TinyViT: A Small Vision Transformer

Lucian Dorin Crainic
Department of Computer Science
crainic.1938430@studenti.uniroma1.it

La Sapienza University of Rome

Abstract

Vision Transformers (ViTs) excel in image recognition but require large datasets and complex architectures. We introduce TinyViT, a minimalist ViT that retains core components patch embedding and attention while drastically reducing scale. By simplifying token processing and prioritizing parameter efficiency, we test whether a tiny ViT can achieve good accuracy despite limited complexity. Experiments on standard benchmarks show TinyViT delivers acceptable results even with reduced data requirements. This work demonstrates that minimalist transformer architectures can learn meaningful representations, offering a pathway to simpler, more accessible models without sacrificing core functionality.

1 Introduction

Transformers, introduced by [Vaswani, 2017] for natural language processing (NLP), have become the dominant architecture for sequence modeling due to their scalability and self-attention mechanisms. Inspired by their success in NLP, [Alexey, 2020] pioneered the Vision Transformer (ViT), demonstrating that transformers can achieve state-of-the-art results in image recognition by treating images as sequences of patch tokens. By splitting an image into fixed-size patches, linearly embedding them, and processing the sequence with a standard transformer encoder, ViT outperformed convolutional neural networks (CNNs) [He et al., 2016] on large-scale datasets like ImageNet when pretrained on massive datasets (e.g., JFT-300M). However, ViT's strong performance comes at a cost: it requires extensive computational resources and large pretraining datasets, raising practical barriers for adoption in settings where such infrastructure is unavailable.

In this work, we aim to (1) introduce the foundational mechanics of Vision Transformers and (2) present TinyViT, a minimalist implementation designed to test the viability of ViTs in simplified, resource-efficient settings. Unlike the original ViT, which emphasizes scaling to massive datasets, TinyViT reduces architectural complexity employing fewer transformer layers, smaller embedding dimensions while retaining the core principles of patch-based processing and self-attention. We evaluate TinyViT on widely adopted benchmarks like CIFAR-10 and CIFAR-100 [Krizhevsky et al., 2009], and STL-10 [Coates et al., 2011].

2 Architecture

In recent years, the Transformer architecture originally developed for natural language processing has been successfully applied to computer vision tasks. The Vision Transformer (ViT) rethinks image classification by treating

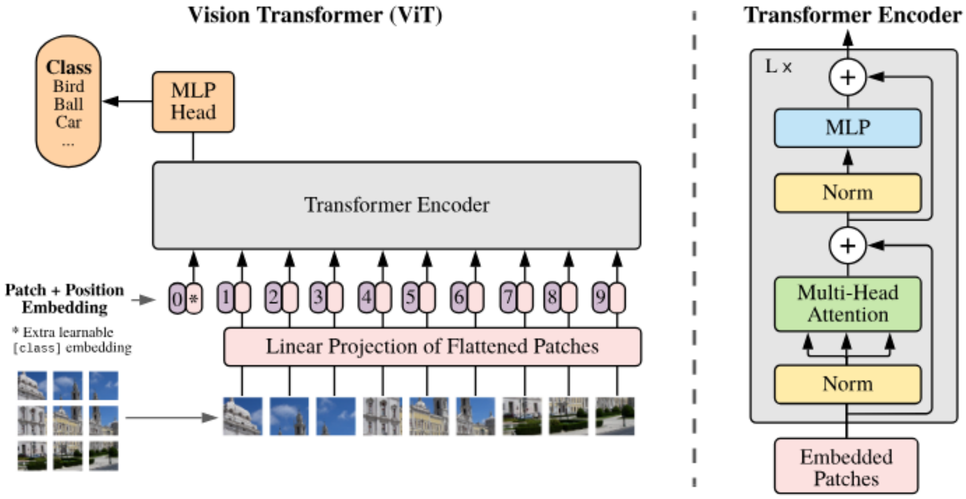

Figure 1: The Vision Transformer (ViT) architecture from [Alexey, 2020]. On the left the model architecture while on the right the Transformer encoder block architecture from [Vaswani, 2017].

an image as a sequence of patches, much like tokens in a sentence, and processes them using standard Transformer blocks. In this chapter, we detail the architecture of the Vision Transformer and provide the mathematical foundations underlying its design.

