中国科学技术大学

University of Science and Technology of China

深度学习导论

实验 2

姓名 徐海阳　　学号 PB20000326　　院系 少年班学院

摘　要

本实用使用 PyTorch 实现卷积神经网络，在 Tiny-ImageNet 数据集上进行图片分类，完成数据生成、模型搭建、模型训练、调参分析及测试性能。特别地，调参分析中重点研究了以下设置对模型性能的影响：dropout, normalization, weight decay, residual connection, network depth 等超参数对分类性能的影响。

经过充分调优，本实验得到了最优的模型。同时，通过详细充分的消融实验、可视化图表和实验分析验证了模型的性能，在测试集上达到 Top 1 Accuracy(ACC)=72.61%。

目录

1 数据生成

这次实验使用 Tiny-Imagenet-200 数据集，包含 200 个类，每个类有 500 张训练图像，50 张验证图像和 50 张测试图像。由于测试图像没有标签，因此使用数据集中的验证集当作测试集，并从训练集中手动划分新的训练集和测试集（9:1）。如此一来，新的 train:val:test = 450:50:50。见 Section 7，我们对 Tiny_ImageNet 的数据格式做了处理，为了符合 torch 和 torchvision 自带的 DataSet 和 Dataloader 类的使用要求，从而省去自己搭建 DataSet 和 Dataloader 的过程。

2 模型搭建

本模型搭建选择基于 ConvNext[1] 的 codebase。ConvNext 是 CVPR2022 的文章，由 Facebook AI Research 和 UC Berkeley 出品。ConvNext 的性能如下图 1左所示，结构如下图 1右所示。相对 ResNet[2]，ConvNext 使用了：(1) 更大的卷积核，3->7；(2) BottleNeck 网络宽度 1:2:1 -> 1:4:1, (3) 激活函数和标准化函数，ReLU->GeLU，BN->LN。

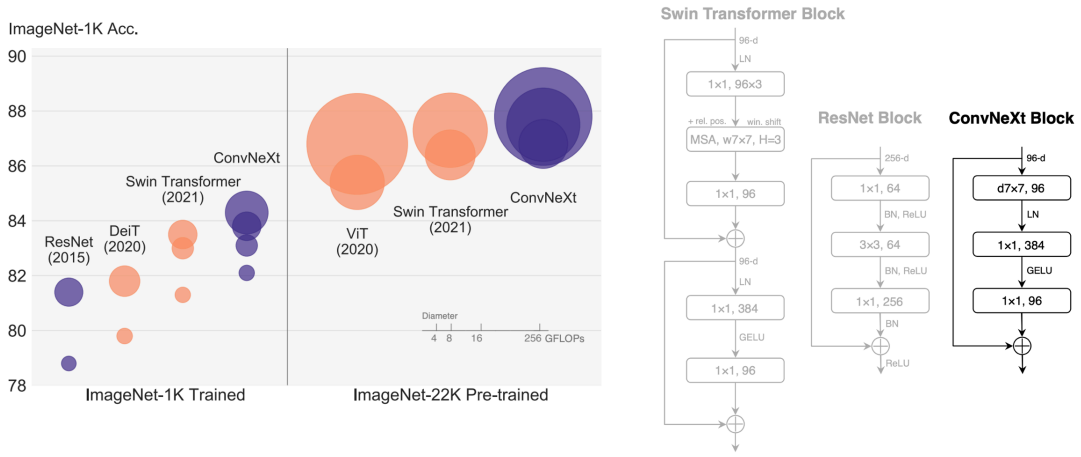

图 1: ConvNext Performance and Unit Structure

本实验中，基于 ConvNext 重新设计了网络结构，简化了网络深度和网络宽度，增加了 dropout, normalization, residual connection 的参数控制；同时设计了不同 scale 的 model，拥有不同的 network depth。基础结构基于如下图 2的 Block，整体结构就是 Block 的堆叠，与 ResNet 类似。

