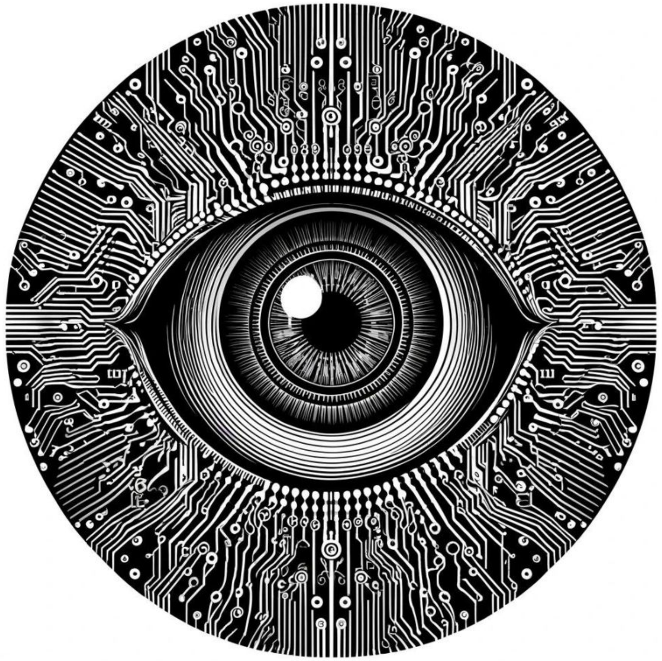
Contrastive Language-Image Pretraining (CLIP)



Antonio Rueda-Toicen

KI Service Zentrum by Hasso-Plattner-Institut HPI

SPONSORED BY THE

Federal Ministry of Education and Research

Learning goals

- Understand how CLIP models are trained
- Classify images with CLIP using a zero-shot approach
- Create image and text embeddings with a pretrained CLIP model
- Discuss CLIP’s uses and limitations

CLIP: ‘Contrastive Language Image Pretraining’

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [*1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1]
Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]

- Connects text and images in a shared embedding space
- Created by OpenAI in 2021
- Trained on **400M image-textual description pairs** scraped from the internet
- Predicts the most relevant text snippet given an image, enabling “zero-shot classification”

