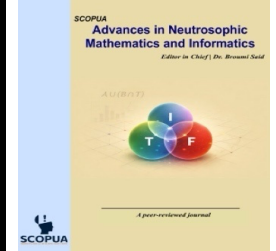




Advances in Neutrosophic Mathematics and Informatics (ANMI)

Vol.1, Issue 2, October 2026
DOI:<https://doi.org/10.64060/ANMIV1i31>



Vision Transformers: From Theory to Implementation

Abdelhafid Berroukham¹, Mohammed Lahraichi^{1*}

¹Faculty of Science, Ibne Tofail University, Kenitra, Morocco

* Corresponding Email: lahraichi.mohamed@gmail.com

Received: 30 March 2026 / Revised: 02 August 2026 / Accepted: 30 September 2026 / Published online: 20 October 2026

This is an Open Access article published under the Creative Commons Attribution 4.0 International (CC BY 4.0) (<https://creativecommons.org/licenses/by/4.0/>). Advances in Neutrosophic Mathematics and Informatics published by SCOPUA (Scientific Collaborative Online Publishing Universal Academy). SCOPUA stands neutral with regard to jurisdictional claims in the published maps and institutional affiliations.

ABSTRACT

The field of computer vision has passed through a paradigm shift with the appearance of Vision Transformers (ViTs), an revolutionary technique inspired by natural language processing (NLP). ViTs represents a paradigm shift, transferring far from traditional convolutional neural networks (CNNs) by the usage of the strength of self-attention mechanisms for comprehensive image understanding. This paper offers a comprehensive explanation of ViTs, starting with an exploration of their theoretical foundations and concepts, along with the mechanisms of self-attention and multi-headed attention, to their implementation. It presents practical insights for researchers and practitioners. Furthermore, we highlighted the actual-world applications of ViTs, which cover a number of domains from healthcare to autonomous systems, in order to reveal their adaptability and potential.

Keywords: Computer Vision; Deep Learning; Vision Transformers; Convolutional Neural Networks

1. Introduction

In the evolution of the computer vision field, Vision Transformers (ViTs)[1] have emerged as a paradigm shift. The vision transformer is an extension of the Transformer[2] for image data, which is originally used for Natural Language Processing (NLP), ViTs is a new way to approach image analysis and understanding. This paper begins with comprehensive exploration of ViTs, illuminating its conceptual foundations, architectural components, and practical implementation.

Traditionally, the field of computer vision has been dominated by Convolutional Neural Networks (CNNs)[3]. Although CNNs have effectively excelled at a variety of image-related tasks, they frequently have limitations due to hierarchical feature extraction (fixed)... In contrast, ViT overcomes the constraints of grids and rely on self-attentional mechanisms to capture long-range dependencies and relationships within images.

The theoretical basis of ViT is strongly founded on the concept of attention, which made it popular initially by the Transformer architecture in NLP[2]. Self-attention mechanisms allow ViT to focus on different parts of an image dynamically, facilitating the capture of contextual information from the image and enabling a comprehensive understanding of visual content. Due to the elimination of fixed receptive fields, ViT can now perform well in many tasks such as image classification[4], [5], object detection[6], [7], image segmentation[8], [9], and image captioning.

In this paper, we will analyze each component of the Vision Transformers architecture, from the initial input embeddings to the position encodings and Multi-Head Attention mechanisms. This deconstruction will clarify the internal workings of ViT and clarify the fundamental differences between ViT and CNN.

After explaining the theoretical background of the ViT, we will provide a guide to their implementation, offering code snippets, libraries, and tools that enable researchers and practitioners to effectively utilize the capabilities of ViTs. Finally, we will explore real-world applications and use cases where ViT surpassed the performance of traditional models and achieve good results.

However, it is essential to acknowledge the challenges and limitations that accompany this transformative shift. ViT requires substantial computational resources during training and might face difficulties in handling extremely high-resolution images. Moreover, the interpretability of ViT remains an ongoing challenge that warrants continued research and development.



