AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

摘要

尽管 Transformer 架构已经成为自然语言处理任务的事实标准，但其在计算机视觉中的应用仍然有限。在视觉领域，注意力机制通常与卷积网络结合使用，或者用于替代卷积网络的某些组件，同时保持整体结构不变。我们证明了这种对 CNN 的依赖并非必要，直接将纯 Transformer 应用于图像块序列可以在图像分类任务中表现非常出色。在对大规模数据进行预训练并迁移到多个中型或小型图像识别基准（如 ImageNet、CIFAR-100、VTAB 等）时，Vision Transformer（ViT）相比于最先进的卷积网络，取得了卓越的结果，同时训练所需的计算资源大幅减少。

关键字：Vision Transformer (ViT)　Image Classification　Self-Attention　Patch Embedding　Large-scale Pre-training

一、 任务介绍

1.1 问题背景与研究意义

基于自注意力机制的架构，尤其是 Transformer (Vaswani et al., 2017)，已经成为自然语言处理（NLP）中的首选模型。主流方法是先在大规模文本语料库上进行预训练，然后在较小的特定任务数据集上进行微调 (Devlin et al., 2019)。得益于 Transformer 的计算效率和可扩展性，现在可以训练参数规模超过 1000 亿的大型模型 (Brown et al., 2020; Lepikhin et al., 2020)。随着模型和数据集的不断增长，性能提升的趋势仍未见到饱和。

然而，在计算机视觉领域，卷积架构依然占据主导地位 (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016)。受 NLP 成功经验的启发，许多研究尝试将类似于卷积神经网络（CNN）的架构与自注意力机制相结合 (Wang et al., 2018; Carion et al., 2020)，有些甚至完全取代卷积 (Ramachandran et al., 2019; Wang et al., 2020a)。然而，这些后者模型虽然在理论上高效，但由于使用了特定的注意力模式，目前尚未能在现代硬件加速器上实现有效的扩展。因此，在大规模图像识别中，经典的类似 ResNet 的架构仍然是最先进的 (Mahajan et al., 2018; Xie et al., 2020; Kolesnikov et al., 2020)。

受 Transformer 在 NLP 领域扩展成功的启发，我们尝试将标准的 Transformer 直接应用于图像处理，并尽可能少地进行修改。具体来说，我们将图像划分为图像块（patches），并将这些图像块的线性嵌入序列作为 Transformer 的输入。图像块的处理方式与 NLP 应用中的词（tokens）类似。我们在图像分类任务中以监督方式训练模型。

在中型数据集（例如 ImageNet）上训练时，如果不使用强正则化，这些模型的准确率仅比同等规模的 ResNet 低几个百分点。这一看似令人失望的结果可以理解：Transformer 缺少 CNN 固有的一些归纳偏置（如平移等变性和局部性），因此在数据不足时难以很好地泛化。

然而，当模型在更大的数据集（1400 万至 3 亿张图像）上进行训练时，情况发生了变化。我们发现，大规模训练的效果超越了归纳偏置。我们的 Vision Transformer（ViT）在经过足够规模的预训练后，在小样本数据集任务中表现出色。当在公共的 ImageNet-21k 数据集或内部的 JFT-300M 数据集上预训练时，ViT 在多个图像识别基准上达到了或超过了当前最先进水平。具体而言，最佳模型在 ImageNet 上的准确率达到 88.55%，在 ImageNet-ReaL 上达到 90.72%，在 CIFAR-100 上达到 94.55%，在包含 19 个任务的 VTAB 测试套件上达到 77.63%。

1.2 相关工作

Transformer 由Vaswani et al. (2017) 提出，用于机器翻译，并且自提出以来已经成为许多 NLP 任务中的最先进方法。基于 Transformer 的大型模型通常在大规模语料库上进行预训练，然后针对具体任务进行微调：BERT (Devlin et al., 2019) 采用去噪的自监督预训练任务，而 GPT 系列则采用语言建模作为其预训练任务 (Radford et al., 2018, 2019; Brown et al., 2020)。

将自注意力机制直接应用于图像会要求每个像素与每个其他像素进行交互，这会导致计算量随着像素数量的平方增长，从而无法扩展到实际的输入规模。因此，为了在图像处理的背景下应用 Transformer，以往进行了多种近似方法尝试。Parmar et al. (2018) 仅在每个查询像素的局部邻域中应用自注意力，而不是全局应用。这种局部多头点积自注意力块可以完全取代卷积操作 (Hu et al., 2019; Ramachandran et al., 2019; Zhao et al., 2020)。另一种研究方向是稀疏 Transformer (Child et al., 2019)，通过可扩展的全局自注意力近似使其可应用于图像。还有一种方式是将注意力应用于大小不一的块中 (Weissenborn et al., 2019)，在极端情况下，仅沿着个别轴应用注意力 (Ho et al., 2019; Wang et al., 2020a)。许多这些专门的注意力架构在计算机视觉任务中表现出了令人期待的结果，但要在硬件加速器上高效实现它们需要复杂的工程设计。

与我们的方法最相关的是 Cordonnier et al. (2020) 的模型，该模型从输入图像中提取 2×2 大小的图像块，并在其上应用完整的自注意力。该模型与 ViT 非常相似，但我们的工作更进一步，证明了通过大规模预训练，标准 Transformer 能够与最先进的 CNN 竞争，甚至表现更优。此外，Cordonnier et al. (2020) 使用 2×2 像素的小图像块，这使得模型仅适用于低分辨率图像，而我们的方法可以处理中等分辨率的图像。

此外，还有许多研究尝试将卷积神经网络（CNN）与自注意力结合，例如通过增强特征图来进行图像分类 (Bello et al., 2019)，或者通过使用自注意力进一步处理 CNN 的输出，例如用于目标检测 (Hu et al., 2018; Carion et al., 2020)、视频处理 (Wang et al., 2018; Sun et al., 2019)、图像分类 (Wu et al., 2020)、无监督目标发现 (Locatello et al., 2020) 或统一的文本-视觉任务 (Chen et al., 2020c; Lu et al., 2019; Li et al., 2019)。

另一个最近相关的模型是 iGPT (Chen et al., 2020a)，它在降低图像分辨率和颜色空间后将 Transformer 应用于图像像素。该模型以生成模型的方式无监督训练，得到的表征可以进行微调或通过线性探测用于分类，其在 ImageNet 上的最高准确率为 72

我们的工作为不断增加的、探索图像识别在比标准 ImageNet 数据集更大规模上的研究文献做出了贡献。使用额外的数据源可以在标准基准上实现最先进的结果 (Mahajan et al., 2018; Touvron et al., 2019; Xie et al., 2020)。此外，Sun et al. (2017) 研究了 CNN 性能如何随着数据集规模的增加而扩展，Kolesnikov et al. (2020); Djolonga et al. (2020) 则实证探索了 CNN 从大规模数据集（如 ImageNet-21k 和 JFT-300M）进行迁移学习的效果。我们也关注这两个数据集，但使用 Transformer 而不是先前工作中基于 ResNet 的模

型进行训练。

二、 任务假设

1. Transformer 架构可以用于图像处理论文假设标准的 Transformer（原本设计用于自然语言处理任务）可以直接应用于图像处理任务，而无需依赖卷积神经网络（CNN）等特定的视觉模型结构。

2. 图像块（Patches）可以类比于 NLP 中的词（Tokens）论文认为图像可以被划分为固定大小的图像块（patches），并将每个图像块嵌入到线性空间后作为 Transformer 的输入序列，类似于 NLP 中的词输入。

3. 大规模数据集的预训练可以弥补模型归纳偏置的缺乏 Transformer 模型天然缺乏 CNN 固有的归纳偏置（如平移等变性和局部性），可能导致在小数据集上的泛化能力不足。然而，论文假设在足够大的数据集上进行预训练可以弥补这一缺点，并使 Transformer 在图像识别任务中表现出色。

4. 大规模训练优于归纳偏置论文提出了一个关键假设：大规模训练在模型性能上优于归纳偏置。即使缺乏 CNN 的特定结构偏置，Transformer 在大规模数据集上的训练仍然可以实现优异的性能。

三、 符号说明

本节对论文中使用的符号进行说明。表 1 总结了这些符号及其对应的含义。

符号	含义
$x \in \mathbb{R}^{H \times W \times C}$	原始输入图像，具有高度 H、宽度 W 和通道数 C。
$P \times P$	图像划分为的块（patch）的大小，每个图像块为 $P \times P$ 像素。
$N = \frac{H \times W}{P^2}$	图像块的总数，将整个图像划分为 N 个非重叠的图像块。
$\mathbf{x}_p \in \mathbb{R}^{P^2 \cdot C}$	每个图像块经过展平（flatten）后的特征向量表示，其中 $P^2 \cdot C$ 为展平后的维度。
$\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$	线性嵌入矩阵，将展平后的图像块投影到 D 维特征空间。
$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_1\mathbf{E}; \mathbf{x}_2\mathbf{E}; \ldots; \mathbf{x}_N\mathbf{E}] + \mathbf{E}_{pos}$	输入序列表示，包括分类标记 $\mathbf{x}_{class}$、每个图像块的线性嵌入，以及位置嵌入 $\mathbf{E}_{pos}$。
$\mathbf{E}_{pos}$	位置嵌入，用于保留图像块在原始图像中的空间位置信息。

符号	含义
L	Transformer 的编码器层数。
$\mathbf{z}_l \in \mathbb{R}^{(N+1)\times D}$	第 l 层编码器的输入序列。
MSA	多头自注意力模块（Multi-Head Self-Attention）。
MLP	多层感知机模块。
$\mathbf{z}'_l = \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}$	MSA 模块后的残差连接输出。
$\mathbf{z}_l = \text{MLP}(\text{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l$	MLP 模块后的残差连接输出。
$\mathbf{h} = \text{LN}(\mathbf{z}_L^0)$	最后一层 Transformer 中分类标记 $\mathbf{z}_L^0$ 的线性层输出，用于最终分类。
$\text{softmax}(\mathbf{Wh})$	分类头，将最终特征映射到类别预测概率。

