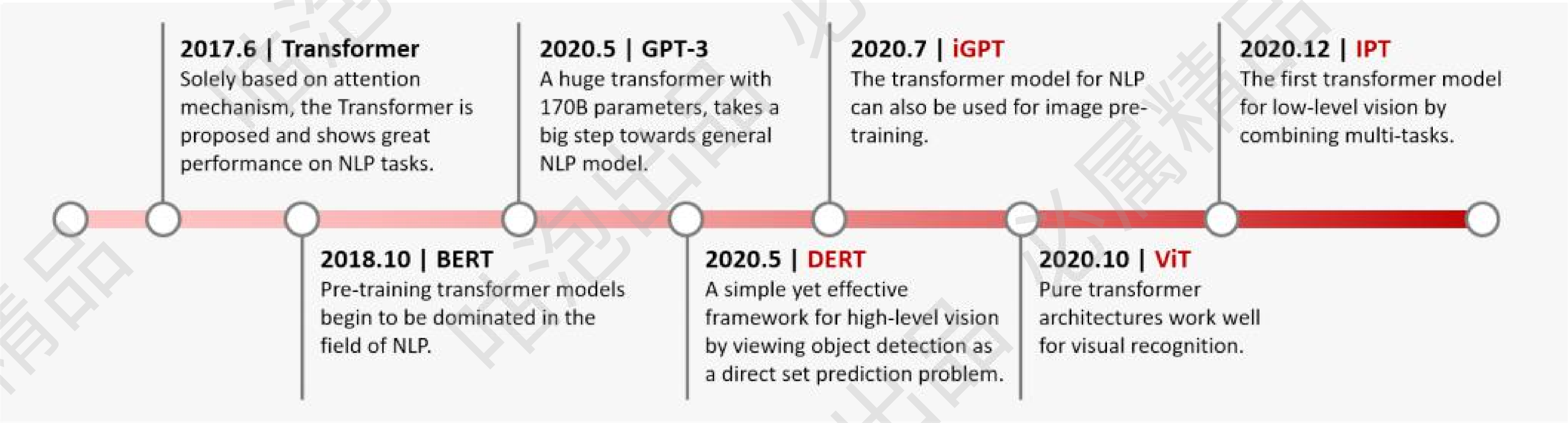
Transformer

- 为啥这么火呢?
  - NLP领域大哥大级别，一统天下好多年了
  - CV界新秀，开场即巅峰；满级大号直接上场
  - 新一代backbone，可直接套用在各项下游任务中
  - 分类，分割，检测各项任务均刷榜；