2.1 Overview

The Vision Transformer (ViT) adapts the Transformer architecture originally developed for natural language processing to image classification tasks. Instead of processing an image as a whole, ViT divides it into a sequence of patches (analogous to tokens) and processes these patches through a stack of Transformer encoder blocks. The key steps in the ViT pipeline are:

1. **Patch Embedding**: The input image is divided into fixed-size patches that are flattened and projected into a latent space.

2. **Positional Encoding**: Since Transformers are permutation-invariant, positional embeddings are added to the patch embeddings.

3. **Transformer Encoder**: A series of Transformer encoder blocks processes the sequence of embeddings using self-attention and feed-forward networks.

4. **Classification Head**: The encoder output is used by a classification head for the final prediction.

2.2 Patch Embedding

Patch Extraction

Given an input image

$$\mathbf{X} \in \mathbb{R}^{H \times W \times C}, \tag{1}$$

where H and W denote the height and width of the image, and C denotes the number of channels, the image is divided into N patches of size $P \times P$. Hence, the number of patches is:

$$N = \frac{HW}{P^2}. \tag{2}$$

Flattening and Linear Projection

Each patch, denoted as $\mathbf{x}_p \in \mathbb{R}^{P \times P \times C}$, is flattened into a vector of dimension $P^2 \cdot C$. A learnable linear projection (implemented as a fully connected layer) maps this vector into a D-dimensional embedding space:

$$\mathbf{z}_p = \mathbf{W}_e \mathbf{x}_p + \mathbf{b}_e \tag{3}$$

where $\mathbf{W}_e \in \mathbb{R}^{D\times(P^2\cdot C)}$ and $\mathbf{b}_e \in \mathbb{R}^D$.

After processing all patches, we obtain a sequence of embeddings:

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N] \in \mathbb{R}^{N\times D}. \tag{4}$$

2.3 Positional Encoding

Transformers do not inherently capture the order or spatial structure of input data. To inject spatial information, learnable positional embeddings are added to the patch embeddings. Let:

$$\mathbf{E}_{\text{pos}} \in \mathbb{R}^{N\times D} \tag{5}$$

be a set of positional embeddings. The combined input to the Transformer encoder is then:

$$\mathbf{Z}_0 = \mathbf{Z} + \mathbf{E}_{\text{pos}}. \tag{6}$$

In some implementations, a special class token $\mathbf{z}_{\text{cls}} \in \mathbb{R}^D$ is prepended to the sequence:

$$\mathbf{Z}_0 = [\mathbf{z}_{\text{cls}}, \mathbf{z}_1, \ldots, \mathbf{z}_N]. \tag{7}$$

2.4 Transformer Encoder

The Vision Transformer uses a stack of L Transformer encoder blocks. Each block is composed of two main components: the Multi-Head Self-Attention (MHSA) mechanism and a Feed-Forward Network (FFN). Layer normalization (LN) and residual connections are applied around each component.

Multi-Head Self-Attention (MHSA)

Let $\mathbf{Z}^{(l-1)} \in \mathbb{R}^{M\times D}$ be the input to the l-th encoder block, where M is the number of tokens (including the class token if used).

Query, Key, and Value

The input is projected into query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) matrices using learnable projection matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{D\times D}$:

- $\mathbf{Q} = \mathbf{Z}^{(l-1)}\mathbf{W}_Q$
- $\mathbf{K} = \mathbf{Z}^{(l-1)}\mathbf{W}_K$
- $\mathbf{V} = \mathbf{Z}^{(l-1)}\mathbf{W}_V$

For multi-head attention, these are divided into h heads. For head i ($i = 1, \ldots, h$), we have:

$$\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{M\times d}, \quad \text{with } d = \frac{D}{h}. \tag{8}$$

Scaled Dot-Product Attention

For head i, the attention is computed as:

$$\text{Attention}_i(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d}}\right)\mathbf{V}_i. \tag{9}$$

The scaling by $\sqrt{d}$ ensures numerical stability.

Concatenation and Final Projection

The outputs from all h heads are concatenated and projected:

$$\text{MHSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\text{Att}_1, \ldots, \text{Att}_h\right)\mathbf{W}_O \tag{10}$$

with $\mathbf{W}_O \in \mathbb{R}^{D\times D}$ being a learned projection matrix.