Aligning text and image embeddings

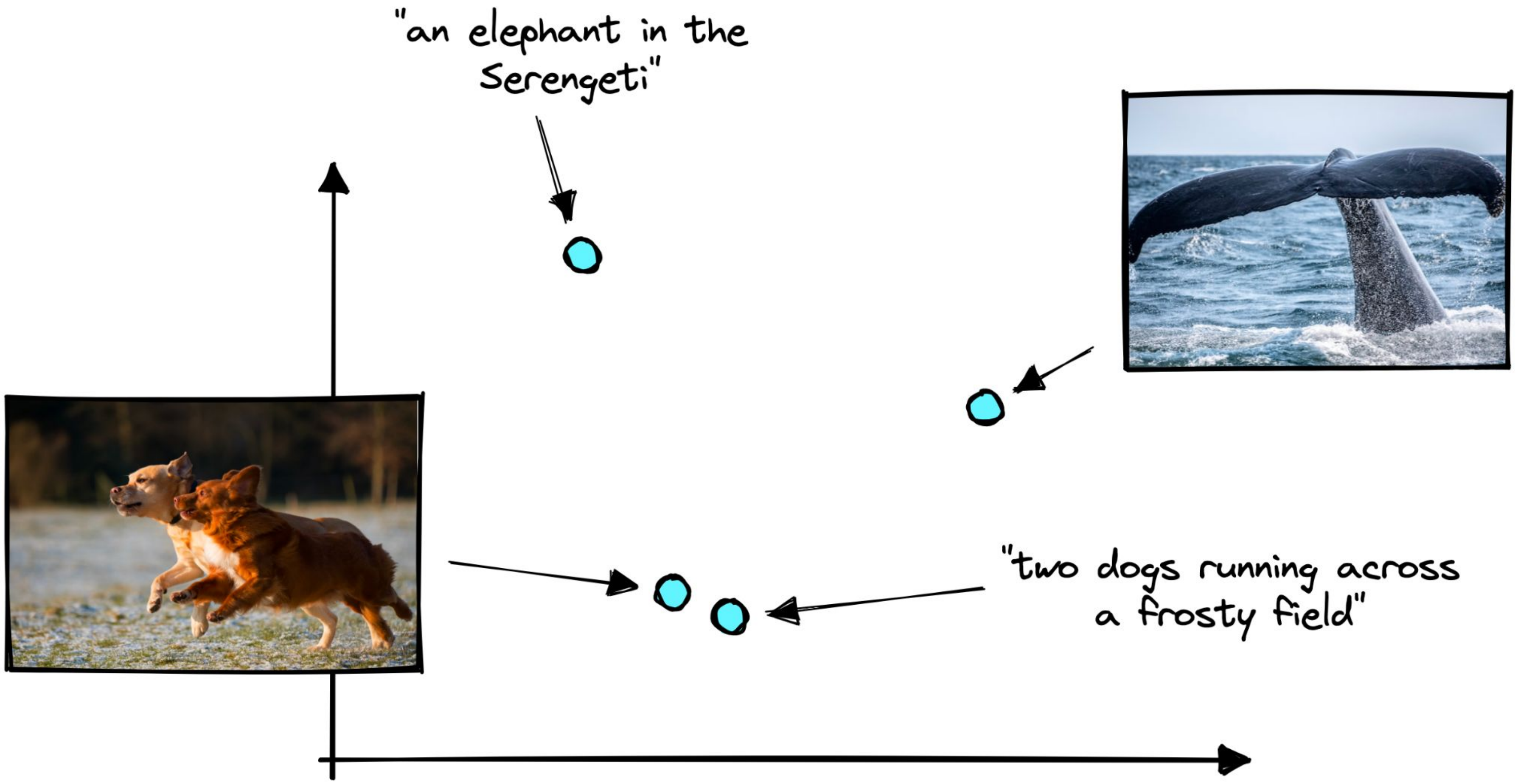
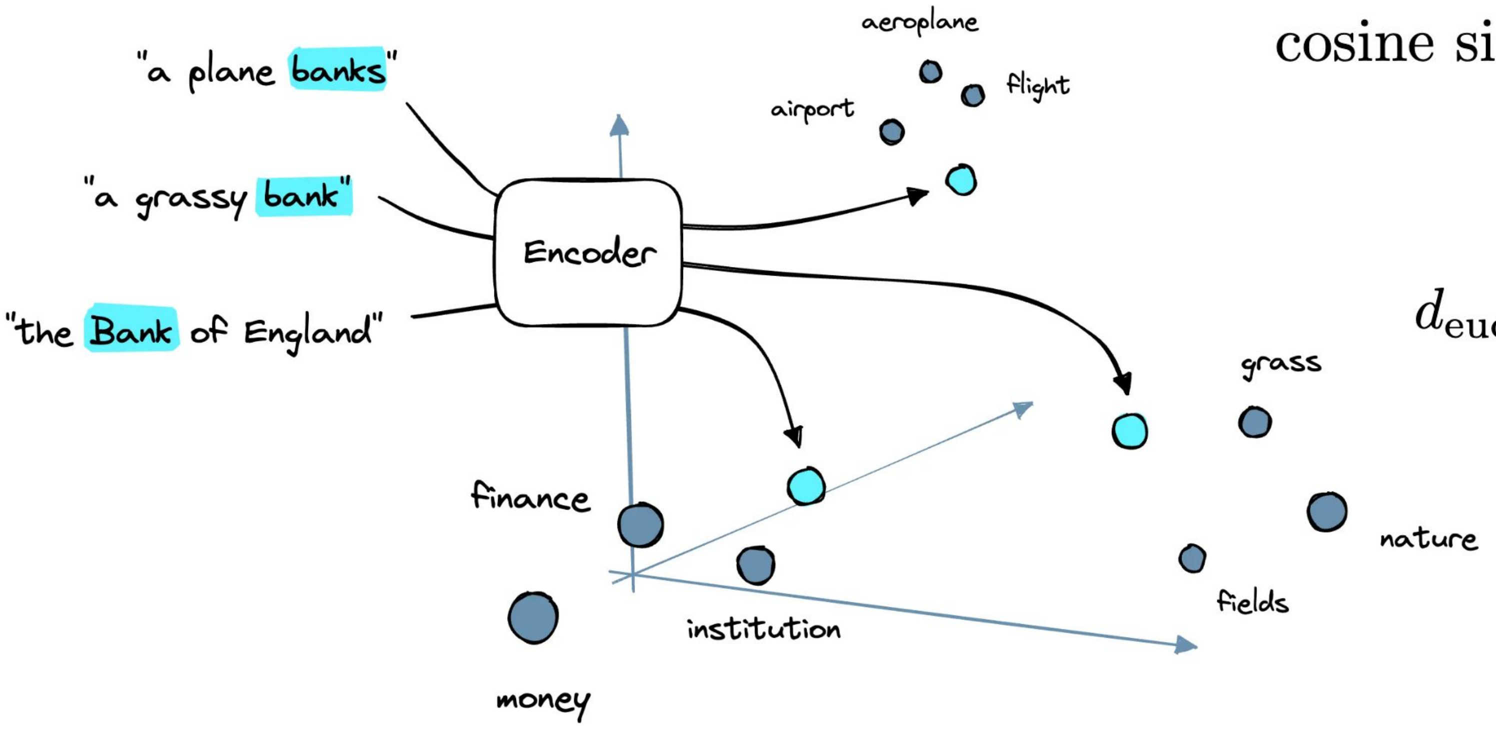


