Understanding OpenAI CLIP & Its Applications

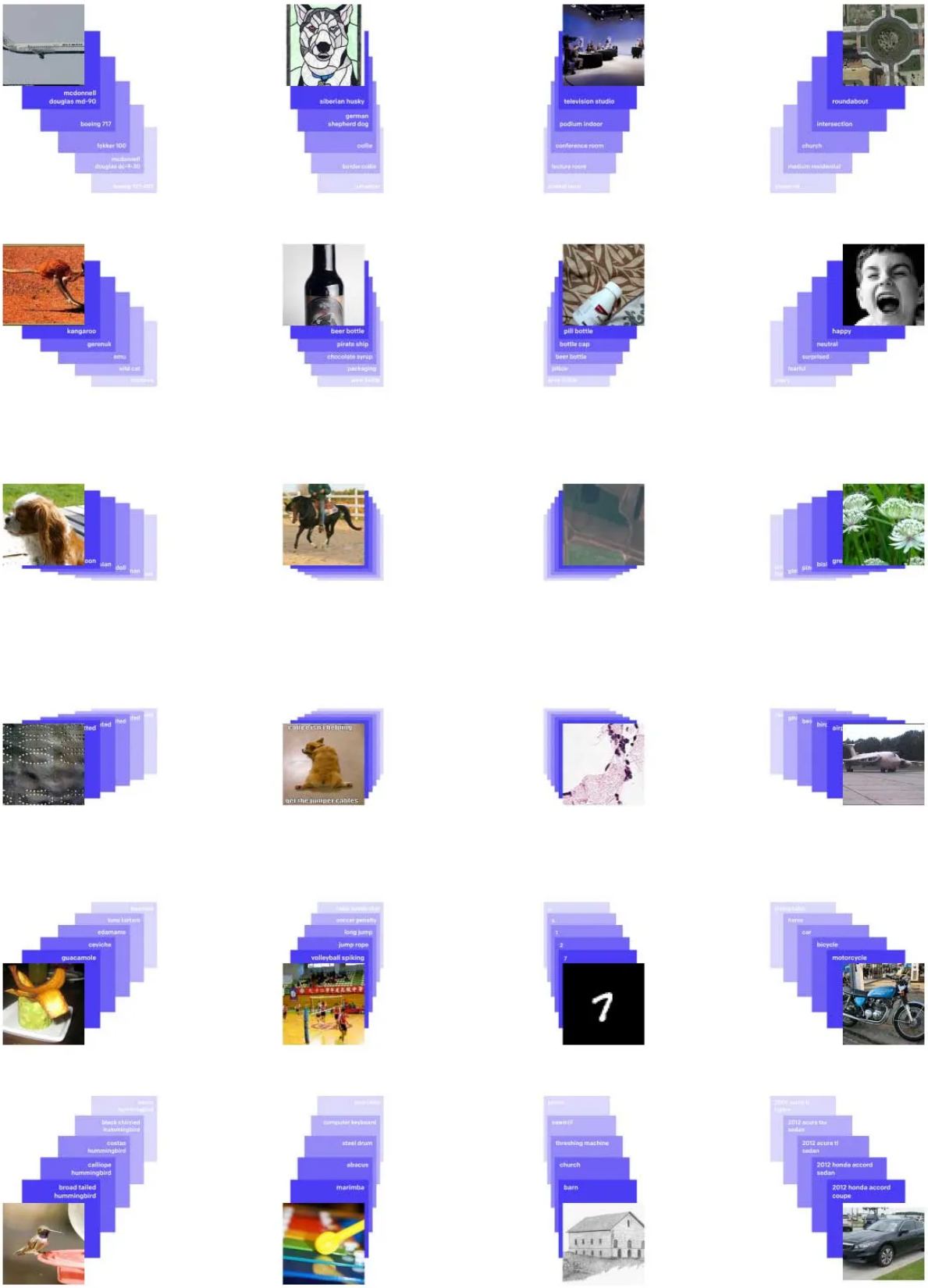
Anshu Kumar · Follow
6 min read · Nov 19, 2022

Hands-On examples of Image Search & Reverse Image Search

In this Article I would cover the the following

- CLIP Architecture and Embedding.
- Core Implementation, Key Ingredients and Training Approach
- *Application of CLIP Embeddings to build Image Search and Reverse Image Search*
- Key Takeaways and Limitations

Let's get Started!!



CLIP: (Contrastive Language–Image Pre-training)

CLIP Learns visual concepts from natural language supervision.

Supervised computer vision systems are trained/fine-tuned with fixed set of labels. This limits the capability of the model as every time model is trained when new label is encountered.

The CLIP uses the prior work of VirTex[1] and ConVIRT[2].

Image descriptions acted as proxy labels for training CLIP. CLIP is helper model of DALL-E.

CLIP can tell whether a text description matches to the image or not.

