CLIP

Contrastive Language Image Pre-training

learn associations between images and text across

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

https://medium.com/thedeephub/openai-clip-bridging-text-and-images-aaf3cd20299e#id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjJjOGEyMGFmN2ZjOThmOTdmNDRiMTQyYjRkNWQwODg0ZWIwOTM3YzQiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCJhenAiOiIyMTYyOTYwMzU4MzQtazFrNnFlMDYwczJ0cDJhMmphbTRsamRjbXMwMHN0dGcuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJhdWQiOiIyMTYyOTYwMzU4MzQtazFrNnFlMDYwczJ0cDJhMmphbTRsamRjbXMwMHN0dGcuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJzdWIiOiIxMTI1NjkzNTA3NTM3NDY5NjA2MTAiLCJlbWFpbCI6Im11aGFtbWFkZmFyb29xNDRAZ21haWwuY29tIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsIm5iZiI6MTczMzMyNjM0NSwibmFtZSI6Im11aGFtbWFkIGZhcm9vcSIsInBpY3R1cmUiOiJodHRwczovL2xoMy5nb29nbGV1c2VyY29udGVudC5jb20vYS9BQ2c4b2NJMG9jM0RyNzNMcG5PX3l0ZTQyVTFkN0pEaHRMdldYZm5tbnJnMC14MkNFWDZWMmxVMj1zOTYtYylsImdpdmVuX25hbWUiOiJtdWhhbW1hZCIsImZhbWlseV9uYW1lIjoiZmFyb29xliwiaWF0IjoxNzMzMzl2NjQ1LCJleHAiOjE3MzMzMzAyNDUsImp0aSI6IjVhZTI0MTMwYWViNWViOTJlYjcyMjNlZDhhY2MxYTU5OTViYmlwYzQifQ.bZqhKTYOxrbIP5EomzkKvyyAf6h-cuYqK0_j9uiaT8fEv76KytC6VmaV3PXoyiPQIt_K2MKSibHEQRhE3KuRzJr--ATue-YUX_F42t0hlZRHqpfvKjpuzIm6N2ujYqO31yYX_ArqzAf4h4CiWEIGqeFZ7X0pwCnz6o31348vKO9Uly6whS57gqu7D61JuzoTicv1wnLR714_XSeRgwqhhtVuI8iZNhJe8bVfyKocdvFN6bqI9bSRRtb9FkjkTExBWoWobweMmU3bMtTUsNGJE_8heYGOoHdzK_ixMcEtP0-ej15xvQJkpzwzu9BWmjRWxGTaOwygxdidJdhtkscfow

https://www.crcv.ucf.edu/courses/cap6412-spring-2024/schedule/ (Mubarak Shah)

https://www.youtube.com/watch?v=jXD6O93Ptks&t=8s

https://medium.com/@VeryFatBoy/quick-tip-build-vector-embeddings-for-video-via-python-notebook-openai-clip-809e3ce5cd17

- **Topics:**
  - **Clip (Variant of Vision transformer)**
  - **Contrastive Learning**
  - **Self-Supervised Learning**
  - **Zero short learning**
  - **Linear prob vs zero short learning**
  - **Implementation – for image retrieval**

Learning Transferable Visual Models from Natural Language Supervision

Alec Radford JongWook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, lya Sutskever

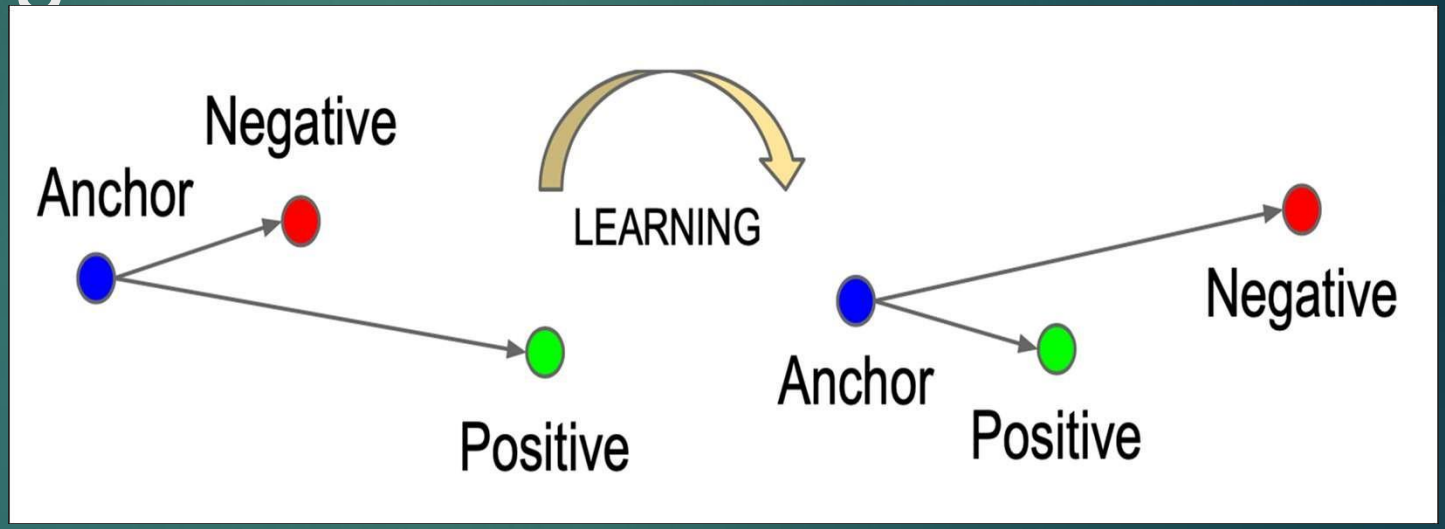
ICML-2021; 25027 Citations

Contrastive Language Image Pre-training (CLIP)

- Mechanism for natural language supervision (image caption) (self supervision: learning from descriptions rather than fixed labels can be very data efficient)
- Pair an image with it's caption using contrastive learning
- Beats fully supervised learning baseline on many datasets
- Can be used as a zero-shot classifier (make it more scalable due to zero short learning compare to simple image captioning)

What is Contrastive Learning?

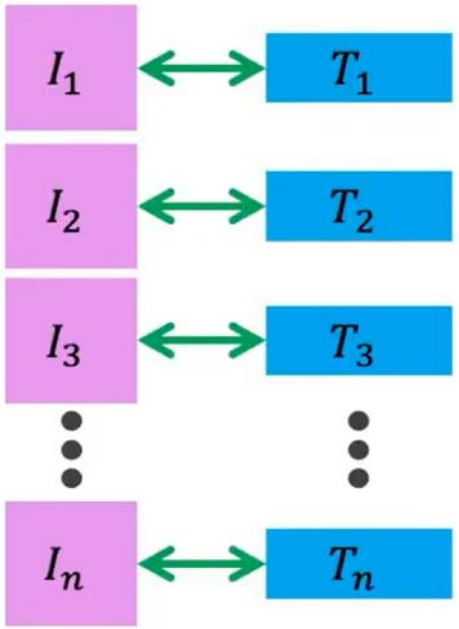
•**Goal**:
Learn representations by **bringing similar data closer** in the embedding space while **pushing dissimilar data apart**.



