深圳大学考试答题纸

(以论文、报告等形式考核专用)

二〇 二一 ～二〇 二二 学年度第 二 学期

课程编号	1502990001	课序号	01	课程名称	机器学习导论	主讲教师	贾森、徐萌	评分	

学号	2019044031	姓名	邓恺俊	专业年级	2019 级计算机科学与技术 06 班

教师评语：

题目： 大数据驱动的多种深度模型在图像分类中的对比（ConvNeXt 篇）

【摘要】本文主要研究的是各个大数据驱动的深度模型在图像分类上的使用，简述了卷积神经网络的发展，从 ResNet->ResNeXt->ConvNeXt，本文主要研究 ConvNext。ConvNeXt 主要是借鉴了 Swin Transformer 设计的思想，不断的进行学习和改进，设计出了比其更优的模型，证明了卷积神经网络的受限大程度上是因为学习策略的问题，并不是 Transformer 就一定优于卷积网络，利用 Transformer 的丹方依旧能够训练处很好的模型。在实验中训练了 ConvNeXt-T，在猫狗数据集上准确率为 99.7%，在三个模型中达到了 state of the art 的水平。

【关键词】大数据；深度学习；图像分类；ConvNeXt；Swin Transformer;

【作者】邓恺俊，刘凯盛，马铭鹤（按姓名字母排序，不分前后）

【代码】本代码已经上传到 Github [1]中，详情可以查看代码，组内分工分别是：马铭鹤负责 ResNet（2016）[4]，刘凯盛负责 ResNeXt[5]，邓恺俊负责 ConvNeXt[2][1]。

[1] Github code：https://github.com/Dreaming-future/Pytorch-Image-Classification/ConvNeXt

[2] ConvNeXt 篇的代码量大约是 600 余行，利用 Pytorch 进行编写，训练以及测试

Background

本次项目中，主要是在大数据驱动的环境下，利用深度学习的多种模型在图像分类上进行应用，人工智能与深度学习在近几年以及风靡全球，生活中的许许多多的应用也许都能通过深度学习来解决，其中图像分类就是一种，这种下游任务早在 2010 年前就有很多人研究，特别是当时举办了 ILSVRC（ImageNet Large Scale Visual Recognition Challenge）。ILSVRC 是近年来机器视觉领域最受追捧也是最具权威的学术竞赛之一，代表了图像领域的最高水平。ImageNet 数据集[3]是 ILSVRC 竞赛使用的是数据集，由斯坦福大学李飞飞教授主导，包含了超过 1400 万张全尺寸的有标记图片。ILSVRC 比赛会每年从 ImageNet 数据集中抽出部分样本，以 2012 年为例，比赛的训练集包含 1281167 张图片，验证集包含 50000 张图片，测试集为 100000 张图片。

近几年，许许多多的深度学习模型都在图像分类中大放光彩，首先是 2012 年的 AlexNet[7]作为卷积神经网络的开山鼻祖，掀起了卷积神经网络的热潮，比较震撼人心的是，在 2016 斩获 CVPR 的 best paper 的 ResNet[4]，第一次在 CIFAR 数据集上超越了人的准确率，这无疑踏出了关键性的一步，紧接着又有许许多多的模型出现，包括 2017 年的 DenseNet[8]和 ResNet 的作者又在 ResNet 改进得到了 ResNeXt[5]。

卷积神经网络如雨后春笋般涌入计算机视觉领域中，但是，自从 ViT 提出之后，在过去的一年里（2021 年），基于 transformer 的模型在计算机视觉各个领域全面超越 CNN 模型。然而，这很大程度上都归功于 Local Vision Transformer 模型，Swin Transformer 是其中重要代表，Swin Transformer[3] 在图像领域（分类下游任务）的全面大幅度超越 CNN 模型， 仿佛印证了 Attetion 论文中 “Attention Is All You Need ”[2]。

Introduction

在新时代中，是否卷积神经网络就已经被时代淘汰了呢！FaceBook 研究所的“A ConvNet for the 2020s”[1]，即 ConvNeXt 这篇文章，通过借鉴 Swin Transformer 精心构建的 tricks，卷积在图像领域反超 Transform。这些技巧对分类问题下游 downstream 的问题也有效果。简单的来说，似乎就是说明，用 Swin Transformer 的丹方，在卷积神经网络中炼丹也有很好的效果。

