上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Transformer 讲稿整理

课程: 人工智能的数学基础

姓名: 刘常靖

日期: 2022 年 6 月

目录

1 任务

1.1 任务概述

Transformer[1] 是用于解决 seqence to swquence(seq2seq) 任务, 即 transformer 实现了以下序列到序列的映射:

$$f : \mathbb{R}^{L_{in} \times d_{in}} \rightarrow \mathbb{R}^{L_{out} \times d_{out}}$$

以翻译任务为例子，其中 L_{in} 代表输入句子 (sentance) 的长度，d_{in} 代表输入单词 (word) 某种编码形式的维度。对于某一种编码方式，其维度是固定值。其中 L_{out} 代表输出句子 (sentance) 的长度，d_{out} 代表输出单词 (word) 的维度。需要注意的是，翻译前后的句子长度可能不同，此外单词的不同顺序也影响翻译结果。

1.2 任务分析

任务分析主要是指对于单词关联之间的分析。以翻译为例的 seq2seq 任务。每一个输出的单词都决定于:

(1) 所有的输入单词;

(2) 所有之前已经输出的单词。

同时单词间的关联关系有如下特点:

(1) 关联的长度可长可短;

(2) 关联强度是可强可弱。

1.3 网络设计的动机

基于对任务的分析，有如下网络设计的动机:

(1) 合理表示输入输出。即采用嵌入 (Embedding) 的方式对单词进行唯一编码和表示;

(2) 进行全局关联。一方面编码器考虑所有的输入，另一方面解码器在训练时考虑编码器的所有输出；

(3) 关联计算方式。单词之间的关系采用内积 (dot-product) 进行度量；

(4) 不同的关联强度的处理。将关联的大小作为权重，并用 softmax 对权重进行归一化。

其中 (3)(4) 就构成了注意力机制 (Attention)。

2 数据预处理

数据预处理主要的功能是将句子中的单词转化成向量表示，并加上位置信息。Transformer 中的单词预处理由词嵌入 (Word Embedding) 转化为向量, 并通过位置编码 (Positional Encoding,PE) 获取位置信息共同组成的。

2.1 词嵌入

为了能让模型处理单词，需要对单词进行表示。最简单的单词表示方法就是 one-hot 编码，但是该方法使得单词之间相互独立等价，进而丢失了单词之间的关联信息。因此 Transformer 中采用词嵌入 Word Embedding 进行单词表示。

设输入单词为为 $\chi \in \mathbb{R}^{n\times N}$, 其中 n 为单词数目，n 为字典数目，即为单词的 one-hot 向量表示。词嵌入实际上实现了一个线性映射:

$$\widetilde{X} = \chi W_E \tag{1}$$

其中 $\widetilde{X} \in \mathbb{R}^{n\times d_m}$ 为词嵌入的输出, $W_E \in \mathbb{R}^{N\times d_m}$ 为词嵌入中可训练的参数。

2.2 位置编码

相比于 RNN，Transformer 的优点是使用了全局信息，即使用了单词的位置信息。位置编码 (Positional Encoding) 用 $PE \in \mathbb{R}^{n\times d_m}$ 表示，其计

算公式如下：

$$\begin{cases} PE_{(pos,2i)} = sin(pos/10000^{2i/d}) \\ PE_{(pos,2i+1)} = cos(pos/10000^{2i/d}) \end{cases} \tag{2}$$

其中 pos 表示单词在句子中的位置，d 表示位置编码的维度，2i 表示偶数的维度，2i+1 表示奇数维度。使用这样的公式计算有如下好处：

(1) 具有平移不变性。两个词的位置编码只依赖于其相对位置；

(2) padding 操作不会影响有意义的单词。

2.3 词嵌入 + 位置编码

Transformer 中的单词是由单词嵌入和位置编码两部分相加组成的，即最终的数据预处理输出为：

$$X = \widetilde{X} + PE \tag{3}$$

其中 $X \in \mathbb{R}^{n \times d_m}$ 即为词嵌入和位置编码的预处理结果，该结果将在多头注意力机制作为输入使用。

3 注意力机制

Transformer 中使用了缩放点积注意力机制 (scaled dot-product attention) 的注意力机制 (Attention)，同时并行使用多个该结构来组成多头注意力机制 (multi-head attention) 。其主要目的是获得单词关联，从而获得单词权重信息。

3.1 缩放点积注意力机制

Transformer 的自注意力机制采用了缩放点积注意力机制 (Scaled Dot-Product Attention) 的结构，即在原始的内积类表示词关联的自注意力机制

