it [00:04,  8.57it/s]
41it [00:04,  8.20it/s]
43it [00:05,  8.75it/s]
45it [00:05, 10.34it/s]
47it [00:05,  9.24it/s]
50it [00:05,  8.64it/s]
INFO:root:val_5-way-5-shot_acc: 51.975  2.824
[11/10 11:40:14] val_5-way-5-shot_acc: 51.975   2.824
[11/10 11:40:14] val_5-way-5-shot_acc: 51.975   2.824
[11/10 11:40:14] val_5-way-5-shot_acc: 51.975   2.824
[11/10 11:40:14] val_5-way-5-shot_acc: 51.975   2.824
[11/10 11:40:14] val_5-way-5-shot_acc: 51.975   2.824
 90%|████████▉ | 54/60 [14:33<01:28, 14.69s/it]INFO:root:
[11/10 11:41:21]
[11/10 11:41:21]
[11/10 11:41:21]
[11/10 11:41:21]
[11/10 11:41:21]
INFO:root:epoch 55/60, iter 1540:
[11/10 11:41:21] epoch 55/60, iter 1540:
[11/10 11:41:21] epoch 55/60, iter 1540:
[11/10 11:41:21] epoch 55/60, iter 1540:
[11/10 11:41:21] epoch 55/60, iter 1540:
[11/10 11:41:21] epoch 55/60, iter 1540:
INFO:root:train_acc: 20.976
[11/10 11:41:21] train_acc: 20.976
[11/10 11:41:21] train_acc: 20.976
[11/10 11:41:21] train_acc: 20.976
[11/10 11:41:21] train_acc: 20.976
```

```
[11/10 11:41:21] train_acc: 20.976

0it [00:00, ?it/s]
1it [00:00,  2.84it/s]
4it [00:00,  6.71it/s]
7it [00:01,  7.38it/s]
10it [00:01,  8.00it/s]
13it [00:01,  8.41it/s]
16it [00:02,  8.80it/s]
19it [00:02,  9.05it/s]
22it [00:02,  8.78it/s]
25it [00:03,  8.79it/s]
28it [00:03,  9.07it/s]
31it [00:03,  9.36it/s]
34it [00:03,  9.67it/s]
37it [00:04,  9.75it/s]
40it [00:04,  9.62it/s]
43it [00:04,  9.55it/s]
46it [00:05,  9.09it/s]
50it [00:05,  8.99it/s]
INFO:root:val_5-way-5-shot_acc: 48.675  2.936
[11/10 11:41:36] val_5-way-5-shot_acc: 48.675   2.936
[11/10 11:41:36] val_5-way-5-shot_acc: 48.675   2.936
[11/10 11:41:36] val_5-way-5-shot_acc: 48.675   2.936
[11/10 11:41:36] val_5-way-5-shot_acc: 48.675   2.936
[11/10 11:41:36] val_5-way-5-shot_acc: 48.675   2.936
 98%|▉▉▉▉▉▉▉▉▉ | 59/60 [15:55<00:14, 14.65s/it]INFO:root:
[11/10 11:42:43]
[11/10 11:42:43]
[11/10 11:42:43]
[11/10 11:42:43]
[11/10 11:42:43]
INFO:root:epoch 60/60, iter 1680:
[11/10 11:42:43] epoch 60/60, iter 1680:
[11/10 11:42:43] epoch 60/60, iter 1680:
[11/10 11:42:43] epoch 60/60, iter 1680:
[11/10 11:42:43] epoch 60/60, iter 1680:
[11/10 11:42:43] epoch 60/60, iter 1680:
INFO:root:train_acc: 20.345
[11/10 11:42:43] train_acc: 20.345
[11/10 11:42:43] train_acc: 20.345
[11/10 11:42:43] train_acc: 20.345
[11/10 11:42:43] train_acc: 20.345
[11/10 11:42:43] train_acc: 20.345

0it [00:00, ?it/s]
1it [00:00,  2.74it/s]
4it [00:00,  6.47it/s]
7it [00:01,  7.55it/s]
10it [00:01,  8.09it/s]
13it [00:01,  8.56it/s]
15it [00:01, 10.10it/s]
17it [00:02,  8.80it/s]
19it [00:02,  8.13it/s]
22it [00:02,  8.64it/s]
25it [00:03,  8.90it/s]
```

