Master's thesis

Master's Programme in Computer Science

Scaling Vision Transformers: Computational Challenges and Efficiency Improvements

Joni Luhtalampi

May 22, 2025

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Joni Luhtalampi

Työn nimi — Arbetets titel — Title

Scaling Vision Transformers: Computational Challenges and Efficiency Improvements

Ohjaajat — Handledare — Supervisors

Prof. Laura Ruotsalainen, Second Examiner Dr. Linsen Gao

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	May 22, 2025	64 pages

Tiivistelmä — Referat — Abstract

Vision Transformers have proven to be a powerful alternative to Convolutional Neural Networks in computer vision applications, demonstrating enhanced performance in image classification, object detection, and segmentation. However, they face scalability challenges due to high computational complexity, memory constraints, and reliance on large datasets. This thesis explores these limitations and presents optimization strategies that have been developed to improve efficiency. Key challenges include the quadratic cost of self-attention, high memory and hardware demands, and the need for extensive training data, especially during the training phase. To address these, we explore data-efficient training methods, token reduction techniques, and architectural developments such as hierarchical models. These techniques demonstrate that optimized models can significantly reduce computational overhead while maintaining their accuracy. Such optimizations also enable their deployment in resource-constrained environments and increase their applicability to real-world applications.

ACM Computing Classification System (CCS)
Computing methodologies → Artificial intelligence → Computer vision → Computer vision problems → Object detection
Computing methodologies → Machine learning → Machine learning approaches → Neural networks
Computer systems organization → Real-time systems → Real-time system architecture

Avainsanat — Nyckelord — Keywords

vision transformers, scalability, computational complexity, optimization, architecture

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Algorithms study track

Contents

1 Introduction

The field of machine learning has witnessed remarkable advancements in the past decade, particularly in the domains of natural language processing (NLP) and computer vision. Everyday users experienced this revolution in practice when OpenAI released ChatGPT (OpenAI, 2022) to public use on November 30, 2022. This marked a groundbreaking shift, as publicly available NLP models made a significant leap forward in their ability to understand and generate text. It was able to not only understand the text but also produce meaningful answers to the user's input.

This groundbreaking change was enabled by the introduction of a new paradigm, known as the *Transformer*, to process sequential data. The Transformer model was introduced in 2017 in the groundbreaking paper *"Attention is All You Need"* (Vaswani et al., 2017), which proposed a model architecture that utilizes a mechanism called 'self-attention' to weigh the importance of different words in a sentence relative to each other, allowing it to capture dependencies regardless of their distance in the sequence. This marked a significant departure from traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which were traditionally used for NLP tasks.

Following this innovation, a series of powerful Transformer-based models such as BERT (Devlin et al., 2019), GPT (Radford et al., 2018), and T5 (Raffel et al., 2020) further advanced the field, demonstrating the scalability, flexibility, and transferability of the Transformer architecture across a wide range of tasks. These developments firmly established the Transformer as the backbone of modern deep learning in sequential data domains.

The success of Transformers in NLP naturally led to their application in other domains as well, including computer vision. Traditionally, CNNs and RNNs have dominated computer vision tasks such as image classification, object detection, and segmentation. CNNs are particularly effective in extracting spatial hierarchies from images (Lecun et al., 1998; He et al., 2016), but are limited in their ability to capture global context. Inspired by the success of transformers in NLP, researchers began to explore the possibility of applying Transformer-based models to vision tasks, leading to the development of *Vision Transformer* (ViT) (Dosovitskiy et al., 2021).

The seminal work *"An Image Is Worth 16x16 Words: Transformers for Image Recognition*

at Scale" (Dosovitskiy et al., 2021), marked a significant milestone in the application of transformers to computer vision. This paper proposed to adapt the architecture of the NLP transformer to process images. Instead of processing sentences as sequences of words, the ViT processes sequences of vectors, which are flattened from the input images by using fixed side patches. The ViT architecture allows the model to capture long-range dependencies across different parts of the image, enabling it to understand complex visual patterns without the need for convolutional layers.

The ViT model demonstrated that transformers could achieve competitive and sometimes superior performance compared to state-of-the-art image classification benchmarks. However, early comparative studies Bhojanapalli et al., 2021 confirmed that while ViTs can outperform CNNs under certain conditions, this typically requires large amounts of training data and substantial computational resources. Consequently, follow-up research quickly recognized several limitations of the original ViT design, particularly with regard to data efficiency, scalability, and computational costs. For example, Touvron, Cord, Douze, et al. (2021) introduced the *Data-efficient Image Transformer* (DeiT), showing that Vision Transformers could be trained effectively even on relatively smaller datasets through the use of knowledge distillation. Similarly, Liu et al. (2021) proposed the *Swin Transformer*, which incorporated hierarchical structure and shifted window attention to improve computational efficiency for inputs with higher resolution.

The introduction of transformer models has fundamentally changed the landscape of both NLP and computer vision. In particular, ViT represents a significant step forward in the application of attention mechanisms to visual data, offering new possibilities for how machines observe and process images. As research in this area continues to evolve, it holds the promise of unlocking new levels of performance and understanding in a wide range of computer vision applications such as image classification, object detection, and image segmentation.

Although ViTs have shown great promise, they also present several challenges (Dosovitskiy et al., 2021), including the need for large-scale datasets for training, the high computational complexity, and considerable memory requirements. These challenges are particularly critical in real-time applications, such as autonomous driving, medical imaging, and high-definition image classification, where scalability becomes an essential factor. For example, autonomous driving systems require low latency and high accuracy to ensure safety. Similarly, high-definition image classification requires powerful processing capabilities to quickly analyze large amounts of high-resolution data. High-performance hardware

often comes at a high cost and can be too large for compact applications, creating both financial and design challenges. These applications highlight the importance of optimizing Vision Transformers to ensure their suitability for real-time deployment while maintaining high prediction accuracy.