上增加了点积缩放的结构。该结构如图3.1所示。输入的参数为 Q(Query),K(Key) 和 V(value)，三个参数都来源于 X, 并通过如下方式构造：

$$\begin{cases} Q = XW^Q \\ K = XW^K \\ V = XW^V \end{cases} \tag{4}$$

其中 $W^Q, W^K, W^K \in \mathbb{R}^{d_m \times d_m}$，这三个参数均为网络中可训练的参数。如图3.1所示，Q 与 K 首先计算其内积计算关联性 (图3.1中 MatMul)。为了防止内积过大，进行缩放 (图3.1中 Scale) 操作，此处对计算的内积除以 $\sqrt{d_k}$，随后进行 softmax 实现归一化。最后将归一化后的值与 V 计算内积 (图中 MatMul)，得到注意力机制的输出。该计算步骤有如下公式表示：

$$X^{[1]} = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5}$$

其中 $X^{[1]} \in \mathbb{R}^{N \times d_m}$, 其维度与 X 相同。

相比于使用 RNN 等网络，自注意力机制可以对一个句子中的所有单词进行并行计算，充分利用了 GPU 等硬件资源，提高了计算效率。同时加入了缩放结构可以防止 softmax 函数的梯度消失。

3.2 多头注意力机制

多头注意力机制 (Multi-Head Attention) 是由多个自注意力机制 scale dot profuct attention 组成的。这是因为单个子注意力机制往往只能学习到一种单词之间的关联，而采用了多头注意力机制可以学习更多的单词之间的关联，提高模型的泛化性。

多头注意力机制的结构如下图3.1所示。其输入的参数为 Q, K, V 的构造方式与 5.1 节类似。Q, K, V 首先经过线性层 (Linear) 分别获得 $Q = [Q_1, Q_2, ..., Q_i, ..., Q_h]$, $K = [K_1, K_2, ..., K_i, ..., K_h]$ 和 $V = [V_1, V_2, ..., V_i, ..., V_h]$,

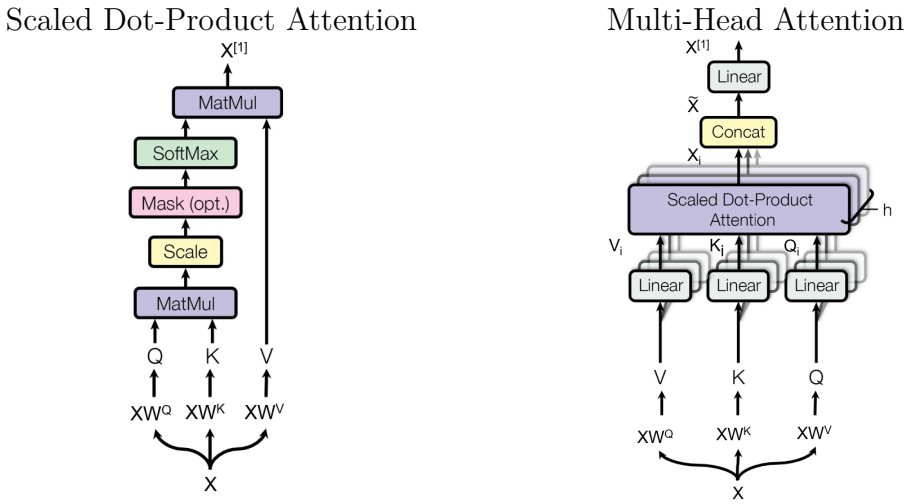

图 1: 左图为 Scaled Dot-Product Attention。右图为 Multi-Head Attention，其包含多个注意力层实现并行计算。

其中 h 为缩放点积注意力机制的层数，i 代表其中第 i 层，其第 i 层的构造方式如下式所示：

$$\begin{cases} Q_i = XW_i^Q \\ K_i = XW_i^K \\ V_i = XW_i^V \end{cases} \tag{6}$$

其中 $W_i^Q, W_i^K \in \mathbb{R}^{n\times d_k}$, $W_i^V \in \mathbb{R}^{n\times d_v}$, 均为可训练的参数。$d_k = d_v = \frac{d_m}{h}$。随后对第 i 层进行缩放点积注意力机制, 得到第 i 层的输出：

$$\widetilde{X_i} = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i \tag{7}$$

其中 $\widetilde{X_i} \in \mathbb{R}^{n\times d_v}$。随后将每一层进行合并操作 (Concat)，得到：

$$\widetilde{X} = [\widetilde{X_1}, \widetilde{X_2}, \cdots, \widetilde{X_h}] \in \mathbb{R}^{n\times hd_v} \tag{8}$$