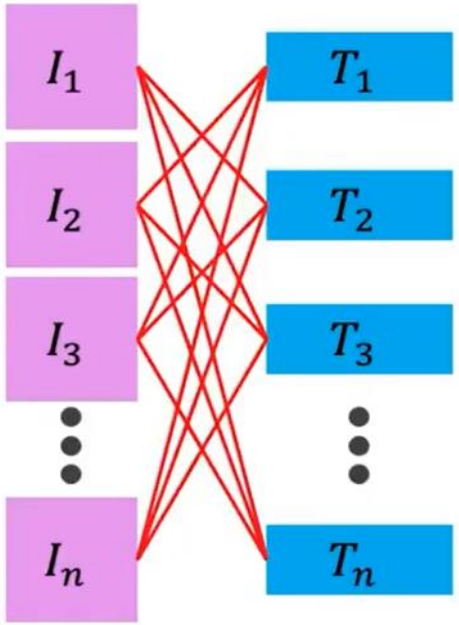
- **Anchor**: A sample (e.g., an image or text) used as the reference point.
- **Positive**: A sample similar to the anchor (e.g., a text that describes the anchor image).
- **Negative**: A sample dissimilar to the anchor (e.g., a text unrelated to the anchor image

CLIP Image-Text Pair Matching

- Increase the cosine similarity of **correct** pairs in a batch



- Reduce the cosine similarity of $n^2 - n$ **incorrect** pairings



Contrastive Learning Objective - similar (image, text) pair

Input Image

$\vec{H}_i$

Image Representation

A dog lying in grass

Input Text

$\vec{H}_t$

Text Representation

$$maximize(\frac{\vec{H}_i \cdot \vec{H}_t}{\|\vec{H}_i\| \times \|\vec{H}_t\|})$$

Contrastive Learning Objective - dissimilar (image, text) pair

Input Image

$\vec{H}_i$

Image Representation

A dog lying in grass

Input Text

$\vec{H}_t$

Text Representation

$$minimize(\frac{\vec{H}_i \cdot \vec{H}_t}{\|\vec{H}_i\| \times \|\vec{H}_t\|})$$

Cosine Similarity: Measuring **Directional Similarity**

$$\vec{A} = [3, 4], \quad \vec{B} = [6, 8]$$

Step 1: Dot Product

The dot product of two vectors $\vec{A} \cdot \vec{B}$ is calculated as:

$$\vec{A} \cdot \vec{B} = (3 \cdot 6) + (4 \cdot 8) = 18 + 32 = 50$$

Step 2: Norms (Magnitudes) of Each Vector

The norm of a vector $\vec{A}$ (denoted as $\|\vec{A}\|$) is computed using the formula:

$$\|\vec{A}\| = \sqrt{x^2 + y^2}$$

For $\vec{A} = [3, 4]$:

$$\|\vec{A}\| = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = 5$$

For $\vec{B} = [6, 8]$:

$$\|\vec{B}\| = \sqrt{6^2 + 8^2} = \sqrt{36 + 64} = 10$$

Step 3: Cosine Similarity Formula

The cosine similarity is given by:

$$\text{cosine similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|}$$

Substituting the values:

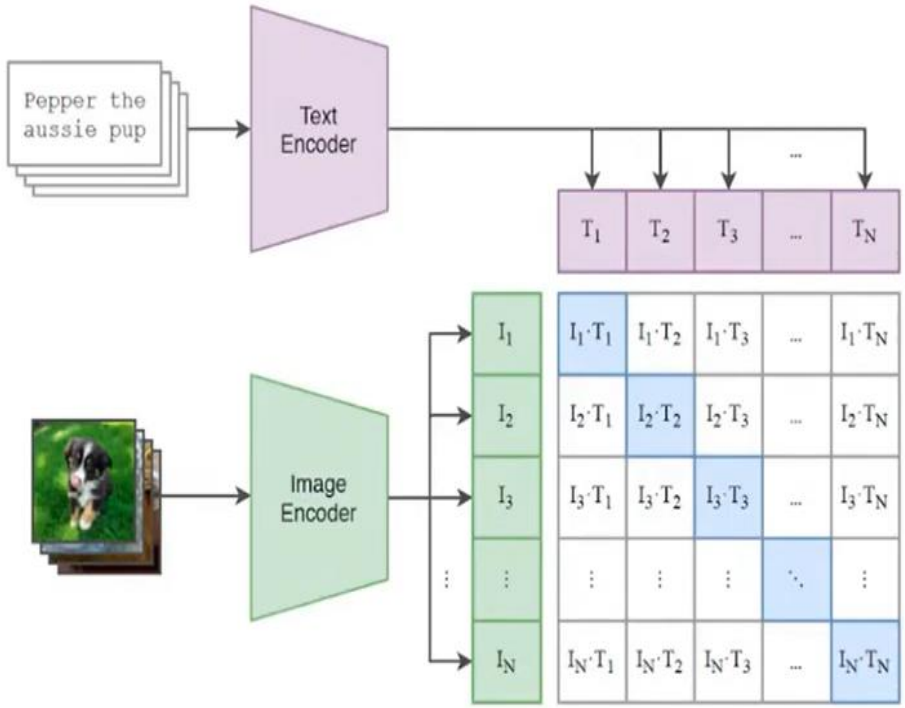
$$\text{cosine similarity} = \frac{50}{5 \cdot 10} = \frac{50}{50} = 1$$

Interpretation

- The cosine similarity is 1, which indicates that A and B point in the same direction.

- Even though B is a scaled version of A (it is twice as large), cosine similarity focuses only on the angle (direction) between vectors, ignoring magnitude differences.

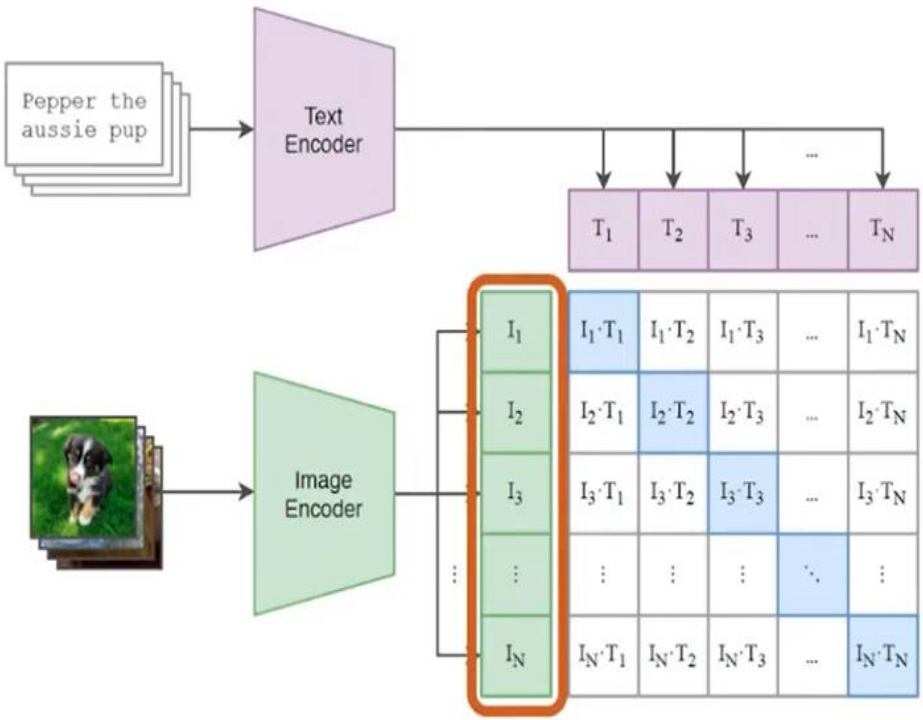
CLIP Contrastive Pre-training



https://arxiv.org/pdf/2103.00020

```
# 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
```

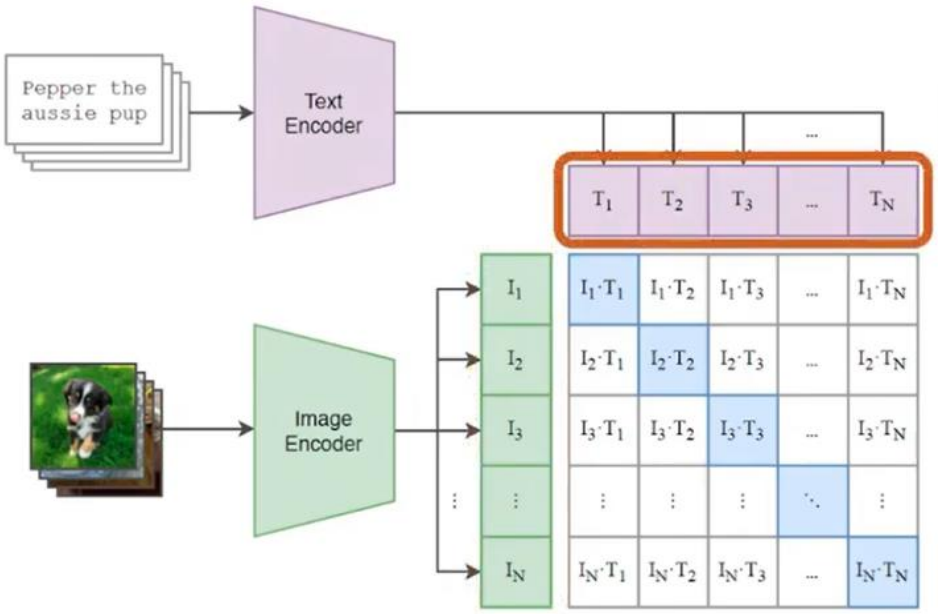
CLIP Contrastive Pre-training



https://arxiv.org/pdf/2103.00020

```
# 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
```

