Lab 5-1: Vision Transformer final-report

CHI YEONG HEO[1],
[1]School of Computer Science and Engineering, Pusan National University, Busan 46241 Republic of Korea

1. Introduction

This report provides an analysis of the paper *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* [1], with a focus on the "vit_base_patch16_224" model, as implemented in [3].

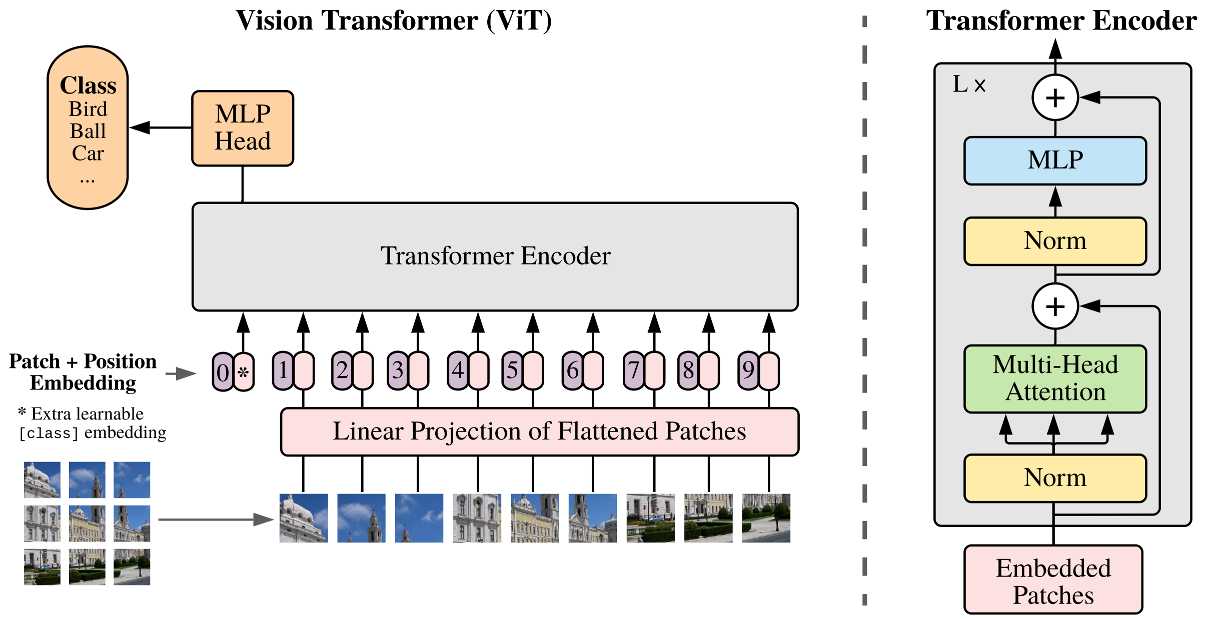

Figure 1: Model overview. ViT split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, ViT uses the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. [2].

2. Vision Transformer

The Vision Transformer (ViT) represents a groundbreaking advancement in applying transformer architecture, originally designed for natural language processing, to the domain of image recognition. Unlike conventional convolutional neural networks (CNNs), ViT divides an image into a series of patches, which are then treated as input tokens analogous to words in natural language processing. This architecture excels in parallel data processing and effectively captures long-range dependencies, demonstrating competitive performance across various benchmarks. Notably, ViT achieves outstanding results on diverse image datasets, particularly when fine-tuned from pre-trained models. By addressing the limitations inherent in CNN-based models, ViT unlocks new possibilities in image recognition, classification, and other computer vision tasks. These attributes have garnered significant attention, solidifying ViT's prominence within the field of computer vision.

The standard Transformer architecture is designed to process 1D sequences of token embeddings. In adapting this structure for 2D images, an input image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ is partitioned into a sequence of flattened 2D patches, denoted as $\mathbf{x}_p \in \mathbb{R}^{N\times(P^2\cdot C)}$. Here, (H,W) represents the resolution of the original image, C is the number of channels, (P,P) is the resolution of each individual image patch, and $N = HW/P^2$ defines the total number of patches, effectively setting the length of the input sequence for the Transformer. Each patch is flattened and then mapped into a D-dimensional space through a trainable linear projection (Eq. 1).