最后再经过一个线性层 (Linear) 得到最终输出。

$$X^{[1]} = \widetilde{X}W^0 \in \mathbb{R}^{N\times d_m} \tag{9}$$

其中 $W^0 \in \mathbb{R}^{hd_v\times d_m}$，为可训练参数。

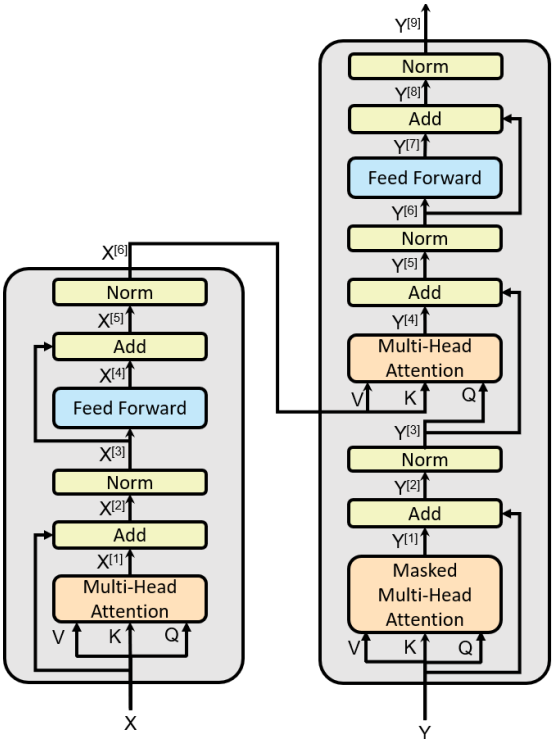

图 2: Transformer 的编码器与解码器结构。左侧为编码器，右侧为解码器。

4 编码器

编码器 (Encoder) 结构如图2所示，其主要包含 4 个部分：多头注意力机制，残差连接 & 层规范化，前馈网络，残差连接 & 层规范化。其中编码器的输入为经过嵌入编码和位置编码的向量 $X \in \mathbb{R}^{n \times d_m}$。

4.1 多头注意力机制

由 $X \in R^{n \times d_m}$ 构造 $Q, K, V \in \mathbb{R}^{n \times d_m}$, 如下式所示：

$$\begin{cases} Q = XW^Q \\ K = XW^K \\ V = XW^V \end{cases} \tag{10}$$

其中 $W^Q, W^K, W^V \in \mathbb{R}^{n \times d_m}$。由 Q, K, V 经多头注意力机制得到 $X^{[1]}$

$$X^{[1]} = MultiHead(Q, K, V) \tag{11}$$

4.2 残差连接 & 层归一化

残差连接可表示为

$$X^{[2]} = X^{[1]} + X \tag{12}$$

随后进行层归一化，对每个单词做归一化。

$$X_{ij}{}^{[3]} = \frac{X_{ij}^{[2]} - \sum_{j=1}^{d_m} X_{ij}{}^{[2]}}{\sqrt{\frac{1}{d_m} \sum_{j=1}^{d_m} \left(X_{ij}^{[2]} - \sum_{j=1}^{d_m} X_{ij}^{[2]}\right)^2}} \tag{13}$$

以上的层规范化可简化为下式：

$$X_{ij}{}^{[3]} = LayerNorm(X_{ij}{}^{[2]}) \tag{14}$$

4.3 前馈网络

前馈网络 (Feed-forward network) 采用全连接网络 (Deep Neural Networks,DNN)，对于每个单词 $X_{i,:}^{[3]}$ 均使用该网络，以实现以下映射：

$$DNN_{\theta_x} : R^{d_m} \to R^{d_m} \tag{15}$$

即：

$$X_{i,:}^{[4]} = DNN_{\theta_x}(X_{i,:}^{[3]}) \tag{16}$$

其中 θ_x 是 CNN 中可训练的参数。

4.4 残差连接 & 层归一化

残差连接

$$X^{[5]} = X^{[4]} + X^{[3]} \tag{17}$$

层归一化

$$X_{ij}{}^{[6]} = LayerNorm(X_{ij}{}^{[5]}) \tag{18}$$

5 解码器

解码器 Decoder 的结构如图2中的右侧所示，其包含 6 个部分：带掩码的多头注意力机制，残差连接 & 层归一化，与输入耦合的多头注意力机制，残差连接 & 层归一化，前馈网络，残差连接 & 层归一化。此处以训练过程为例子阐述其计算流程。

