



多模态模型 - CLIP

作者: Calvin

QQ: 179209347

Mail: 179209347@qq.com

介绍

笔记简介:

- 面向对象: 深度学习初学者
- 依赖课程: **线性代数, 统计概率**, 优化理论, 图论, 离散数学, 微积分, 信息论

知乎专栏:

<https://zhuanlan.zhihu.com/p/693738275>

Github & Gitee 地址:

https://github.com/mymagicpower/AIAS/tree/main/deep_learning

https://gitee.com/mymagicpower/AIAS/tree/main/deep_learning

* 版权声明:

- 仅限用于个人学习
- 禁止用于任何商业用途

CLIP (Contrastive Language-Image Pre-training)

CLIP (Contrastive Language-Image Pre-training) 是由OpenAI开发的一种多模态（文本和图像）预训练模型。CLIP模型通过学习如何对文本和图像进行对比，从而实现跨模态的理解。这种对比学习的方法使得CLIP能够在没有任何监督标签的情况下学习到文本和图像之间的语义关系。

CLIP模型的核心思想是**将文本和图像嵌入到一个共同的语义空间中**，使得相关的文本描述和图像内容在这个空间中的表示彼此靠近，而不相关的则远离。这种设计使得CLIP模型能够在各种任务上表现出色，如图像分类、图像检索、文本分类等。