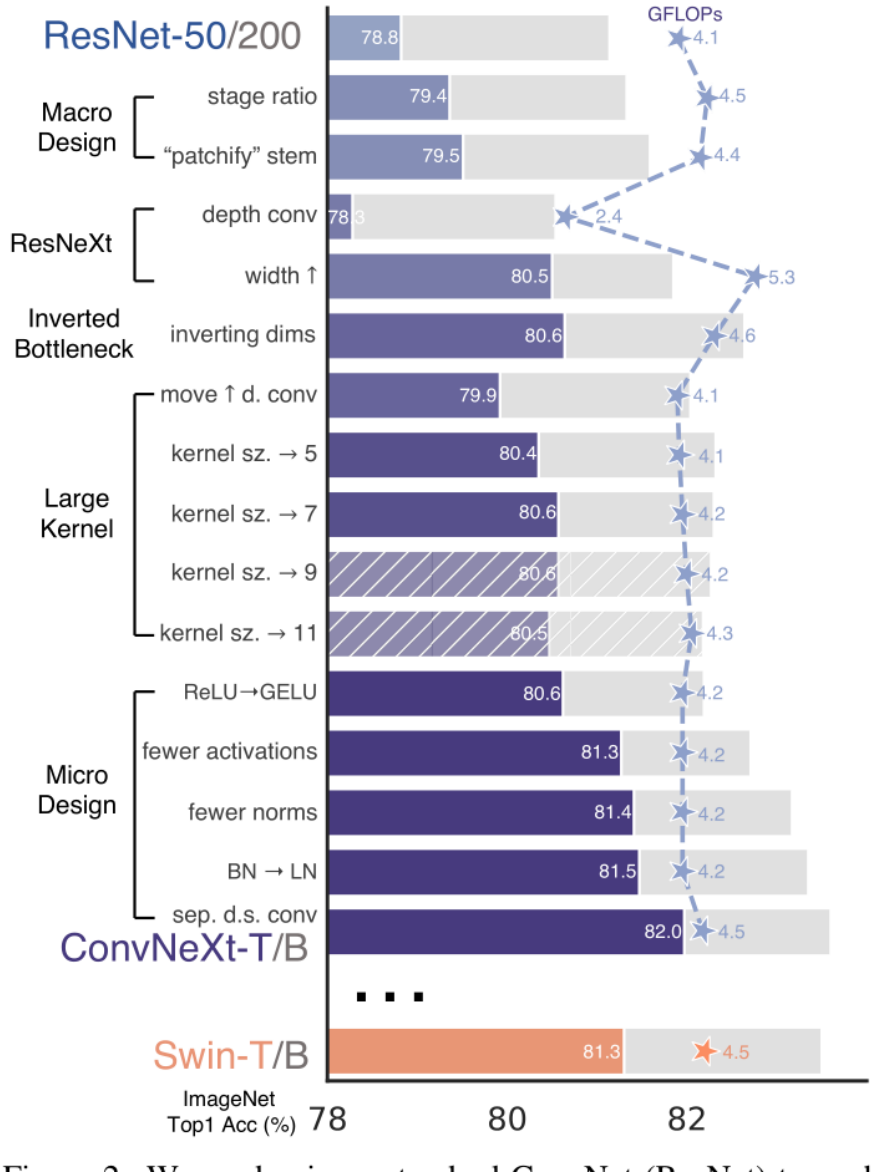
本篇文章就是基于这个新时代的卷积神经网络，本文为 ConvNeXt 篇，就此展开卷积的新时代，讲述从 ResNet->ResNeXt->ConvNeXt 的一个过程和方法，简要介绍设计的思想和方法，以及结合相关代码的思想和方法，最终给出实验结果。

Modernizing a ConvNet:a Roadmap

ConvNeXt 是基于 ResNet-50 的结构，参考 Swin Transformer 的思想进行现代化改造，直到卷积模型超过 trans-based 方法的 SOTA 效果。ConvNeXt 从原始的 ResNet 出发，逐步加入 Swin Transformer 的 trick，来改进模型。论文中适用 ResNet 模型：ResNet50 和 ResNet200。其中 ResNet50 和 Swin-T 有类似的 FLOPs（4G vs 4.5G），而 ResNet200