Image from https://www.pinecone.io/learn/series/image-search/clip/

Text encoders

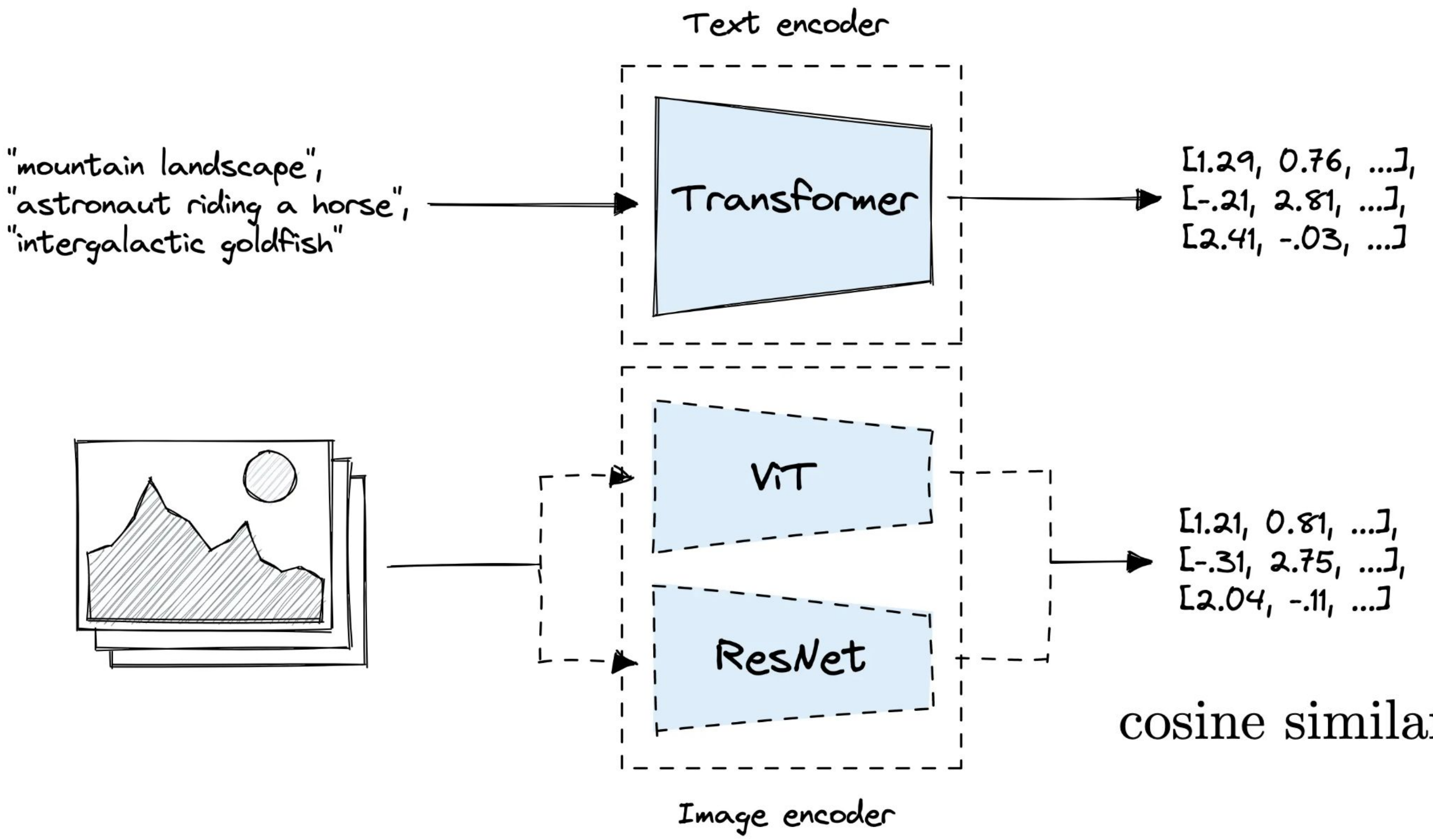


$$\text{cosine similarity}(x_1, x_2) = \frac{x_1 \cdot x_2}{\|x_1\| \, \|x_2\|}$$

$$d_{\text{euclid}}(x_1, x_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

Image from https://www.pinecone.io/learn/series/image-search/vision-transformers/

CLIP’s architecture



Both encoders map inputs to 512 dimensional embeddings

$$\text{cosine similarity}(x_1, x_2) = \frac{x_1 \cdot x_2}{\|x_1\| \, \|x_2\|}$$