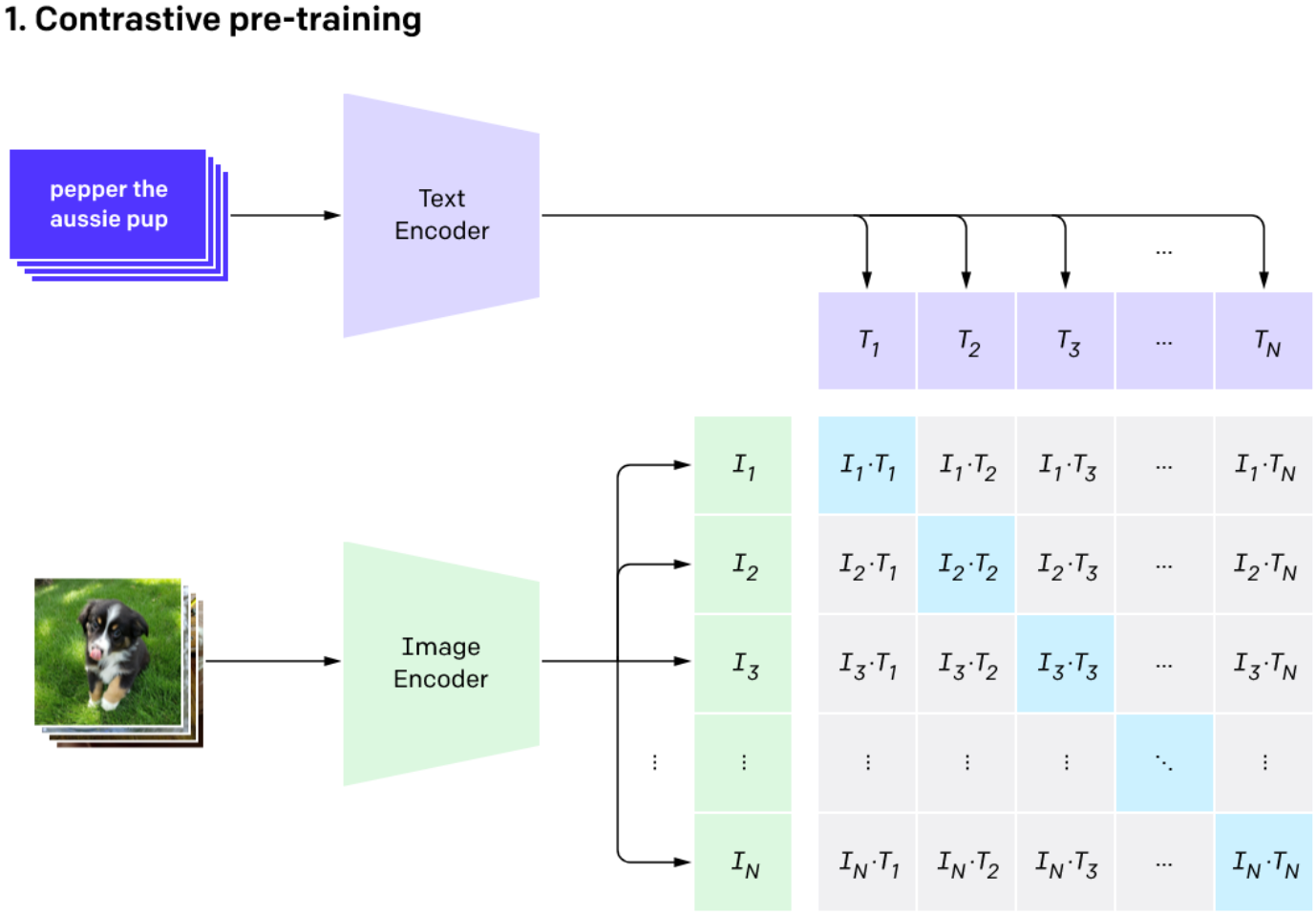
CLIP is zero shot by design hence not restricted to number of labels. It does not constrain the model to reduce to one concept or a label. This helps CLIP to learn multiple concepts present in image. Hence CLIP performs reasonably well on unseen image datasets and text descriptions.

What are the key ingredients of CLIP?

1. Large Dataset:

CLIP is trained over WebImage Text(WIT) 400M image-text pair. Diverse dataset crawled from internet. More data is better.

2. Contrastive Pre-Training:



I and T vectors represent the embeddings of Image Batch and Text Batch. I_i T_i represents the dot product of matched Image and Text Embeddings. Off diagnonal elements are not matching Image and Text Description.

In contrastive learning we are trying to maximise diagonal value (I_1,T_1), (I_2,T_2)... I_N,T_N while minimising off diagonal elements.

Most of the learning coming from negative image description. In a batch of 32,768 there is only one positive pair. CLIP's learns most by, what this image is **not** about. I mean the model learns a lots from off diagonal values minimisation.

Core Implementation of CLIP

```
# 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
```

Core Implementation of CLIP, and loss calculation

3. Computational Efficiency

a. Contrastive objective for connecting text with images.

b. Adoption of the Vision Transformer. Gained 3x compute efficiency compared to ResNet model. Vision Transformer is used as Image Encoder.

c. Web Scale supervision seems to surpass manual curation dataset.

Major Problems in Deep Learning and How CLIP helps?