[3] https://image-net.org/

和 Swin-B 有类似的 FLOPs（15G）。首先做的改进是调整训练策略，然后是模型设计方面的递进优化：宏观设计->ResNeXt 化->改用 Inverted bottleneck->采用 large kernel size->微观设计。由于模型性能和 FLOPs 强相关，所以在优化过程中尽量保持 FLOPs 的稳定，也就是希望，在计算能力相同的情况下，比较模型性能的优劣。



图表 1 ResNet-50 -> ConvNeXt 的 Roadmap

Training methods

在 ConvNeXt 中，它的优化策略借鉴了 Swin Transformer。具体的优化策略包括：

（1）将训练 Epoch 数从 90 增加到 300；

（2）优化器从 SGD 改为 AdamW；

（3）更复杂的数据扩充策略，包括 Mixup，CutMix，RandAugment，Random Erasing 等；

（4）增加正则策略，例如随机深度，标签平滑，EMA 等。

更具体的预训练和微调的超参数如图。AdamW， 300epoch，预训练学习率 4e-3，weight decay= 0.05，batchsize = 4096；微调学习率 5e-5， weight decay= 1e-8 ，batchsize = 512，这样的方法，精度直接从 76.1 升到了 78.8

(pre-)training config	ConvNeXt-T/S/B/L ImageNet-1K 224^2	ConvNeXt-T/S/B/L/XL ImageNet-22K 224^2
weight init	trunc. normal (0.2)	trunc. normal (0.2)
optimizer	AdamW	AdamW
base learning rate	4e-3	4e-3
weight decay	0.05	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
batch size	4096	4096
training epochs	300	90
learning rate schedule	cosine decay	cosine decay
warmup epochs	20	5
warmup schedule	linear	linear
layer-wise lr decay [6, 12]	None	None
randaugment [14]	(9, 0.5)	(9, 0.5)
mixup [90]	0.8	0.8
cutmix [89]	1.0	1.0
random erasing [91]	0.25	0.25
label smoothing [69]	0.1	0.1
stochastic depth [37]	0.1/0.4/0.5/0.5	0.0/0.0/0.1/0.1/0.2
layer scale [74]	1e-6	1e-6
head init scale [74]	None	None
gradient clip	None	None
exp. mov. avg. (EMA) [51]	0.9999	None

图表 2 训练方式

Statio ration

卷积神经网络发展了这么多年，其结构都是经过无数研究者之手雕琢出来的，比如多阶段设计思想。如下图的 ResNet50 为例，多阶段 block 数为 [3,4,6,3]。Swin Transformer 参考了这样的多阶段结构，相应比例为 [2,2,6,2]，更大的 Swin 为 [2,2,18,2]，于是，ConvNeXt 就参考了一下 Swin Transformer 高比例的设置，将多阶段 block 数从 ResNet 的 [3,4,6,3] 变成了 [3,3,9,3]，这样做精度涨了 0.6 个点左右->79.4%

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图表 3 ResNet 的结构

实际上，在 RegNet 和 EfficientNetV2 的论文中都有指出，后面的 stages 应该占用更多的计算量。所以说，将后面的 stage 调为 9 是对结果有好处的。ConvNeXt 参考这样的设计，根据不同的计算量 FLOPs，设计不同的参数大小的网络结构。

- Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers =$\{2, 2, 18, 2\}$

- ConvNeXt-T: $C = (96, 192, 384, 768)$, $B = (3, 3, 9, 3)$
- ConvNeXt-S: $C = (96, 192, 384, 768)$, $B = (3, 3, 27, 3)$
- ConvNeXt-B: $C = (128, 256, 512, 1024)$, $B = (3, 3, 27, 3)$
- ConvNeXt-L: $C = (192, 384, 768, 1536)$, $B = (3, 3, 27, 3)$
- ConvNeXt-XL: $C = (256, 512, 1024, 2048)$, $B = (3, 3, 27, 3)$