Image from https://www.pinecone.io/learn/series/image-search/clip /

Maximizing cosine similarity of matching text and image embeddings

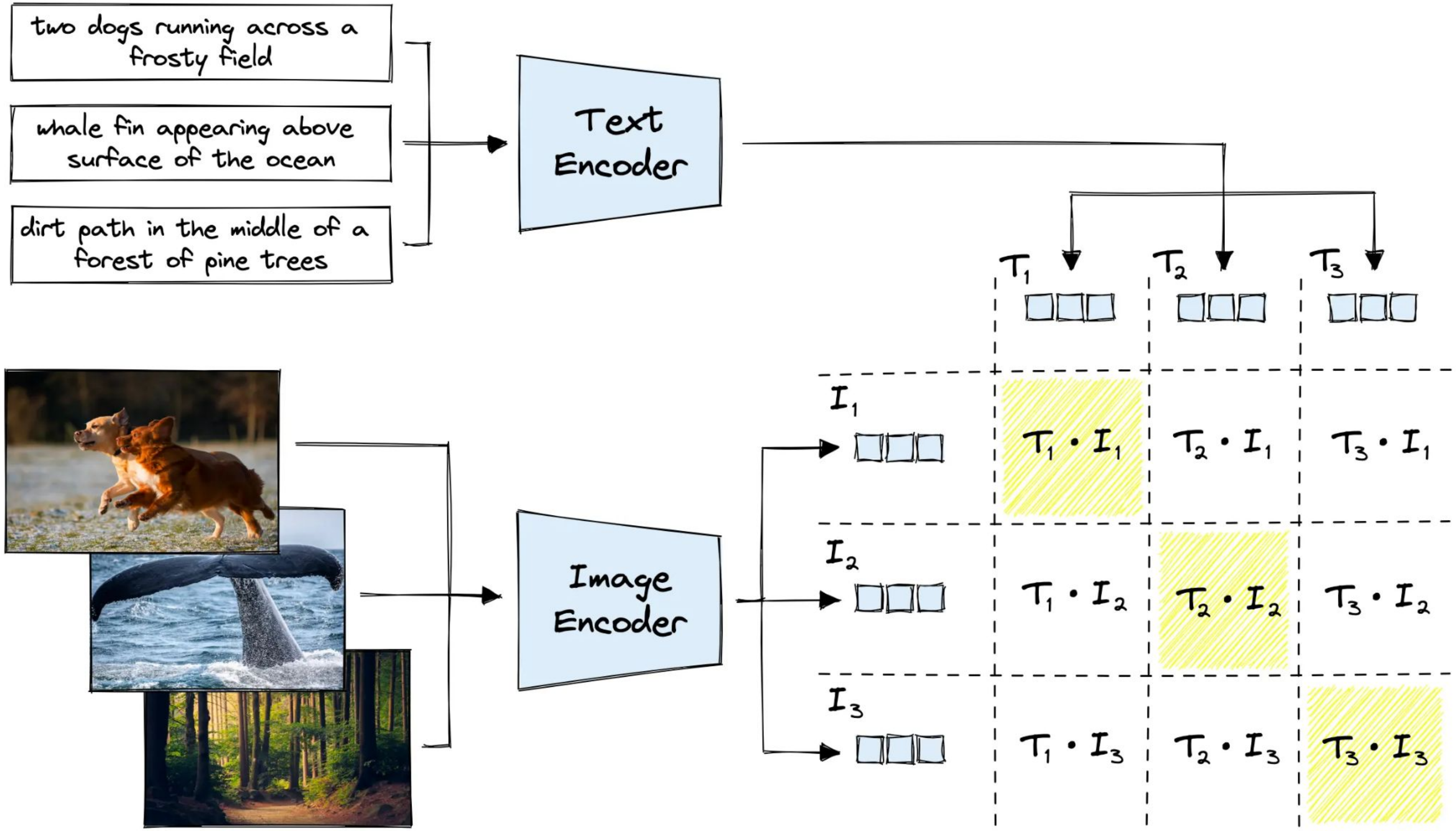
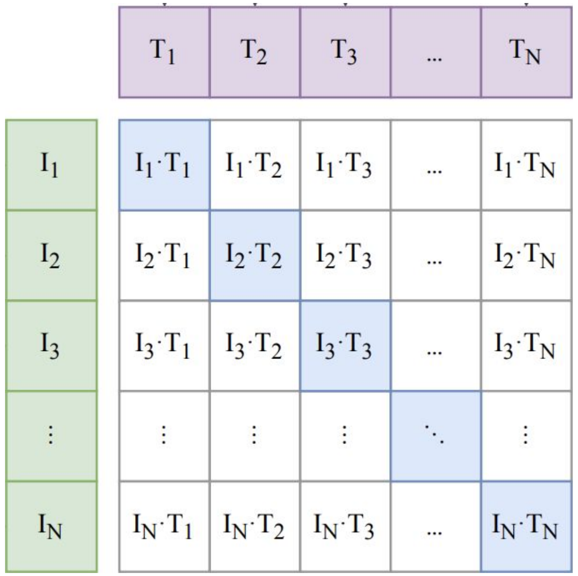