Overall, Vision Transformers represent a paradigm shift in computer vision, offering a creative perspective on how to capture and analyze visual information. This paper aims to present an overview between the theoretical foundations of ViT and its practical implementation, providing a comprehensive resource for those eager to explore and leverage this approach in their computer vision tasks. Vision Transformers is a world where attention to detail transcends the pixel grid and paves the way for a new era of visual understanding.

The rest of the paper is organized as follows: First, we give an overview of the theoretical background behind Vision Transformers. Then we present the architecture of the ViT in detail, in the section tree we provide a practical implementation of ViT, and finally a conclusion.

2. THEORETICAL BACKGROUND

2.1 Theoretical foundations

The theoretical foundations of Vision Transformers (ViTs) are based on the concept of attention mechanisms, which was originally popularized by the Transformer architecture in the domain of natural language processing (NLP)[2]. ViT uses self-attention mechanisms at its core to process visual data. Unlike traditional Convolutional Neural Networks (CNNs) with fixed receptive fields, ViT dynamically analyzes the importance of various image regions when making predictions. Which means that ViT is excellent at capturing long-range dependencies, allowing them to take into account the context and relationships between different parts of an image at the same time. Key to this process are the Queries, Keys, and Values that ViT computes for each pixel in an image, allowing them to adapt and learn from the content of different regions. Thus, ViT surpasses the limitations of grid-based processing, enabling a full comprehension of complex visual information. As a result, the theoretical foundations of ViTs mark a paradigm shift in computer vision, opening the door to the possibility of more context understanding and adaptive models for image processing

2.2 The key concepts of Vision Transformer

There are two essential concepts that support the transformative capabilities of Vision Transformers (ViT)[1], Self-Attention mechanisms and Multi-Head Attention. Self-attention mechanisms enable ViT to capture complex relationships within an image by dynamically assigning different levels of importance to various regions during processing. This flexibility is essential to understand the context and long-range dependencies in visual data. Multi-head attention extends this concept by allowing ViT to perform attention operations multiple times in parallel, each one concentrating on different parts of the input images, as you can see in the Figure 1, where each head focuses in a specific region. This parallelization improves the model's ability to capture multiple types of information and makes it easier to understand the different features of an image. The combination of self-attention mechanisms and multi-head attention makes ViT able to successfully distinguish pertinent information, model dependencies, and hierarchical features. Thus, a powerful model for image understanding and analysis in diverse applications and an excellent tool for complex computer vision tasks.