Transformer

发家史

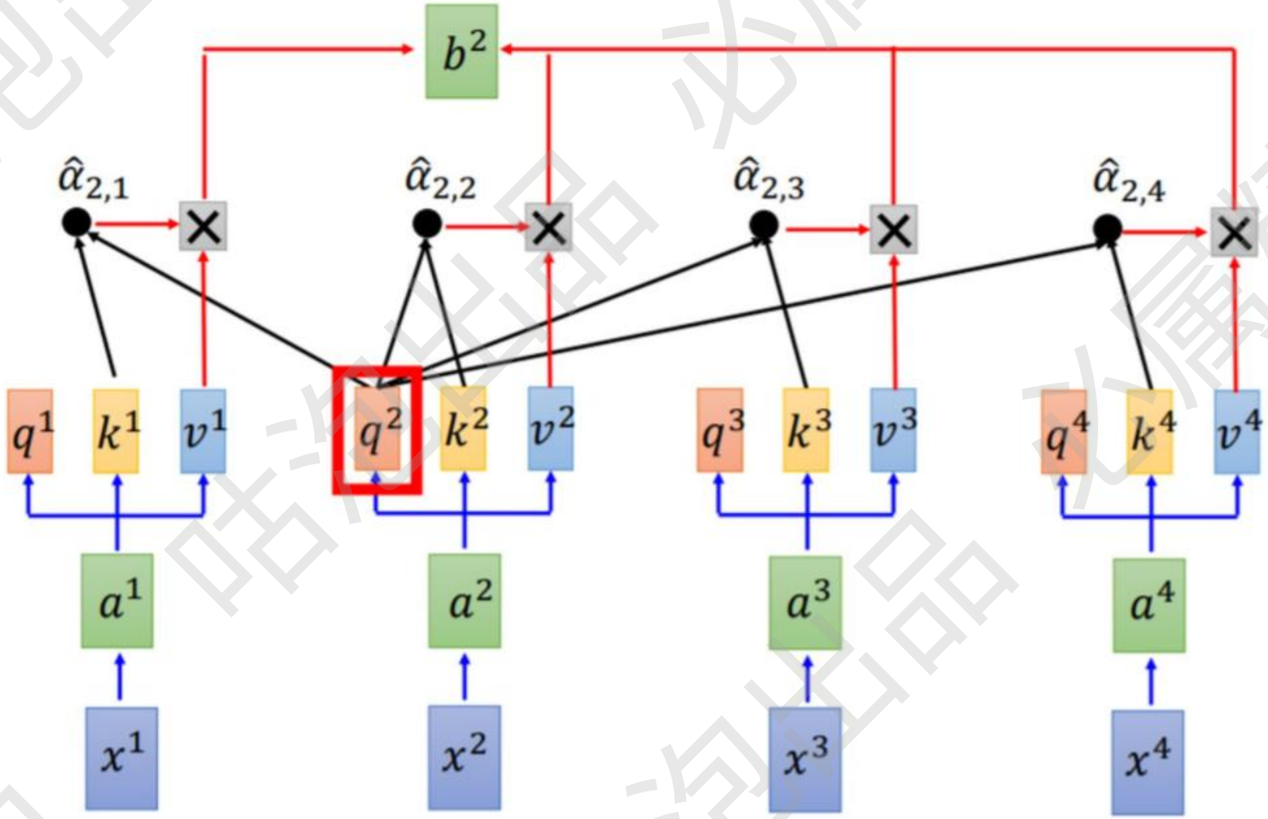
17年NLP大爆发，20年轰动CV圈（明星是怎么练成的）



Transformer

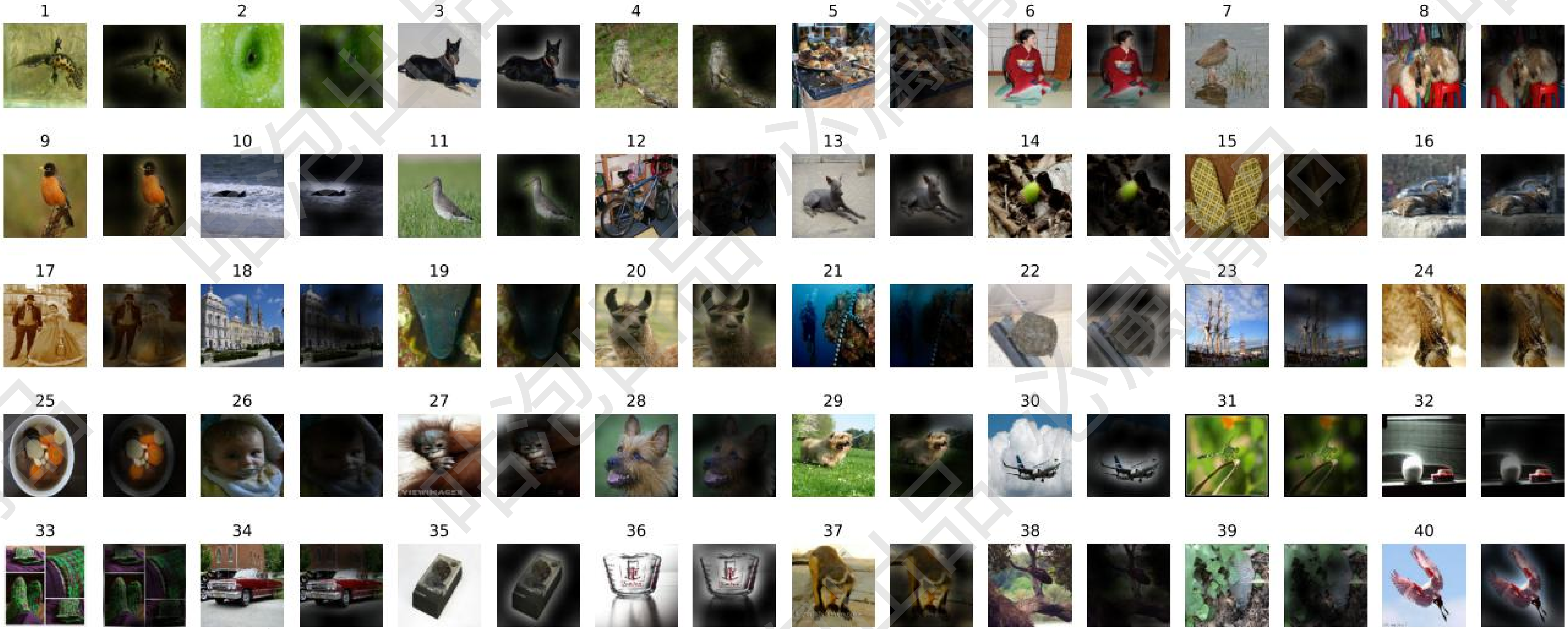
- 回忆一下咋干活的来着

  - 就是重新组合各大输入向量，得到更完美的特征



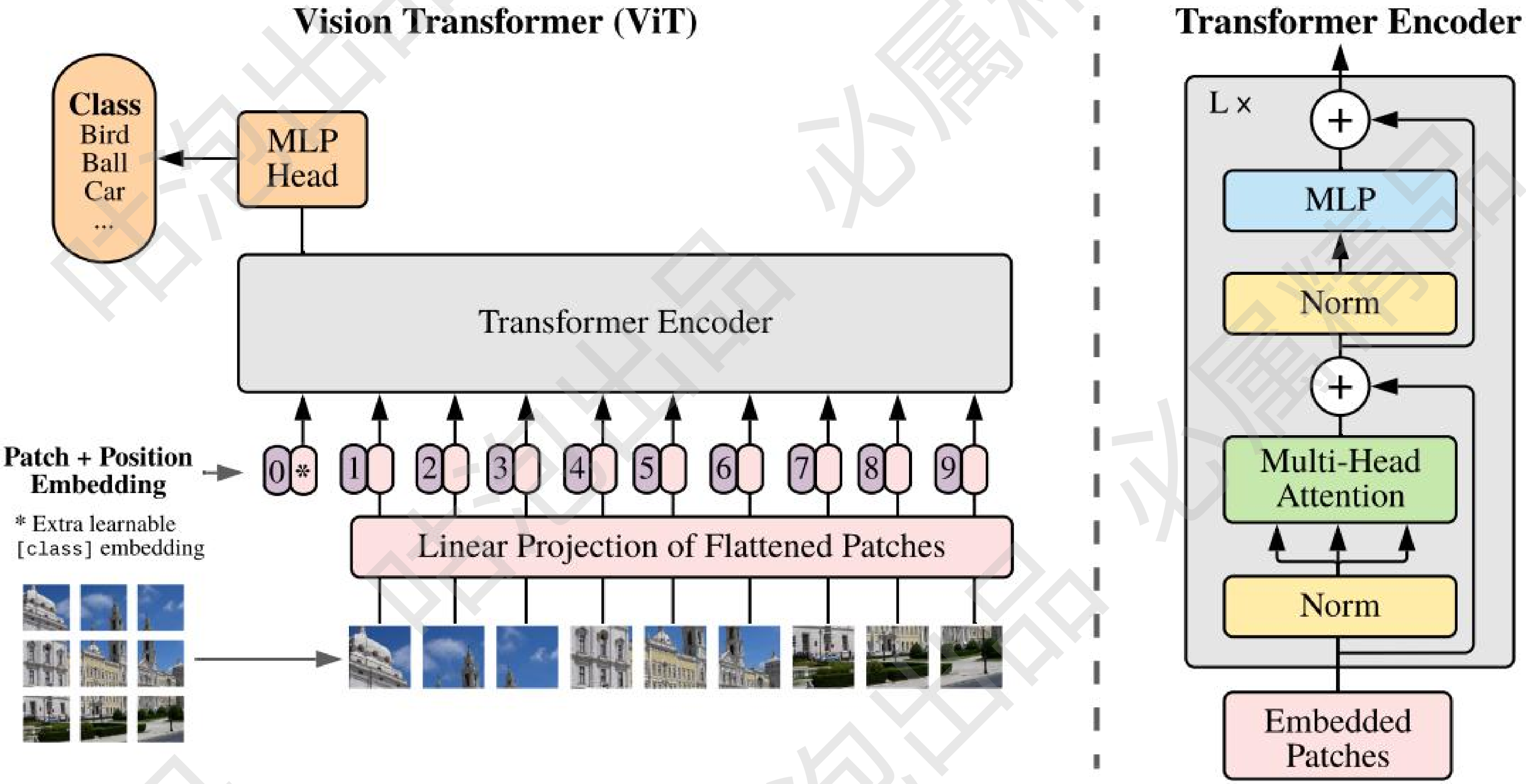
Transformer

视觉中的Attention



Transformer

整体架构分析



Transformer

- CNN最大的问题是什么?
  - 格局，眼界；这两词往上一整，就把这页PPT显得高的上了!
  - CNN中的格局和眼界是什么？不就是感受野嘛!
  - 想要获得大的感受野（全局的信息）就必须堆叠很多层卷积
  - 这问题就来了，不断卷积+池化的操作感觉有点麻烦还不一定好

