```
!pip install pymilvus pillow sentence-transformers
```

Show hidden output

```
from sentence_transformers import SentenceTransformer
from PIL import Image
import requests
from io import BytesIO

# Load CLIP image embedding model
model = SentenceTransformer("sentence-transformers/clip-ViT-L-14")
```

```
import requests
from PIL import Image
from io import BytesIO
from concurrent.futures import ThreadPoolExecutor, as_completed

base_folder_url = "https://github.com/skarifahmed/RingFIR/raw/main/data/RingFIR/"


def process_single_folder(folder_num, base_folder_url, model):
    embeddings = []
    image_urls = []

    folder_id = str(folder_num).zfill(3)
    base_url = f"{base_folder_url}{folder_id}/{folder_id}_"
    i = 1

    while True:
        img_number = str(i).zfill(3)
        img_url = f"{base_url}{img_number}.png"
        response = requests.get(img_url)

        if response.status_code != 200:
            break

        image_urls.append(img_url)
        image = Image.open(BytesIO(response.content)).convert("RGB")
        emb = model.encode(image)
        embeddings.append(emb)
        i += 1

    print(f"Folder {folder_id}: {len(embeddings)} images processed")
    return embeddings, image_urls


def process_image_folders_parallel(base_folder_url, model, start_folder=1, end_folder=46, max_workers=5):
    all_embeddings = []
    all_image_urls = []

    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        futures = [
            executor.submit(process_single_folder, folder_num, base_folder_url, model)
            for folder_num in range(start_folder, end_folder + 1)
        ]

        for future in as_completed(futures):
            emb, urls = future.result()
            all_embeddings.extend(emb)
            all_image_urls.extend(urls)

    print("All folders processed.")
    print(len(all_embeddings), len(all_image_urls))
    return all_embeddings, all_image_urls



all_embeddings, all_image_urls = process_image_folders_parallel(base_folder_url, model, 1, 46, 5)
```

```
Folder 001: 22 images processed
Folder 005: 24 images processed
Folder 006: 16 images processed
Folder 004: 43 images processed
Folder 002: 45 images processed
Folder 003: 58 images processed
```

```
Folder 007: 49 images processed
Folder 009: 36 images processed
Folder 010: 36 images processed
Folder 008: 53 images processed
Folder 012: 25 images processed
Folder 015: 22 images processed
Folder 013: 30 images processed
Folder 014: 32 images processed
Folder 011: 55 images processed
Folder 016: 34 images processed
Folder 018: 30 images processed
Folder 020: 38 images processed
Folder 017: 58 images processed
Folder 019: 60 images processed
Folder 023: 43 images processed
Folder 022: 63 images processed
Folder 021: 80 images processed
Folder 025: 79 images processed
Folder 028: 57 images processed
Folder 024: 124 images processed
Folder 026: 100 images processed
Folder 031: 41 images processed
Folder 029: 80 images processed
Folder 027: 116 images processed
Folder 035: 21 images processed
Folder 032: 69 images processed
Folder 033: 38 images processed
Folder 034: 31 images processed
Folder 030: 103 images processed
Folder 037: 22 images processed
Folder 036: 35 images processed
Folder 041: 41 images processed
Folder 039: 66 images processed
Folder 038: 80 images processed
Folder 042: 58 images processed
Folder 043: 38 images processed
Folder 040: 106 images processed
Folder 044: 70 images processed
Folder 045: 73 images processed
Folder 046: 97 images processed
All folders processed.
2497 2497
```

```
from google.colab import drive
drive.mount('/content/drive')

import json

config_path = '/content/drive/MyDrive/creds/milvus_cred.json'

with open(config_path, 'r') as f:
    milvus_config = json.load(f)

alias = milvus_config.get("alias", "default")
uri = milvus_config["uri"]
token = milvus_config["token"]
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from pymilvus import connections, Collection
from pymilvus import connections

connections.connect(
    alias=alias,
    uri=uri,
    token=token
)
```

```
from pymilvus import FieldSchema, CollectionSchema, DataType, Collection
```

```
fields = [
    FieldSchema(name="id", dtype=DataType.INT64, is_primary=True, auto_id=True),
    FieldSchema(name="embedding", dtype=DataType.FLOAT_VECTOR, dim=768),
    FieldSchema(name="image_url", dtype=DataType.VARCHAR, max_length=500)
]
```

```
schema = CollectionSchema(fields, description="RingFIR embeddings")
collection = Collection("ringfir", schema)  # This creates the collection if it doesn't exist

data = [
    all_embeddings, all_image_urls
]


insert_result = collection.insert(data)


collection.flush()

from pymilvus import Index

index_params = {
    "index_type": "IVF_FLAT",  # You can also use HNSW or ANNOY
    "metric_type": "L2",       # "IP" for cosine similarity
    "params": {"nlist": 128}   # nlist depends on your dataset size
}

collection.create_index("embedding", index_params=index_params)




collection.load()

print(f"Inserted {len(all_embeddings)} rows into collection.")
```

```
Inserted 2497 rows into collection.
```

```
query_url = "https://raw.githubusercontent.com/skarifahmed/RingFIR/main/data/RingFIR/042/042_003.png"  # Query image
response = requests.get(query_url)
image = Image.open(BytesIO(response.content)).convert("RGB")

image_path = "/content/sample_data/earing_search.png"
image = Image.open(image_path).convert("RGB")

emb = model.encode(image)

results = collection.search(
    data=[emb],
    anns_field="embedding",
    param={"metric_type": "L2", "params": {"nprobe": 10}},
    limit=3,
    output_fields=["image_url"]
)

for hit in results[0]:
    print(f"Score: {hit.score}, Image URL: {hit.entity.get('image_url')}")
```

```
Score: 91.69725036621094, Image URL: https://github.com/skarifahmed/RingFIR/raw/main/data/RingFIR/014/014_028.png
Score: 91.69725036621094, Image URL: https://github.com/skarifahmed/RingFIR/raw/main/data/RingFIR/014/014_028.png
Score: 99.69913482666016, Image URL: https://github.com/skarifahmed/RingFIR/raw/main/data/RingFIR/014/014_011.png
```