CLIP模型的特点：

- 多模态嵌入
- 对比学习
- 训练数据
- 自监督学习
- 跨任务应用

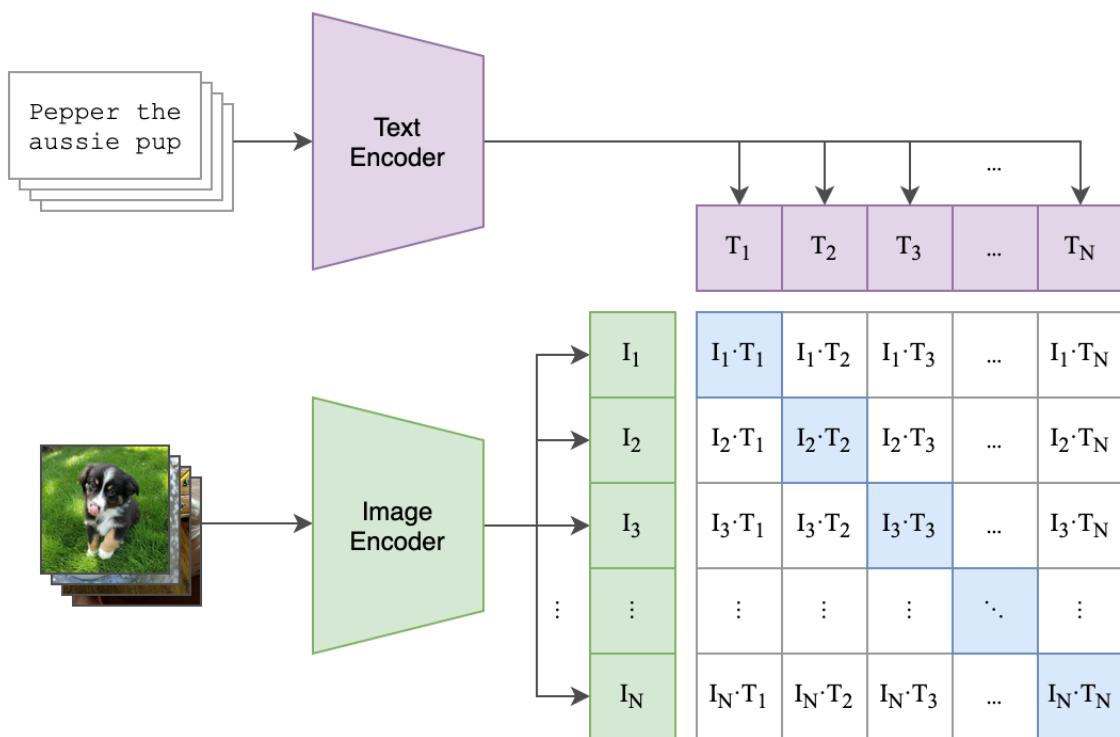


CLIP模型训练

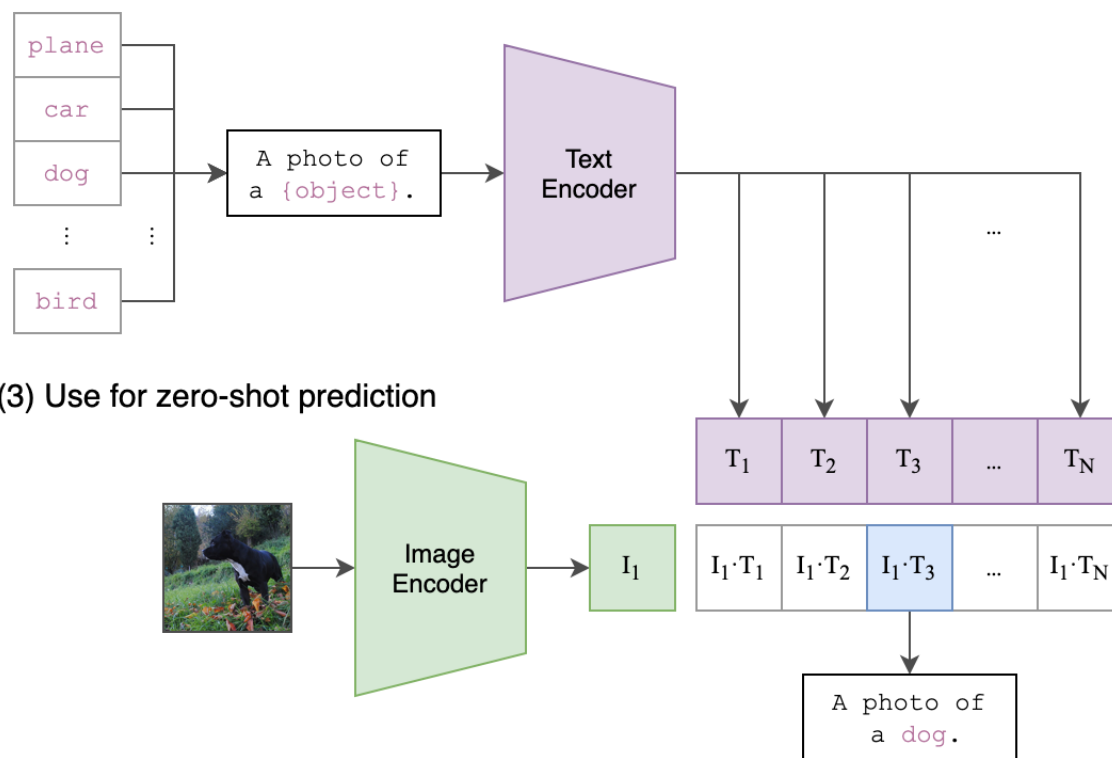
CLIP模型训练分为三个阶段：

- Contrastive pre-training: 预训练阶段，使用图片 - 文本对进行对比学习训练；
- Create dataset classifier from label text: 提取预测类别文本特征；
- Use for zero-shot prediction: 进行 Zero-Shoot 推理预测；

(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction

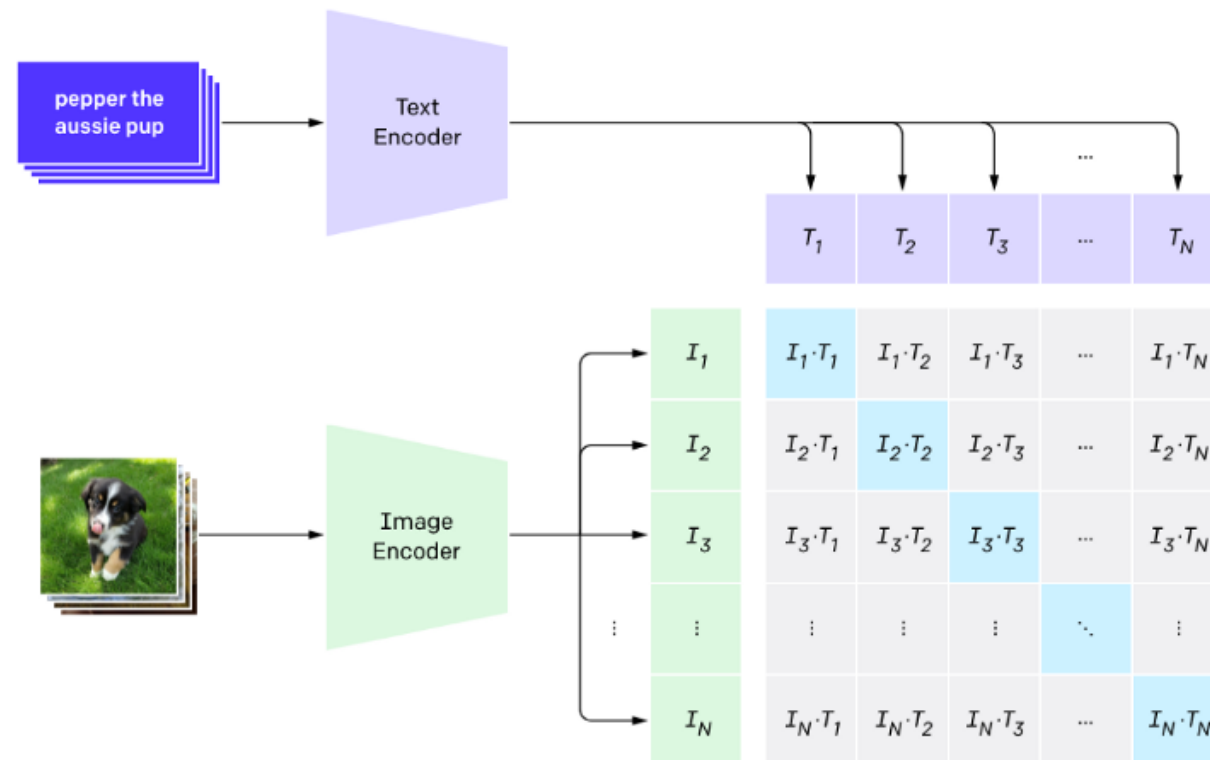
CLIP模型 – 模型训练

CLIP模型使用两种独立的网络结构来作为图像编码和文本编码的主干网络（backbone），其中：

- image_encoder: 负责编码图像的神经网络架构（eg, ResNet, EfficientNet, Transformer, Vision Transformer等）
- text_encoder: 负责编码文本信息的神经网络架构（eg, CBOW 或 BERT等）