表 1　论文符号说明

四、 模型的建立和求解

4.1 模型建立

本文提出的模型基于 Transformer 架构对图像进行分类。核心思想是: 将图像划分为固定大小的图像块（patch），展平后通过线性嵌入投影到高维特征空间，形成输入序列。加入分类标记和位置嵌入，用于分类任务和保留空间信息。后输入到标准的 Transformer 网络中 (仅使用编码器)。将最后一层编码器中的分类标记向量 (class_token) 送入全连接分类头，生成最终的类别预测。

其中，使用图像块（patches）代替 CNN 的卷积核，使图像能够以序列形式输入 Transformer，从而利用 Transformer 擅长的序列建模能力。多头自注意力机制能建模全局信息，比 CNN 的局部感受野更适合复杂模式的学习。ViT 保留了标准 Transformer 的架构，简化了设计，同时方便扩展到更大的模型和数据集。利用大规模数据集进行预训练，弥补 Transformer 在视觉任务中缺乏 CNN 归纳偏置的不足。通过这种设计，ViT 能够在多个图像识别任务中达到甚至超越 CNN 的性能。

模型的图片如图 1所示，下面是整个模型具体的搭建过程。

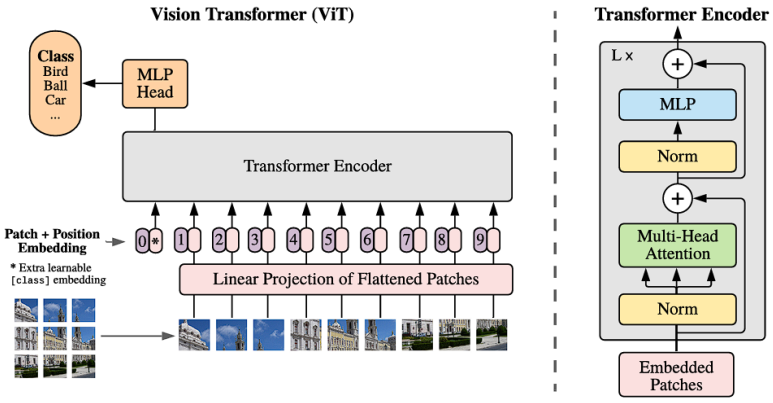

图 1　我们将图像划分为固定大小的图像块，对每个块进行线性嵌入，添加位置嵌入，并将生成的向量序列输入到标准的 Transformer 编码器中。为了进行分类，我们采用标准方法，在序列中添加一个额外的可学习“分类标记（classification token）”

4.1.1 输入处理

(1) 输入图像的尺寸为 $x \in \mathbb{R}^{H\times W\times C}$，其中：

- H：图像高度；
- W：图像宽度；
- C：通道数（如 RGB 图像的 $C=3$）。

(2) 图像被划分为大小为 $P\times P$ 的非重叠图像块（patch），每个图像块展平为一维向量 $\mathbf{x}_p \in \mathbb{R}^{P^2\cdot C}$。图像总共被划分为：

$$N=\frac{H\times W}{P^2}$$

个图像块。

(3) 每个展平的图像块通过线性投影矩阵 $\mathbf{E}\in\mathbb{R}^{(P^2\cdot C)\times D}$ 嵌入到 D 维特征空间中：

$$\mathbf{x}'_p=\mathbf{x}_p\mathbf{E},\quad \mathbf{x}'_p\in\mathbb{R}^D.$$

(4) 在所有图像块前添加一个分类标记 $\mathbf{x}_{\text{class}}\in\mathbb{R}^D$，然后为每个图像块和分类标记添加位置嵌入，形成最终输入序列：

$$\mathbf{z}_0=[\mathbf{x}_{\text{class}};\mathbf{x}'_1;\mathbf{x}'_2;\ldots;\mathbf{x}'_N]+\mathbf{E}_{\text{pos}},$$

其中 $\mathbf{E}_{\text{pos}}\in\mathbb{R}^{(N+1)\times D}$ 为位置嵌入。

4.1.2 Transformer 编码器

Transformer 编码器部分包含 L 层，每一层由以下两个模块组成：

（1）多头自注意力模块（MSA） - 多头自注意力通过以下公式计算全局关系：

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}}\right)\mathbf{V},$$

其中 $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ 分别是查询（query）、键（key）和值（value）矩阵，D_k 是缩放因子。

- 多头自注意力机制的输出表示为：

$$\text{MSA}(\mathbf{z}_{l-1}) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\mathbf{W}^O,$$

其中 h 为注意力头的数量，$\mathbf{W}^O$ 是输出投影矩阵。

- 最终通过残差连接更新为：

$$\mathbf{z}'_l = \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}.$$

（2）多层感知机模块（MLP） - 自注意力后的输出通过多层感知机（MLP）模块进一步处理，MLP 包括两层全连接网络和激活函数：

$$\text{MLP}(\mathbf{z}'_l) = \sigma(\mathbf{z}'_l\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2,$$

其中：

- σ 为激活函数（如 GELU）；
- $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ 为可学习参数。

- 同样通过残差连接更新：

$$\mathbf{z}_l = \text{MLP}(\text{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l.$$

4.1.3 输出分类

1. Transformer 编码器的最后一层输出为 $\mathbf{z}_L \in \mathbb{R}^{(N+1)\times D}$，其中分类标记的向量 $\mathbf{z}_L^0$ 被用作最终特征表示。

2. 分类标记通过分类头（全连接层）预测最终类别：

$$\mathbf{h} = \text{LN}(\mathbf{z}_L^0),$$

$$\hat{y} = \text{softmax}(\mathbf{W}\mathbf{h}),$$

其中 $\mathbf{W}$ 是分类头的权重矩阵。

4.2 模型求解

Vision Transformer (ViT) 的求解过程基于反向传播和梯度下降，用于更新模型参数以最小化损失函数。具体过程如下：

4.2.1 损失函数

在分类任务中，常用的损失函数是交叉熵损失函数。假设模型的预测输出为 $\hat{y}$，真实标签为 y，类别数量为 K，则交叉熵损失函数定义为：

$$\mathcal{L} = -\sum_{k=1}^{K} y_k \log(\hat{y}_k),$$

其中：

- $\hat{y}_k$ 是模型对第 k 类的预测概率，由 softmax 计算得到：

$$\hat{y}_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)},$$

  z_k 是分类头输出的第 k 类的得分；
- y_k 是真实类别的独热编码（one-hot encoding）。

4.2.2 反向传播过程

通过反向传播计算损失函数对模型参数的梯度，具体包括以下步骤：

（1）前向传播 将输入图像经过模型的所有层计算，得到预测输出 $\hat{y}$ 和损失 $\mathcal{L}$。

（2）损失对输出的梯度 计算损失函数对分类头输出 z_k 的梯度：

$$\frac{\partial \mathcal{L}}{\partial z_k} = \hat{y}_k - y_k.$$

（3）梯度逐层回传

- 分类头：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{h}^\top \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{z}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}},$$

  其中 $\mathbf{h}$ 是分类头输入。
- Transformer 层：对每一层（从后往前）依次计算损失对自注意力模块、MLP 模块和位置嵌入的梯度，更新这些模块的参数。
- 嵌入层：损失通过嵌入层向输入图像块的嵌入投影矩阵 $\mathbf{E}$ 回传，更新嵌入矩阵的参数。

4.2.3 参数优化过程

通过优化算法更新模型参数，本篇论文中所用优化器为 Adam。其核心公式如下：

（1）Adam 优化器公式

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

其中：

- η：学习率；
- $\hat{m}_t, \hat{v}_t$：一阶动量和二阶动量的偏差校正值；
- ϵ：防止除零的小值。

（2）动量更新

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial \mathcal{L}}{\partial \theta}, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\frac{\partial \mathcal{L}}{\partial \theta}\right)^2,$$

其中 β_1, β_2 为动量的指数衰减系数。

（3）参数更新循环 每次迭代包括以下步骤：

- 前向传播：计算损失 $\mathcal{L}$；
- 反向传播：计算所有参数的梯度 $\frac{\partial \mathcal{L}}{\partial \theta}$；
- 参数更新：根据 Adam 规则更新参数。

五、 实验

我们评估了 ResNet、Vision Transformer (ViT) 和混合模型的表示学习能力。为了理解每个模型对数据的需求，我们在不同规模的数据集上进行预训练，并评估多个基准任务。在考虑模型预训练的计算成本时，ViT 表现得非常出色，在大多数识别基准上达到了最先进的水平，同时预训练成本较低。最后，我们进行了一项小规模的自监督实验，显示了自监督 ViT 在未来的潜力。

5.1 设置

数据集。为了探索模型的可扩展性，我们使用了 ILSVRC-2012 ImageNet 数据集，该数据集包含 1000 个类别和 130 万张图片（我们在接下来的内容中简称为 ImageNet），它的超集 ImageNet-21k 包含 21000 个类别和 1400 万张图片 (Deng et al., 2009)，以及 JFT 数据集 (Sun et al., 2017)，其中包含 18000 个类别和 3.03 亿张高分辨率图片。我们根据 Kolesnikov et al. (2020) 的方法，对预训练数据集进行了去重，以避免与下游任务的测试集重合。我们将这些数据集上训练的模型转移到多个基准任务上：使用原始验证标签和清理后的 ReaL 标签的 (Beyer et al., 2020)，CIFAR-10/100 (Krizhevsky, 2009)，Oxford-IIIT Pets (Parkhi et al., 2012)，以及 Oxford Flowers-102 (Nilsback & Zisserman, 2008)。对于这些数据集，预处理过程参考了Kolesnikov et al. (2020)。

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

表 2　Details of Vision Transformer model variants.

