ViT 在小型数据集上的性能表现探究

姓名: 杨炫锟、余俊洁
学号: 523030910248、523030910244

作业任务: 基于 Vision Transformer 的图像分类研究
组别: 40 组

目录

1 项目概述

本研究聚焦于 **Vision Transformer (ViT)** 在图像分类任务中的应用，特别是在 **CIFAR-10** 数据集上的性能表现。图像分类是计算机视觉的核心任务，传统卷积神经网络（如 **ResNet**）表现优异，而 **Transformer** 模型在自然语言处理领域的成功启发了其在视觉任务中的探索 (Dosovitskiy et al., 2021). **ViT** 通过将图像分割为 patch 并输入 **Transformer** 编码器进行特征提取，但在小数据集上因缺乏局部特征归纳偏差而性能不稳定。本项目目标包括：复现 **ViT**，并考察其在 CIFAR-10 上的性能，设计 **ViT** 与 **ResNet** 的混合模型，分析超参数影响，引入数据增强策略以提升性能。

截至目前，我们已实现 **ViT** 基础模型与混合模型，完成了超参数调优、数据增强实验及可视化分析，实验结果显示最优模型 `Top-1 Accuracy` 为 92.54%，`Top-5 Accuracy` 为 99.57%。仍需进一步优化混合模型参数组合与训练效率。

2 背景

图像分类是计算机视觉的基础任务，传统 **CNN**（如 **ResNet**）已在该领域取得了显著成功。近年来，**Transformer** 模型在自然语言处理中的突破启发了其在视觉任务中的应用 (Dosovitskiy et al., 2021). 然而，基于 **Transformer** 架构的视觉模型 **ViT** 对训练数据量十分敏感，且缺乏局部特征的归纳偏差，在小数据集（如 **CIFAR-10**）上性能并不稳定。

因此，我们此次实验的研究目标包括：

- 复现 **ViT**，在 **CIFAR-10** 上对其进行训练，评估其对小数据集的适应性。
- 结合 **ViT** 与 **ResNet**，设计混合模型以提升分类性能。
- 通过一系列消融实验分析关键参数（如 `patch` 大小、嵌入维度、层数）的影响并进行超参数调优。
- 引入数据增强策略，比较其对模型准确率和鲁棒性的影响。

3 ViT 基础模型性能

我们根据 **ViT** 基础模型的架构进行复现，在没有进行任何参数调优，模型正则化，数据增广等方式的前提下，将其在 **CIFAR-10** 上进行训练，并得到其初始性能表现，结果如下图所示：

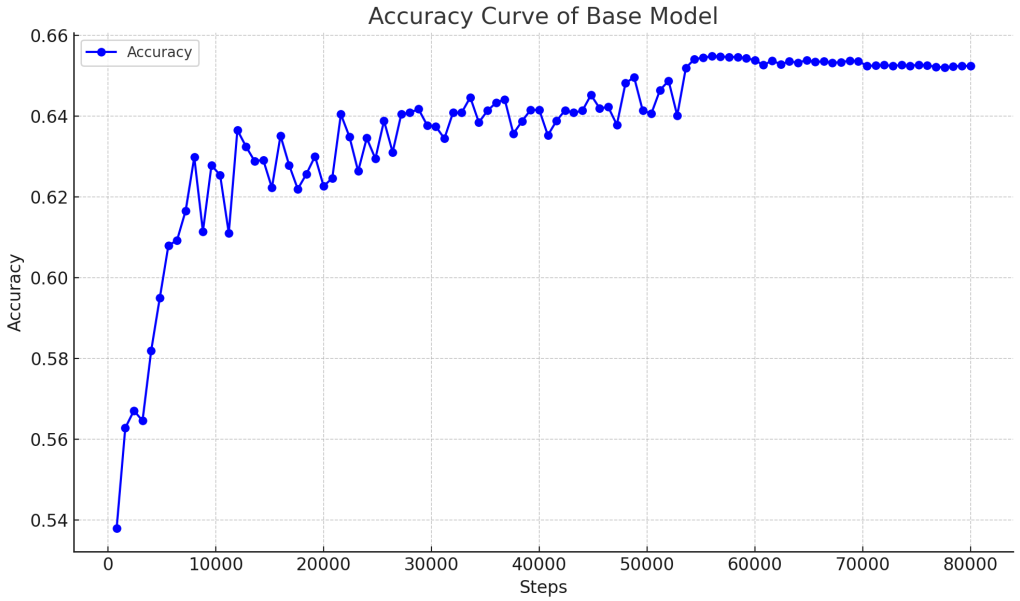


图 1: **ViT** 基础模型性能

实验结果显示，基础模型性能不理想，因此我们需通过改进模型架构，进行超参数调优或数据增强等策略进一步实验优化 **ViT** 性能。

4 模型架构

4.1 基础 ViT 模型架构

在原论文中中，作者使用的图像大小为 $224 * 224$，并选用 `Patch_size` 为 $16 * 16$。由于我们使用的数据集为 **CIFAR-10**，每张图像的大小为 $32 * 32$。我们并没有对其尺寸进行调整，而是直接使用原始图像大小，并设置 `Patch_size` 为 $4 * 4$。

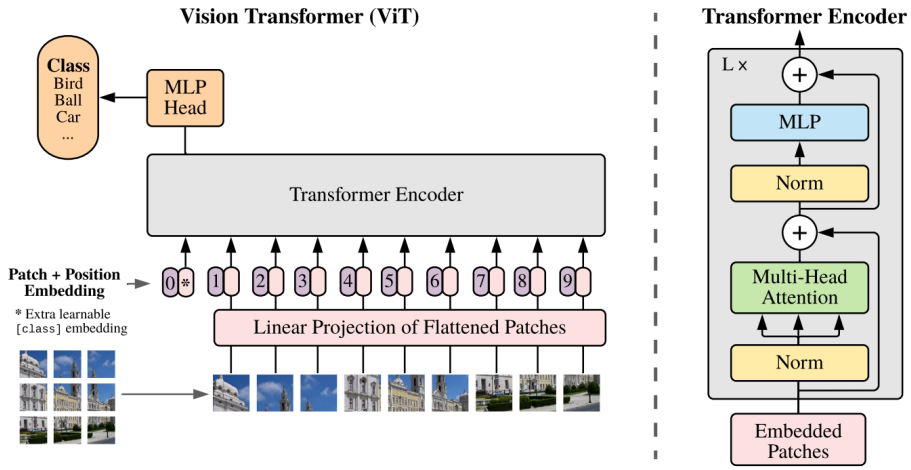


图 2: **ViT** 基础模型架构

我们将这种模型设置命名为 **ViT-Basic**。

4.2 混合模型架构

我们参考 **ViT** 论文中和 **ResNet** 的结合的实现，提出了基于 **CIFAR-10** 数据集的混合模型，并分为以下两种超参数设置：

- 进行三次下采样，最终提取得到的特征图尺寸为 $4*4$，通道数为 256，并设置 **ViT** 中的 `Patch_size` 为 $1*1$
- 进行两次下采样，最终提取得到的特征图尺寸为 $8*8$，通道数为 256，同样设置 `Patch_size` 为 $1*1$