1. **Costly Dataset :** Methods that tries to mitigate the costly dataset problems are

- Self Supervised Learning — [3]

- Contrastive Methods — [4]
- Self Training Approaches — [5]
- Generative Modelling — [6]

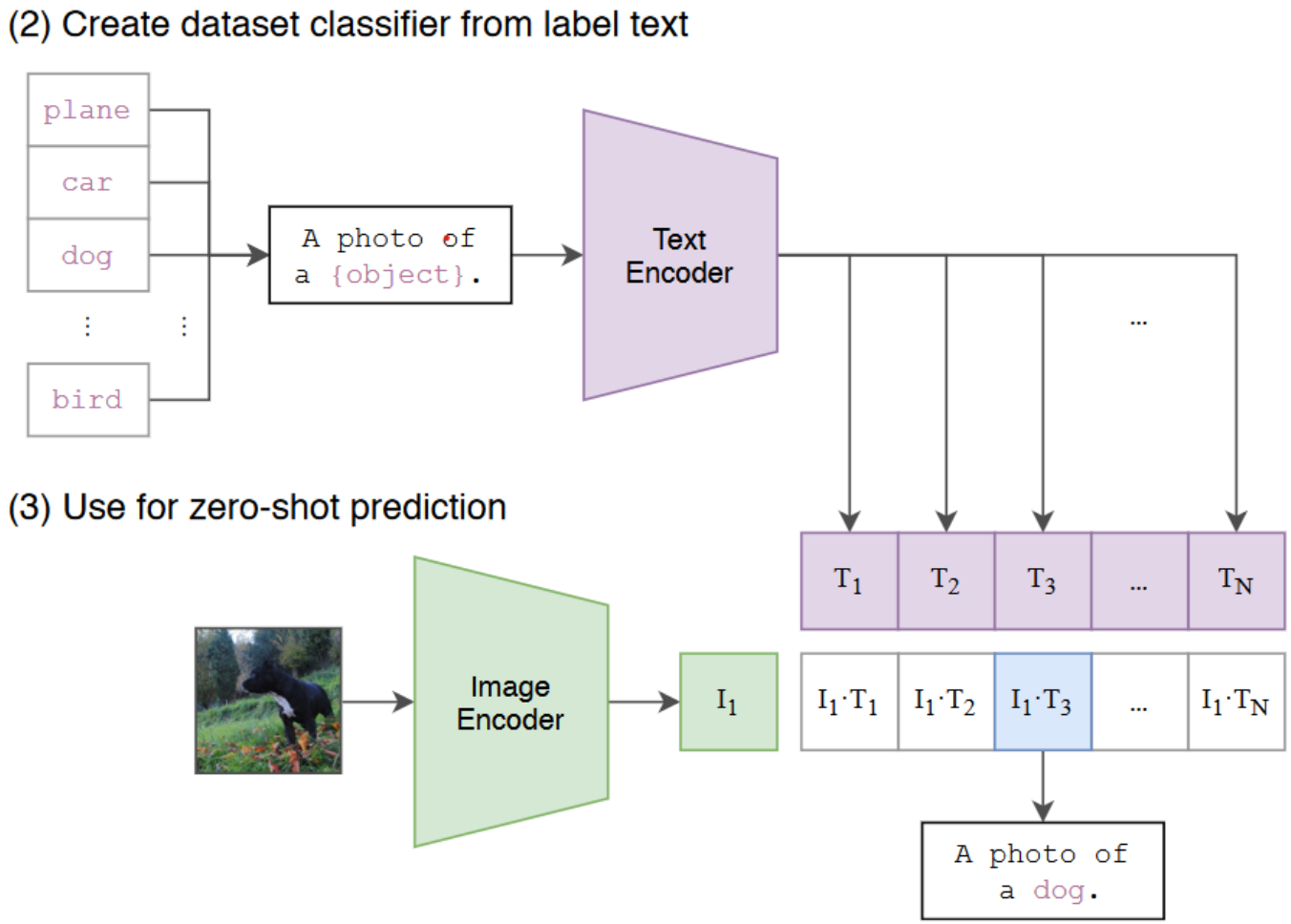
2. Narrow : Perform good only on training dataset domain and weakly generalise out of training examples. To Apply CLIP to a new task, we just need to encode the task description. Accuracy of these tasks as good as supervised counterparts and many times better.

3. Generalisation : Current DL methods benchmark performance is good compared to real performance. Reason being they are trained on benchmark dataset and evaluated on benchmark dataset.

CLIP model can be evaluated without being trained on the benchmark dataset.

How CLIP is used to do prediction Tasks

CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in the dataset.



Key Takeaways from CLIP

1. CLIP is highly efficient:

- Contrastive objective for connecting text with images.
- Adoption of the Vision Transformer. Gained 3x compute efficiency compared to ResNet model.

2. CLIP is generalisable

- The best CLIP model outperforms the best publicly available ImageNet model, the Noisy Student EfficientNet-L2[7], on 20 out of 26 different transfer datasets.

3. Limited performance on fine-grained and systematics tasks.

Applications of CLIP

CLIP encodes text and image in same embedding space. This gives us opportunity to work with intersection of image and text modalities.

Few Applications which could be build with CLIP are

- **Zero Shot Image classification:** Using Clip embedding out of the box to do zero shot image classification.
- **Fine Tuned Image Classification:** Adding a classification head and fine tune the head for specific fine-grained systematic image classification.
- **Semantic Image Retrieval :** Text to image and Reverse Image search both are possible with rich CLIP embeddings.
- **Content Moderation :** If you prompt CLIP correctly , we can filter out graphic or NSFW images out of the box.
- **Image Ranking :** It's not just factual representations that are encoded in CLIP's memory. It also knows about qualitative concepts as well
- **Image Captioning :** With the feature vectors from CLIP have been wired into GPT-2 to output an English description for a given image. Have a look at this repo for implementation details. [8]
- **Deciphering Blurred Images :** Inverse Problems Leveraging Pre-Trained Contrastive Representations, researchers have shown how CLIP can be used to interpret extremely distorted or corrupted images.