Similar to the `[class]` token in BERT, a learnable embedding $\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}$ is prepended to the sequence of embedded patches, serving as the image representation $\mathbf{y}$ obtained at the output of the Transformer encoder ($\mathbf{z}_L^0$) (Eq. 4). For classification, an MLP with one hidden layer during pre-training, replaced by a single linear layer in fine-tuning, is applied to $\mathbf{z}_L^0$.

Positional embeddings are added to each patch embedding to retain spatial information, forming the complete input sequence for the encoder.

The Transformer encoder consists of alternating layers of multi-head self-attention (MSA) and MLP blocks (Eq. 2, 3), each preceded by Layer Normalization (LN) and followed by a residual connection. The MLP block has two layers and uses a GELU activation function.

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \quad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{1}$$

$$\mathbf{z'}_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1\ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, \qquad \ell = 1\ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

Inductive Bias. The ViT model incorporates less image-specific inductive bias compared to CNNs. In CNNs, characteristics such as locality, two-dimensional neighborhood structure, and translation equivariance are explicitly integrated across layers, shaping the model's entire structure. In contrast, ViT models have only local and translationally equivariant biases within the MLP layers, while the self-attention layers operate on a global scale.

Hybrid Architecture. An alternative to using raw image patches as input involves employing feature maps derived from a CNN. In this hybrid approach, the patch embedding projection $\mathbf{E}$ (Eq. 1) is applied to patches that are extracted from a CNN feature map, combining the benefits of CNNs and Transformers.

3. Implementation Details

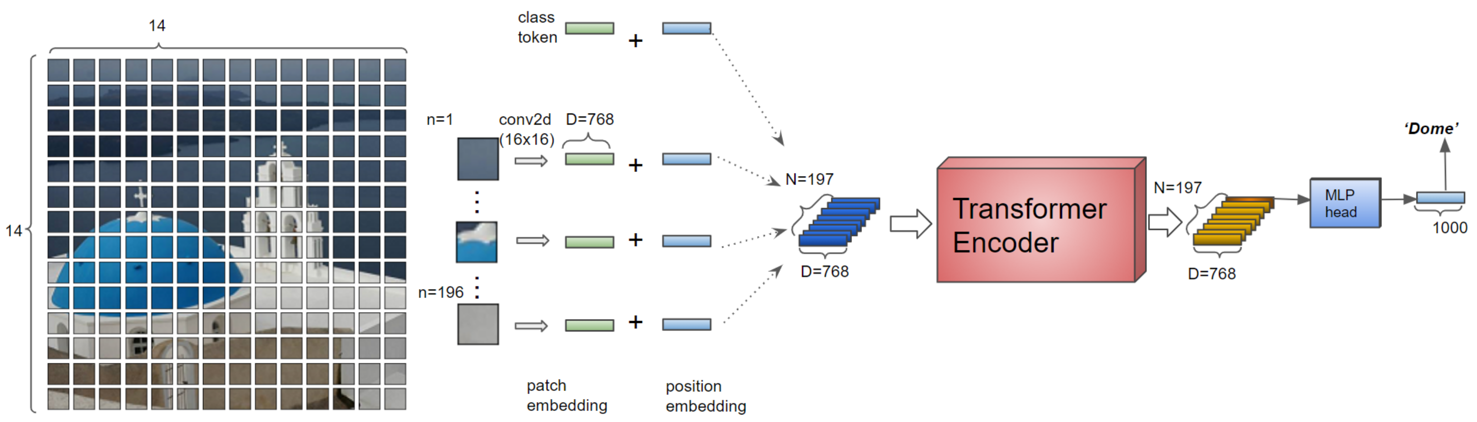

Figure 2: The inference pipeline of Vision Transformer

The inference pipeline of the Vision Transformer (ViT) comprises four primary steps:

- **Image patch extraction**
  - ▷ The input image of size 224×224 is divided into 14×14 patches using a two-dimensional convolutional filter with a kernel size of 16×16 and a stride of 16×16.
- **Addition of positional embeddings**
  - ▷ A learnable positional embedding vector is added to each patch embedding vector, and the resulting sequence is input into the transformer encoder.
- **Transformer encoder processing**
  - ▷ The transformer encoder processes the input, maintaining identical dimensions for the input and output vectors.
- **Classification using the MLP head**
  - ▷ The first output token from the transformer encoder is passed through a multi-layer perceptron (MLP) head to produce the final classification result.

