集成 CNNs 和 Transformer 进行跨领域图像分类

林跃
大连理工大学 人工智能学院
辽宁省大连市 116024
iamtsuki@mail.dlut.edu.cn

摘要

我训练了2个标准的卷积神经网络和1个标准的Transformer，用这3个模型分别进行知识蒸馏得到了4个轻量级模型，这些轻量级模型的预测结果在计算机视觉课程图像分类竞赛中达到了69.54%的分数，获得了第四名。我使用了多种数据增强技术，从而让模型适应各种不同风格的图像。代码和模型都在https://github.com/Yue-0/DutImageClassification。

1 介绍

自从AlexNet[1]在图像识别领域取得成功后，卷积神经网络就成为了解决计算机视觉问题的主流方法，并飞速发展。随着ViT[2]把Transformer结构[3]引入到计算机视觉领域中，基于Transformer的方法[4]逐渐取代了卷积神经网络。然而，Transformer需要大量的训练数据，且ConvNeXt[5]等最新的卷积神经网络仍有与Transformer竞争的能力。因此，卷积神经网络和Transformer都是目前计算机视觉领域主流的方法。

为了充分发挥卷积神经网络和Transformer各自的优势，我在计算机视觉课程的图像分类挑战赛[6]中训练了两个卷积神经网络和一个Transformer，它们分别是ResNet[7]、ConvNeXt[8]和SwinTransformerV2[9]。由于挑战赛限制了模型的参数量和计算量，我对这些标准模型进行知识蒸馏，得到了4个符合挑战赛要求的轻量级模型，它们分别具有ResNet、RepVGG[10]、ConvNeXt和SwinTransformer的结构。大部分模型都能在挑战赛的测试集上取得较高的精度，如图1所示。最后，我把这些轻量级型模型的结果进行集成，在挑战赛中取得了69.54%的分数，比baseline高了11.885%。

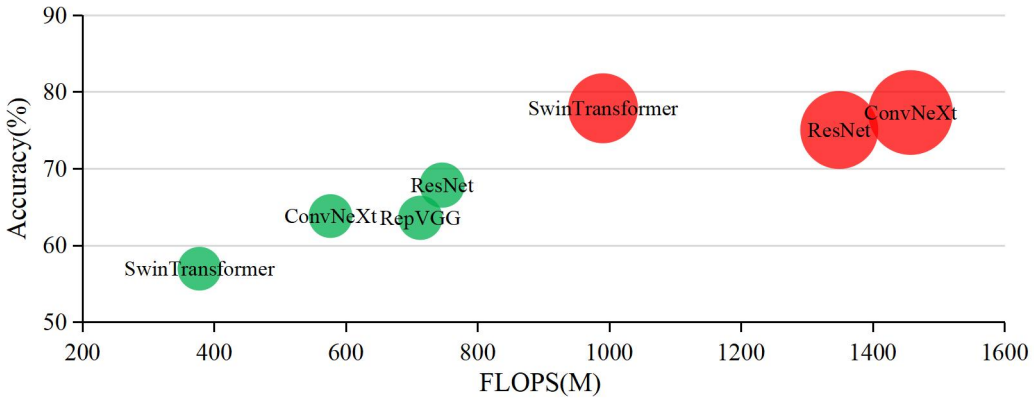

图 1：各种模型在测试集上的准确率。红色的是标准模型，绿色的是满足挑战赛要求的轻量级模型。横坐标表示模型处理大小为 3×128×128 的图像所需的计算量，纵坐标表示模型在前 40%的测试集上的准确率，气泡的大小表示模型的参数量。

2 相关工作

使用卷积神经网络或Transformer进行图像分类是计算机视觉领域最基础的任务之一。最近，CoCa[11]和PaLI[12]等模型都在图像分类任务上取得了非常优秀的结果。然而，单一的模型往往不能取得最好的结果，因为模型可能陷入局部最优状态，并且单个模型的表达能力并不全面。在之前的大部分图像分类比赛中，成绩优秀的团队都使用了集成学习的技巧，如2014年ILSVRC挑战赛的冠军就集成了多个GoogLeNet[13]的结果。为了满足集成学习对模型“好而不同”的要求，我把多个具有不同结构的优秀模型的结果进行集成，从而使得分得到了一定程度的提升。

数据增强是防止神经网络过拟合最有效的方法之一，也是在数据有限的情况下提升神经网络泛化能力的关键手段。大多数经典的数据增强使用随机几何变换、随机颜色变化、随机图像融合等策略，可以很好地防止神经网络过拟合，但是对神经网络泛化能力的提升有限，无法让神经网络适应不同领域的图像。与其他数据增强方法不同，我在它们的基础上使用了随机风格转换，把原始图片以一定的概率转化为其他风格的图像，如动漫图像、素描图像、油画图像等，消融实验表明这些数据增强是我在图像分类挑战赛中取得高分的主要原因之一。

知识蒸馏[14]是模型压缩的一种常用方法，它利用性能更好的大型模型的监督信息来训练小型的轻量化模型。有很多提高知识蒸馏效率和性能的方法，如FitNets[15]和KDGAN[16]等。我使用最基础的知识蒸馏方法得到满足挑战赛要求的轻量级模型。

3 方法

3.1 数据增强

计算机视觉课程图像分类挑战赛的难点在于进行跨领域图像分类，这要求模型具有非常强的泛化能力。挑战赛的训练集和验证集都是使用相机直接拍摄的普通图像。然而，测试集却包含了普通图像、动漫图像、素描图像、油画图像等多种风格的图像。为了提高模型的泛化能力，我通过使用多种数据增强的方式来提高模型的泛化能力，主要包括风格转换、几何变换、颜色变换、图像融合等。

由于训练集图像风格单一，而测试集图像风格多样，为了提升测试集精度，我对训练集图像进行风格变换以适应测试集图像的概率分布。经过粗略统计，测试集图像的风格大致分为普通图像、动漫图像、素描图像和具有梵高风格的油画图像，而训练集图像全部都为普通图像。为了让模型学习到各种风格的图像的特征，我使用AnimeGAN[17]的模型和权重将训练集的图像转换为动漫风格，使用FastNeuralStyle[18]的模型和权重将训练集图像转换为具有梵高风格的油画图像，使用基于图像梯度的方法将训练集的图像转换为素描风格。通过随机应用这三种数据增强方法，可以让模型同时学习不同风格的图像，从而更好地适应测试集图像的分布情况。图2展示了分别应用这三种数据增强策略后得到的训练图像，以及部分测试集图像。在训练阶段，以75%的概率应用三种风格转换中的一种，且应用每种风格转换的概率相同。

为了更好地提高模型的性能，我还使用了MixUp[19]、CutOut[20]、CutMix[21]和随机颜色变换等数据增强方法，在训练时将以50%的概率触发其中一种，且触发每种方法的概率相等。其中颜色变换是通过改变图像的亮度和对比度实现的，此外，还会以一定的概率将变换后的图像灰度化。由于素描图像是灰度图像，所以如果图像被转换为素描风格，将不会触发颜色变换。数据增强的具体细节在附录A中描述。