This thesis aims to explore the scalability challenges and optimization strategies for Vision Transformers across a variety of computer vision applications. More precisely, it aims to address the following research questions.

1. *What are the scalability limitations of Vision Transformers in real-time applications, such as high-resolution image classification and autonomous driving?*

   This question focuses on identifying the specific constraints that impact the use of ViTs in time-sensitive, high-input environments. By examining these constraints, the thesis aims to provide a comprehensive understanding of the conditions under which Vision Transformers can be utilized effectively, as well as the adjustments required to make them feasible for such applications.

2. *How can Vision Transformer architectures be optimized to reduce computational costs while maintaining high accuracy in large-scale image recognition tasks?*

   When ViTs are deployed at scale, minimizing computational overhead becomes crucial for practical usage, particularly when dealing with large datasets, high-resolution images, or hardware limitations. This research question investigates various optimization strategies to balance the trade-off between computational efficiency and model accuracy.

The thesis begins by providing essential background knowledge on the Transformer model, starting with its adaptation to NLP and concluding with the modifications required for its application in machine vision. To maintain a manageable scope, the thesis assumes that the reader has a basic understanding of deep learning concepts, particularly neural networks. The book *Deep learning* (Goodfellow et al., 2016) is a recommended reference for any knowledge gaps. Once the basic principles of ViT are introduced, the thesis explores the scalability limitations of the original ViT model (Dosovitskiy et al., 2021). Subsequently, it examines various enhancements and optimizations that have been introduced to improve its performance in different vision tasks, addressing challenges such as computational efficiency, dependence on extensive datasets, and architecture refinements. Finally, the thesis ends with conclusions.

After reading this thesis, the reader should have a clear understanding of the major scalability limitations of ViTs, the motivations behind various proposed enhancements, and the trade-offs involved in optimizing ViTs for real-world deployment.

2 Vision Transformer

In 2017, eight scientists working at Google published a landmark research paper, "Attention is All You Need" (Vaswani et al., 2017), which has revolutionized natural language processing by introducing a new paradigm for processing sequential data. The paper introduced a new deep learning architecture, known as the Transformer, which was inspired by and builds on the foundational concept of attention introduced in a paper that had been released three years earlier (Bahdanau et al., 2015). Although the attention mechanism was already introduced earlier, Vaswani et al. (2017) significantly extended and generalized the concept. They took the idea of attention further by completely removing the recurrence and relying entirely on the attention mechanism.

This chapter goes through the fundamental concepts and architecture of transformers, providing the necessary background knowledge to understand their application in computer vision. We will begin by going through the core principles of the transformer model introduced by Vaswani et al. (2017). After that, the chapter will continue to its adaptation to computer vision, mainly focusing on the concepts introduced in a landmark research paper called "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale" (Dosovitskiy et al., 2021). By exploring the core principles of transformers, this chapter sets the stage for a deeper examination of their role in vision tasks, laying the groundwork for the subsequent discussions and analyses in this thesis.

2.1 Traditional Transformer model

The Transformer model is built on a flexible and parallelized architecture, which fundamentally relies on an encoder-decoder structure. Figure 2.1 shows an overview of the model structure. It uses stacked self-attention and fully connected points-wise layers for both the encoder and the decoder, which is shown in the left and right halves of the figure 2.1. Unlike previous sequence models (Goodfellow et al., 2016) such as recurrent neural networks (RNNs) and long-short-term memory networks (LSTMs), the Transformer processes the input data in parallel rather than sequentially, allowing greater computational efficiency and scalability.

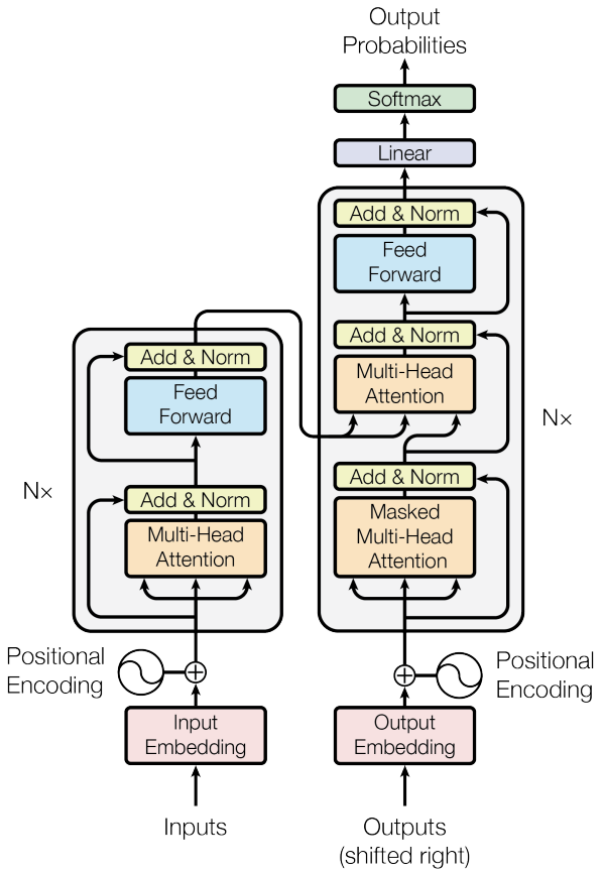

Figure 2.1: The Transformer architecture (Vaswani et al., 2017). On the left the encoder stack and on the right the decoder stack. Both stacks are composed of N identical layers which are depicted in the figure.