3 模型训练

用于训练的 main.py 如图 3，也基于 ConvNext 的 codebase 完成。

模型训练时，运行命令行如下图 4所示 (运行环境 4 * RTX3090)：

可以看到，第 7、8 行为本实验主要调整的参数：

drop_path: e.g. 0.0 / 0.1 / 0.2 / 0.3 / 0.4 / 0.5,

normalization: e.g. true / false,

weight_decay: e.g. 0.05 / 0.10 / 0.15 / 0.20,

residual: e.g. true / false。

在第 6 行 model 处是对 network depth 的调整，我设计了 3 种深度的 model，分别是：

convnext_small([3, 3, 9, 3]), convnext_base([3, 3, 18, 3]), convnext_large([3, 3, 27, 3])。即有 4 个 stage，每个 stage 有 x 个 Block。

```python
class Block(nn.Module):
    """ Lab Demand: dropout, normalization, learning rate decay, residual connection, network depth
    Args:
        dim (int): Number of input channels.
        drop_path (float): Stochastic depth rate. Default: 0.0
        normalization (bool): Whether to use normalization. Default: True
        residual (bool): Whether to use residual connection. Default: True
    """
    def __init__(self, dim, drop_path=0., normalization=True, residual=True):
        super().__init__()
        self.dwconv = nn.Conv2d(dim, dim, kernel_size=7, padding=3, groups=dim) # depthwise conv
        if normalization:
            self.norm = LayerNorm(dim, eps=1e-6)
            self.gamma = nn.Parameter(1e-6 * torch.ones((dim)), requires_grad=True)
        self.pwconv1 = nn.Linear(dim, 4 * dim) # pointwise/1x1 convs, implemented with linear layers
        self.act = nn.GELU()
        self.pwconv2 = nn.Linear(4 * dim, dim)
        self.drop_path = DropPath(drop_path) if drop_path > 0. else nn.Identity()
        self.normalization = normalization
        self.residual = residual

    def forward(self, x):
        input = x
        x = self.dwconv(x)
        x = x.permute(0, 2, 3, 1) # (N, C, H, W) -> (N, H, W, C)
        if self.normalization:
            x = self.norm(x)
        x = self.pwconv1(x)
        x = self.act(x)
        x = self.pwconv2(x)
        if self.normalization:
            x = self.gamma * x
        x = x.permute(0, 3, 1, 2) # (N, H, W, C) -> (N, C, H, W)
        if self.residual:
            x = input + self.drop_path(x)
        return x
```

图 2: Block in the Model

4 调参分析

将训练好的模型在验证集上进行测试，以 Top 1 Accuracy(ACC) 作为网络性能指标。然后，对 dropout, normalization, weight decay, residual connection, network depth 进行调整，再重新训练、测试，并分析对模型性能的影响。为了节省计算量和统一对比，调参分析中的所有模型都运行 120 epochs。

首先，我们分析 dropout 对于模型性能的影响。Dropout 是一种常用的正则化方法，可以减轻过拟合问题，从而提高模型的泛化能力。Dropout Ratio 表示在训练过程中需要随机丢弃的神经元比例。

Dropout Ratio	Val Acc@1(%) ↑
0.0	53.3
0.1	**54.7**
0.2	53.4
0.3	53.6
0.4	53.5
0.5	52.6

表 1: Ablation of Dropout Ratio

```
# ------------------------------------------------------------------------------------------
# Modified by Haiyang Xu for 2023SP Deep Learning Lab in USTC
# Modify the model for Lab2

# ------------------------------------------------------------------------------------------
# Copyright (c) Meta Platforms, Inc. and affiliates.

# All rights reserved.

# This source code is licensed under the license found in the
# LICENSE file in the root directory of this source tree.
# ------------------------------------------------------------------------------------------
```

图 3: ConvNext Database

```
python -m torch.distributed.run --nproc_per_node=4 main.py \
--input_size 64 --epochs 120 \
--batch_size 64 --lr 4e-3 --update_freq 4 \
--data_set TINY_IMNET \
--data_path /mnt/nvme2/xuhaiyang/data/tiny-imagenet-200/ \
--model convnext_small \
--drop_path 0.1 --normalization true \
--weight_decay 0.1 --residual true \
--output_dir outputs/m-s+dp-01+n-T+wd-01+r-T
```