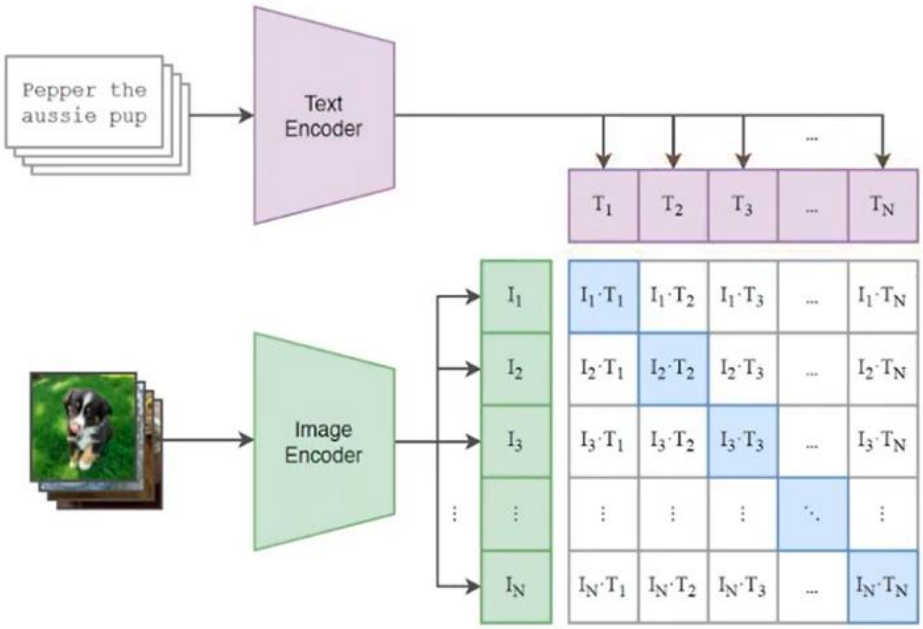
CLIP Contrastive Pre-training



https://arxiv.org/pdf/2103.00020

```
# 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
```

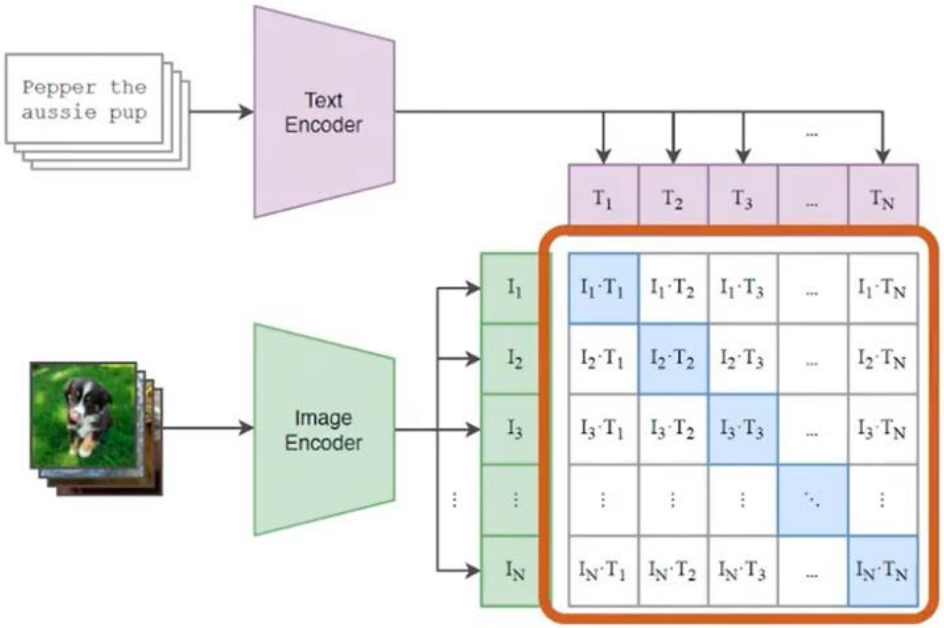
CLIP Contrastive Pre-training



https://arxiv.org/pdf/2103.00020

```
# 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
```

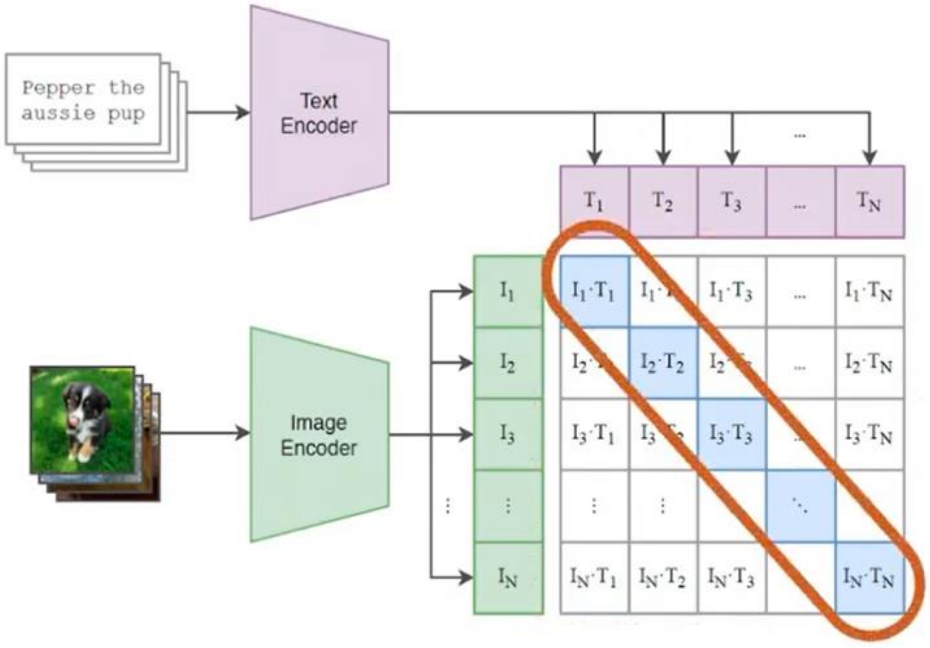
CLIP Contrastive Pre-training



https://arxiv.org/pdf/2103.00020

```
# 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
```

