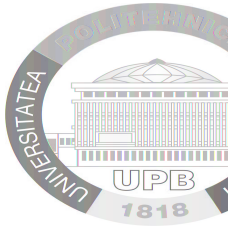


From Pixels to Meaning:

Organizing visual data with semantic embeddings

Advisor
Prof. dr. ing. Bogdan DUMITRESCU

Vlad-Ștefan TUDOR

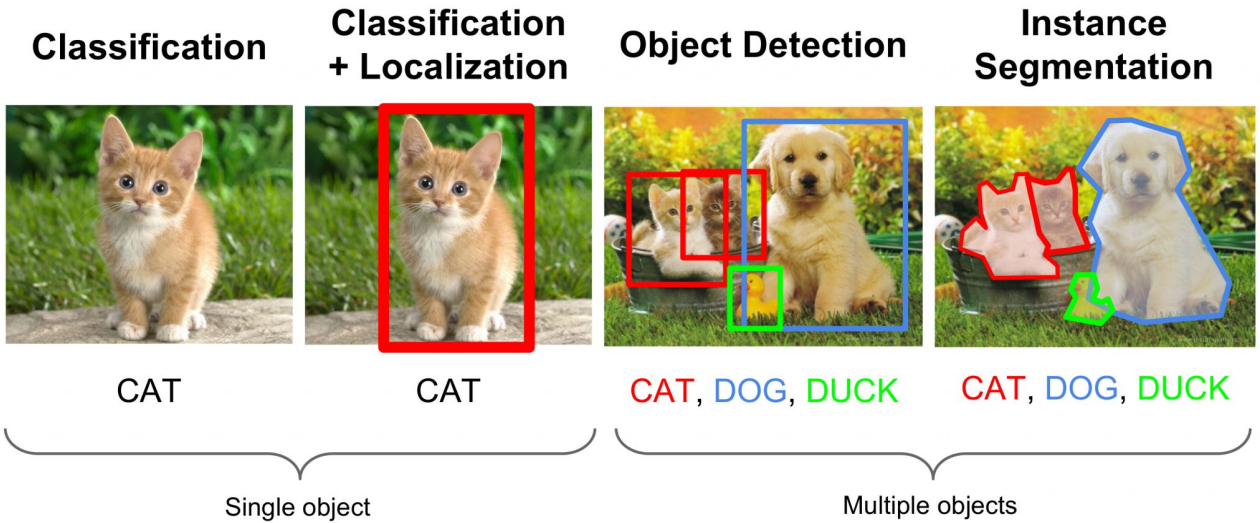
Problem Definition:

Large Volumes of unstructured data

How can visual data be interpreted?

1. The classical approach:

   Image classification & Object Detection

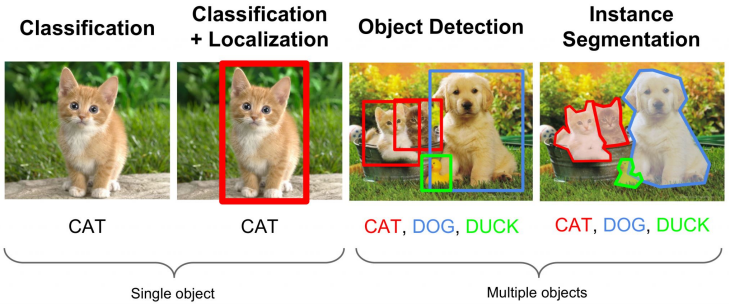


(source)

How can visual data be interpreted?

The classical approach:

Image classification & Object Detection



Pros: robust, predictable, models are easy to train

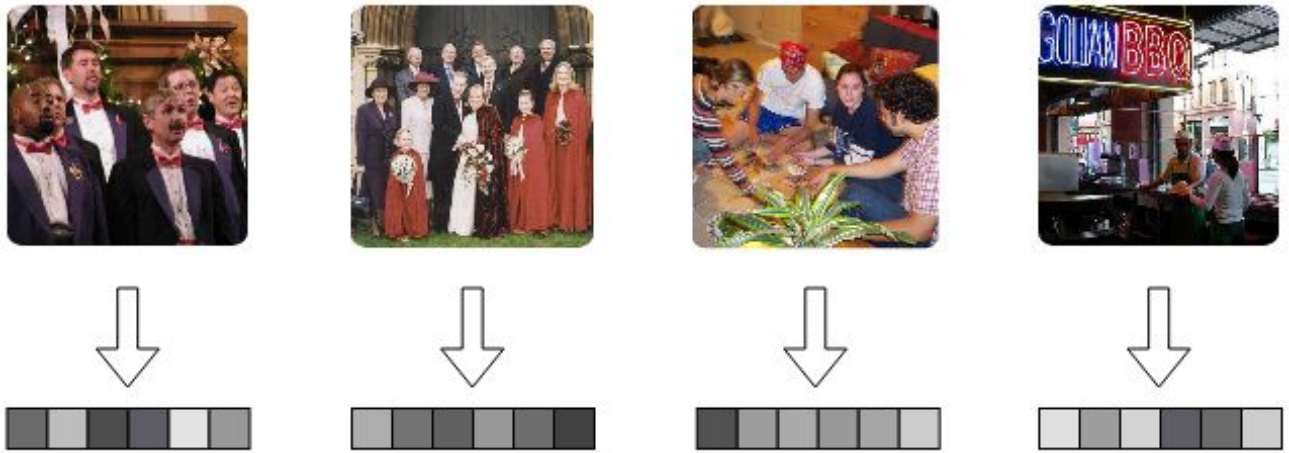
Cons: needs labeled data, hard to define a generic solution

(source)

How can visual data be interpreted?

A different approach:

Image representation

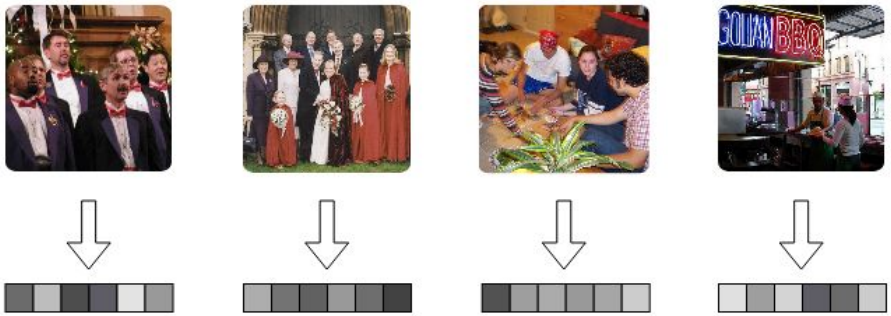


Numerical arrays (embeddings) semantically represent the content

How can visual data be interpreted?

A different approach:

Image representations (embeddings)



Pros: organic representation, allow for emergent properties

Cons: need large volumes of data and more specialized training methods.

Embeddings: defining similarity

Are these images similar?

What *consistent* score for similarity can we assign for the images?