1. Image Search Using CLIP Embeddings

[2]:
```
from sentence_transformers import SentenceTransformer, util
from PIL import Image
import glob
import torch
import pickle
import zipfile
from IPython.display import display
from IPython.display import Image as IPImage
import os
from tqdm.autonotebook import tqdm

# Here we load the multilingual CLIP model. Note, this model can only encode text.
# If you need embeddings for images, you must load the 'clip-ViT-B-32' model
model = SentenceTransformer('clip-ViT-B-32-multilingual-v1')
```

Load The Text Encoder Using SentenceTransformer

```
# Lets compute the image embeddings.

#For embedding images, we need the non-multilingual CLIP model
img_model = SentenceTransformer('clip-ViT-B-32')

img_names = list(glob.glob(f'{data_path}*.jpg'))
print("Images:", len(img_names))
img_emb = img_model.encode([Image.open(filepath) for filepath in img_names], batch_size=128, convert_to_tensor=True, show_progress_bar=Tru
```

```
ftfy or spacy is not installed using BERT BasicTokenizer instead of ftfy.
Images: 1000
Batches: 100%|██████████| 8/8 [00:15<00:00,  1.98
s/it]
```

Load The Image Encoder Model Using SentenceTransformer

In [29]:

```
# Next, we define a search function.
def search(query, k=4):
    # First, we encode the query (which can either be an image or a text string)
    query_emb = model.encode([query], convert_to_tensor=True, show_progress_bar=False)

    # Then, we use the util.semantic_search function, which computes the cosine-similarity
    # between the query embedding and all image embeddings.
    # It then returns the top_k highest ranked images, which we output
    hits = util.semantic_search(query_emb, img_emb, top_k=k)[0]

    matched_images = []
    for hit in hits:
        matched_images.append(Image.open(img_names[hit['corpus_id']]))

    plot_images(matched_images, query)
    #print(matched_images)
```

In [48]:

```
search("white dogs playing with ball")
```

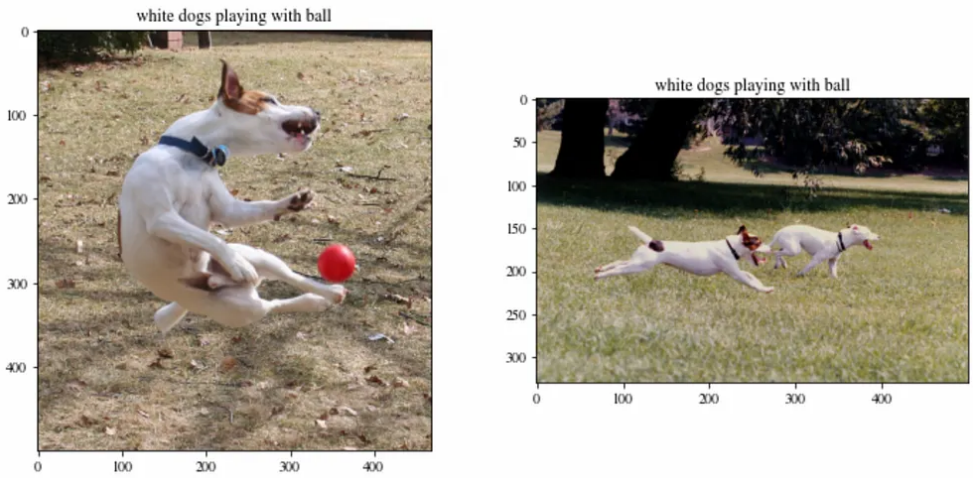
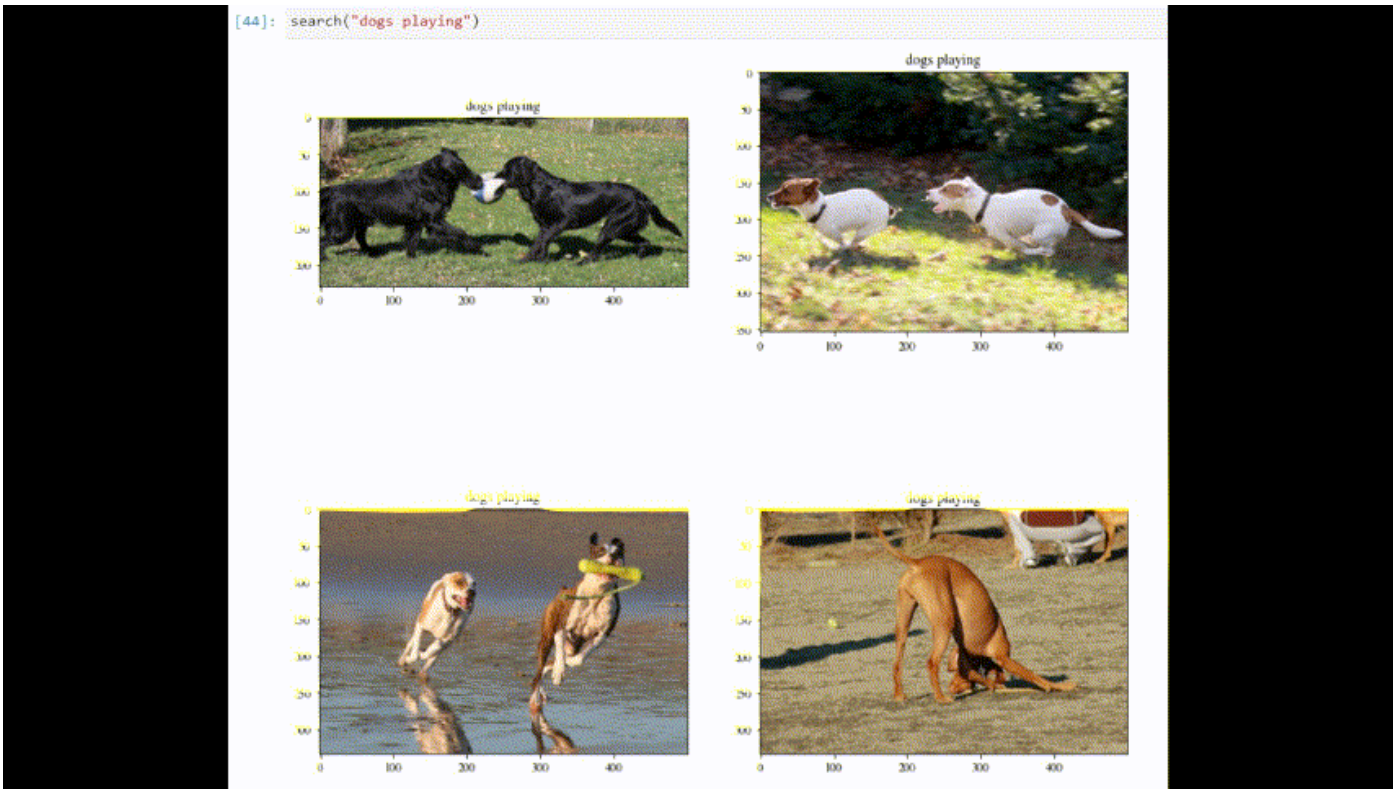


