Vision Transformer

Nebin K Raj

Abstract

- The Vision Transformer (ViT) represents a pioneering approach in the field of computer vision by leveraging the transformer architecture originally designed for natural language processing tasks. Departing from the conventional convolutional neural network (CNN) paradigm, ViT treats images as sequences of patches, thereby facilitating the application of transformer-based models to image recognition tasks.
- In the ViT framework, input images are divided into fixed-size patches, which are then transformed into embedding vectors. These patches, along with positional encodings to preserve spatial information, serve as the input to a transformer encoder architecture. The transformer encoder consists of multiple layers of self-attention mechanisms and feedforward neural networks, enabling the model to capture both local and global dependencies within the image.
- Experimental results demonstrate the effectiveness of ViT across various image classification benchmarks, showcasing competitive performance compared to traditional CNN architectures, particularly on large-scale datasets. ViT exhibits notable advantages such as efficient computation, effective utilization of global image context, and adaptability to diverse visual tasks through pre-training and fine-tuning strategies.
- Overall, the Vision Transformer presents a promising direction for advancing the state-of-the-art in image recognition and understanding, offering insights into the potential of transformer-based architectures beyond natural language processing domains.

Objective

1. **Efficiently Process Image Data**: ViT aims to efficiently process image data by treating images as sequences of patches, enabling the application of the transformer architecture to computer vision tasks.
2. **Capture Global Context**: By leveraging self-attention mechanisms within the transformer architecture, ViT seeks to capture both local and global dependencies within images, facilitating effective recognition of complex visual patterns and structures.
3. **Achieve Competitive Performance**: The objective of ViT is to achieve competitive performance on image classification benchmarks, demonstrating its effectiveness compared to traditional convolutional neural network (CNN) architectures, particularly on large-scale datasets.
4. **Enable Transfer Learning**: ViT aims to enable transfer learning by pre-training on large-scale image datasets and fine-tuning on specific downstream tasks, thereby leveraging knowledge learned from diverse visual data sources.
5. **Explore New Architectures**: Beyond image classification, the objective of ViT is to explore the potential of transformer architectures for various computer vision tasks, such as object detection, semantic segmentation, and image generation.

- Overall, the objective of the Vision Transformer is to broaden the scope of transformer-based models, showcasing their adaptability and effectiveness in the domain of computer vision while pushing the boundaries of state-of-the-art performance on image recognition tasks.

Introduction

- The Vision Transformer (ViT) represents a groundbreaking advancement in the field of computer vision, introducing a novel approach to image recognition tasks by drawing inspiration from transformer architectures originally developed for natural language processing (NLP). Traditionally, convolutional neural networks (CNNs) have been the dominant paradigm for image processing tasks due to their ability to capture spatial hierarchies efficiently. However, ViT challenges this convention by treating images as sequences of patches and applying self-attention mechanisms to capture both local and global dependencies within the image.
- At its core, the ViT architecture consists of several key components. First, the input image is divided into fixed-size patches, each of which is then linearly embedded to create a sequence of token representations. These token representations, along with positional encodings to preserve spatial information, serve as the input to a transformer encoder architecture. The transformer encoder consists of multiple layers of self-attention mechanisms and feedforward neural networks, enabling the model to capture intricate relationships between patches across the entire image.
- One of the defining features of ViT is its ability to leverage the power of self-attention mechanisms, allowing the model to attend to relevant patches and capture long-range dependencies within the image. This approach not only enables ViT to achieve competitive performance on standard image classification benchmarks but also offers advantages such as efficient computation and the ability to capture global image context effectively.
- In summary, the Vision Transformer represents a significant paradigm shift in computer vision, showcasing the potential of transformer architectures beyond NLP tasks. By reimagining images as sequences of tokens and leveraging self-attention mechanisms, ViT offers a promising avenue for advancing the state-of-the-art in image recognition and understanding while opening doors to new possibilities in computer vision research and applications.

Methodology

1. **Patch Extraction**: The input image is divided into a grid of fixed-size square patches. Each patch represents a local region of the image. These patches are then linearly embedded into lower-dimensional vectors.
2. **Positional Encoding**: Since the ViT architecture does not inherently encode spatial information like traditional convolutional neural networks (CNNs), positional encodings are added to the patch embeddings. These positional encodings provide information about the spatial layout of the patches within the image.
3. **Tokenization**: The patch embeddings, along with the positional encodings, are treated as sequences of tokens. These tokenized sequences serve as the input to the transformer encoder architecture.
4. **Transformer Encoder**: The tokenized sequences are fed into a standard transformer encoder architecture, which consists of multiple layers of self-attention mechanisms and feedforward neural networks. The self-attention mechanism allows the model to capture both local and global dependencies within the image by attending to relevant patches across the entire image.