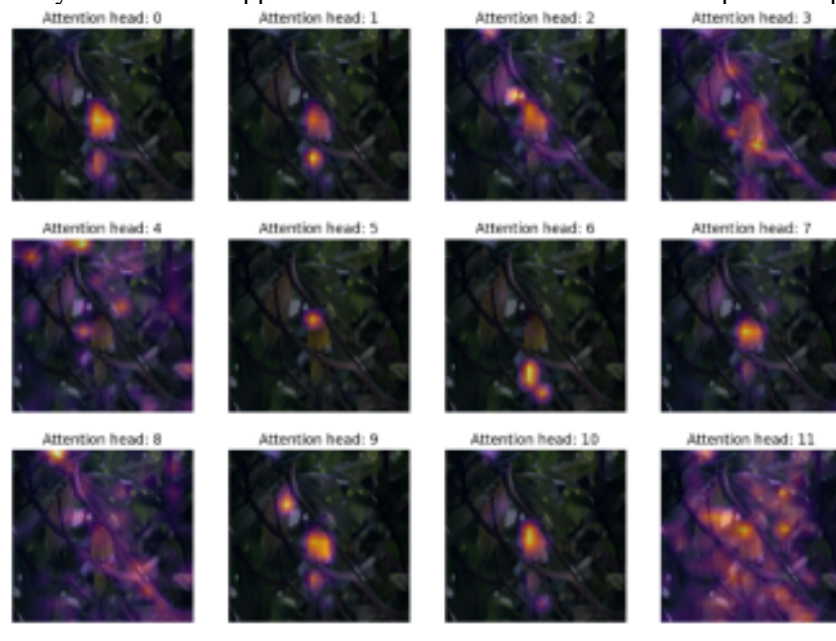


Figure 1: illustration of the Attention weights of each head

3. ARCHITECTURE OF VISION TRANSFORMERS



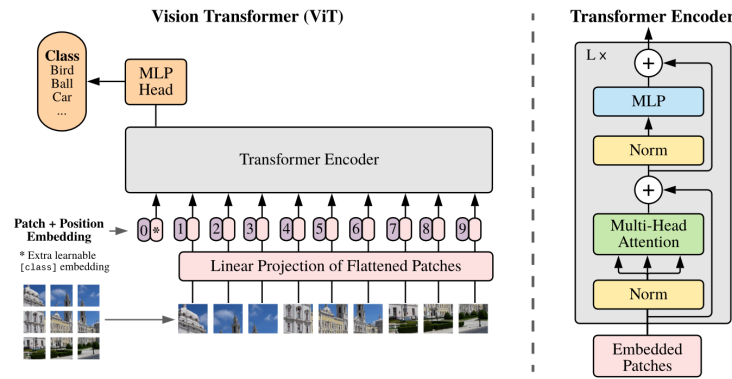


Figure 2: Original architecture of ViT[1]

Figure 2 illustrates the original architecture of Vision Transformers (ViT)[1], which marks a revolutionary change from the traditional Convolutional Neural Networks (CNNs). At its core, ViTs rely on the foundational Transformer architecture[2], adapted for the domain of computer vision. ViT begins by splitting an image with a resolution of 224x224 into patches with fixed sizes 16x16, a total of 196 patches are created from one image, then it flatten the image patches, and create lower-dimensional linear embeddings from these flattened image patches. These patch embeddings combined with positional encodings to provide the localization of each patch in the image, serve as the input to the encoder of the model as you can see in the Figure 2. The encoder block is a series of stacked Transformer encoder layers (typically 12), each one composed of Multi-Head Self-Attention mechanisms (MSA), Feed Forward Neural Networks, Normalization block and Residual Connections. Figure 3 illustrates the internal elements of each component of the encoder block in detail. The Self-Attention mechanisms enable ViT to weigh the relevance of different patches in the image dynamically, capturing both local and global relationships. The Multi-Head Attention mechanism parallelizes this process, allowing the model to focus on various aspects of the input simultaneously, It has three inputs which are Queries, Keys and values. Skip connections and layer normalization contribute to stable training, the skip connection are used at different places in the Transformer encoder, these connections are mainly to improve the flow of information to avoid vanishing gradients. Importantly, the output of the final Transformer layer is often aggregated, an MLP heads are attached for specific tasks, such as image classification. This architectural design empowers ViT to adapt dynamically to different image contents and captures complex spatial relationships, making them highly effective in a wide range of computer vision tasks.