Image Search Using CLIP



Head Over to this github repo to find the complete example

applied_clip/Image_Search_multilingual.ipynb at main · akgeni/applied_clip

Applications of CLIP. Contribute to akgeni/applied_clip development by creating an account on GitHub.

github.com

Reverse Image Search Using CLIP Embeddings

We can also build Revers Image Search, which searches Images given its input is also an image. Google Lens is an example of reverse image search.

I am demoing a scalable way to build Reverse Image Search using Annoy Indexer. Annoy Indexer can be used as Nearest Neighbour Embedding Search. Read more about it here https://github.com/spotify/annoy

[7]:
```
img_model = SentenceTransformer('clip-ViT-B-32')
```

```
Downloading:   0%|          | 0.00/690 [00:00, ?B/s]
Downloading:   0%|          | 0.00/4.03k [00:00, ?B/s]
Downloading:   0%|          | 0.00/525k [00:00, ?B/s]
Downloading:   0%|          | 0.00/316 [00:00, ?B/s]
Downloading:   0%|          | 0.00/605M [00:00, ?B/s]
Downloading:   0%|          | 0.00/389 [00:00, ?B/s]
Downloading:   0%|          | 0.00/604 [00:00, ?B/s]
Downloading:   0%|          | 0.00/961k [00:00, ?B/s]
Downloading:   0%|          | 0.00/1.88k [00:00, ?B/s]
Downloading:   0%|          | 0.00/116 [00:00, ?B/s]
Downloading:   0%|          | 0.00/122 [00:00, ?B/s]
ftfy or spacy is not installed using BERT BasicTokenizer instead of ftfy.
```

[8]:
```
def get_image_embeddings_from_path(img_model, data_path):
  img_names = list(glob.glob(f'{data_path}*.[jpg][png][jpeg]'))
  print("Images:", len(img_names))
  img_emb = img_model.encode([Image.open(filepath) for filepath in img_names],
                             batch_size=128, convert_to_tensor=True, show_progress_bar=True)

  return {"img_names": img_names, "img_emb": img_emb, "id": range(1, len(img_names)+1)}


def get_image_embedding(img_model, image_path):

  img_emb = img_model.encode([Image.open(image_path)],
                             batch_size=1, convert_to_tensor=True, show_progress_bar=True)
  return img_emb


#get_image_embedding(img_model, data_path+"42637986_135a9786a6.jpg")
```

Load CLIPImage Encoder, Functions to read image data and convert to embeddings using CLIP