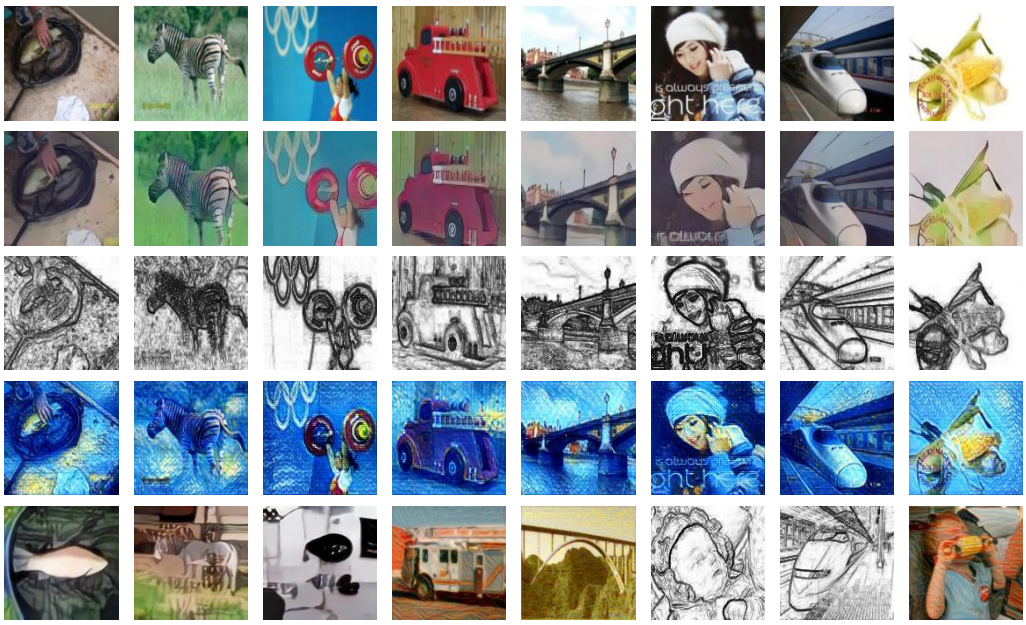

图 2：基于风格变换的数据增强。第一行是原始的训练集图像，第二行是应用动漫风格转换的结果，第三行是应用素描风格转换的结果，第四行是应用梵高风格转换的结果，第五行是部分测试集图像。

3.2 网络结构

我训练了3个具有不同结构的神经网络，并对它们分别进行知识蒸馏，得到了4个轻量级的神经网络。具体来讲，我把ResNet-50、ConvNeXt-Tiny和SwinTransformerV2-T作为教师模型，使用torchvision提供的预训练权重[22]进行初始化，并在挑战赛的数据集上进行微调。这3个模型具有不同的结构，可以进行优势互补，有利于集成学习。然而，这些标准模型的参数量和计算量都较大，不满足挑战赛对模型尺寸的要求，因此，我对这三个模型进行轻量化，并使用知识蒸馏的方法训练轻量化模型。我使用ResNet-50蒸馏得到26层的ResNet和11层的RepVGG，使用ConvNeXt-Tiny蒸馏得到ConvNeXt-Nano，使用SwinTransformerV2-T蒸馏得到嵌入向量维度更低、注意力头数更少的10层的SwinTransformerV2-N。

按照计算机视觉骨干模型的设计惯例，每个模型都具有4个stage，表1列出了标准模型和我设计的轻量级模型的结构区别。我设计的轻量级模型是通过减少标准模型的深度或特征通道数得到的，而RepVGG则是根据VGG-11[23]修改得到的。每个轻量级模型的具体结构都在附录B中详细描述。

表1：标准模型和轻量级模型的结构。*x*b, *y*c表示卷积神经网络的stage共有*x*个block，特征通道数为*y*；*x*d, *y*h表示Transformer的stage的深度为*x*，注意力头的数量为*y*。每个模型的参数量和计算量都是在类别数为50且输入图像尺寸为3×128×128的情况下测得。

	stage1	stage2	stage3	stage4	Params	FLOPs
ResNet-50	3b, 64c	4b, 128c	6b, 256c	3b, 512c	26.31M	1349.23M
ConvNeXt-Tiny	3b, 96c	3b, 192c	9b, 384c	3b, 768c	27.85M	1457.47M
SwinTransformerV2-T	2d, 3h	2d, 6h	6d, 12h	2d, 24h	18.98M	990.55M
ResNet-26	3b, 64c	3b, 64c	3b, 128c	3b, 256c	7.95M	746.26M
RepVGG	1b, 64c	2b, 64c	2b, 128c	2b, 256c	7.62M	713.09M
ConvNeXt-Nano	3b, 64c	3b, 128c	9b, 256c	3b, 256c	7.43M	576.71M
SwinTransformerV2-N	2d, 2h	2d, 4h	4d, 8h	2d, 16h	7.42M	377.7M

我对这些轻量级模型的结果进行集成，得到了集成结果。在集成结果的模型中，每个模型的大小都符合挑战赛的要求。我使用最简单的集成学习的方法进行集成，集成的结果为

$$\boldsymbol{e} = \mathrm{softmax}\left(\sum_{i=1}^{N} \boldsymbol{r}_i\right)$$

式中，$\boldsymbol{r}_i$ 是第 i 个模型输出的结果，$\boldsymbol{X}$ 是输入图像，N 是模型的个数。

3.3 训练

我对标准模型和轻量级模型采用不同的训练方法，所有训练都使用了标签平滑方法[24]，其它配置如表2所示。所有的轻量级模型均在温度系数$T = 1$的条件下由对应的标准模型进行知识蒸馏得到，蒸馏权重$\alpha = 0.7$，使用交叉熵损失函数，目标函数为

$$Loss = \alpha L\left(\mathrm{softmax}\left(\frac{\boldsymbol{s}(\boldsymbol{X})}{T}\right), \mathrm{softmax}\left(\frac{\boldsymbol{t}(\boldsymbol{X})}{T}\right)\right) + (1-\alpha)L\left(\mathrm{softmax}\left(\boldsymbol{f}_s(\boldsymbol{X})\right), \boldsymbol{y}\right)$$

式中，L 表示交叉熵损失函数，$\boldsymbol{s}$ 是学生模型，$\boldsymbol{t}$ 是教师模型，$\boldsymbol{X}$ 是输入图像，$\boldsymbol{y}$ 是经过标签平滑后得到的标签向量，T 是蒸馏温度，α是蒸馏权重。事实上，目标函数的第一项应该使用 KL 散度损失函数，而非交叉熵损失函数，这是我的一个疏忽，但幸运的是，交叉熵损失函数也能正常工作。

表2：各个模型的训练配置。前3行是标准模型，后4行是轻量级模型。标准模型使用了预训练权重，而轻量级模型都是从头开始训练的。warmup是指在第一轮训练时进行线性学习率预热，具体细节在附录C中详细描述。

	base lr	epochs	batch size	lr decay	weight decay	warmup
ResNet-50	0.1	120	256	div 10	1×10^{-4}	×
ConvNeXt-Tiny	0.002	120	256	cosine	1×10^{-8}	×
SwinTransformerV2-T	0.00025	120	224	cosine	1×10^{-8}	×
ResNet-26	0.1	180	256	div 10	1×10^{-4}	×
RepVGG	0.075	180	192	cosine	5×10^{-5}	√
ConvNeXt-Nano	0.002	180	192	cosine	2×10^{-3}	√
SwinTransformerV2-N	0.0004	180	128	cosine	1×10^{-3}	√

