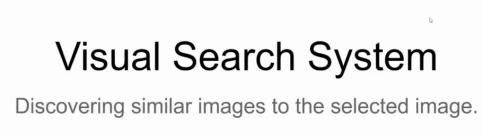
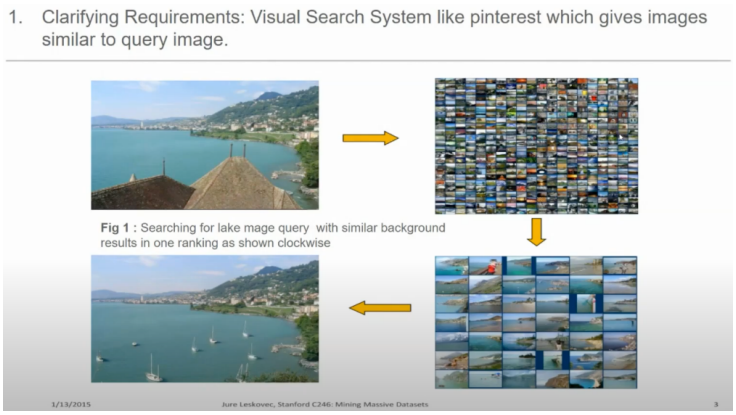
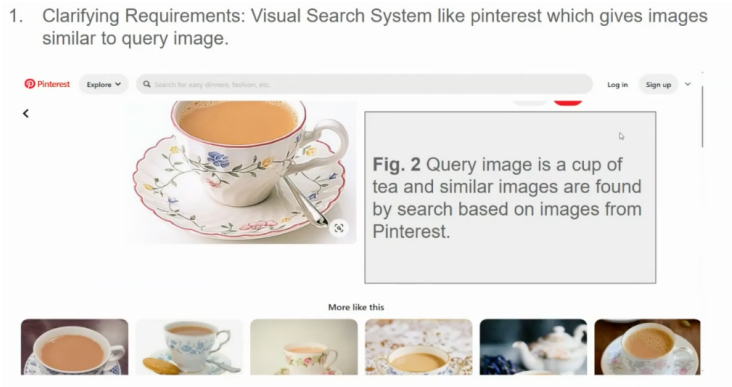
ML System Design Visual Search System

Video Link:
https://www.youtube.com/watch?v=5mkegPovTI8&list=PL_b_MRp1BnEj4UuHv1jAGeO1a0VTRXsAk&index=5

1. Objective

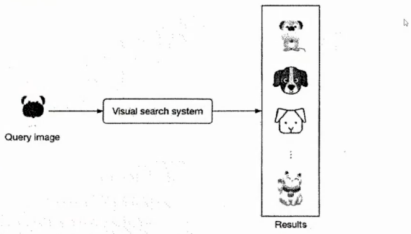






2. Framing problem as an ML task

2. Framing problem as an ML task : Accurately retrieve images that are visually similar to the image that user provided.
   a. Specifying i/o of system: Input will be user's query image and output will be ranked list of images by similarities.

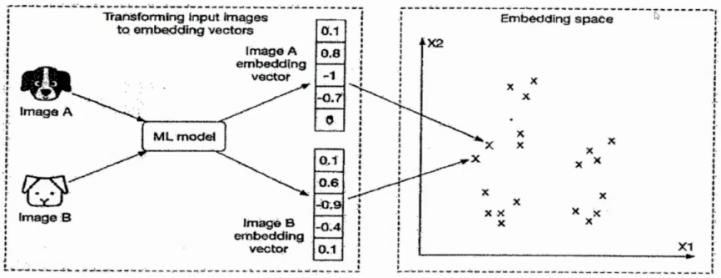


3. Choosing the right ML category:Our Visual search system falls under Ranking Problem.
   In Ranking problem, goal is to rank collection of items based on relevance of query so that more relevant items appear higher in the list.
   Search Engines, recommendation systems, document retrieval and online advertising can be termed as ranking problem.