[39]:
```
def create_index(index_path_name, image_embed_dict, num_trees=30, verbose=True):

    #image_embed_dict = get_image_embeddings_from_path(img_model, data_path)

    annoy_lmdb = index_path_name+".lmdb"
    annoy_index = index_path_name+".annoy"
    embed_size = image_embed_dict['img_emb'][0].shape[0]

    if verbose:
        print("Vector size: {}".format(embed_size))

    env = lmdb.open(annoy_lmdb, map_size=int(1e9))

    if not os.path.exists(annoy_index) or not os.path.exists(annoy_lmdb):
        i = 0
        a = annoy.AnnoyIndex(embed_size)
        with env.begin(write=True) as txn:
            for index, (i, vec, img_name) in enumerate(zip(image_embed_dict['id'],
                                                           image_embed_dict['img_emb'],
                                                           image_embed_dict['img_names'])):
                a.add_item(i, vec)
                id = 'ID_%d' % i
                word = 'I_' + img_name
                txn.put(id.encode(), word.encode())
                txn.put(word.encode(), id.encode())
                i += 1
                if verbose:
                    if i % 1000 == 0:
                        print(i, '...')
        if verbose:
            print("Starting to build")
        a.build(num_trees)
        if verbose:
            print("Finished building")
        a.save(annoy_index)
        if verbose:
            print("Annoy index saved to: {}".format(annoy_index))
            print("lmdb map saved to: {}".format(annoy_lmdb))
    else:
        print("Annoy index and lmdb map already in path")
```

[40]:
```
create_index("flickr_sample", image_embed_dict)
```

```
Vector size: 512
```

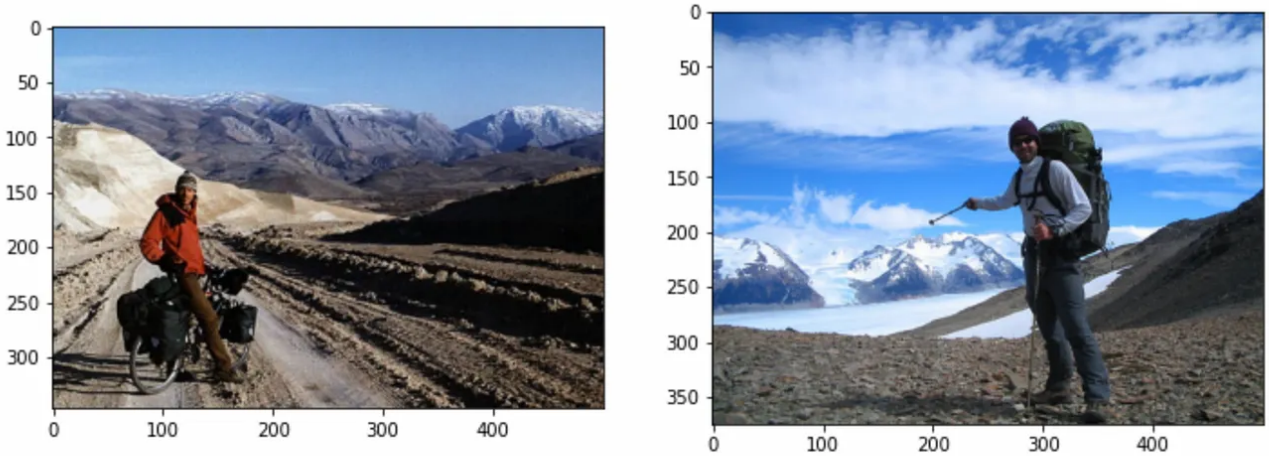
Index the Image Embeddings using Annoy Indexing.

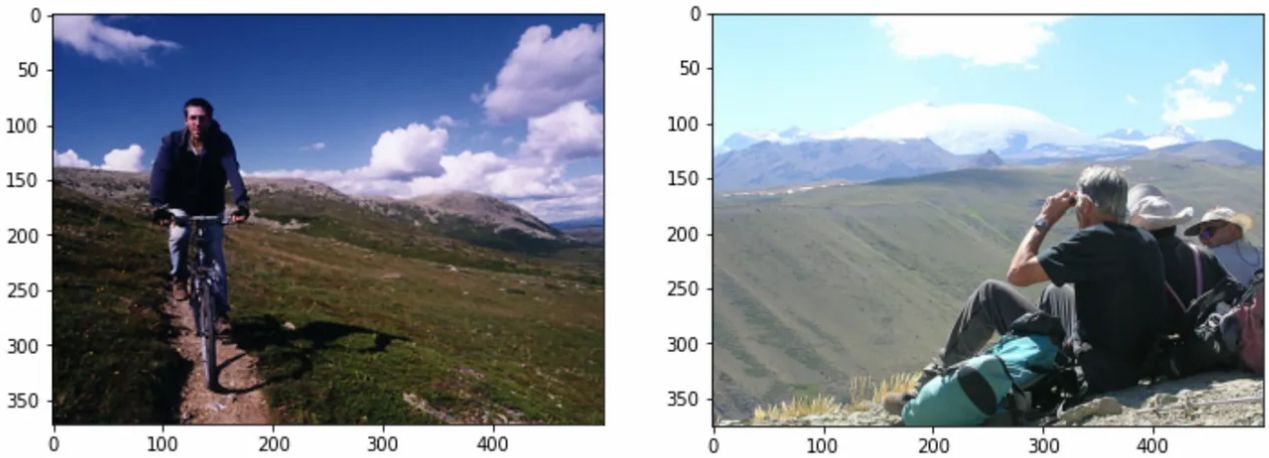
In [113…

```
query_image = "drive/MyDrive/Temp/Flickr_1000/95734038_2ab5783da7.jpg"

#query_image = random.choice(image_embed_dict['img_names'])

similar_images = search_images_by_image(img_model,
                                        query_image,
                                        num_results=10,
                                        verbose=False
                                        )

plot_images([Image.open(img) for img in similar_images], Image.open(query_image))
```

```
Batches:   0%|          | 0/1 [00:00, ?it/s]
```

Given the Query Image above





Following Results have been obtained.