Image from https://www.pinecone.io/learn/series/image-search/clip/

Training algorithm

```
# 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
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

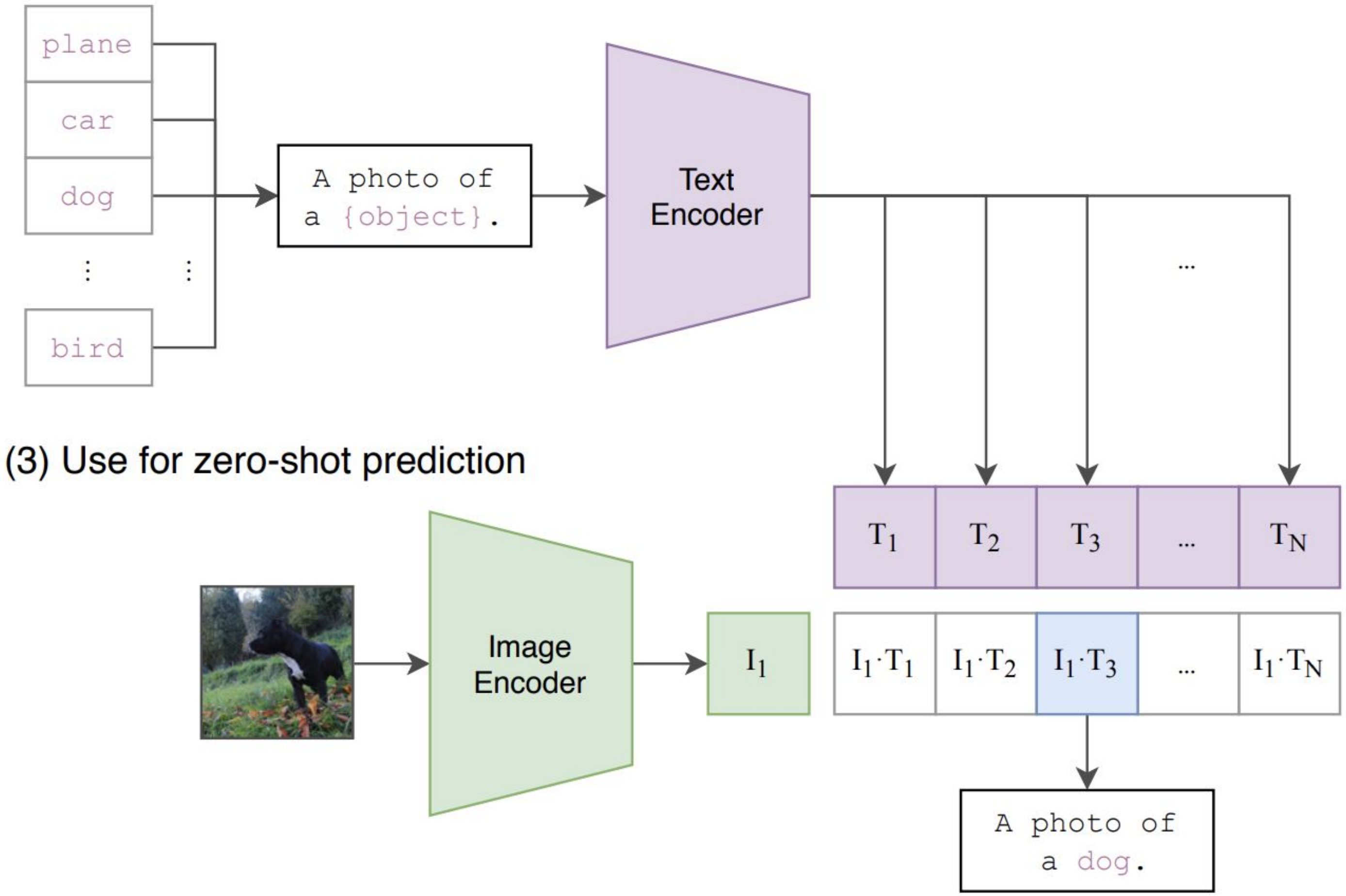


$$H(p,q) = -\sum_i p(i)\log q(i)$$

$$H(q,p) = -\sum_i q(i)\log p(i)$$

$$\text{symmetric CE loss} = \frac{H(p,q)+H(q,p)}{2}$$

Zero-shot classification with CLIP



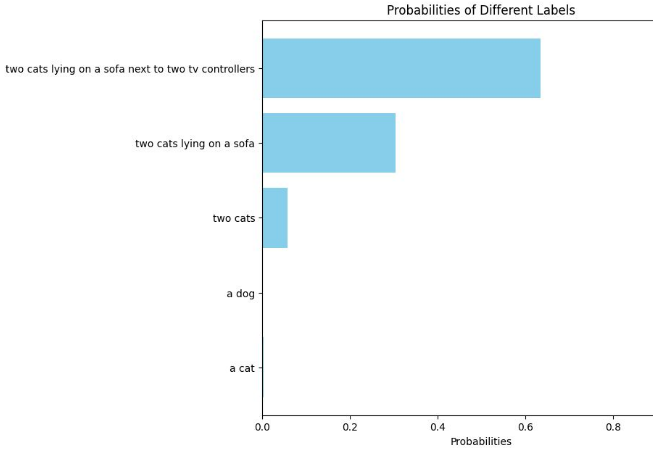
The model is able to create good classifiers without extra training

$$\mathrm{softmax}(x_i, T) = \frac{e^{x_i/T}}{\sum_{j=1}^{n} e^{x_j/T}}$$

Image from Learning Transferable Visual Models From Natural Language Supervision

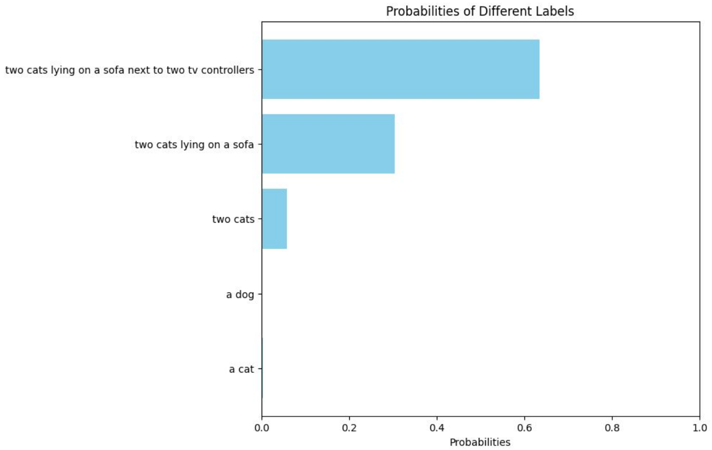
Producing embeddings with CLIP (½)