Residual Connection and Normalization

A residual connection and layer normalization are applied:

$$\mathbf{Z}'^{(l)} = \text{LN}\left(\mathbf{Z}^{(l-1)} + \text{MHSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})\right). \tag{11}$$

Feed-Forward Network (FFN)

Each encoder block contains a two-layer FFN applied to each token independently:

$$\text{FFN}(\mathbf{z}) = \text{GELU}\left(\mathbf{z}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2, \tag{12}$$

where:

- $\mathbf{W}_1 \in \mathbb{R}^{D\times D_{\text{ff}}}$,
- $\mathbf{W}_2 \in \mathbb{R}^{D_{\text{ff}}\times D}$,
- D_{ff} is the dimension of the hidden layer,
- GELU is the Gaussian Error Linear Unit activation.

A residual connection and layer normalization follow:

$$\mathbf{Z}^{(l)} = \text{LN}\left(\mathbf{Z}'^{(l)} + \text{FFN}(\mathbf{Z}'^{(l)})\right). \tag{13}$$

2.5 Classification Head

For classification tasks, the output corresponding to the class token is used. For instance, if a class token $\mathbf{z}_{\text{cls}}$ is used, the final prediction is obtained as:

$$\hat{y} = \mathrm{softmax}\left(\mathbf{W}_c \mathbf{z}_{\mathrm{cls}} + \mathbf{b}_c\right), \qquad (14)$$

where $\mathbf{W}_c \in \mathbb{R}^{K \times D}$ and $\mathbf{b}_c \in \mathbb{R}^K$ are the weights and bias of the classification layer, and K is the number of classes. Alternatively, a global average pooling can be applied to the token embeddings before the linear projection.

3 Differences from Convolutional Neural Networks

The differences between Convolutional Neural Networks (CNNs) architecture in Figure 2 and Vision Transformers (ViTs) come from their architectural philosophies and how they process visual data. CNNs rely on convolutional layers with localized receptive fields, using spatial hierarchies (kernels and pooling) to progressively extract features. These layers inherently prioritize local spatial relationships and translation invariance, making CNNs data efficient for smaller datasets. In contrast, Vision Transformers treat images as sequences of flattened patches, leveraging self attention mechanisms to model global interactions between all patches from the first layer. This allows ViTs to capture long range dependencies directly but often requires large scale training data to generalize effectively, as they lack the inductive biases (locality) inherent to CNNs. Additionally, ViTs replace spatial hierarchies with a uniform processing of patch tokens across all layers, while CNNs gradually reduce spatial resolution while expanding channel depth.

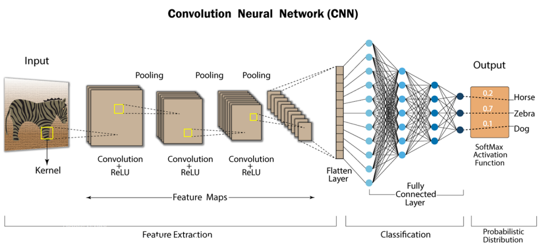

Figure 2: A typical Convolutional Neural Network (CNN) architecture with convolutional and pooling layers.

4 Tiny Vision Transformer (Tiny ViT)

The Vision Transformer (ViT) has proven to be highly effective for image classification by leveraging self-attention mechanisms instead of conventional convolutional operations. Since the original Vision Transformer requires large datasets and significant computational power, TinyViT is designed specifically for smaller datasets and demands less computation. To address this, a compact variant, Tiny Vision Transformer (TinyViT), has been implemented with modifications aimed at efficiency. TinyViT is designed to operate efficiently on smaller images such as 32x32 but can also be adapted for other resolutions like 96x96, dividing them into 4x4 patches, resulting in 64 tokens per image. These patches are projected into an embedding space of 128 dimensions, which serve as input to the Transformer encoder. The model consists of six Transformer layers with eight attention heads each, along with feed-forward networks featuring a hidden dimension of 512. These design choices allow for a balance between model capacity and computational feasibility, leading to approximately 1.21 million trainable parameters.