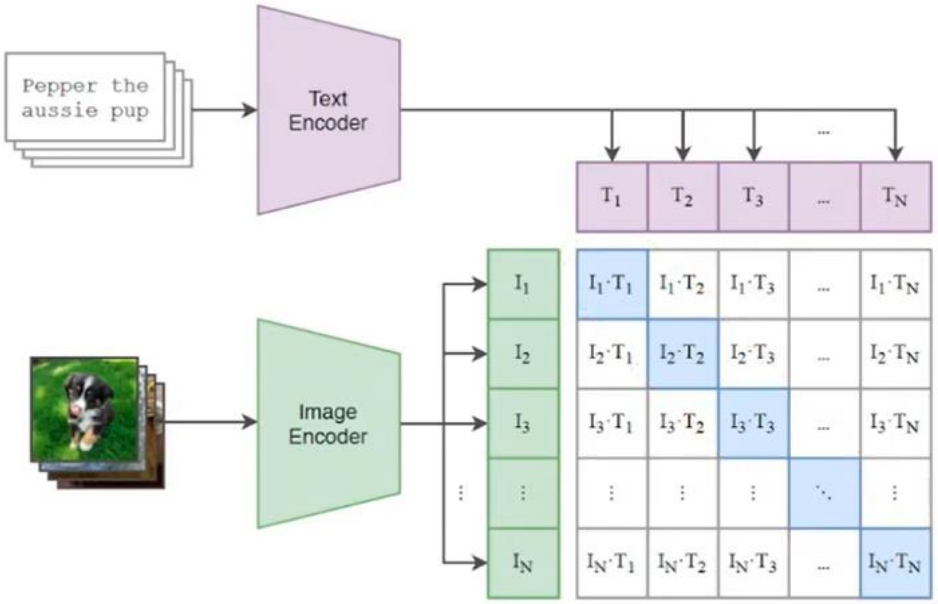
CLIP Contrastive Pre-training



https://arxiv.org/pdf/2103.00020

```
# 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 Contrastive Pre-training

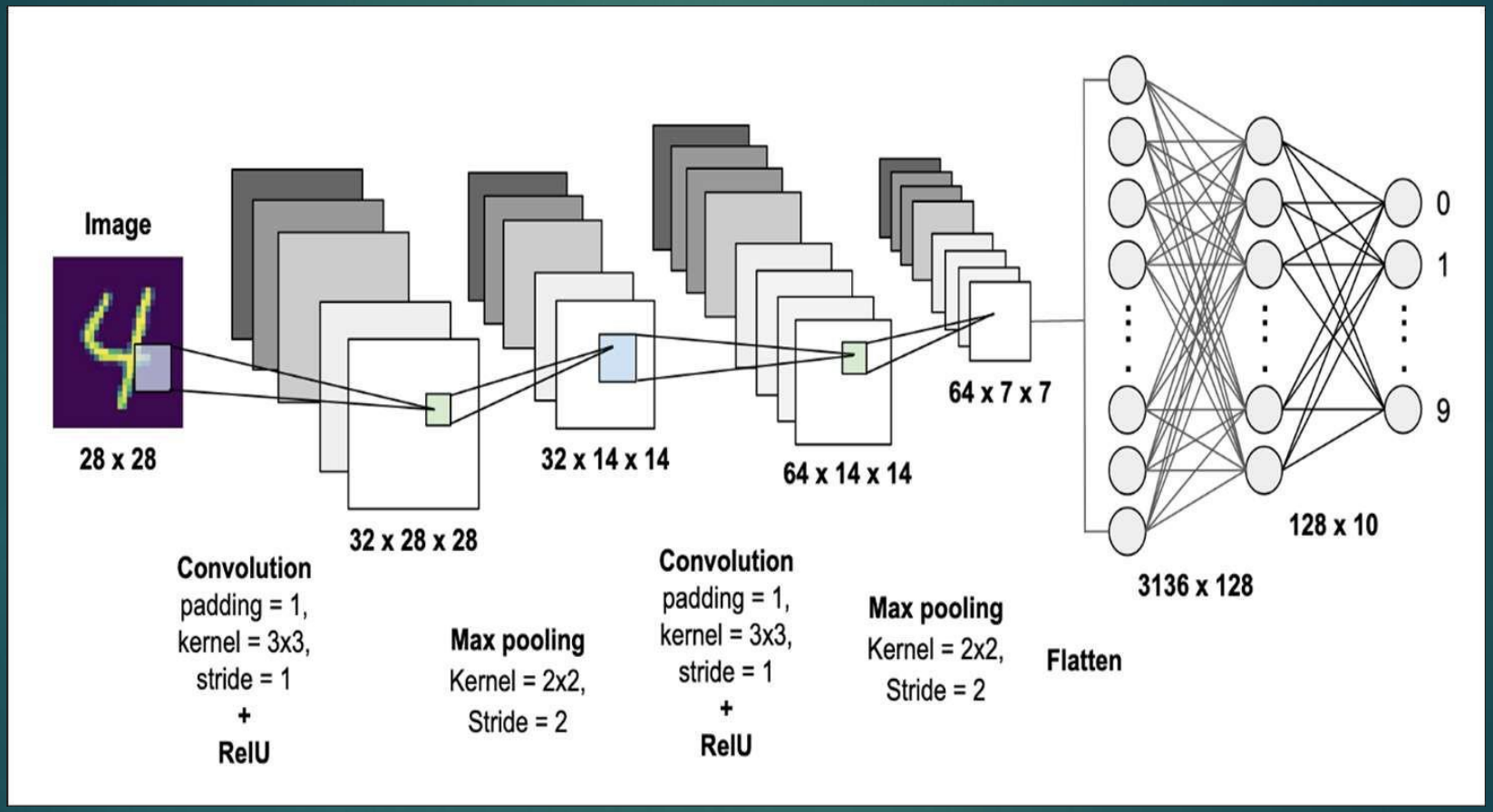


https://arxiv.org/pdf/2103.00020