我们还在 19 个任务的 VTAB 分类套件 (Zhai et al., 2019b) 上进行了评估。VTAB 评估在使用每个任务 1000 个训练样本时的低数据量迁移，任务被分为三组：**自然任务**——如上述的 Pets、CIFAR 等，**专业任务**——医学和卫星影像，和**结构化任务**——需要几何理解的任务如定位。

模型变体。我们基于 BERT (Devlin et al., 2019) 使用的配置构建 ViT 模型，具体信息在 2中总结。直接采用了来自 BERT 的“基础”型号和“大型”型号，同时我们增加了更大的“巨型”型号。在接下来的内容中，我们用简短的标记来表示模型的大小和输入补丁的大小，例如，ViT-L/16 表示“大型”变体，补丁大小为 16x16。需要注意的是，Transformer 的序列长度与补丁大小的平方成反比，因此补丁大小较小的模型计算成本较高。

对于基线 CNNs，我们使用 ResNet (He et al., 2016)，但将批量归一化层 (Ioffe & Szegedy, 2015) 替换为组归一化 (Wu & He, 2018)，并使用标准化卷积 (Qiao et al., 2019)。这些修改改善了迁移效果 (Kolesnikov et al., 2020)，我们将修改后的模型称为“ResNet (BiT)”。对于混合模型，我们将中间特征图输入到补丁大小为一个“像素”的 ViT 中。为了实验不同的序列长度，我们可以 (i) 使用常规 ResNet50 的第 4 阶段的输出，或 (ii) 去除第 4 阶段，在第 3 阶段放置相同数量的层（保持总层数不变），并使用拓展后的第 3 阶段的输出。选项 (ii) 导致序列长度增加 4 倍，也使得 ViT 模型计算成本更高。

训练和微调。我们使用 Adam 优化器 (Kingma & Ba, 2015) 训练所有模型，包括 ResNets，参数为 $\beta_1 = 0.9, \beta_2 = 0.999$，批量大小为 4096，并应用权重衰减 0.1，这被发现对于所有模型的迁移非常有用（附录 4.1显示，相较于常规做法，Adam 在我们的设置中比 SGD 对于 ResNets 效果略好）。我们使用线性学习率预热和衰减，详情见附录 2.1。对于微调，我们对所有模型使用带动量的 SGD，批量大小为 512，见附录 2.1.1。对于高分辨率的 ImageNet 结果（见表 3），我们对 ViT-L/16 的批量大小为 512，ViT-H/14 为 518，并且使用了Polyak & Juditsky (1992) 的因子为 0.9999 的平均法 (Ramachandran et al., 2019; Wang et al., 2020b)。

评估指标。我们在下游数据集上通过少样本或微调精度报告结果。微调精度捕捉了每个模型在各自数据集上的微调性能。少样本精度通过求解一个正则化最小二乘回归问题得到，该问题将一部分训练图像的（冻结的）表示映射到 $\{-1, 1\}^K$ 目标向量。这个

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	$88.4/88.5^*$
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

表 3　在流行的图像分类基准上与最新技术的比较。我们报告了在三次微调运行中，准确率的均值和标准差。我们模型在 JFT-300M 数据集上预训练，且在所有数据集上均优于基于 ResNet 的基线，同时在预训练时所需的计算资源显著更少。在较小的公共 ImageNet-21k 数据集上预训练的我们的模型表现也很好。* 在 Touvron et al. (2020) 中报告的稍微改进的 88.5% 结果。

公式允许我们以封闭形式恢复精确解。虽然我们主要关注微调性能，但有时使用线性少样本精度进行快速即席评估，其中微调代价太高。

5.2 与当前技术水平的比较

我们首先将最大的模型——ViT-H/14 和 ViT-L/16——与文献中的最新 CNN 进行比较。第一个比较点是大规模迁移（BiT）(Kolesnikov et al., 2020)，它使用大规模 ResNets 进行监督迁移学习。第二个是 Noisy Student (Xie et al., 2020)，这是一种大型 EfficientNet，使用半监督学习在 ImageNet 和无标签的 JFT-300M 上进行训练。目前，Noisy Student 在 ImageNet 上以及 BiT-L 在此处报告的其他数据集上是最新技术水平。所有模型均在 TPUv3 硬件上训练，我们报告了预训练每个模型所需的 TPUv3-核心天数，即用于训练的 TPU v3 核心数（每个芯片 2 个）乘以培训天数。

表3显示了结果。预训练在 JFT-300M 上的较小 ViT-L/16 模型在所有任务上表现优于 BiT-L（预训练于相同数据集），同时训练所需的计算资源大幅减少。更大的模型 ViT-H/14 进一步提高了性能，尤其是在更具挑战性的数据集上——ImageNet、CIFAR-100 和 VTAB 套件。有趣的是，该模型预训练所需的计算量仍然比之前的最新技术水平更少。然而，我们注意到，预训练效率可能不仅受架构选择的影响，还受到其他参数的影响，

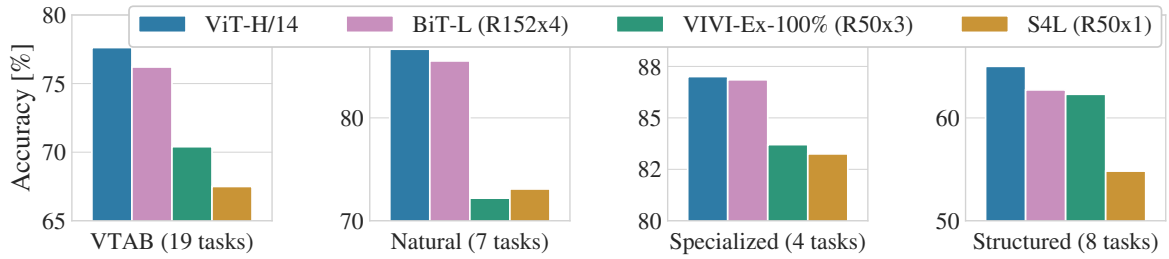

图 2　Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

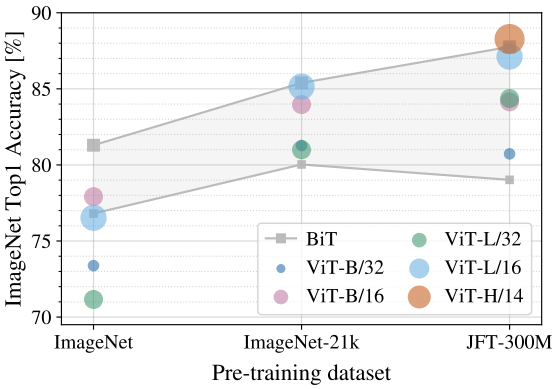

图 3　迁移到 ImageNet 。虽然在小数据集上预训练的大型 ViT 模型表现逊色于 BiT ResNets（阴影区域），但在大型数据集上预训练时则表现出色。同样，随着数据集的增大，更大的 ViT 变体也超越了较小的变体。

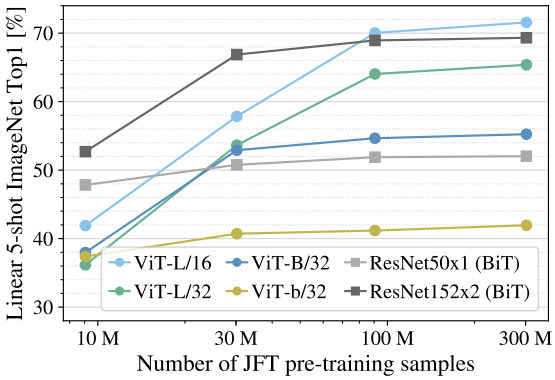

图 4　在 ImageNet 上的线性少量样本评估与预训练规模的关系。ResNets 在较小的预训练数据集上表现更佳，但很快达到平台期，而 ViT 在较大预训练数据集上表现更好。ViT-b 是将 ViT-B 的所有隐层维度减半的版本。

如训练计划、优化器、权重衰减等。在第5.4节中，我们提供了不同架构的性能和计算受控研究。最后，预训练在公共 ImageNet-21k 数据集上的 ViT-L/16 模型在大多数数据集上的表现也很好，同时预训练所需的资源更少：可以使用 8 核心的标准云 TPUv3 在大约 30 天内进行训练。

图 2将 VTAB 任务分解为各自的组，并与此基准上的先前 SOTA 方法进行比较：BiT，VIVI——在 ImageNet 和 YouTube 上共同训练的 ResNet (Tschannen et al., 2020)，以及 S4L——在 ImageNet 上进行的监督加半监督学习 (Zhai et al., 2019a)。ViT-H/14 在自然和结构化任务中表现优于 BiT-R152x4 和其他方法。在专门任务中，前两种模型的性能相似。

5.3 预训练数据要求

当在大型 JFT-300M 数据集上预训练时，Vision Transformer 表现良好。与 ResNets 相比，对于视觉的归纳偏差较少，那么数据集的大小究竟有多关键呢？我们进行了两系列实验。

首先，我们在逐渐增大的数据集上预训练 ViT 模型：ImageNet、ImageNet-21k 和 JFT-300M。为了提升在较小数据集上的表现，我们优化了三个基本的正则化参数——权重衰减、dropout 和标签平滑。图 3显示了微调到 ImageNet 后的结果（其他数据集的结果见表 6）。当在最小的数据集 ImageNet 上预训练时，ViT-Large 模型的表现逊色于 ViT-Base 模型，尽管进行了（适度的）正则化。在 ImageNet-21k 预训练时，它们的表现相似。只有在 JFT-300M 上，我们才能看到大模型的完全优势。图 3还展示了不同大小的 BiT 模型所覆盖的性能区域。BiT CNN 在 ImageNet 上的表现优于 ViT，但在更大的数据集上，ViT 超越了。