我们将以上这两种模型设置分别命名为 **ViT-Hybrid-1** 和 **ViT-Hybrid-2**。

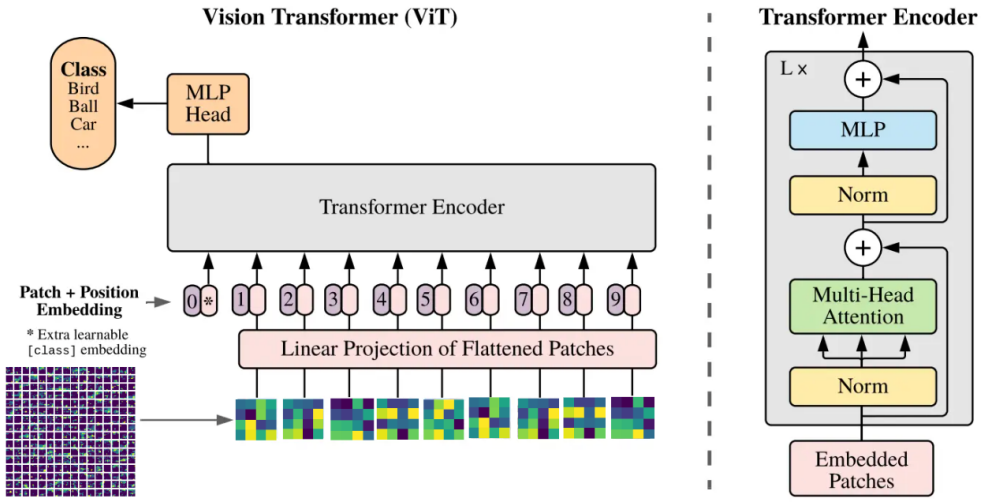


图 3: **ViT** 混合模型架构

5 模型架构研究

在模型架构研究部分，我们主要进行了下面几种模型设置的探究，具体参数设置以及模型参数量可参看附录??。

5.1 Transformer 中自注意力头数的影响

在这一部分我们使用 **ViT-Hybrid-2**，并探索 **Transformer** 中自注意力头数对模型性能的影响，得到图4中的结果。

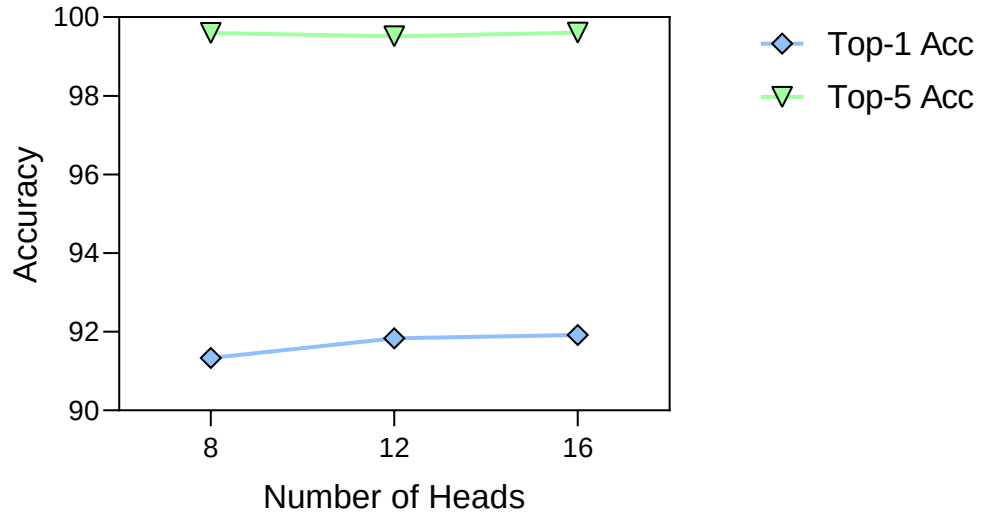


图 4: 自注意力头数的影响

主要观察 **Top-1 Accuracy**，在其余参数设置选用默认设置的前提下，模型的性能随着头数的增多而增强，我们可以理解为随着自注意力头数的增多，模型对输入图片理解更加细致，因此能有更好的性能。头数由 12 上升至 16 的性能提升并没有头数从 8 上升至 12 的性能提升大，在头数为 12 的时候已经有较好的表现。因此，12 头是最佳选择，兼顾性能和效率，而 16 头额外收益不大。

5.2 `Transformer Block` 数量的影响

在这一部分我们仍使用 **ViT-Hybrid-2**，并探索 `Transformer Block` 数量对模型性能的影响，得到图5中的结果。

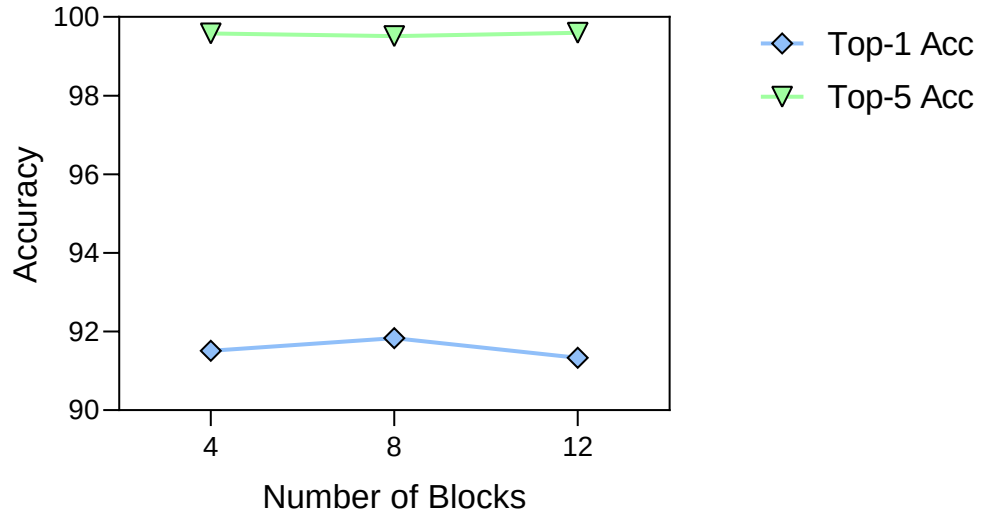


图 5: `Transformer Block` 数量的影响

主要观察 **Top-1 Accuracy**，在其余参数设置选用默认设置的前提下，在模型层数上升的过程中，模型性能表现出现先上升后下降的趋势。层数设置为 4 时，可能是由于模型过于简单导致性能下降；层数设置为 12 时，可能是由于模型过于复杂，训练迭代次数不够，导致优化困难。因此，在同样的默认迭代次数的条件下，设置层数为 8 是最优的。

5.3 Patch_size 的影响

在这一部分，我们使用 **ViT-Basic**，并探索 Patch_size 对模型性能的影响，得到图6中的结果。

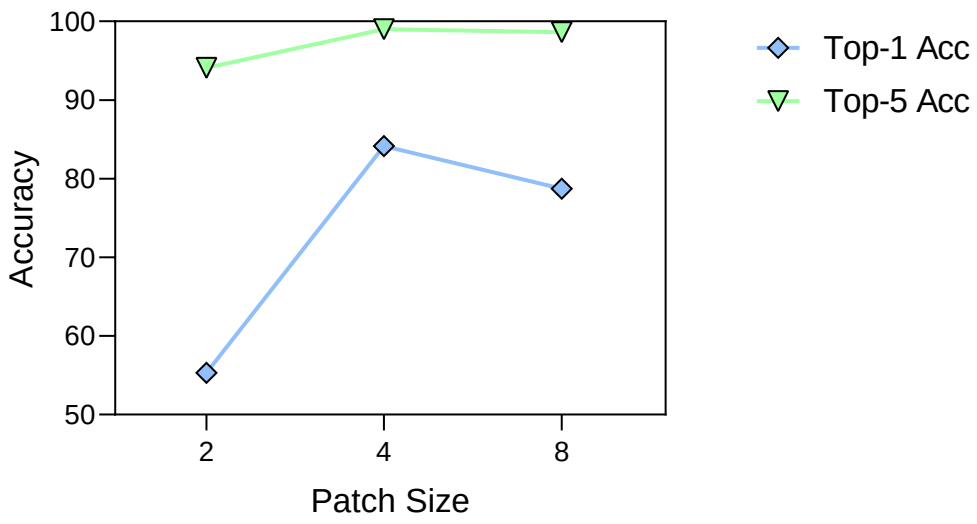


图 6: Patch_size 的影响

主要观察 **Top-1 Accuracy**，在其余参数设置选用默认设置的前提下，Patch_size 设置为 4 时，模型性能表现最佳，表明适中的 Patch_size 能有效捕捉特征。与此同时，Patch_size 设置为 2 和 8，模型性能表现均显著下降，可能的原因是，过小的 Patch_size 可能导致信息不足，过大又容易丢失细节。