Compared to standard ViT models, which process high-resolution images with larger patch sizes and higher embedding dimensions, TinyViT scales down key components while maintaining self-attention's effectiveness. Traditional ViT architectures often employ embedding dimensions over 768 and deeper networks, making them impractical for CIFAR-10. By reducing the embedding dimension, the number of Transformer layers, and the patch size, TinyViT significantly lowers computational requirements while retaining essential representation learning capabilities. This makes it a more accessible and efficient solution for limited-resource environments.

The motivation behind TinyViT is to leverage the advantages of ViT while ensuring feasibility for lower-resolution datasets. Given CIFAR-10's 32x32 images, a full-scale ViT model would be excessive. By optimizing model depth, embedding dimensions, and MLP hidden size, TinyViT ensures efficient training without compromising performance. The reduced number of parameters makes it compatible with standard GPUs and avoids excessive memory consumption. Ultimately, this lightweight model demonstrates the viability of self-attention mechanisms in small-scale applications, maintaining the benefits of Transformer-based architectures while being resource-efficient.

Table 1: Parameters of the TinyViT Model for CIFAR-10

Parameter	**Value**
Number of Classes	10
Embedding Dimension	128
Image Size	32
Patch Size	4
Input Channels	3
Number of Attention Heads	8
Number of Transformer Layers	6
MLP Hidden Dimension	512
Number of Patches	64
Total Trainable Parameters	~1.21M

5 Datasets

The experiments in this work utilize three widely recognized image classification benchmarks: CIFAR-10, CIFAR-100, and STL-10. The CIFAR-10 dataset contains 60,000 RGB 32x32 images divided into 10 classes, with 6,000 images per class, offering a balanced benchmark for evaluating basic recognition capabilities. CIFAR-100 shares the same total number of images as CIFAR-10 but includes 100 fine-grained classes, each represented by 600 images, thereby introducing greater categorization complexity. STL-10 consists of 130,000 higher-resolution 96x96 RGB images across 10 classes, featuring a predefined split of 5,000 labeled training images and 8,000 test images, along with additional unlabeled data for semi-supervised learning. These datasets collectively provide a multi-scale evaluation framework, testing the TinyViT architecture across varying class granularities, data volumes, and image resolutions.

Figure 3: CIFAR-10 dataset samples images and classes [Krizhevsky et al., 2009].

The experiments uses standard train-test splits for all datasets: CIFAR-10 and CIFAR-100 each include 50,000 training and 10,000 test images, while STL-10 uses 5,000 labeled training images and 8,000 test samples. For CIFAR-10 and CIFAR-100, training data undergoes augmentation via random cropping (32x32 with 4-pixel padding) and horizontal flipping, followed by normalization using channel-wise means (0.4914, 0.4822, 0.4465) and standard deviations (0.2470, 0.2435, 0.2616). Test images are directly normalized without augmentation. STL-10, with its higher 96x96 resolution, employs more extensive training augmentations: 12-padded random cropping, horizontal flipping, 15-degree rotations, color jittering (brightness, contrast, saturation), and random grayscaling (10% probability), normalized with dataset-specific statistics (means: 0.4467, 0.4398, 0.4066; stds: 0.2603, 0.2566, 0.2713). Test sets for all datasets apply only resizing, tensor conversion, and identical normalization to ensure evaluation consistency. These transformations balance generalization during training and standardization during testing.

6 Results

In this chapter, the results of the experiments conducted are presented. The chapter discusses the evaluation metrics used to measure the performance of the model and the results obtained from the experiments.

Table 2: Performance comparison between Tiny ViT and CNN models across different datasets

Dataset	Model	Accuracy	F1	Recall	MCC	Precision
CIFAR-10	**Tiny ViT**	**82.59**	**82.43**	**82.59**	**80.70**	**82.76**
	CNN	80.67	80.55	80.67	78.53	80.56
CIFAR-100	**Tiny ViT**	**59.40**	**59.04**	**59.40**	**59.00**	**60.00**
	CNN	46.13	44.57	46.13	45.61	45.61
STL-10	Tiny ViT	64.27	64.30	64.27	60.47	66.13
	CNN	**68.36**	**68.10**	**68.36**	**64.95**	**68.78**

Note: All values are percentages (%). Bold indicates best performance in category.