第二，我们在 9M、30M 和 90M 的随机子集以及完整的 JFT-300M 数据集上训练模型。我们没有对较小的子集进行额外的正则化，并对所有设置使用相同的超参数。这样，我们评估的是模型的内在属性，而不是正则化的影响。然而，我们使用了提前停止，并报告了训练期间实现的最佳验证精度。为了节省计算资源，我们报告少样本线性精度而不是完全微调精度。图 4包含结果。Vision Transformer 在较小数据集上比 ResNets 更容易过拟合，即便计算成本相似。例如，ViT-B/32 的速度略快于 ResNet50；但在 9M 子集上的表现较差，而在 90M 以上子集上表现更好。ResNet152x2 和 ViT-L/16 也是如此。这个结果进一步强化了这样的直觉：卷积归纳偏差在较小数据集上是有用的，而对于较大的数据集，从数据中直接学习相关模式已经足够，甚至是有益的。

总体而言，ImageNet 上的少样本结果（图 4），以及 VTAB 上的低数据结果（表 3）为非常低数据迁移提供了良好前景。进一步分析 ViT 的少样本特性是未来研究一个令人兴奋的方向。

5.4 规模研究

我们通过评估从 JFT-300M 的迁移性能来进行不同模型的受控规模研究。在这种情况下，数据大小并不会限制模型的性能，我们评估每个模型的性能与预训练成本之间的关系。模型集包括：7 个 ResNets，R50x1、R50x2、R101x1、R152x1、R152x2，预训练 7 个周期，以及 R152x2 和 R200x3 预训练 14 个周期；6 个 Vision Transformer，ViT-B/32、B/16、L/32、L/16，预训练 7 个周期，加上 L/16 和 H/14 预训练 14 个周期；以及 5 个混合模型，R50+ViT-B/32、B/16、L/32、L/16 预训练 7 个周期，加上 R50+ViT-L/16 预训练 14 个周期（对于混合模型，名称末尾的数字不是表示 patch_size，而是表示 ResNet 主干网络的总下采样比例）。

图 5显示了迁移性能与总预训练计算资源之间的关系（有关计算成本的详细信息见

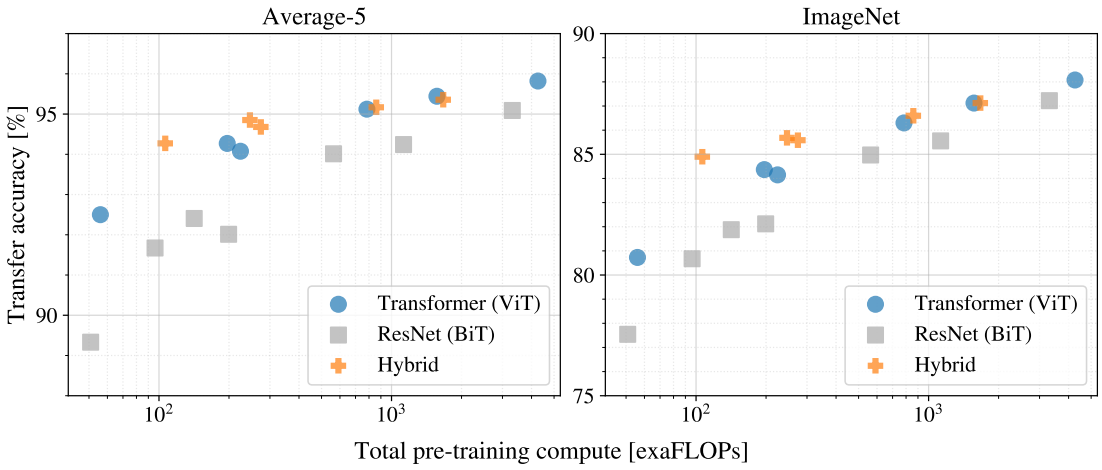

图 5　不同架构的性能与预训练计算资源的关系：Vision Transformers、ResNets 和混合架构。Vision Transformers 通常在相同计算预算下优于 ResNets。对于较小的模型，混合架构相较于纯 Transformer 取得了改进，但对于较大的模型，这一差距消失。

附录 4.5）。每个模型的详细结果在附录表 7中提供。我们可以观察到一些模式。首先，在性能/计算折衷方面，Vision Transformer 主导 ResNets。ViT 使用的计算资源约为后者的 $\frac{1}{2}-\frac{1}{4}$，以达到相同的性能（在 5 个数据集上的平均值）。其次，在较小的计算预算下，混合模型略微优于 ViT，但在较大模型时差异消失。这一结果有些令人惊讶，因为人们可能会认为卷积的局部特征处理在任何规模下都能支持 ViT。第三，Vision Transformer 在尝试的范围内似乎未饱和，激励了未来的扩展努力。

5.5 检查 Vision Transformer

为了开始理解 Vision Transformer 是如何处理图像数据的，我们分析其内部表示。Vision Transformer 的第一层将 flattened patches 线性投影到一个低维空间中。图 7（左）显示了学习到的嵌入滤波器的主成分。这些成分类似于每个 patch 内部细结构的低维表示的合理基函数。

在投影之后，将学习到的 position embedding 添加到 patch representations 中。图 7（中）显示了模型学习将图像中的距离编码到 position embedding 的相似性中，即距离较近的 patch 通常具有更相似的 position embedding。此外，行列结构似乎出现；在同一行/列的 patch 具有相似的 embedding。最后，对于较大的网格，有时可以明显看到正弦结构（附录 D）。position embeddings 学习表示 2D 图像拓扑，这解释了为何手工设计的 2D-aware embedding 变体没有带来改

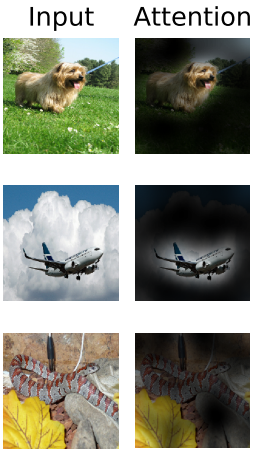

图 6　输出 token 对输入空间的注意力示例。

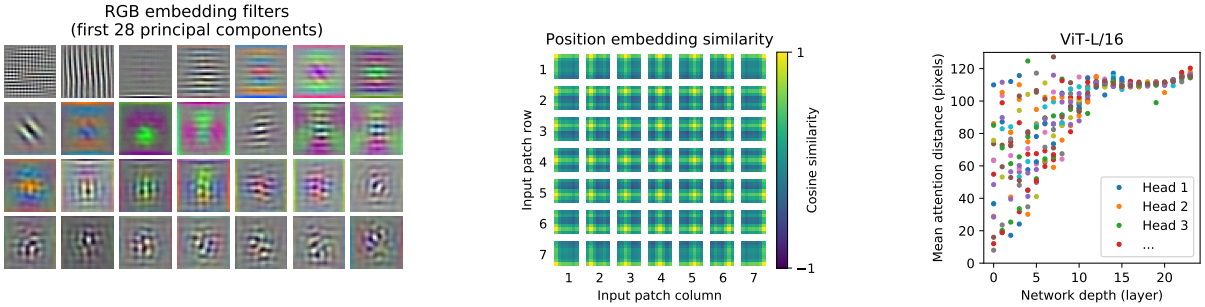

图 7　Left: ViT-L/32 RGB 值的初始线性嵌入的滤波器。Center: ViT-L/32 位置嵌入的相似性。瓷砖显示了与指示的行和列的补丁位置嵌入之间的余弦相似性，以及所有其他补丁的位置信息。Right: 不同头部和网络深度所关注的区域大小。每个点表示一层中 16 个头部之一在图像上的平均注意距离。

进（Appendix 4.4）。

自注意力机制使得 ViT 在最低层也能够整合整个图像的信息。我们调查网络在多大程度上利用了这一能力。具体来说，我们根据注意力权重计算信息整合的图像空间中的平均距离（图 7，右）。这个“注意力距离”类似于 CNN 中的感受野大小。我们发现一些注意力头在最低层就已经关注到了大多数图像，显示模型确实利用了全局整合信息的能力。其他注意力头在低层中的注意力距离始终较小。这种高度局部化的注意力在应用 ResNet 作为 Transformer 之前的混合模型中不太明显（图 7，右），这表明它可能在功能上类似于 CNN 中的早期卷积层。此外，注意力距离随着网络深度的增加而增加。总体来说，我们发现模型关注于与分类语义相关的图像区域（图 6）。

5.6 自我监督

Transformer 在自然语言处理任务上展现了令人印象深刻的性能。然而，它们成功的原因不仅在于出色的可扩展性，还在于大规模的自我监督预训练 (Devlin et al., 2019; Radford et al., 2018)。我们也对自我监督进行了初步探索，通过 *masked patch prediction*，模仿 BERT 中使用的掩码语言建模任务。在自我监督预训练下，我们较小的 ViT-B/16 模型在 ImageNet 上达到了 79.9% 的准确率，相较于从头训练提升了 2%，但仍然比监督预训练低 4%。附录 2.1.2包含了更多细节。我们将对对比预训练 (Chen et al., 2020b; He et al., 2020; Bachman et al., 2019; Hénaff et al., 2020) 的探索留待未来工作。