图表 4 ConvNeXt 的改进

Patchy

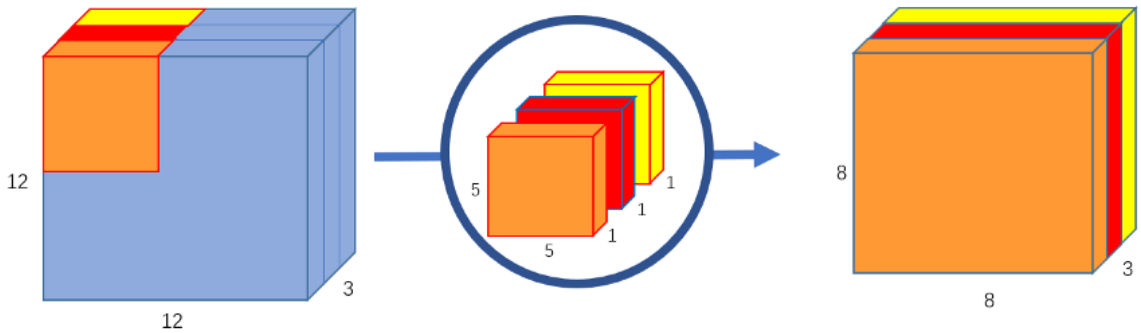
对于 ImageNet 数据集，我们通常采用 224x224 的输入尺寸，这个尺寸对于 ViT 等基于 Transformer 的模型来说是非常大的，它们通常使用一个步长为 4，大小也为 4 的卷积将其尺寸降采样到 56x56 。因为这个卷积的步长和大小是完全相同的，所以它又是一个无覆盖的卷积，或者叫 Patchify 的卷积。这一部分在 Swin-Transformer 中叫做 stem 层，它是位于输入之后的一个降采样层。

在 ConvNeXt 中，Stem 层也是一个步长为 4，大小也为 4 的卷积操作，这一操作将准确率从 79.4%提升至 79.5%，GFLOPs 从 4.5 降到 4.4%。也有人指出使用覆盖的卷积（例如步长为 4，卷积核大小为 7 的卷积）能够获得更好的表现。

```
1.  stem = nn.Sequential(
2.      nn.Conv2d(in_chans, dims[0], kernel_size=4, stride=4),
3.      LayerNorm(dims[0], eps=1e-6, data_format="channels_first")
4.  )
```

ResNeXt-ify

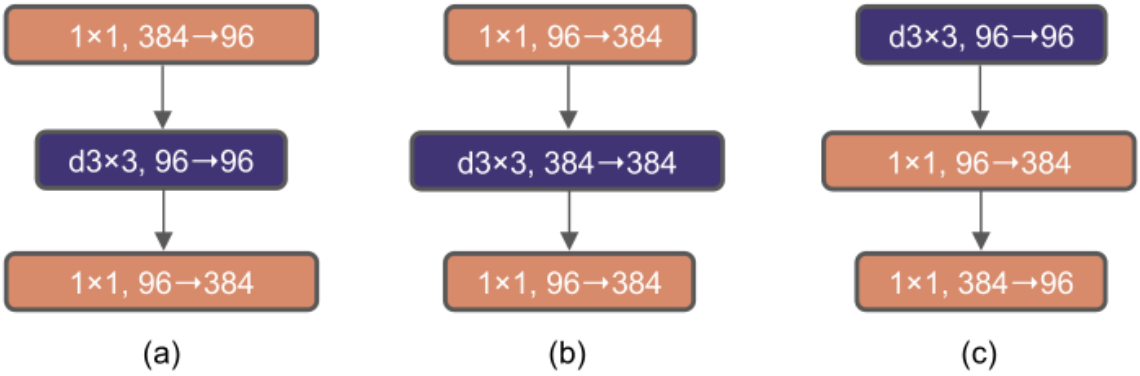
在我们的第二部分中详细描述了 ResNeXt 网络，这一部分主要使用到了深度可分离卷积，深度可分离卷积最先是在 MobileNet 中提出的，目的是为了解决在嵌入式设备中进行推理的过程。深度可分离卷积对参数和精度有一个很好的 trade off，所以 ConvNeXt 也参照 ResNeXt 的思想，将普通卷积替换成深度可分离卷积，为了和 Swin-T 的 FLOGs 相近，论文使用深度卷积即极限的分组卷积并将宽度从 64 扩展为 96，这样也得到了很好的结果，精度上升了 1%->80.5%



图表 5 深度可分离卷积

Inverted Bottleneck

除此之外，我们发现 Transformer 的 FFN 层是一个 Inverted Bottleneck 结构。ResNeXt 的 Bottleneck 的设计是大通道维度->小通道维度(3×3 卷积)->大通道维度，所以借鉴 MobileNetV2[6]的中的 Inverted Bottleneck 和参考 Swin Transformer 的设计，ConvNext 的 Bottleneck 设计为小通道维度->大通道维度(3×3 卷积)->小维度。Inverted Bottleneck 特点是两头窄中间宽，于是 ConvNeXt Block 在 Res Block 上做了改进，维度变化为 96->384->96,精度:80.5%->80.6%（ResNet-200：81.9%->82.6）。



