Member-only story

Image Classification with Vision Transformer

How to classify images with the help of Transformer-based model

Ruben Winastwan · Follow
Published in Towards Data Science
13 min read · Apr 13

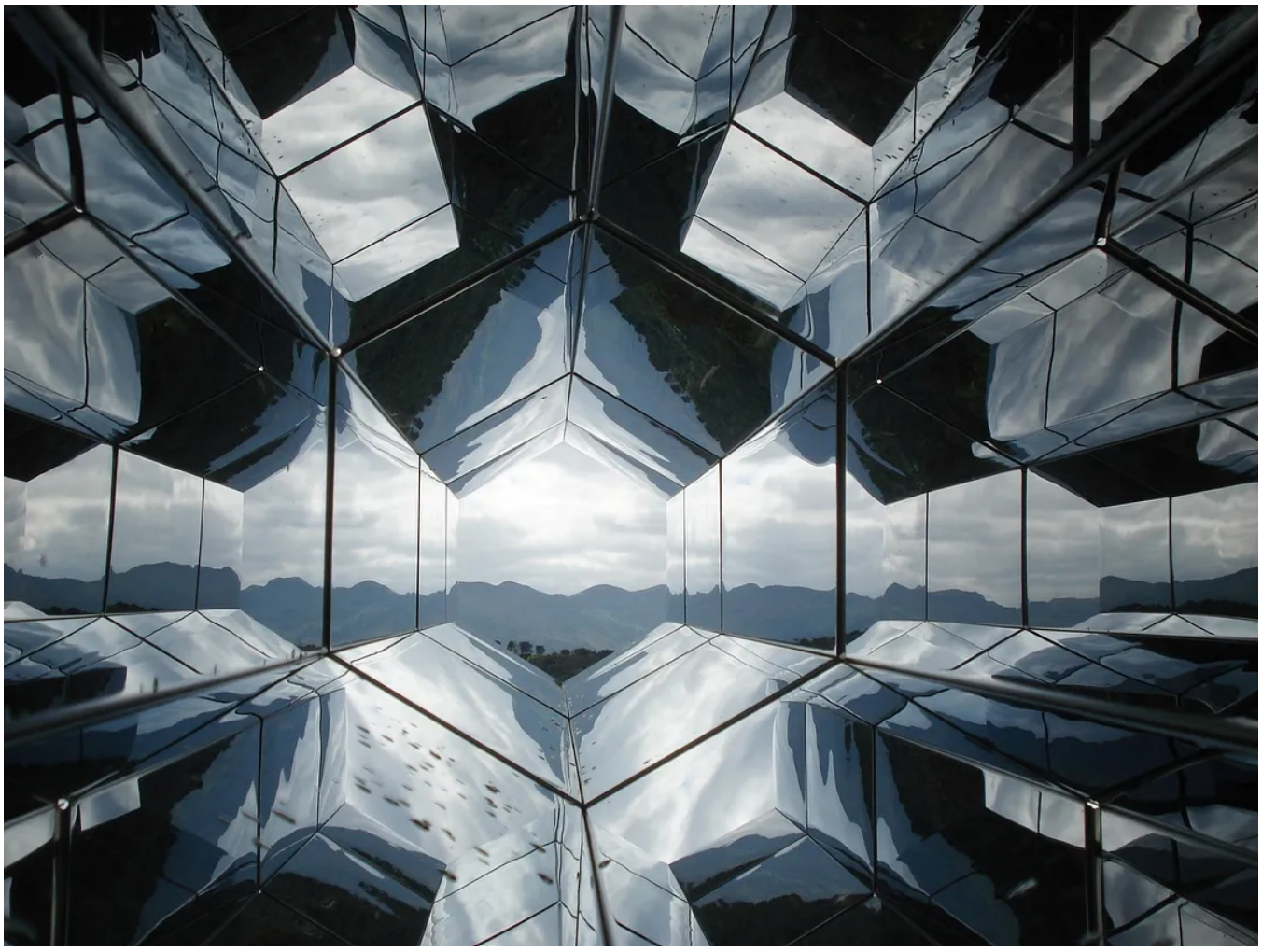


Photo by drmakete lab on Unsplash

Since its introduction in 2017, Transformer has been widely recognized as a powerful encoder-decoder model to solve pretty much any language modeling task.

BERT, RoBERTa, and XLM-RoBERTa are a few examples of state-of-the-art models in language processing that use a stack of Transformer encoders as the backbone in their architecture. ChatGPT and the GPT family also use the decoder part of Transformer to generate texts. It's safe to say that almost any state-of-the-art model in natural language processing incorporate Transformer in its architecture.

Transformer performance is so good that it seems wasteful not to use it for tasks beyond natural language processing, like computer vision for example. However, the big question is: can we actually use it for computer vision tasks?

It turns out that Transformer also has a good potential to be applied to computer vision tasks. In 2020, Google Brain team introduced a Transformer-based model that can be used to solve an image classification task called Vision Transformer (ViT). Its performance is very competitive in comparison with conventional CNNs on several image classification benchmarks.