We will be discussing Representation Learning.

Representation learning is a machine learning approach focused on automatically discovering and organizing features or representations from raw data. Instead of manually designing features, the model learns how to represent data in ways that make it easier for algorithms to perform tasks like classification, clustering, or prediction. Deep learning, such as neural networks, is commonly used for this as it can learn hierarchical and complex features directly from the data. This approach is widely used in image processing, NLP, and recommendation systems.

Representation Learning - The process of representing something on computer is called embedding. Embeddings are numeric, semantic representations of the contents of an image. The model maps input images to points in N-dimensional embedding space.



Similar images in the embedding space

An embedding is a way of converting items (like words, products, or even people) into numbers, making it easier for computers to understand and work with them. It's like creating a unique "location" for each item in a large "map" of numbers, where similar items are placed closer together. This helps computers recognize relationships, such as how

"apple" is similar to "banana" because both are fruits, even though they’re represented by different numbers. Embeddings are essential in tasks like recommendation systems, search engines, and language translation because they help algorithms find and understand patterns.

Embeddings are numerical representations of items like words, images, or products, mapped into a multi-dimensional "embedding space." In this space, each embedding (a vector of numbers) represents an item, and the distance between these embeddings reflects how similar or different the items are.

For example, suppose we have an image embedding model. When we process images of, say, dogs and cats, each image is transformed into an embedding in this space. If two images show similar things—like two different photos of a golden retriever—their embeddings will be close to each other in the space. Images of cats, on the other hand, will form a separate cluster, closer to other cats.

Finding Similar Images Using Embedding Space

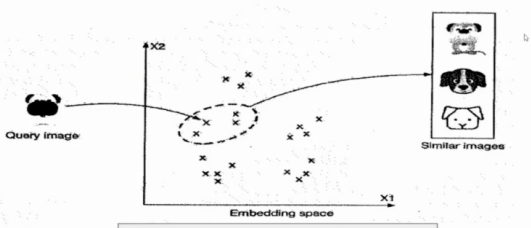
To find images similar to a given one, we can calculate the "distance" between the embedding of the query image and the embeddings of other images in the dataset. A common way to do this is by measuring cosine similarity or Euclidean distance.

For example:

1. **Query Image**: You input a photo of a golden retriever.
2. **Embedding Search**: The model transforms this image into an embedding vector.
3. **Compare Distances**: The model finds other embeddings in the dataset that are closest to this vector.
4. **Results**: Images of other dogs, especially golden retrievers, appear as the most similar matches.

This method is used in applications like Google Images, where you can search by image, and it finds visually similar images.

- Similarity score is calculated of query images and other images on the platform by measuring their distance in the embedding space.



Top 3 images similar to query image

3. Data

Data preparation:

Image -The metadata about image.

Creators upload images and system stores the images and their metadata such as owner id, upload time, tags, etc. Table shows simplified image metadata

ID	Owner ID	Upload time	Manual tags
1	8	1658451341	Zebra
2	5	1658451841	Pasta, Food, Kitchen
3	19	1658821820	Children, Family, Party

Users - The metadata about users.

User data contains demographic attributes associated with users, such as age, gender location, email etc. Table shows the Users data.

ID	Username	Age	Gender	City	Country	Email
1	johnduo	26	M	San Jose	USA	john@gmail.com
2	hs2008	49	M	Paris	France	hsieh@gmail.com
3	alexish	16	F	Rio	Brazil	alexh@yahoo.com

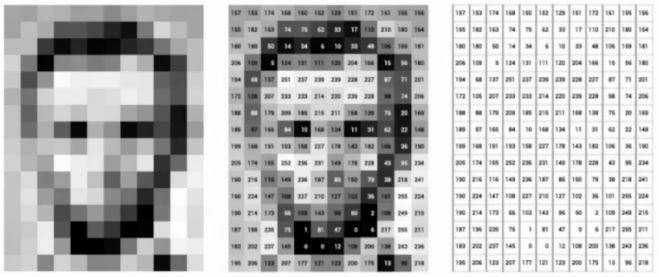
User-Image interaction.