5.4 混合模型与原始模型的比较

在这一部分，我们探索了不同混合模型以及原始模型的性能表现，得到图7中的结果。

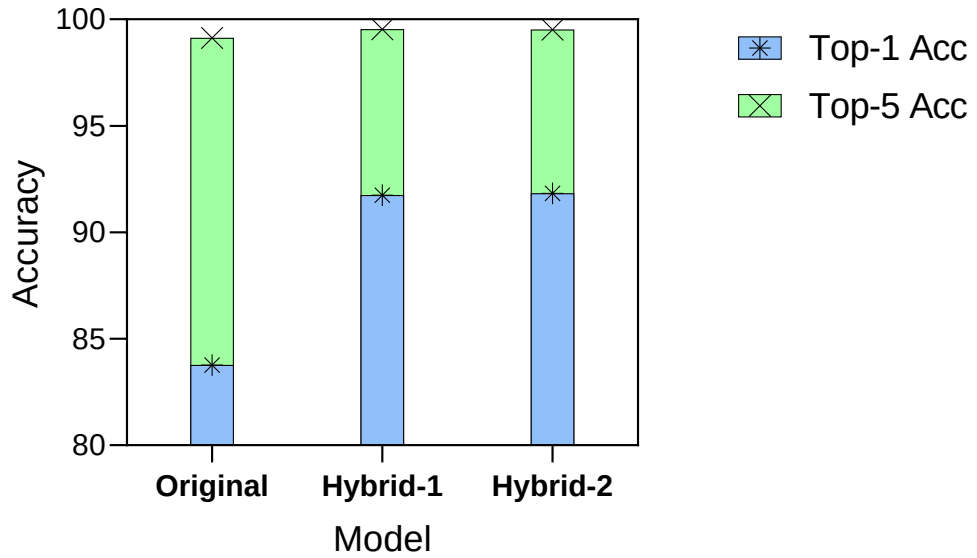


图 7: 混合模型与原始模型的比较

主要观察 **Top-1 Accuracy**，不难看出，在其余参数设置选用默认设置的前提下，**ViT-Hybrid-1** 和 **ViT-Hybrid-2** 的性能均高于 **ViT-Basic**，表明混合模型在单类预测上有显著提升，可能因混合结构增强了特征提

取能力。与此同时，$4*4$ 特征图和 $8*8$ 特征图的性能表现相差很小，表明特征图大小对性能影响有限，可能因通道数一致或者原始图片尺寸过小，信息表达能力相近。同时也说明，**ResNet** 提取的特征可能已捕获足够语义信息，混合模型的 **ViT** 部分进一步优化了分类能力，掩盖了特征图大小差异。

5.5 `Hidden size` 和 `MLP dim` 的影响

在 **ViT** 中，`Hidden size` 和 `MLP dim` 是两个很重要的参数，贯穿于整个 **Transformer** 的始终。在每个 `Transformer Block` 的最后会经历一个简单的单隐藏层 `MLP`，由维度由 `Hidden size` 变换到 `MLP dim` 再变换回 `Hidden size`，因此，下面我们始终保持 `MLP dim` 大于等于 `Hidden size` 设置了 5 组实验，得到图8中的实验结果。

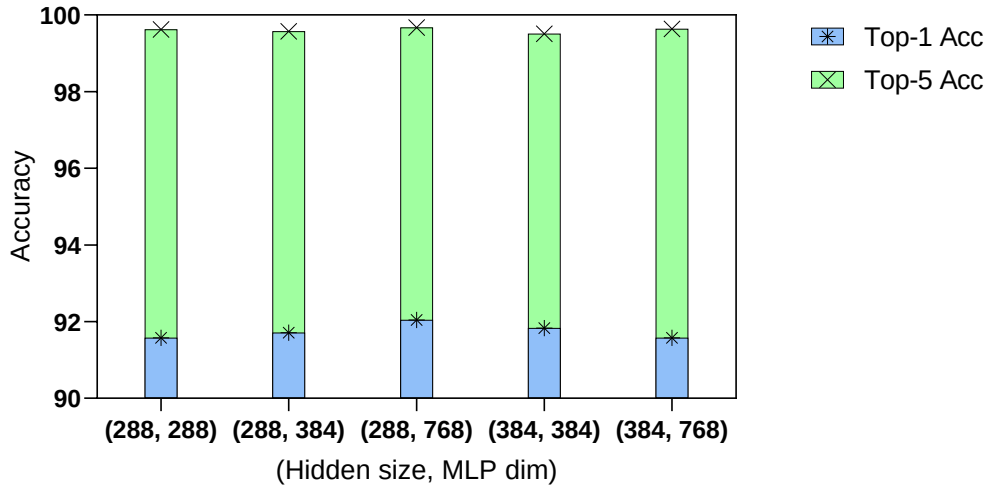


图 8: `Hidden size` 和 `MLP dim` 的影响

主要观察 **Top-1 Accuracy**，在其余参数设置选用默认设置的前提下，五组实验中，`Hidden size` 取 288，`MLP dim` 取 768 时，模型性能最优，`Hidden size` 和 `MLP dim` 的增大会导致模型容量增大，但也会带来优化困难，或者过拟合等问题，况且，我们所使用的数据集很小，模型太复杂并不合适。因此，最优参数为 $(288, 768)$。

6 正则化探究

在这一部分中，我们使用 **ViT-Hybrid-2**，探索了三种基础的正则化方法，以及更进阶的随机深度的方法。

6.1 基本正则化方法

在基本正则化方法中，我们探索了 **Weight Decay**、**Attention Dropout** 以及 **Dropout** 三种方法，并设置不同的参数，得到9中的结果。

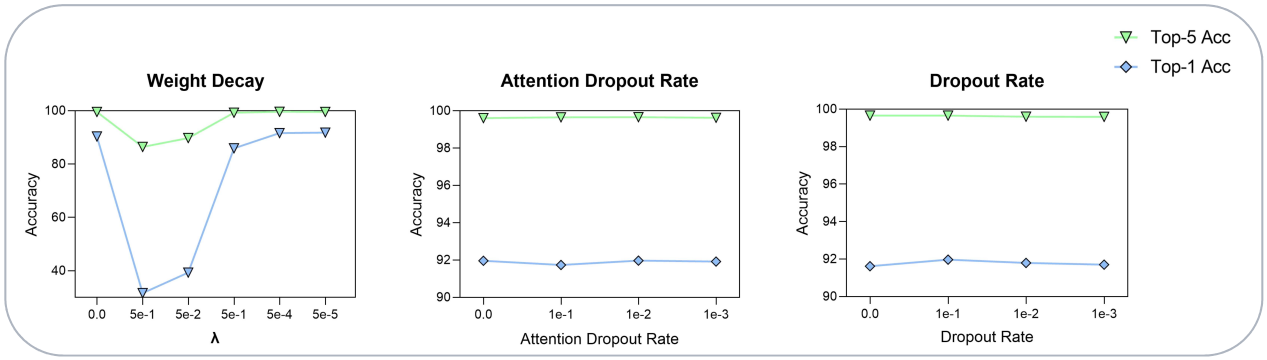


图 9: 三种基本正则化方法

主要观察 **Top-1 Accuracy**，在 **Weight Decay** 方法中，模型性能随 λ 的减小先大幅提升，5e-4 后趋于稳定，基本达到不进行 **Weight Decay** 的水平，可能的原因是较大的 λ 会过度惩罚模型权重，限制模型学习复杂特征，导致欠拟合，此外，较大的 λ 增强 $L2$ 正则化，权重向零靠拢，削弱模型容量，且过大的 λ 可能抑制必要参数调整。与此同时，不难看出，模型在不同 `Attention Rate` 和 `Dropout Rate` 的设置下均有较好的性能表现，体现出了较强的鲁棒性。