为了比较各种优化器和各种学习率策略的区别，我对不同的网络结构使用了不同的优化器和不同的学习率策略，并使它们尽可能与原始论文保持一致。我使用SGD[25]训练ResNet和RepVGG，使用AdamW[26]训练ConvNeXt和SwinTransformer。我并没有使用RMSprop[27]和Adam[28]，具体原因和细节在附录C中描述。在训练ResNet时，我使用固定的学习率进行训练，当模型的验证集精度趋于平稳时，把学习率除以10；在训练其他网络时，使用余弦退火策略[29]。这么做的原因一方面是为了在有限的实验时间内比较两种学习率策略的区别，另一方面是为了和原论文保持一致。

4 实验

4.1 数据集

我使用计算机视觉课程图像分类挑战赛提供的数据集进行实验，没有使用任何额外的数据。数据集分为训练集、验证集和测试集，共50个类别。训练集和验证集分别包含25k和2.5k个图像，每个类别的图像数相同；测试集包含14.5k个对应类别的4个不同领域的图片。实验中的验证集精度是我自己测得的，而测试集精度是通过向竞赛服务器提交结果得到的。

4.2 CNNs和Transformer

我先对3个预训练的标准模型进行微调，为了便于对比，每个标准模型都训练120 epochs。图3显示了各个标准模型在训练过程中的训练集损失值和验证集精度的变化情况。需要说明的是，这里的验证集是经过数据增强后的验证集，即对验证集中的每一张图像都进行3种风格变换，最终验证集的图像数量是原始验证集图像数量的4倍。因为经过数据增强后的验证集的数据分布与测试集更加接近，所以经过数据增强后的验证集精度可以更加有效地反映出模型在测试集上的性能。在后面的结果中，如果没有特殊说明，验证集均为经过数据增强的验证集。

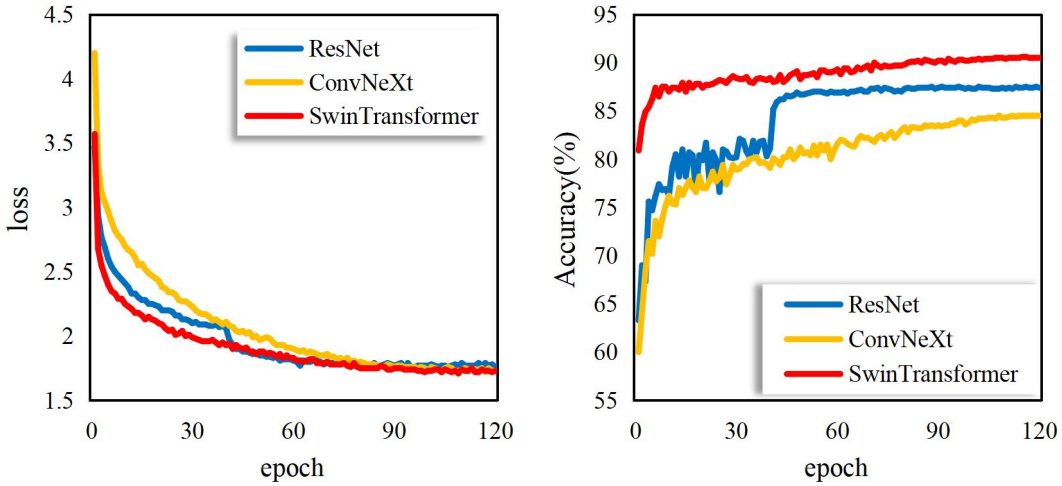

图 3：标准模型的训练过程。左边的图是训练集 loss 值的变化情况，loss 值是该 epoch 所有 loss 值的平均值；右边的图是验证集精度的变化情况。

从图3的结果可见，Transformer的性能明显优于CNNs，这可能得益于Transformer的预训练模型使用了更多的预训练数据，因为在训练轻量级模型时，没有经过预训练的Transformer并没有表现出与很强的性能，如图4所示。

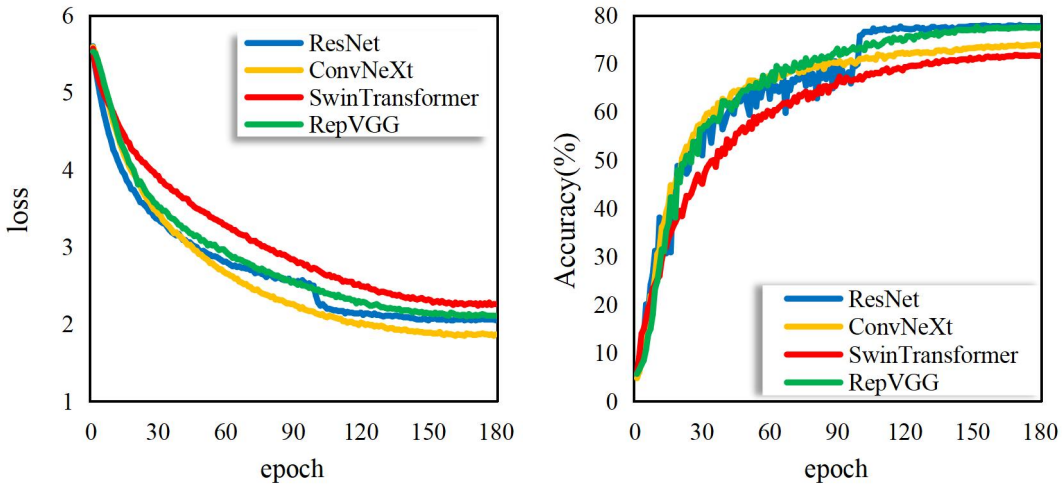

图 4：轻量级模型的训练过程。所有模型都是从头开始训练的，为了便于对比，所有模型都训练了 180 epochs。左边的图是训练集 loss 值的变化情况，loss 值是标签损失和蒸馏损失的加权和，图中画出的 loss 值是该 epoch 所有 loss 值的平均值；右边的图是验证集精度的变化情况。

SwinTransformer的轻量级模型的性能远低于卷积神经网络，这可能是因为Transformer本身不具有卷积的归纳偏置能力导致的。Transformer模型需要更多的训练数据才能表现出优秀的结果，而挑战赛的数据集不足以从头开始训练一个优秀的Transformer模型，因为它包含的数据量太少。另一方面，也可能是因为轻量级的SwinTransformer使用了极低的嵌入维度（64维）、和非常少的head数量（2, 4, 8, 16），在减少参数量和计算量的同时丢失了特征提取能力，导致其性能下降严重。

我分别对3个标准模型和4个轻量级模型的结果进行集成，其中RepVGG被等价转化为VGG后进行测试。标准模型的结果的集成可以在前40%的测试集上达到80.413%的精度，轻量级模型的结果的集成则在前40%的测试集上达到了68.327%的精度，比标准模型的集成结果下降了12.086%。如此严重的精度损失可能是因为我使用的知识蒸馏方法过于基础导致的。由于SwinTransformer的轻量级模型的精度太低，所以在最终提交时，我只使用了三个精度较高的卷积神经网络的结果。最终提交的结果在前40%的测试集上的精度为68.758%，比baseline高10.552%；在后60%的测试集上的精度则达到了69.540%，比baseline高11.885%，但是与第一名仍有5%的差距。然而，与大部分其他选手的方法不同，我的方法并没有进行任何超参数的调试实验，我使用的所有超参数都遵循尽可能与原始论文保持一致的原则，只使用了一套超参数进行了一次实验。