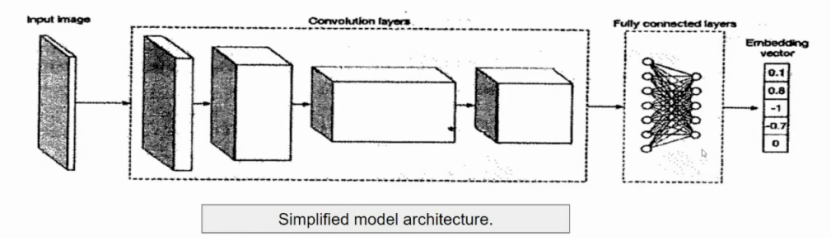
Interaction data contains different type of user interactions.Based on requirements gathered , the primary type of interactions are impressions and clicks. Table shows the overview of interaction data.

User ID	Query image ID	Displayed image ID	Position in the displayed list	Interaction type	Location (lat, long)	Timestamp
8	2	6	1	Click	38.8951 -77.0364	1658450539
6	3	9	2	Click	38.8951 -77.0364	1658451341
91	5	1	2	Impression	41.9241 -89.0389	1658451365

Feature Engineering:

- Resizing - resizing the image in a fix size.
- Scaling - scale the pixel values of image in 0 to 1.
- Grayscaling - converting the image to shades of gray.
- z score normalization - scale values of pixels to have mean of 0 and variance of 1.
- Consistent color mode - having consistent colors in image.





Simplified model architecture.

- Embedding is a dimensionality reduction technique. It is a lower dimensional vector representation of high dimensional feature vectors (i.e., raw input data) like words or images.
- Generally, image embedding algorithms extract distinct features in an image and represent them with dense vectors (i.e., unique numerical identifiers) in a different dimensional space.
- A CNN is a deep neural network model architecture containing two sets of blocks: convolutional and classification (fully connected layers) blocks.

Diagram Explanation

1. **Input Image**: The process starts with an input image, which is a high-dimensional array of pixel values.
2. **Convolutional Layers**:
    - The CNN applies multiple convolutional layers to the image. These layers detect various features like edges, textures, shapes, and complex patterns.
    - Convolutional layers progressively transform the raw pixel data into more abstract feature representations. Each layer builds on the features learned by the previous one.
3. **Fully Connected Layers**:
    - After the convolutional layers, the features extracted from the image are flattened and fed into fully connected (dense) layers.
    - These layers further combine and process the features to capture more complex relationships and representations, ultimately reducing the data to a lower-dimensional vector.
4. **Embedding Vector**:
    - The final output from the fully connected layers is an **embedding vector**, a dense, lower-dimensional representation of the original image.
    - This vector is a series of numbers (as shown, e.g., [0.1, 0.8, -1, 0.7, 0]) that uniquely represents the image in a more compact form, which is useful for various downstream tasks like image retrieval or similarity comparison.

Bullet Point Explanation

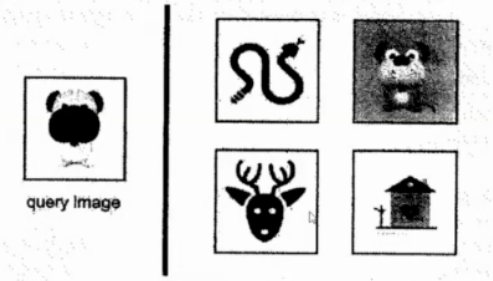
- **Embedding as Dimensionality Reduction**:
    - Embeddings are a **dimensionality reduction technique**. They transform high-dimensional data (such as raw images) into a lower-dimensional vector representation.
    - This is beneficial because it allows for the storage and processing of complex data in a compressed form without losing essential information.
- **Feature Extraction for Dense Vectors**:
    - The process of creating embeddings involves **extracting distinct features** from the input data, whether it's images or text.
    - For images, these features could include patterns, colors, shapes, or textures that uniquely define the content of the image.
    - The extracted features are then represented as a **dense vector** (embedding) in a multi-dimensional space, where similar images (in terms of content or visual features) are closer together.
- **CNN Architecture**:
    - A **CNN (Convolutional Neural Network)** is used here as the model architecture.
    - CNNs typically consist of two main types of layers:
        - **Convolutional layers** for feature extraction (these detect patterns within the data).
        - **Fully connected layers** (classification layers) to interpret and summarize these features into a lower-dimensional output.