(source)

Embeddings: defining similarity

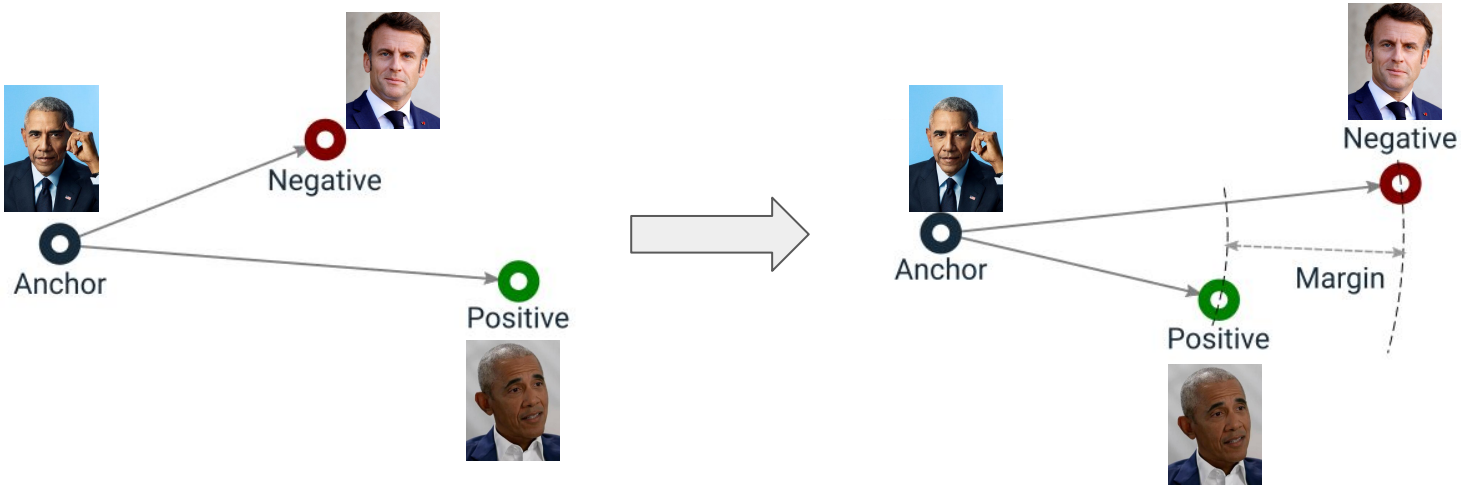
Similarity between images: hard to quantify
Similarity between embeddings: supports any mathematical operation

Idea: Transform all images into a numerical representation, then apply a distance metric:

- Cosine
- Euclidean
- Manhattan
- …

How to generate robust image embeddings?

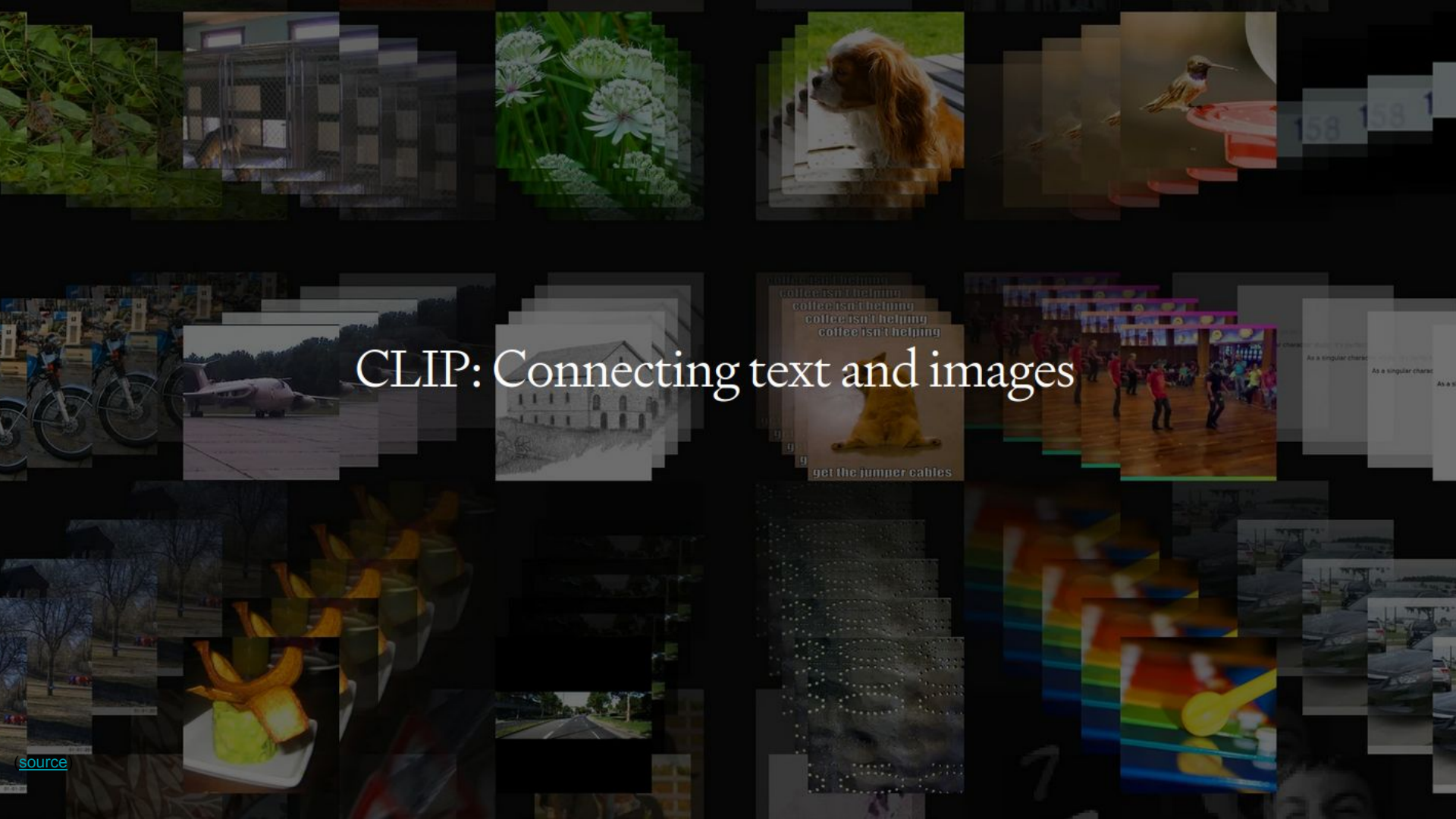
Compare embeddings themselves: optimize similarity / contrast