5.1 数据预处理

在训练阶段，译文已知 $\Upsilon \in \mathbb{R}^{n_0 \times N_0}$，其中 n_0 代表译文里的单词数目，N_0 代表在译文字典里单词长度。通过词嵌入和位置编码可以获得预处理的译文 $Y \in \mathbb{R}^{n_0 \times d_m}$:

$$Y = \Upsilon W_{\widetilde{E}} + \widetilde{PE} \tag{19}$$

其中 $W_{\widetilde{E}} \in \mathbb{R}^{N_0 \times d_m}$ 为词嵌入的可训练参数。$\widetilde{PE} \in \mathbb{R}^{n_0 \times d_m}$ 为位置编码输出。

5.2 带掩码的多头注意力机制

由 $Y \in \mathbb{R}^{n_0 \times d_m}$ 可构造 Q, K, V, 如下式所示

$$\begin{cases} Q = YW^Q \\ K = XW^K \\ V = XW^V \end{cases} \tag{20}$$

其中 $Q, K, V \in \mathbb{R}^{n_0 \times d_m}, W^Q, W^K, W^V \in \mathbb{R}^{n_0 \times d_m}$。$Q, K, V$ 经过线性层 Linear 分别获得:

$$\begin{cases} Q_i = XW_i^Q \\ K_i = XW_i^K \\ V_i = XW_i^V \end{cases} \tag{21}$$

其中 $W_i^Q, W_i^K \in \mathbb{R}^{n \times d_k}$, $W_i^V \in \mathbb{R}^{n \times d_v}$, 均为可训练的参数。$d_k = d_v = \frac{d_m}{h}$。

为了能让当前输出的单词能够学习到之前所有的输出单词，同时不受到之后单词的影响，在此处构造掩码：

$$\widetilde{A}_{i,j} = \begin{cases} \frac{Q_{i,j}K_{i,j}}{\sqrt{d_k}}, i \geq j \\ -10^9, i < j \end{cases} \tag{22}$$

注意：如果有 padding 的话，在末尾不会产生影响。随后，第 i 层的缩放点积的输出 $\widetilde{Y}_i$ 如下所示

$$\widetilde{Y}_i = softmax(\widetilde{A}_i)V_i \tag{23}$$

随后将每一层进行合并操作 (Concat)，得到：

$$\widetilde{Y} = [\widetilde{Y}_1, \cdots, \widetilde{Y}_i, \cdots, \widetilde{Y_h}] \in \mathbb{R}^{n_0 \times hd_v} \tag{24}$$

最后再经过一个线性层 (Linear) 得到最终输出：

$$Y^{[1]} = \widetilde{Y}W^0 \in \mathbb{R}^{n_0 \times d_m} \tag{25}$$

其中 $W^0 \in \mathbb{R}^{hd_v \times d_m}$ 为可训练参数。

5.3 残差连接 & 层归一化

$$Y^{[2]} = Y^{[1]} + Y \tag{26}$$

$$Y^{[3]} = LayerNorm(Y^{[2]}) \tag{27}$$

5.4 输入输出耦合的多头注意力机制

此处的多头注意机制的 Q, K, V 的来源与之前有所不同, 因此。其中 Q(需要查询的信息) 来源于解码器的 $Y^{[3]}$, K, V 来自编码器的 $X^{[6]}$

$$\begin{cases} Q = Y^{[3]}W^Q \\ K = X^{[6]}W^K \\ V = X^{[6]}W^V \end{cases} \tag{28}$$

其中 $Y^{[3]} \in \mathbb{R}^{n_0 \times d_m}$,$X^{[6]} \in \mathbb{R}^{n \times d_m}$, $W^Q, W^K, W^V \in \mathbb{R}^{d_m \times d_m}$。