Applications of Embedding Vectors

- **Similarity Search**: Embedding vectors allow similar images to be easily found. Since images with similar features have embeddings that are close in the embedding space, we can calculate distances (e.g., using cosine similarity) to find images that are visually similar to a query image.
- **Image Classification and Clustering**: The embedding vector can also be used to classify or cluster images. For example, images of cats and dogs will have distinct clusters in the embedding space, making it easy to identify which category a new image belongs to based on its embedding.

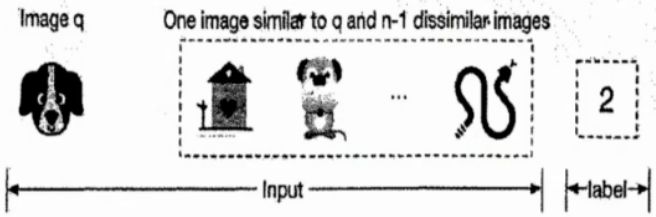
Model Training :

- Contrastive training - in this technique, the model learns general features without labels by teaching the model to learn about similar and dissimilar images.



Contrastive Training

- Constructing Dataset -



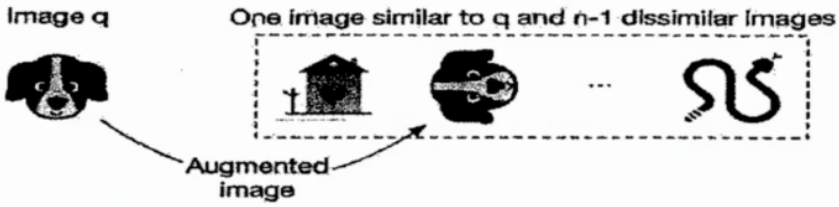
Training data point

To construct a training data point - we will randomly choose a query image, a positive image and (n-1) negative images from total n images.

To select query image we can:

- Use human judgment
- User clicks as proxy for similarities.
- Artificially augment the image

After constructing we can also switch to other techniques or also combine.



Data augmentation to create similar image.

Contrastive Loss

Contrastive loss is a loss function used in machine learning, particularly in tasks involving similarity learning, such as image and text matching. It aims to minimize the distance between similar pairs while maximizing the distance between dissimilar pairs.

1. **Loss Function**: The contrastive loss for a pair of inputs is defined as:

$$L(y, \hat{y}) = (1 - y) \cdot \frac{1}{2} (\hat{y})^2 + y \cdot \frac{1}{2} (\max(0, m - \hat{y}))^2$$

where:

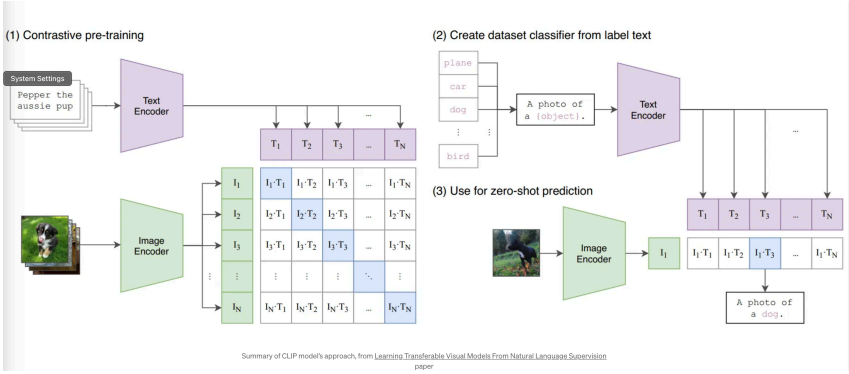
- y is a binary label (1 for similar pairs, 0 for dissimilar).
- m is a margin that specifies how far apart dissimilar pairs should be.