训练阶段： 通过将图像和文本分别编码成向量，建立一个相似度矩阵来进行训练。训练的目标是让匹配的图文对的相似度高于不匹配的图文对，类似于分类任务。

1. Contrastive pre-training

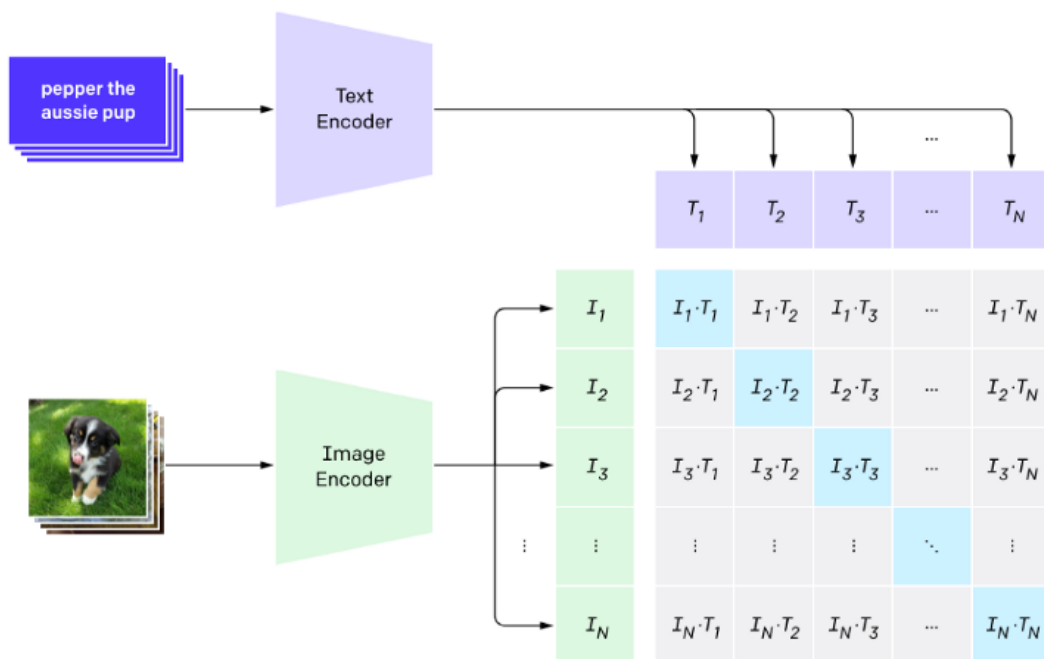


CLIP模型 – 模型训练

算法伪代码:

- 首先, 将图像和文本先分别输入一个图像编码器 (image encoder) 和一个文本编码器 (text encoder), 得到图像和文本的向量表示 I_f 和 T_f 。
- 然后, 将图像和文本向量表示映射到一个联合的多模态空间 (joint multimodal space), 得到新的可直接进行比较的图像和文本向量表示 I_e 和 T_e , 计算图像和文本向量之间的cosine相似度。
- 最后, 对比学习的目标函数就是让正样本对的相似度较高, 负样本对的相似度比较低。

