Multimodal Large Language Models

Introduction

- When you think about large language models (LLMs), multimodality might not be the first thing that comes to mind. After all, they are language models! But we can quickly see that models can be much more useful if they're able to handle types of data other than text.
- A model that is able to handle text and images (each of which is called a modality) is said to be multimodal, as we can see in Figure 9-1.

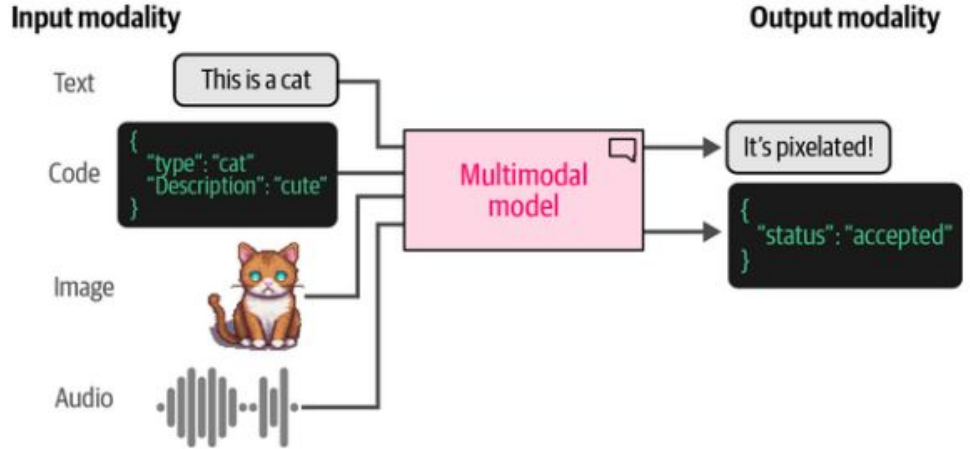

Figure 9-1. Models that are able to deal with different types (or modalities) of data, such as images, audio, video, or sensors, are said to be multimodal. It's possible for a model to accept a modality as input yet not be able to generate in that modality.

Transformers for Vision

Researchers have been looking at a way to generalize some of the Transformer's success to the field of computer vision. The method they came up with is called the Vision Transformer (ViT), which has been shown to do tremendously well on image recognition tasks compared to the previously default convolutional neural networks (CNNs). Like the original Transformer, ViT is used to transform unstructured data, an image, into representations that can be used for a variety of tasks, like classification, as illustrated in Figure 9-2.

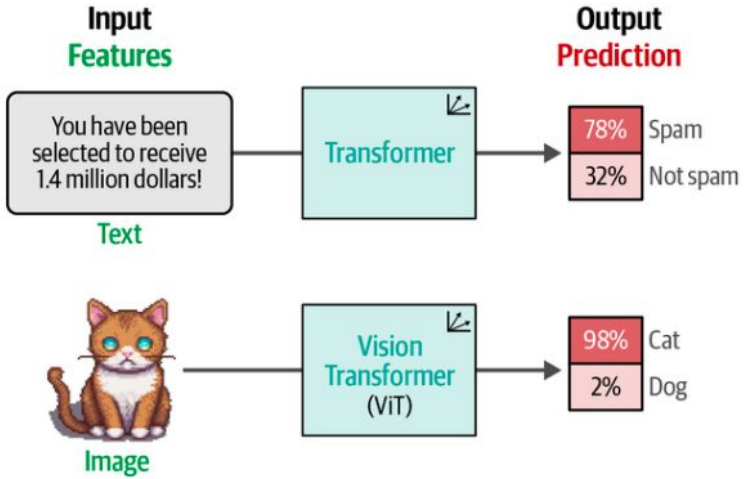

Figure 9-2. Both the original Transformer as well as the Vision Transformer take unstructured data, convert it to numerical representations, and finally use that for tasks like classification.

Converting Image to Words

Since an image does not consist of words this tokenization process cannot be used for visual data. Instead, the authors of ViT came up with a method for tokenizing images into “words,” which allowed them to use the original encoder structure. Imagine that you have an image of a cat.

This image is represented by a number of pixels, let’s say 512 × 512 pixels. Each individual pixel does not convey much information but when you combine patches of pixels, you slowly start to see more information.