6.1 Evaluation Metrics

These metrics will be used to show the effectiveness of the approaches proposed.

Accuracy score

Accuracy is a proportion of correct predictions, considering both true positives and true negatives, among a total number of samples. The formula used to calculate accuracy is the following:

$$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{15}$$

where **TP** is a number of true positives, **TN** is a number of true negatives, **FP** is a number of false positives and **FN** is a number of false negatives.

Precision score

Precision is the ability of a classifier not to label as positive a sample that is negative. The formula used to calculate precision is the following:

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{16}$$

Recall score

Recall is the ability of a classifier to find all the positive samples. The formula used to calculate the recall is the following:

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{17}$$

F1 score

F1 score is a harmonic mean of precision and recall. The formula used to calculate the F1 score is the following:

$$\frac{2\times(\text{precision}\times\text{recall})}{\text{precision}+\text{recall}} \tag{18}$$

Matthews correlation coefficent

Matthews correlation coefficent (or φ coefficient) takes into account true and false positives and negatives and is regarded as a balanced measure that can be used even if the classes are of very different sizes. Formula used to calculate φ coefficient is as follows:

$$\frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{19}$$

6.2 Results

All models were trained for 100 epochs using a batch size of 64, with an AdamW optimizer configured with a learning rate of 3e-4 and a weight decay of 0.05. The loss function used was cross-entropy loss with label smoothing of 0.1.

Table 2 presents the performance comparison between the Tiny ViT and CNN models across different datasets. The following comments can be made based on the results:

1. On **CIFAR-10**, Tiny ViT slightly outperformed CNN, achieving 82.59% accuracy vs. 80.67%. It also had a higher F1 score, recall, and MCC, indicating better generalization. The precision advantage (82.76 vs. 80.56) suggests Tiny ViT made fewer false positives.

2. For **CIFAR-100**, Tiny ViT performed significantly better (59.40% accuracy vs. 46.13% for CNN). The gap was consistent across F1 score, recall, and MCC, indicating Tiny ViT handled the increased class complexity better. A notable precision boost (60.00 vs. 45.61) further sup-

ports its robustness in fine-grained classification.

3. On **STL-10**, CNN outperformed Tiny ViT (68.36% accuracy vs. 64.27%). The CNN model had better F1, recall, and MCC, though Tiny ViT had higher precision. The larger image size in STL-10 might explain why CNN performed better, as its hierarchical feature extraction could be more effective than Tiny ViT's attention-based mechanism.

Overall, Tiny ViT showed clear advantages in CIFAR-10 and CIFAR-100, particularly in handling complex class distributions, while CNN performed better on STL-10, possibly due to its ability to capture local features in higher-resolution images.

7 Conclusion and Future Work

In this chapter a conclusion is drawn from the results obtained in the previous chapter. The chapter also discusses the future work that can be done to improve the model.

7.1 Conclusion

In conclusion, the implementation of the Tiny ViT architecture has shown promising results, often performing on par with, and occasionally surpassing, standard Convolutional Neural Networks (CNNs) in image recognition tasks. This achievement underscores the potential of Vision Transformers (ViTs) as a robust alternative to the well-established CNNs, offering exploration beyond conventional CNN implementations. While we did not achieve the performance levels of the standard ViT architecture, primarily due to the absence of large-scale datasets for pre-training and subsequent transfer learning, we successfully demonstrated that a scaled-down ViT architecture can obtain good results without the need for massive datasets.

7.2 Future Work

Future work to evaluate the performance of the proposed tiny Vision Transformer (ViT) architecture on two key computer vision tasks: object detection and semantic segmentation. Specifically, the tiny ViT could be implemented forr frameworks like DETR [Carion et al., 2020] for object detection and Segmenter [Strudel et al., 2021] for segmentation, enabling a direct comparison with simpler CNN-based architectures traditionally used for these tasks. Additionally, the Tiny ViT architecture could be further optimized by exploring different hyperparameters, such as the number of layers, hidden units, and attention heads, to improve its performance.

References

[Alexey, 2020] Alexey, D. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929.*

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

[Coates et al., 2011] Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[Strudel et al., 2021] Strudel, R., Garcia, R., Laptev, I., and Schmid, C. (2021). Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272.

[Vaswani, 2017] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.