4.3 消融实验

为了证明数据增强、结果集成和知识蒸馏的有效性，我进行了一系列消融实验。

首先，为了证明数据增强的有效性，我分别去除了随机风格变换和随机图像融合及颜色变换，以相同的其它配置训练ResNet-50。为了节约训练资源，我只对ResNet-50进行了实验，并在验证集和前40%的测试集上进行了评估，结果如表4所示。实验结果表明这两种数据增强方法都对模型精度的提升起到了一定的作用，而且随机风格变换起到的作用远高于随机图像融合及颜色变换起到的作用。在不使用随机风格变换的情况下，模型虽然可以在原始验证集上达到92.96%的正确率，但是在测试集上仅有60.34%的正确率，比原始验证集降低了32.62%；而使用随机风格变换后，虽然模型在原始验证集上的精度有所下降，但其测试集上的正确率达到了75.02%，提高了14.68%，而且模型在原始验证集和测试集上的精度差缩小到了15.22%，是不使用随机风格变换的一半以下。而去除随机图像及颜色变换后，模型在原始数据集上的正确率反而有所提升，且在验证集和测试集上的精度较使用全部数据增强的模型下降都不足0.5%，这从另一个角度说明了随机风格变换是我在挑战赛中取得较高分数的主要原因之一。

表4：数据增强的消融实验结果。测试集只包含前40%的数据，验证集是指经过数据增强的验证集，标有*的验证集是原始的验证集。

数据增强	验证集*	验证集	测试集
使用全部数据增强	90.24%	**87.54%**	**75.02%**
去除随机风格变换	**92.96%**	53.74%	60.34%
去除随机图像融合和随机颜色变换	90.40%	87.38%	74.62%

其次，为了证明集成各个模型预测结果的有效性，我对比了单一模型的结果和集成结果的前40%测试集精度，如表5所示。集成的结果在前40%的测试集上达到了80.413%的精度，高于任意一个单一模型的结果的精度。

表5：集成结果的精度提升。所有结果都来自预训练的标准模型。测试集只含前40%的数据。

模型	ResNet	ConvNeXt	SwinTransformer	Ensemble
测试集精度	75.017%	77.293%	77.844%	**80.413%**

然而，如果集成的结果中存在一个性能较差模型的结果，则会影响集成结果的精度，表6列出了集成不同轻量级模型结果的前40%测试集精度。表中数据表明，当集成的结果中含有精度较低的模型（SwinTransformerV2-N）的结果时，集成结果的精度显著低于不含低精度模型结果的精度。最终，仅集成精度较高的三个轻量级卷积神经网络的结果得到的精度优于同时集成四个轻量级模型结果的精度，这也是我最终提交的结果。而将ResNet的结果与精度较低的SwinTransformer的结果集成时，精度反而有所下降。另外，集成精度较低的三个模型的结果后的精度不如单一的ResNet的精度。一个有趣的发现是，四个模型的集成后的精度与仅集成ResNet和RepVGG的精度是相同的。

表6：轻量级模型结果集成的消融实验结果。使用的模型均为从头开始训练的轻量级模型。测试集只包含前40%的数据。由于向服务器提交结果的次数有限，只对部分情况进行了实验。

ResNet	RepVGG	ConvNeXt	SwinTransformer	测试集精度
√				67.827%
	√			63.568%
		√		63.793%
			√	56.931%
√	√			68.327%
√			√	63.051%
	√	√	√	65.948%
√	√	√		**68.758%**
√	√	√	√	68.327%

最后，我对知识蒸馏的有效性进行了消融实验。我对比了使用ResNet-50蒸馏和不使用任何教师模型的ResNet-26的训练过程，如图5所示。虽然使用知识蒸馏的模型比不使用知识蒸馏的模型的收敛速度更慢，但最终精度更高。这是因为使用知识蒸馏的模型不仅需要学习标签的监督知识，还需要学习教师模型的监督知识，所以收敛速度较慢，而正是因为使用知识蒸馏的模型学习了更多的知识，所有最终的精度比不使用知识蒸馏的模型高。

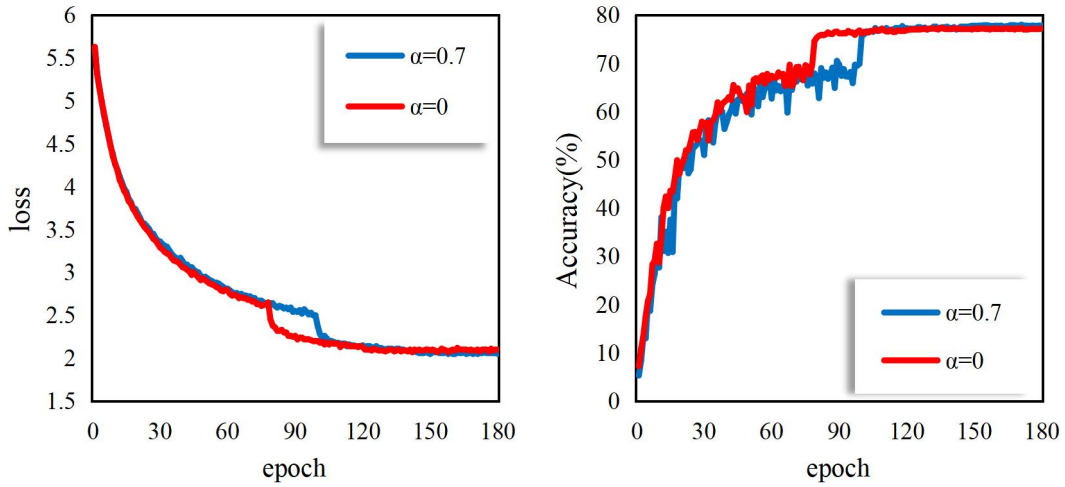

图 5：知识蒸馏的消融实验结果。α表示知识蒸馏的权重，α的值为 0 表示不使用知识蒸馏。对于不使用知识蒸馏的模型（红色），损失值是模型的输出与标签向量的交叉熵损失；对于使用知识蒸馏的模型（蓝色），损失值是模型输出与标签向量的交叉熵损失和模型输出与教师模型输出的交叉熵损失值的加权求和值，其中后者的权重为$\alpha = 0.7$。

5 结论

我对具有不同结构的多种神经网络进行集成，得到了高精度的集成模型，取得了计算机视觉课程图像分类挑战赛的第x名。通过实验证明了数据增强对提高模型表达能力的有效性，验证了集成学习对提高精度的有效性。同时也验证了Transformer等一些先进的模型取得高精度的原因并不是结构设计的新颖性，而是得益于海量的训练数据。在不允许使用额外训练数据和预训练权重的小规模数据集竞赛中，传统的卷积神经网络仍然具有很强的优势。

参考文献

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

[2] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[4] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 10012-10022).

[5] Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., & Xie, S. (2023). ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders. arXiv preprint arXiv:2301.00808.

[6] https://www.kaggle.com/competitions/dlut-cv-project-2023-image-classification

[7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[8] Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11976-11986).