```
import torch
from transformers import CLIPProcessor, CLIPModel
from PIL import Image

# Load model and inputs
model =
CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
processor =
CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
image = Image.open("cats.jpg")
cat_text = ['a cat', 'a dog', 'two cats',
            'two cats lying on a sofa',
             “””two cats lying on a sofa
             next to two tv controllers”””]
```



Producing embeddings with CLIP (2/2)

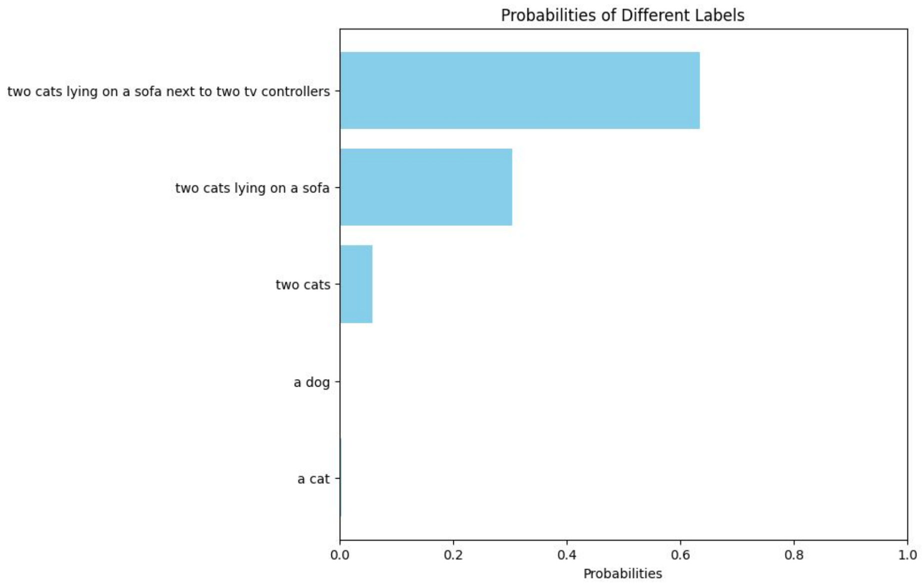
```python
# Get embeddings
inputs = processor(text=cat_text, images=image, return_tensors="pt",
padding=True)
outputs = model(**inputs)

# Calculate similarities
sims = torch.nn.functional.cosine_similarity(
    outputs.image_embeds[:, None],
    outputs.text_embeds[None, :],
    dim=-1
)

# Print results
for text, sim in zip(cat_text, sims[0]):
    print(f"{text}: {sim:.3f}")
```



Zero-shot classification with CLIP

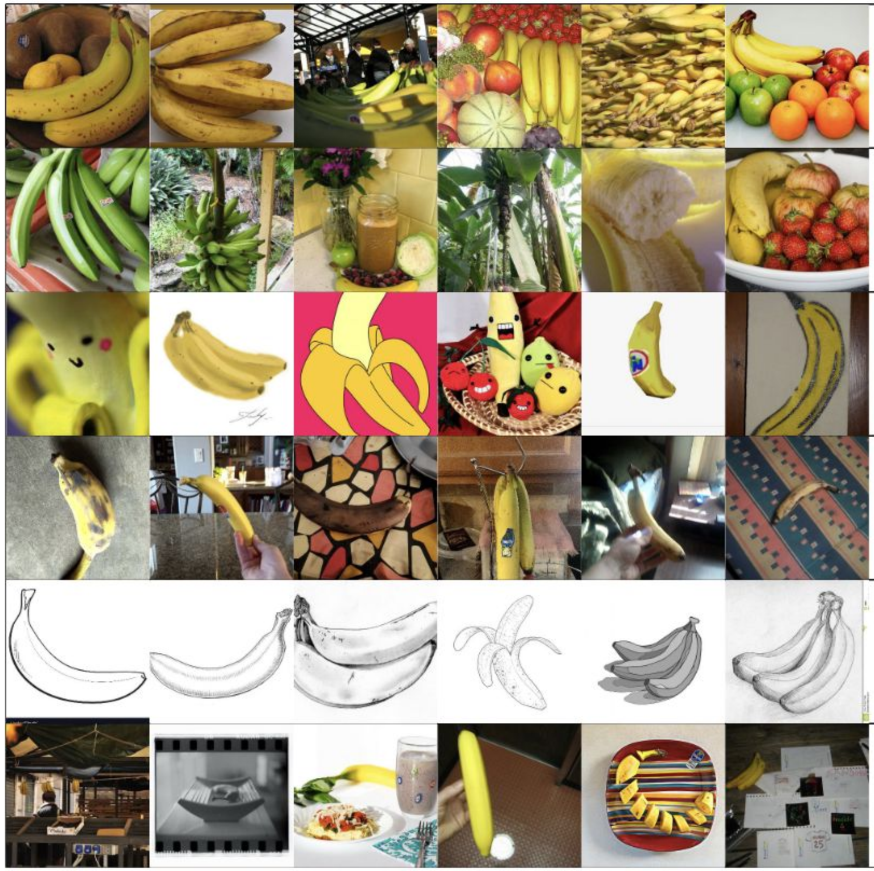
```
# Notice what happens with the output probabilities
# when we make the labels more specific
cat_text = ['a cat',
            'a dog',
            'two cats',
           ' two cats lying on a sofa',
            'two cats lying on a sofa next to two tv controllers'
            ]