Example from Triplet Loss

Objective: minimize distance between similar-kind input embeddings

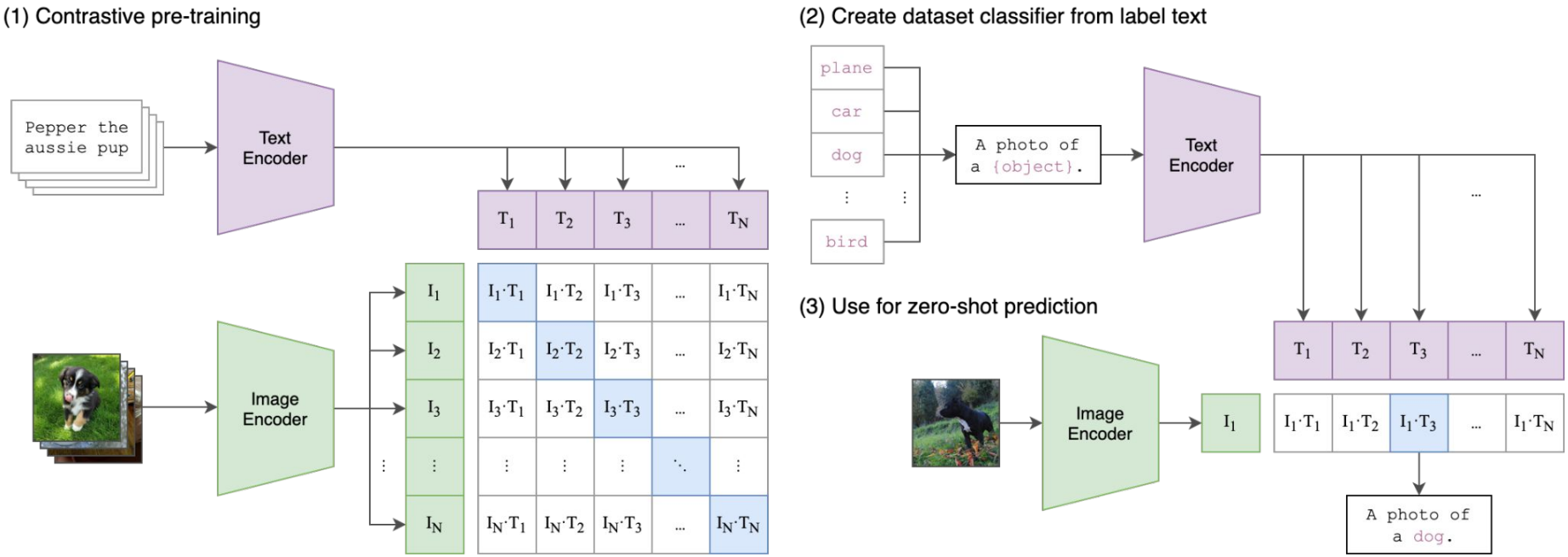
(source)



CLIP: Connecting text and images

source

CLIP model: similarity between images and captions

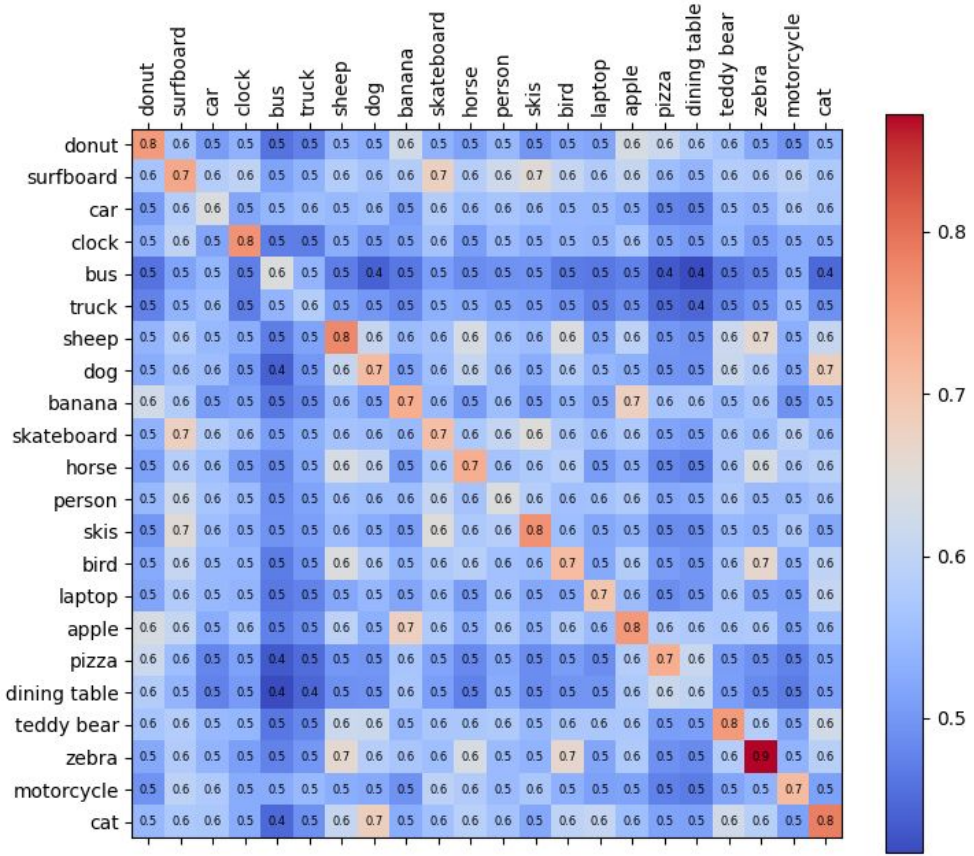


Text encoder + image encoder project inputs in a shared embedding space

(source)

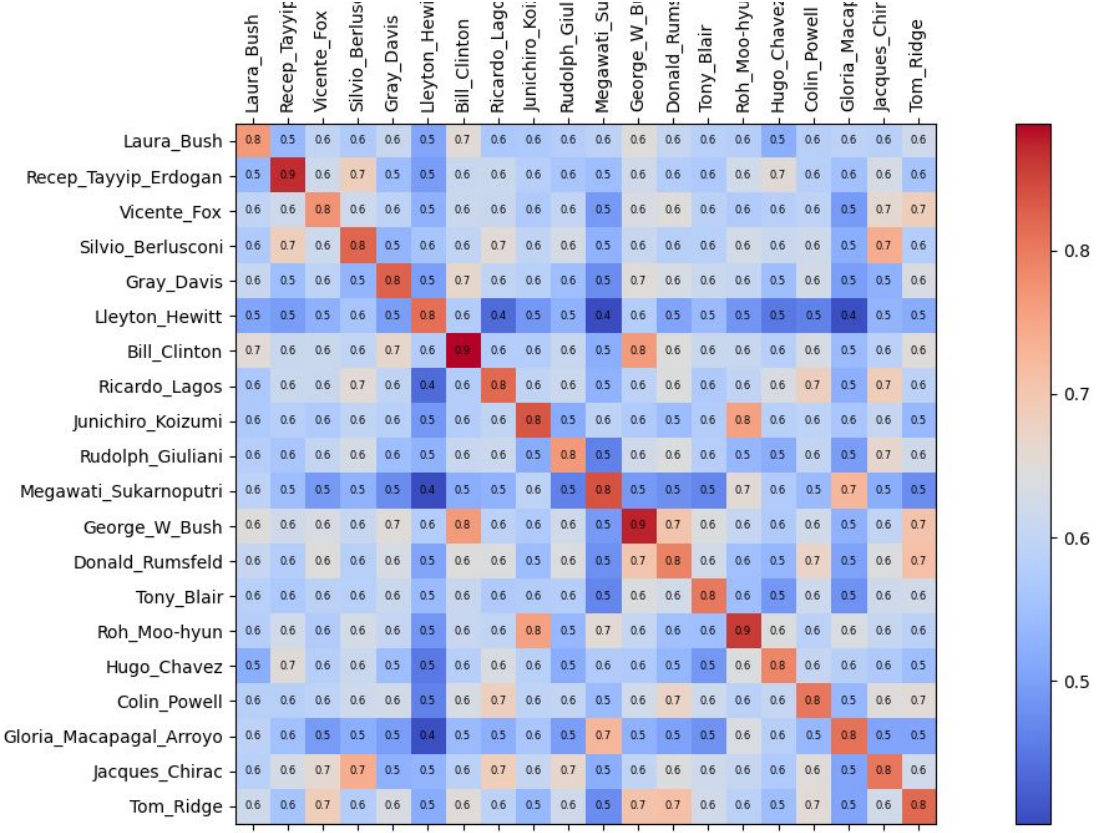
CLIP embeddings: image-to-image similarity