[9] Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., ... & Guo, B. (2022). Swin transformer v2: Scaling up capacity and resolution. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 12009-12019).

[10] Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021). Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13733-13742).

[11] Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., & Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. arXiv preprint arXiv:2205.01917.

[12] Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A. J., Padlewski, P., Salz, D., ... & Soricut, R. (2022). Pali: A jointly-scaled multilingual language-image model. arXiv preprint arXiv:2209.06794.

[13] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[14] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[15] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.

[16] Wang, X., Zhang, R., Sun, Y., & Qi, J. (2018). Kdgan: Knowledge distillation with generative adversarial networks. Advances in neural information processing systems, 31.

[17] Chen, J., Liu, G., & Chen, X. (2020). Animegan: A novel lightweight gan for photo animation. In International symposium on intelligence computation and applications (pp. 242-256). Springer, Singapore.

[18] Holden, D., Habibie, I., Kusajima, I., & Komura, T. (2017). Fast neural style transfer for motion data. IEEE computer graphics and applications, 37(4), 42-49.

[19] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.

[20] DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

[21] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6023-6032).

[22] https://pytorch.org/vision/stable/models.html

[23] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[24] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[25] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147). PMLR.

[26] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

[27] Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, 14(8), 2.

[28] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[29] Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983.

附录 A 数据处理与数据增强

A.1 数据处理

在进行数据增强之前，需要先对数据进行预处理，方便进行数据增强。训练阶段和测试阶段使用不同的数据处理方式。

在训练阶段，需要对图像数据进行随机几何变换，这种变换也可以视为一种数据增强技术。具体来讲，由于训练图像的尺寸不一，需要先将它们调整为相同的尺寸，才能输入到神经网络中进行计算。图像分类任务的神经网络接受的图像尺寸通常为3×224×224，但是挑战赛的训练集图像尺寸（宽×高）的平均值为161×137，远小于224×224，因此，我将图像的短边大小调整为160，而长边被等比例缩放。然后，在得到的图像中随机裁剪出尺寸为128×128的区域，并以50%的概率进行水平翻转，由此得到的图像最为预处理后的图像。这一操作相当于至少把数据集扩充为了原来的2048倍（随机裁剪把数据集至少扩充为了原来的32×32倍，随机水平翻转扩充为原来的2倍）。为了使神经网络更好地学习，我使用了标签平滑的方法，把one-hot向量标签改为

$$\boldsymbol{y}_i = \begin{cases} 1-\varepsilon, & i = c \\ \dfrac{\varepsilon}{N-1}, & i \neq c \end{cases}$$

式中，N为类别总数，$\varepsilon \in (0, 1)$是较小的超参数，c是当前图像的类别。在挑战赛中，$N = 50$，我设置$\varepsilon = 0.1$。

在测试阶段，我使用相同的方法把图像的短边大小调整为160，然后取调整后的图像及其水平镜像的左上、左下、右上、右下和中间共10个大小均为128×128的区域，把这10个图像构成一个mini-batch输入到神经网络中，然后取这10个图像的预测结果的平均值作为对这副图像的预测结果。

A.2 数据增强

我使用的数据增强包括随机风格变换、随机图像融合和随机颜色变化。

随机风格变换包括动漫风格变换、素描风格变换和梵高油画风格变换。我使用两种不同的生成对抗网络对图像进行动漫风格变换和梵高油画风格变换，其中，动漫风格转换的模型权重来自https://github.com/TachibanaYoshino/AnimeGANv2，梵高油画风格转换的模型权重来自https://github.com/jcjohnson/fast-neural-style。我使用基于图像梯度的方法将图像转换为素描风格，设$\boldsymbol{d}_x$和$\boldsymbol{d}_y$分别是图像在水平方向和竖直方向的梯度，gray是把RGB彩色图像映射为灰度图像的函数，则素描图像sketch的生成公式为

$$sketch = \text{gray}\left(255 \cdot \max\left\{0, \min\left\{\left(\boldsymbol{d}_x^2 + \boldsymbol{d}_y^2\right)^{-0.5}, 1\right\}\right\}\right)$$

除了随机风格变换以外，我还使用了MixUp、CutOut、CutMix和随机颜色变换等数据增强方法进一步提高数据的多样性。

MixUp通过一定的比例将两幅图像融合为一副图像进行数据增广。设$\boldsymbol{X}_1$和$\boldsymbol{X}_2$是两幅不同的图像，$\boldsymbol{y}_1$和$\boldsymbol{y}_2$是这两幅图像的one-hot向量标签，$p \in (0, 1)$是一个随机数，经过MixUp后的图像为

$$\boldsymbol{X} = p\boldsymbol{X}_1 + (1-p)\boldsymbol{X}_2$$

转换后图像的标签为

$$\boldsymbol{y} = p\boldsymbol{y}_1 + (1-p)\boldsymbol{y}_2$$

CutOut通过将图像中随机矩形区域的像素置零来进行数据增强，起到类似于Dropout的效果。为了防止CutOut抹去过多的像素值，我设置被抹去的矩形区域的长和宽都不超过图像大小的三分之一。

CutMix通过将图像中随机矩形区域的像素变为训练集中其他图像对应区域的像素值来进行数据增强，同时图像的one-hot向量标签也要变为两幅图像的融合标签。设图像$\boldsymbol{X}_1$的高和宽分别为H和W，$\boldsymbol{X}_1$的一个宽为w、高为h的矩形区域被图像$\boldsymbol{X}_2$的对应区域替换，则$\boldsymbol{X}_1$在替换后的图像中所占的比例为

$$p = 1 - \frac{wh}{WH}$$

若 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 的 one-hot 向量标签分别为 $\boldsymbol{y}_1$ 和 $\boldsymbol{y}_2$，则融合后的标签为

$$\boldsymbol{y} = p\boldsymbol{y}_1 + (1-p)\boldsymbol{y}_2$$

考虑到图像风格的一致性，$\boldsymbol{X}_1$和$\boldsymbol{X}_2$必须是由同一个风格变换函数（或不经过任何风格变换）得到的，即$\boldsymbol{X}_1$和$\boldsymbol{X}_2$的风格是相同的。

事实上，CutOut可以视为CutMix的特例，因为CutOut是将图像与像素全为0且标签相同的图像进行CutMix的结果。因此，我把MixUp、CutOut和CutMix统称为图像融合方法。
如果图像没有进行随机图像融合，则有概率进行随机颜色变换。随机颜色变换是通过改变图像的亮度和对比度实现的，基于 torchvision 提供的 ColorJitter 实现，亮度和对比度的偏移程度是取自参数为(0.9, 1.1)的均匀分布的两个随机数。如果图像不是素描风格的，还会以 10% 的概率对其进行灰度化。下面的伪代码给出了完整的数据增强逻辑和步骤。

算法：*data_augment*
输入：经过数据处理后的图像 $\boldsymbol{X}$, 对应的 one-hot 向量标签 $\boldsymbol{y}$, 训练集 D
输出：经过数据增强后的图像 $\boldsymbol{X}'$, 对应的向量标签 $\boldsymbol{y}'$