图表 6 Inverted Bottleneck

Large Kernel Sizes

ViT 由于 non-local 自注意力的实现方式，使得每一层都具有全局感受野，虽然 Swin Transformer 在自注意力模块中引入了 local windows，但是 window 的尺寸也不会小于 7×7，不同于当下卷积网络使用 3×3 卷积堆叠代替大卷积核以获得更好的精度。在 2014 年论文的 VGG 中说，利用堆叠的 3x3 的卷积核进行提取特征，并且论文中说通过堆叠多个 3x3 的窗口可以替代一个更大的窗口，所以现在主流的卷积神经网络都是采用 3x3 大小的窗口。不过这样子也会存在一些缺陷，会导致感受野的范围比较小，而利用大的卷积核可以增大感受野，在某一程度可以得到更多的信息。

接着作者将 depthwise conv 的卷积核大小由 3x3 改成了 7x7（和 Swin Transformer 一样），当然作者也尝试了其他尺寸，包括 3, 5, 7, 9, 11 发现取到 7 时准确率就达到了饱和。并且准确率从 79.9% (3×3)增长到 80.6% (7×7)。并且比较神奇的是，发现取到 7 的时候，我们的精度达到了最大，并且这个也是 Swin Transformer 的窗口大小一样的。

Micro Design

除此之外，还有一些更小的差异，比如激活函数和 Normalization 等等，我们可以比较一下关于 Swin Transformer 和 ConvNeXt 的不同。

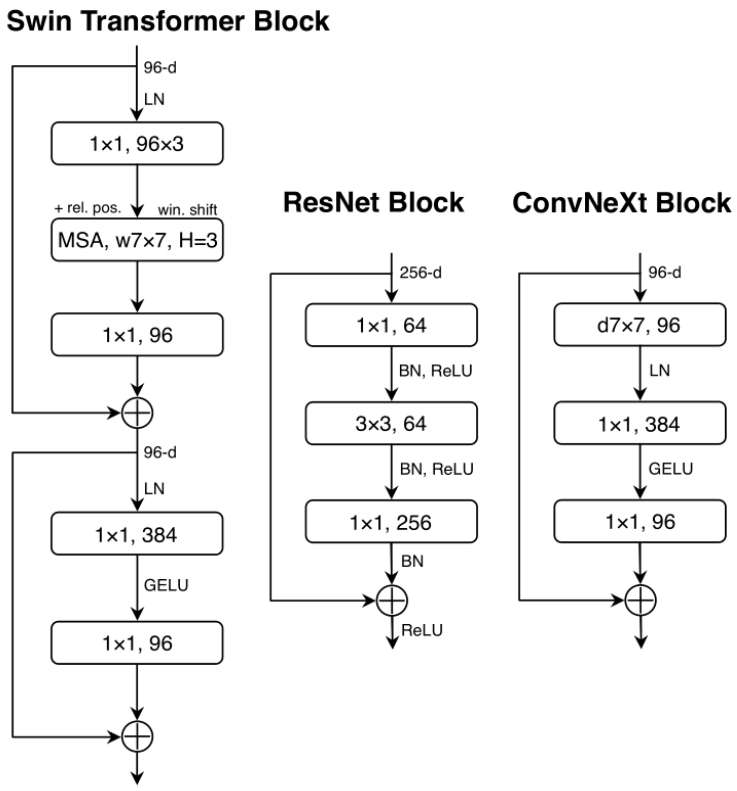


Figure 4. **Block designs** for a ResNet, a Swin Transformer, and a

图表 7 ConvNeXt Block

替换 ReLU 为 GeLU：在 Transformer 中激活函数基本用的都是 GELU，而在卷积神经网络中最常用的是 ReLU，于是将激活函数替换成了 GELU，替换后发现准确率没变化；

更少的激活函数：使用更少的激活函数。在卷积神经网络中，一般会在每个卷积层或全连接后都接上一个激活函数。但在 Transformer 中并不是每个模块后都跟有激活函数，比如 MLP 中只有第一个全连接层后跟了 GELU 激活函数。接着在 ConvNeXt Block 中也减少激活函数的使用，改动后 ConvNext 只保留了两个 1×1 卷积之间的激活函数，**精度：80.6%->81.3%**；