Inner and inter-class similarity for samples from the COCO dataset.



CLIP embeddings: image-to-image similarity

Inner and inter-class similarity for samples from the LFW dataset.



CLIP embeddings: text-to-image similarity

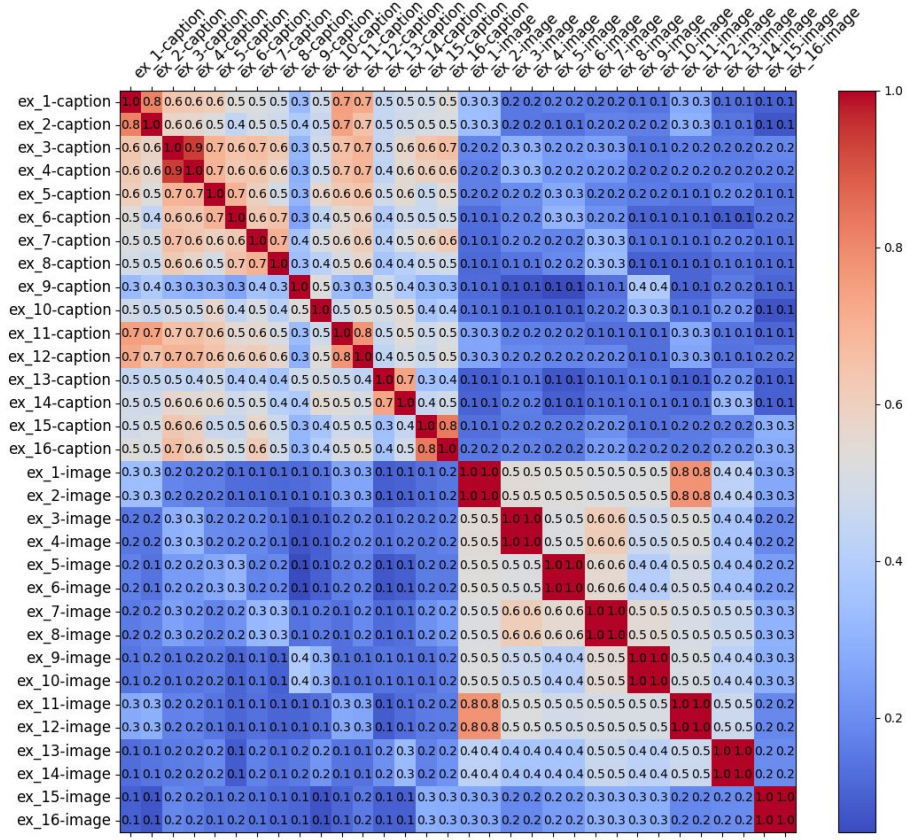
Similarity between images and captions from COCO dataset

- Compare image to its corresponding captions
- Compare captions to captions
- Compare images to images

Example 1:
Caption 0: A set of three pizzas in a display case.next to desserts.
Caption 1: four different pizzas in a glass case and a few other food items

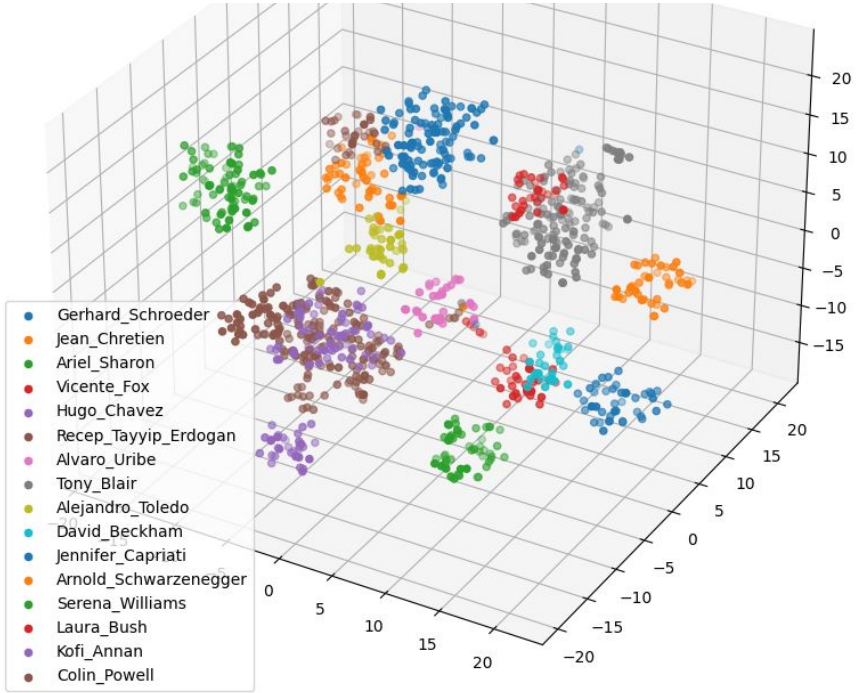
Example 2:
Caption 0: A remote control sitting on a table with the telivision in the background.
Caption 1: A remote control on a table for the television.