inputs = processor(text=cat_text,
                   images=[cats_img],
                   return_tensors="pt", padding=True)
cat_outputs = model(**inputs)
cat_logits_per_image = cat_outputs.logits_per_image
cat_probs = (cat_logits_per_image/temperature).softmax(dim=1)
```

$$\text{softmax}(x_i, T) = \frac{e^{x_i/T}}{\sum_{j=1}^{n} e^{x_j/T}}$$



Transferable representations: CLIP against a ResNet101 pretrained on Imagenet

Dataset Examples



Dataset	ImageNet ResNet101	Zero-Shot CLIP	Δ Score
ImageNet	**76.2**	**76.2**	0%
ImageNetV2	64.3	**70.1**	+5.8%
ImageNet-R	37.7	**88.9**	+51.2%
ObjectNet	32.6	**72.3**	+39.7%
ImageNet Sketch	25.2	**60.2**	+35.0%
ImageNet-A	2.7	**77.1**	+74.4%

Note that CLIP was trained on a dataset of **400 million images** scraped from the Internet

The Imagenet1K dataset has only **1.28 million training images**

CLIP is a much more capable “foundation model”

Resnet101

~44.5 million parameters

CLIP ViT-L/14@336px:

~428 million parameters

Training Resnet101: about 90 V100 GPU hours (4 days)
Training CLIP ViT-L/14: about ~16,000 V100 GPU hours (666 days)

Limitations against fully supervised models

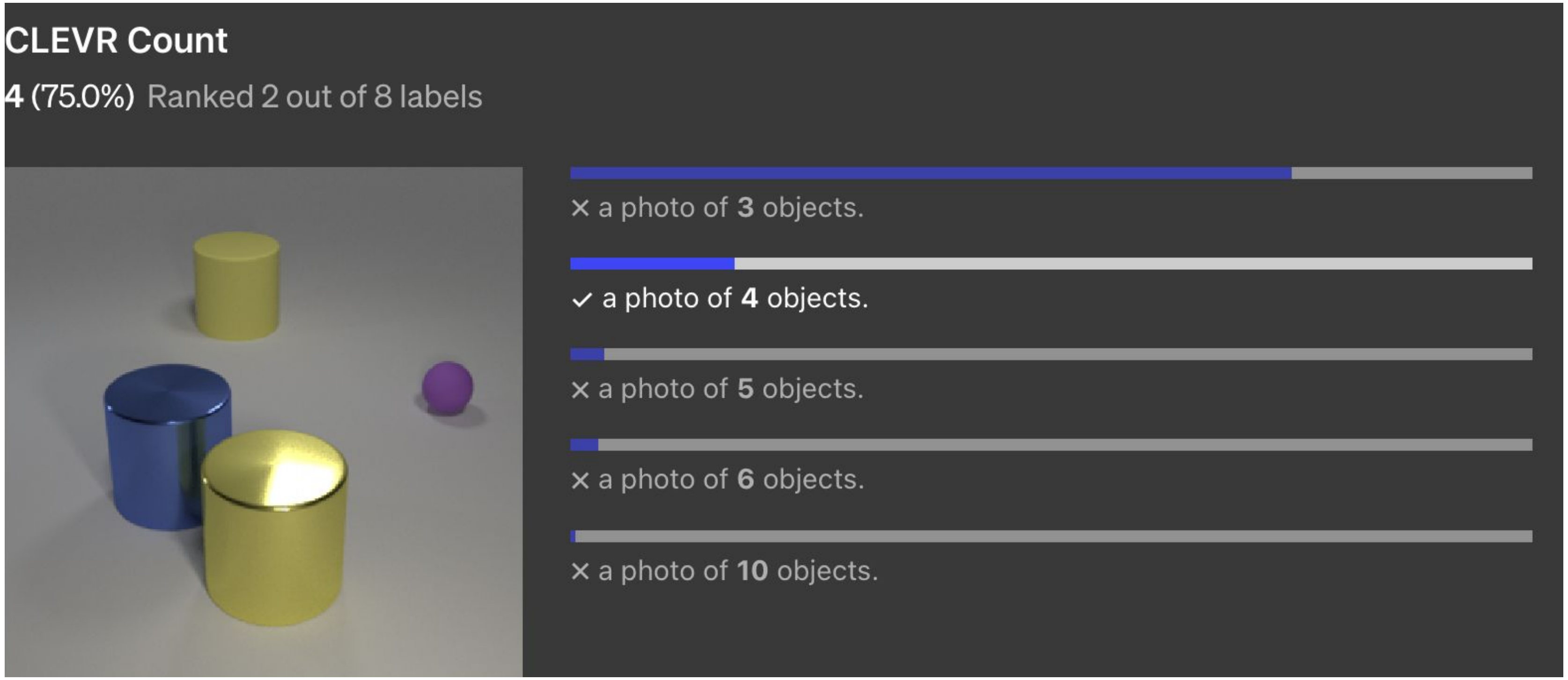
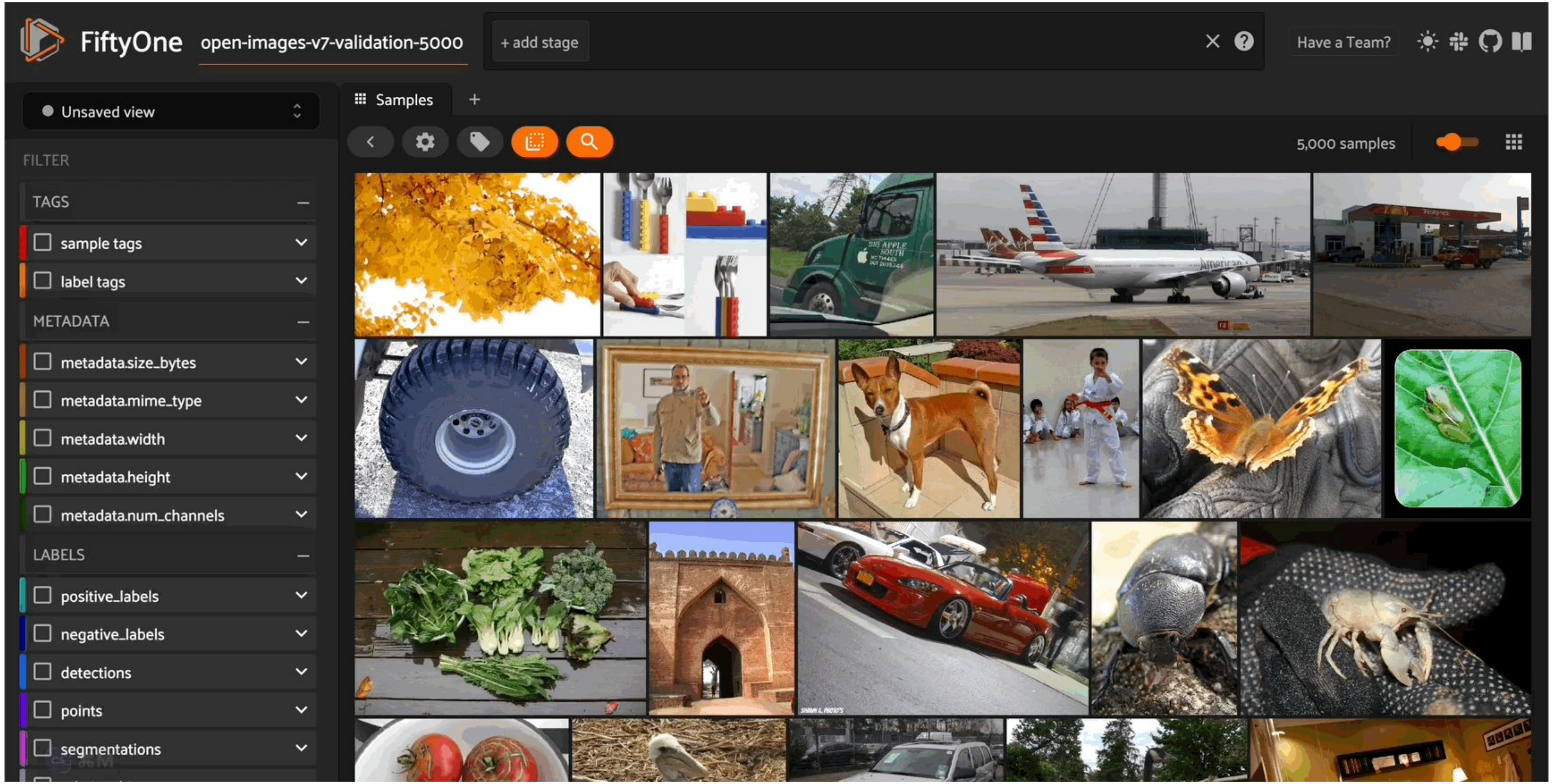


Image from
https://openai.com/index/clip/

The authors comment on the limitations section of the paper