通过多头注意力机制得到输入和输出的耦合信息 $Y^{[4]}$:

$$Y^{[4]} = MultiHead(Q, K, V) \tag{29}$$

其中 $Y^{[4]} \in \mathbb{R}^{n_0 \times d_m}$ 为该多头注意力机制的输出。

5.5 残差连接 & 层归一化

$$Y^{[5]} = Y^{[4]} + Y^{[3]} \tag{30}$$

$$Y^{[6]} = LayerNorm(Y^{[5]}) \tag{31}$$

5.6 前馈网络

前馈网络采用全连接网络 DNN, 以实现以下映射:

$$DNN_{\theta_y} : \mathbb{R}^{d_m} \to \mathbb{R}^{d_m} \tag{32}$$

$$Y^{[7]}_{i,:} = DNN_{\theta_y}(Y^{[6]}_{i,:}) \tag{33}$$

其中 θ_x 是可训练的参数。

5.7 残差连接 & 层归一化

$$Y^{[8]} = Y^{[7]} + Y^{[6]} \tag{34}$$

$$Y^{[9]} = LayerNorm(Y^{[8]}) \tag{35}$$

6 输出

输出包含线性化 Linaer 和 softmax 两部分，最终输出词的概率分布，其中线性化如下式所示：

$$Y^{[10]} = Y^{[9]}W_{out} \tag{36}$$

其中 $W_{out} \in \mathbb{R}^{d_m \times N_0}$,$n_0$ 即为译文的字典长度。随后对 $Y^{[10]}$ 的每一行进行 softmax：

$$Y^{[11]} = softmax(Y^{[10]}) \tag{37}$$

其中 $Y^{[11]} \in \mathbb{R}^{n_0 \times N_0}$ 为最终输出。

7 损失函数

对第 i 个单词，损失函数为对训练出译文的 $Y^{[11]}$ 和真实的译文 Y^* 做交叉熵：

$$L_i = CrossEntropy(Y^{[11]}_{i,:}, Y^*_{i,:}) \tag{38}$$

因此总的损失函数为：

$$L = \frac{1}{n_0}\sum_{i=0}^{n_0} L_i \tag{39}$$

8 测试模式

在训练模式下，编码器和解码器都是并行运行；而在测试模式下，编码器并行运行，但解码器串行运行，或者说逐字翻译。下式展示了所述的逐字翻译过程：

$$Y^{P[k]} \rightarrow Y^{P[k+1]} \tag{40}$$

其中 $Y^{P[k]}$ 代表在翻译预测过程（P）中第 k 次翻译的结果，其包含翻译的 k 个单词。而第 k+1 次的翻译需要以 $Y^{P[k]}$ 作为输入，即所有之前已经翻译过单词的作为解码器输入，得到新的 k+1 个单词的翻译预测结果。

以上的迭代过程从输入解码器的 START 标签开始，到解码器预测输出 END 标签为止。

9 Vision Transformer(ViT) 概述

9.1 网络结构

ViT[2] 将处理 1D 数据的 Tramsformer 应用在了处理 2D 数据的图像分类领域。如下图3所示,ViT 的模型图像分割为固定大小的 patch，并对每个 patch 进行嵌入和位置编码，并将生成的向量序列提供给 Transformer 中的编码器。此外为了执行分类，ViT 向序列中添加额外的可学习 “分类标记”。令图片为 $x \in \mathbb{R}^{H\times W\times C}$，其中 H, W, C 分别代表图片的长度尺寸，宽度尺寸和通道数量。为了方便编码器处理，将 X 展平并分割成 N 个长宽均为 P 的 patch，即：

$$x_p = [x_p^1, x_p^2, \cdots, x_p^N]^T \tag{41}$$

其中 $x_p \in \mathbb{R}^{N\times P^2C}$ 代表展平后的 x，$x_p^i \in \mathbb{R}^{P^2C}$ 为其中第 i 个 patch。$N = HW/P^2$