---

style ← 从 *Original*、*Anime*、*Sketch* 和 *Van* 中等概率随机选择一个
$\boldsymbol{X}$ ← *convert*($\boldsymbol{X}$, *style*) # 风格转换
a ← 从 *True* 和 *False* 中等概率随机选择一个
if *a* **do**
    a ← 从 *MixUp*、*CutOut*、*CutMix* 和 *Color* 中随机选择一个
    if *a* = *Color* **do**
        $\boldsymbol{X}'$ ← *color_jit*($\boldsymbol{X}$, 0.1, 0.1), $\boldsymbol{y}'$ ← $\boldsymbol{y}$
    else
        rect ← 随机选择一块宽和高均不大于 $\boldsymbol{X}$ 的宽和高的三分之一的矩形区域
        if *a* = *CutOut* **do**
            $\boldsymbol{X}_0$ ← $\boldsymbol{O}$, $\boldsymbol{y}_0$ ← $\boldsymbol{y}$
        else
            $\boldsymbol{X}_0$, $\boldsymbol{y}_0$ ← 从 D 中随机采样
            $\boldsymbol{X}_0$ ← *convert*($\boldsymbol{X}_0$, *style*)
        end
        $\boldsymbol{X}'$, $\boldsymbol{y}'$ ← *fusion*($\boldsymbol{X}$, $\boldsymbol{y}$, $\boldsymbol{X}_0$, $\boldsymbol{y}_0$, *rect*, *a*) # 图像融合
    end
else
    $\boldsymbol{X}'$ ← $\boldsymbol{X}$, $\boldsymbol{y}'$ ← $\boldsymbol{y}$
end

附录 B 网络结构

我设计了4个轻量级模型，它们分别具有ResNet、RepVGG、ConvNeXt和SwinTransformer的结构，每个轻量级模型的参数量和计算量都符合挑战赛的要求。

我设计的轻量级ResNet共有26层，称为ResNet-26。网络结构与标准的ResNet十分相似，但是深度和特征通道数不同。网络共包含1个输入层、4个ResBlocks和1个线性层，每个ResBlocks包含3个ResBlock，每个ResBlock都由若干个卷积层和shortcut构成，如图B.1所示。

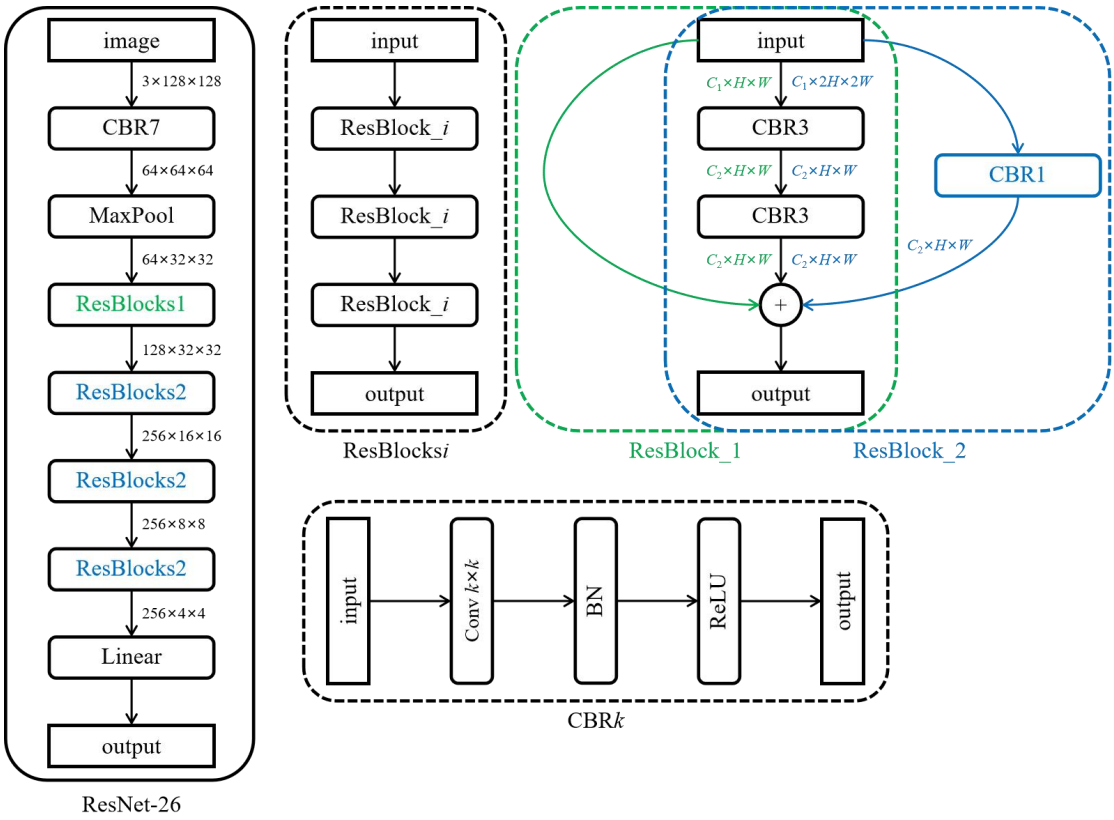

图 B.1：ResNet-26 网络结构图。左侧实线框中的是 ResNet-26，由若干个小组件构成，各个小组件的具体结构绘制在右侧的虚线框中。

RepVGG的结构如图B.2所示。网络包含8个RepBlock和1个多层感知机，多层感知机是由3个全连接层构成的，而每个RepBlock都可以通过卷积重参数化的形式等价转换为1个卷积层。因此，RepVGG可以被等价地转换为具有11层（8个卷积层和3个全连接层）的VGG网络。在RepBlock中，每个卷积层都是没有偏置的。在重参数化时，先将RepBlock中的虚线连接（如果存在）添加一个1×1的卷积层，卷积层的参数为单位矩阵，这并不会改变网络计算的结果，因为

$$\boldsymbol{X} = \boldsymbol{E} * \boldsymbol{X}$$

式中，$\boldsymbol{E}$是单位矩阵，$*$是卷积运算。于是，RepBlock中的各路结构都是先卷积再标准化的结构。卷积层和BN层可以等价地转化为一个带偏置的卷积层，若原卷积层的权重为$\boldsymbol{W}_0$，BN层的缩放参数、偏置参数、均值和标准差分别为$\boldsymbol{\gamma}$、$\boldsymbol{\beta}$、$\boldsymbol{\mu}$和$\boldsymbol{\sigma}$，则转化后的卷积层的权重$\boldsymbol{W}$和偏置$\boldsymbol{b}$分别为

$$\begin{cases} \boldsymbol{W} = \dfrac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}} \boldsymbol{W}_0 \\ \boldsymbol{b} = \boldsymbol{\beta} - \boldsymbol{\mu} \dfrac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}} \end{cases}$$

从而，RepBlock中的三路（或二路，事实上，若为二路结构，则可令第三路卷积的权重和偏置都为零，从而视为三路结构，在后面的讨论中，均按三路结构讨论，因为二路结构是三路结构的特例）结构都变为了一个卷积层。而1×1的卷积层可以视为特殊的3×3卷积层，即最外侧参数都为0的3×3卷积层。所以，RepBlock中的三路结构都是3×3的卷积层。根据卷积运算的分配律，有

$$\sum_{i=1}^{n}(\boldsymbol{W}_i * \boldsymbol{X} + \boldsymbol{b}_i) = \left(\sum_{i=1}^{n}\boldsymbol{W}_i\right) * \boldsymbol{X} + \sum_{i=1}^{n}\boldsymbol{b}_i$$