6.2 随机深度方法

我们提出的随机深度的方法如图10所示，对于每一个 `Block`，数据有 p 的概率不进入该 `Block`，起到了随机跳过更新某些 `Block` 的作用。不难得到，每一次训练中，模型实际更新的 `Block` 数量的期望为：

$$\mathbb{E}\left[N_{update}\right] = N_{block} \times (1-p) \tag{1}$$

我们针对参数概率 p 进行了多组实验，得到图11所示的结果。

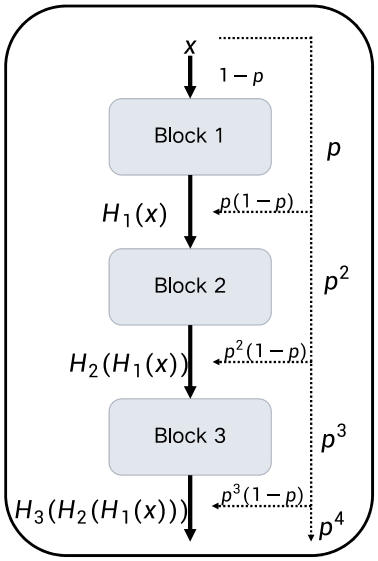


图 10: 随机深度方法

主要观察 **Top-1 Accuracy**，较大的 p 会导致模型性能略微下降，这可能是因为模型实际更新的 `Block` 数量少，且每次更新不平稳。而适宜的 p，如（1e-2）能给模型的性能带来一定提升，这可能是因为模型随机跳过一些 `Block`，对于其余 `Block` 的更新具有一定的辅助作用，在一定程度上能够提升模型的泛化能力。

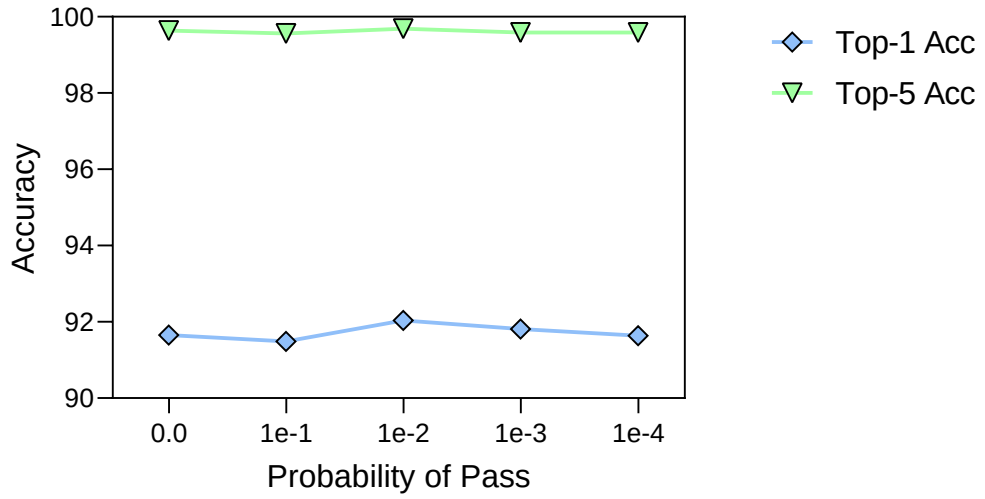


图 11: 随机深度实验结果

7 数据增强

7.1 数据增广方式

在数据增强部分，我们采用了官方数据增强库与自定义增强方式相结合的方法进行研究，并使用消融实验探索最佳的组合方式与参数选择：具体使用的增广方式有：

- AutoAugment：由 Google Brain 在 2019 年提出，针对于 Cifar-10 数据集，Google 最终选择了 25 个最优子策略组成 CIFAR-10 Policy。在训练过程中，模型会从这 25 个子策略中随机选择一个应用于每张图像上，提升样本多样性，同时通过强化学习搜索组合，使增强变换协同作用最大化性能提升。(Cubuk et al., 2019)。
- RandAugment：由 Google Brain 在 2020 年提出，RandAugment 采用了一个统一的随机增强机制：仅使用两个可调超参数 N 和 M，对图像数据进行增强。(Cubuk et al., 2020)。
- 自定义增强：CutMix、MixUp、RandomCropPaste。
  - CutMix: 首先从当前 batch 中随机选两张图像，根据 beta 分布决定一个比例 λ。使用 λ 计算一个区域的尺寸（矩形窗口），将随机图像中的该区域替换到原图上相同位置，最终标签为两个图像标签的加权组合，进行跨图像的数据增强。
  - MixUp: 同样的，首先从一个 batch 中随机选两张图，然后通过 beta 分布采样得到混合因子 λ。按 λ 进行像素级加权混合两张图像。标签也按 λ 进行线性组合，以此进行跨图像的数据增强。
  - RandomCropPaste: 首先根据 beta 分布从原图中随机裁剪出一个小区域，以一定的概率随机水平翻转该区域或整张图像，然后将裁剪出的区域粘贴回原图的另一个位置，粘贴时使用线性加权的融合方式进行数据的自体增强。

由于 RandAugment 的调参更为方便，我们在综合比较 AutoAugment 库和 RandAugment 库之后选择了 RandAugment 库，并通过 ['None','RandAugment']+ ['None','CutMix','MixUp','RandomCropPaste','Batch_Random'] 的组合形式进行数据增广。具体的说，即首先选择是否对数据使用 RandAugment 库进行初步增广，再选择是否使用自定义增广的方式对数据进行进一步增广以及选择进一步增广的话，是选用哪一种自定义增广的方式还是说对于每一个 batch 随机采用一种自定义增广方式。

以下是一些经过我们的数据增广前后的图片的呈现：

Image 1 - Class: cat

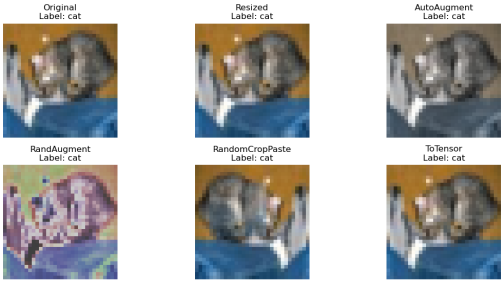


Image 2 - Class: ship

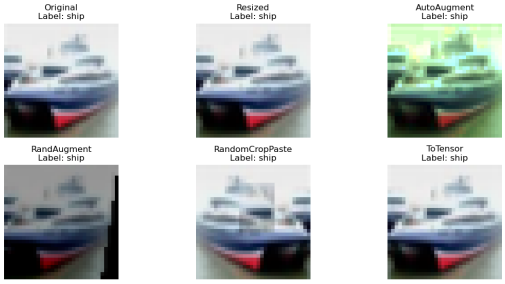


Image 4 - Class: airplane

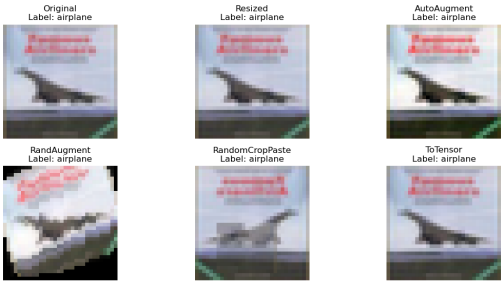
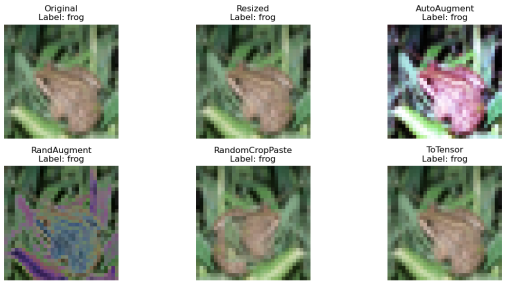
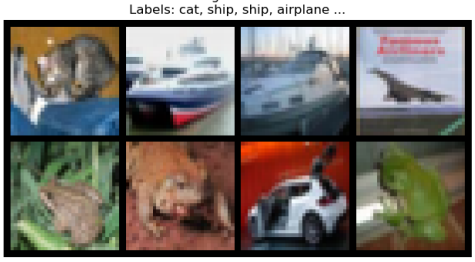


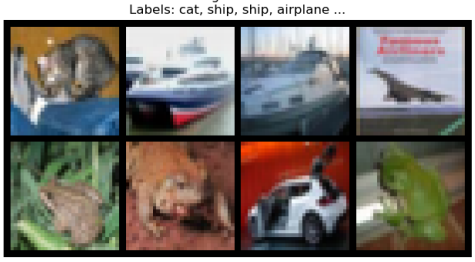
Image 5 - Class: frog



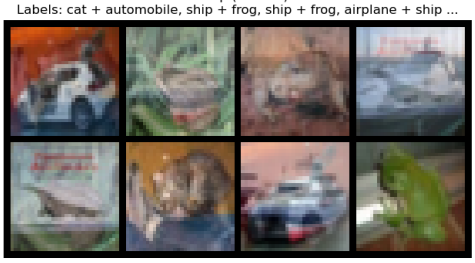
Original Batch
Labels: cat, ship, ship, airplane ...