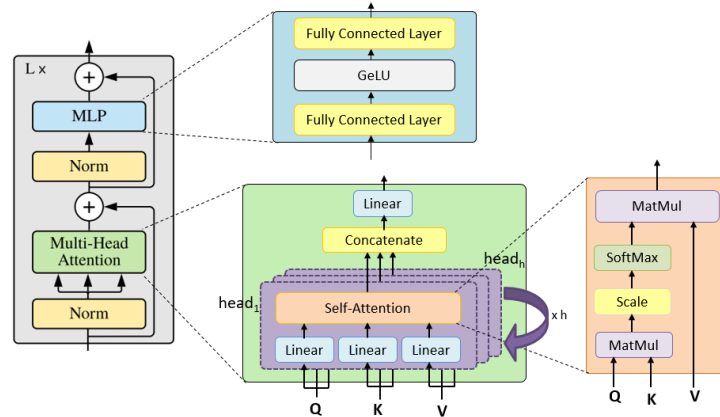


Figure 3: The Vision Transformers encoder block

Theoretically, Vision Transformer encoder receives as input a 1D sequence of patch embeddings. To handle 2D images, it reshapes the image $X \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $X_p \in \mathbb{R}^{N \times (P,P,C)}$, where (H,W) is the resolution of the original image, C is the number of channels, (P,P) is the resolution of each image patch, and $N=HW/P^2$ is the number of patches. The ViT uses fixed size of the patch embeddings D through all of its layers. Equation 1 presents the output of the initial patch embedding layer, where X_{class} presents the class token, X_p is the patch embedding and E_{pos} is position embedding.

For every layer from 1 to L which is the total number of encoder layers, there is a Multi-Head Attention layer (MSA) wrapping a Normalization layer (LN) Equation 2, the addition on the end of the equation is equivalent to a residual connection which add the input to the output the MSA block.

The MLP block can be represented by the equation 3, where a Multilayer Perceptron layer (MLP) wraps a Normalization layer (LN), and the addition on the end presents a residual connection.



Finally, for the last layer L , the output y is the 0 index token of z wrapped in a Normalization layer (LN), and it can be presented by the equation 4.

$$Z_0 = [X_{\text{class}}; X_{1p}E; X_{2p}E; \dots; X_{Np}E] + E_{\text{pos}} \quad (1)$$

Where $E \in \mathbb{R}^{(P \cdot P \cdot C) \times D}$, $E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$

$$z'_{\ell} = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1 \dots L \quad (2)$$

$$z_{\ell} = \text{MLP}(\text{LN}(z'_{\ell})) + z'_{\ell}, \quad \ell = 1 \dots L \quad (3)$$

$$y = \text{LN}(z_L^0) \quad (4)$$

4. PRACTICAL IMPLEMENTATION

In this section we will present the fundamental steps to implement the vision transformers model from scratch using the PyTorch framework, we are going to implement the model block by block with a bottom-up approach.

In the first step, the inputs are transformed into numeric vectors or embeddings. In the transformer the inputs are specific tokens that are transformed to embeddings, typically these are words. While, in the Vision transformer the embeddings for images are done using image patches. In practice we patch an image into 16x16 pixel tiles then we pass each of the patches through the first component of the ViT which is the embedding block to give us the embeddings.

To perform the patching efficiently, the input image consists of three color channels (red, green and blue). We also have a specific width and height plus potentially a batch dimension if we have more than one image.

And as you can see in the figure 1, to rearrange the image on the top into patches of a specific patch size on the bottom, which we will use as input for the embedding block, we use of einops which is a package that lets us rearrange multi-dimensional arrays and tensors in various ways. The reshaping can be expressed through the following syntax:

`Rearrange('b c (h p1) (w p2) -> b (h w) (p1 p2 c)')`

Where h and w indicate the dimensions of the input, c indicate the number of channels, and b indicate the batch dimension.

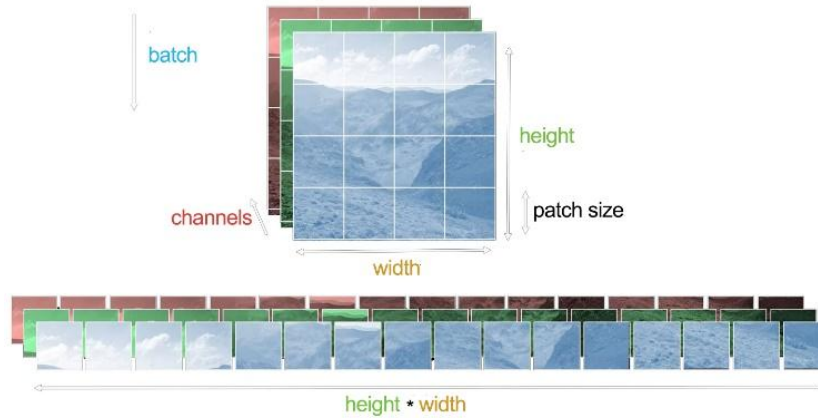


Figure 4: illustration of the image patches

The embedding block is simply a fully connected neural network, this means each of the color values is put into neurons and passed through some linear layers the output is an embedding.