六、 模型的分析

6.1 ViT 有效性分析

6.1.1 全局建模能力

ViT 采用 Transformer 架构的核心在于自注意力机制（Self-Attention）。自注意力机制允许每个输入的 patch（图像块）与其他所有 patch 进行交互，从而捕捉图像中全局的依赖关系。这种全局建模能力使 ViT 能够理解图像的整体结构和长距离的特征关联，这是传统 CNN 难以实现的。传统 CNN 通过卷积操作逐层扩大感受野，依赖于多层堆叠来实现全局信息的捕捉。相比之下，ViT 的自注意力机制天生具备全局感知能力，无需通过增加层数来扩大感受野。这不仅提升了信息传递的效率，也减少了梯度消失或爆炸的风险。

6.1.2 灵活的特征表示

ViT 将图像划分为固定大小的 patch，并通过线性映射将其转换为高维向量。这种方法赋予 ViT 更大的灵活性，因为每个 patch 被视为一个独立的 “token”，类似于 NLP 中的词向量。相比于 CNN 依赖固定的卷积核，ViT 的 patch 嵌入方式更适应不同图像内容的多样性。Transformer 中的多头自注意力机制允许模型在不同的表示子空间中同时关注不同的特征。这种多样性的特征表示能力，使得 ViT 能够捕捉到更丰富和多样化的视觉信息，提升了模型的表达能力。

6.1.3 位置编码的有效性

ViT 通过位置编码（Positional Encoding）为每个 patch 提供位置信息，弥补了 Transformer 本身不具备位置信息的不足。位置编码使得模型能够理解 patch 在图像中的空间关系，从而保留了图像的空间结构信息。除了固定的位置编码，ViT 还可以采用可学习的位置编码，使得模型能够更灵活地适应不同的空间关系。这种可学习性进一步增强了 ViT 对图像空间结构的理解能力。。

6.2 ViT 相对于卷积神经网络减少运算资源开销分析

5.4 规模研究的实验中，我们可以明显的观察到，在一定区间内，达到相同性能的前提下，ViT 的计算资源开销约为传统 ResNet 的 $\frac{1}{2}-\frac{1}{4}$。ViT 通过将图像划分为 patch，将输入序列长度从整个图像的像素数减少到 patch 数。例如，对于 224×224 的图像，16×16 的 patch 划分将生成 $14 \times 14 = 196$ 个 patch。相比于处理每个像素，处理 patch 数显著减少了输入序列长度，从而降低了自注意力机制的计算复杂度。

其次，传统的 ResNet 网络，需要 101，152 甚至更多的层数才能学习一个非常大数据集上的特征，完成分类任务，这大大提高了运算开销，相对而言 ViT 仅仅需要 12/24 个 Transformer Encoder 便可以实现特征的学习。并且在实验中表明，Vision Transformer 仍然未饱和，不仅大大降低了运算开销，更显示了 Vision Transformer 架构的巨大学习容量的潜力。

6.3 ViT 的不足

6.3.1 对大规模数据的依赖

ViT 在大规模数据集上表现优异，但其性能高度依赖于大量的训练数据。这是由于 Transformer 架构本身具有大量的参数，需要大量的数据来避免过拟合。然而，获取大规模标注数据集在实际应用中往往困难且昂贵。在小规模数据集上，ViT 的表现通常不如卷积神经网络（CNN）。这主要是因为 ViT 缺乏 CNN 固有的局部特征提取能力，导致在数据有限的情况下，模型难以有效泛化。

6.3.2 位置编码的敏感性

ViT 依赖位置编码来保留 patch 的位置信息，然而不同类型的位置编码（如固定位置编码和可学习位置编码）对模型性能有显著影响。位置编码的不当选择可能导致模型无法有效理解图像的空间结构。Transformer 架构本身缺乏对序列顺序的内在感知，ViT 必须依赖位置编码来弥补这一不足。这使得模型在处理不同尺寸或变形的图像时，需要重新调整位置编码策略，增加了模型的复杂性。

6.3.3 需要更多的正则化和数据增强

由于 Transformer 模型较为复杂，相对于传统的 CNN 网络参数众多，很容易过拟合到较小的数据集上。为了防止过拟合，ViT 通常需要更复杂的正则化方法，如数据增强、Dropout、Stochastic Depth 等。这些方法的应用增加了训练过程的复杂性和计算开销。ViT 在训练过程中高度依赖数据增强策略来提升模型的泛化能力。然而，设计和实施有效的数据增强策略需要大量的实验和经验，限制了模型的快速迭代和应用。

6.4 ViT 的改进以及应用

6.4.1 混合架构的设计

ViT 擅长全局建模，但在捕捉局部特征方面不如 CNN。通过引入卷积层或其他局部特征提取模块，可以增强模型对低级特征（如边缘、纹理）的捕捉能力。例如，Swin Transformer 采用了层级结构和局部窗口自注意力机制，成功结合了 CNN 的优势。

6.4.2 稀疏自注意力

标准自注意力机制的计算复杂度为 $\mathcal{O}(N^2)$，其中 N 为输入序列长度。引入稀疏自注意力机制，通过限制每个 token 仅与部分 token 进行交互，可以显著降低计算开销。例如，Big Bird 和 Sparse Transformer 提出了稀疏注意力模式，提升了计算效率。

6.4.3 知识蒸馏

上文提到 Vision Transformer 模型在大数据集上的表现显著的优于中小数据集上的表现，甚至超过了当前最好的基于卷积的分类效果。所以我们可以通过通过知识蒸馏技术，可以将大型 ViT 模型的知识迁移到较小的学生模型中，减少模型参数量和计算开销。例如，已经有的工作 DeiT（Data-efficient Image Transformer）通过知识蒸馏显著提升了 ViT 在小数据集上的表现。

七、 总结

我们已经探索了 Transformer 在图像识别中的直接应用。与之前在计算机视觉中使用自注意力的工作不同，我们没有在架构中引入任何图像特定的归纳偏置，除了初始的 patch 提取步骤。相反，我们将图像解释为一系列 patch，并通过在自然语言处理（NLP）中使用的标准 Transformer 编码器进行处理。这种简单而可扩展的策略在与大规模数据集的预训练结合时，出人意料地取得了良好的效果。因此，Vision Transformer 在许多图像分类数据集上达到了或超越了现有的最先进水平，同时预训练的成本相对较低。

虽然这些初步结果令人鼓舞，但仍然面临许多挑战。其中之一是将 ViT 应用于其他计算机视觉任务，如目标检测和语义分割。我们的结果，结合 Carion 等（2020）的研究，表明这种方法有潜力。另一个挑战是继续探索自监督预训练方法。我们的初步实验表明，自监督预训练能带来一些提升，但自监督预训练与大规模监督预训练之间仍然存在较大的差距。最后，进一步扩展 ViT 的规模可能会带来性能的进一步提升。

参考文献

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In ACL, 2020.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In NeurIPS, 2019.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In ICLR, 2019.

I. Bello, B. Zoph, Q. Le, A. Vaswani, and J. Shlens. Attention augmented convolutional networks. In ICCV, 2019.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In Large Scale Kernel Machines. MIT Press, 2007.

Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? arXiv, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. arXiv, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In ECCV, 2020.

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, and Heewoo Jun. Generative pretraining from pixels. In ICML, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In ICML, 2020b.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: UNiversal Image-TExt Representation Learning. In ECCV, 2020c.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. arXiv, 2019.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In ICLR, 2020.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL, 2019.

Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D'Amour, Dan Moldovan, Sylvan Gelly, Neil Houlsby, Xiaohua Zhai, and Mario Lucic. On robustness and transferability of convolutional neural networks. arXiv, 2020.

Oguzhan Gencoglu, Mark van Gils, Esin Guldogan, Chamin Morikawa, Mehmet Süzen, Mathias Gruber, Jussi Leinonen, and Heikki Huttunen. Hark side of deep learning – from grad student descent to automated machine learning. arXiv, 2019.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In AISTATS, 2011.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning, volume 1. MIT Press, 2016.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In NeurIPS, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In CVPR, 2020.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv, 2016.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527–1554, 2006.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. arXiv, 2019.

Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In CVPR, 2018.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In ICCV, 2019.

Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang. Ccnet: Criss-cross attention for semantic segmentation. In ICCV, 2020.

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In ICML, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In ECCV, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1:541–551, 1989.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv, 2020.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. In Arxiv, 2019.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. arXiv, 2020.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In NeurIPS. 2019.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In ECCV, 2018.

M. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In ICVGIP, 2008.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In CVPR, 2012.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In ICML, 2018.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization, 30(4):838–855, 1992. doi: 10.1137/0330046. URL https://doi.org/10.1137/0330046.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Weight standardization. arXiv preprint arXiv:1903.10520, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical Report, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical Report, 2019.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In NeurIPS, 2019.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In NeurIPS, 2016.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In ICCV, 2017.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In ICCV, 2019.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. In NeurIPS. 2019.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy: Fixefficientnet. arXiv preprint arXiv:2003.08237, 2020.

Michael Tschannen, Josip Djolonga, Marvin Ritter, Aravindh Mahendran, Neil Houlsby, Sylvain Gelly, and Mario Lucic. Self-supervised learning of video-induced visual invariances. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In ECCV, 2020a.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. arXiv preprint arXiv:2003.07853, 2020b.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation. In ACL, 2019.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In CVPR, 2018.

Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In ICLR, 2019.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. arxiv, 2020.

Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In CVPR, 2020.

Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S^4L: Self-Supervised Semi-Supervised Learning. In ICCV, 2019a.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867, 2019b.

Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In CVPR, 2020.