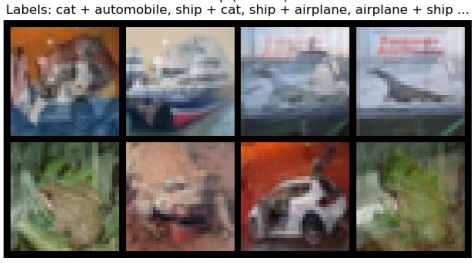
Original Batch
Labels: cat, ship, ship, airplane ...

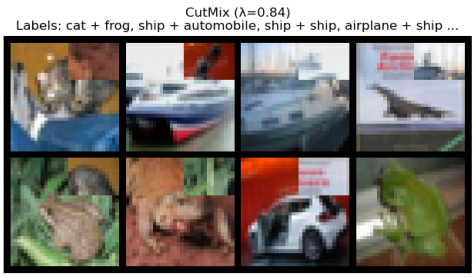


MixUp (λ=0.37)
Labels: cat + automobile, ship + frog, ship + frog, airplane + ship ...



MixUp (λ=0.67)
Labels: cat + automobile, ship + cat, ship + airplane, airplane + ship ...





7.2 RandAugment 效果

我们在控制自定义增广形式以及参数的情况下，进行了针对于 RandAugment 的消融实验，以探究 RandAugment 对于我们模型训练的帮助效果，结果如下图所示：

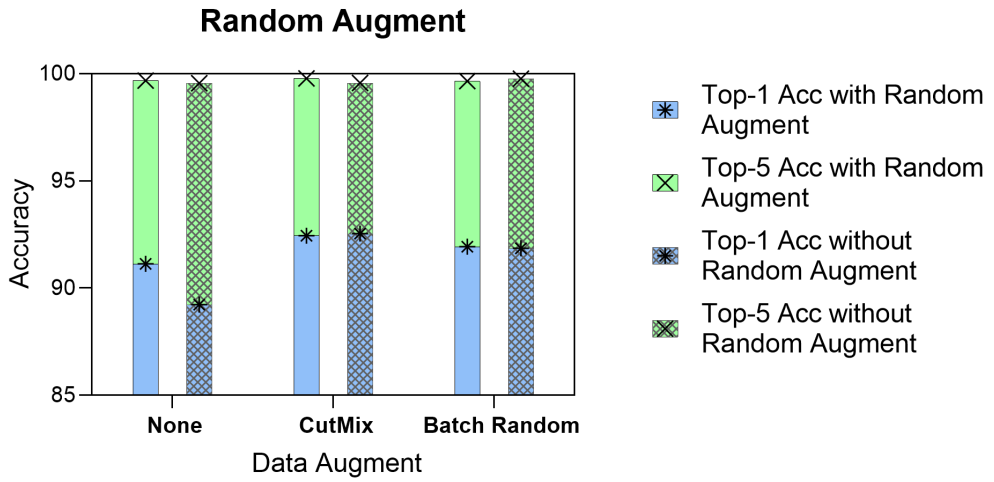


图 12: RandAugment 消融实验结果

消融实验结果显示，尽管 RandAugment 在已有数据增强的情况下效果并不明显，但其依然有较好的性质：

1. 能够使在没有其他复杂数据增强的情况下使模型达到较好的效果。
2. 使用 RandAugment 的增广方式在训练模型时消耗的显存（7000MiB 左右）只需要 CutMix 或 Batch Random 的一半（14000MiB 左右），更为轻量。

7.3 自定义增强效果

同时为了探究比较各种自定义增强方式的效果，在控制未使用 RandAugment 的基础上，对自定义增强方式进行消融实验，得到下图的结果：

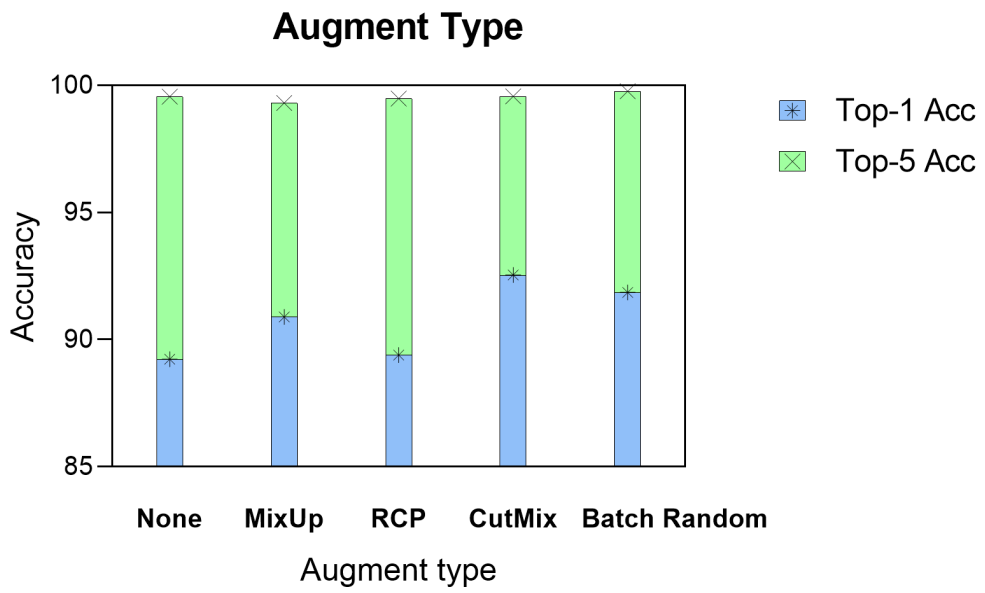


图 13: 自定义增强方式消融实验结果

可见，在选用 CutMix 或者三种自定义增强方式每个 batch 随机的情况下，能让模型得到较好的训练效果。

8 可视化

在这一部分中，我们仿照 **ViT** 原论文，进行了下面的探究以及可视化。

8.1 注意力图

我们选用我们训练得到的最优的模型，参考 (Abnar & Zuidema, 2020)，进行了注意力可视化。

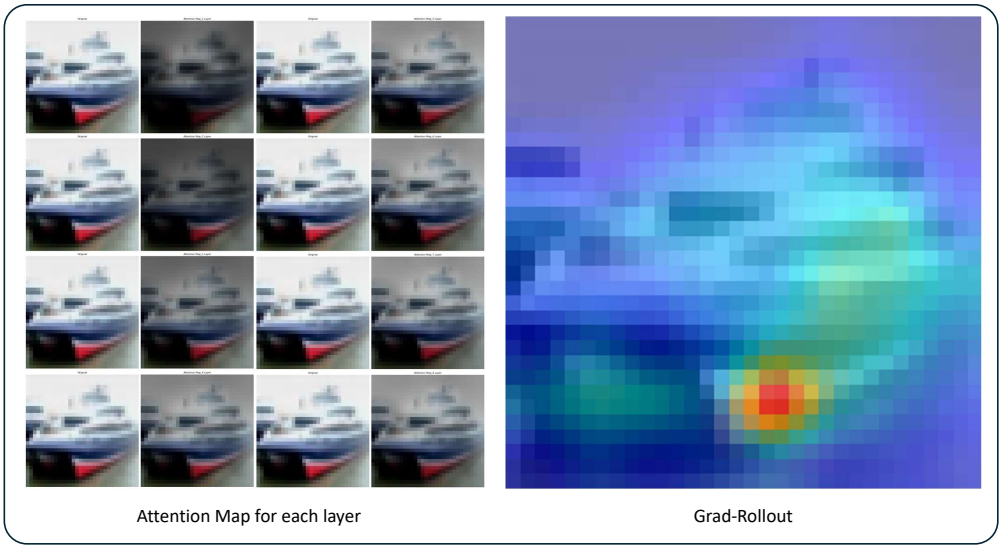


图 14: 注意力图

左图中左侧从上往下依次是 1 至 4 层，，可以看出，随着层数加深，注意力权重较大的区域会有一定变化，但是均能集中于能体现目标类特征的区域。此外，较浅的层会更加关注物体的局部特征，而较深的层会有更加全局的视野，这种表现类似于传统卷积神经网络中的感受野。右图中是注意力的梯度图，不难发现，模型能基本识别出图中物体的轮廓，并在船体等区域有着较高的关注度，表明模型有着较强的甄别能力。