For the implementation, we define a pytorch module called patch embedding which does the reshaping described above and after that it applies a linear transformation, which gives us these embedding vectors, this is an example that take a sample data point from the dataset:

```
class PatchEmbedding(nn.Module):
    def __init__(self, in_channels=3, patch_size=8, emb_size=128):
        self.patch_size = patch_size
        super().__init__()
        self.projection = nn.Sequential(
            Rearrange('b c (h p1) (w p2) -> b (h w) (p1 p2 c)', p1=patch_size, p2=patch_size),
            nn.Linear(patch_size*patch_size*in_channels, emb_size))
    def forward(self, x:Tensor) - Tensor:
        x = self.projection(x)
        return x
```

A positional embedding is added to all of the input embedding vectors, which are vectors that make the model understand where each patch is placed in the original image.

The positional embeddings are typically added on top of the input embedding vectors and not concatenated, in the implementation, the positional embedding is a learnable parameter vector per patch with a specific dimension as you can see here:



```
self.pos_embedding = nn.Parameter(
    torch.randn(1, num_patches + 1, emb_dim))
```

And the last component of the ViT is the transformer encoder, from the right part of the figure , you can see the components that need to be implemented namely : the Multi-Head Attention block, the Normalization, a Feed Forward Network and finally the Residual Connections. On the left of this block we see that this block is repeated L times (Typically 12).

Multi-head attention is the scale dot product attention mechanism of the Transformer and allows sharing information between the different inputs, as you can see in the figure, it has three inputs which are queries, keys and values. There is a PyTorch implementation of the Multi-Head attention block, so no need to implement this block from scratch.

This is an example of the implementation of attention where we used Multi-Head Attention from a torch, then we have a linear transformation for each of the Keys, Queries and Values and our input patch goes through each of them and then we apply attention and return the attention output.

```
class Attention(nn.Module):
    def __init__(self, dim, n_heads, dropout):
        super().__init__()
        self.n_heads = n_heads
        self.att = torch.nn.MultiheadAttention(embed_dim=dim, num_heads=n_heads, dropout=dropout)
        self.q = torch.nn.Linear(dim, dim)
        self.k = torch.nn.Linear(dim, dim)
        self.v = torch.nn.Linear(dim, dim)
    def forward(self, x):
        q = self.q(x)
        k = self.k(x)
        v = self.v(x)
        attn_output, attn_output_weights = self.att(x, x, x)
        return attn_output
```

The Normalization block in the transformer is layer Normalization, it normalizes all inputs in a layer for each individual sample.

```
class PreNorm(nn.Module):
    def __init__(self, dim, fn):
        super().__init__()
        self.norm = nn.LayerNorm(dim)
        self.fn = fn
    def forward(self, x, **kwargs):
        return self.fn(self.norm(x), **kwargs)
```

For the Feed Forward block, it's just a linear layer that takes the attention-weighted vectors as inputs and transforms them for the next layer.

Practically, we have two linear layers and in between them, we have the Gaussian Error Linear Units (GELU)[10] which is the activation function that is used, and we have dropouts in between to avoid overfitting.

```
class FeedForward(nn.Sequential):
    def
    init__(self, dim, hidden_dim, dropout=0.):
        super().__init__(
            nn.Linear(dim, hidden_dim),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, dim),
            nn.Dropout(dropout))
```

Finally Residual Connection are used at different places in the Transformer, these connections are mainly to improve the flow of information to avoid vanishing gradients.