2.1.1 Encoder and Decoder Stacks

The Transformer model architecture can be divided into two main components: an encoder and a decoder. These components are composed of multiple layers or "stacks," where each stack is an identical layer that applies transformations to the input and output (Figure 2.1). The encoder-decoder architecture enables its powerful sequence-to-sequence learning capability. It allows the transformer model to take text as an input and to process meaningful text as an output. The number of layers depends on the specific transformer model, but in general additional layers enable the model to gain a more nuanced understanding of the input data. Each layer in the stack builds on the outputs of the previous layer, allowing the model to have a better understanding of the relationships between elements in the input or output sequences.

Although Transformers popularized the encoder-decoder architecture, the foundational

concept originates from sequence modeling, such as in sequence-to-sequence models for machine translation (Sutskever et al., 2014; Goodfellow et al., 2016). Transformers refined this idea by replacing recurrent operations with parallel attention mechanisms.

Encoder

The encoder is responsible for reading and encoding the input sequence into a series of continuous representations. It contains N identical layers and each of them has two sub-layers:

1. multi-head self-attention mechanism,
2. position-wise fully connected feed-forward network.

Around each of the two sub-layers residual connections (He et al., 2016) are used and are followed by layer normalization (Ba et al., 2016). The encoder processes the entire input sequence simultaneously, using the self-attention mechanism to focus on different parts of the sequence. Vaswani et al. (2017) used $N = 6$, but popular transformer models such as BERT (Devlin et al., 2019), GPT (Radford et al., 2018), and T5 (Raffel et al., 2020) have even more layers.

Decoder

The decoder, similar to the encoder, consists of a stack of identical layers. Vaswani et al. (2017) used $N = 6$. However, unlike the encoder, each decoder layer has an additional sub-layer: a multi-head attention mechanism that focuses on the output of the encoder stack. Similarly to the encoder, residual connections are used around each of the sub-layers and are followed by layer normalization.

One key difference in the decoder's self-attention sub-layer is the addition of a masking mechanism, which ensures that the predictions for position i can only depend on the known outputs at positions less than i. The *position* represents the specific place where a word or part of it is held in the text. This masking prevents a position from "seeing" future positions in the text, which means that the prediction at each position is based only on past and current words.

2.1.2 Input embeddings

In the Transformer model, the first step in processing input data is converting the raw tokens, such as words and sub-words in NLP, into continuous vector representations known as input embeddings. It is crucial to translate discrete input tokens into high-dimensional continuous vectors that the model can effectively process. We can think of embeddings as a way to translate words into a language that computers can understand and process. We need embeddings because in practice computers understand numbers, not text.

The very first step in input embeddings is called *tokenization*. It is a process in which each word is translated into a structured form that models can process. In NLP, there are three common types of tokenization:

1. *Word-level tokenization* is the simplest form of tokenization, where input text is split into individual words based on spaces or punctuation. For example, the sentence *"This is an input."* would be tokenized as follows.

   $$[\text{"}This\text{"}, \text{"}is\text{"}, \text{"}an\text{"}, \text{"}input\text{"}, \text{"}.\text{"}]$$

   Word-level tokenization is straightforward, but it struggles with rare or unseen words with fixed-size vocabulary and fails to capture the structure of words with common roots or affixes.

2. *Character-level tokenization* treats each character in the text as a separate token. For example, the word *"Input"* would be tokenized as follows.

   $$[\text{"}I\text{"}, \text{"}n\text{"}, \text{"}p\text{"}, \text{"}u\text{"}, \text{"}t\text{"}]$$

   This allows the model to work with a much smaller vocabulary but produces longer sequences, which can slow down training and interfere with inferring word connections.

3. *Subword-level tokenization* breaks words into subword units based on frequency and context. This is also known as WordPiece tokenization (Devlin et al., 2019). For example, the word *"Unhappiness"* can be split as follows.

   $$[\text{"}Un\text{"}, \text{"}\#\#happiness\text{"}]$$

   This method allows rare or complex words to be divided into smaller and more common units, which helps the model to generalize new words previously unseen.

Subword-level tokenization offers a good balance between word-level and character-level tokenization, and it is widely used within common transformer models with various sizes of vocabularies.

After the input sentence is split into tokens with a suitable tokenization type, each token is translated into a numeric token ID. A classification token is also added to the beginning of the sentence, and a separator token is placed at the end to help the model infer the sentence's meaning easier.

The token ID of each word is a constant value, but depends on the selected tokenization type and its vocabulary side. For example, BERT uses WordPiece embeddings with 30k tokens as its vocabulary (Devlin et al., 2019). Thus, token IDs have a range of roughly 0 to 30k.

Finally, each word or subword token is assigned a unique embedding vector, which is learned during training. This process essentially uses only a lookup table, called *embedding layer*, that stores a vector representation for each unique token in the vocabulary. Each token in the input sequence is assigned to its corresponding embedding vector. The embedding layer can be generalized as follows.

For a given vocabulary V, the embedding layer can be represented as a matrix $E \in \mathbb{R}^{|V| \times d}$, where $|V|$ is the size of the vocabulary and d is the embedding dimension. Each row of the matrix E corresponds to the embedding of a specific token. Now, for each token t from the input sequence, its embedding vector $E_t \in \mathbb{R}^d$ is retrieved from the embedding matrix E.

The embedding process serves the following key purposes:

1. **Mapping discrete tokens to continuous space**: Input embeddings convert discrete input tokens into continuous vectors. This enables the machine learning models, which operate in continuous space, to perform computations on the vectors.

2. **Capturing Semantic Meaning**: The input embeddings encode semantic properties of the tokens, allowing the model to learn relationships and similarities between words or sub-words. For example, words with similar semantic meanings may be mapped to vectors that are close to each other in the vector space.

3. **Dimensionality Reduction**: Instead of using a sparse high-dimensional representation like one-hot encoding (Goodfellow et al., 2016), embeddings provide dense low-dimensional representations that capture more information in fewer dimensions.