2. **Training Data Construction**:
   - **Similar Pairs**: These can be created by augmenting the same data point (e.g., image transformations) or by selecting similar instances (e.g., images of the same class).
   - **Dissimilar Pairs**: These are created by pairing data points from different classes or categories.

CLIP (Contrastive Language-Image Pre-training)

CLIP is a model developed by OpenAI that learns visual concepts from natural language descriptions. It uses a contrastive learning approach to align images and text.

- **Training**: CLIP is trained on a large dataset of images paired with textual descriptions. During training, it learns to maximize the similarity between the image and its corresponding text while minimizing it for non-corresponding pairs.
- **Architecture**: It typically employs two encoders: one for images and one for text. Both encoders map their inputs into a shared embedding space, where contrastive loss is applied.

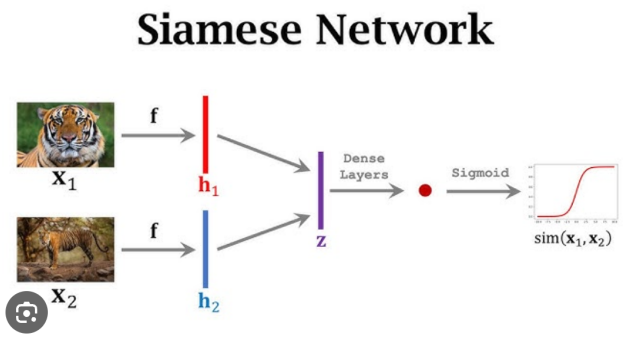


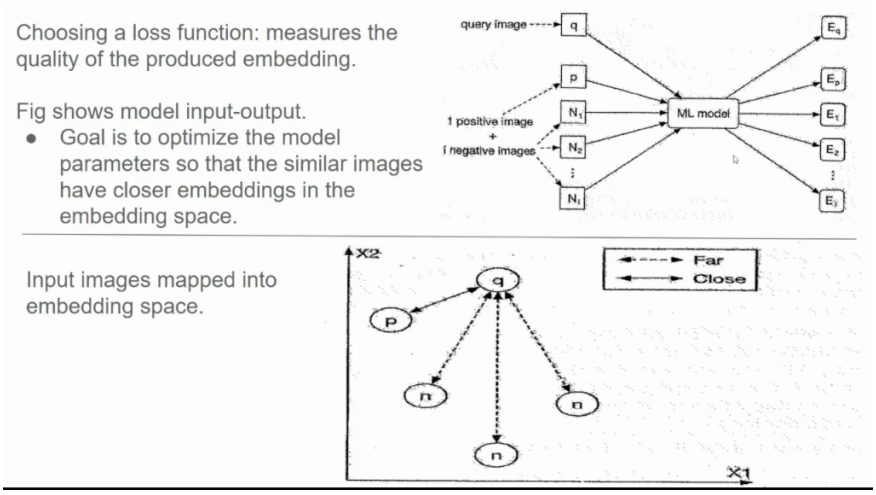
Summary of CLIP model's approach, from Learning Transferable Visual Models From Natural Language Supervision paper

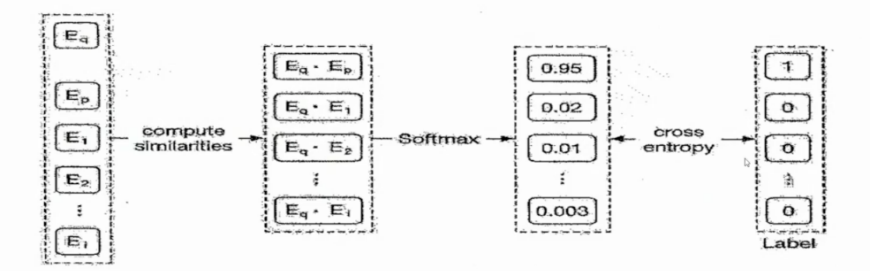
Siamese Network

A **Siamese network** is a neural network architecture that contains two or more identical subnetworks (sharing the same parameters) that process different inputs.