8.2 特征图

为了探究 **ResNet** 对模型性能提升的帮助具体体现在哪些方面，我们使用性能表现最好的 **ViT-Hybrid-2**，对图14中的原始图片进行特征提取，其最终输入 **ViT** 中的特征图如图15所示。

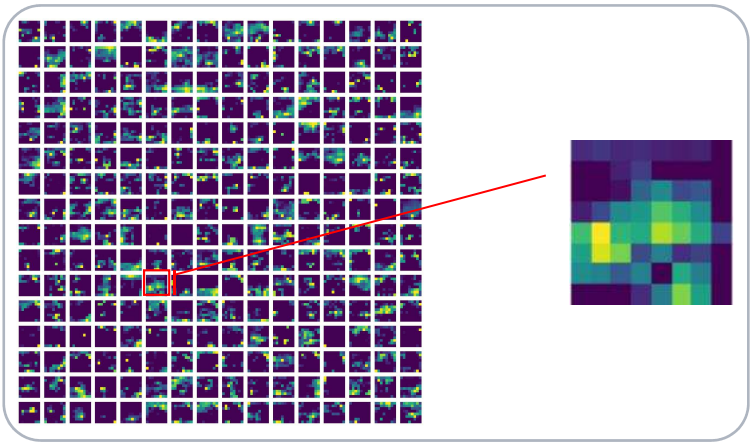


图 15: 特征图

从图15中不难看出，最终通道数设置为 256 具有较多冗余，但其中也不乏有能够较好表征原始图像中物体轮廓以及景深的特征图。这引发了我们的思考，对于 $32 * 32$ 的图像，256 维的通道或许太多了，这也有待于我们后续的深入探究。

8.3 可视化 Patch Embedding

在这一部分中，我们仿照原文中对 Patch Embedding 中的卷积层进行了可视化探究。我们使用 **ViT-Basic** 模型，同样取出 Patch Embedding 中的卷积层中前 28 个主成分，得到图16中的结果。

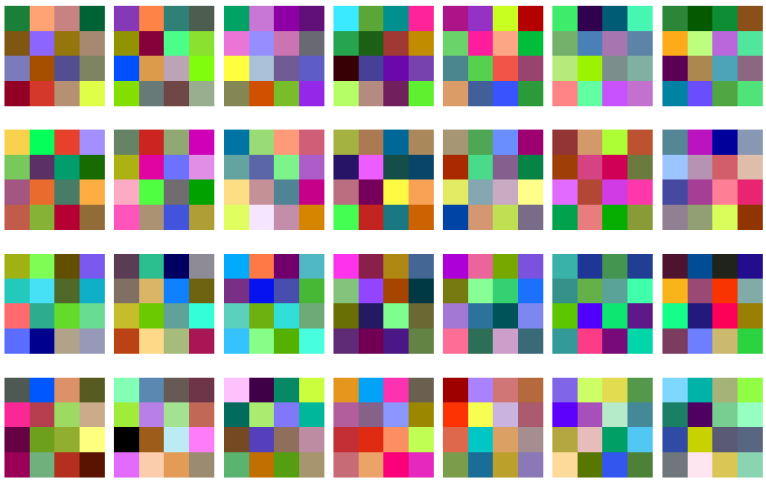


图 16: Patch Embedding

我们的结果并没有像原文中那样强的表征能力，可解释性较差。这是因为我们没有对图片进行扩展，而是直接使用 $32 * 32$ 的原始图像尺寸，导致我们选择的 patch_size 较小，进而导致这一模块的可解释性较差，这同样也有待于我们后续的探索。

8.4 注意力距离

我们选用最优的模型，参考原文，观察了各个深度以及各个注意力头的注意力距离，得到图17中的结果。

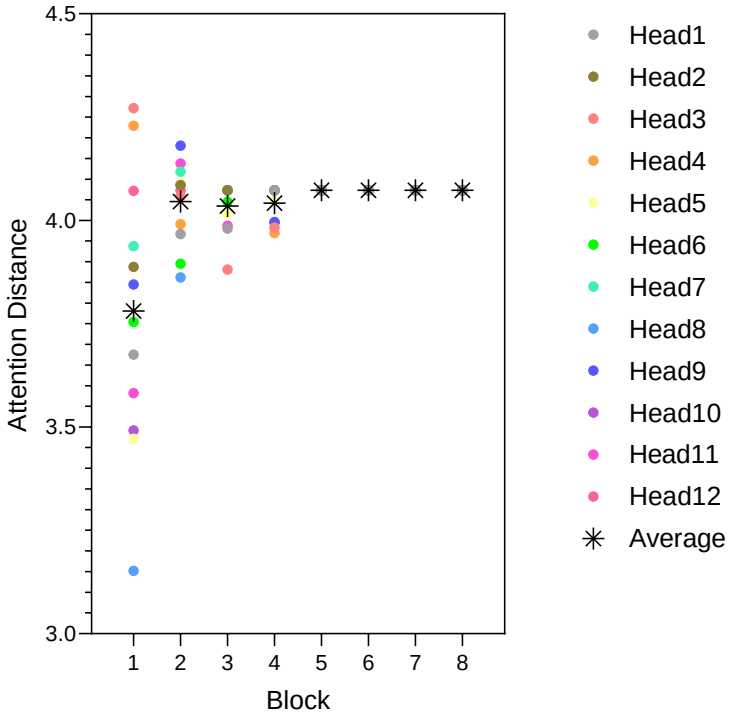


图 17: 注意力距离

不难发现，随着层数加深，平均注意力距离是上升的并逐渐饱和，表明较浅的层主要注重捕捉局部特征，较深的层则更加注重全局信息。与此同时，我们发现，随着层数加深，每个注意力头的注意力距离分布会由分散逐渐变得集中，这可能由于在较浅的层中，特征信息更加错综复杂，而在较深的层中信息变得更加凝练。

8.5 位置编码相似度

在原始论文中，作者对比了 1-d 、2-d 和相对位置这三种位置编码模式，同时比较了可学习编码与人工编码，最终得出这几种方式的表现都相差不大的结论。在我们的模型中，我们始终选择可学习的位置编码。为了探究模型对位置编码的学习能力以及最终学习到的编码的表征能力，我们仿照原文，考察模型最终学习到的位置编码中各个位置的位置编码向量的之间的相似度，得到如图18中的结果。