1. **Classification Head**: The output of the transformer encoder is typically passed through a classification head, which predicts the class label for the input image. This classification head can be a simple linear layer or a more complex architecture depending on the specific task.
2. **Training**: The ViT model is trained using supervised learning on a large-scale dataset, such as ImageNet. During training, the model learns to map input images to their corresponding class labels by adjusting the parameters of the transformer encoder and classification head through backpropagation and optimization algorithms like stochastic gradient descent.
3. **Fine-tuning and Transfer Learning**: After pre-training on a large-scale dataset, the ViT model can be fine-tuned on specific downstream tasks or datasets. Fine-tuning involves further adjusting the parameters of the model using task-specific data to improve performance on the target task. This process leverages transfer learning, allowing the model to generalize across diverse visual tasks.

- Overall, the methodology of the Vision Transformer combines principles from transformer architectures, such as self-attention mechanisms, with innovative approaches for processing and representing image data as sequences of tokens. This methodology enables ViT to achieve competitive performance on image recognition tasks while offering advantages such as efficient computation and effective capture of global image context.

Code

Load model

In [1]:
```
# Installs the vit_jax package from Github.
!pip install -q git+https://github.com/google-research/vision_transformer
```

```
     |████████████████████████████████| 92 kB 583 kB/s
     |████████████████████████████████| 197 kB 9.2 MB/s
     |████████████████████████████████| 77 kB 4.5 MB/s
     |████████████████████████████████| 4.6 MB 27.4 MB/s
     |████████████████████████████████| 288 kB 48.2 MB/s
     |████████████████████████████████| 4.3 MB 48.6 MB/s
     |████████████████████████████████| 140 kB 61.3 MB/s
     |████████████████████████████████| 216 kB 65.9 MB/s
     |████████████████████████████████| 51 kB 6.9 MB/s
     |████████████████████████████████| 98 kB 7.5 MB/s
     |████████████████████████████████| 72 kB 593 kB/s
     |████████████████████████████████| 511.7 MB 5.6 kB/s
     |████████████████████████████████| 511.7 MB 4.4 kB/s
     |████████████████████████████████| 4.9 MB 48.1 MB/s
  Building wheel for vit-jax (setup.py) ... done
  Building wheel for flaxformer (setup.py) ... done
  Building wheel for ml-collections (setup.py) ... done
```

In [2]:
```
import jax
import jax.numpy as jnp
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_datasets as tfds
import tqdm

from vit_jax import models
```

In [3]:
```
# Currently available LiT models
[name for name in models.model_configs.MODEL_CONFIGS if name.startswith('LiT')]
```

Out[3]:
```
['LiT-B16B', 'LiT-L16L', 'LiT-L16S', 'LiT-L16Ti']
```

In [4]:
```
model_name = 'LiT-B16B'

lit_model = models.get_model(model_name)
# Loading the variables from cloud can take a while the first time...
lit_variables = lit_model.load_variables()
# Creating tokens from freeform text (see next section).
tokenizer = lit_model.get_tokenizer()
# Resizing images & converting value range to -1..1 (see next section).
image_preprocessing = lit_model.get_image_preprocessing()
# Preprocessing op for use in tfds pipeline (see last section).
pp = lit_model.get_pp()
```

```
Loading params from cloud: gs://vit_models/lit/LiT-B16B.npz
100%|██████████| 75/75 [00:53<00:00,  1.39it/s]
⚠️ Reusing local copy: LiT-B16B.npz
```

Use model

In [5]:
```
# Let's load some sample images from tfds.
# Alternatively you can also load these images from the internet / your Drive.
ds = tfds.load('imagenette', split='train')
images_list = [
    example['image'].numpy()
    for _, example in zip(range(5), ds)
]
# Note that this is a list of images with different shapes, not a four
# dimensional tensor.
[image.shape for image in images_list]
```

```
Downloading and preparing dataset 1.45 GiB (download: 1.45 GiB, generated: 1.46 GiB, total: 2.91 GiB) to ~/tensorfl
ow_datasets/imagenette/full-size-v2/1.0.0...
Dl Completed...: 0 url [00:00, ? url/s]
Dl Size...: 0 MiB [00:00, ? MiB/s]
Extraction completed...: 0 file [00:00, ? file/s]
Generating splits...:   0%|          | 0/2 [00:00<?, ? splits/s]
Generating train examples...:   0%|          | 0/9469 [00:00<?, ? examples/s]
Shuffling ~/tensorflow_datasets/imagenette/full-size-v2/1.0.0.incomplete7T8YUJ/imagenette-train.tfrecord*...: …
Generating validation examples...:   0%|          | 0/3925 [00:00<?, ? examples/s]
Shuffling ~/tensorflow_datasets/imagenette/full-size-v2/1.0.0.incomplete7T8YUJ/imagenette-validation.tfrecord*…
Dataset imagenette downloaded and prepared to ~/tensorflow_datasets/imagenette/full-size-v2/1.0.0. Subsequent calls
will reuse this data.
```