- **Functionality**: The network computes a similarity score between the outputs of the subnetworks. It is often used with contrastive loss, where it learns to distinguish between similar and dissimilar pairs.
- **Applications**: Common applications include face verification, image retrieval, and any task where determining similarity is crucial.



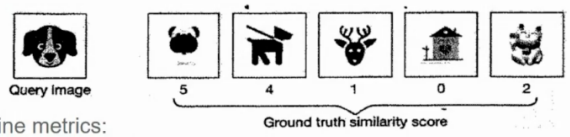




- Compute similarities - computing the similarities with the query image and the embeddings of other images.
- Softmax - A softmax function is applied over the computed distances and makes sures the value sums upto 1 which allows the values to be treated as probabilities.
- Cross Entropy - checks the predicted probabilities with ground truth labels.

We can also use pretrained models to reduce the training time.

Evaluation:
Offline Evaluation -Query image, candidate image and similarity of each candidate image and query image pairs. Precision and Ranking quality of ranked list will be measured.



Some offline metrics:

- Mean Reciprocal Rank (MRR) -
- Recall@k - **Recall@k = (# of recommended items @k that are relevant) / (total # of relevant items)**
- Precision@k
- Mean Average Precision (mAP) - relevant and irrelevant
- Normalized Discounted Cumulative Gain (nDCG)

For online metric we will use Click-through rate (CTR).

1. Mean Reciprocal Rank (MRR)

- **Definition**: MRR is a statistical measure used to evaluate the effectiveness of an information retrieval system. It is particularly useful in scenarios where a system returns a ranked list of results.
- **Calculation**: It is calculated as the average of the reciprocal ranks of the first relevant result for a set of queries.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$$

where |Q||Q||Q| is the total number of queries, and ranki\text{rank}_iranki is the rank of the first relevant result for query iii.

2. Recall@k

- **Definition**: Recall@k measures the ability of a model to retrieve relevant items in the top kkk results.
- **Calculation**: It is calculated as the ratio of the number of relevant items retrieved in the top kkk to the total number of relevant items available.

$$\text{Recall@k} = \frac{\text{Number of relevant items in top } k}{\text{Total number of relevant items}}$$

3. Precision@k

- **Definition**: Precision@k assesses the accuracy of the top kkk retrieved items, indicating how many of them are relevant.
- **Calculation**: It is computed as the ratio of relevant items in the top kkk to the total items retrieved in that set.

$$\text{Precision@k} = \frac{\text{Number of relevant items in top } k}{k}$$

4. Mean Average Precision (mAP)

- **Definition**: mAP is an average of precision values at different levels of recall across multiple queries. It is commonly used in information retrieval and object detection tasks.
- **Calculation**: It is computed as the mean of average precision values for each query.

$$\text{mAP} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \text{AP}_i$$

where APi\text{AP}_iAPi is the average precision for query iii, calculated as the average of precision scores at each point a relevant item is retrieved.

5. Normalized Discounted Cumulative Gain (NDCG)

- **Definition**: NDCG is a measure of ranking quality, emphasizing the importance of highly ranked relevant items. It is particularly useful when the relevance of items is graded rather than binary.
- **Calculation**: It combines the concept of cumulative gain (relevance scores) with a discounting factor based on the rank position.

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

NDCG is then calculated as:

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

where IDCG is the ideal DCG, the DCG obtained from the perfect ranking of items.