1. Contrastive pre-training



```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

CLIP模型 – 推理验证

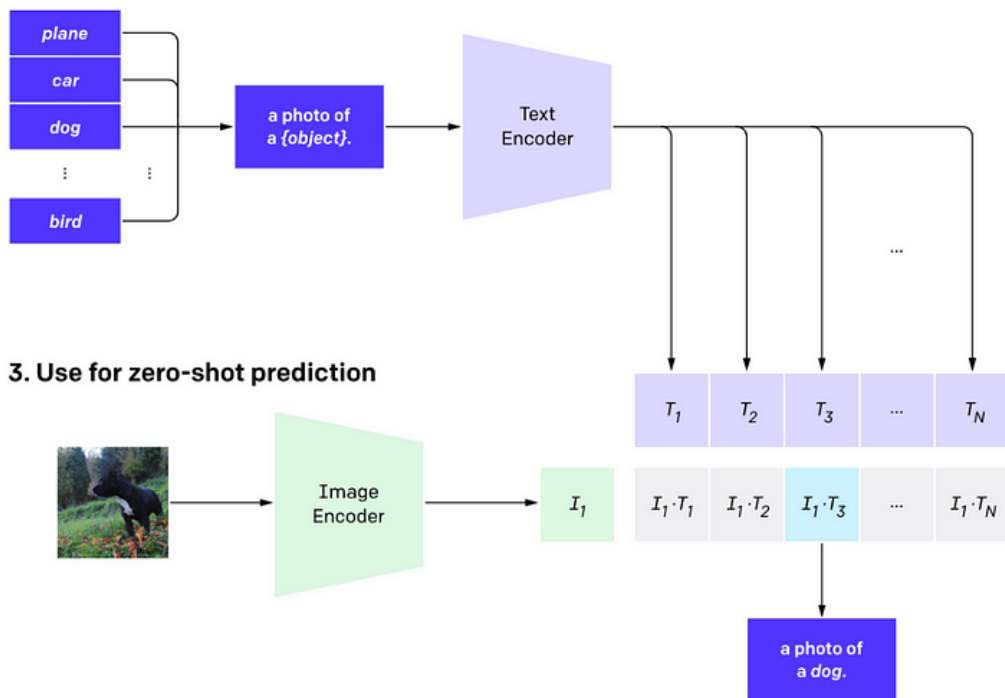
推理验证：用户可以自定义新文本来判断图像是否匹配。通过比较内积值的大小，可以确定它们的匹配程度。这种方法可以应用于各种任务，如判断图像内容或者进行分类。

经典的分类训练只关心模型是否可以正确预测图像的分类标签。

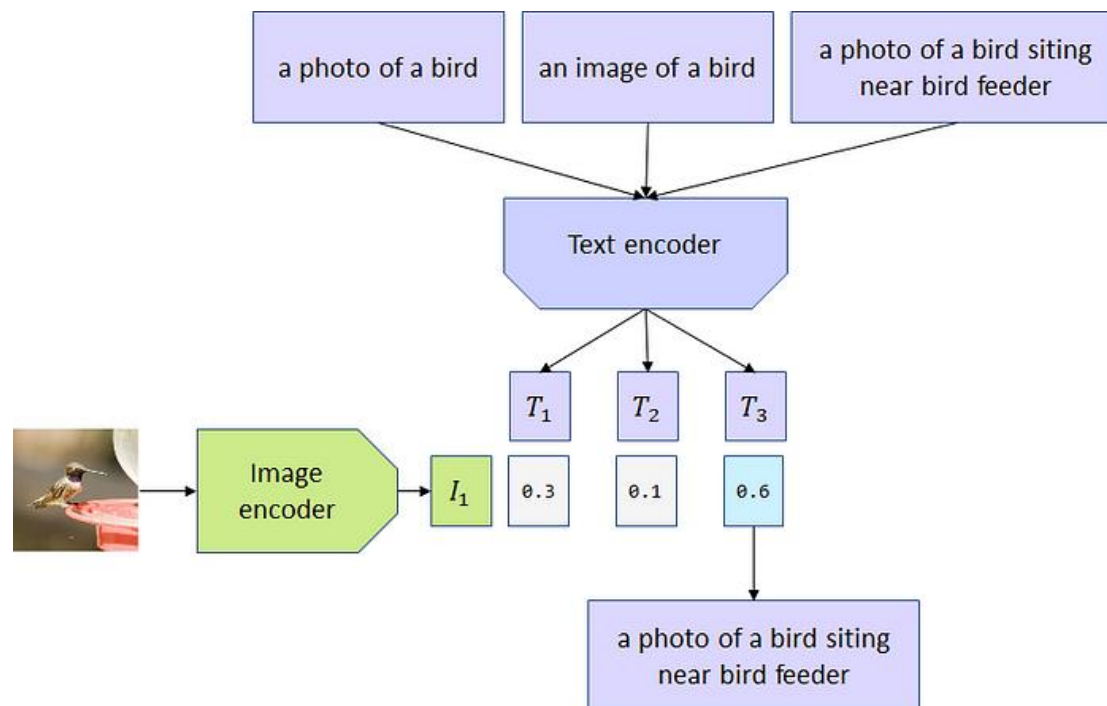
而CLIP模型在大规模数据集上完成的训练，这使得CLIP模型还学习到了图像的各方面信息。

例如，CLIP模型对用于图像描述的单词很敏感。文本 “a photo of a bird”、“a photo of a bird siting near bird feeder” 或 “an image of a bird” 与相同的图像匹配产生的概率是不同的。

2. Create dataset classifier from label text



3. Use for zero-shot prediction

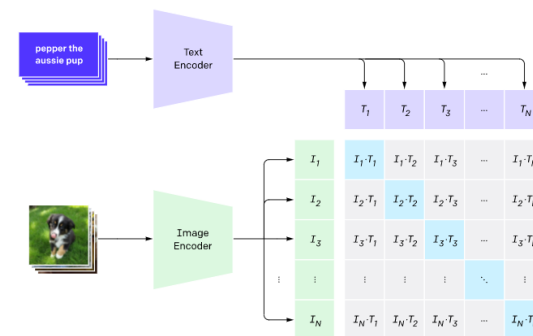


应用举例 - 图像文本跨模态搜索

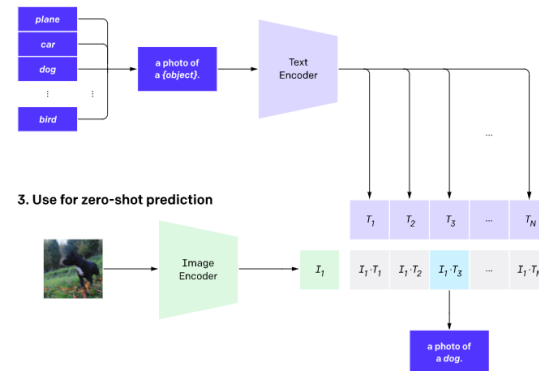