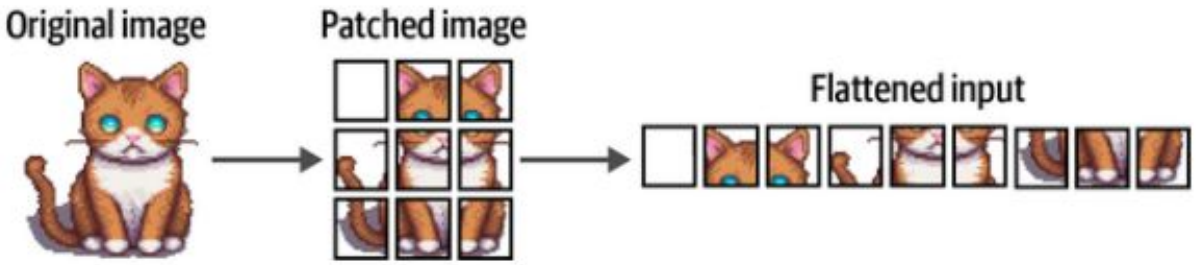

Figure 9-4. The “tokenization” process for image input. It converts an image into patches of subimages.

ViT uses a principle much like that. Instead of splitting up text into tokens, it converts the original image into patches of images. In other words, it cuts the image into a number of pieces horizontally and vertically as illustrated in Figure 9-4.

Converting Image to Words

that the moment the embeddings are passed to the encoder, they are treated as if they were textual tokens. From that point forward, there is no difference in how a text or image trains.

the ViT is often used to make all kinds of language models multimodal. One of the most straightforward ways to use it is during the training of embedding models.

The main algorithm behind ViT. After patching the images and linearly projecting them, the patch embeddings are passed to the encoder and treated as if they were textual tokens.

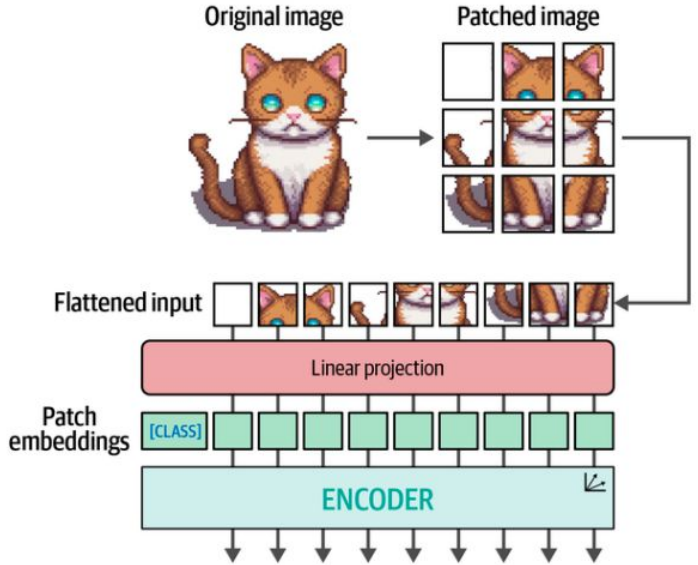

Figure 9-5. The main algorithm behind ViT. After patching the images and linearly projecting them, the patch embeddings are passed to the encoder and treated as if they were textual tokens.

Multimodal Embeddings

we used embedding models to capture the semantic content of textual representations, such as papers and documents. We saw that we could use these embeddings or numerical representations to find similar documents, apply classification tasks, and even perform topic modeling.

we will look at embedding models that can capture both textual as well as visual representations. We illustrate this in Figure 9-6.