The dimension of the embedding vectors, denoted earlier as d, is an important hyperparameter that should be chosen carefully. If the embedding dimension is too small, the embeddings might not be able to capture enough semantic information about the input tokens. However, if the dimension is too large, it can increase the computational cost and may lead to overfitting, especially with a limited amount of training data. The original Transformer model by Vaswani et al. (2017) set the embedding dimension to typically 512 or 1024 based on the size of the model, but even the BERT base model has 768 dimensions (Devlin et al., 2019).

Let us use the sentence *"This is an input."* as an example to take a closer look at the input embedding process:

1. At first, the input sentence is tokenized based on the tokenization type. We will use word-level tokenization as an example. The input gets the following form.

   This is an input .

2. Next, the classification token, [CLS], is also added to the beginning of the sentence, and the separator token, [SEP], is placed at the end. After that, the input has the following form.

   [CLS] This is an input . [SEP]

3. Then, each token is translated to its token ID. Each word has its own token ID which is based on the chosen vocabulary size. Now, the input will have the following format.

   101 2023 2003 1037 7953 1012 102

4. Finally, embedding vectors are retrieved from the embedding matrix for each token. The final output of the embedding could look something like the following where the embedding vectors are presented as columns.

$$\begin{bmatrix} -0.18342 & 0.13421 & -0.02873 & 0.07218 & -0.19875 & 0.16743 & -0.10348 \\ 0.14523 & -0.05674 & 0.09231 & -0.11239 & 0.08754 & -0.17894 & 0.03567 \\ -0.04512 & 0.10237 & -0.19283 & 0.07325 & -0.13489 & 0.19876 & -0.02365 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

2.1.3 Positional Encoding

Traditional neural network architectures such as RNNs and CNNs process the input data in a step-by-step manner. Thus, they inherently understand the order in which words appear in the sentence. However, the Transformer model processes words in parallel rather than sequentially, which provides computational efficiency. In exchange, we lose sequential information of the input, and the Transformers does not have any built-in mechanism to preserve the order of input tokens. To address this, Vaswani et al. (2017) introduced positional encoding, which explicitly adds information about the position of each token in the sequence.

The goal of positional encoding is to enable the model to differentiate between different positions of tokens, allowing the Transformer to interpret sequences of the words in the sentence in a meaningful way. Without positional encoding, the model would treat a sentence like *"The cat sat on the mat."* the same as *"The mat sat on the cat."*, because it would not know the order in which the words, or more precisely the tokens, were presented in the input.

In the original transformer implementation (Vaswani et al., 2017), the positional encoding is based on alternating sine and cosine functions of different frequencies, which ensure that each position has a unique representation.

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$
$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

where:

- pos is the position of the token in the sequence.
- i is the dimension index.
- d_{model} is the dimensionality of the input embeddings.

These position embeddings are then added to the input embeddings with the matrix addition operation, before the result is fed into the transformer layer (Figure 2.1).

Vaswani et al. (2017) also experimented with the use of learned positional embeddings, but the two versions produced nearly identical results. Thus, they chose the sinusoidal

version because it may allow the model to extrapolate to sequence lengths longer than those encountered during training. There are also other choices for positional encoding, learned and fixed (Gehring et al., 2017).

Let us take a closer look at how the positional encoding is defined for *"This is an input."*, which we already used as an example for input embeddings.

1. Let us start by recalling what the input embedding for the input looks like:

[CLS]	This	is	an	input	.	$[SEP]$
−0.18342	0.13421	−0.02873	0.07218	−0.19875	0.16743	−0.10348
0.14523	−0.05674	0.09231	−0.11239	0.08754	−0.17894	0.03567
−0.04512	0.10237	−0.19283	0.07325	−0.13489	0.19876	−0.02365
...	...	...	...	...	...	...

2. Next, we define the values for pos and i to perform calculations for positional encodings in the next step. For simplicity, only the first four dimensions on both axes are shown.

[CLS]	This	is	*an*	
$pos = 0$	$pos = 1$	$pos = 2$	$pos = 3$	
$PE_{(0,0)}$	$PE_{(1,0)}$	$PE_{(2,0)}$	$PE_{(3,0)}$	$i = 0$
$PE_{(0,1)}$	$PE_{(1,1)}$	$PE_{(2,1)}$	$PE_{(3,1)}$	$i = 0$
$PE_{(0,2)}$	$PE_{(1,2)}$	$PE_{(2,2)}$	$PE_{(3,2)}$	$i = 1$

   Note how the dimension index i is defined. Based on the positional encoding formula, the index i increases in every two dimensions of the encoding vector. For example $PE_{(pos,2i+1)} = PE_{(0,1)}$, when $i = 0$ and $pos = 0$.

3. Then, we calculate positional encodings with d_{model}. As an example, we will do this only for a few positions:

$$PE_{(1,0)} = PE_{(1,2*0)} = sin\left(\frac{1}{10000^{2*0/512}}\right) = sin(1) \approx 0.84147$$
$$PE_{(2,1)} = PE_{(2,2*0+1)} = cos\left(\frac{2}{10000^{2*0/512}}\right) = cos(2) \approx -0.41615$$
$$PE_{(3,2)} = PE_{(3,2*1)} = sin\left(\frac{3}{10000^{2*1/512}}\right) \approx sin(2.894) \approx 0.2451$$
$$PE_{(0,3)} = PE_{(0,2*1+1)} = cos\left(\frac{0}{10000^{2*1/512}}\right) = cos(0) = 1$$

4. Finally, positional encodings are added to the input embeddings. For simplicity, we will do the math only for the first three dimensions on both axes.