- 特征向量空间 (由图片 & 文本组成)



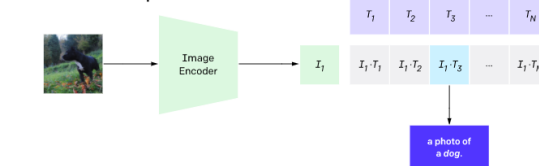
1. Contrastive pre-training



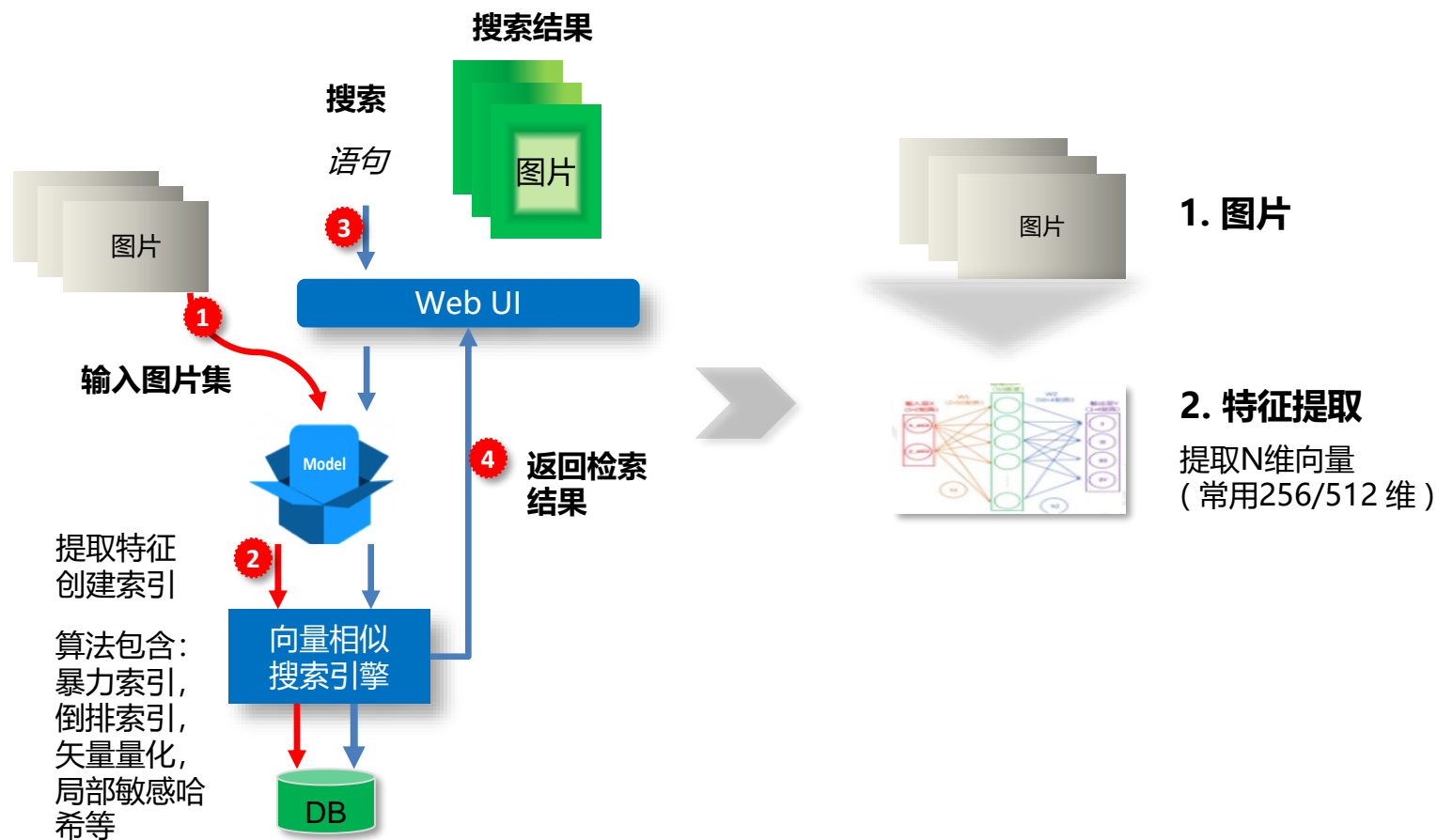
2. Create dataset classifier from label text



3. Use for zero-shot prediction

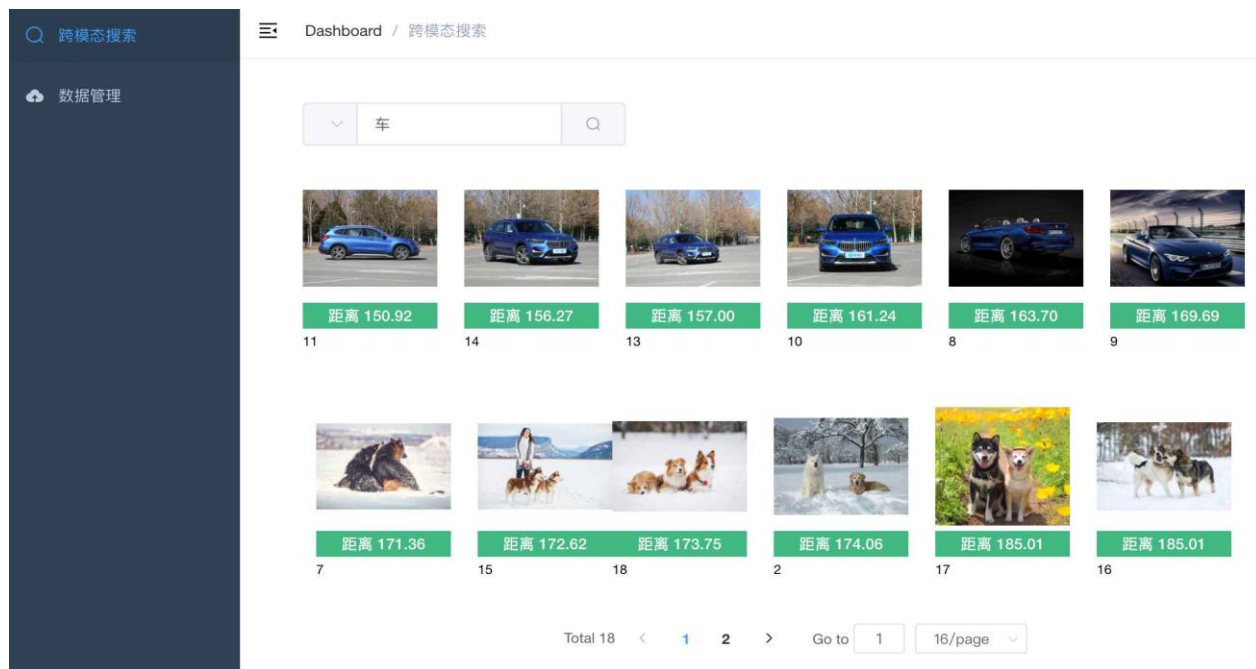


应用举例 - 图像文本跨模态搜索

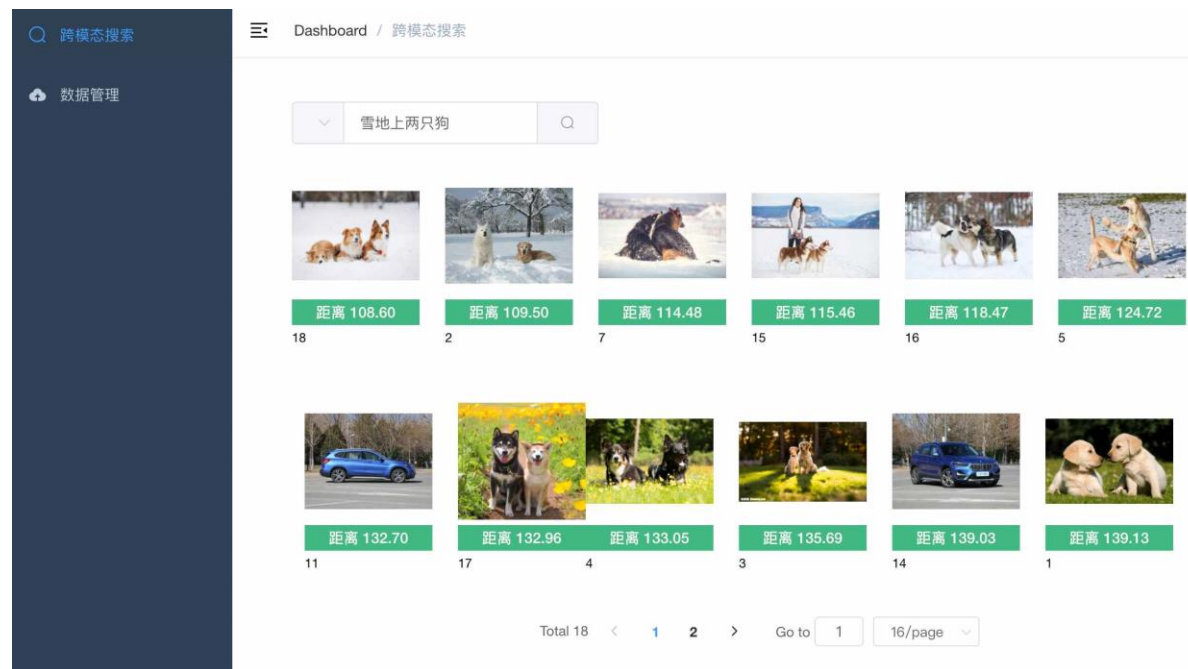


应用举例 - 图像文本跨模态搜索 - 以文搜图

例子1 输入文本：车



例子2 输入文本：雪地上两只狗



应用举例 - 图像文本跨模态搜索 - 以图搜图

跨模态搜索-文本搜图


跨模态搜索-以图搜图

数据管理

Dashboard / 跨模态搜索-以图搜图


TopK: 50

请上传图片或拖入图片




查询

重置




距离 0.00

100




距离 15.88

104




距离 20.05

103




距离 29.31

101




距离 68.61

99




距离 70.17

98




距离 123.44

105




距离 135.31

97



