1. Baseline

Libraries

In [ ]:
```
import os
from pathlib import Path
from tqdm import tqdm
from easydict import EasyDict as edict

import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.utils as vutils

from data.ms1m import get_train_loader
from data.lfw import LFW

from backbone.arcfacenet import SEResNet_IR
from margin.ArcMarginProduct import ArcMarginProduct

from util.utils import save_checkpoint, test
```

Configuration

In [ ]:
```
config = edict()
config.train_root = '/Users/egebilge/Developer/Computer Vision/Face Recognition With ArcFace/ArcFace/dataset/MS1M'
config.lfw_root = '/Users/egebilge/Developer/Computer Vision/Face Recognition With ArcFace/ArcFace/dataset/lfw_aligned_112'
config.lfw_file_list = '/Users/egebilge/Developer/Computer Vision/Face Recognition With ArcFace/ArcFace/dataset/lfw_pair.txt'

config.mode = 'se_ir'
config.depth = 50
config.margin_type = 'ArcFace'
config.feature_dim = 512
config.batch_size = 32
config.lr = 0.01
config.milestones = [5, 8, 10]
config.total_epoch = 12

config.save_path = './saved'
config.save_dir = os.path.join(config.save_path, f'{config.mode}_{config.depth}_{config.margin_type}_{config.feature_dim}') # save model like: se_ir_50_ArcFace_512

# Check if MPS is available
mps_available = torch.backends.mps.is_available()

# Print MPS availability
print("Is MPS available?", mps_available)

# Check if the current version of PyTorch was built with MPS activated
print("Was the current version of PyTorch built with MPS activated?", torch.backends.mps.is_built())

# Set device based on MPS availability
config.device = torch.device("mps" if mps_available else "cpu")
print("Configured device:", config.device)

config.num_workers = 2
config.pin_memory = True
```

```
Is MPS available? True
Was the current version of PyTorch built with MPS activated? True
Configured device: mps
```

In [ ]:
```
os.makedirs(config.save_dir, exist_ok=True)
```

Data Loader

In [ ]:
```
transforms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])

train_loader,class_num = get_train_loader(config)
```

In [ ]:
```
import torch.utils
import torch.utils.data

lfw_dataset = LFW(config.lfw_root, config.lfw_file_list, transforms)
lfw_loader = torch.utils.data.DataLoader(lfw_dataset, batch_size=32, shuffle=False, num_workers=config.num_workers)
```

Model

In [ ]:
```
model = SEResNet_IR(config.depth, config.feature_dim,mode=config.mode).to(config.device)
margin = ArcMarginProduct(config.feature_dim, class_num).to(config.device)
```

In [ ]:
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD([{'params': model.parameters()},
                       {'params': margin.parameters()}],
                        lr=config.lr, weight_decay=5e-4, momentum=0.9,nesterov=True)

def schedule_lr():
    for params in optimizer.param_groups:
        params['lr'] /= 10

    #print(optimizer)
```

Train

In [ ]:
```
best_acc = 0.0
for epoch in range(config.total_epoch):

    #train
    model.train()
    print(f'Epoch {epoch}/{config.total_epoch}')
    if epoch in config.milestones:
        schedule_lr() # 5, 8, 10 epochs
    for data in tqdm(train_loader):
        img,label = data
        img = img.to(config.device)
        label = label.to(config.device)
        optimizer.zero_grad()
        feature = model(img)
        output = margin(feature,label)
        loss = criterion(output,label)
        loss.backward()
        optimizer.step()

    #test
    model.eval()
    lfw_acc = test(config,model,lfw_dataset,lfw_loader)
    print(f'lfw_acc: {lfw_acc} loss: {loss.item()}')
    is_best = lfw_acc > best_acc
    best_acc = max(best_acc,lfw_acc)
    save_checkpoint({
        'epoch': epoch,
        'model_state_dict': model.state_dict(),
        'margin_state_dict': margin.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'best_acc': best_acc
    }, is_best, checkpoint=config.save_dir)
```

```
Epoch 0/12
100%|██████████| 911/911 [09:42<00:00,  1.56it/s]
lfw_acc: 0.7231666666666666 loss: 12.241194725036621
best model saved

Epoch 1/12
100%|██████████| 911/911 [09:41<00:00,  1.57it/s]
lfw_acc: 0.7918333333333333 loss: 9.217211723327637
best model saved

Epoch 2/12
100%|██████████| 911/911 [09:48<00:00,  1.55it/s]
lfw_acc: 0.8141666666666667 loss: 7.717184543609619
best model saved

Epoch 3/12
100%|██████████| 911/911 [09:40<00:00,  1.57it/s]
lfw_acc: 0.8184999999999999 loss: 7.924939155578613
best model saved

Epoch 4/12
100%|██████████| 911/911 [09:38<00:00,  1.57it/s]
lfw_acc: 0.8261666666666667 loss: 4.220844745635986
best model saved

Epoch 5/12
100%|██████████| 911/911 [09:36<00:00,  1.58it/s]
lfw_acc: 0.8460000000000001 loss: 4.6469268798828125
best model saved

Epoch 6/12
100%|██████████| 911/911 [09:34<00:00,  1.59it/s]
lfw_acc: 0.8486666666666667 loss: 3.2952840328216553
best model saved

Epoch 7/12
100%|██████████| 911/911 [09:34<00:00,  1.58it/s]
lfw_acc: 0.8525 loss: 2.1109778881073
best model saved

Epoch 8/12
100%|██████████| 911/911 [09:34<00:00,  1.59it/s]
lfw_acc: 0.8486666666666667 loss: 2.3444104194641113
Epoch 9/12
100%|██████████| 911/911 [09:34<00:00,  1.59it/s]
lfw_acc: 0.8469999999999999 loss: 3.403931140899658
Epoch 10/12
100%|██████████| 911/911 [09:42<00:00,  1.56it/s]
lfw_acc: 0.8496666666666666 loss: 2.252744197845459
Epoch 11/12
100%|██████████| 911/911 [09:44<00:00,  1.56it/s]
lfw_acc: 0.8465 loss: 1.5389292240142822
```

In [ ]: