**Introduction to Image Embeddings**

[[Abdulkader Helwan](https://abdulkaderhelwan.medium.com/?source=post_page-----55b8247d13f2--------------------------------)](https://abdulkaderhelwan.medium.com/?source=post_page-----55b8247d13f2--------------------------------)

[Abdulkader Helwan](https://abdulkaderhelwan.medium.com/?source=post_page-----55b8247d13f2--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Fd1f2a9be01f6&operation=register&redirect=https%3A%2F%2Fabdulkaderhelwan.medium.com%2Fintroduction-to-image-embeddings-55b8247d13f2&user=Abdulkader+Helwan&userId=d1f2a9be01f6&source=post_page-d1f2a9be01f6----55b8247d13f2---------------------post_header-----------)

5 min read

·

Oct 1

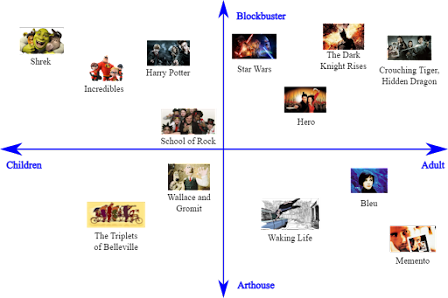
7

This [blog post](https://www.ai-contentlab.com/2023/09/what-are-image-embeddings.html) discusses image embeddings and its implementation in Python. I hope you find it useful and informative.

This post was published by [AI-ContentLab](https://www.ai-contentlab.com/2023/09/what-are-image-embeddings.html)

**Introduction**

Image embeddings are numerical representations of images that capture their semantic meaning and visual features. They are useful for many applications, such as image search, image classification, image retrieval, and image similarity. In this blog post, we will learn what image embeddings are, why they are important, and how to generate them using Python and some open-source libraries.



[Source](https://www.ai-contentlab.com/2023/09/what-are-image-embeddings.html)

**What are image embeddings?**

An image embedding is a vector of numbers that represents an image in a high-dimensional space. For example, an image of a cat can be embedded as a vector of 384 numbers, such as [0.12, -0.34, …, 0.05]. Each number in the vector corresponds to a feature or an attribute of the image, such as color, shape, texture, etc. The vector captures the essence of the image and allows us to compare it with other images using mathematical operations.

Image embeddings are also called image features or image descriptors. They are derived from image models that have been trained on large datasets of images to learn how to extract meaningful information from them. There are different types of image models, such as convolutional neural networks (CNNs), autoencoders, generative adversarial networks (GANs), etc. Each type of model has its own advantages and disadvantages, depending on the task and the data.

**Why are image embeddings important?**

Image embeddings are important because they enable us to perform various tasks that require understanding the content and the context of images. Some examples of these tasks are:

- Image search: Given a query image or a text description, we can find similar images in a database or on the web using image embeddings. For example, we can search for images of flowers that look like roses or images of cars that are red.

- Image classification: Given an image, we can assign it to one or more categories based on its content using image embeddings. For example, we can classify an image as animal, plant, or mineral or as cat, dog, or bird.

- Image retrieval: Given a set of images, we can rank them according to their relevance or similarity to a query using image embeddings. For example, we can retrieve images of landscapes that are most similar to a given image or images of products that are most relevant to a user’s preference.

- Image similarity: Given two images, we can measure how similar or different they are using image embeddings. For example, we can compare two images of faces and determine if they belong to the same person or not.

These tasks are useful for many applications and domains, such as e-commerce, social media, education, entertainment, security, etc. Image embeddings allow us to leverage the vast amount of visual information available on the web and in our devices and make sense of it.

**How to generate image embeddings using Python?**

There are many ways to generate image embeddings using Python and some open-source libraries. In this section, we will show you how to use two popular libraries: scikit-learn and Hugging Face transformers.

* Scikit-learn

Scikit-learn is a library for machine learning in Python that provides various tools for data analysis and modeling. One of the tools that scikit-learn offers is principal component analysis (PCA), which is a technique for dimensionality reduction. PCA can be used to generate linear image embeddings by projecting high-dimensional images onto a lower-dimensional subspace that preserves most of the variance in the data.

To use PCA to generate image embeddings using scikit-learn, we need to follow these steps:

1. Import the necessary modules:

import numpy as np

import skimage.io as io

from sklearn.decomposition import PCA2. Load the images as numpy arrays:# Load an example imageimg = io.imread('cat.jpg')# Convert the image to grayscaleimg = np.mean(img, axis=2)# Flatten the image into a 1D arrayimg = img.reshape(-1)# Repeat the same process for other images# Stack all the images into a 2D arrayX = np.stack([img1, img2, img3, ...])3. Create and fit a PCA model:# Create a PCA model with 384 componentspca = PCA(n\_components=384)# Fit the model on the datapca.fit(X)4. Transform the data using the model:# Transform the data into 384-dimensional vectorsX\_embedded = pca.transform(X)5. Use the embedded vectors for any task:# Use X\_embedded for any task

* Hugging Face transformers

Hugging Face transformers is a library for natural language processing (NLP) in Python that provides various models and pipelines for text analysis and generation. One of the models that Hugging Face transformers offers is CLIP, which is a vision-language model that can generate image embeddings from any image using a pretrained convolutional neural network (CNN) from OpenAI. CLIP can be used to generate deep image embeddings that are robust and versatile for many tasks.

To use CLIP to generate image embeddings using Hugging Face transformers, we need to follow these steps:

1. Install the library:

pip install transformers

2. Import the necessary modules:

import torch

from PIL import Imagefrom transformers import CLIPProcessor, CLIPModel

3. Load the model and the processor:

# Load the model

model = CLIPModel.from\_pretrained("openai/clip-vit-base-patch32")# Load the processorprocessor = CLIPProcessor.from\_pretrained("openai/clip-vit-base-patch32")

4. Load and preprocess the images:

# Load an example image

img = Image.open("cat.jpg")# Preprocess the image using the processorinputs = processor(images=img, return\_tensors="pt", padding=True)

5. Generate the embeddings using the model:

# Geneprate the embeddings using the model

outputs = model(\*\*inputs)# Extract the image embeddings from the outputsimg\_embeds = outputs.image\_embeds

6. Use the embedded vectors for any task:

# Use img\_embeds for any task

**Conclusion**

In this blog post, we learned what image embeddings are, why they are important, and how to generate them using Python and some open-source libraries. We showed how to use scikit-learn and Hugging Face transformers to generate linear and deep image embeddings, respectively. We hope that this blog post was educational, easy to understand, and useful for you. If you want to learn more about image embeddings and their applications, you can check out these resources:

- [Getting Started With Embeddings]: A tutorial by Hugging Face on how to use their Inference API to embed images and text.  
- [Image Embeddings: Image similarity and building embeddings with modern computer vision]: A blog post by Romain Beaumont on how to build a visual search engine using image embeddings.  
- [Image Embedding guide for Python]: A guide by Google Developers on how to use MediaPipe to embed images using Python.  
- [imgbeddings]: A Python package by Minimaxir to generate image embeddings using OpenAI’s CLIP model via Hugging Face transformers.  
- [Hyperbolic Image Embeddings]: A research paper by Valentin Khrulkov et al. on how to use hyperbolic geometry to embed images.