距离 138.65

95



距离 141.66

94

共 18 条

< 1 >

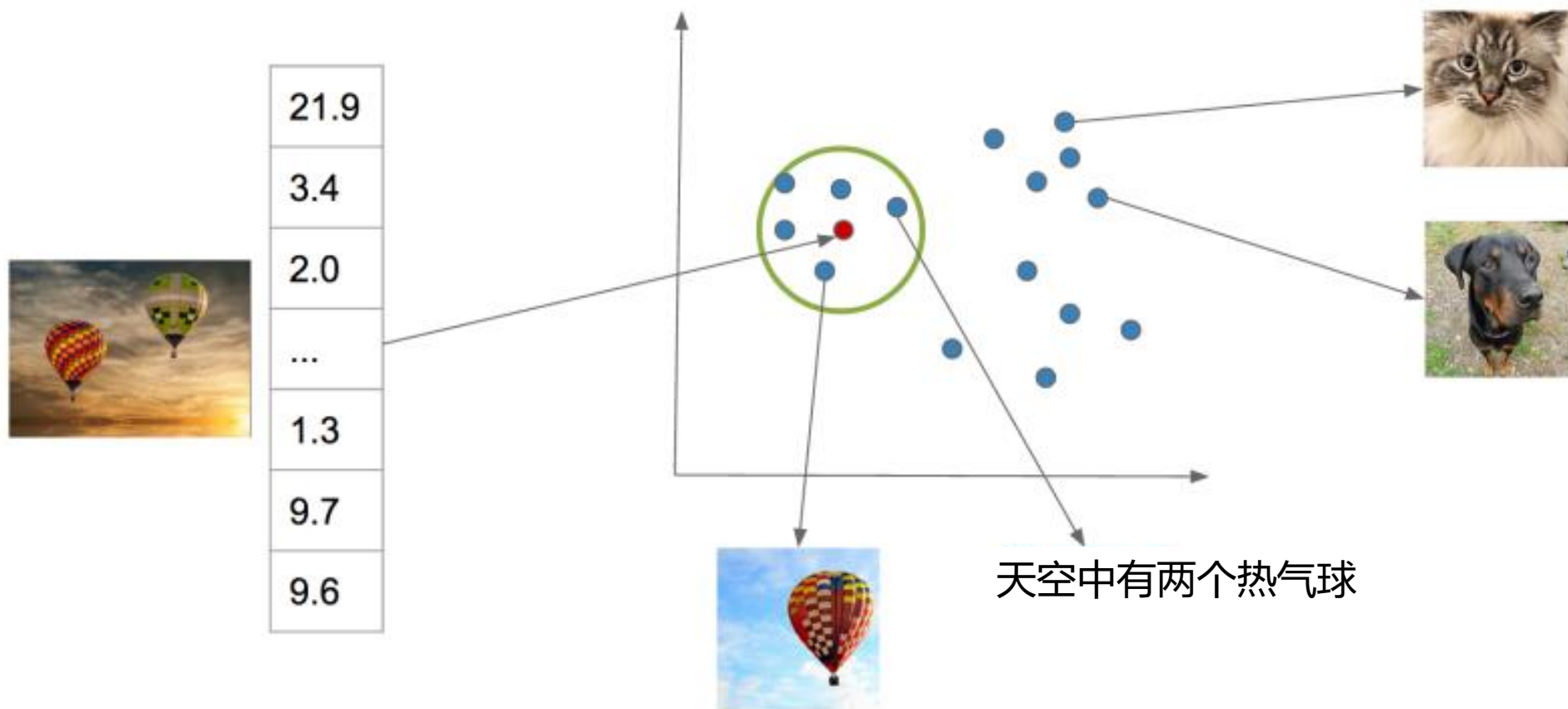
前往 1 页

20条/页

All rights reserved by www.aias.top , mail: 179209347@qq.com

图像&文本的跨模态搜索 1: N 比对 - k-NN搜索

向量搜索例子：把图片转换成向量然后进行k-NN搜索从而实现相似图片匹配功能。



向量数据库 – 索引策略

每种索引都有自己的适用场景，如何选择合适的索引可以简单遵循如下原则：

- 1) 当查询数据规模小，且需要100%查询召回率时，用 FLAT；
- 2) 当需要高性能查询，且要求召回率尽可能高时，用 IVFFLAT；
- 3) 当需要高性能查询，且磁盘、内存、显存资源有限时，用 IVFSQ8H；
- 4) 当需要高性能查询，且磁盘、内存资源有限，且只有 CPU 资源时，用 IVFSQ8。

浮点型向量

距离计算方式	索引类型
欧氏距离 (L2)	• FLAT
	• IVF_FLAT
	• IVF_SQ8
	• IVF_SQ8H
内积 (IP)	• IVF_PQ
	• RNSG
	• HNSW
	• ANNOY

二值型向量

距离计算方式	索引类型
杰卡德距离 (Jaccard)	• FLAT
谷本距离 (Tanimoto)	• IVF_FLAT
汉明距离 (Hamming)	