附录 A　Multihead Self-attention

Standard qkv self-attention (SA, Vaswani et al. (2017)) is a popular building block for neural architectures. For each element in an input sequence $z \in \mathbb{R}^{N \times D}$, we compute a weighted sum over all values v in the sequence. The attention weights A_{ij} are based on the pairwise similarity between two elements of the sequence and their respective query q^i and key k^j representations.

$$[q, k, v] = zU_{qkv} \qquad U_{qkv} \in \mathbb{R}^{D \times 3D_h}, \tag{1}$$

$$A = softmax\left(qk^\top / \sqrt{D_h}\right) \qquad A \in \mathbb{R}^{N \times N}, \tag{2}$$

$$SA(z) = Av\,. \tag{3}$$

Multihead self-attention (MSA) is an extension of SA in which we run k self-attention operations, called "heads", in parallel, and project their concatenated outputs. To keep compute and number of parameters constant when changing k, D_h (Eq. 1) is typically set to D/k.

$$MSA(z) = [SA_1(z); SA_2(z); \cdots; SA_k(z)]\, U_{msa} \qquad U_{msa} \in \mathbb{R}^{k \cdot D_h \times D} \tag{4}$$

附录 B　Experiment details

2.1　Training

Table 4 summarizes our training setups for our different models. We found strong regularization to be key when training models from scratch on ImageNet. Dropout, when used, is applied after every dense layer except for the the qkv-projections and directly after adding positional- to patch embeddings. Hybrid models are trained with the exact setup as their ViT counterparts. Finally, all training is done on resolution 224.

2.1.1　Fine-tuning

We fine-tune all ViT models using SGD with a momentum of 0.9. We run a small grid search over learning rates, see learning rate ranges in Table 5. To do so, we use small sub-splits from the training set (10% for Pets and Flowers, 2% for CIFAR, 1% ImageNet) as development set and train on the remaining data. For final results we train on the entire training set and evaluate on the respective test data. For fine-tuning ResNets and hybrid models we use the exact same setup, with the only exception of ImageNet where we add another value 0.06 to the learning rate sweep. Additionally, for ResNets we also run the setup of Kolesnikov et al. (2020) and select the best results across this run and our sweep. Finally, if not mentioned otherwise, all fine-tuning experiments run at 384 resolution (running fine-tuning at different resolution than training is common practice (Kolesnikov et al., 2020)).

Models	Dataset	Epochs	Base LR	LR decay	Weight decay	Dropout
ViT-B/{16,32}	JFT-300M	7	$8 \cdot 10^{-4}$	linear	0.1	0.0
ViT-L/32	JFT-300M	7	$6 \cdot 10^{-4}$	linear	0.1	0.0
ViT-L/16	JFT-300M	7/14	$4 \cdot 10^{-4}$	linear	0.1	0.0
ViT-H/14	JFT-300M	14	$3 \cdot 10^{-4}$	linear	0.1	0.0
R50x{1,2}	JFT-300M	7	10^{-3}	linear	0.1	0.0
R101x1	JFT-300M	7	$8 \cdot 10^{-4}$	linear	0.1	0.0
R152x{1,2}	JFT-300M	7	$6 \cdot 10^{-4}$	linear	0.1	0.0
R50+ViT-B/{16,32}	JFT-300M	7	$8 \cdot 10^{-4}$	linear	0.1	0.0
R50+ViT-L/32	JFT-300M	7	$2 \cdot 10^{-4}$	linear	0.1	0.0
R50+ViT-L/16	JFT-300M	7/14	$4 \cdot 10^{-4}$	linear	0.1	0.0
ViT-B/{16,32}	ImageNet-21k	90	10^{-3}	linear	0.03	0.1
ViT-L/{16,32}	ImageNet-21k	30/90	10^{-3}	linear	0.03	0.1
ViT-$*$	ImageNet	300	$3 \cdot 10^{-3}$	cosine	0.3	0.1

表 4 Hyperparameters for training. All models are trained with a batch size of 4096 and learning rate warmup of 10k steps. For ImageNet we found it beneficial to additionally apply gradient clipping at global norm 1. Training resolution is 224.

Dataset	Steps	Base LR
ImageNet	20 000	{0.003, 0.01, 0.03, 0.06}
CIFAR100	10 000	{0.001, 0.003, 0.01, 0.03}
CIFAR10	10 000	{0.001, 0.003, 0.01, 0.03}
Oxford-IIIT Pets	500	{0.001, 0.003, 0.01, 0.03}
Oxford Flowers-102	500	{0.001, 0.003, 0.01, 0.03}
VTAB (19 tasks)	2 500	0.01

表 5 Hyperparameters for fine-tuning. All models are fine-tuned with cosine learning rate decay, a batch size of 512, no weight decay, and grad clipping at global norm 1. If not mentioned otherwise, fine-tuning resolution is 384.

When transferring ViT models to another dataset, we remove the whole head (two linear layers) and replace it by a single, zero-initialized linear layer outputting the number of classes required by the target dataset. We found this to be a little more robust than simply re-initializing the very last layer.

For VTAB we follow the protocol in Kolesnikov et al. (2020), and use the same hyperparameter setting for all tasks. We use a learning rate of 0.01 and train for 2500 steps (Tab. 5). We chose this setting by running a small sweep over two learning rates and two schedules, and selecting the setting with the highest VTAB score on the 200-example validation sets. We follow the pre-processing used in Kolesnikov et al. (2020), except that we do not use task-specific input resolutions. Instead we find that Vision Transformer benefits most from a high resolution (384×384) for all tasks.

2.1.2 Self-supervision

We employ the *masked patch prediction* objective for preliminary self-supervision experiments. To do so we corrupt 50% of patch embeddings by either replacing their embeddings with a learnable `[mask]` embedding (80%), a random other patch embedding (10%) or just keeping them as is (10%). This setup is very similar to the one used for language by Devlin et al. (2019). Finally, we predict the 3-bit, mean color (i.e., 512 colors in total) of every corrupted patch using their respective patch representations.

We trained our self-supervised model for 1M steps (ca. 14 epochs) with batch size 4096 on JFT. We use Adam, with a base learning rate of $2 \cdot 10^{-4}$, warmup of 10k steps and cosine learning rate decay. As prediction targets for pretraining we tried the following settings: 1) predicting only the mean, 3bit color (i.e., 1 prediction of 512 colors), 2) predicting a 4×4 downsized version of the 16×16 patch with 3bit colors in parallel (i.e., 16 predictions of 512 colors), 3) regression on the full patch using L2 (i.e., 256 regressions on the 3 RGB channels). Surprisingly, we found that all worked quite well, though L2 was slightly worse. We report final results only for option 1) because it has shown best few-shot performance. We also experimented with 15% corruption rate as used by Devlin et al. (2019) but results were also slightly worse on our few-shot metrics.

Lastly, we would like to remark that our instantiation of masked patch prediction doesn't require such an enormous amount of pretraining nor a large dataset such as JFT in order to lead to similar performance gains on ImageNet classification. That is, we observed diminishing returns on downstream performance after 100k pretraining steps, and see similar gains when pretraining on ImageNet.

附录 C Additional Results

We report detailed results corresponding to the figures presented in the paper. Table 6 corresponds to Figure 3 from the paper and shows transfer performance of different ViT models pre-trained on datasets of increasing size: ImageNet, ImageNet-21k, and JFT-300M. Table 7

		ViT-B/16	ViT-B/32	ViT-L/16	ViT-L/32	ViT-H/14
ImageNet	CIFAR-10	98.13	97.77	97.86	97.94	-
	CIFAR-100	87.13	86.31	86.35	87.07	-
	ImageNet	77.91	73.38	76.53	71.16	-
	ImageNet ReaL	83.57	79.56	82.19	77.83	-
	Oxford Flowers-102	89.49	85.43	89.66	86.36	-
	Oxford-IIIT-Pets	93.81	92.04	93.64	91.35	-
ImageNet-21k	CIFAR-10	98.95	98.79	99.16	99.13	99.27
	CIFAR-100	91.67	91.97	93.44	93.04	93.82
	ImageNet	83.97	81.28	85.15	80.99	85.13
	ImageNet ReaL	88.35	86.63	88.40	85.65	88.70
	Oxford Flowers-102	99.38	99.11	99.61	99.19	99.51
	Oxford-IIIT-Pets	94.43	93.02	94.73	93.09	94.82
JFT-300M	CIFAR-10	99.00	98.61	99.38	99.19	99.50
	CIFAR-100	91.87	90.49	94.04	92.52	94.55
	ImageNet	84.15	80.73	87.12	84.37	88.04
	ImageNet ReaL	88.85	86.27	89.99	88.28	90.33
	Oxford Flowers-102	99.56	99.27	99.56	99.45	99.68
	Oxford-IIIT-Pets	95.80	93.40	97.11	95.83	97.56

表 6　Top1 accuracy (in %) of Vision Transformer on various datasets when pre-trained on ImageNet, ImageNet-21k or JFT300M. These values correspond to Figure 3 in the main text. Models are fine-tuned at 384 resolution. Note that the ImageNet results are computed without additional techniques (Polyak averaging and 512 resolution images) used to achieve results in Table 3.

corresponds to Figure 5 from the paper and shows the transfer performance of ViT, ResNet, and hybrid models of varying size, as well as the estimated computational cost of their pre-training.

name	Epochs	ImageNet	ImageNet ReaL	CIFAR-10	CIFAR-100	Pets	Flowers	exaFLOPs
ViT-B/32	7	80.73	86.27	98.61	90.49	93.40	99.27	55
ViT-B/16	7	84.15	88.85	99.00	91.87	95.80	99.56	224
ViT-L/32	7	84.37	88.28	99.19	92.52	95.83	99.45	196
ViT-L/16	7	86.30	89.43	99.38	93.46	96.81	99.66	783
ViT-L/16	14	87.12	89.99	99.38	94.04	97.11	99.56	1567
ViT-H/14	14	88.08	90.36	99.50	94.71	97.11	99.71	4262
ResNet50x1	7	77.54	84.56	97.67	86.07	91.11	94.26	50
ResNet50x2	7	82.12	87.94	98.29	89.20	93.43	97.02	199
ResNet101x1	7	80.67	87.07	98.48	89.17	94.08	95.95	96
ResNet152x1	7	81.88	87.96	98.82	90.22	94.17	96.94	141
ResNet152x2	7	84.97	89.69	99.06	92.05	95.37	98.62	563
ResNet152x2	14	85.56	89.89	99.24	91.92	95.75	98.75	1126
ResNet200x3	14	87.22	90.15	99.34	93.53	96.32	99.04	3306
R50x1+ViT-B/32	7	84.90	89.15	99.01	92.24	95.75	99.46	106
R50x1+ViT-B/16	7	85.58	89.65	99.14	92.63	96.65	99.40	274
R50x1+ViT-L/32	7	85.68	89.04	99.24	92.93	96.97	99.43	246
R50x1+ViT-L/16	7	86.60	89.72	99.18	93.64	97.03	99.40	859
R50x1+ViT-L/16	14	87.12	89.76	99.31	93.89	97.36	99.11	1668

表 7　Detailed results of model scaling experiments. These correspond to Figure 5 in the main paper. We show transfer accuracy on several datasets, as well as the pre-training compute (in exaFLOPs).