I had a small dataset of around ~1K Flicr Images. We can still see how Trekking, and Scene and Bicycle concepts are captures.

For full code Please head over to this: https://github.com/akgeni/applied_clip/blob/main/scalable_reverse_image_search/scalable_reverse_image_search_clip.ipynb

CLIP applications are not limited to these examples shown above. A more inclusive list of application we have seen above. Feel free to experiment with them.

I know the article has become little longer, but I wont let you go without informing about limitations of CLIP. Here are they

CLIP Limitations

- Fails to count number of objects in image.
- Fails on fine-grained image classification like celebrity identification, car model identification, flower species etc.
- Weakly performs on handwritten image MNIST. 88% Zero Shot accuracy.

For Any doubts, Clarification or correction Feel free to connect me on LinkedIN: https://www.linkedin.com/in/anshu19/

Resources:

Simple Implementation of OpenAI CLIP model: A Tutorial

A tutorial on simple implementation of CLIP model in PyTorch.

towardsdatascience.com

GitHub - moein-shariatnia/OpenAI-CLIP: Simple implementation of OpenAI CLIP model in PyTorch.

I am happy to find out that this code has been used and cited in the following papers: Domino: Discovering Systematic...

github.com

What is OpenAI's CLIP and how to use it?

CLIP is a gigantic leap forward, bringing many of the recent developments from the realm of natural language processing...

blog.roboflow.com

Thank You for reading this article. I hope you would have found it useful.

OpenAI | Deep Learning | Computer Vision | Naturallanguageprocessing | Computer Science

Follow

Written by Anshu Kumar

106 Followers

Data Scientist, Author. Building Semantic Search and Recommendation Engine.

More from Anshu Kumar

Anshu Kumar

LLaMA: Concepts Explained (Summary)

Pre-normalization, SwiGLU, Rotary Embeddings

4 min read · Mar 1

26 3

Anshu Kumar

Zeroshot Classification

Machine learning with no Data and without Training

4 min read · Sep 16, 2021

14

Anshu Kumar

Non Functional Requirement For ML Systems

Majority of Machine Learning(ML) systems built today are focusing only on some of the functional requirements. As the ML systems mature and...

3 min read · Jul 7, 2021

4

	Connoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prec Posi
ample ir]	**Aonnoisseurs** of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prec Neg
ample similar]	Connoisseurs of Chinese **footage** will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prec Neg

Anshu Kumar

NLP Augmentation Hands-On

Part-2 TextAttack(HuggingFace) library for Data Augmentation

3 min read · Apr 22, 2021


10

See all from Anshu Kumar

Recommended from Medium

JeremyK

Unlocking OpenAI CLIP. Part 1: intro to zero-shot classification

You will find plenty of articles about OpenAI CLIP on the Internet. So what's new here will you ask?