Out[5]:
```
[(101, 125, 3), (375, 500, 3), (335, 500, 3), (429, 500, 3), (359, 500, 3)]
```

In [6]:
```
# Note that our preprocessing converts to floats ranging from -1..1 !
images = image_preprocessing(images_list)
images.shape, images.min(), images.max()
```

Out[6]:
```
((5, 224, 224, 3), -1.0, 1.0)
```

In [7]:
```
plt.figure(figsize=(15, 4))
plt.imshow(np.hstack(images) * .5 + .5)
plt.axis('off');
```

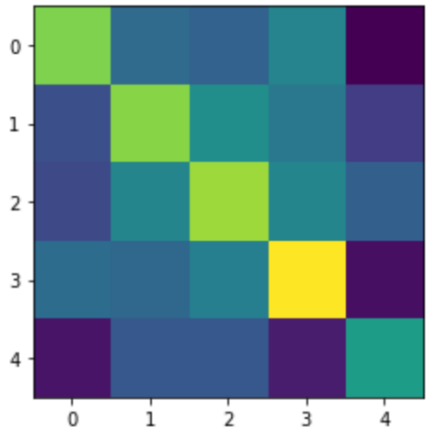
In [8]:
```
texts = [
    'itap of a cd player',
    'a photo of a truck',
    'gas station',
    'chainsaw',
    'a bad photo of colorful houses',
]
tokens = tokenizer(texts)
tokens.shape
```

Out[8]:
```
(5, 16)
```

In [9]:
```
# Embed both texts and images with a single model call.
# See next section for embedding images/texts separately.
zimg, ztxt, out = lit_model.apply(lit_variables, images=images, tokens=tokens)
```

In [10]:
```
plt.imshow(ztxt @ zimg.T)
```

Out[10]:
```
<matplotlib.image.AxesImage at 0x7f7cc4dd5b90>
```



In [11]:
```
probs = np.array(jax.nn.softmax(out['t'] * ztxt @ zimg.T, axis=1))
pd.DataFrame(probs, index=texts).style.background_gradient('Greens', vmin=0, vmax=1).format('{:.2%}')
```

Out[11]:

	0	1	2	3	4
itap of a cd player	99.71%	0.04%	0.02%	0.23%	0.00%
a photo of a truck	0.01%	99.55%	0.37%	0.07%	0.00%
gas station	0.00%	0.11%	99.77%	0.11%	0.01%
chainsaw	0.00%	0.00%	0.01%	99.99%	0.00%
a bad photo of colorful houses	0.02%	0.80%	0.82%	0.03%	98.34%

`tfds` zero-shot evaluation

In [12]:
```
#@markdown `imagenet_classnames`
# From https://github.com/openai/CLIP/blob/main/notebooks/Prompt_Engineering_for_ImageNet.ipynb
imagenet_classnames = "tench;goldfish;great white shark;tiger shark;hammerhead shark;electric ray;stingray;rooste
```

In [13]:
```
# # For imagenet evaluation, first prepare the dataset on a GCS bucket as
# # described in
# # https://www.tensorflow.org/datasets/catalog/imagenet2012
# # and then replace `data_dir` below with that GCS bucket.
# # If you get a `PermissionDeniedError`, try restarting the kernel.
# from google.colab import auth
# auth.authenticate_user()  # Required to access access protected GCS buckets.
# builder = tfds.builder('imagenet2012', data_dir='gs://tensorflow-datasets/datasets')
# ds_test = builder.as_dataset('validation')
# info = builder.info
# classnames = imagenet_classnames
```

Conclusion

- In conclusion, the Vision Transformer (ViT) stands as a pioneering advancement in the realm of computer vision, revolutionizing traditional approaches to image recognition tasks. By reimagining images as sequences of tokens and leveraging the power of transformer architectures originally designed for natural language processing (NLP), ViT opens new avenues for understanding and processing visual data.
- Throughout this journey, ViT introduces a novel methodology that combines patch extraction, positional encoding, tokenization, and transformer encoder architecture to effectively capture both local and global dependencies within images. By treating images as sequences of patches and applying self-attention mechanisms, ViT demonstrates remarkable adaptability and performance on standard image classification benchmarks.
- Moreover, ViT offers several key advantages, including efficient computation, effective utilization of global image context, and the ability to leverage transfer learning for diverse visual tasks. Through pre-training on large-scale datasets and fine-tuning on specific downstream tasks, ViT showcases its versatility and generalization capabilities across a wide range of applications in computer vision.
- In essence, the Vision Transformer heralds a new era in computer vision, where the boundaries between images and language blur, and the possibilities for understanding and interpreting visual data are limitless. As we continue to explore the vast landscape of computer vision, ViT serves as a guiding light, illuminating the path towards deeper insights, richer understanding, and transformative applications in the visual domain.