[CLS]	This	is
$-0.18342 + 0.00000$	$0.13421 + 0.84147$	$-0.02873 + 0.90930$
$0.14523 + 1.00000$	$-0.05674 + 0.54030$	$0.09231 - 0.41615$
$-0.04512 + 0.00000$	$0.10237 + 0.76625$	$-0.19283 + 0.98547$

Which equals to:

[CLS]	This	is
-0.18342	0.97568	0.88057
1.14523	0.48356	-0.32384
-0.04512	0.86862	0.79264

2.1.4 Attention Mechanism

Unlike traditional RNNs and CNNs that inherently suffer from limitations in handling long-range dependencies or sequential input, the Transformer model has the ability to capture dependencies between tokens regardless of their distance within the input sequence. This ability is largely attributed to its attention mechanisms. This mechanism can be divided into two key types: to *Self-Attention* and to *Multi-Head Attention*. Let us take a closer look at both, starting with self-attention, which provides a core principle for multi-head attention.

Self-Attention

Self-attention is a core building block of the Transformer architecture. It enables the model to weigh the importance of other tokens in the input sequence to the token that is currently being processed. This mechanism allows each token in the input sequence to interact with other tokens and to assign varying degrees of attention to them based on contextual relevance. It allows each token to focus only on relevant parts of the sequence, capturing both local and long-range dependencies in parallel. Let us take a look at *"The bank of the river was flooded."* sentence as an example. Without contextual relevance provided by the self-attention mechanism, the model would most probably assume that the word *"bank"* refers to the financial institution and not to the geographical part of the river, since the financial institute word occurs much more often in English.

The core element of self-attention is an attention function, which can be described as mapping a query and a set of key-value pairs to an output. It relies on three key vectors for each token: *Query (Q)*, *Key (K)* and *Value (V)* vectors, which are linear transformations of the embeddings of the input token. The primary goal is to compute a weighted sum of the values for each token, where the weights are determined by the similarity between the query of a token and the keys of all other tokens.

Before the introduction of the Transformer model, the two most commonly used attention functions were additive attention and dot-product (multiplicative) attention (Vaswani et al., 2017). While these two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, as it can be implemented using highly optimized matrix multiplication code. Thus, dot-product attention would be an ideal choice for the attention function.

$$Attention(Q, K, V) = softmax(QK^T)V$$

Unfortunately, additive attention outperforms dot-product attention in the larger dimension of query and key vectors. Vaswani et al. (2017) believed that for larger dimensions, the resulting dot products become significantly larger, which can drive the softmax function into areas with very small gradients, potentially hindering effective learning. Thus, they decided to scale the dot products by $\frac{1}{\sqrt{d_k}}$ and came up with the following attention formula:

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

where d_k is the dimension of the key vector K. This particular attention was named as "Scaled Dot-Product Attention" (Figure 2.2).

Let us take a bit closer look at the Scaled Dot-Product Attention step by step:

1. Initially, the embedding of each token is linearly projected onto the Q, K, and V vectors using learned weight matrices. These matrices are learned during the training process, where millions or even billions of sentences are used as training data. Vaswani et al., 2017 used 4.5M and 37M sentence datasets in their original article, but, for example, the original BERT model was already trained with 3.3B sentences (Devlin et al., 2019).

2. Next, the attention score between the tokens i and j is calculated by taking the dot-product of the query vector Q for the token i and the key vector K for the token j.

   $$AttentionScore(Q, K) = QK^T$$

   This measures how much focus the token i should place on the token j. The higher the score, the more emphasis should be placed on the token j. See Figure 2.3 for an example of long-distance dependency of tokens.

3. The attention score is then scaled by the factor $\sqrt{d_k}$, where d_k is the dimension of the key vector. This ensures numerical stability during training, as explained above.

4. Once the raw scaled attention scores are calculated, those are passed through a softmax function to normalize them. The softmax function converts the given input into a probability distribution:

   $$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}},$$

   where z is an input vector, i and j are indices of its elements and n is the length of the vector z.

5. Finally, the output of the previous step is multiplied by the token's value vector. The final representation for each token is the weighted sum of the value vectors, weighted by the token's attention probabilities.

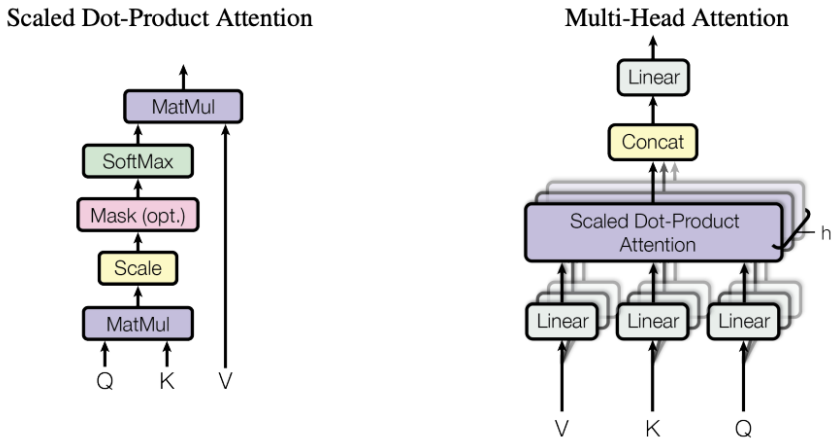

Figure 2.2: Attention mechanism (Vaswani et al., 2017). On the left Self-Attention (Scaled Dot-Product Attention) and on the right Multi-Head Attention consists of several attention layers running in parallel.

Multi-Head Attention

Multi-head attention is an extension of self-attention which enables the model to focus on different aspects of the input sequence simultaneously. Instead of performing a single attention function at the time, Vaswani et al. (2017) found that it is beneficial to project the input into multiple parallel subspaces, by performing the attention function in parallel with multiple 'heads'. This allows the model to capture various aspects of word associations, sentence structure, or dependencies at different granularity levels. Each of the heads computes the attention independently, which enables the model to learn to focus on distinct relationships between the tokens, as visualized in Figure 2.3. Finally, the results of all attention heads are concatenated and again projected linearly, resulting in the final output, as shown in Figure 2.2.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

In their work, Vaswani et al. (2017) employed $h = 8$ parallel attention layers, or heads. For each of them they used $d_k = d_v = d_{model}/h = 64$. Due to the dimensionality reduction

of each head, the computational cost was about the same as it would have been with a single-head attention with full dimensionality.