附录 D　Additional Analyses

4.1　SGD vs. Adam for ResNets

ResNets are typically trained with SGD and our use of Adam as optimizer is quite unconventional. Here we show the experiments that motivated this choice. Namely, we compare the fine-tuning performance of two ResNets – 50x1 and 152x2 – pre-trained on JFT with SGD and Adam. For SGD, we use the hyperparameters recommended by Kolesnikov et al. (2020). Results are presented in Table 8. Adam pre-training outperforms SGD pre-training on most datasets and on average. This justifies the choice of Adam as the optimizer used to pre-train

	ResNet50		ResNet152x2	
Dataset	Adam	SGD	Adam	SGD
ImageNet	77.54	78.24	84.97	84.37
CIFAR10	97.67	97.46	99.06	99.07
CIFAR100	86.07	85.17	92.05	91.06
Oxford-IIIT Pets	91.11	91.00	95.37	94.79
Oxford Flowers-102	94.26	92.06	98.62	99.32
Average	89.33	88.79	94.01	93.72

表 8　Fine-tuning ResNet models pre-trained with Adam and SGD.

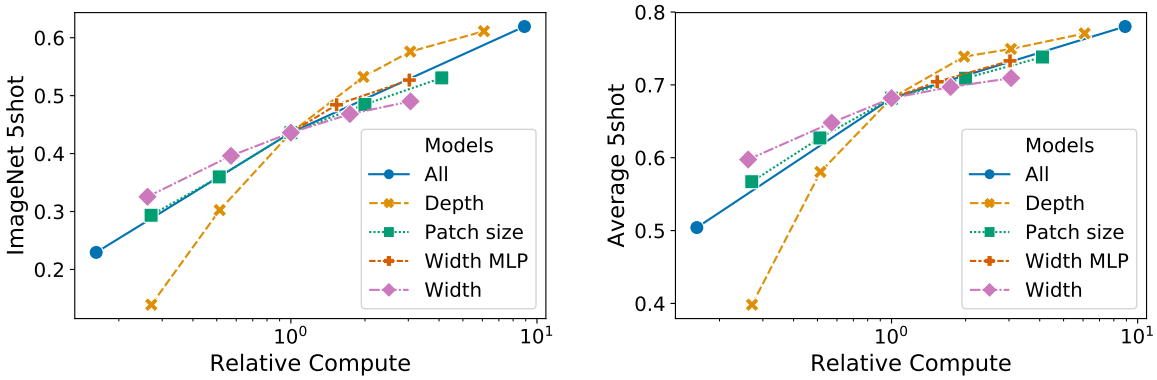

图 8　Scaling different model dimensions of the Vision Transformer.

ResNets on JFT. Note that the absolute numbers are lower than those reported by Kolesnikov et al. (2020), since we pre-train only for 7 epochs, not 30.

4.2　Transformer shape

We ran ablations on scaling different dimensions of the Transformer architecture to find out which are best suited for scaling to very large models. Figure 8 shows 5-shot performance on ImageNet for different configurations. All configurations are based on a ViT model with 8 layers, $D = 1024$, $D_{MLP} = 2048$ and a patch size of 32, the intersection of all lines. We can see that scaling the depth results in the biggest improvements which are clearly visible up until 64 layers. However, diminishing returns are already visible after 16 layers. Interestingly, scaling the width of the network seems to result in the smallest changes. Decreasing the patch size and thus increasing the effective sequence length shows surprisingly robust improvements without introducing parameters. These findings suggest that compute might be a better predictor of per-

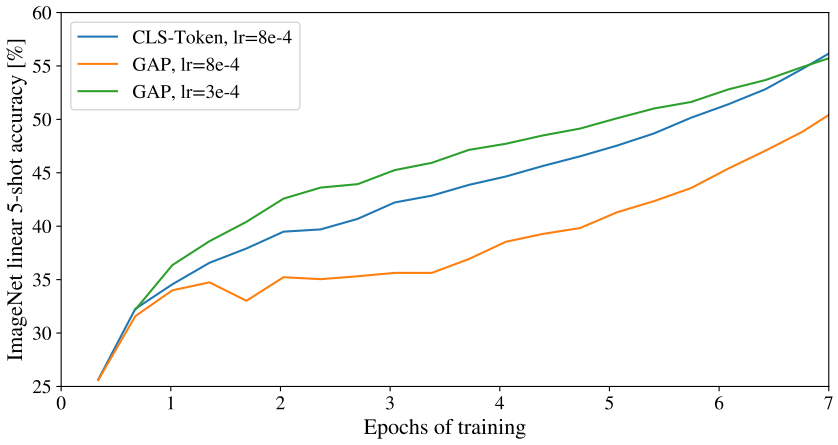

图 9 Comparison of class-token and global average pooling classifiers. Both work similarly well, but require different learning-rates.

formance than the number of parameters, and that scaling should emphasize depth over width if any. Overall, we find that scaling all dimensions proportionally results in robust improvements.

4.3 Head Type and `class` token

In order to stay as close as possible to the original Transformer model, we made use of an additional `[class]` token, which is taken as image representation. The output of this token is then transformed into a class prediction via a small multi-layer perceptron (MLP) with tanh as non-linearity in the single hidden layer.

This design is inherited from the Transformer model for text, and we use it throughout the main paper. An initial attempt at using only image-patch embeddings, globally average-pooling (GAP) them, followed by a linear classifier—just like ResNet's final feature map—performed very poorly. However, we found that this is neither due to the extra token, nor to the GAP operation. Instead, the difference in performance is fully explained by the requirement for a different learning-rate, see Figure 9.

4.4 Positional Embedding

We ran ablations on different ways of encoding spatial information using positional embedding. We tried the following cases:

- Providing no positional information: Considering the inputs as a bag of patches.
- 1-dimensional positional embedding: Considering the inputs as a sequence of patches in the raster order (default across all other experiments in this paper).

Pos. Emb.	Default/Stem	Every Layer	Every Layer-Shared
No Pos. Emb.	0.61382	N/A	N/A
1-D Pos. Emb.	0.64206	0.63964	0.64292
2-D Pos. Emb.	0.64001	0.64046	0.64022
Rel. Pos. Emb.	0.64032	N/A	N/A

表 9　Results of the ablation study on positional embeddings with ViT-B/16 model evaluated on ImageNet 5-shot linear.

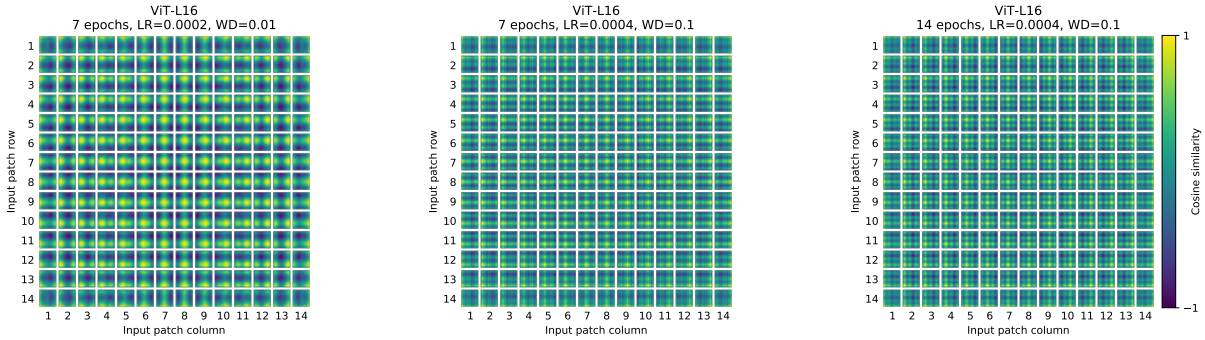

图 10　Position embeddings of models trained with different hyperparameters.

- 2-dimensional positional embedding: Considering the inputs as a grid of patches in two dimensions. In this case, two sets of embeddings are learned, each for one of the axes, X-embedding, and Y-embedding, each with size $D/2$. Then, based on the coordinate on the path in the input, we concatenate the X and Y embedding to get the final positional embedding for that patch.
- Relative positional embeddings: Considering the relative distance between patches to encode the spatial information as instead of their absolute position. To do so, we use 1-dimensional Relative Attention, in which we define the relative distance all possible pairs of patches. Thus, for every given pair (one as query, and the other as key/value in the attention mechanism), we have an offset $p_q - p_k$, where each offset is associated with an embedding. Then, we simply run extra attention, where we use the original query (the content of query), but use relative positional embeddings as keys. We then use the logits from the relative attention as a bias term and add it to the logits of the main attention (content-based attention) before applying the softmax.