更少的正则化层：使用更少的 Normalization。同样在 Transformer 中，Normalization 使用的也比较少，接着也减少了 ConvNeXt Block 中的 Normalization 层，只保留了 depthwise conv 后的 Normalization 层。此时准确率已经达到了 81.4%，已经超过了 Swin-T, 精度：81.3%->81.4%；

BN 替换为 LN：将 BN 替换成 LN。Batch Normalization（BN）在卷积神经网络中是非常常用的操作了，它可以加速网络的收敛并减少过拟合。但在 Transformer 中基本都用的 Layer Normalization（LN），因为最开始 Transformer 是应用在 NLP 领域的，BN 又不适用于 NLP 相关任务。接着作者将 BN 全部替换成了 LN，发现准确率还有小幅提升达到了 81.5%,精度：81.4%->81.5%；

独立的下采样层：ResNet 的下采样操作是在每个阶段的开始阶段使用步长为 2 的 3×3 卷积和直连步长为 2 得 1×1 卷积完成，Swin Transformers 中则是在不同阶段之间进行独立得下采样，ConvNext 采用相同得策略，使用步长为 2 得 2×2 卷积进行空间下采样，这个改动会导致训练不稳定，所以在下采样操作前、Stem 后以及全局池化层之后加入了一些 LN 层来稳定训练，**精度：81.5%->82.0%**。

```python
class Block(nn.Module):
    def __init__(self, dim, drop_path=0., layer_scale_init_value=1e-6):
        super().__init__()
        self.dwconv = nn.Conv2d(dim, dim, kernel_size=7, padding=3, groups=dim) # depthwise conv
        self.norm = LayerNorm(dim, eps=1e-6)
        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.gamma = nn.Parameter(layer_scale_init_value * torch.ones((dim)), requires_grad=True) if layer_scale_init_value > 0 else None
        self.drop_path = DropPath(drop_path) if drop_path > 0. else nn.Identity()
    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)
        x = self.norm(x)
        x = self.pwconv1(x)
        x = self.act(x)
        x = self.pwconv2(x)
        if self.gamma is not None:
            x = self.gamma * x
        x = x.permute(0, 3, 1, 2) # (N, H, W, C) -> (N, C, H, W)
        x = input + self.drop_path(x)
        return x
```

ConvNeXt Variants

对于 ConvNeXt 网络，论文提出了 T/S/B/L 四个版本，计算复杂度刚好和 Swin Transformer 中的 T/S/B/L 相似，下图给出了他们的参数结构和在 ImageNet-1K 的 Top-1 的准确率

网络结构	输入尺寸	通道数	Block数	准确率
ConvNeXt-T	224	96, 192, 384, 768	3, 3, 9, 3	82.1%
ConvNeXt-S	224	96, 192, 384, 768	3, 3, 27, 3	83.1%
ConvNeXt-B	384	128, 256, 512, 1024	3, 3, 27, 3	85.1%
ConvNeXt-L	384	192, 384, 768, 1536	3, 3, 27, 3	85.5%
ConvNeXt-XL	384	256, 512, 1024, 2048	3, 3, 27, 3	87.8%

图表 8 ConvNeXt varients

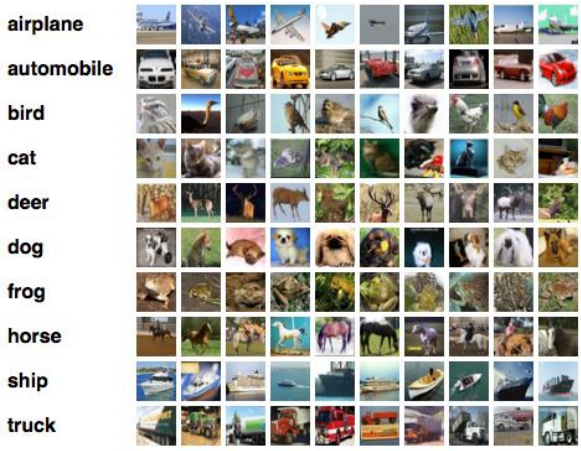
Experiments

在我们实验过程中，我们主要利用了两个数据集，分别是 CIFAR10 数据集以及 kaggle 上的经典数据集（猫狗数据集[4]）作为训练集和测试集，这里也介绍一下数据集。