令

$$\begin{cases} \boldsymbol{W}' = \sum_{i=1}^{3}\boldsymbol{W}_i = \sum_{i=1}^{3}\frac{\boldsymbol{\gamma}_i}{\boldsymbol{\sigma}_i}(\boldsymbol{W}_i)_0 \\ \boldsymbol{b}' = \sum_{i=1}^{3}\boldsymbol{b}_i = \sum_{i=1}^{3}\left(\boldsymbol{\beta}_i - \boldsymbol{\mu}_i\frac{\boldsymbol{\gamma}_i}{\boldsymbol{\sigma}_i}\right) \end{cases}$$

则RepBlock被等价转化为了一个权重为$\boldsymbol{W}'$、偏置为$\boldsymbol{b}'$的3×3的卷积层。在转化前，RepVGG的FLOPs为807.59M，不符合挑战赛的要求，但是重参数化操作把BN层融合进了卷积层，并把多个卷积层合并在了一起，使模型的FLOPs下降到了713.09M，且没有任何精度损失。

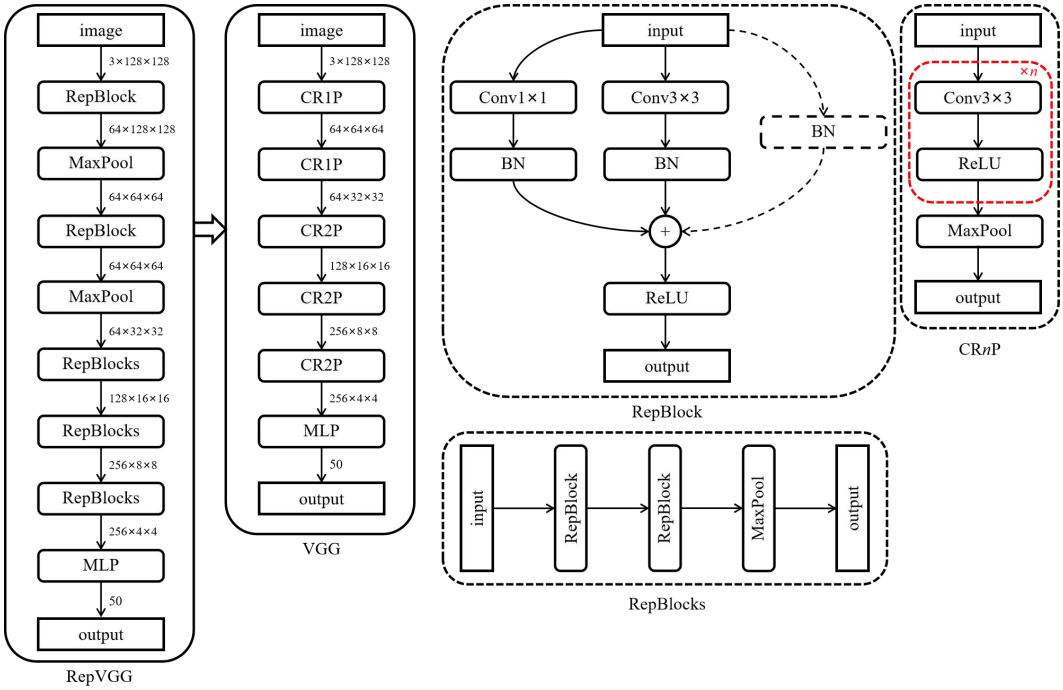

图 B.2：RepVGG 网络结构图。左侧实线框中的是 RepVGG 和由其等价转化而来的 VGG，它们都由若干个小组件构成，各个小组件的具体结构绘制在右侧的虚线框中。RepBlock 中的虚线连接表示只有当 input 的通道数和 output 的通道数相同时才存在。MLP 共有 3 层，两个隐藏层的维度均为 1024。

我设计的轻量级ConvNeXt网络称为ConvNeXt-Nano，它的结构与ConvNeXt-Tiny完全相同，只是特征的通道数发生了变化，各层特征的通道数由96、192、384和768下降为了64、128、256和256，网络结构如图B.3所示。

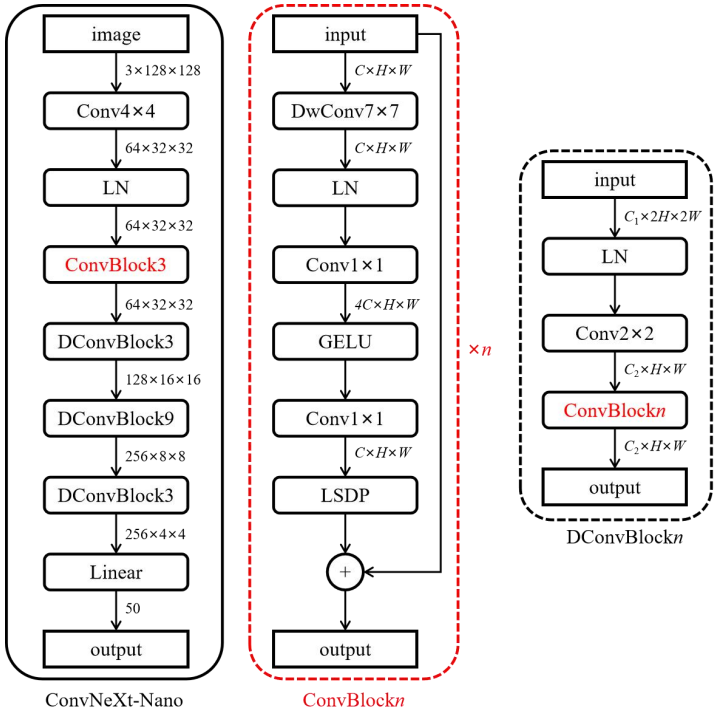

图 B.3：ConvNeXt-Nano 网络结构图。左侧实线框中的是 ConvNeXt-Nano，右侧虚线框中的是构成它的一些组件。LSDP 表示 LayerScale 和 DropPath，DwConv 表示深度可分离卷积，LN 表示 LaterNorm。

我设计的轻量级Transformer称为SwinTransformerV2-N，它的结构与SwinTransformerV2-T类似，但有三个不同之处。第一是嵌入特征向量的维度由96维降低到了64维；第二是深度减少了2层，由12层减少为10层；第三是注意力头的数量变为了原来的三分之二，即由3、6、12和24个变为了2、4、8和16个。图B.4示意了我设计的轻量级SwinTransformerV2-N的结构。处理大小为3×224×224的图像时，SwinTransformerV2-N的计算量仅为377.7M FLOPs，仅为ResNet-26的一半，这表明SwinTransformerV2-N可能存在过度轻量化的问题，但它的参数量仍有7.42M，与其他三个模型相当。