Let us take a closer look how the multi-head attention operates:

1. At first, the query, key and value vectors are projected into h subspaces with different learned linear projections:

$$Q_i = QW_i^Q, \quad K_i = KW_i^K, \quad V_i = VW_i^V,$$

where W_i^Q, W_i^K and W_i^V are learned projection matrices, $i \in \{1, ..., h\}$ and h is the number of heads.

2. Then the scaled dot-product attention is computed independently for each head:

$$head_i = Attention(Q_i, K_i, V_i) = softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i,$$

where d_k is the dimensionality of the key vectors.

3. Finally, the outputs of all heads are concatenated and linearly transformed:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O,$$

where W^O is the learned weight matrix for linear transformation.

2.1.5 Feed Forward Networks

In addition to the multi-head attention mechanism, each layer of the Transformer's encoder and decoder contains a feed forward network (FFN) (Goodfellow et al., 2016). This network operates independently at each position in the sequence, applying the same linear transformations to all positions. The FFN consists of two linear transformations with a ReLU activation function between. For ReLU see Goodfellow et al. (2016). Mathematically, the operation can be described as follows:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

where:

- W_1 and W_2 are learned weight matrices from layers 1 and 2 respectively,

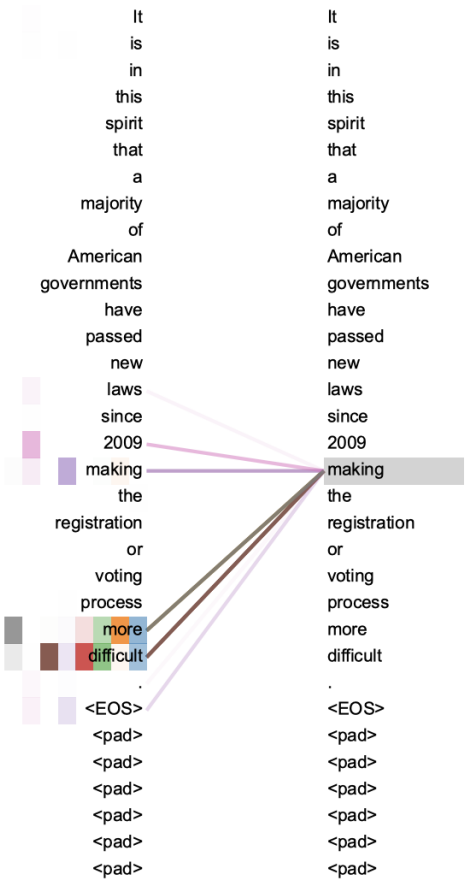

Figure 2.3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6 (Vaswani et al., 2017). Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads.

- b_1 and b_2 are bias terms,
- $max(0, \cdot)$ is ReLU activation function.

This simple two-layer architecture allows the model to learn complex transformations of the input data. While attention layers capture the relationships between tokens, the FFN refines the representations and introduces non-linear transformations, further improving the model's ability to generalize and capture complex patterns.

The dimensionality of the input and output is the same, but the inner (hidden) layer usually has a much larger dimension. For example, in the original Transformer architecture, the input and output have a dimensionality of 512, while the inner layer has a dimensionality of 2048 (Vaswani et al., 2017).

2.1.6 Layer Normalization And Residual Connections

In the Transformer architecture, *layer normalization* (Ba et al., 2016) and *residual connections* (He et al., 2016) are vital to maintaining the stability and efficiency of training deep networks. These techniques, applied to both the encoder and decoder stacks, ensure smooth training and enable the model to learn complex patterns effectively without encountering issues such as vanishing or exploding gradients.

Layer Normalization

Layer normalization (Ba et al., 2016) is a technique in deep learning that is used to stabilize the training process and improve the performance of neural networks. It addresses the internal covariate shift problem, where the distribution of activations within a layer changes during training, making it difficult for the network to learn effectively. Unlike batch normalization (Ioffe and Szegedy, 2015), layer normalization is performed for each training example individually. This means that the mean and variance of the activations are calculated for each layer separately, and then the activations are scaled and shifted to have a standard normal distribution (0 mean and variance of 1). This helps maintain the stability by preventing the model's activations from fluctuating too much.

Mathematically, layer normalization can be described as follows:

$$LayerNorm(x) = \frac{x - \mu}{\sigma} \cdot \gamma + \beta$$

where x is the input, μ is the mean and σ is the variance of the activations in the layer, and γ and β are learnable parameters that scale and shift the normalized output.

Residual Connections

Residual connections (He et al., 2016) are another vital component that helps prevent issues related to gradient degradation in deep neural networks. In the Transformer model, residual connections are used to "skip" each sub-layer by adding the input of the layer directly to its output before applying layer normalization.

For each sub-layer x, this process is described as:

$$Output = LayerNorm(x + Sublayer(x))$$

where $Sublayer(x)$ is the function implemented by the sub-layer itself.

This allows the model to pass directly through the original input, ensuring that the information can flow through the network. If the sub-layer fails to learn, the residual connection ensures that at least the initial information is preserved to the next layer.

2.2 Modifications For Vision Transformer

Traditionally, convolutional neural networks (CNNs) have been the predominant model for vision tasks. However, in 2021 the Vision Transformer (ViT) was introduced in the landmark paper "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale" (Dosovitskiy et al., 2021). This work demonstrated that with appropriate modifications, the Transformer model could achieve competitive and even superior results against various image recognition benchmarks.