3.1. Image patch extraction

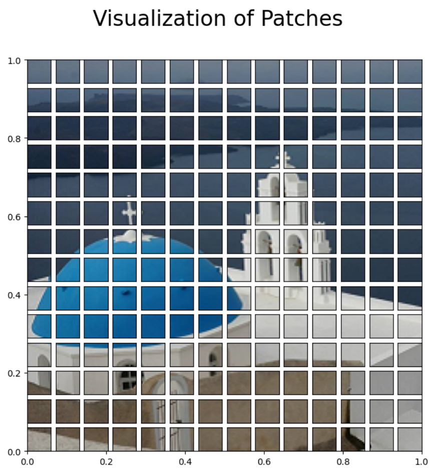

Figure 3: Visualization of image patch splitting

The input image, with dimensions 224×224, is divided into 14×14 patches using a two-dimensional convolutional filter with a kernel size of 16×16 and a stride of 16×16. Each patch is subsequently transformed into an embedding vector with a dimensionality of 768. Consequently, the output of the image patch extraction process transforms the input tensor, with dimensions $(B, 3, 224, 224)$, into a tensor with dimensions $(B, 196, 768)$, where B denotes the batch size.

3.2. Addition of positional embeddings

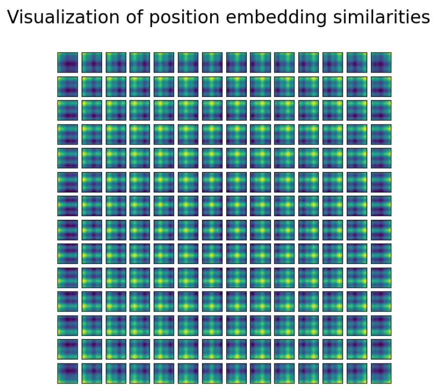

Figure 4: Visualization of positional embedding similarities. Each cell in the grid represents the cosine similarity between the positional embedding of the corresponding patch and all other patches. Brighter regions within each cell indicate higher similarity values (maximum of +1), while darker regions indicate lower similarity values (minimum of -1).

A learnable positional embedding vector is added to each patch embedding vector, and the resulting sequence is input into the transformer encoder. The pre-trained "vit_base_patch16_224" model has pre-trained positional embeddings. Fig. 4 shows that the trained positional embeddings show that closer patches tend to have more similar position embeddings. Further, the row-column structure appears; patches in the same row/column have similar embeddings.

3.3. Transformer encoder processing

The transformer encoder processes the input, maintaining identical dimensions for the input and output vectors. A learnable class token is prepended to the patch embedding vectors as the 0th vector. $197(1 + 14 \times 14)$ learnable position embedding vectors are added to the patch embedding vectors.