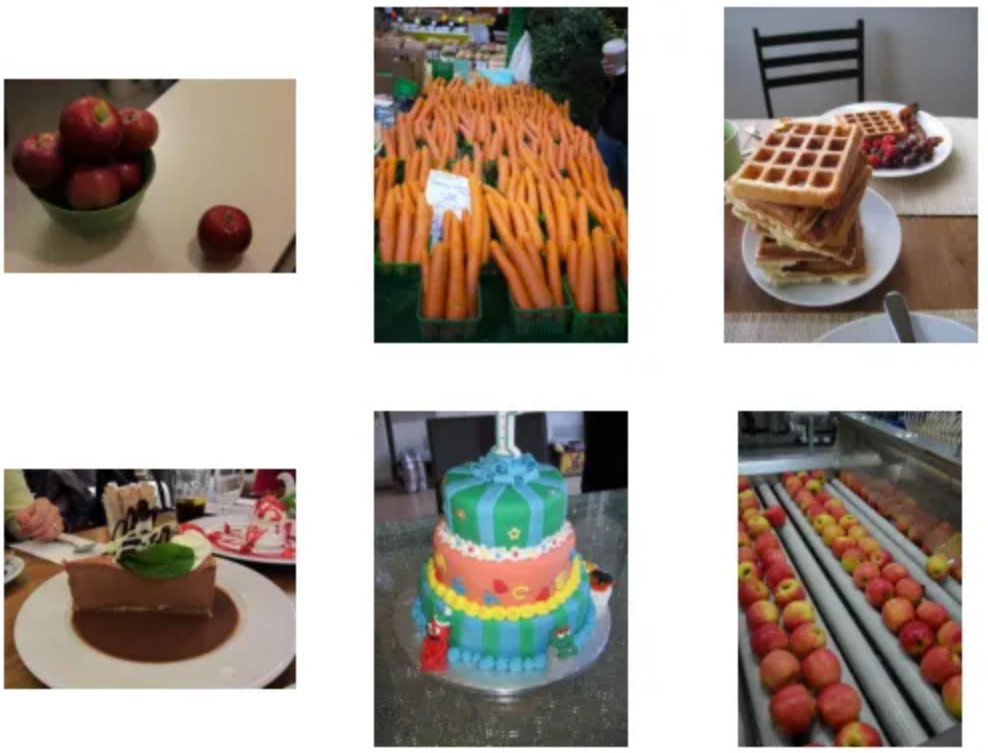
Therefore, in this article, we're going to talk about this model. Specifically, we're going to talk about how a ViT model works and how we can fine-tune it on our own custom dataset with the help of HuggingFace library for an image classification task.

So, as the first step, let's get started with the dataset that we're going to use in this article.

About the Dataset

We will use a snack dataset that you can easily access from `dataset` library from HuggingFace. This dataset is listed as having a CC-BY 2.0 license, which means that you are free to share and use it, as long as you cite the dataset source in your work.

Let's take a sneak peek of this dataset:



Subset of images in the dataset

We only need a few lines of code to load the dataset, as you can see below:

```
!pip install -q datasets

from datasets import load_dataset

# Load dataset
dataset = load_dataset("Matthijs/snacks")
print(dataset)

# Output
  '''
  DatasetDict({
      train: Dataset({
          features: ['image', 'label'],
          num_rows: 4838
      })
      test: Dataset({
          features: ['image', 'label'],
          num_rows: 952
      })
```

```
        validation: Dataset({
            features: ['image', 'label'],
            num_rows: 955
        })
    })'''
```

The dataset is a dictionary object that consists of 4898 training images, 955 validation images, and 952 test images.

Each image comes with a label, which belongs to one of 20 snack classes. We can check these 20 different classes with the following code:

```
print(dataset["train"].features['label'].names)

# Output
'''
['apple','banana','cake','candy','carrot','cookie','doughnut','grape',
'hot dog', 'ice cream','juice','muffin','orange','pineapple','popcorn',
'pretzel','salad','strawberry','waffle','watermelon']'''
```

And let's create a mapping between each label and its corresponding index.

```
# Mapping from label to index and vice versa
labels = dataset["train"].features["label"].names
num_labels = len(dataset["train"].features["label"].names)
label2id, id2label = dict(), dict()
for i, label in enumerate(labels):
    label2id[label] = i
    id2label[i] = label

print(label2id)
print(id2label)

# Output
'''
{'apple': 0, 'banana': 1, 'cake': 2, 'candy': 3, 'carrot': 4, 'cookie': 5, 'dou
{0: 'apple', 1: 'banana', 2: 'cake', 3: 'candy', 4: 'carrot', 5: 'cookie', 6: '
'''
```

How ViT Works

Before the introduction of ViT, the fact that a Transformer model relies on self-attention mechanism raised a big challenge for us to use it for computer vision tasks.

The self-attention mechanism is the reason why Transformer-based models can differentiate the semantic meaning of a word used in different contexts. For example, a BERT model can distinguish the meaning of the word *'park'* in sentences *'They park their car in the basement'* and *'She walks her dog in a park'* due to self-attention mechanism.

Now that we know the dataset that we're working with, let's take a closer look at ViT architecture.

One important thing that we need to know before we move on is the fact that each image has varying dimension. Therefore, we need to perform some image preprocessing steps before feeding the images into the model for fine-tuning purposes.

However, there is one problem with self-attention: it's a computationally expensive operation as it requires each token to attend every other token in a sequence.

Now if we use self-attention mechanism on image data, then each pixel in an image would need to attend and be compared to every other pixel. The problem is, if we increase the pixel value by one, then the computational cost would increase quadratically. This is simply not feasible if we have an image with a reasonably large resolution.