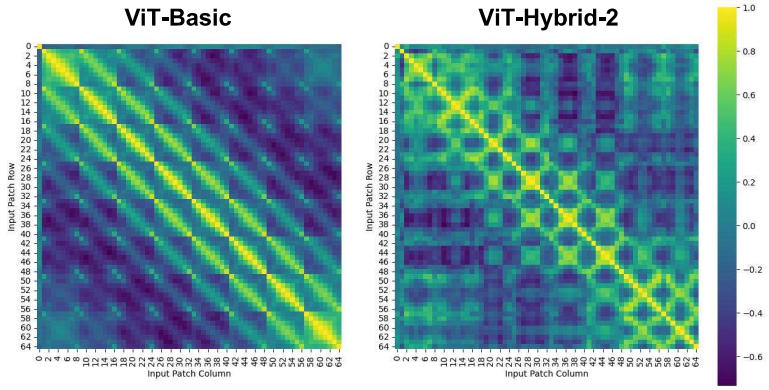


图 18: 位置嵌入相似性

我们分别选用 **ViT-Basic** 和 **ViT-Hybrid-2** 中的两个最优模型来进行该实验，得到了比较惊人的发现。首先，我们的预期是在主对角线上会出现一组极大值，但是结果表明在平行于对角线的区域也出现了极大值，经过我们的思考后，我们猜想这正表明了模型中可学习的位置编码向量对图片的二维空间信息有较强的表征能力，由于最终由特征图获得的 patch 有 8 行 8 列，第一个 patch 和第九个 patch 之间应该与第一个 patch 和第二个 patch 的位置关系应该是相似的，于是平行于主对角线的区域会出现多个极大值。其次，我们发现第零个 patch 和其余 patch 之间的位置关系联系是较弱的，这可能是因为在第零个 patch 是我们人为加进去的 `cls_token`，通过自注意力机制学习其余 token 中的信息，用来作为最终分类的判别依据，因此相对位置的联系较弱。此外，不难看出，在混合模型中，位置编码相似度的表象会削弱，我们的猜想是，使用 **ResNet** 提取特征之后，空间信息被压缩了，导致位置编码相似度并没有体现出很强的规律性。当然，以上的猜想同样也有待于我们后续进行深入探究。

9 实验结果

9.1 消融实验结果汇总

根据上述的思路，我们使用模型架构研究，正则化探究，数据增强的策略，以消融实验的方式对模型进行超参数调优，具体的实验数据（包括消融对象选择，参数选择，模型效果评估等）详见附录**??**。

9.2 最优模型

在进行充分的实验之后，我们得到的从零开始训练的最优模型（**Cifar_No_3**）在 `CIFAR-10` 上能够达到 92.54% 的 Top-1 准确率和 99.57% 的 Top-5 准确率。

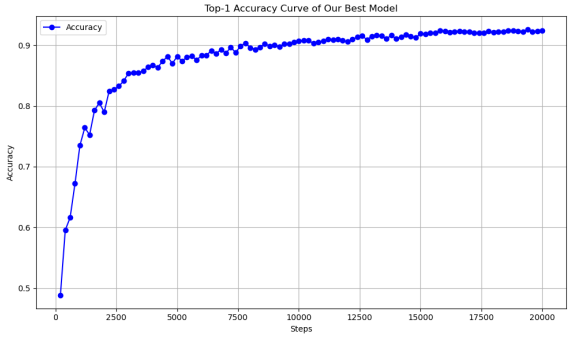


图 19: Top-1 Accuracy Curve of Our Best Model

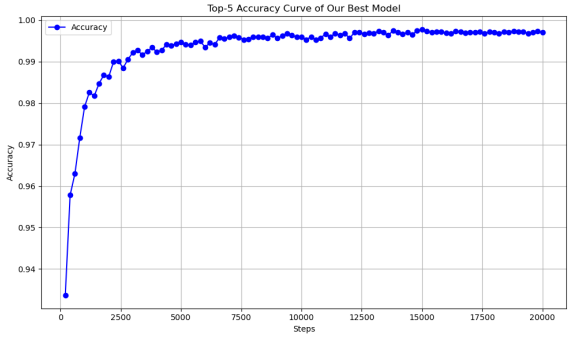


图 20: Top-5 Accuracy Curve of Our Best Model

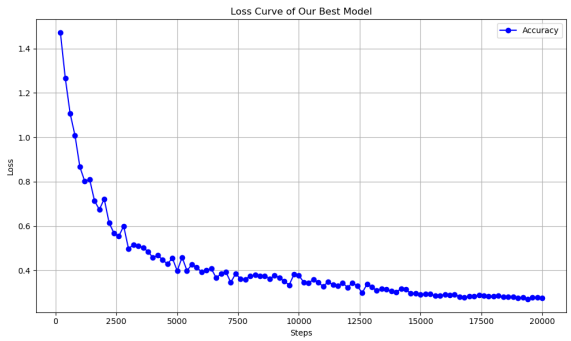


图 21: Loss Curve of Our Best Model

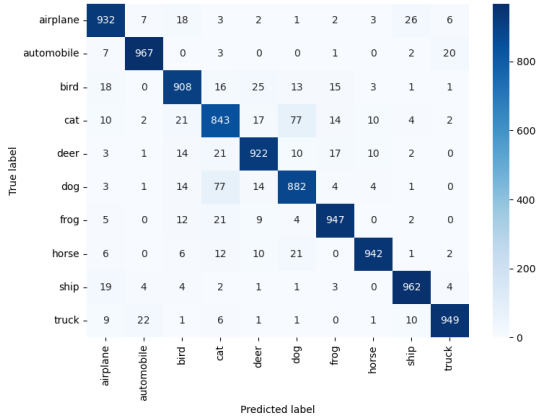


图 22: Confusion Matrix of Our Best Model

其具体的参数选择为:

LR	WD	DP	ADP	SD	RAUG	AUG	MU	CM	RCP
1e-3	5e-5	0.0	0.0	1e-3	False	cutmix	0.2	0.8	(1.0, 0.5)
Res	**#B**	**#H**	**HS**	**MD**	**PS**	**Area**	**Top-1**	**Top-5**	**PRM**
2	8	12	384	384	xx	AUG	92.54	99.57	9.18

表 1: 最佳模型的参数选择

10 与传统 CNN 的效果对比

为评估我们的 **ViT** 模型在 `CIFAR-10` 图像分类任务中的有效性，我们将其与经典的 CNN（如 **ResNet**、**VGG**、**GoogLeNet**）相比，此处以 **ResNet** 系列模型为例进行性能对比，涵盖测试精度以及参数量等维度。(**ResNet** 具体评估效果见下图 (He et al., 2016))。

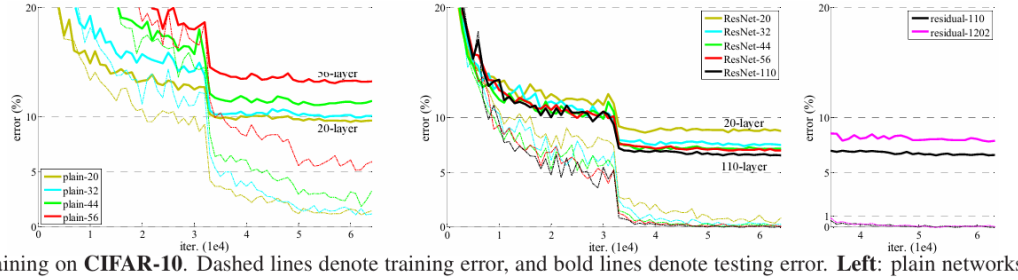


Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

*Table.*1 Classification accuracy on the test set

ResNet Type	Acc on training set	Acc on test set	Params
RestNet-20 (without BN and drop out)	90.3%	74.6%	277K
ResNet-20	98.2%	90.1%	277K
ResNet-56	98.4%	91.3%	850K
RestNet-110	98.1%	93.39%	1.7M
ResNet-164	99.2%	94.2%	2.5M

1. 准确率对比：

   - **Top-1 Accuracy：** **ViT** 相比 **ResNet-20** 与 **ResNet-56** 分别提升 $+$**2.44** 与 $+$**1.24** 个百分点，但较 **ResNet-110**（-0.85 pp）与 **ResNet-164**（-1.66 pp）略有劣势。
   - **Top-5 Accuracy：** **ViT** 达到 **99.57**%，距离 100% 仅 0.43 pp；虽然并没有关于 **ResNet** 的明确的 Top-5 Accuracy 数据，但按其 Top-1 推算约在 99.7% 左右，二者基本持平。

2. 参数效率：我们的 **ViT** 模型使用约 9.2M 参数，约是 **ResNet-164** 的 3.7 倍，却仍低于其 Top-1 Accuracy，表明当前配置下参数效率偏低；缩减隐藏维度或引入参数共享可能会有所改善。

3. 收敛与泛化：**ResNet** 在约 3×10^4 次迭代学习率衰减后测试误差陡降，直到约 4×10^4 次迭代才逐渐收敛，而我们的模型的迭代次数在 15000 步之后就逐渐收敛，其收敛效率更高。