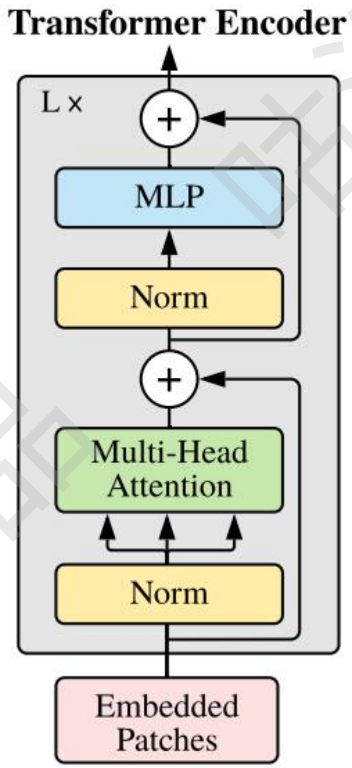
Transformer

- transformer的格局
  - 根本不需要堆叠，直接就可以获得全局信息
  - CNN就像一个穷秀才考状元；transformer直接当驸马爷了
  - 但是驸马爷也不是好当的，银子（训练数据）得到位才行
  - 纯transformer结构已经在CV界起义了，CNN是否会沦陷？

Transformer

公式介绍

- 输入patch（P*P*C）经过全连接E得到(P*P*D)
- N+1表示额外找一个patch表示分类特征，位置编码也同理
- 输出矩阵维度与输入矩阵维度一致，重复多层即可



$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \quad (1)$$

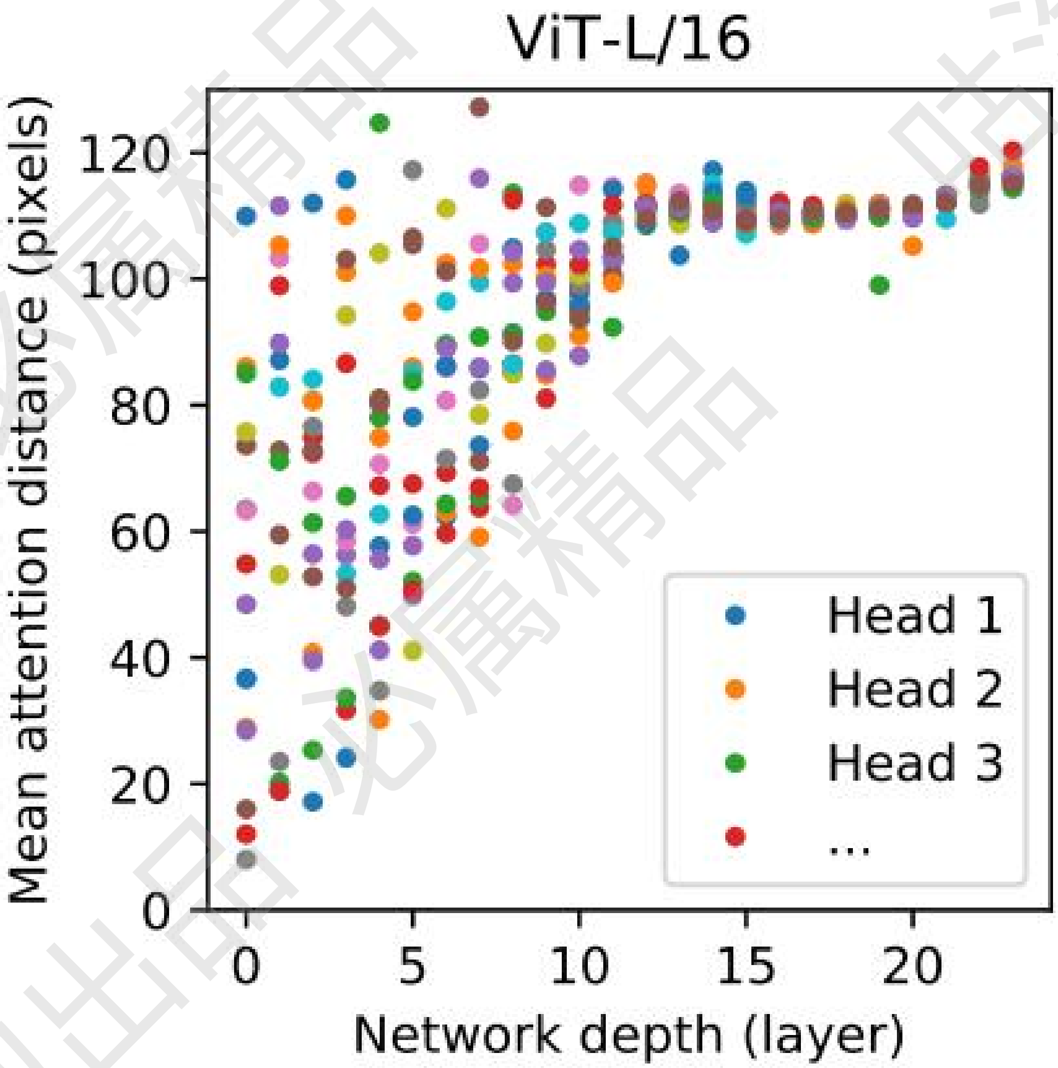
$$\mathbf{z'}_\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1\ldots L \quad (2)$$