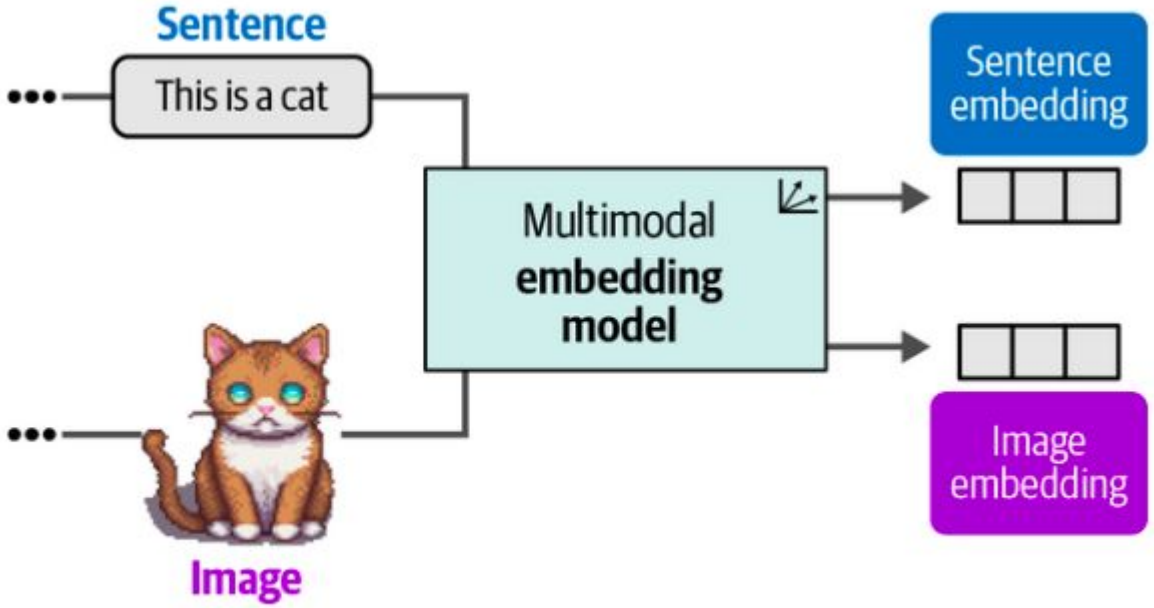

Figure 9-6. Multimodal embedding models can create embeddings for multiple modalities in the same vector space.

Advantages of multimodal embeddings

- An advantage is that this allows for comparing multimodal representations since the resulting embeddings lie in the same vector space (Figure 9-7).
- Despite having coming from different modalities, embeddings with similar meaning will be close to each other in vector space.

- we can find images based on input text and vice-versa

- There are a number of multimodal embedding models, but the most well-known and currently most-used model is Contrastive Language-Image Pre-training (**CLIP**).

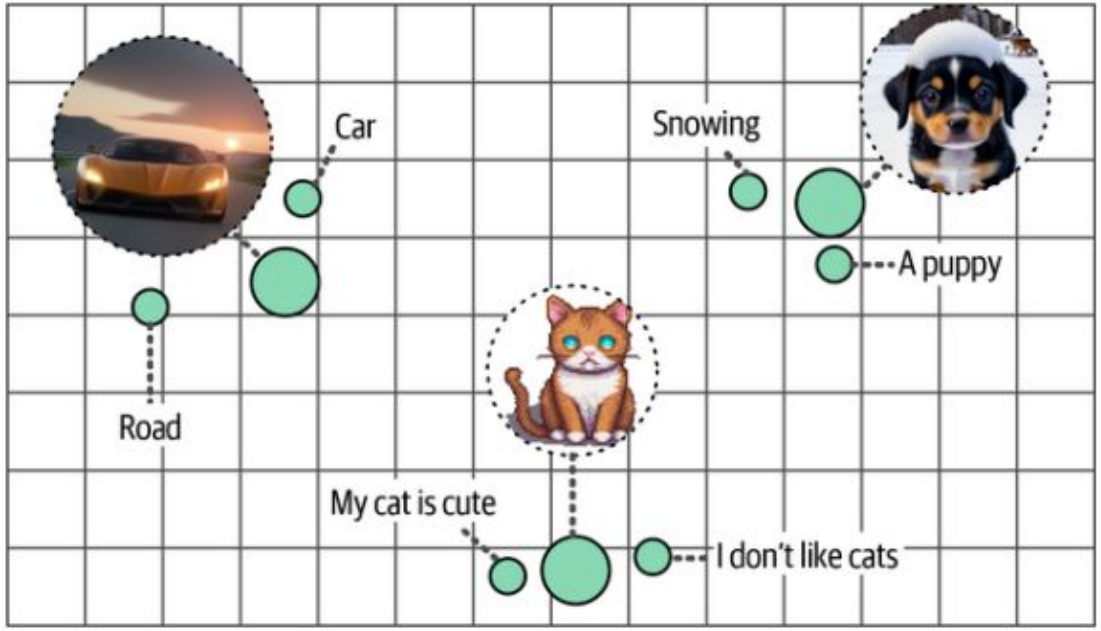

Figure 9-7. Despite having coming from different modalities, embeddings with similar meaning will be close to each other in vector space.