因此，整体来看，我们的 **ViT** 在一定程度上已经接近甚至超越了一部分经典 CNN 模型在 `Cifar-10` 这样的小数据集上的测试效果。

参考文献

Abnar, S., & Zuidema, W. 2020, Quantifying Attention Flow in Transformers. https://arxiv.org/abs/2005.00928

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., & Le, Q. V. 2019, in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 113–123, doi: 10.1109/CVPR.2019.00020

Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. 2020, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 3008–3017, doi: 10.1109/CVPRW50498.2020.00359

Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. 2021, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://arxiv.org/abs/2010.11929

He, K., Zhang, X., Ren, S., & Sun, J. 2016, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778, doi: 10.1109/CVPR.2016.90

附录

A 超参数与消融实验表

No.	LR	WD	DP	ADP	SD	RAUG	AUG	MU	CM	RCP	Res	#B	#H	HS	MD	PS	Area	Top-1	Top-5	PRM
0	1e-3	5e-5	0.0	0.0	1e-3	False	**None**	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**AUG**	89.23	99.56	9.18
1	1e-3	5e-5	0.0	0.0	1e-3	False	**mixup**	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**AUG**	90.89	99.31	9.18
2	1e-3	5e-5	0.0	0.0	1e-3	False	**r_c_p**	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**AUG**	89.39	99.48	9.18
3	1e-3	5e-5	0.0	0.0	1e-3	False	**cutmix**	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**AUG**	92.54	99.57	9.18
4	1e-3	5e-5	0.0	0.0	1e-3	False	**batch_random**	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**AUG**	91.86	99.77	9.18
5	1e-3	5e-5	0.0	0.0	**0.0**	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**SD**	91.65	99.64	9.18
6	1e-3	5e-5	0.0	0.0	**1e-1**	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**SD**	91.49	99.56	9.18
7	1e-3	5e-5	0.0	0.0	**1e-2**	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**SD**	92.04	99.69	9.18
8	1e-3	5e-5	0.0	0.0	**1e-3**	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**SD**	91.81	99.59	9.18
9	1e-3	5e-5	0.0	0.0	**1e-4**	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**SD**	91.64	99.59	9.18
10	1e-3	**0.0**	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**WD**	90.35	99.59	9.18
11	1e-3	**5e-1**	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**WD**	31.62	86.45	9.18
12	1e-3	**5e-2**	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**WD**	39.28	89.76	9.18
13	1e-3	**5e-3**	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**WD**	85.93	99.27	9.18
14	1e-3	**5e-4**	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**WD**	91.64	99.61	9.18
15	1e-3	**5e-5**	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**WD**	91.81	99.57	9.18
16	1e-3	5e-5	**0.0**	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**DP**	91.62	99.65	9.18
17	1e-3	5e-5	**1e-1**	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**DP**	91.97	99.65	9.18
18	1e-3	5e-5	**1e-2**	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**DP**	91.80	99.60	9.18
19	1e-3	5e-5	**1e-3**	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**DP**	91.70	99.59	9.18
20	1e-3	5e-5	0.0	**0.0**	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**ADP**	91.96	99.61	9.18
21	1e-3	5e-5	0.0	**1e-1**	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**ADP**	91.74	99.65	9.18
22	1e-3	5e-5	0.0	**1e-2**	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**ADP**	91.97	99.66	9.18
23	1e-3	5e-5	0.0	**1e-3**	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	2	8	12	384	384	xx	**ADP**	91.92	99.63	9.18
24	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	**0**	8	12	384	384	(4, 4)	**Res**	83.77	99.13	7.16
25	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	**1**	8	12	384	384	xx	**Res**	91.75	99.54	9.85
26	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	**2**	8	12	384	384	xx	**Res**	91.83	99.51	9.18
27	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	384	384	**(2, 2)**	**PS**	55.32	94.10	7.22
28	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	384	384	**(4, 4)**	**PS**	84.15	99.00	7.16
29	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	384	384	**(8, 8)**	**PS**	78.72	98.61	7.19
30	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	**4**	12	384	384	xx	**NB**	91.51	99.58	5.63
31	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	**12**	12	384	384	xx	**NB**	91.66	99.58	12.74
32	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	**8**	384	384	xx	**NH**	91.33	99.60	9.18
33	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	**16**	384	384	xx	**NH**	91.92	99.61	9.18
34	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	**288**	**288**	xx	**HS&MD**	91.58	99.62	6.05
35	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	**288**	**384**	xx	**HS&MD**	91.71	99.57	6.49
36	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	**288**	**768**	xx	**HS&MD**	92.04	99.67	8.26
37	1e-3	5e-5	0.0	0.0	1e-3	False	batch_random	0.2	0.8	(1.0, 0.5)	0	8	12	**384**	**768**	xx	**HS&MD**	91.58	99.64	11.55
38	1e-3	5e-5	0.0	0.0	1e-3	False	cutmix	0.2	2.0	(1.0, 0.5)	0	8	12	384	384	xx	**CM**	92.21	99.74	9.18
39	1e-3	5e-5	0.0	0.0	1e-3	False	cutmix	0.2	0.1	(1.0, 0.5)	0	8	12	384	384	xx	**CM**	92.35	99.69	9.18
40	1e-3	5e-5	0.0	0.0	1e-3	False	mixup	0.8	0.8	(1.0, 0.5)	0	8	12	384	384	xx	**MU**	91.55	99.33	9.18
41	1e-3	5e-5	0.0	0.0	1e-3	False	mixup	2.5	0.8	(1.0, 0.5)	0	8	12	384	384	xx	**MU**	91.43	99.59	9.18

B 参数调优表

NO.	aug_type	cutmix	mixup	random_crop	rand_aug	(HL,MLP)	top_1	top_5
Test_1	**Cutmix**	0.8	xx	xx	**(2,9)**	xx	0.9142	0.9974
Test_2	**None**	xx	xx	xx	**(2,9)**	xx	0.8918	0.9941
Test_3	**batch_random**	0.8	2.5	(1.0,0.8)	**(4,15)**	xx	0.9193	0.9966
Test_4	**None**	xx	xx	xx	**(4,15)**	xx	0.9113	0.9968
Test_5	**None**	xx	xx	xx	**(4,15)**	xx	0.9117	0.9968
Test_6	**Cutmix**	0.8	xx	xx	**(4,15)**	xx	0.9225	0.9971
Test_7	**Mixup**	xx	2.5	xx	**(4,15)**	xx	0.9130	0.9966
Test_8	**Cutmix**	0.8	xx	xx	**(3,15)**	xx	0.9232	0.9980
Test_9	**Cutmix**	0.8	xx	xx	**(4,15)**	**(288,768)**	0.9244	0.9978
Test_10	**Cutmix**	0.8	xx	xx	**(2,15)**	xx	0.9228	0.9974
Test_11	**Cutmix**	0.8	xx	xx	False	**(288,768)**	0.9229	0.9968

表 3: 参数调优

C 符号与缩写对照表

符号/缩写	含义
LR	Learning Rate（学习率）
WD	Weight Decay（权重衰减系数）
DP	Dropout Rate（全连接层 Dropout）
ADP	Attention Dropout Rate（注意力层 Dropout）
SD	Stochastic Depth（随机深度）
RAUG	RandAugment 是否启用
AUG	数据增强策略
MU	MixUp 数据增强参数
CM	CutMix 数据增强参数
RCP	Random Crop Paste 数据增强参数
Res	采用的 ResNet 模型遍号
#B	Number of Block （Block 的数量）
#H	Number of Head （头数）
HS	Hidden Size （隐藏层维度）
MD	MLP Dimension （MLP 维度）
PS	Patch Size（Patch 尺寸）
Area	消融实验研究的对象
Top-1	Top-1 Accuracy
Top-5	Top-5 Accuracy
PRM	模型参数量（以 MiB 为单位）

D 实验硬件与软件环境

- **GPU**: NVIDIA RTX 3090 $\times$ 4 (24 GB $\times$ 4)
- **CUDA**: 12.4 **Python**: 3.10.16
- **PyTorch**: 2.5.1 **TorchVision**: 0.20.1