$$\mathbf{z}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, \qquad \ell = 1\ldots L \quad (3)$$

$$\mathbf{y} = \mathrm{LN}(\mathbf{z}_L^0) \quad (4)$$

Transformer

- 格局有多大呢?
  - 可以对比CNN来感受野
  - 浅层就能捕获较大范围信息
  - 可能5层就顶CNN30层了
  - 全局信息丰富，更好理解整个图像



Transformer

- 位置编码

- 结论：编码有用，但是怎么编码影响不大，干脆用简单的得了

- 2D（分别计算行和列的编码，然后求和）的效果还不如1D的
  每一层都加共享的位置编码也没啥太大用

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

Transformer

- 效果分析（/14表示patch的边长是多少）

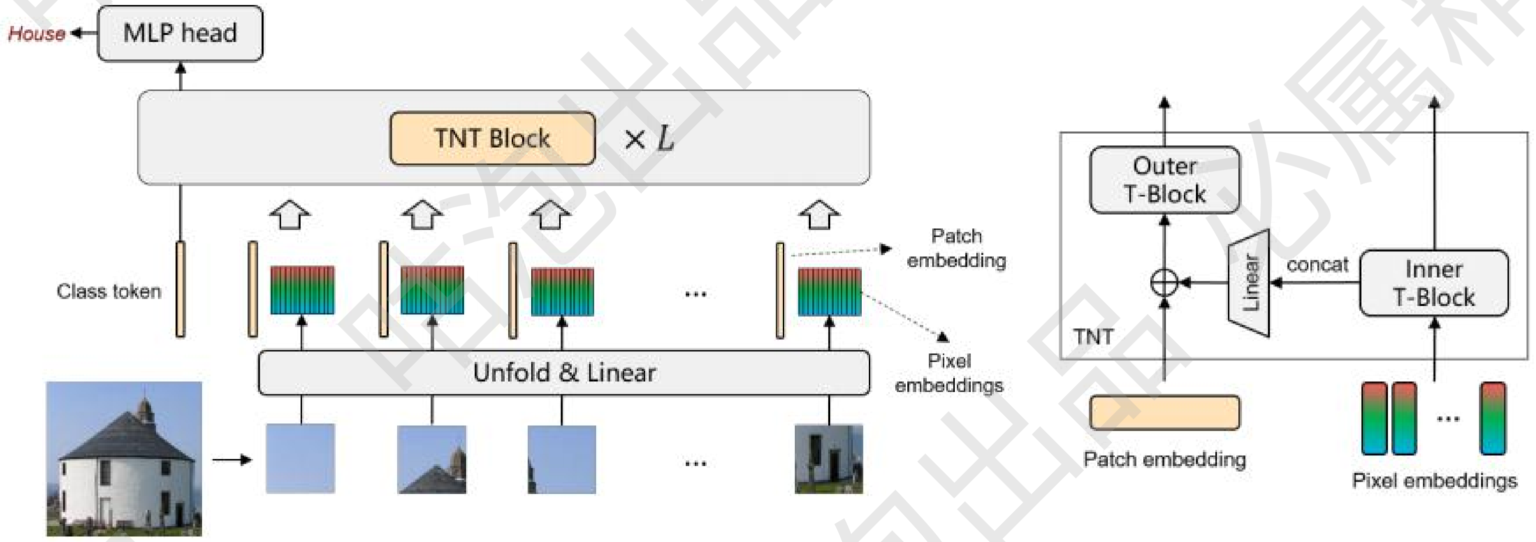
name	Epochs	ImageNet	ImageNet ReaL	CIFAR-10	CIFAR-100	Pets	Flowers	exaFLOPs
ViT-B/32	7	80.73	86.27	98.61	90.49	93.40	99.27	55
ViT-B/16	7	84.15	88.85	99.00	91.87	95.80	99.56	224
ViT-L/32	7	84.37	88.28	99.19	92.52	95.83	99.45	196
ViT-L/16	7	86.30	89.43	99.38	93.46	96.81	99.66	783
ViT-L/16	14	87.12	89.99	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