```
# 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
```

See toy Example

Supervised Classification



Supervised Learning vs Zero Shot Learning

SUPERVISED LEARNING

- Labeled data
- Training phase
- Final prediction on labeled data

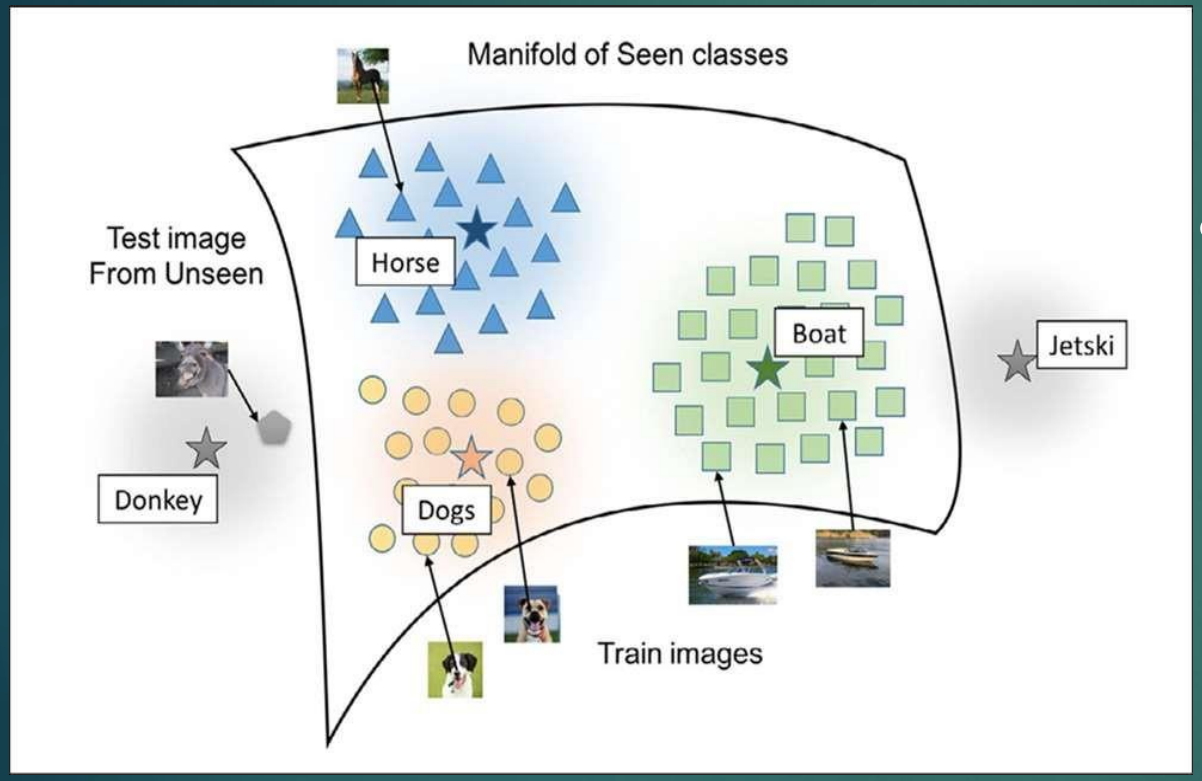
ZERO SHOT LEARNING

- Data can be labeled/unlabeled
- No training
- Accuracy on the final results

Motivation of zero shot learning

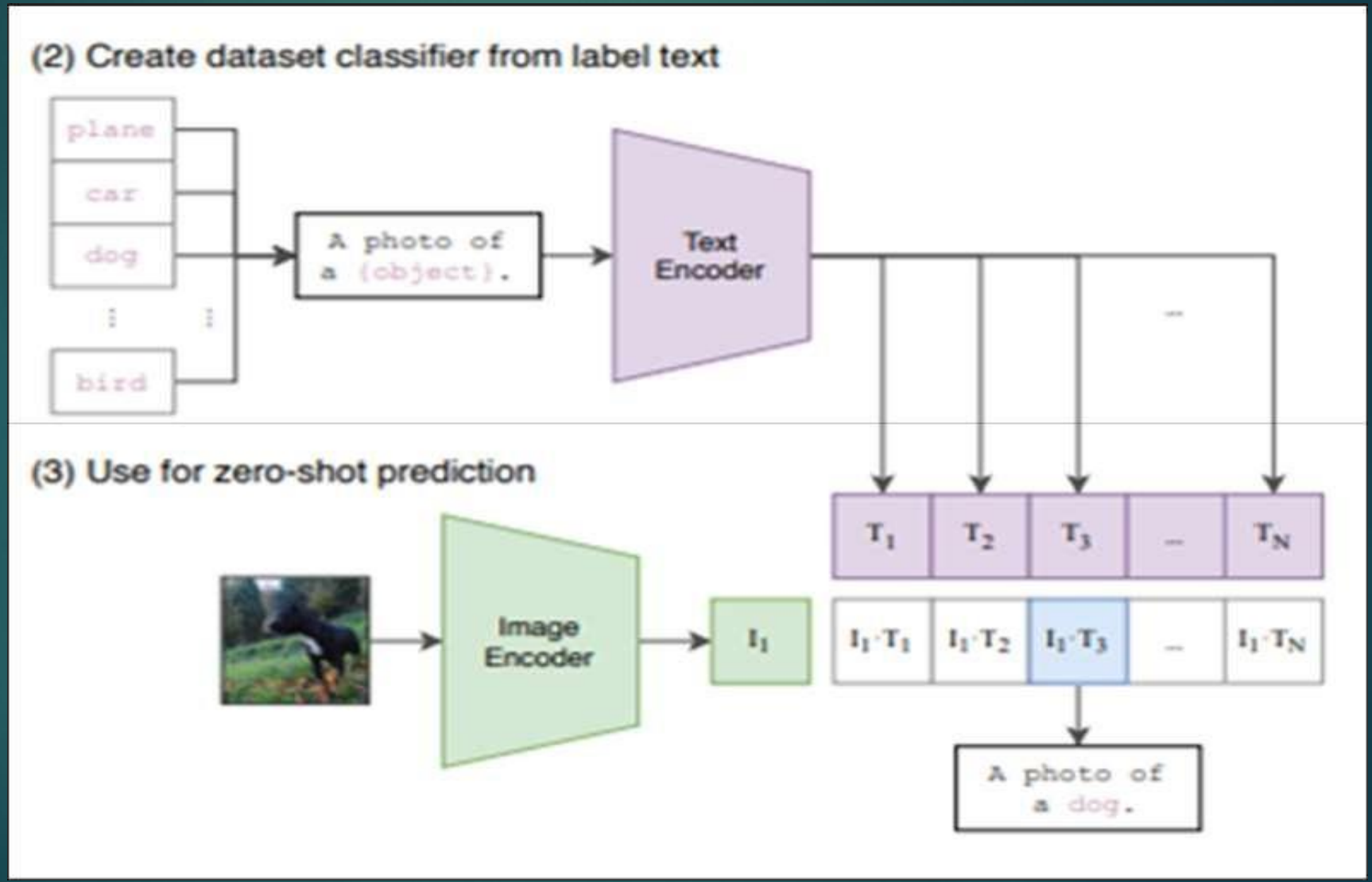
- Supervised learning like Image classification models are limited:
  - Fixed number of labels(The ImageNet dataset, one of the largest efforts in this space, required over 25,000 workers to annotate 14 million images for 22,000 object categories. In contrast, CLIP learns from text–image pairs that are already publicly available on the internet.)
  - Generalization
- CLIP overcomes these limitations.

What is Zero-Shot Learning? IDEA:



- To train on one dataset and generalizing on unseen categories.

CLIP for Zero-Shot Image Classificaiton



WebImageText(WIT) Dataset

- Motivation for using natural language is the vast amounts of data

- **Dataset Construction: 400M Image/Text Pairs**

- **500,000 High-Frequency Words**

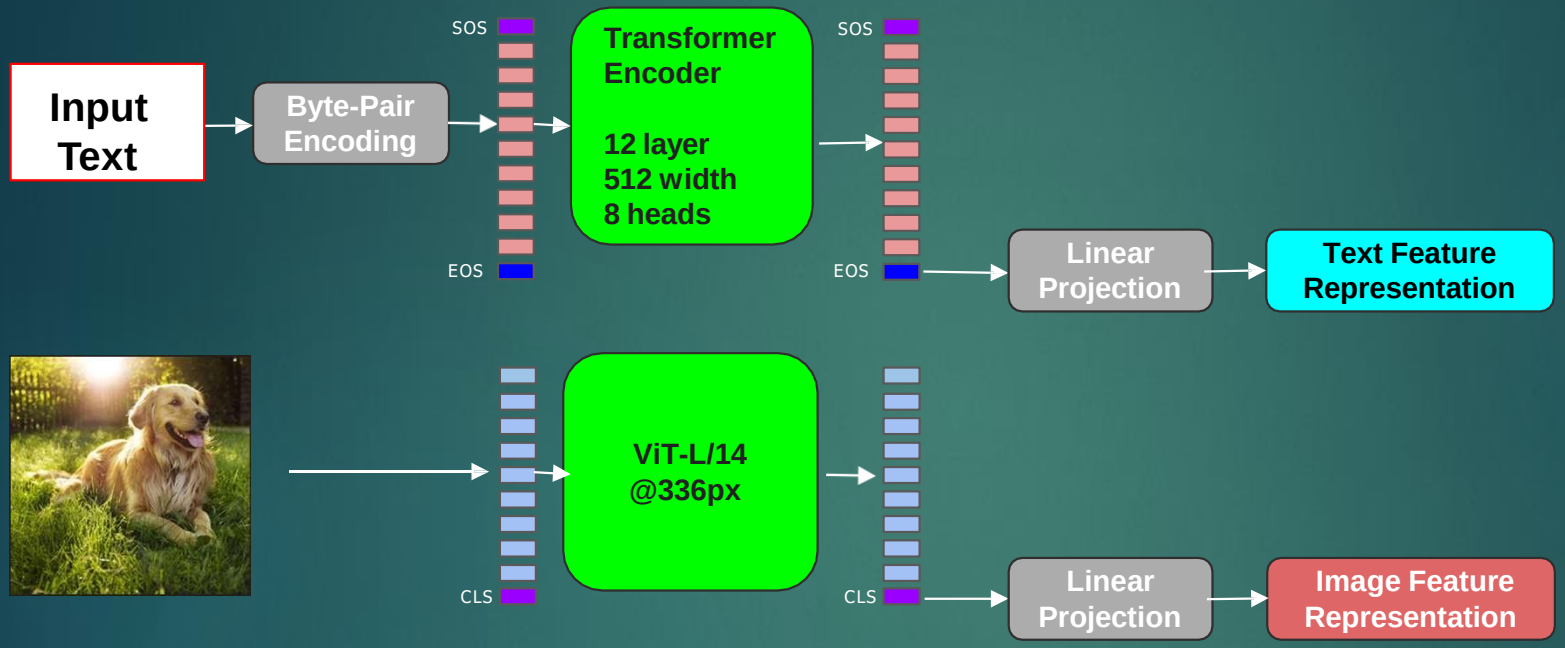
- Ensuring **balance** in the dataset used for training models like CLIP by **Setting the Limit (Cap)**:

  a. A maximum of **20,000 image-text pairs per query** was allowed.
  b. For example:
     1. If a query like **"cat"** retrieves 100,000 pairs, only the **top 20,000 pairs** are included in the dataset.
     2. If a query like **"ammonite"** retrieves only 5,000 pairs, all of them are used (as it's below the cap).

- Used for pre-training CLIP

  Pretraining refers to the process where CLIP acquires general knowledge by training on a large dataset in its initial phase. This knowledge can then be applied to a wide range of tasks, such as zero-shot learning, image retrieval, or serving as a validator in image generation models.

CLIP Architecture