CLIP embeddings: text-to-image similarity



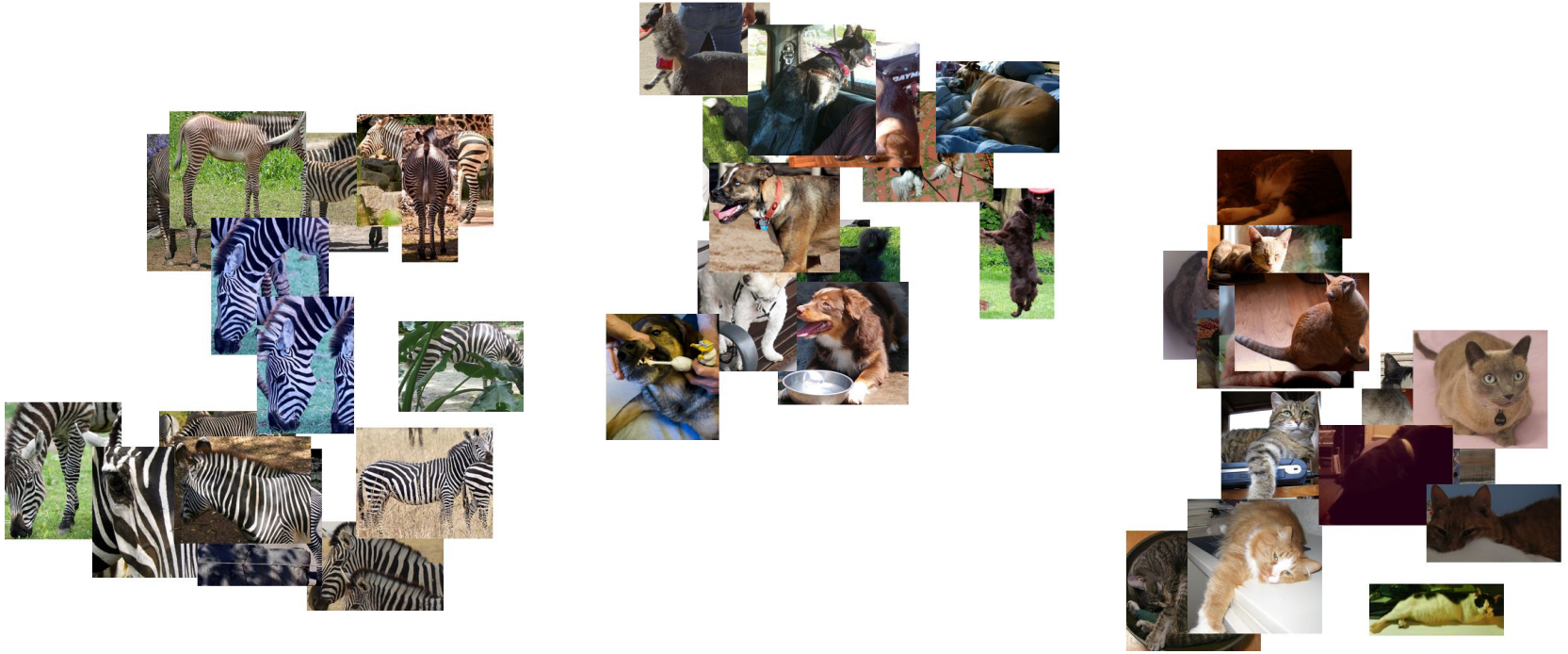
Leveraging embeddings

Exploring the embeddings space geometry (t-SNE algorithm)



Leveraging embeddings

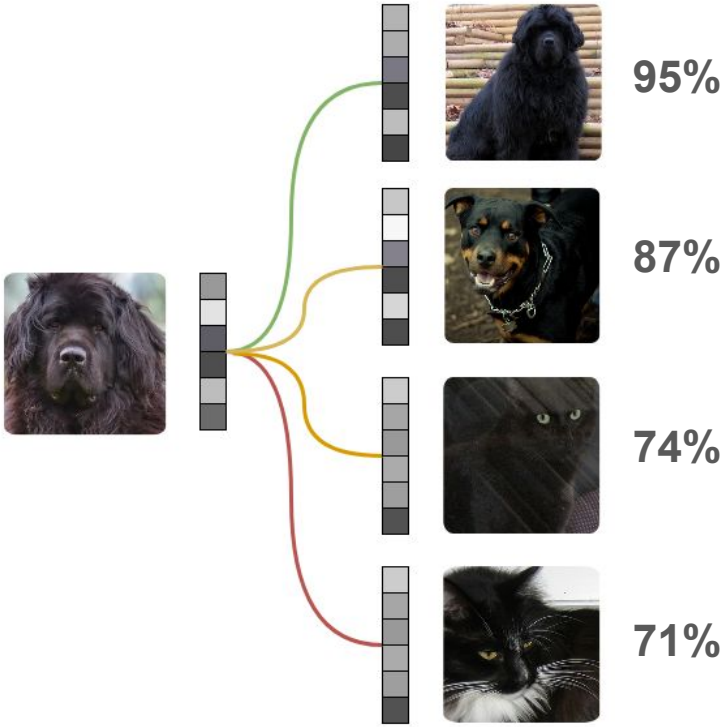
Identifying groups of similar samples with clustering (DBSCAN)



Leveraging embeddings

Search with specific queries

The indexed samples are ordered by their similarity score to the query

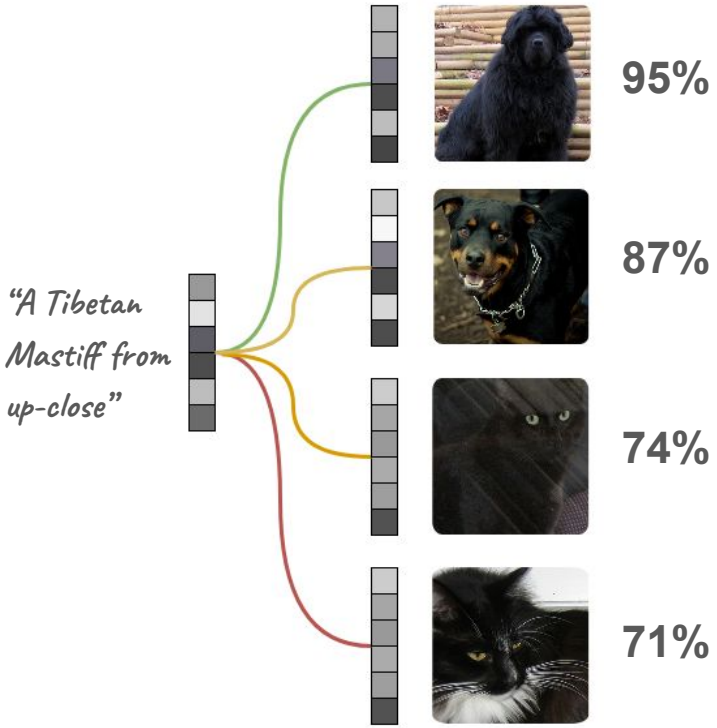


Leveraging embeddings

Search with specific queries

Since CLIP can process both images and text, captions can be used as queries directly

(actually, the original training criteria)



Customization:

Fine-tuning the base model for specific behaviour

The CLIP model understands *some* named entities.

What if we want the model to understand some specific persons, places or actions?