The Vision Transformer adapts key components of the Transformer architecture with a few modifications to handle images. This section will go through these modifications that enable Transformers to work effectively for vision tasks without using the traditional CNNs.

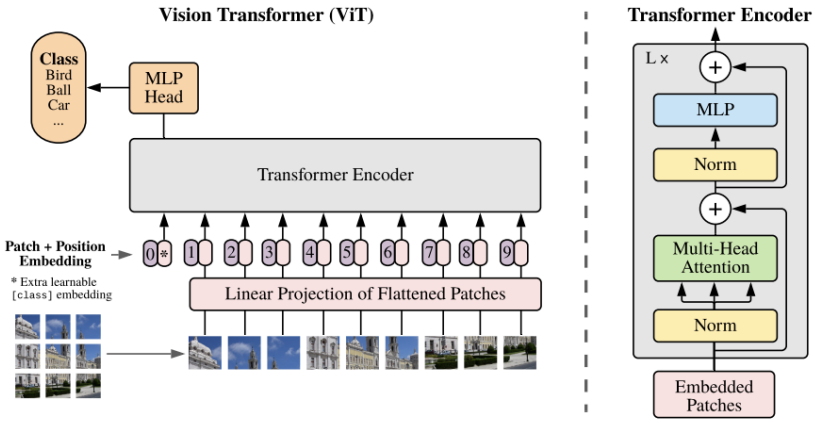

Figure 2.4: Vision transformer model overview (Dosovitskiy et al., 2021). An image is split into fixed-size patches, each of them are linearly embedded, position embeddings are added, and the resulting sequence of vectors are fed into a standard Transformer encoder. In order to perform classification, the standard approach of adding an extra learnable "classification token" to the sequence is used.

2.2.1 Patch Embeddings

Unlike text, which is inherently tokenized as words, images need a different approach to create tokens. The naive application of self-attention to images would require that each

pixel attends to every other pixel. Unfortunately, this does not scale to realistic input sizes due to the quadratic cost in the number of pixels.

As the original Transformer receives an input as a 1D sequence of token embeddings, patch tokenization was introduced by Cordonnier et al. (2020). Patch tokenization divides an image into fixed-size non-overlapping patches and treats each patch as a 'word' in the Transformer's vocabulary. Cordonnier et al. (2020) extracted patches of size 2×2 pixels from the input image and applied full self-attention on top. Unfortunately, due to the usage of a small patch size, the model is applicable only for small-resolution images. Dosovitskiy et al. (2021) went a bit further with this approach and typically used 16×16 pixels as a patch size, allowing their model to handle medium-resolution images as well.

To handle 2D images, the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is reshaped into a sequence of flattened 2D patches $\mathbf{x_p} \in \mathbb{R}^{N \times (P^2 \times C)}$, 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, which also serves as the effective input sequence length for the Transformer (Dosovitskiy et al., 2021).

In other words, each image is split into N squares (patches), and then each patch is flattened into a 1D vector (Figure 2.4). For example, if a patch is 16×16 pixels with three color channels (RGB), it results in a vector of size $16 \times 16 \times 3 = 768$ values.

As the Transformer uses a constant latent vector size D through all of its layers, Dosovitskiy et al. (2021) flatten the patches and map to D dimensions with a trainable linear projection (Figure 2.4). The output of this projection is named as *patch embeddings*.

2.2.2 Classification Token

The Vision Transformer introduces a learnable classification token (CLS) that is attached to the beginning of the input sequence of patch embeddings before it is passed through the Transformer layers (Dosovitskiy et al., 2021). This token plays a key role in enabling the model to perform image classification tasks, and the concept is adapted from BERT, where the CLS token is used for classification purposes (Devlin et al., 2019).

Once added, the CLS token is processed together with the patch embeddings through the self-attention and feed-forward layers in the Transformer. At each layer of self-attention, the CLS token can gather information from all other patch embeddings in the sequence. This process allows it to aggregate a global representation of the entire image progressively as it passes through each layer. After the sequence has passed through all Transformer

layers, the final embedding contains a summary of the information from all patches in the image, compressed into a single vector.

The final embedding of the CLS token is then passed into a linear classifier that produces the final class probabilities for the image. This is a critical step, which enables the CLS token to work as a bridge between the ViT's ability to encode the image information and the final classification output.

2.2.3 Positional Encoding

As the traditional Transformer (Vaswani et al., 2017) processes each token individually, it lacks any notion of spatial relationships between the image patches. To retain spatial information in ViT, positional encodings are added to the patch embeddings. These encodings are learnable 1D positional embeddings, allowing the model to dynamically adapt the spatial context. They provide the model with information about the relative position of each patch, maintaining spatial coherence across the image. Although there are more advanced 2D-aware position embeddings, Dosovitskiy et al. (2021) did not observe significant performance gains compared to 1D positional embeddings.

3 Scalability Limitations

In computer vision, *scalability* refers to a model's ability to handle large datasets, high-resolution images, and complex image features while maintaining efficiency and accuracy. Unlike CNNs, which process images in localized regions using fixed convolutional filters (Dosovitskiy et al., 2021; Goodfellow et al., 2016), ViTs use self-attention mechanisms that enable them to model long-range dependencies across entire images. However, this flexibility comes with computational trade-offs. Originally developed for language processing, the Transformer model has evolved to handle image-based tasks by adapting its attention mechanisms and architecture to process high-dimensional visual data. Unfortunately, these traditional ViTs encounter scalability challenges because of the complex structure of image data, which is considerably more resource intensive than textual data.