图 4: Train Script

如表 1所示，当 Dropout Ratio 为 0.1 时，模型的验证准确率最高，达到了 54.7%，而当 Dropout Ratio 为 0.5 时，模型的验证准确率最低，只有 52.6%。在其余的 Dropout Ratio 的值下，模型的验证准确率都与 Dropout Ratio 为 0.0 时相差不大。从表格中可以看出，当 Dropout Ratio 值为 0.0 时，即没有使用 Dropout 正则化，模型的验证准确率为 53.3%。而当 Dropout Ratio 值为 0.1 时，模型的验证准确率达到最高点，可以推断出在这种情况下 Dropout 正则化起到了一定的作用。然而，当 Dropout Ratio 值继续增加时，模型的验证准确率并没有进一步提高，甚至略微下降，这说明当 Dropout Ratio 超过一定值时，会对模型的性能产生负面影响。因此，使用适当的 Dropout Ratio 可以提高模型的性能，但是过高或过低的 Dropout Ratio 都可能导致模型性能的下降。基于此，本实验使用 dropout 为 0.1。

接着，我们分析 Normalization 和 Residual 对模型性能的影响。Normalization 技术通常被用来对数据进行预处理，以缩放数据范围、使数据更易于处理。对于神经网络，常用的 normalization 技术包括 Batch Normalization、Layer Normalization 等。使用 normalization 技术可以减少内部协方差移位，提高模型的收敛速度和稳定性; Residual 技术则是一种跨层连接技术，可以帮助模型在训练过程中更好地捕获数据中的信息。使用 residual 技术可以避免梯度消失或梯度爆炸等问题，提高模型的训练速度和稳定性。

Norm	Residual	Val Acc@1(%) ↑
✗	✗	0.5
✓	✗	0.5
✗	✓	11.5
✓	✓	**54.7**

表 2: Ablation of Norm and Residual

如表 2所示，当同时使用 normalization 和 residual 时，模型的验证准确率最高，达到了 54.7%。而当不使用这两种技术时，模型的验证准确率都非常低，仅为 0.5%（相当于随机分类，模型啥也没学到）。可以推断出，使用 normalization 和 residual 技术对于提高模型的性能非常重要。这也不难理解：图像分类的数据分布还是比较复杂的，因此 normalization 减少内部协方差移位可以大幅提高模型的收敛速度和稳定性；而本模型本身比较深（convnext_small 54 层, convnext_base 81 层, convnext_large 108 层），因此 residual 对于模型也非常重要，否则会出现梯度消失和梯度爆炸现象。基于此，本实验同时使用 normalization 和 residual。

然后，我们分析 weight_decay 对于模型性能的影响。Weight decay 技术是一种正则化方法，通过对模型的权重进行惩罚，可以减少模型的过拟合风险。使用 weight decay 技术可以控制模型的复杂度，避免模型对训练数据的过度拟合，提高模型的泛化能力。

Weight Decay	Val Acc@1(%) ↑
0.05	54.7
0.10	55.7
0.15	56.4
0.20	**57.0**
0.30	54.9

表 3: Ablation of Weight Decay

如表 3所示，当 weight decay 值为 0.20 时，模型的验证准确率最高，达到了 57.0%。因此，使用适当的 weight decay 技术可以减轻模型的过拟合现象，提高模型的泛化能力。太多的正则化也不好，会损害模型的性能。本实验选择 Weight Decay 为 0.2。

接着，我们分析 network depth 对于模型性能的影响。带有 residual 的卷积神经网络，往往是越深拟合能力越强，网络性能也就越强。

Network Depth	Val Acc@1(%) ↑
[3, 3, 9, 3]	54.7
[3, 3, 9, 3]	55.1
[3, 3, 9, 3]	**55.9**

表 4: Ablation of Network Depth

根据表格 4 的结果，我们可以看到网络深度对模型性能有着显著的影响。我们使用了不同的卷积层深度来探究网络深度的影响，四个数字分别对应四个 stage 的深度。从表格中可以看出，当网络深度为 [3, 3, 27, 3] 时，模型的验证准确率达到了 55.9%，是所有网络深度中最高的。这表明，增加网络深度可以提高模型的性能，但同时也可能增加模型的计算和训练成本，因此需要权衡和平衡。因此，我们选择网络深度为 [3, 3, 18, 3]。