*Authors also tested many other ResNet/ViT variants, but found this ViT to perform the best

- The model learns through **contrastive learning**:
  - For a **given input image and text pair**, their feature representations are **brought closer together** in the shared embedding space.
  - **Dissimilar image-text pairs are pushed farther apart.**
  - This enables the model **to match text to images** (and vice versa) effectively.

Steps for Inputting Text to the Text Encoder:

1. **Tokenization (BPE):**
    1. Example: "playing soccer" → ['play', 'ing', 'soc', 'cer'].
2. **Mapping to Numeric IDs:**
    1. Example: ['play', 'ing', 'soc', 'cer'] → [345, 567, 890, 234].
3. **Adding Special Tokens:**([SOS]) and [EOS] are added.
4. **Embedding Layer:** converting into dense vector representation
    1. The numeric IDs are passed through a **learned embedding layer**, converting them **into dense vector** representations.
5. **Positional Encoding**
6. **Transformer Encoder**

Input Image:

- The image is processed using a **Vision Transformer (ViT-L/14)** with an input resolution of **336×336 pixels**.
- This ViT splits the image into smaller patches and processes them as a sequence (like words in a sentence).
  - **L**: A large ViT model.
  - **14**: The size of each image patch (14x14 pixels).
- The image input resolution is **336×336 pixels**, which means each input image is resized to this fixed size before processing.
- L mean is 24 transformer layers (self attention and feed forward repeat 24 times)

Some CLIP details

Training

- Trained on 400M image-text pairs from the internet
- Batch size of 32,768
- 32 epochs over the dataset

Cosine learning rate decay :

- Cosine learning rate decay is a schedule for adjusting the learning rate during the training process of a machine learning model. .
- The learning rate starts high and decreases following a cosine curve, which allows the model to converge smoothly and avoid overshooting or plateauing.)

Architecture