7 min read · Jun 15

5

Federico Bianchi in Towards Data Science

Teaching CLIP Some Fashion

Training FashionCLIP, a domain-specific CLIP model for Fashion

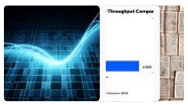
10 min read · Mar 7

220 1

Lists

Now in AI: Handpicked by Better Programming

266 stories · 153 saves



Natural Language Processing

635 stories · 245 saves

AI Regulation

6 stories · 128 saves

Practical Guides to Machine Learning

10 stories · 477 saves

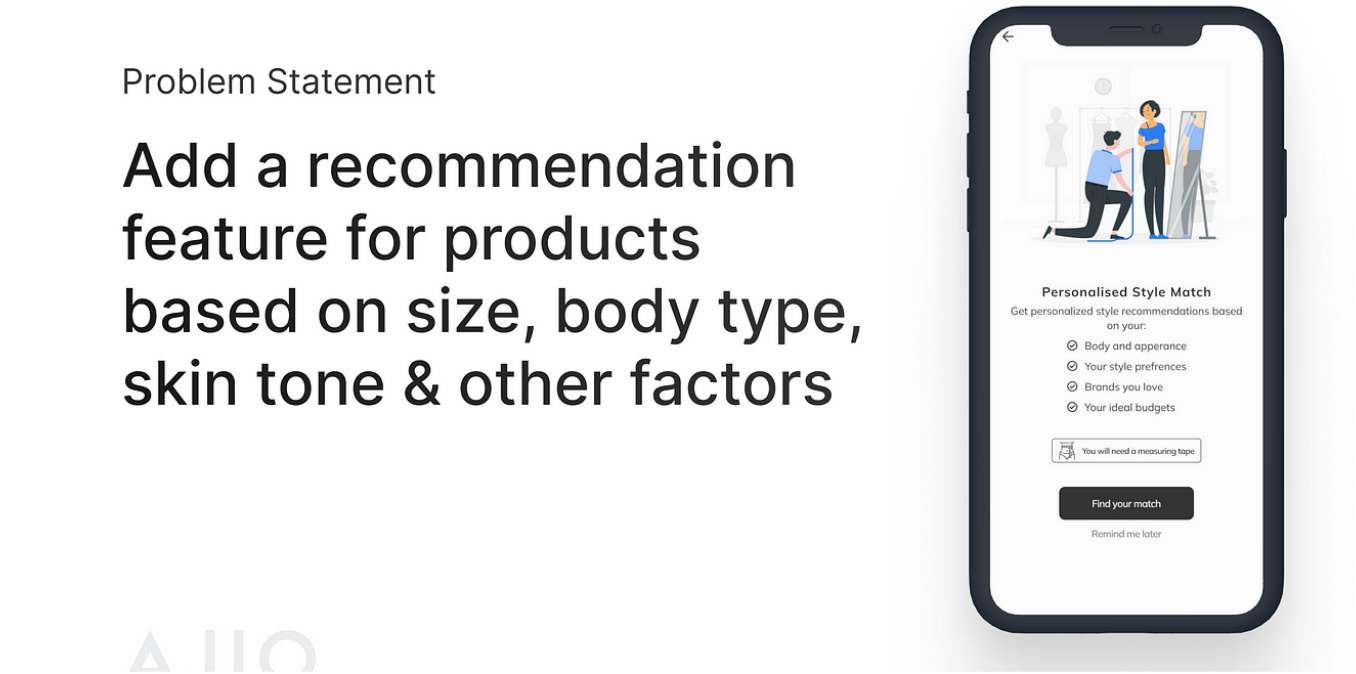
Shashank Vats in *AI monks.io*

A Guide to Fine-Tuning CLIP Models with Custom Data

Artificial intelligence and machine learning have come a long way in recent years, with advances in the field allowing researchers and...

5 min read · Jun 1

70 2



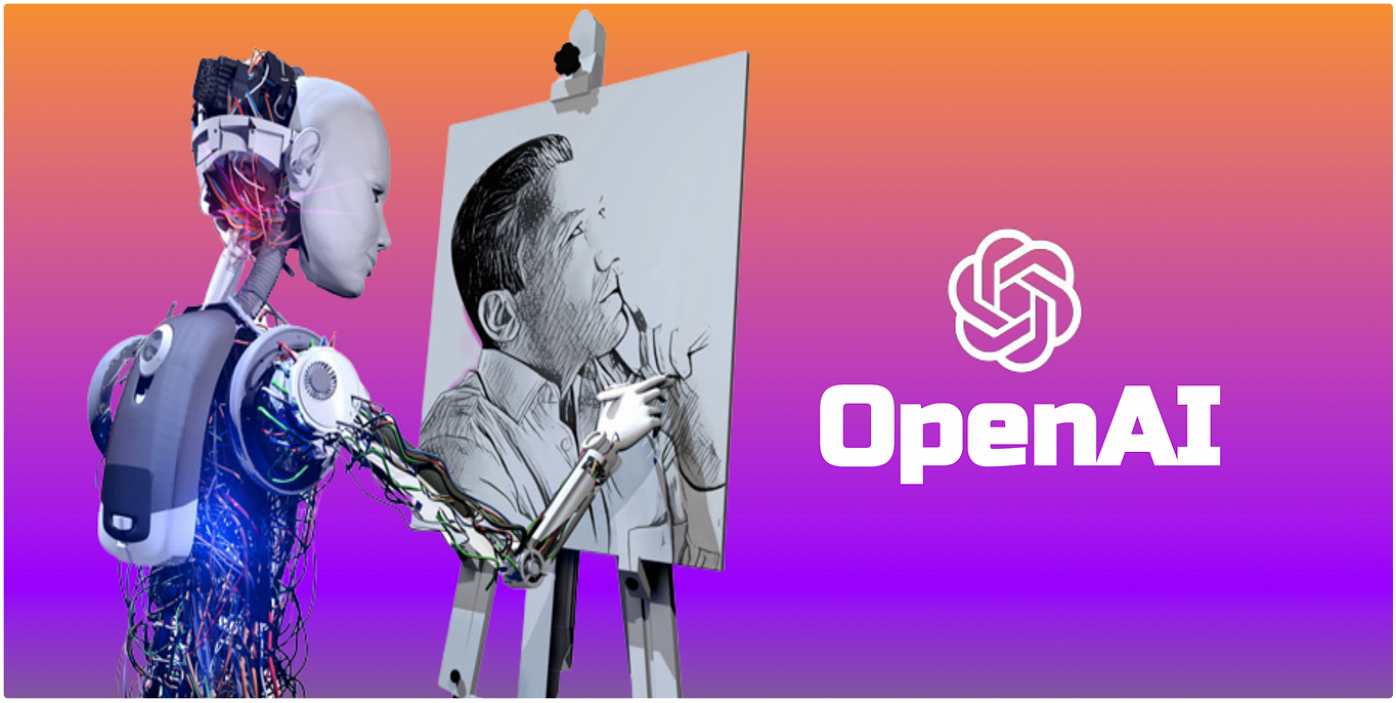
Aditi Mishra

Helping users make confident purchases on Ajio through a body-inclusive recommendation system

As the user base and brand partners began to grow, there was a growing need to show the relevant content to the right eye.

8 min read · Jun 11

201 2



 Chikku George in Globant

Image Generation using OpenAI API with NodeJs

As part of this article, we will use the OpenAI API to generate an image or piece of art from a natural language description. In response...

5 min read · Apr 10

25

Mauo Salem

The Chat GPT Hype Is Over—Now Watch How Google Will Kill Chat GPT

Introduction

3 min read · Sep 10

4

See more recommendations