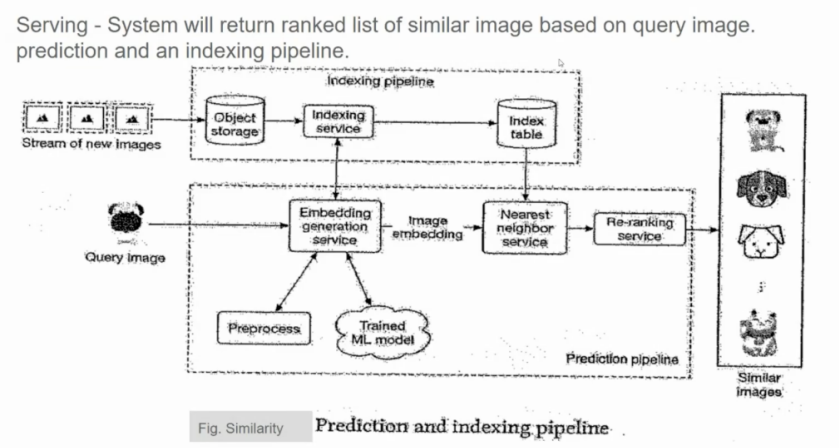
6. Click-Through Rate (CTR)

- **Definition**: CTR is a metric used to measure the effectiveness of online advertising campaigns, indicating the percentage of users who click on an ad after seeing it.
- **Calculation**: It is calculated as the ratio of the number of clicks to the number of impressions.

$$\text{CTR} = \frac{\text{Number of clicks}}{\text{Number of impressions}} \times 100$$

These metrics provide different insights into the performance of information retrieval systems, recommendation systems, and online advertising campaigns, allowing for a more comprehensive evaluation of effectiveness.

5. Serving



Prediction Pipeline:

1.Embedding Generation Service

This service computes the embedding E of input query image through pre-processing and Trained ML model.

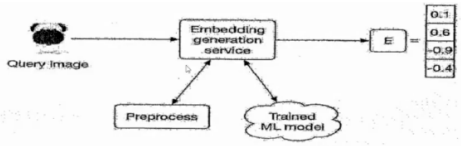


Figure 2.20: Embedding generation service

2. Nearest Neighbour Service.

Once we have the query image(q's) embedding E, the Nearest Neighbour Service is a Search to computes the most relevant images from the image Index (S) that have E' close to E.

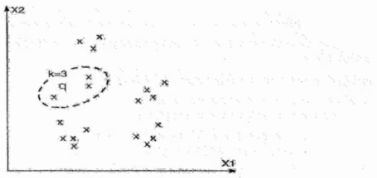


Figure 2.21: Top 3 nearest neighbors to image q in the embedding space

3. Reranking Service - Incorporates business-level logic and policies.

This service incorporates business level logic and policies e.g. to filter duplicates, remove private images, before displaying the ranked results to user.

Indexing Pipeline:

- Indexing services - indexes all the images, keeps the index table updated but also increases the memory usage.

Nearest Neighbour Search algorithms - they are core components of search, retrieval and recommendation system.

1. Exact Nearest Neighbour
2. Approximate Nearest Neighbour

1. Exact NN - searches the entire index table and calculates the distance between each points with query q and returning the k nearest points.

Time complexity of Exact NN is O(NXD), where N is total number of points and D is point dimension which is too low for large models. (Good for limited Data set)

2. Approximate NN (ANN) - ANN is used when showing similar results to user is sufficient. (Efficient for large Data set)

Let's discuss few ANN types.

ANN types (cont):

- Tree based ANN.
- Locality sensitive hashing (LSH) based ANN.
- Clustering based ANN.

1. Tree based ANN
   - Tree is formed by algorithm by splitting the space into multiple partition and leveraging the characteristics to perform a faster search.
   - Adding new criteria to each node eg. gender = male, then all the nodes on left subtree will be female.

   Non- leaf nodes split the space in two partition given the criterion.