CLIP: Connecting Text and Images

- CLIP is an embedding model that can compute embeddings of both images and texts. The resulting embeddings lie in the same vector space, which means that the embeddings of images can be compared with the embeddings of text.
- This comparison capability makes CLIP, and similar models, usable for tasks such as:
  - **Zero-shot classification** - We can compare the embedding of an image with that of the description of its possible classes to find which class is most similar.
  - **Clustering -** We can compare the embedding of an image with that of the description of its possible classes to find which class is most similar.
  - **Search -** Across billions of texts or images, we can quickly find what relates to an input text or image.
  - **Generation -** Use multimodal embeddings to drive the generation of images (e.g.,stable diffusion)

How CLIP Creates Multimodal Embeddings

The procedure of CLIP is actually quite straightforward. Imagine that you have a dataset with millions of images alongside captions as we illustrate in Figure 9-8.

Figure 9-8. The type of data that is needed to train a multimodal embedding model.

This dataset can be used to create two representations for each pair, the image and its caption. To do so, CLIP uses a text encoder to embed text and an image encoder to embed images. As is shown in Figure 9-9, the result is an embedding for both the image and its corresponding caption.

How CLIP Creates Multimodal Embeddings - step1

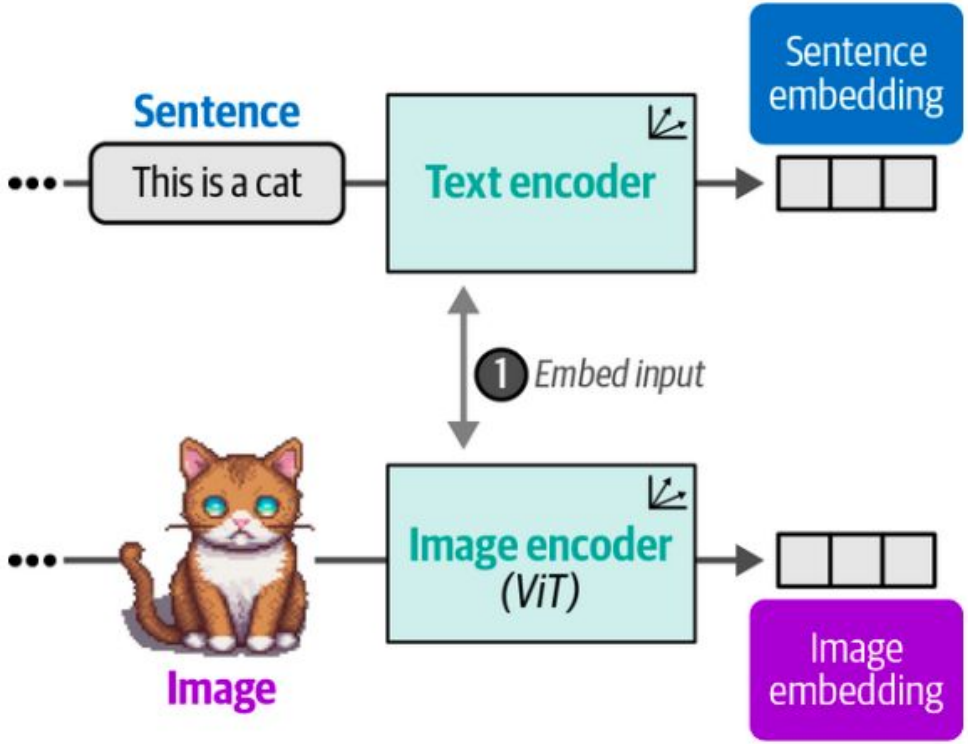

Figure 9-9. In the first step of training CLIP, both images and text are embedded using an image and text encoder, respectively.