- ResNet-based or ViT-based image encoder
- Transformer-based text encoder

Testing

After training the CLIP's image encoder and text encoder use it for

1. Linear Probe: **Use the trained image encoder and add linear layer and trained it for classification (Fine turning)**

2. Zero-shot Prediction: **from CLIP's pre-training to directly perform tasks without any additional training on the target dataset**

Linear Probe CLIP- fine tuning

- Train a linear classifier on another dataset using CLIP features

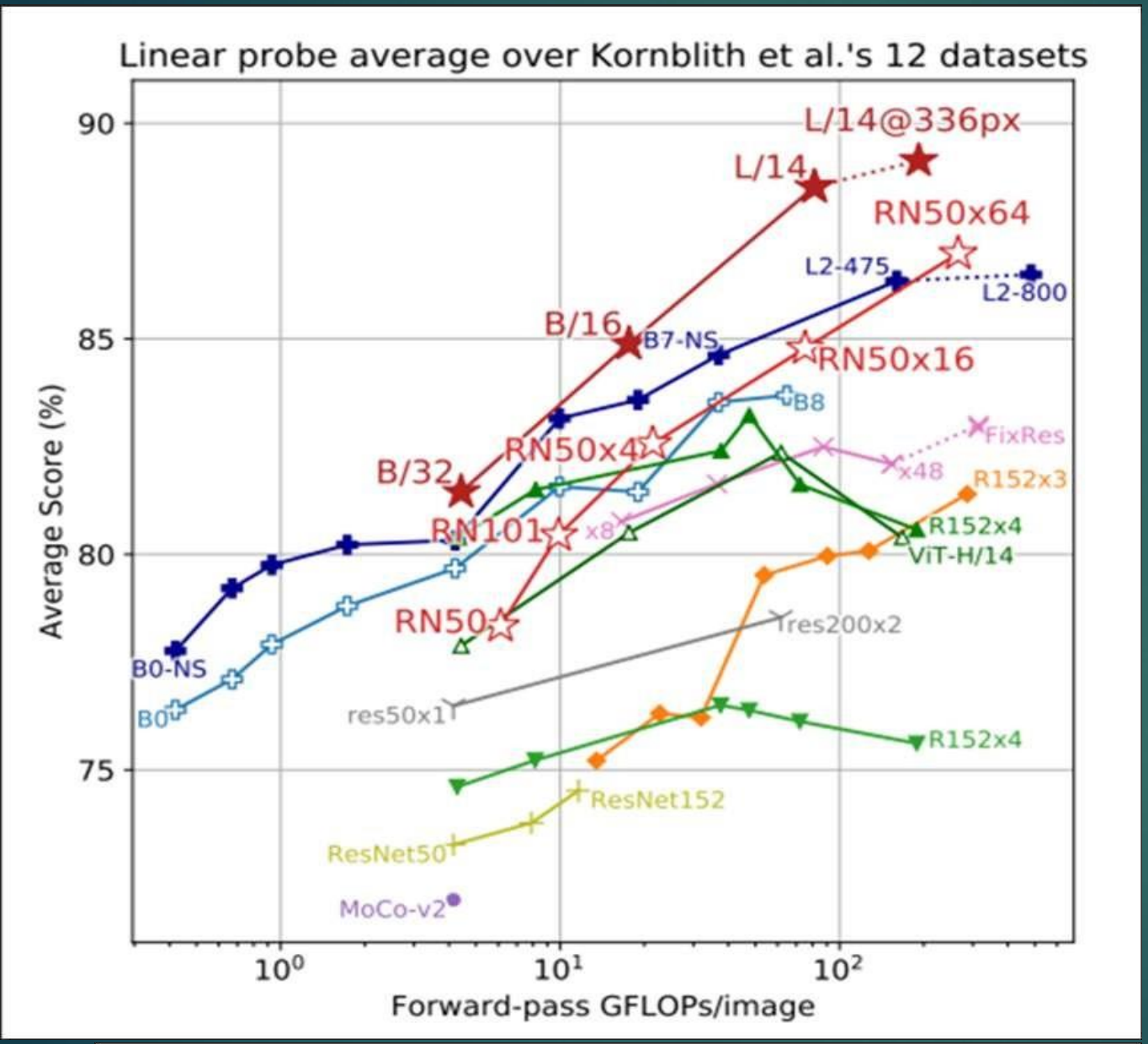
Kornblith et al.'s 12 datasets

Dataset	Classes	Train size	Test size	Evaluation metric
Food-101	102	75,750	25,250	accuracy
CIFAR-10	10	50,000	10,000	accuracy
CIFAR-100	100	50,000	10,000	accuracy
Birdsnap	500	42,283	2,149	accuracy
SUN397	397	19,850	19,850	accuracy
Stanford Cars	196	8,144	8,041	accuracy
FGVC Aircraft	100	6,667	3,333	mean per class
Pascal VOC 2007 Classification	20	5,011	4,952	11-point mAP
Describable Textures	47	3,760	1,880	accuracy
Oxford-IIIT Pets	37	3,680	3,669	mean per class
Caltech-101	102	3,060	6,085	mean-per-class
Oxford Flowers 102	102	2,040	6,149	mean per class

Extended 27 Datasets

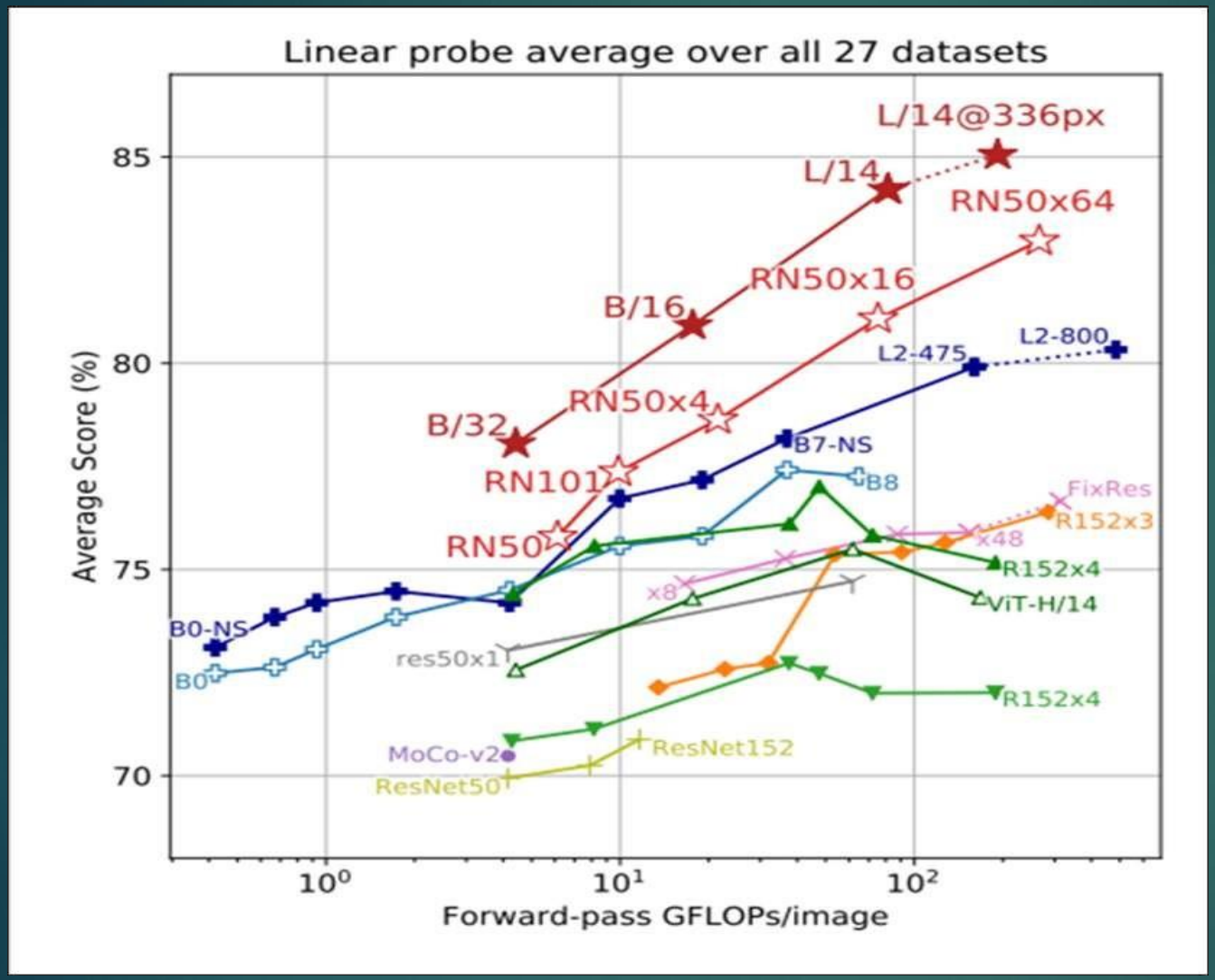
Dataset	Classes	Train size	Test size	Evaluation metric
Food-101	102	75,750	25,250	accuracy
CIFAR-10	10	50,000	10,000	accuracy
CIFAR-100	100	50,000	10,000	accuracy
Birdsnap	500	42,283	2,149	accuracy
SUN397	397	19,850	19,850	accuracy
Stanford Cars	196	8,144	8,041	accuracy
FGVC Aircraft	100	6,667	3,333	mean per class
Pascal VOC 2007 Classification	20	5,011	4,952	11-point mAP
Describable Textures	47	3,760	1,880	accuracy
Oxford-IIIT Pets	37	3,680	3,669	mean per class
Caltech-101	102	3,060	6,085	mean-per-class
Oxford Flowers 102	102	2,040	6,149	mean per class
MNIST	10	60,000	10,000	accuracy
Facial Emotion Recognition 2013	8	32,140	3,574	accuracy
STL-10	10	1000	8000	accuracy
EuroSAT	10	10,000	5,000	accuracy
RESISC45	45	3,150	25,200	accuracy
GTSRB	43	26,640	12,630	accuracy