   Leaf node will indicate a specific region in the space.

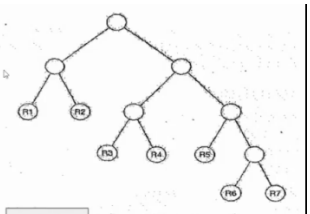


fig1 A formed tree from the points

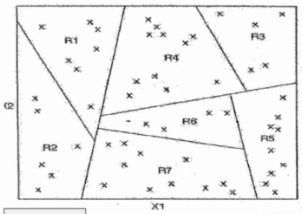


fig2 Partitioned space by the tree

2. Locality sensitive hashing (LSH) based ANN (cont): Uses particular hash functions to reduce the dimension of points and are stored in buckets.This improves search time.

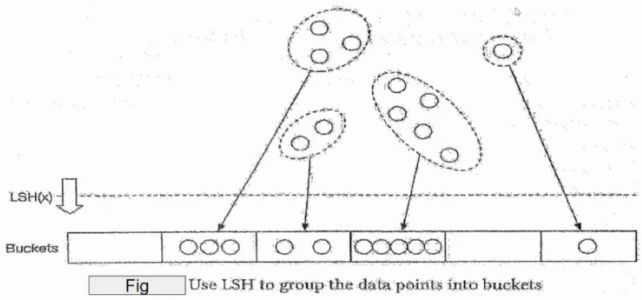


Fig Use LSH to group the data points into buckets

3. Clustering based ANN - Clusters are formed based on similarities and when algorithm is searching only the cluster where the query belongs to is searched.

How LSH Works:

1. **Hash Functions:** LSH uses specially designed hash functions that map similar points to the same bucket with high probability. These functions are often based on random projections or p-stable distributions.
2. **Hashing and Bucketing:** Data points are hashed using multiple hash functions, and each hash function maps the point to a specific bucket.
3. **Querying:** When a query point arrives, it is hashed using the same hash functions.
4. **Candidate Retrieval:** The algorithm only needs to search the buckets that the query point hashes to. This significantly reduces the search space compared to a brute-force search.
5. **Similarity Calculation:** Within the candidate buckets, the query point is compared to the data points using a similarity metric (e.g., Euclidean distance, cosine similarity) to find the nearest neighbors.

Feature	Tree-based ANN	LSH-based ANN	Clustering-based ANN
How It Works	Utilizes spatial partitioning data structures (e.g., KD-trees, Ball trees) to organize data points hierarchically, allowing efficient querying of nearest neighbors.	Uses hash functions that map similar data points to the same buckets with high probability, enabling efficient search among reduced candidate sets.	Groups data points into clusters and searches for nearest neighbors within clusters, often using techniques like k-means.
When to Use	When the data is low-dimensional and has a relatively uniform distribution, allowing for efficient partitioning and query times.	When dealing with high-dimensional data, as it approximates nearest neighbors effectively without the curse of dimensionality.	When data can be effectively clustered, allowing for improved search efficiency within clusters, and when approximate solutions are acceptable.
When Not to Use	Not suitable for high-dimensional data (curse of dimensionality), as performance degrades significantly.	May not work well for data types that are not amenable to hashing or when fine-grained accuracy is required.	Less effective if clusters are not well-separated or if the dataset is too small, as clustering can introduce noise.
Scale of Data	Works well for medium-scale datasets (thousands to millions of points) but struggles with very large datasets.	Scales well to very large datasets (millions to billions of points) due to its hashing mechanism.	Suitable for medium to large datasets but may not perform well with very high dimensions or extremely large datasets.
Speed of Inference	Fast inference time due to efficient traversal of tree structures, typically logarithmic complexity in low dimensions.	Fast inference, generally O(1) lookup time for hashed buckets but may require scanning multiple buckets, depending on hash collisions.	Inference speed depends on the clustering algorithm used; generally faster than exhaustive search but slower than tree-based methods in low dimensions.