This is residual block:



```

class ResidualAdd(nn.Module):
    def __init__(self, fn):
        super().__init__()
        self.fn = fn

    def forward(self, x, **kwargs):
        res = x
        x = self.fn(x, **kwargs)
        x += res
        return x

```

This block works by specifying a function and then this function is wrapped and you can add the inputs before and after the function and concatenate them.

Finally, we can block all these components together to get the final version of the ViT.

During the experiments, we noticed that after training the model, it was not giving good results because we were training the model from scratch, and when you train the model from scratch you need lots of data and you need to train the model for more epochs in order to get good results. For that it is always the best practice to use the pre-trained model and then fine-tune that model for your custom requirement, you will only need to modify the classification head of the pre-trained model and work with your custom dataset.

5. CONCLUSION

In conclusion, this paper serves as a bridge between theory and practice for Vision transformers, highlighting the importance of understanding the theoretical background of vision transformer and their different concepts specifically the Multi-head self-Attention mechanism; we also highlighted a concise guide on to how to implement the different blocks of the architecture of Vision Transformers architecture. Overall, this paper presents practical insights for researchers and practitioners to discover the ViT and exploit it in their projects.

References

- [1] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” arXiv, Jun. 03, 2021. Accessed: May 22, 2022. [Online]. Available: <http://arxiv.org/abs/2010.11929>
- [2] A. Vaswani *et al.*, “Attention Is All You Need.” arXiv, Dec. 05, 2017. Accessed: May 22, 2022. [Online]. Available: <http://arxiv.org/abs/1706.03762>
- [3] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks.” arXiv, Dec. 02, 2015. Accessed: May 22, 2022. [Online]. Available: <http://arxiv.org/abs/1511.08458>
- [4] J. Y.-Y. Lin, S.-M. Liao, H.-J. Huang, W.-T. Kuo, and O. H.-M. Ou, “Galaxy Morphological Classification with Efficient Vision Transformer.” arXiv, Feb. 03, 2022. Accessed: Jun. 30, 2022. [Online]. Available: <http://arxiv.org/abs/2110.01024>
- [5] H. Chen *et al.*, “GasHis-Transformer: A Multi-scale Visual Transformer Approach for Gastric Histopathology Image Classification.” arXiv, Feb. 17, 2022. Accessed: Jun. 01, 2022. [Online]. Available: <http://arxiv.org/abs/2104.14528>
- [6] Y. Lee and P. Kang, “AnoViT: Unsupervised Anomaly Detection and Localization With Vision Transformer-Based Encoder-Decoder,” *IEEE Access*, vol. 10, pp. 46717–46724, 2022, doi: 10.1109/ACCESS.2022.3171559.
- [7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers.” arXiv, May 28, 2020. Accessed: May 18, 2022. [Online]. Available: <http://arxiv.org/abs/2005.12872>
- [8] S. Zheng *et al.*, “Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers.” arXiv, Jul. 25, 2021. Accessed: May 18, 2022. [Online]. Available: <http://arxiv.org/abs/2012.15840>
- [9] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, “Segmenter: Transformer for Semantic Segmentation.” arXiv, Sep. 02, 2021. Accessed: May 05, 2023. [Online]. Available: <http://arxiv.org/abs/2105.05633>
- [10] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs).” arXiv, Jul. 08, 2020. Accessed: Feb. 19, 2023. [Online]. Available: <http://arxiv.org/abs/1606.08415>

Declarations

Competing Interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declaration of AI Use: Not Applicable.

Funding Statement: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement: Data will be made available on request from the corresponding author.

Author Contribution Statement: Both Authors Contributed Equally.



Author(s) Bio / Authors' Note

Abdelhafid Berroukham is from the Faculty of Science, Ibne Tofail University, Kenitra, Morocco. Email: a.berroukham@gmail.com

Mohammed Lahraichi is from the Faculty of Science, Ibne Tofail University, Kenitra, Morocco. Email: lahraichi.mohamed@gmail.com

