深度学习 作业二

- 姓名：吕奇澹
- 学号：SA23006171

实验要求

使用 pytorch 或者 tensorflow 实现卷积神经网络，在 ImageNet 数据集上进行图片分类。研究 dropout, normalization, learning rate decay, residual connection, network depth等超参数对分类性能的影响。

实验过程

数据集处理

Python

```python
class TinyImagenet(Dataset):
    dataRecord: Dict = dict()
    trainRadio: float = -1
    validRadio: float = -1

    def __init__(
        self,
        trainRadio: float = 0.8,
        validRadio: float = 0.2,
        type: Literal["train", "valid", "test"] = "train",
    ) -> None:
        super().__init__()
        if type not in {"train", "valid", "test"}:
            raise ValueError("'%s' is not a supported type" % type)

        assert abs(trainRadio + validRadio - 1) < 1e-5
        _trainRadio = TinyImagenet.trainRadio
        _validRadio = TinyImagenet.validRadio
        if _trainRadio != -1 and abs(trainRadio - _trainRadio) > 1e-5:
            raise ValueError(
                f"There is already a dataset with trainRadio {_trainRadio
}"
            )
        if _validRadio != -1 and abs(validRadio - _validRadio) > 1e-5:
            raise ValueError(
                f"There is already a dataset with validRadio {_validRadio
}"
            )

        self.dataDir: str = None
        self.labels: List[str] = None
        self.label2id: Dict[str, int] = None
        self.trainData: List[Tuple[torch.Tensor, int]] = []
        self.validData: List[Tuple[torch.Tensor, int]] = []
        self.transform = transforms.ToTensor()
        TinyImagenet.trainRadio = trainRadio
        TinyImagenet.validRadio = validRadio

        self.getData()
        self.getLabels()
        if type == "test":
            self.dealValid()
            self.data = self.validData
        else:
            if not TinyImagenet.dataRecord:
```

```
                self.dealTrain()
                stateOri = random.getstate()
                random.seed(0)
                random.shuffle(self.trainData)
                random.setstate(stateOri)
                N = len(self.trainData)
                trainNum = int(N * trainRadio)
                TinyImagenet.dataRecord["train"] = self.trainData[:trainNum]
                TinyImagenet.dataRecord["valid"] = self.trainData[trainNum:]
            self.data = TinyImagenet.dataRecord[type]
        print(f"finish {type} dataset")

    def getData(self):
        """获取数据集文件"""
        self.root = os.path.join("..", "data")
        self.dataDir = os.path.join(self.root, "tiny-imagenet-200")
        if not os.path.exists(self.dataDir):
            os.makedirs(self.root,exist_ok=True)
            dataFile = os.path.join(self.root, "tiny-imagenet-200.zip")
            if not os.path.exists(dataFile):
                url = "http://cs231n.stanford.edu/tiny-imagenet-200.zip"
                print(f"download data from {url}")
                content = requests.get(url, allow_redirects=True).content
                with open(dataFile, "wb") as f:
                    f.write(content)
            print("extract zip file")
            with zipfile.ZipFile(dataFile) as f:
                f.extractall(self.root)

    def getLabels(self):
        """获取数据集的所有 label"""
        filename = os.path.join(self.dataDir, "wnids.txt")
        assert os.path.exists(filename)

        with open(filename, "r", encoding="utf8") as f:
            self.labels = [label.strip() for label in f.readlines()]
            self.label2id = {label: index for index, label in enumerate(self.labels)}

    def dealTrain(self):
        """处理 train 数据集"""
        self.trainData = []
        for label in self.labels:
            statFile = os.path.join(self.dataDir, "train", label, f"{label}_boxes.txt")
            with open(statFile, "r", encoding="utf8") as f:
```

```
                for line in f:
                    line = line.strip().split()
                    picName = line[0]
                    picFile = os.path.join(
                        self.dataDir, "train", label, "images", picName
                    )
                    self.trainData.append((picFile, self.label2id[label])
)
    def dealValid(self):
        """处理 valid 数据集"""
        statFile = os.path.join(self.dataDir, "val", "val_annotations.tx
t")
        self.validData = []
        with open(statFile, "r", encoding="utf8") as f:
            for line in f:
                line = line.strip().split()
                picName = line[0]
                label = line[1]
                picFile = os.path.join(self.dataDir, "val", "images", pic
Name)
                self.validData.append((picFile, self.label2id[label]))

    def __getitem__(self, index) -> Tuple[torch.Tensor, int]:
        picFile, label = self.data[index]
        img = cv2.imread(picFile).reshape(64, 64, 3)
        img = self.transform(img)
        return (img, label)

    def __len__(self) -> int:
        return len(self.data)
```