Solution: Train the model a little more…

Fine-tuning CLIP on custom data

Experiment Dataset:
LFW (Labeled Faces in the wild)

Problem 1: Fine-tuning can lead to model “forgetting” the initial data

Problem 2: The model already knows the entities in the dataset

(source)

Fine-tuning CLIP on custom data

Problem 1: Fine-tuning can lead to model “forgetting” the initial data

Solution: Merge the target dataset with a background dataset (COCO)

Problem 2: The model already knows the entities in the dataset

Solution: Swap the entities in the LFW dataset

“A photo of George W Bush”

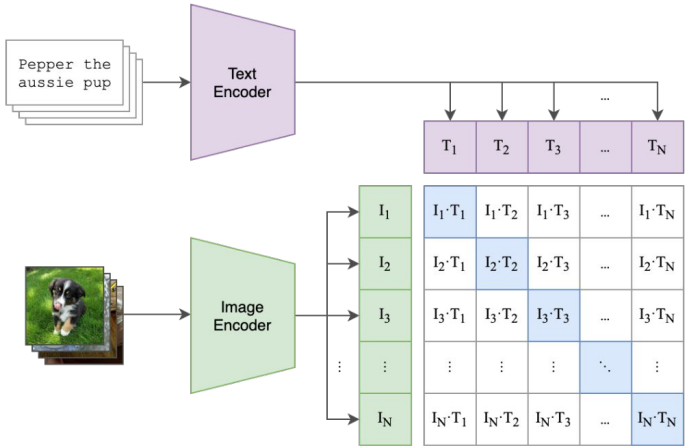
“A male surfer carrying a white board exiting the ocean.”

“Portrait of Jacques Chirac”

“Painting of a cloth, a teapot, sugar, and three oranges.”

Fine-tuning CLIP on custom data

Experiment Setup:
Loss function



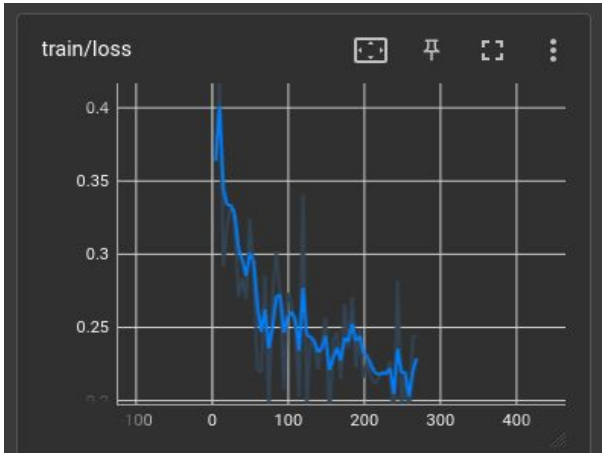

```
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
# scaled pairwise cosine similarities [n, n]
logits = 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
```

In essence, a classification task between images embeddings and captions embeddings in the batch

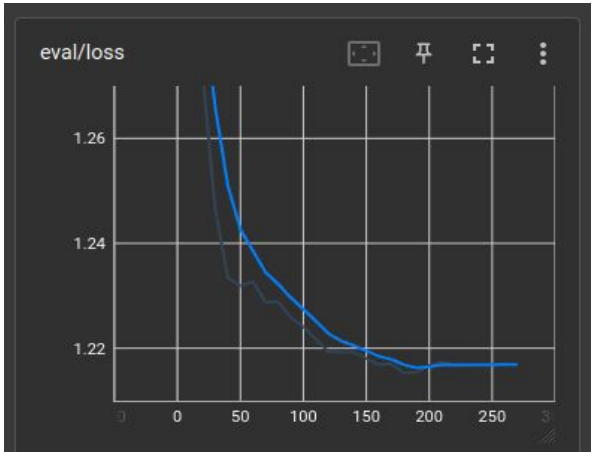
Fine-tuning CLIP on custom data

Experiment results

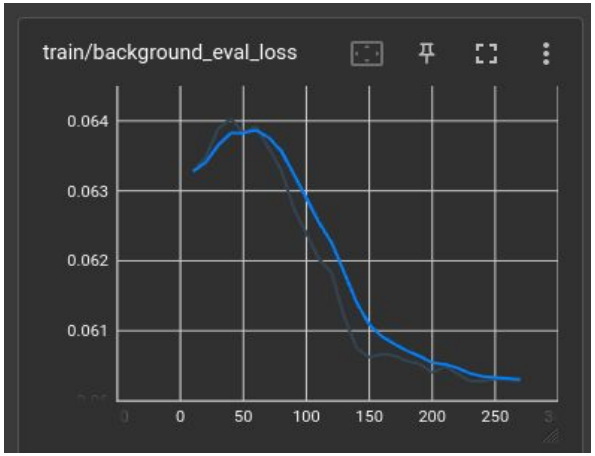
Recall that the loss penalizes cosine distance between the embeddings of an image and its corresponding caption.



Target + background datasets



Target dataset (evaluation)



Background dataset (evaluation)

(x-axis shows experiment progress in training steps)

Fine-tuning CLIP on custom data

Experiment results

Evaluating performance in terms of image-to-caption cosine similarity

Model	Target Dataset	Benchmark Dataset
Base model	0.194	0.312
Trained model	0.221	0.341

Model increased in performance on the target dataset, while retaining performance on the background dataset

Core concept: Organising images at scale

Who's photo collection also looks like this?

Goal:

An end-to-end system that can infer structure over a collection of unlabeled images

Designing the system: core components

Database: store image, embeddings and results information through runs

Image Processing: generating embeddings from images

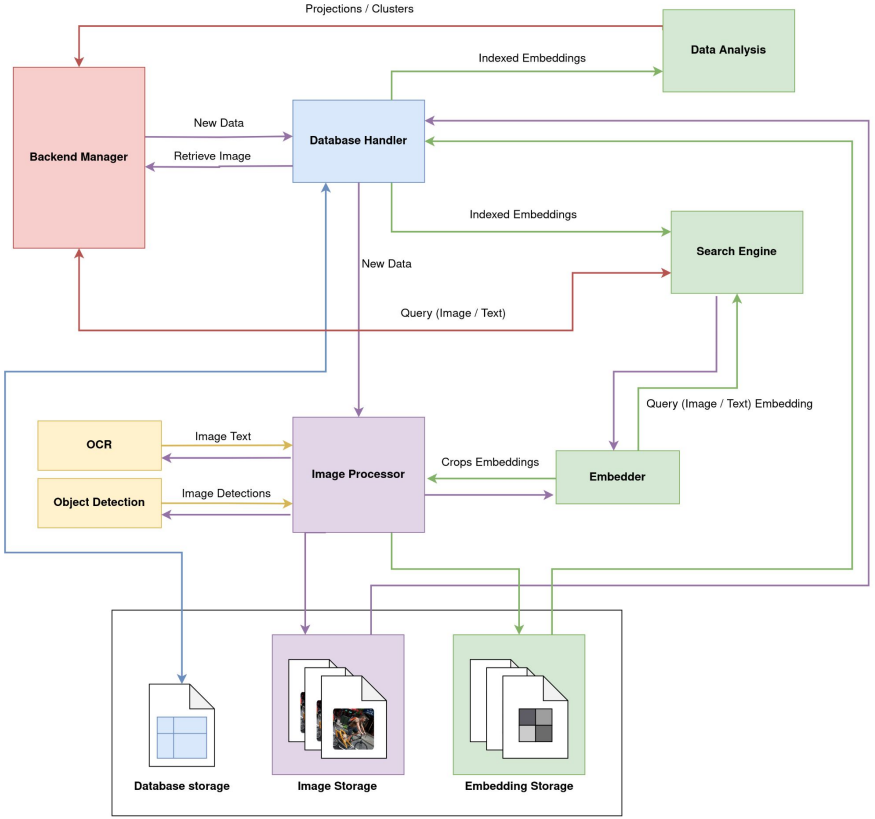
Search Engine: allow for text / image - based search