Transformer

- TNT：Transformer in Transformer

  - VIT中只针对pathch进行建模，忽略了其中更小的细节

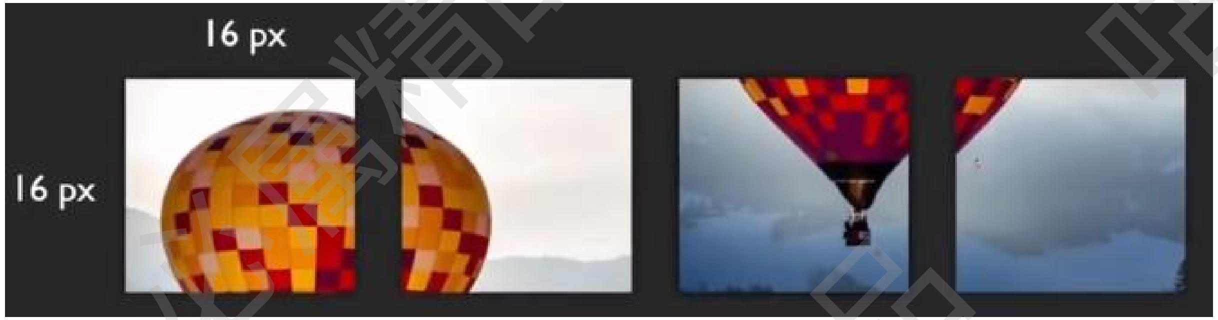
  - 实验结果要比VIT强一些，做的更细了



(a) TNT framework.　(b) TNT block.