模型搭建

我们使用了自己搭建的模型和直接调用torchvision中的resnet模型进行实验

Python

```python
class MyModule(nn.Module):
    def __init__(
        self,
        depth: int,
        dropout: bool = False,
        normalize: bool = False,
        has_residual: bool = False,
    ):
        super().__init__()
        in_channel = 3
        self.channel = 64
        self.expansion = 2
        """
        64*64*3
        32*32*64
        16*16*64
        """
        self.model = nn.Sequential(
            nn.Conv2d(in_channel, self.channel, kernel_size=7, stride=2, padding=3),
            nn.BatchNorm2d(self.channel) if normalize else nn.Identity(),
            nn.ReLU(),
            nn.AvgPool2d(kernel_size=3, stride=2, padding=1),
        )
        for _ in range(depth):
            self.model.append(
                ResidualBlock(self.channel, dropout, normalize, has_residual)
            )
            self.model.append(
                nn.Conv2d(self.channel, self.channel * self.expansion, kernel_size=1)
            )
            self.model.append(nn.AvgPool2d(kernel_size=2, stride=2, padding=1))
            self.channel *= self.expansion
        self.model.append(nn.AdaptiveAvgPool2d((1, 1)))
        self.model.append(nn.Flatten())
        self.model.append(nn.Linear(self.channel, 200))
        self.model.append(nn.Softmax(dim=1))

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        return self.model(x)
```

模型训练

我们定义train_once函数进行训练

Python

```python
def trainOnce(
    depth: int, dropout: bool, normalize: bool, has_residual: bool, lrDecay: float
):
    epochs = 10
    model = MyModule(depth, dropout, normalize, has_residual)
    model = resnet50()
    optimizer = optim.Adam(model.parameters(), lr=1e-3)
    criterion = nn.CrossEntropyLoss()
    if lrDecay:
        scheduler = LambdaLR(optimizer, lr_lambda=lambda epoch: 0.95**epoch)
    trainData = TinyImagenet(type="train")
    validData = TinyImagenet(type="valid")

    batchSize = 512
    trainLoader = DataLoader(
        trainData,
        batch_size=batchSize,
        shuffle=True,
        num_workers=8,
        pin_memory=True,
        prefetch_factor=2,
    )
    validLoader = DataLoader(
        validData,
        batch_size=batchSize,
        shuffle=True,
        num_workers=8,
        pin_memory=True,
        prefetch_factor=2,
    )

    trainLossList = []
    validLossList = []
    model.to(device)
    for epoch in range(epochs):
        start = time.time()
        model.train()
        trainAvgLoss = 0
        for pic, label in trainLoader:
            pic = pic.to(device)
            label = label.to(device)
            optimizer.zero_grad()
            output = model(pic)
```

```
            loss = criterion(output, label)
            loss.backward()
            optimizer.step()
            trainAvgLoss += loss.item() * batchSize
        trainAvgLoss /= len(trainData)
        trainLossList.append(trainAvgLoss)

        with torch.no_grad():
            model.eval()
            validAvgLoss = 0
            for pic, label in validLoader:
                pic = pic.to(device)
                label = label.to(device)
                output = model(pic)
                loss = criterion(output, label)
                validAvgLoss += loss.item() * batchSize
            validAvgLoss /= len(validData)
            validLossList.append(validAvgLoss)
        end = time.time()
        print(
            f"epoch {epoch}, train loss {trainAvgLoss}, valid loss {validA
vgLoss}, time {end-start}"
        )

        if lrDecay:
            scheduler.step()
    acc = calTop1Acc(model, trainLoader)
    return trainLossList, validLossList, acc, model
```

超参数搜索及结果

```python
def searchHyperParameter():
    depthList = [6, 5, 4]
    normalizeList = [True, False]
    dropoutList = [True, False]
    residualList = [True, False]
    lrDecayList = [True, False]
    bestAcc = 0
    bestParam = None
    for depth in depthList:
        for dropout in dropoutList:
            for normalize in normalizeList:
                for residual in residualList:
                    for lrDecay in lrDecayList:
                        param = (depth, dropout, normalize, residual, lrDecay)
                        print("(depth, dropout, normalize, residual, lrDecay)")
                        print(param)
                        _, _, acc, _ = trainOnce(*param)
                        print(f"ACC {acc}\n")
                        if acc > bestAcc:
                            bestAcc = acc
                            bestParam = param
    print("\nbest Param:", bestParam)
    print("best Acc:", bestAcc)
    return bestParam
```