超结构 (superstructure)	FLAT
子结构 (substructure)	

向量数据库 – 索引策略 – 距离计算方式

欧氏距离 (L2)

欧氏距离计算的是两点之间最短的直线距离。

欧氏距离的计算公式为：

$$d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a}) = \sqrt{\sum_{i=1}^n (b_i - a_i)^2}$$

其中 $\mathbf{a} = (a_1, a_2, \dots, a_n)$ 和 $\mathbf{b} = (b_1, b_2, \dots, b_n)$ 是 n 维欧氏空间中的两个点。

欧氏距离是最常用的距离计算方式之一，应用广泛，适合数据完整，数据量纲统一的场景。

内积 (IP)

两条向量内积距离的计算公式为：

$$p(A, B) = A \cdot B = \sum_{i=1}^n a_i \times b_i$$

假设有 A 和 B 两条向量，则 $\|A\|$ 与 $\|B\|$ 分别代表 A 和 B 归一化后的值。内积更适合计算向量的方向而不是大小。

如需使用点积计算向量相似度，则必须对向量作归一化处理。处理后点积与余弦相似度等价。

假设 X' 是向量 X 的归一化向量：

$$X' = (x'_1, x'_2, \dots, x'_n), X' \in \mathbb{R}^n$$

两者之间的关系为：