CIFAR10 数据集

CIFAR10[5]是 kaggle 计算机视觉竞赛的一个图像分类项目。该数据集共有 60000 张 32*32 彩色图像，一共分为"plane"，“car”，“bird”,“cat”，“deer”，“dog”，“frog”，“horse”,“ship”，“truck” 10 类，每类 6000 张图。有 50000 张用于训练，构成了 5 个训练批，每一批 10000 张图；10000 张用于测试，单独构成一批。

- CIFAR-10 是 3 通道的彩色 RGB 图像，而 MNIST 是灰度图像。
- CIFAR-10 的图片尺寸为 32×32， 而 MNIST 的图片尺寸为 28×28，比 MNIST 稍大。
- 相比于手写字符， CIFAR-10 含有的是现实世界中真实的物体，不仅噪声很大，而且物体的比例、 特征都不尽相同，这为识别带来很大困难。



图表 9 CIFAR10 **数据集**

最后我们在这几个数据集进行了测试，在这一部分，我并没有在 CIFAR10 数据集中进行训练和测试，因为我发现训练和测试的结果不是很好。得到这样结果的原因可能是，比如 Transformer 是需要大数据集的，所以简单的 CIFAR10 的数据集，并且分辨率只有 32x32，所以蕴含的信息比较少，神经网络很难去训练得到更好的结果，我也尝试着增大分辨率，并且也在猫狗数据集上进行训练和测试，也得到了很好的结果。我们最后在猫狗数据集上得到了 99.7%的准确率，类比于 ResNet50 和 ResNeXt50_32x4d 来说，准确率提高了大概有 1.3%，达到了 state of the art。

Model	Data set	Accuracy
ResNet50	Cats and dogs	98.2%
ResNeXt50_32x4d	Cats and dogs	98.4%
ConvNeXt-T	Cats and dogs	99.7%
ResNet50	Cifar10	93.5%
ResNext50	Cifar10	94%

[4] https://www.kaggle.com/competitions/dog-vs-cat-classification
[5] https://www.kaggle.com/competitions/cifar10-cn

Conclusion

ConvNeXt 并没有特别复杂或者创新的结构，它的每一个网络细节都是已经在不止一个网络中被采用。而就是靠这些策略的互相配合，却也达到了 ImageNet Top-1 的准确率，它涉及这些边角料的动机也非常简单：Transformer 或者 Swin-Transformer [3]怎么做，我也对应的调整，效果好就保留。

随着 Transformer 逐渐在计算机视觉领域中大放异彩，但是也没有什么资料说明 Self-Attention 是在原理上比卷积更适用于图像任务。它本质上还是矩阵的密集计算，它的提出有很大的功劳是得益于计算性能的提升。在初期 AlexNet 提出的时候，还是利用两个 GPU 并行计算进行的，而随着时代的发展，我们的计算性能也是慢慢的提升的。在没有足够的理论的支撑下，我们无法预测 Transformer 能在计算机视觉掀起多大的波澜，还是要看硬件性能的提升效率，新范式的提出时间，以及卷积的反抗。

本文介绍的 ConvNeXt 就是 CNN 的一个很好的反击，它在保持 CNN 结构的基础之上，通过“借鉴”Swin Transformer 等方法的调参技巧，证明了 Transformer 在视觉领域上的突出表现并不是 Transformer 在理论上更适合图像数据，而只是近年来的诸多的提升准确率的小 Trick 带来的附加作用，就像前不久的 ResNet-Timm，它也能在仅仅修改调参技巧也能把残差网络的准确率提升到 80%+。

我不想就此否定 Transformer 在视觉方向的突出贡献，它将 Transformer 在 NLP 领域的进展带来的经验技巧融入到 CV 领域，对 CV 领域的发展也是非常具有促进意义的，而且对目前的多模态深度学习的发展也非常重要。我们也期待将来有一天，我们能找到一个比 CNN 和 Transformer 都更适合于图像任务的结构范式。

Reference

[1] Liu, Zhuang, et al. "A ConvNet for the 2020s." arXiv preprint arXiv:2201.03545 (2022).
[2] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.
[3] Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." arXiv preprint arXiv:2103.14030 (2021).
[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[5] Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[6] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[7] Krizhevsky, A. , I. Sutskever , and G. Hinton . "ImageNet Classification with Deep Convolutional Neural Networks." Advances in neural information processing systems 25.2(2012).
[8] Huang, G. , et al. "Densely Connected Convolutional Networks." IEEE Computer Society IEEE Computer Society, 2016.