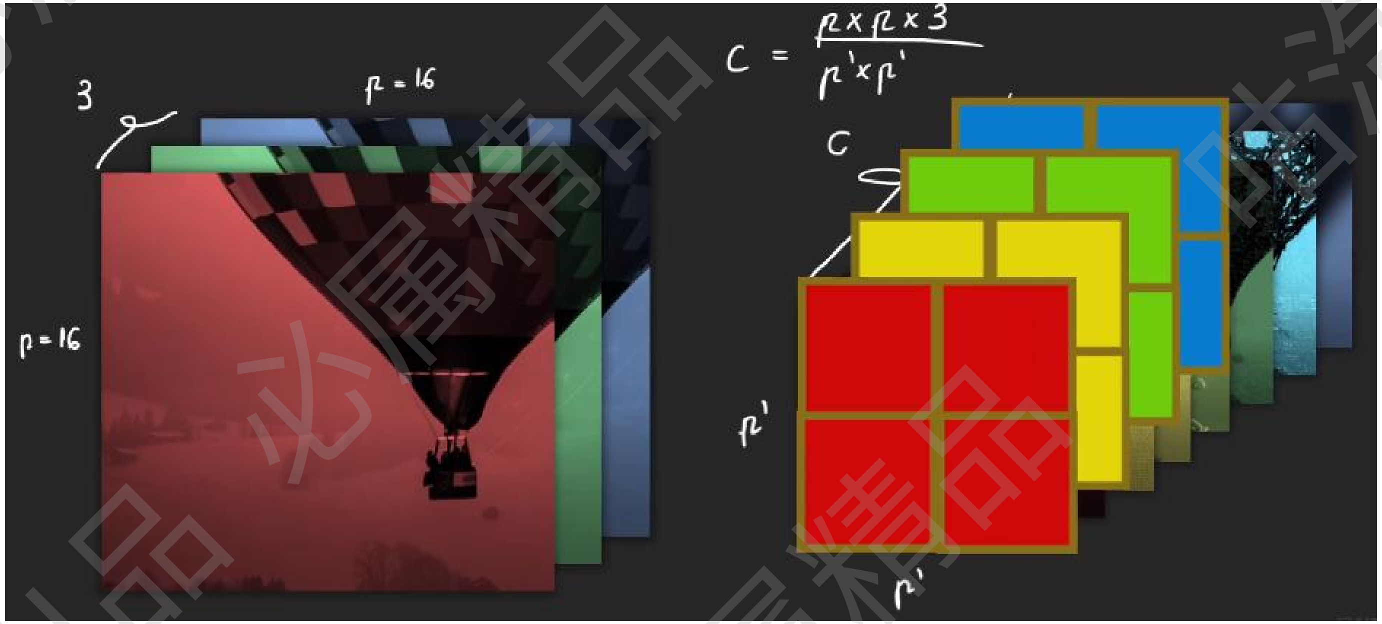
Transformer

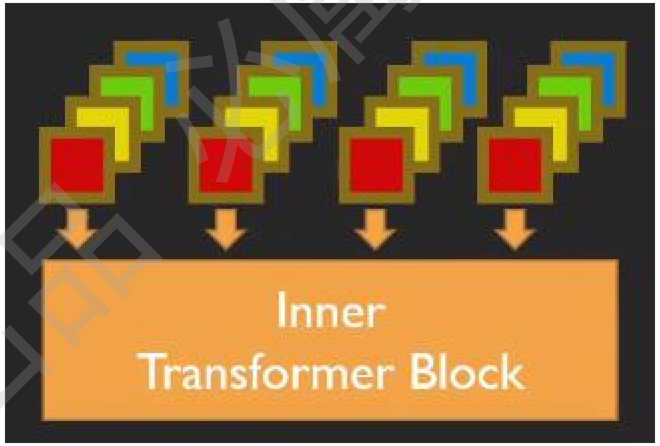
- TNT的基础组成
  - 外部transformer处理的序列：
  - 内部transformer:
  - 重组成多个超像素（4个像素点）
  - 把重组的序列继续做transformer



Transformer

- TNT的序列构建
  - 内部序列构建：
  - 重构内部序列
  - 组合后的序列继续transformer
  - 这里的transformer跟传统的一摸一样

