In [ ]:
```
%load_ext autoreload
%autoreload 2
```

In [ ]:
```
import sys
sys.path.append("../model")
sys.path.append("../tools")
from constants import *
from Tip_adapter import TipAdapter
from dataset import FlickrAndPairs, TestMeta, TaskAWithLabel
import torch
from torch.utils.data import DataLoader, random_split
import torch.nn as nn
import matplotlib.pyplot as plt
```

```
Loading pipeline components...: 100%|██████████| 7/7 [00:01<00:00,  6.29it/s]
```

In [ ]:
```
data = FlickrAndPairs(path="/data4/saland/data/real_fake_pairs_1000_name.pt",load_from_disk=True)
taskA = TaskAWithLabel(path_to_csv="../../misc/scanFinal.csv",
                       path_to_taskA="/data4/saland/data/taskA.pt")
# data = TestMeta("/data4/saland/data/test_meta.pt",load_from_disk=True)
```

```
100%|██████████| 10080/10080 [00:04<00:00, 2387.48it/s]
```

Tip-Adapter

In [ ]:
```
tip = TipAdapter(100, data)
test_meta = TestMeta(path="/data4/saland/data/test_meta.pt",load_from_disk=True)
tip.get_accuracy(test_meta.features,test_meta.label,"cpu")
```

Out[ ]:
```
0.7576740980148315
```

In [ ]:
```
tip.get_accuracy(taskA.features,taskA.label,"cpu")
```

```
100%|██████████| 10080/10080 [00:04<00:00, 2340.46it/s]
```
Out[ ]:
```
0.6840277910232544
```

In [ ]:
```
for k in (1,2,3,4,8,16,32,64,100,500,1000):
    tip = TipAdapter(k,data)
    print(f"accuracy on taskA    with cache of size {len(tip)}:",tip.get_accuracy(taskA.features,taskA.label,"cpu"))
    print(f"accuracy on testMeta with cache of size {len(tip)}:",tip.get_accuracy(test_meta.features,test_meta.label,"cpu"))
```

```
accuracy on taskA    with cache of size 2: 0.6822420358657837
accuracy on testMeta with cache of size 2: 0.6530447006225586
accuracy on taskA    with cache of size 4: 0.692460298538208
accuracy on testMeta with cache of size 4: 0.7086349129676819
accuracy on taskA    with cache of size 6: 0.7045634984970093
accuracy on testMeta with cache of size 6: 0.7354629635810852
accuracy on taskA    with cache of size 8: 0.7547619342803955
accuracy on testMeta with cache of size 8: 0.7449463605880737
accuracy on taskA    with cache of size 16: 0.7176587581634521
accuracy on testMeta with cache of size 16: 0.7693411707878113
accuracy on taskA    with cache of size 32: 0.6936507821083069
accuracy on testMeta with cache of size 32: 0.7455078363418579
accuracy on taskA    with cache of size 64: 0.6016865372657776
accuracy on testMeta with cache of size 64: 0.7120040059089661
accuracy on taskA    with cache of size 128: 0.6962301731109619
accuracy on testMeta with cache of size 128: 0.7583603858947754
accuracy on taskA    with cache of size 200: 0.6840277910232544
accuracy on testMeta with cache of size 200: 0.7576740980148315
accuracy on taskA    with cache of size 1000: 0.6862103343009949
accuracy on testMeta with cache of size 1000: 0.751185417175293
accuracy on taskA    with cache of size 2000: 0.6839285492897034
accuracy on testMeta with cache of size 2000: 0.7528699636459351
```

In [ ]:
```
for alpha in range(6):
    tip = TipAdapter(4,data,alpha=alpha)
    print(f"accuracy on taskA    with alpha={alpha}:",tip.get_accuracy(taskA.features,taskA.label,"cpu"))
    print(f"accuracy on testMeta with alpha={alpha}:",tip.get_accuracy(test_meta.features,test_meta.label,"cpu"))
```

```
accuracy on taskA    with alpha=0: 0.6501984000205994
accuracy on testMeta with alpha=0: 0.4730471670627594
accuracy on taskA    with alpha=1: 0.7364087104797363
accuracy on testMeta with alpha=1: 0.6531070470809937
accuracy on taskA    with alpha=2: 0.7532737851142883
accuracy on testMeta with alpha=2: 0.7110681533813477
accuracy on taskA    with alpha=3: 0.7559523582458496
accuracy on testMeta with alpha=3: 0.7311579585075378
accuracy on taskA    with alpha=4: 0.7545635104179382
accuracy on testMeta with alpha=4: 0.7400798797607422
accuracy on taskA    with alpha=5: 0.7547619342803955
accuracy on testMeta with alpha=5: 0.7449463605880737
```