在数据的预处理部分进行嵌入和位置编码，并加入可学习的分类标签：

$$z_0 = [x_{class}, x_p^1 E, x_p^2 E, \cdots, x_p^N E]^T + E_{pos} \tag{42}$$

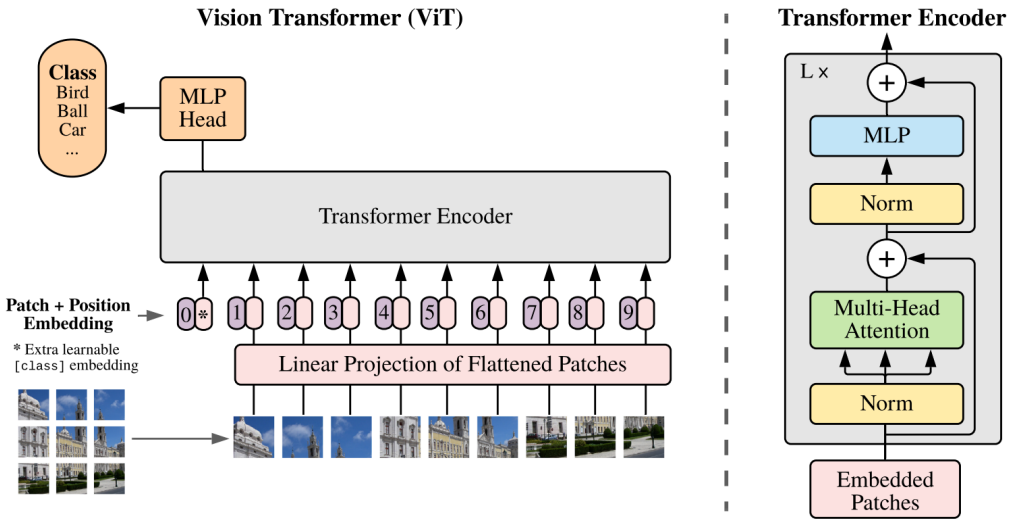

图 3: ViT 结构

其中 z_0 为构造的第 0 次迭代的向量；$E \in \mathbb{R}^{P^2C\times D}$ 是图片嵌入的映射矩阵；$E_{pos} \in \mathbb{R}^{(N+1)\times D}$ 为位置编码；$x_{class} \in \mathbb{R}^D$ 为可学习的分类标签。

从 z_0 开始进行下式43，44的迭代：

$$z_l^{'} = MSA(LN(z_{l-1})) + z_{l-1}, l = 1 \cdots L \tag{43}$$

$$z_l = MLP(LN(z_l^{'})) + z_L^{'}, l = 1 \cdots L \tag{44}$$

其中43首先进行层归一化 (Layer Normalization, LN)，再进行多头自注意力机制 (Multi-head self-attention, MSA)，最后进行残差连接。式44 首先进行层归一化，进行多层感知器 (Multilayer Perceptron, MLP), 最后进行残差连接。

经过以上迭代最终得到 $z^L = [z_1^0, z_L^1, \cdots, z_L^N] \in \mathbb{R}^{N+1}$，从中取 z_L^0，即图片的标签部分进行预测：

$$y = LN(z_L^0) \tag{45}$$

$$y^s = softatmax(yW^F) \tag{46}$$

其中 y^s 为计算的类别，$W^F \in \mathbb{R}^{D \times N_c}$ 为可训练参数，N_c 即为分类的数量。

9.2 网络性能

下表展示了 ViT 与其他 SOTA 在一系列数据集上的比较。表中数值代表了精度的平均值和标准偏差，三次微调运行的平均值。在 JFT-300M 数据集上预训练的 ViT 模型在所有数据集上都优于基 ResNet，同时预训练所需的计算资源大大减少。

	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 1: ViT 与 SOTA 的图像分类基准进行比较

此外注意到 ViT 也有一定的缺陷。下图4展示了 ViT 和 ResNet 在不同量级的数据集下的预测结果。当数据集越小，ResNet 性能越好，同时比 ViT 稳定的快；只有当数据集较大时，ViT 性能才超过 RestNet。

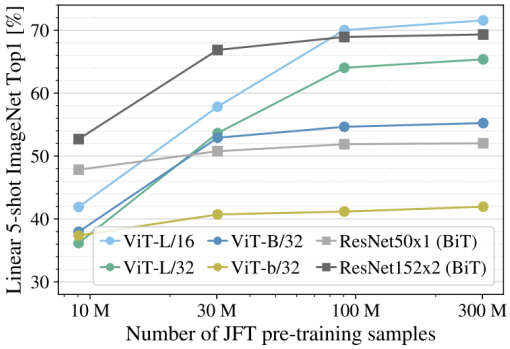

图 4: ViT 结构

10 自我评分与总结

10.1 自我评分

自我评分:90/100

评分理由:

(1) 本报告较为忠实地还原了 Tranformer 和 ViT 的手稿，内容详略得当，公式较为完整清晰。同时结合公式对图片进行重新编辑，较为详细地展示了公式在结构图中的对应位置 (如图3.12所示)。

(2) 本报告在结合课程回放基础上，也结合了“李沐学 AI”和“李宏毅机器学习/深度学习”中的相关内容进行补充。

(3)

10.2 总结

我觉得老师的课程还是很不错的，特别是频率原则的讲解令人印象深刻。但是有以下一些建议和意见想提出:

(1) 直接看录播的体验不是特别好，还是希望能直接讲课;

(2) 强化学习部分花了很多课时讲解，但这部分知识量比较大，而且和前序和后序课程没有特别紧密的联系。此外这部分的讲解部分对我来说还是具有一定难度，最后能学到并理解的部分不是特别多。

参考文献

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.