KITTI	4	6,770	711	accuracy
Country211	211	43,200	21,100	accuracy
PatchCamelyon	2	294,912	32,768	accuracy
UCF101	101	9,537	1,794	accuracy
Kinetics700	700	494,801	31,669	mean(top1, top5)
CLEVR Counts	8	2,000	500	accuracy
Hateful Memes	2	8,500	500	ROC AUC
Rendered SST2	2	7,792	1,821	accuracy
ImageNet	1000	1,281,167	50,000	accuracy

Results - Efficiency – Kornblith 12 dataset



- CLIP's ResNet based model 0underperforms EfficientNet
- ViT based CLIP outperforms everything

Results - Efficiency - Extended

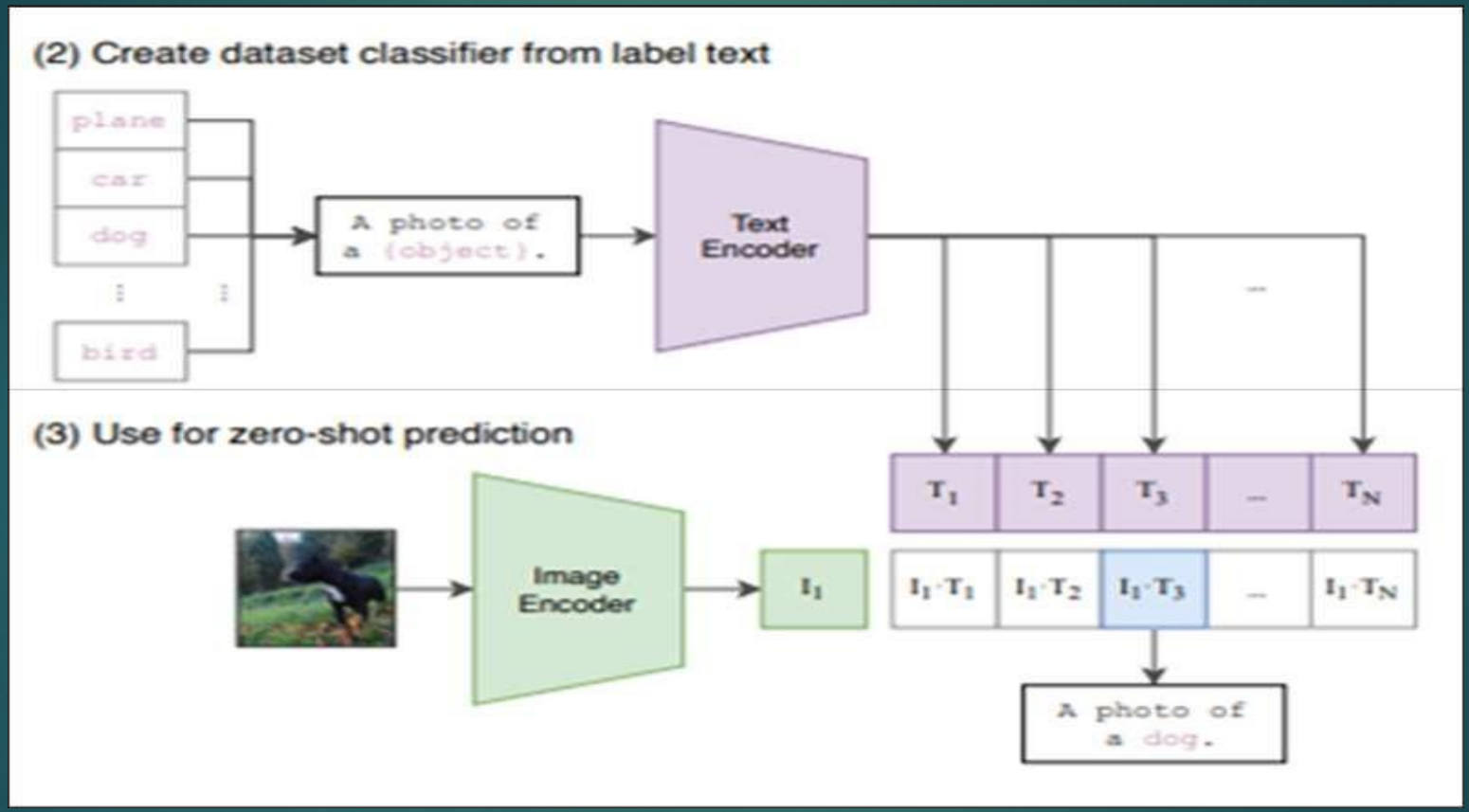


- On the extended testing suite, both CLIP versions outperform all other models
- Performance gap increases with GFLOPS

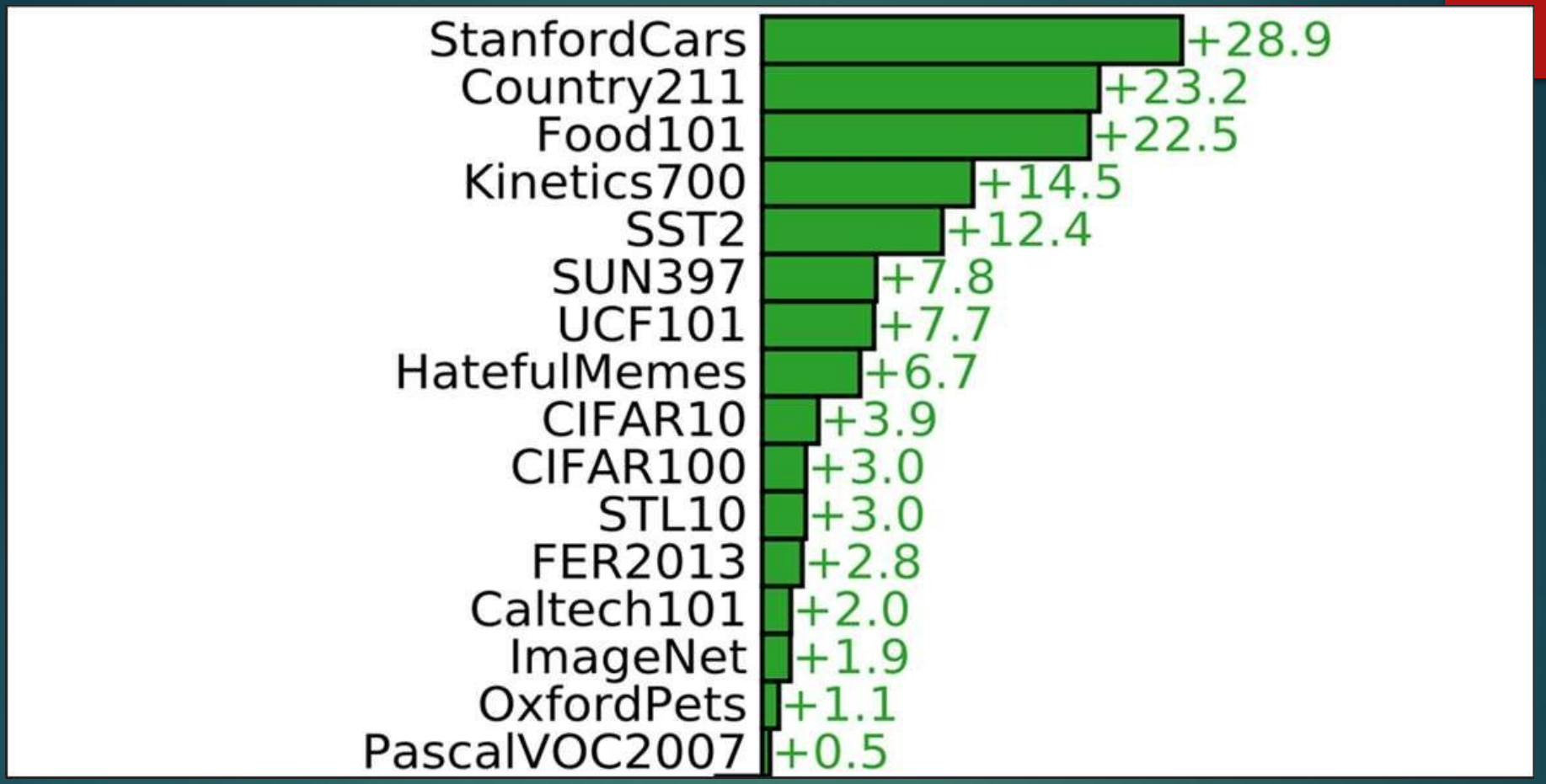
Zero-Shot

CLIP's pre-training to directly perform tasks without any additional training on the target dataset

Zero shot prediction

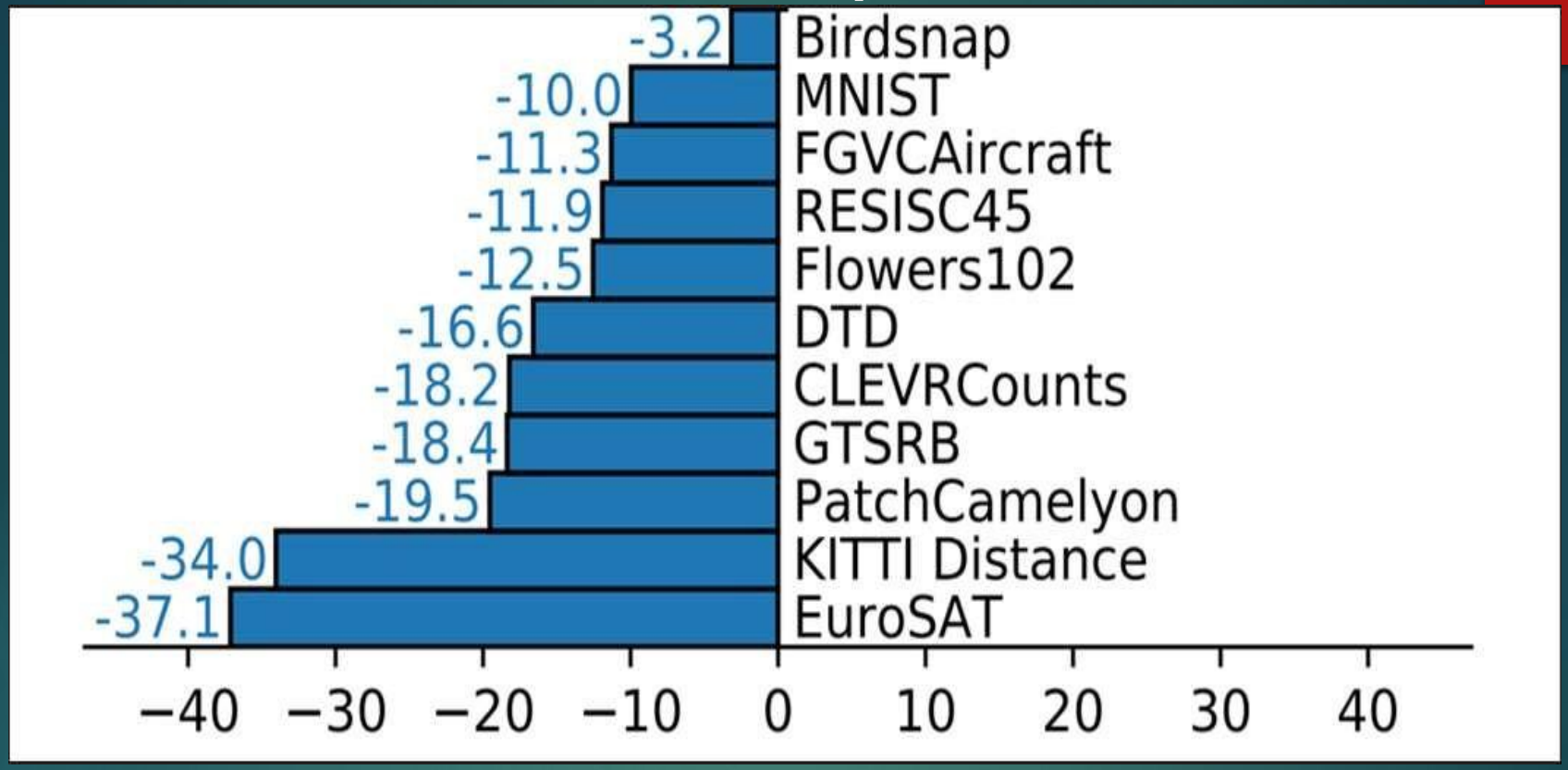


Linear prob vs zero sort Results - Accuracy



- Zero-shot CLIP using ResNet50 backbone is compared to off the shelf ResNet50 (means that the ResNet50 architecture is being utilized as the image encoder within the CLIP framework)
- CLIP outperforms on a wide variety of popular datasets
- For video, a single frame was sampled

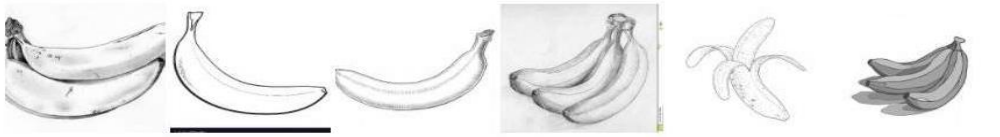
Results - Accuracy



Δ Score (%)
Zero-Shot CLIP vs. Linear Probe on ResNet50

- Underperforms on many other datasets
- Mostly on specialized/complex datasets
- EuroSAT for satellite images, Tumor classification
- Makes intuitive sense, Zero-shot CLIP is highly generalized
- Not suited for hyper specific tasks unless fine-tuned

Zero-shot CLIP is much more robust

DATASET	IMAGENET RESNET101	CLIP VIT-L
 ImageNet	76.2%	76.2%
 ImageNet V2	64.3%	70.1%
 ImageNet Rendition	37.7%	88.9%
 ObjectNet	32.6%	72.3%
 ImageNet Sketch	25.2%	60.2%
 ImageNet Adversarial	2.7%	77.1%

Demo
image retrieval

https://colab.research.google.com/drive/1hYHb0FTdKQCXZs3qCwVZnSuVGrZU2Z1w?usp=sharing#scrollTo=bvXzfQqgi6IT
https://www.youtube.com/watch?v=GLa7z5rkSf4