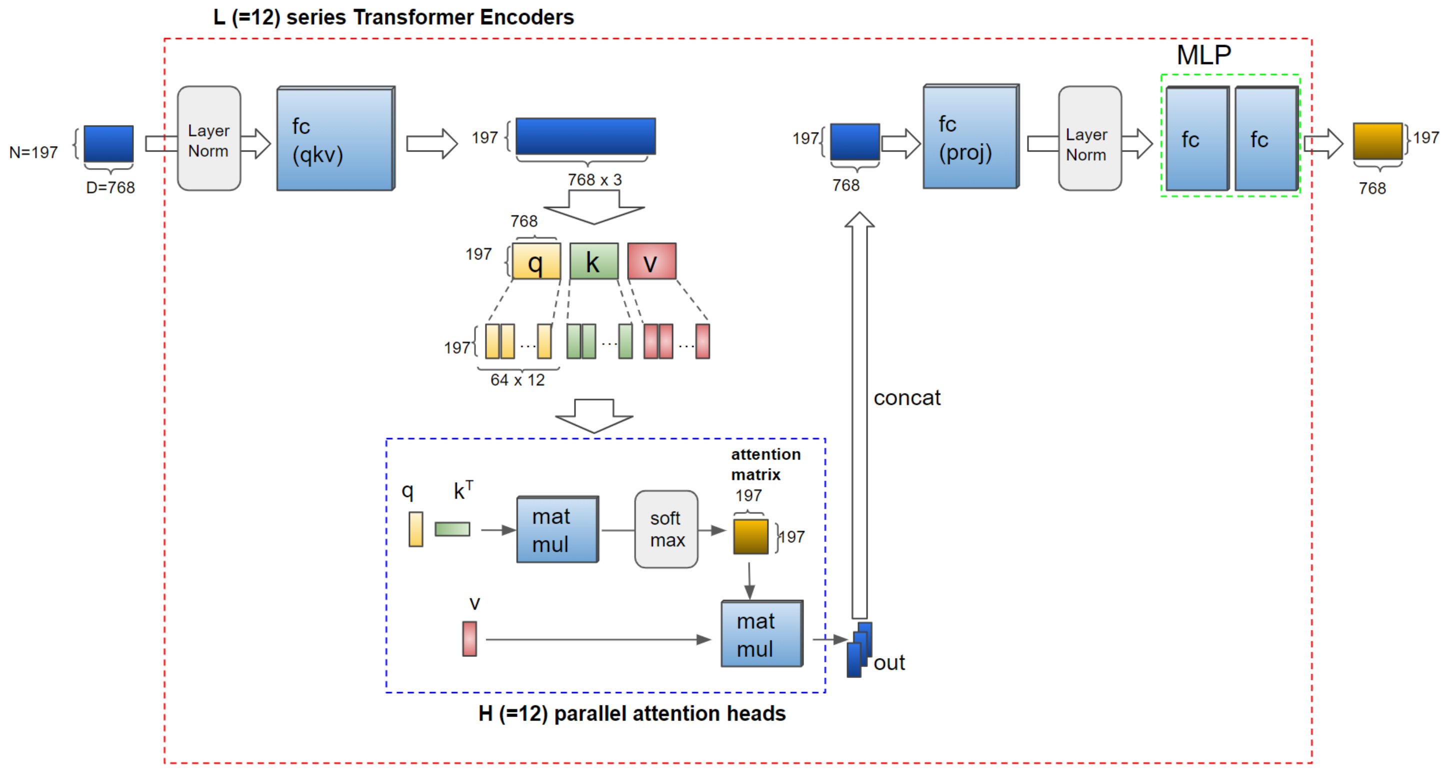

Figure 5: The inference pipeline of Vision Transformer

The process within the transformer encoder consists of the following steps:

- A total of $N = 197$ embedding vectors are passed through a series of $L = 12$ transformer encoder layers.
- Each embedding vector is expanded to a dimension of 768×3 and subsequently divided into query (q), key (k), and value (v) components.
- The outputs from the attention heads are concatenated to match the shape of the input vectors to the encoder.
- Following the attention mechanism, the vectors are processed through a layer normalization step, followed by two fully connected layers, resulting in the encoder's output.

3.3.1. Scaled Dot-Product Attention

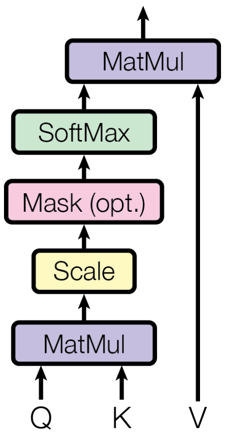

Figure 6: Scaled Dot-Product Attention.

The ViT employs the transformer encoder architecture proposed in [2]. This encoder utilizes an attention mechanism called "Scaled Dot-Product Attention" (Figure 6). The input to this mechanism consists of queries and keys, each of dimensionality d_k, and values of dimensionality d_v. The attention mechanism computes the dot products between the query and all keys, scales them by dividing each result by $\sqrt{d_k}$, and applies a softmax function to generate the attention weights, which are then used to weight the values.

In practical implementations, the attention function is computed simultaneously for a set of queries, which are aggregated into a matrix Q. Similarly, the keys and values are packed into matrices K and V, respectively. The output matrix is then computed as follows:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

As noted in [2], for large values of d_k, the magnitude of the dot products increases significantly, which can cause the softmax function to operate in regions with extremely small gradients.