下一步，我们分析输入图片的 resolution 对模型性能的影响。

根据表格 5 的结果，我们可以看到图片分辨率对模型性能有着显著的影响。我们使用了不同的图片分辨率来探究网络深度的影响，当图片分辨率为 256 时，模型的验证准确率达到了 55.9%，是所有图片分辨率中最高的。这表明，增加图片分辨率可以提高模型的性能，但同时也会大幅增加模型的计算和训练成本。因此我们最终选择了图片分辨率为 (192, 192)，这样可以平衡计算量和性能。

Image Resolution	Val Acc@1(%) ↑
(64, 64)	54.7
(128, 128)	63.6
(192, 192)	67.0
(256, 256)	**68.8**

表 5: Ablation of Resolution

5 测试性能

最终，根据调参，我们选择 dropout ratio = 0.1，采用 normalization 和 residual，weight decay = 0.2, network depth = [3, 3, 18, 3], input resolution = (192, 192), training epochs = 240 (不早挺，收敛性见下图 7，240 epochs 收敛）进行训练。

训练结束之后，进行测试。在命令行输入如图 5，得到在最优模型在测试集上的 Top 1 Accuracy(ACC) 为 72.61%，如图 6。

```
python main.py --model convnext_base --eval true \
--resume /mnt/nvme2/xuhaiyang/USTC/DL/Lab2/outputs/m-b+dp-02+n-T+wd-005+r-T+e-240+i-192/checkpoint-best.pth \
--input_size 192 --data_set TINY_IMNET \
--data_path /mnt/nvme2/xuhaiyang/data/tiny-imagenet-200/ \
--drop_path 0.2 --normalization true \
--weight_decay 0.05 --residual true \
--use_amp true
```

图 5: Test Evaluate Command

```
Resume checkpoint /mnt/nvme2/xuhaiyang/USTC/DL/Lab2/outputs/m-b+dp-02+n-T+wd-005+r-T+e-240+i-192/checkpoint-best.pth
With optim & sched!
Eval only mode
Test:  [  0/105]  eta: 0:05:25  loss: 0.4798 (0.4798)  acc1: 92.7083 (92.7083)  acc5: 97.9167 (97.9167)  time: 3.0996  data: 1.5448  max mem: 3772
Test:  [ 10/105]  eta: 0:00:35  loss: 1.0496 (1.0729)  acc1: 78.1250 (77.9356)  acc5: 92.7083 (91.2879)  time: 0.3781  data: 0.1406  max mem: 3772
Test:  [ 20/105]  eta: 0:00:21  loss: 1.0496 (1.0990)  acc1: 77.0833 (77.5794)  acc5: 91.6667 (90.9226)  time: 0.1058  data: 0.0002  max mem: 3772
Test:  [ 30/105]  eta: 0:00:15  loss: 0.8977 (1.0516)  acc1: 79.1667 (78.0914)  acc5: 92.7083 (91.4651)  time: 0.1058  data: 0.0002  max mem: 3772
Test:  [ 40/105]  eta: 0:00:11  loss: 1.1116 (1.1423)  acc1: 76.0417 (75.9654)  acc5: 90.6250 (90.5234)  time: 0.1059  data: 0.0002  max mem: 3772
Test:  [ 50/105]  eta: 0:00:09  loss: 1.2818 (1.2009)  acc1: 70.8333 (74.6936)  acc5: 88.5417 (89.6242)  time: 0.1058  data: 0.0001  max mem: 3772
Test:  [ 60/105]  eta: 0:00:06  loss: 1.2133 (1.2193)  acc1: 68.7500 (74.0608)  acc5: 87.5000 (89.3101)  time: 0.1058  data: 0.0001  max mem: 3772
Test:  [ 70/105]  eta: 0:00:05  loss: 1.3431 (1.2614)  acc1: 68.7500 (73.1661)  acc5: 87.5000 (88.6297)  time: 0.1059  data: 0.0001  max mem: 3772
Test:  [ 80/105]  eta: 0:00:03  loss: 1.3222 (1.2759)  acc1: 72.9167 (73.1739)  acc5: 86.4583 (88.3488)  time: 0.1059  data: 0.0001  max mem: 3772
Test:  [ 90/105]  eta: 0:00:02  loss: 1.1425 (1.2841)  acc1: 75.0000 (73.0655)  acc5: 87.5000 (88.1410)  time: 0.1059  data: 0.0001  max mem: 3772
Test:  [100/105]  eta: 0:00:00  loss: 1.0944 (1.2864)  acc1: 70.8333 (72.7620)  acc5: 89.5833 (88.3457)  time: 0.1058  data: 0.0001  max mem: 3772
Test:  [104/105]  eta: 0:00:00  loss: 1.2033 (1.2903)  acc1: 70.8333 (72.6100)  acc5: 90.6250 (88.4100)  time: 0.1037  data: 0.0000  max mem: 3772
Test: Total time: 0:00:14 (0.1364 s / it)
* Acc@1 72.610 Acc@5 88.410 loss 1.290
Accuracy of the network on 10000 test images: 72.61000%
```