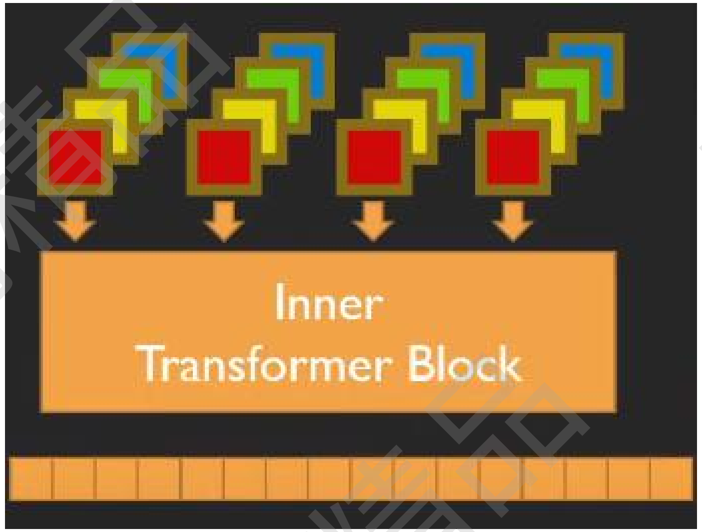


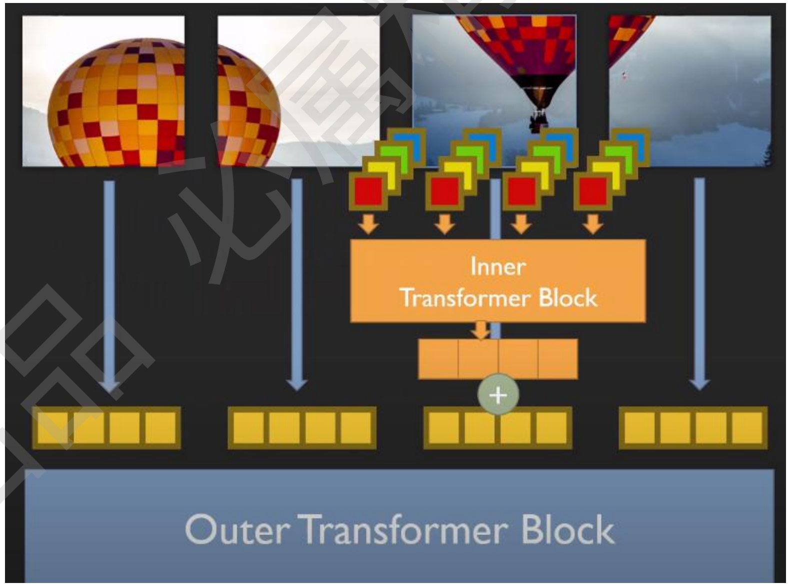


Transformer

TNT的基本计算

- 内部transformer重组成新的向量：
- 新向量再通过全连接改变输出特征大小
- 内部组合后的向量与patch编码大小相同
- 最后与原始输入pathch向量进行相加





Transformer

- TNT位置编码实验

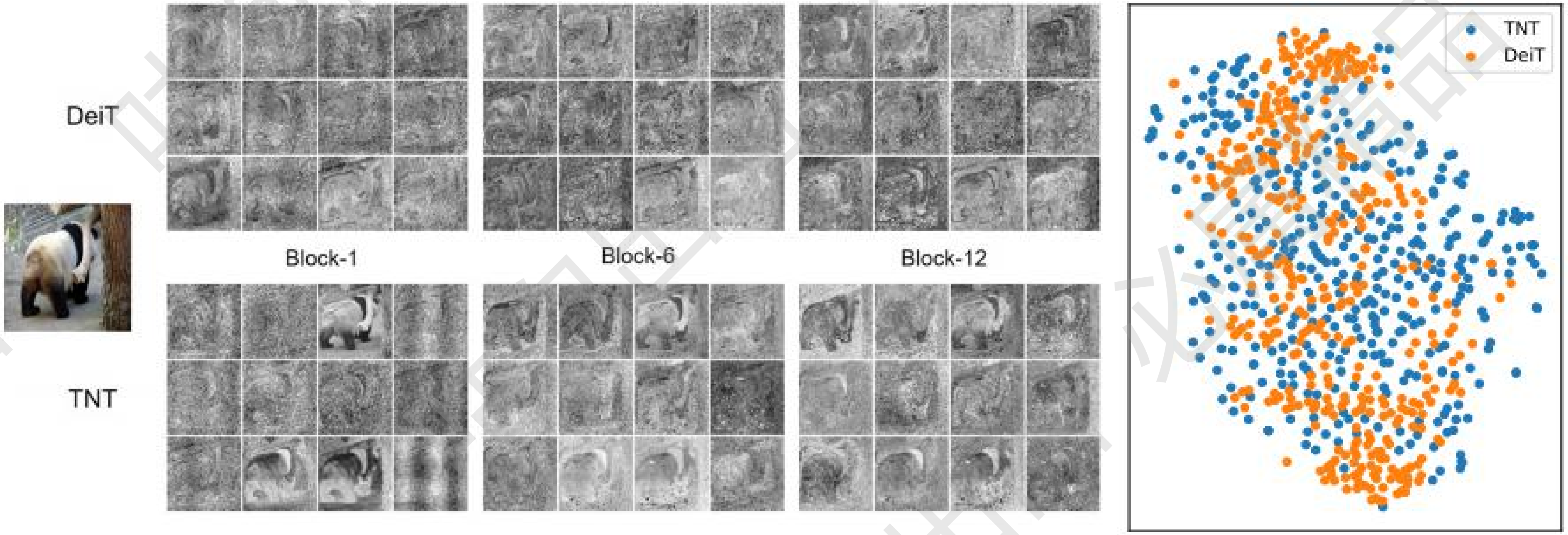
  - 内外兼修，都加编码效果最好

Model	Patch position encoding	Pixel position encoding	Top-1 (%)
TNT-S	✗	✗	80.5
	✓	✗	80.8
	✗	✓	80.7
	✓	✓	81.3

Transformer

- TNT的PatchEmbedding的可视化

  特征更鲜明，分布更多样性



(a) Feature maps in Block-1/6/12. (b) T-SNE of Block-12.