To demonstrate why the dot products increase with d_k, assume that the components of the query vector q and the key vector k are independent random variables with a mean of 0 and a variance of 1. Their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, will then have a mean of 0 and a variance of d_k. To mitigate this effect, the dot products are scaled by a factor of $\frac{1}{\sqrt{d_k}}$, ensuring stable gradient magnitudes during training.

3.3.2. Multi-Head Attention

Instead of applying a single attention function using keys, values, and queries of dimensionality d_{model}, it has been found beneficial to project the queries, keys, and values h times using different learned linear projections into d_k, d_k, and d_v dimensions, respectively. Each of these projected versions undergoes the attention function in parallel, producing d_v-dimensional output values. These outputs are then concatenated and linearly projected once more to yield the final output, as illustrated in Figure 7.

The use of multi-head attention enables the model to jointly attend to in-

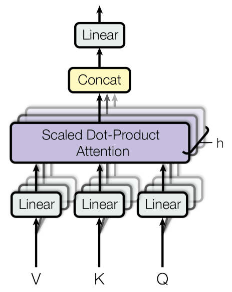

Figure 7: Multi-Head Attention consists of several attention layers running in parallel.

formation from different representation subspaces at various positions. In contrast, using a single attention head results in averaging across these subspaces, which can inhibit expressiveness.

The multi-head attention mechanism is formally defined as:

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_\text{h})W^O\\ \text{where head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

Here, $W_i^Q \in \mathbb{R}^{d_\text{model}\times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model}\times d_v}$, and $W^O \in \mathbb{R}^{hd_v\times d_\text{model}}$ are the parameter matrices for the projections.

The "vit_base_patch16_224" model employs $h = 12$ parallel attention heads. For this configuration, the dimensions are set as $d_k = d_v = d_\text{model}/h = 64$. By reducing the dimensionality of each attention head, the overall computational cost remains comparable to that of single-head attention using the full dimensionality.

3.3.3. Attention Matrix

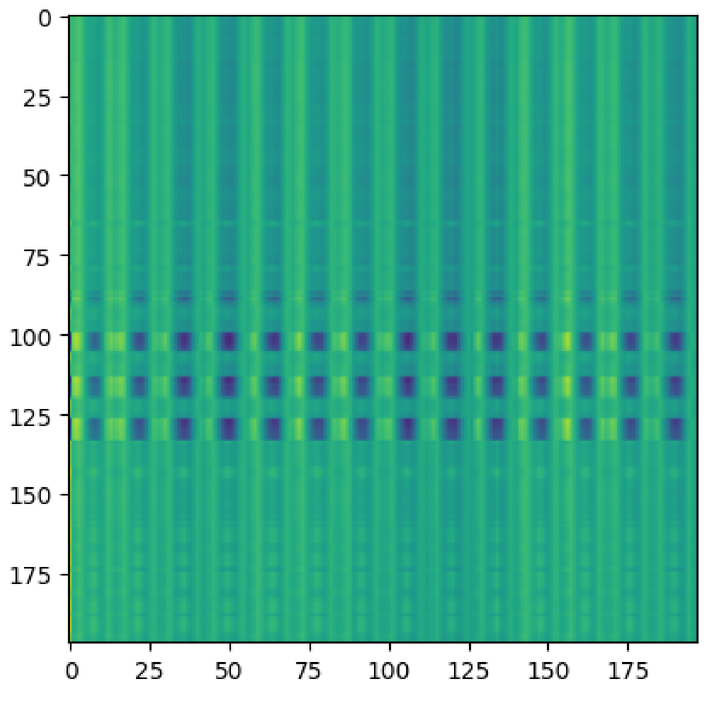

Figure 8: Visualization of the attention matrix for the Santorini input image in attention head 3.

The attention matrix is computed using the formula:

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right).$$

This matrix reveals how each token "attends" to other tokens, illustrating the relationships between them. By leveraging this property, we can visualize how each attention head in the ViT model focuses on specific parts of an image. Different attention heads emphasize various image patches, enabling a comprehensive understanding of the entire image by aggregating multiple perspectives.