图 6: Evaluation Result on Test Set

同时，得到 Train 和 Val 的 loss curve，如图 7。同时画出了在 validation set 上的 Top 1 Accuracy(ACC) Curve，如图 8。可以看出，模型在 240 epochs 充分收敛。Train Loss 始终大于 Val Loss，看出模型事实上过拟合了。但是在 Test Set 上的 Top 1 Accuracy(ACC) 和 Validation Set 的 Top 1 Accuracy(ACC) 是差不多的。并且都是在 240 epochs 左右收敛。说明（1）模型的拟合能力很强，过拟合了 Train Set；（2）Train / Val / Test Set 的数据分布较为统一。

6 实验总结

本实验通过大量充分的实验来在 Tiny-ImageNet 数据集上进行图片分类。通过对 dropout, normalization, weight decay, residual connection, network depth 等超参数的调优，在测试集上达到了较好的效果。分辨率增大、Epoch 增加的情况下，有进一步提高 Top 1 Accuracy(ACC) 的空间。

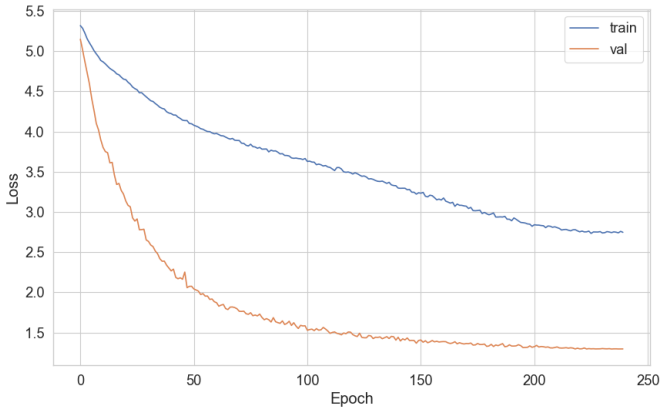

图 7: Loss Curve on Train / Val Set

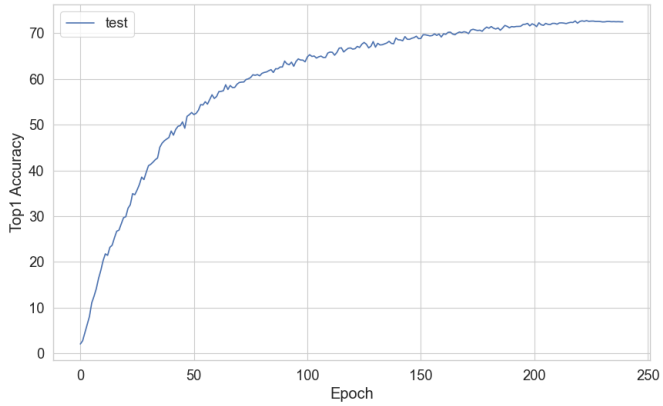

图 8: Top 1 Accuracy(ACC) Curve on Val Set

7 附录

src 文件夹是源程序的文件夹，包含所有的模型、训练、验证代码，训练和验证的命令见图 4和图 5; 包含一个 process_tiny.py 文件，需要放到解压后的 Tiny_ImageNet 数据文件夹下运行，来满足 torchvision DataSet 类的格式要求。

results 文件夹是输出 log 的文件夹，包含所有调参过程 ablation study 的 log。

参考文献

[1] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778, IEEE, June 2016.