$$x'_i = \frac{x_i}{\|X\|} = \frac{x_i}{\sqrt{\sum_{i=1}^n (x_i)^2}}$$

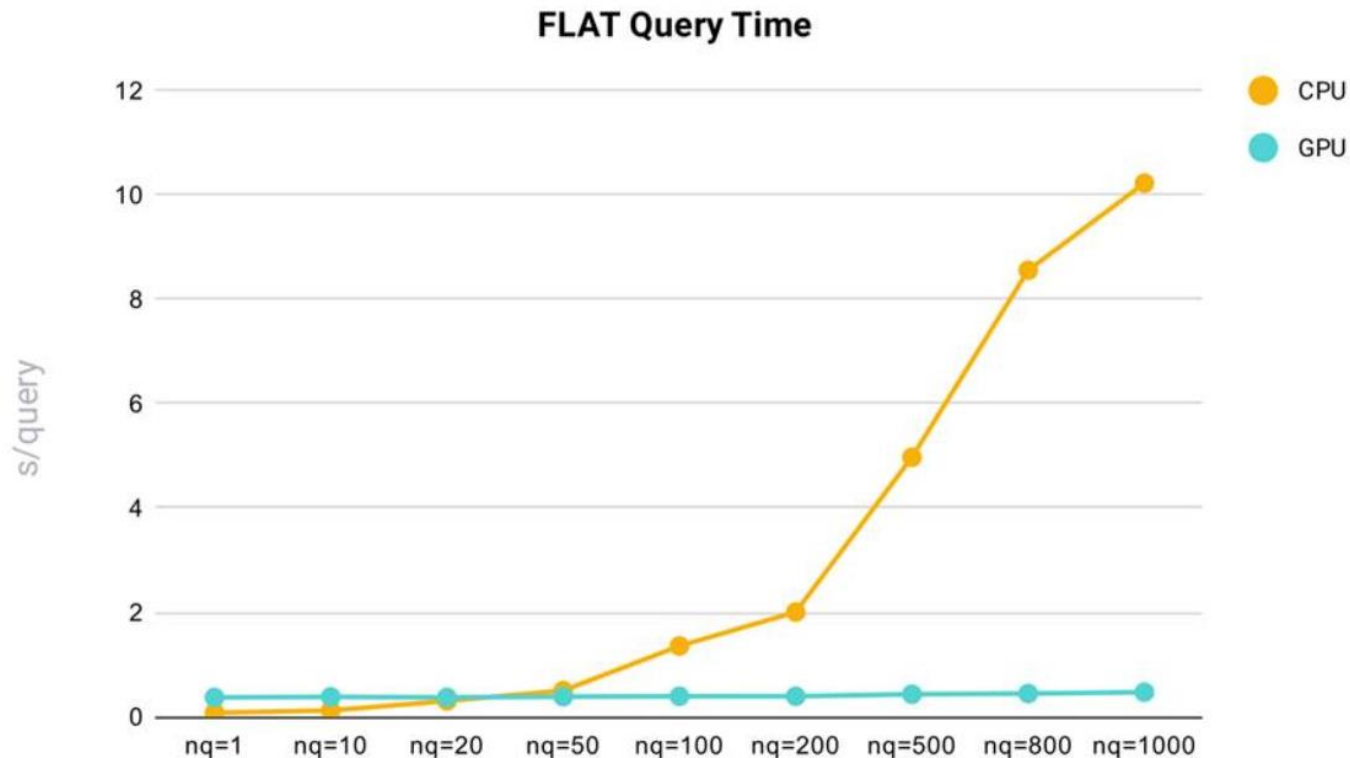
向量数据库 – 索引策略 – FLAT

FLAT 并不是一种真正的索引，但由于它与其它的索引有一致的接口及使用方法，把它视为一种特殊的索引。FLAT 的查询速度在所有的索引中是最慢的，但是当需要查询的次数较少，构建索引的时间无法被有效均摊时，它反而是最有效的查询方式。

优点：

- 100%查询召回率
- 无需训练数据，无需配置任何系统参数，也不会占用额外的磁盘空间

缺点：查询速度慢

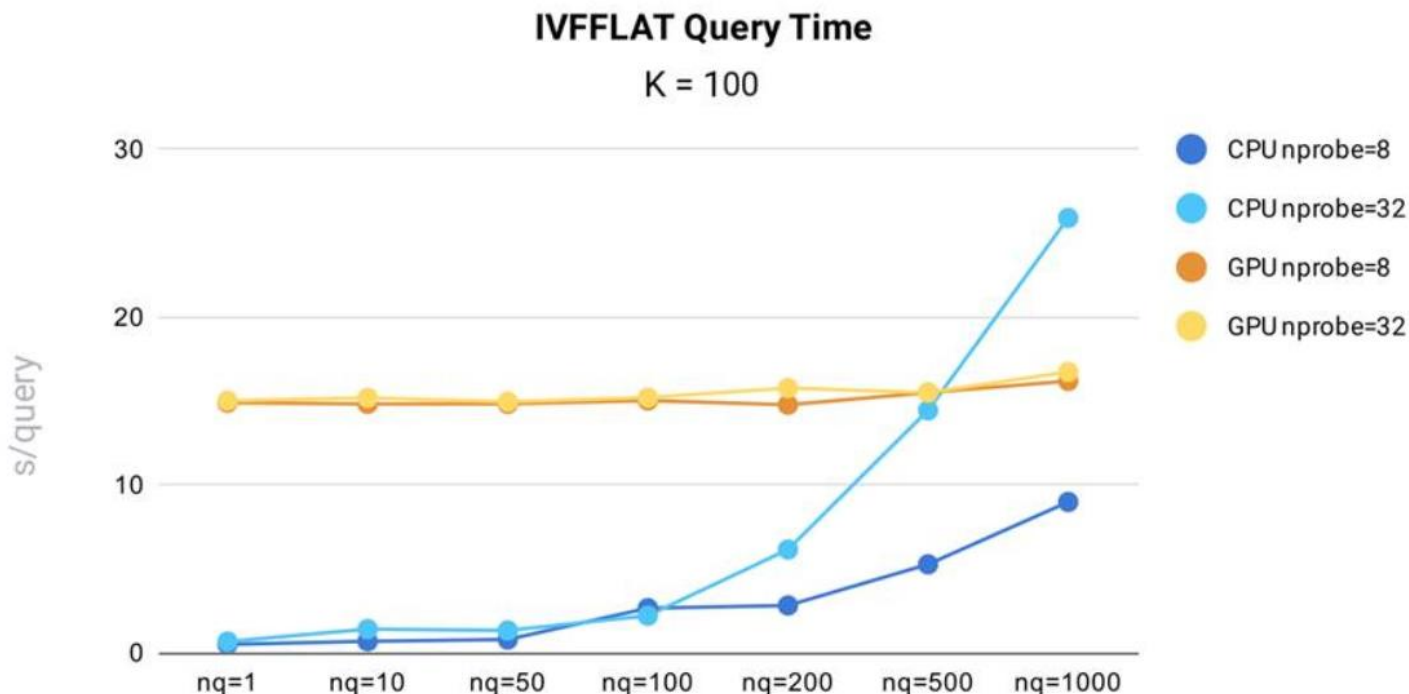


向量数据库 – 索引策略 – IVFFLAT

IVFFLAT 是最简单的索引类型。在聚类时，向量被直接添加到各个分桶中，不做任何压缩，存储在索引中的数据与原始数据大小相同。查询速度与召回率之间的权衡由参数 `nprobe` 来控制。`nprobe` 越大，召回率越高，但查询时间越长。IVFFLAT 是除了 FLAT 外召回率最高的索引类型。

- 优点：查询召回率高
- 缺点：占用空间大

用公开数据集 sift-1b (10亿条128维向量) 建立 IVFFLAT 索引，并分别只用 CPU 或 GPU 做查询，在不同 `nprobe` 参数下测得的查询时间随 `nq` 变化曲线如图：





Thank

You