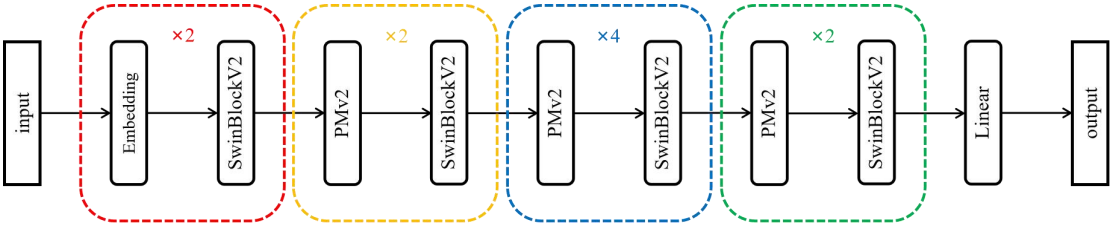

图 B.4：SwinTransformerV2-N 的结构。SwinBlockV2 表示 SwinTransformerBlockV2，PMv2 表示 PatchMergingV2，这两者的详细结构与原论文完全一致。

这四个模型具有完全不同的结构，只要每个模型都能在测试集上取得较高的精度，就能满足集成学习对模型“好而不同”的要求，从而可以对这些模型进行集成。

附录 C 优化器和学习率

C.1 优化器

在深度学习中，常用的优化器有带动量的SGD、RMSProp、Adam和AdamW等，每个优化器都有不同的优化算法，但每个优化算法都有统一的算法框架。

算法：*optimization*
输入：待优化的参数 $\boldsymbol{W}_0$，目标函数 L，学习率α，迭代次数 T
输出：优化后的参数 $\boldsymbol{W}_t$

1. 初始化一阶动量 $\boldsymbol{m}_0$ 和二阶动量 v_0
2. **for** t = 1, 2, ..., T **do**
3. 计算梯度：$\boldsymbol{g}_t \leftarrow \nabla L(\boldsymbol{W}_t)$
4. 更新一阶动量 $\boldsymbol{m}_t$ 和二阶动量 $\boldsymbol{v}_t$
5. 计算下降反方向：$\boldsymbol{d}_t \leftarrow \boldsymbol{m}_t$
6. 更新参数：$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \alpha v_t^{-0.5}\boldsymbol{d}_t$
7. **end**

各个优化算法的不同之处在于一阶动量和二阶动量的更新公式的不同。

在动量SGD中，一阶动量被初始化为$\boldsymbol{m}_0 = \boldsymbol{0}$，二阶动量恒为1，$\boldsymbol{m}_t$的更新公式为

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + (1-\beta)\boldsymbol{g}_t$$

式中，β是超参数，在我的实验中，取$\beta = 0.9$。带动量的SGD优化器早在1986年就被提出，然而，它至今仍是用于优化卷积神经网络参数最主流的优化器。我使用SGD作为ResNet和RepVGG的优化器。

RMSProp使用二阶动量对学习率进行自适应调整，一阶动量恒等于梯度，二阶动量被初始化为$v_0 = 0$，更新公式为

$$v_t = \gamma v_{t-1} + (1-\gamma)\boldsymbol{g}_t^2$$

式中，γ是超参数。RMSProp虽然做到了自适应调整学习率，但由于没有使用一阶动量的信息，所以它并不是一个常用的优化器。Adam集成了SGD的一阶动量和RMSProp的二阶动量，更新公式为

$$\begin{cases} \boldsymbol{m}_t = \dfrac{\beta \boldsymbol{m}_{t-1} + (1-\beta)\boldsymbol{g}_t}{1-\beta^t} \\ v_t = \dfrac{\gamma v_{t-1} + (1-\gamma)\boldsymbol{g}_t^2}{1-\gamma^t} \end{cases}$$

Adam是自然语言处理领域中最常用的优化器之一，ViT也使用了Adam优化算法。然而，Adam的weight decay不能很好地用来实现L2正则化，而AdamW解决了这一问题。现在几乎所有的先进模型都是用AdamW作为优化器，如ConvNeXt和SwinTransformer，因为这种优化器不仅集成了SGD和RMSProp，还进行了相应的优化。我在实验中也使用AdamW作为ConvNeXt和SwinTransformer的优化器。

在我的实验中发现，使用SGD优化器时往往需要较大的基础学习率，而AdamW优化器对基础学习率的要求并不苛刻。我在训练使用AdamW的模型时使用了较小的基础学习率，在训练使用SGD的模型时使用了较大的基础学习率，因为SGD对小学习率的收敛速度较慢。

C.2 学习率

我使用了多种不同的学习率策略进行实验，主要有线性预热、余弦退火等。

线性预热是让学习率从零开始线性增长，直到达到基础学习率的值。设线性预热的步数为T，基础学习率为α_T，则第t步的学习率为

$$\alpha_t = \frac{t}{T}\alpha_T$$

按照经验，使用线性预热有助于模型更快收敛。在我的实验中，训练ResNet-26时没有使用线性预热，训练其他轻量级模型时均使用了1个epoch的线性预热，但是并没有发现线性预热对模型收敛有较明显的影响。

余弦退火是目前深度学习中最常用的学习率衰减策略。设模型训练的总迭代轮数为T，基础学习率为α，一种简单的余弦退火方式是将第t轮迭代的学习率设置为

$$\alpha_t = \frac{\alpha}{2} + \frac{\alpha}{2}\cos\frac{\pi t}{T}$$

由于余弦函数在区间$(0, \pi)$上是单调递减的函数，所以学习率的值会从α一直衰减到0。为了防止最后几轮训练时的学习率过低，可以将上面的公式修正为

$$\alpha_t = \varepsilon + \frac{\alpha - \varepsilon}{2} + \frac{\alpha - \varepsilon}{2}\cos\frac{\pi t}{T}$$

或

$$\alpha_t = \frac{\alpha}{2} + \frac{\alpha}{2}\cos\frac{\pi t}{(1+\varepsilon)T}$$

式中，ε是一个较小的正数。然而，我并没有使用这两种修正公式，这是我的一个失误，导致我在训练的轻量级的RepVGG、ConvNeXt和SwinTransformer时，最后几个epoch的学习率几乎为零。图C给出了训练ResNet-26和RepVGG时学习率的变化情况，训练其他两个轻量级模型时学习率的变化情况与RepVGG相同。

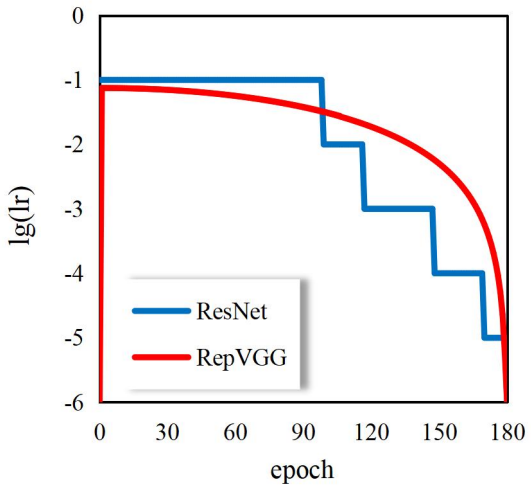

图 C：学习率变化情况，注意纵坐标是学习率的以 10 为底的对数值。ResNet 采用了基于验证集精度的学习率调整策略，当验证集精度连续 10 个 epochs 都不再升高时，学习率就减少为原来的十分之一；RepVGG 的学习率采用余弦退火策略，并且在第一个 epoch 使用线性预热。ResNet 的基础学习率为 0.1，RepVGG 的基础学习率为 0.075。