In addition to different ways of encoding spatial information, we also tried different ways of incorporating this information in our model. For the 1-dimensional and 2-dimensional positional embeddings, we tried three different cases: (1) add positional embeddings to the inputs

right after the stem of them model and before feeding the inputs to the Transformer encoder (default across all other experiments in this paper); (2) learn and add positional embeddings to the inputs at the beginning of each layer; (3) add a learned positional embeddings to the inputs at the beginning of each layer (shared between layers).

Table 9 summarizes the results from this ablation study on a ViT-B/16 model. As we can see, while there is a large gap between the performances of the model with no positional embedding and models with positional embedding, there is little to no difference between different ways of encoding positional information. We speculate that since our Transformer encoder operates on patch-level inputs, as opposed to pixel-level, the differences in how to encode spatial information is less important. More precisely, in patch-level inputs, the spatial dimensions are much smaller than the original pixel-level inputs, e.g., 14×14 as opposed to 224×224, and learning to represent the spatial relations in this resolution is equally easy for these different positional encoding strategies. Even so, the specific pattern of position embedding similarity learned by the network depends on the training hyperparameters (Figure 10).

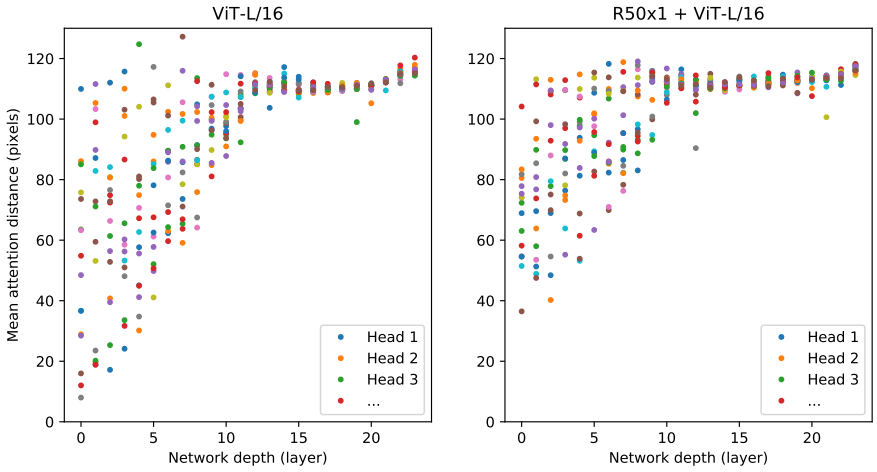

图 11 Size of attended area by head and network depth. Attention distance was computed for 128 example images by averaging the distance between the query pixel and all other pixels, weighted by the attention weight. Each dot shows the mean attention distance across images for one of 16 heads at one layer. Image width is 224 pixels.

4.5 Empirical Computational Costs

We are also interested in real-world speed of the architectures on our hardware, which is not always well predicted by theoretical FLOPs due to details like lane widths and cache sizes. For this purpose, we perform timing of inference speed for the main models of interest, on a

TPUv3 accelerator; the difference between inference and backprop speed is a constant model-independent factor.

Figure 12 (left) shows how many images one core can handle per second, across various input sizes. Every single point refers to the peak performance measured across a wide range of batch-sizes. As can be seen, the theoretical bi-quadratic scaling of ViT with image size only barely starts happening for the largest models at the largest resolutions.

Another quantity of interest is the largest batch-size each model can fit onto a core, larger being better for scaling to large datasets. Figure 12 (right) shows this quantity for the same set of models. This shows that large ViT models have a clear advantage in terms of memory-efficiency over ResNet models.

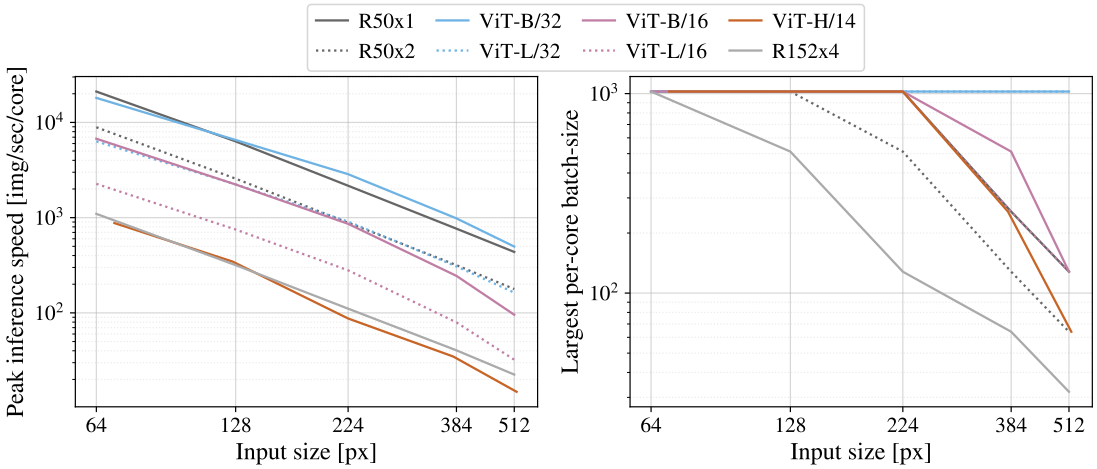

图 12 Left: Real wall-clock timings of various architectures across input sizes. ViT models have speed comparable to similar ResNets. Right: Largest per-core batch-size fitting on device with various architectures across input sizes. ViT models are clearly more memory-efficient.

4.6 Axial Attention

Axial Attention (Huang et al., 2020; Ho et al., 2019) is a simple, yet effective technique to run self-attention on large inputs that are organized as multidimensional tensors. The general idea of axial attention is to perform multiple attention operations, each along a single axis of the input tensor, instead of applying 1-dimensional attention to the flattened version of the input. In axial attention, each attention mixes information along a particular axis, while keeping information along the other axes independent. Along this line, Wang et al. (2020b) proposed the AxialResNet model in which all the convolutions with kernel size 3×3 in a ResNet50 are re-

placed by axial self-attention, i.e. a row and column attention, augmented by relative positional encoding. We have implemented AxialResNet as a baseline model.[1].

Moreover, we have modified ViT to process inputs in the 2-dimensional shape, instead of a 1-dimensional sequence of patches, and incorporate Axial Transformer blocks, in which instead of a self-attention followed by an MLP, we have a a row-self-attention plus an MLP followed by a column-self-attention plus an MLP.

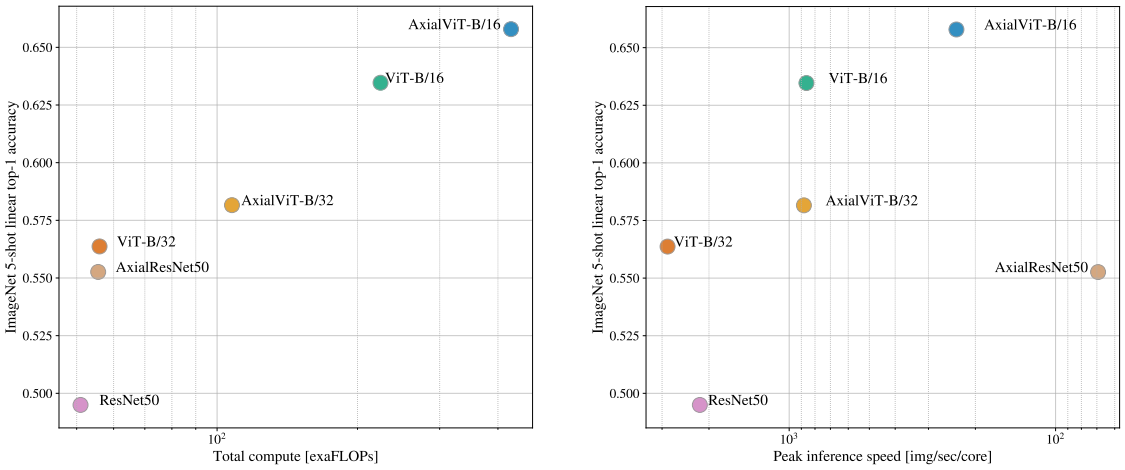

图 13 Performance of Axial-Attention based models, in terms of top-1 accuracy on ImageNet 5-shot linear, versus their speed in terms of number of FLOPs (left) and inference time (left).

Figure 13, present the performance of Axial ResNet, Axial-ViT-B/32 and Axial-ViT-B/16 on ImageNet 5shot linear, when pretrained on JFT dataset, verses the pretraining compute, both in terms of number of FLOPs and inference time (example per seconds). As we can see, both Axial-ViT-B/32 and Axial-ViT-B/16 do better than their ViT-B counterpart in terms of performance, but it comes at the cost of more compute. This is because in Axial-ViT models, each Transformer block with global self-attention is replaced by two Axial Transformer blocks, one with row and one with column self-attention and although the sequence length that self-attention operates on is smaller in axial case, there is a extra MLP per Axial-ViT block. For the Axial-ResNet, although it looks reasonable in terms of accuracy/compute trade-off (Figure 13, left), the naive implementation is extremely slow on TPUs (Figure 13, right).

[1]Our implementation is based on the open-sourced PyTorch implementation in `https://github.com/csrhddlam/axial-deeplab`. In our experiments, we reproduced the scores reported in (Wang et al., 2020b) in terms of accuracy, however, our implementation, similar to the open-source implementation, is very slow on TPUs. Therefore, we were not able to use it for extensive large-scale experiments. These may be unlocked by a carefully optimized implementation.