Data Analyzer: implement clustering and projections on the generated embeddings

API: expose endpoints for relevant methods of the system

Frontend: user interface for uploading images and visualizing results

Designing the system: Architecture

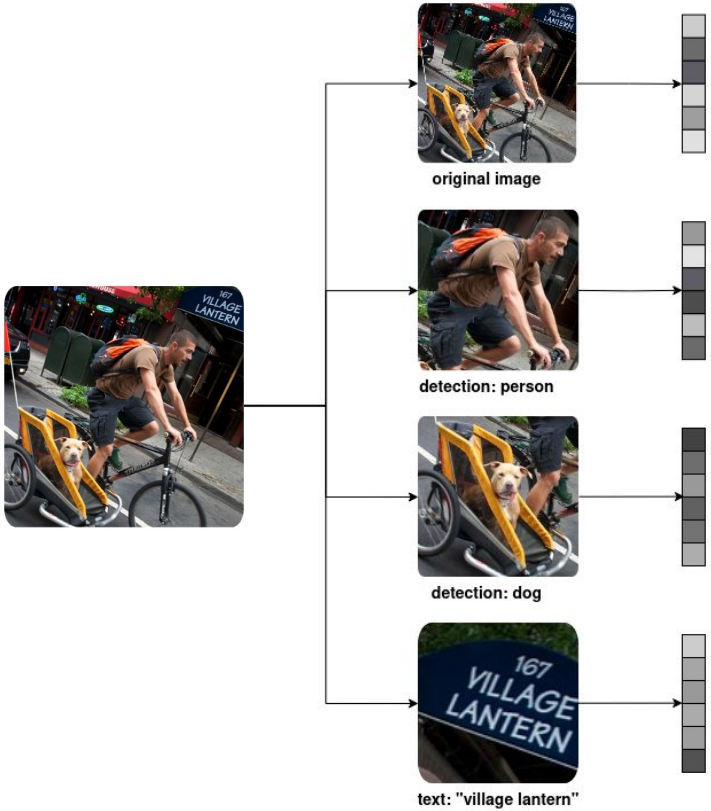


Designing the system: Image Processor

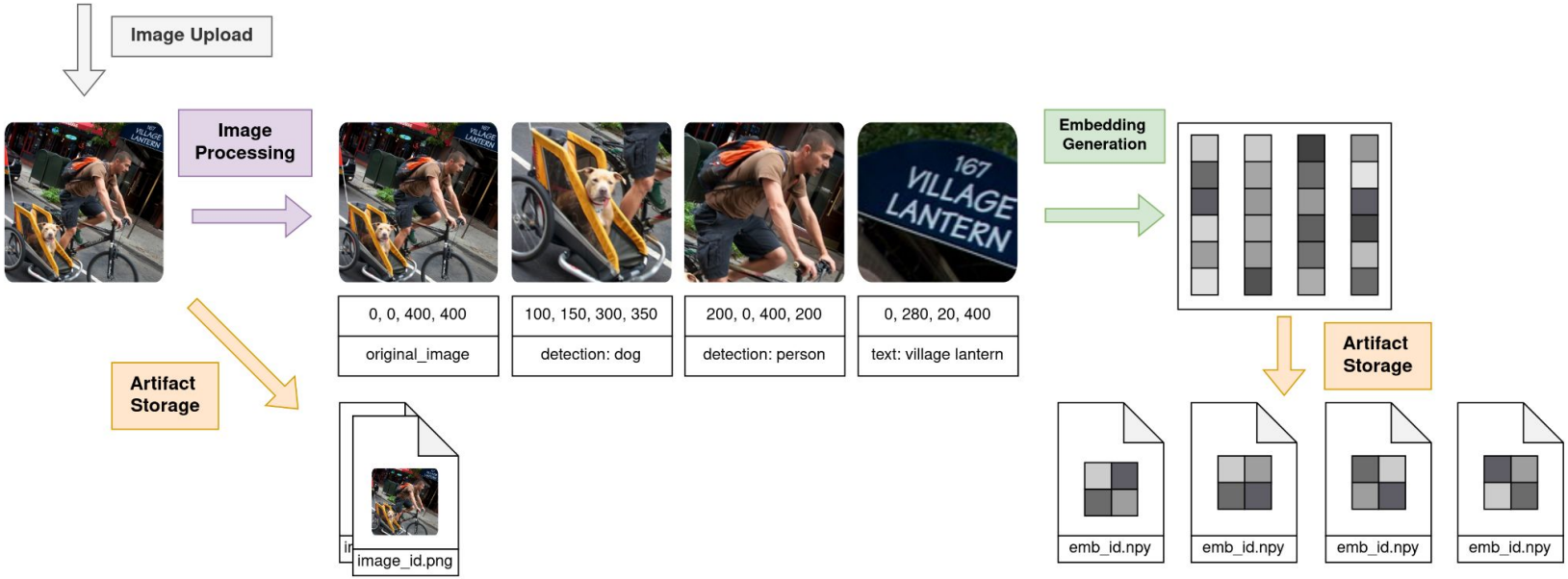
Idea: use an Object Detector and an OCR system to generate multiple embeddings per image

Pro: allows for focused analysis, person / object recognition

Con: might introduce false positives (especially for clustering)