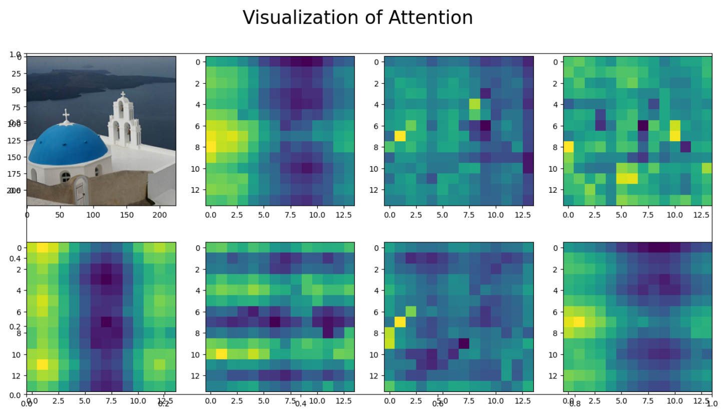

Figure 9: Visualization of the attention matrices for attention heads 0 to 6, focusing on the patch indexed at 100. Brighter areas indicate higher attention values, while darker areas represent lower attention values. Each attention head emphasizes different regions of the image, highlighting diverse aspects of the input.

3.4. Classification using the MLP head

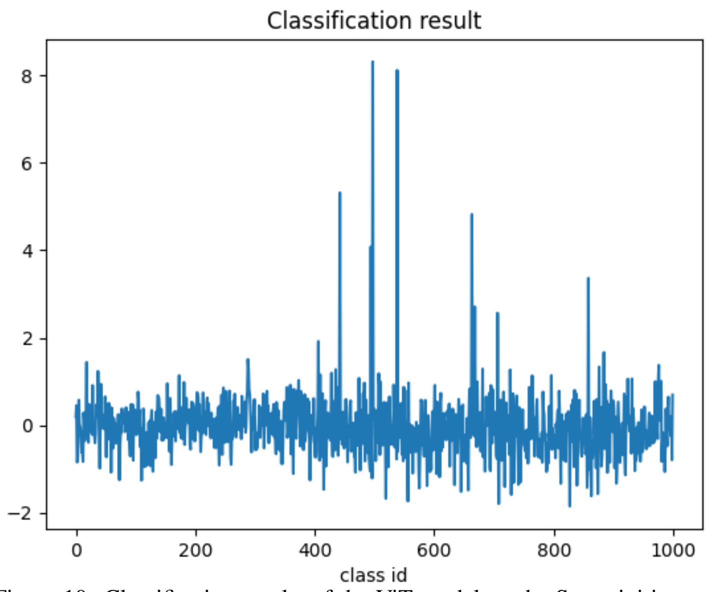

Figure 10: Classification results of the ViT model on the Santorini image. The confidence logits for each class are shown. The class ID 497 has the highest value among 1000 classes, leading the model to predict the image as "church, church_building," which is correct.

The first output token from the transformer encoder, referred to as the class token, is passed through an MLP head to produce the final classification.

In the "vit_base_patch16_224" model, the MLP head applies a linear transformation that maps the input dimension of 768 to the output dimension of 1000, corresponding to the number of classes. The model predicts the class of an input image by selecting the class label with the highest logit value after the MLP head.

4. Conclusion

This report has presented a comprehensive analysis of the Vision Transformer (ViT) architecture and its components, emphasizing its novel approach to image classification through the use of transformers. This report detailed the core mechanisms, including positional embeddings, multi-head attention, and the final classification process using the MLP head. By visualizing the attention matrices and position embedding similarities, it demonstrated how ViT attends to different parts of the image, effectively capturing spatial relationships and semantic information.

The classification results highlight the model's ability to correctly predict image labels with high confidence, showcasing its effectiveness in leveraging attention mechanisms for understanding complex visual features. Furthermore, the modular and scalable design of ViT opens opportunities for applying transformer-based architectures to a wide range of vision tasks.

In conclusion, the Vision Transformer represents a paradigm shift in computer vision, successfully applying transformer principles from NLP to image processing. Its performance and flexibility suggest that it will continue to play a pivotal role in advancing state-of-the-art visual recognition systems.

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[3] Ross Wightman. PyTorch Image Models. URL https://github.com/huggingface/pytorch-image-models.