that CLIP often fails to perform as well as a fully supervised model fine tuned for the task.
Notable examples:

- digit recognition (MNIST)
- tumor classification (PatchCamelyon)
- satellite imaging (EuroSAT)
- object counting (CLEVR Count)

Semantic search with CLIP



CLIP guides image generation of diffusion models

Prompt:

“Create an image of an astronaut riding a horse in pencil drawing style”

Image by OpenAI’s Dall-E 3

Summary

CLIP learns joint embeddings

- CLIP creates a shared embedding space for images and text
- These multimodal embeddings have applications on semantic search and image generation

CLIP excels at zero-shot classification

- CLIP performs well on unseen tasks without additional training

CLIP has limitations

- CLIP is not always better than models fine-tuned for specific tasks
- Retraining CLIP with a ViT base is expensive both computationally and data-wise

SPONSORED BY THE
Federal Ministry of Education and Research

Further reading

Learning Transferable Visual Models From Natural Language Supervision

- https://arxiv.org/abs/2103.00020

A Google Search Experience for Computer Vision Data

- https://voxel51.com/blog/a-google-search-experience-for-computer-vision-data/

Multi-modal ML with OpenAI's CLIP

- https://www.pinecone.io/learn/series/image-search/clip/

SPONSORED BY THE
Federal Ministry of Education and Research