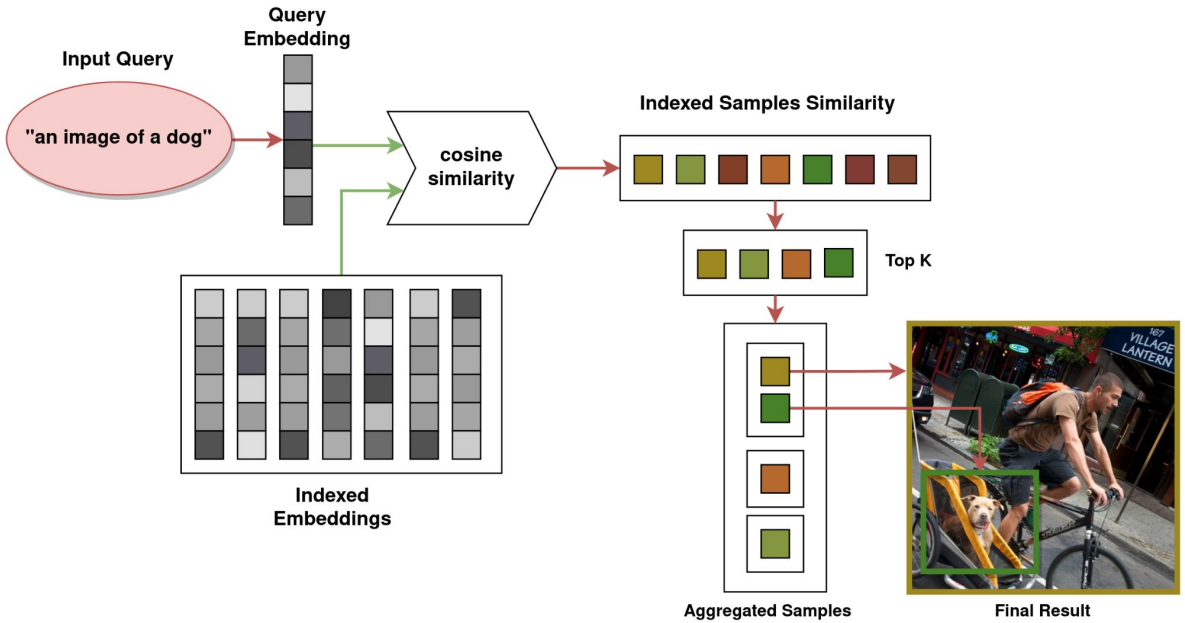
Designing the system: Image Processor



The module handles all image-related operations

Each resulted sample (original image or crop) is treated as an independent entity

Designing the system: Search Engine



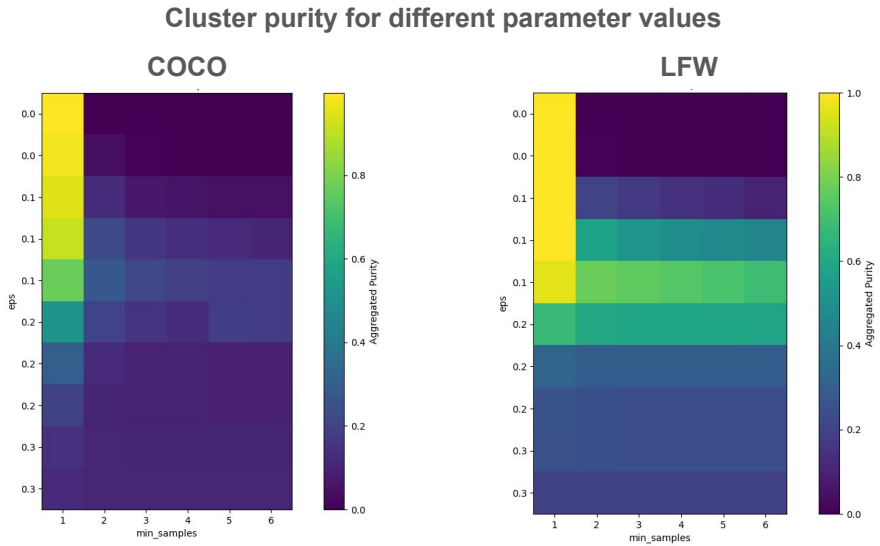
Embedding computations leverage GPU, if available.

Designing the system: Clustering

Implemented using DBSCAN algorithm (scikit-learn)

Choice of parameters:

- **Minimum points:** The number of samples in a neighborhood for a point to be considered as a core point.
- **radius (eps):** The maximum distance between two samples for one to be considered as in the neighborhood of the other.

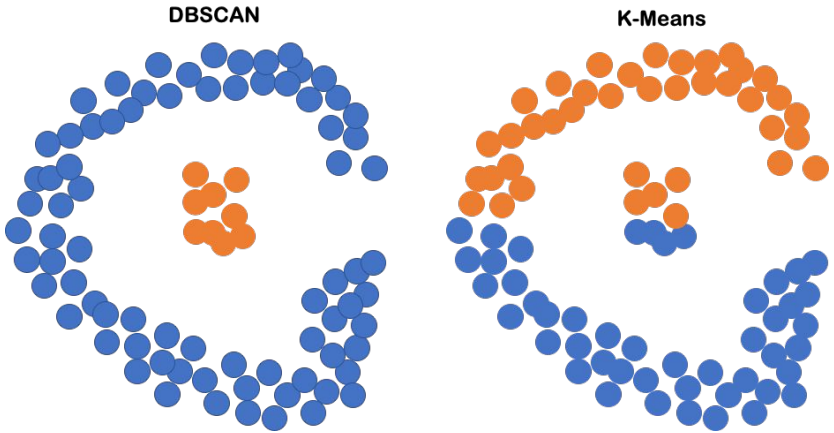


Note: Purity penalizes clusters that have multiple different ground-truth labels.
This means that for single-element clusters the purity is 100%)

Designing the system: Clustering

Advantages of DBSCAN:

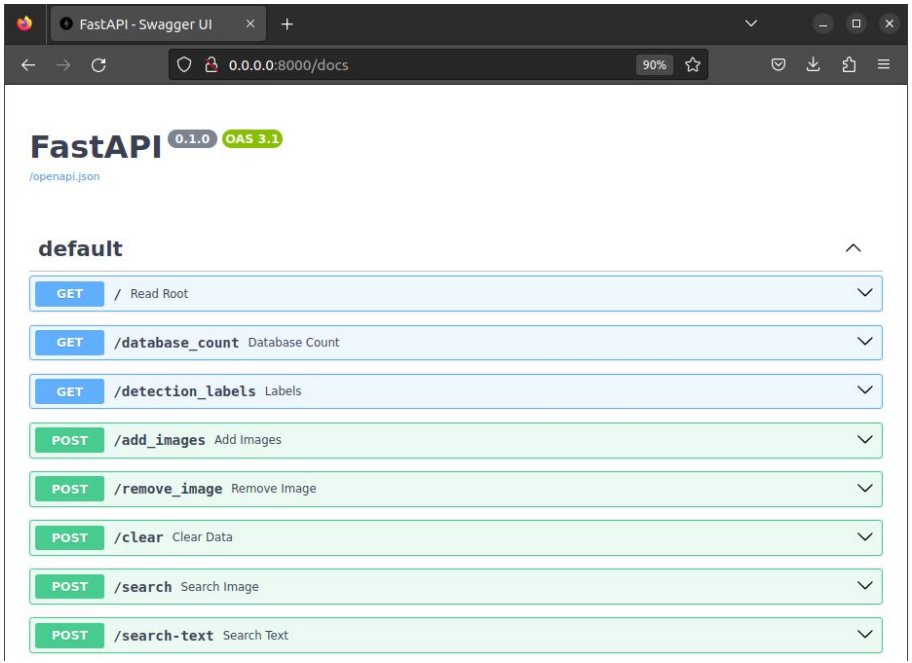
- doesn’t require the target number of clusters
- density-based (understands more complex shapes)
- robustness to outliers



(source)

Designing the System: API

The frontend communicates with the backend through an API



Demo

App: http://184.105.5.51:9000/

API: http://184.105.5.51:8000/docs

Questions & Answers