```
27it [00:03, 10.04it/s]
29it [00:03,  9.93it/s]
31it [00:03,  8.84it/s]
33it [00:03, 10.12it/s]
35it [00:03,  9.52it/s]
37it [00:04,  8.29it/s]
40it [00:04,  8.28it/s]
43it [00:04,  8.71it/s]
46it [00:05,  9.05it/s]
48it [00:05, 10.17it/s]
50it [00:05,  8.85it/s]
INFO:root:val_5-way-5-shot_acc: 49.875  2.637
[11/10 11:42:58] val_5-way-5-shot_acc: 49.875   2.637
[11/10 11:42:58] val_5-way-5-shot_acc: 49.875   2.637
[11/10 11:42:58] val_5-way-5-shot_acc: 49.875   2.637
[11/10 11:42:58] val_5-way-5-shot_acc: 49.875   2.637
[11/10 11:42:58] val_5-way-5-shot_acc: 49.875   2.637
100%|██████████| 60/60 [16:24<00:00, 16.40s/it]
INFO:root:training finished!
[11/10 11:42:58] training finished!
[11/10 11:42:58] training finished!
[11/10 11:42:58] training finished!
[11/10 11:42:58] training finished!
[11/10 11:42:58] training finished!
INFO:root:------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
INFO:root:the best epoch is 40/60
[11/10 11:42:58] the best epoch is 40/60
[11/10 11:42:58] the best epoch is 40/60
[11/10 11:42:58] the best epoch is 40/60
[11/10 11:42:58] the best epoch is 40/60
[11/10 11:42:58] the best epoch is 40/60
INFO:root:the best 5-way 5-shot val acc is 55.200
[11/10 11:42:58] the best 5-way 5-shot val acc is 55.200
[11/10 11:42:58] the best 5-way 5-shot val acc is 55.200
[11/10 11:42:58] the best 5-way 5-shot val acc is 55.200
[11/10 11:42:58] the best 5-way 5-shot val acc is 55.200
[11/10 11:42:58] the best 5-way 5-shot val acc is 55.200
INFO:root:------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
[11/10 11:42:58] ------------------------
INFO:root:evaluating on test set:
[11/10 11:42:58] evaluating on test set:
[11/10 11:42:58] evaluating on test set:
[11/10 11:42:58] evaluating on test set:
[11/10 11:42:58] evaluating on test set:
[11/10 11:42:58] evaluating on test set:
<ipython-input-16-46a1110d7e3f>:275: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the default pickle modu
```

```
le implicitly. It is possible to construct malicious pickle data which will execute
arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/S
ECURITY.md#untrusted-models for more details). In a future release, the default valu
e for `weights_only` will be flipped to `True`. This limits the functions that could
be executed during unpickling. Arbitrary objects will no longer be allowed to be loa
ded via this mode unless they are explicitly allowlisted by the user via `torch.seri
alization.add_safe_globals`. We recommend you start setting `weights_only=True` for
any use case where you don't have full control of the loaded file. Please open an is
sue on GitHub for any issues related to this experimental feature.
  model.load_state_dict(torch.load(trainer_save_path))
*******Starting Evaluate********
```

```
50it [00:05,  9.24it/s]
INFO:root:5-way-1-shot acc: 52.73       2.47
[11/10 11:43:18] 5-way-1-shot acc: 52.73        2.47
[11/10 11:43:18] 5-way-1-shot acc: 52.73        2.47
[11/10 11:43:18] 5-way-1-shot acc: 52.73        2.47
[11/10 11:43:18] 5-way-1-shot acc: 52.73        2.47
[11/10 11:43:18] 5-way-1-shot acc: 52.73        2.47
50it [00:05,  8.99it/s]
INFO:root:5-way-5-shot acc: 51.45       2.53
[11/10 11:43:39] 5-way-5-shot acc: 51.45        2.53
[11/10 11:43:39] 5-way-5-shot acc: 51.45        2.53
[11/10 11:43:39] 5-way-5-shot acc: 51.45        2.53
[11/10 11:43:39] 5-way-5-shot acc: 51.45        2.53
[11/10 11:43:39] 5-way-5-shot acc: 51.45        2.53
```

In [15]:

```
import sys
import os
import torch
import yaml

#**************************************************************************
# DEVICE MAP FUNCTION
# Gets the device map location for the GPU
#**************************************************************************
def get_device_map(gpu):
    cuda = lambda x: 'cuda:%d'%x
    temp = {}
    for i in range(4):
        temp[cuda(i)]=cuda(gpu)
    return temp
#**************************************************************************

# Setup the file paths
data_path = os.path.abspath('/content/')
test_path = "/content/dogs/test_pre"
model_path = '/content/model_Conv-4-sgd-lr_1e-01-gamma_2e-02-epoch_5-stage_3-decay_

gpu = 0
torch.cuda.set_device(gpu)
model = BiFRN(resnet=False)
model.cuda()
model.load_state_dict(torch.load(model_path,map_location=get_device_map(gpu)),stric
model.eval()
```

```
with torch.no_grad():
    for way in [30, 25, 20, 15, 10, 5]:
        for shot in [30, 25, 20, 15, 10, 5, 1]:
            mean,interval = meta_test(data_path=test_path,
                                      model=model,
                                      way=way,
                                      shot=shot,
                                      pre=True,
                                      transform_type=None,
                                      trial=5)
            print('%d-way-%d-shot acc: %.3f\t%.3f'%(way,shot,mean,interval))
```

```
<ipython-input-15-fa9fc00e7c13>:27: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  model.load_state_dict(torch.load(model_path,map_location=get_device_map(gpu)),strict=True)
5it [00:03,  1.33it/s]
30-way-30-shot acc: 11.875      0.924
5it [00:03,  1.49it/s]
30-way-25-shot acc: 11.750      1.482
5it [00:02,  1.72it/s]
30-way-20-shot acc: 12.417      1.117
5it [00:02,  2.00it/s]
30-way-15-shot acc: 10.917      0.884
5it [00:02,  2.45it/s]
30-way-10-shot acc: 10.167      0.706
5it [00:01,  3.06it/s]
30-way-5-shot acc: 9.667        1.147
5it [00:01,  3.88it/s]
30-way-1-shot acc: 5.375        1.116
5it [00:03,  1.62it/s]
25-way-30-shot acc: 14.000      1.028
5it [00:02,  1.81it/s]
25-way-25-shot acc: 13.150      0.861
5it [00:02,  2.08it/s]
25-way-20-shot acc: 14.150      0.926
5it [00:02,  2.39it/s]
25-way-15-shot acc: 13.500      1.434
5it [00:01,  2.95it/s]
25-way-10-shot acc: 10.900      1.260
5it [00:01,  3.68it/s]
25-way-5-shot acc: 11.700       0.594
5it [00:01,  4.59it/s]
```

```
25-way-1-shot acc: 7.750        1.775
5it [00:02,  2.05it/s]
20-way-30-shot acc: 16.688      1.822
5it [00:02,  2.31it/s]
20-way-25-shot acc: 16.812      2.962
5it [00:01,  2.64it/s]
20-way-20-shot acc: 15.938      1.960
5it [00:01,  3.07it/s]
20-way-15-shot acc: 15.062      2.132
5it [00:01,  3.54it/s]
20-way-10-shot acc: 13.875      1.157
5it [00:01,  4.54it/s]
20-way-5-shot acc: 13.375       0.319
5it [00:00,  5.74it/s]
20-way-1-shot acc: 11.438       1.183
5it [00:01,  2.78it/s]
15-way-30-shot acc: 22.333      2.137
5it [00:01,  3.04it/s]
15-way-25-shot acc: 21.417      2.609
5it [00:01,  3.51it/s]
15-way-20-shot acc: 21.583      2.556
5it [00:01,  4.02it/s]
15-way-15-shot acc: 19.417      2.137
5it [00:01,  4.98it/s]
15-way-10-shot acc: 19.333      1.543
5it [00:00,  6.11it/s]
15-way-5-shot acc: 15.333       1.093
5it [00:00,  7.54it/s]
15-way-1-shot acc: 11.750       1.952
5it [00:01,  4.09it/s]
10-way-30-shot acc: 25.625      3.341
5it [00:01,  4.60it/s]
10-way-25-shot acc: 26.625      2.607
5it [00:00,  5.24it/s]
10-way-20-shot acc: 28.000      3.941
5it [00:00,  6.29it/s]
10-way-15-shot acc: 29.875      2.508
5it [00:00,  7.30it/s]
10-way-10-shot acc: 25.000      4.085
5it [00:00,  8.99it/s]
10-way-5-shot acc: 24.250       3.657
5it [00:00, 10.89it/s]
10-way-1-shot acc: 18.750       3.305
5it [00:00,  8.28it/s]
5-way-30-shot acc: 40.500       7.949
5it [00:00,  9.33it/s]
5-way-25-shot acc: 50.000       4.597
5it [00:00, 10.52it/s]
5-way-20-shot acc: 45.250       3.492
```

```
5it [00:00, 12.12it/s]
5-way-15-shot acc: 43.750       8.287
5it [00:00, 13.81it/s]
5-way-10-shot acc: 34.750       4.669
5it [00:00, 16.17it/s]
5-way-5-shot acc: 37.000        4.679
5it [00:00, 20.96it/s]
5-way-1-shot acc: 31.000        7.046
```