Tip-Adapter-F

In [ ]:
```
device = "cuda:0"
tip_F = TipAdapter(50, data,device=device)
tip = TipAdapter(50, data,device=device)
```

In [ ]:
```
ft_data = TestMeta(path="/data4/saland/data/test_meta.pt",load_from_disk=True)
```

In [ ]:
```
tip_F.F_train
```

Out[ ]:
```
Parameter containing:
tensor([[-0.0136, -0.0105,  0.0072,  ...,  0.0377, -0.0176, -0.0032],
        [ 0.0132,  0.0452,  0.0132,  ..., -0.0054,  0.0031,  0.0111],
        [ 0.0295,  0.0077,  0.0422,  ...,  0.0049, -0.0926,  0.0768],
        ...,
        [ 0.0143,  0.0288,  0.0248,  ..., -0.0174, -0.0237,  0.0027],
        [-0.0065, -0.0263,  0.0403,  ..., -0.0178, -0.0178, -0.0255],
        [-0.0197, -0.0379, -0.0126,  ..., -0.0319, -0.0008,  0.0046]],
       device='cuda:0', requires_grad=True)
```

In [ ]:
```
tip_F.train()
tip_F.F_train.requires_grad = True

lr = 1e-3
batch_size = 64
# n_epochs = 200

loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(tip_F.parameters(), lr=lr)

rng = torch.Generator().manual_seed(SEED)
train_data, test_data, validation_data = random_split(ft_data,[0.7,0.2,0.1],generator=rng)

train_loader = DataLoader(train_data,batch_size=batch_size,shuffle=True)
test_loader  = DataLoader(test_data,batch_size=len(test_data),shuffle=True)
val_loader  = DataLoader(validation_data,batch_size=len(validation_data),shuffle=True)
```

In [ ]:

```
val_accuracy = []
for n_epochs in (5,10,20,50,100):
    loss_history = []
    print("fine-tune on n_epochs =",n_epochs)
    tip_F = TipAdapter(50, data,device=device)
    tip_F.train()
    tip_F.F_train.requires_grad = True

    lr = 1e-3
    batch_size = 64
    # n_epochs = 200

    loss_fn = nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(tip_F.parameters(), lr=lr)

    rng = torch.Generator().manual_seed(SEED)
    for epoch in range(1,n_epochs+1):
        for idx, batch in enumerate(train_loader):
            # prediction and loss
            pred = tip_F(batch["features"].to(device))
            loss = loss_fn(pred,batch["label"].type(torch.LongTensor).to(device))

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

        loss, current = loss.item(), idx*batch_size + len(batch["features"])
        # if epoch%10 == 0 and epoch > 0:
        #     loss_history.append(loss)
        #     print(f"loss: {loss:>7f}  [{epoch:>5d}/{n_epochs:>5d}]")
    print("tip-adapter-F accuracy on taskA:",tip_F.get_accuracy(taskA.features,taskA.label,device=device))
print()
print("tip-adapter accuracy on taskA:  ",tip.get_accuracy(taskA.features,taskA.label,device=device))
```

```
fine-tune on n_epochs = 5
tip-adapter-F accuracy on taskA: 0.8500000238418579
fine-tune on n_epochs = 10
tip-adapter-F accuracy on taskA: 0.8629960417747498
fine-tune on n_epochs = 20
tip-adapter-F accuracy on taskA: 0.8804563879966736
fine-tune on n_epochs = 50
tip-adapter-F accuracy on taskA: 0.8946428894996643
fine-tune on n_epochs = 100
tip-adapter-F accuracy on taskA: 0.9000000357627869

tip-adapter accuracy on taskA:   0.6868055462837219
```