How CLIP Creates Multimodal Embeddings – step 2

The pair of embeddings that are generated are compared through cosine similarity. As we saw in Chapter 4, cosine similarity is the cosine of the angle between vectors, which is calculated through the dot product of the embeddings and divided by the product of their lengths.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

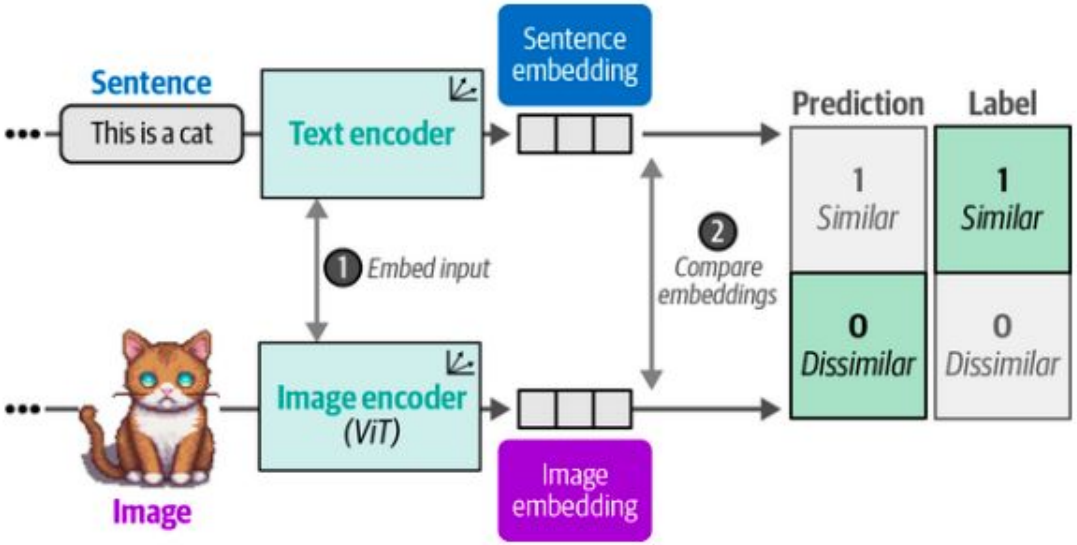

Figure 9-10. In the second step of training CLIP, the similarity between the sentence and image embedding is calculated using cosine similarity.

How CLIP Creates Multimodal Embeddings – step 3

After calculating their similarity, the model is updated and the process starts again with new batches of data and updated representations (Figure 9-11). This method is called contrastive learning, and we will go in depth into its

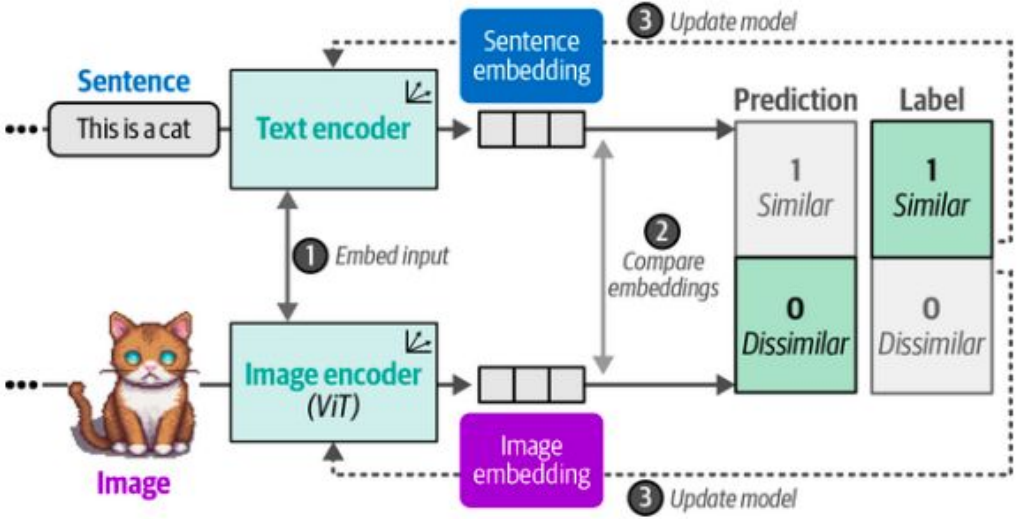

Figure 9-11. In the third step of training CLIP, the text and image encoders are updated to match what the intended similarity should be. This updates the embeddings such that they are closer in vector space if the inputs are similar.

Open-CLIP