我们对depthList, normalizeList, dropoutList, residualList, lrDecayList 进行超参数搜索具体范围为：

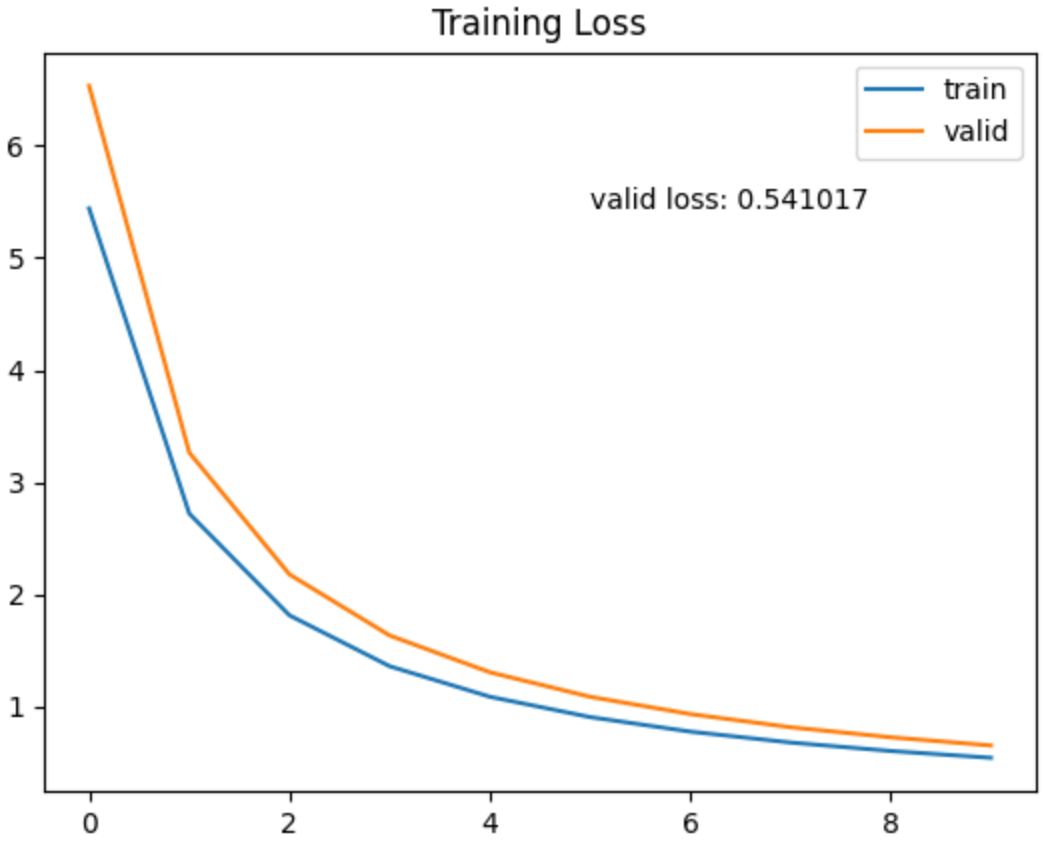
- depthList in [6, 5, 4]
- normalizeList in [True, False]
- dropoutList in [True, False]
- residualList in [True, False]
- lrDecayList in [True, False]

超参数搜索的最终结果为：

- depthList in [4]
- normalizeList in [True]
- dropoutList in [True]
- residualList in [True]
- lrDecayList in [True]

实验结果

最优超参数对应的训练损失可视化图像为：



将最优超参数搜索结果在测试集上进行测试的最终结果为：

Test ACC: 0.656375

实验收获

1. dropout是一个可以很好防止过拟合的办法，没有dropoutout，现有的网络容易过拟合
2. batchnorm和dropout一起用并没有之前认为会导致效果变差的情况，至少这个实验没有出现
3. 越深的网络不一定能带来更好的结果，我使用resnet152,resnet101，均没有resnet50效果好
4. batch_size也不是越大越好，我使用batch_size=512达到了最好的结果，但是batch_size=8224效果却很差
5. 综合尝试alexnet, vgg, resnet以后发现还是resnet效果最好