This chapter explores the scalability limitations of ViTs, focusing on computational complexity, memory constraints, hardware limitations, and ViT's dependence on extensive datasets. The following sections will analyze each of them separately and create a deeper understanding of these limitations. Understanding these limitations is crucial for formulating strategies to enhance the applicability of ViTs. This chapter sets the stage for the next chapters, which explore recent advances, specific techniques, and architectural innovations. The objective of these efforts is to enhance the efficiency and scalability of ViT, making it more accessible and practical for real-world applications.

3.1 Computational Complexity Challenges

One of the primary scalability constraints of ViTs is their significant computational demand. The self-attention mechanism, which is central to Transformer architectures, has a quadratic time and memory complexity with respect to the number of input tokens (Vaswani et al., 2017). In the original ViT model (Dosovitskiy et al., 2021), the number of input tokens is directly proportional to the resolution of the image, leading to an exponential increase in memory and computational demand as the size of the image increases. An input image of spatial resolution $H \times W$ is divided into a sequence of flattened, non-overlapping patches of size $P \times P$. This results in $N = \frac{HW}{P^2}$ input tokens, excluding classification tokens. Consequently, the number of tokens N scales quadratically with the

dimensions H and W of the input image, assuming a fixed patch size P. This relationship means that the $O(N^2)$ complexity of self-attention translates into a computational cost that scales quadratically with the $H \times W$ image resolution.

This quadratic scaling poses a significant challenge for processing high-resolution images. For example, increasing the size of the image from 128×128 pixels to 1024×1024 pixels, while keeping the patch size constant, increases the number of tokens by a factor of $(\frac{1024}{128})^2 = 8^2 = 64$. This leads to a $64^2 = 4096$ fold increase in the computational cost and memory footprint of the self-attention layers. This quadratic scaling makes it impractical to apply standard ViTs to high-resolution images without substantial computational resources (Khan et al., 2022; Liu et al., 2021).

The scalability of ViTs is a critical factor in their future adaptation to real-life computer vision tasks. The need for scalable solutions in ViTs is becoming increasingly important due to the growing complexity of computer vision applications. Computationally demanding computer vision tasks like high definition image classification, video analysis, and autonomous driving push the boundaries of ViTs. These applications often require real-time performance and high accuracy, highlighting the need for efficient ViT variants that can operate under resource constraints.

In response, recent research efforts have focused on reducing the computational complexity of ViTs. These include strategies such as sparse attention mechanisms with linear attention approximations (Jeevan and Sethi, 2022), hierarchical models (Liu et al., 2021; W. Wang et al., 2021), token reduction strategies (Xu et al., 2022; Rao et al., 2021), and efficient training methods (Touvron, Cord, Douze, et al., 2021; Touvron, Cord, Sablayrolles, et al., 2021; Bao et al., 2022). These strategies will be explored in more detail in the upcoming chapters.

3.2 Hardware Limitations and Memory Constraints

Beyond computational cost, ViTs face significant memory and hardware constraints, particularly during training. The requirement to store full attention maps and intermediate activations scales quadratically with image size, leading to prohibitive memory usage on standard GPUs and TPUs. Training large-scale ViTs on datasets such as JFT-300M (Sun et al., 2017) typically requires high-end hardware, including multi-GPU or -TPU setups, which are not accessible for most researchers or smaller organizations. Such hardware configurations are typically available only to well-funded research labs and large tech com-

panies, contributing to a growing accessibility gap in AI research (Strubell et al., 2019). This accessibility gap hinders innovation outside of the major technology companies and research institutions. In addition, the US has issued new restrictions on the export of US-developed computer chips that power AI, to prevent their rivals like China and Russia from accessing the advanced technology (CNN, 2025). This will further increase hardware inequality between countries and organizations.

In addition, high memory demands can lead to long training times and inefficient hardware utilization. Training traditional ViTs in reasonable time requires high-performance computing clusters, often composed of hundreds of GPUs or TPUs that run for extended periods. For example, pre-training of the largest model variant in the standard ViT with the JFT-300M dataset would have taken almost 7 years (2.5k days) to train with a single TPUv3 core (Vaswani et al., 2017). Even when distributed across high-performance computing clusters, such training is costly, energy-intensive, and often unsustainable for smaller institutions (Strubell et al., 2019). This topic will be explored further in this chapter.

In addition to training challenges, the deployment of ViTs on edge devices, such as unmanned aerial vehicles (UAVs), mobile platforms, and embedded systems, presents significant hardware limitations. These instruments typically encounter significant limitations in terms of power, memory, and processing capabilities, posing challenges for the efficient operation of ViTs. Unlike traditional CNNs, which can be optimized for low-power environments using techniques such as pruning and quantization, ViTs require substantial computational power due to their self-attention mechanisms.

The need for large amounts of on-device memory introduces additional complexity to the challenge, as storing even a moderately sized ViT model can exceed the memory capacity of embedded hardware. As an example, we can take a look at the memory requirements of two different model variants of the original ViT: ViT-Base and ViT-Large (Dosovitskiy et al., 2021). ViT-Base contains approximately 86 million parameters, while ViT-Large has approximately 632 million parameters. If we assume that each parameter usually requires 4 bytes of storage in a standard 32-bit floating-point format, ViT-Base alone demands about 344 MB of memory just for storing model weights, whereas a better performing ViT-Large can require up to 2.5 GB. These storage requirements do not even include additional memory needs for activations, gradients, and optimizer states during inference and training. Therefore, the memory requirements are even higher in reality. Edge devices, such as Raspberry Pi 4 with 1-8GB of RAM (Raspberry Pi Foundation, 2019) or NVIDIA

Jetson Nano with 4GB of RAM (NVIDIA Corporation, 2019), can struggle to accommodate these large models. In many cases, the maximum memory demand during inference significantly surpasses the on-device memory capacity, necessitating memory swapping or offloading computations to cloud-based servers, which introduces latency and power inefficiencies. When considering real-time applications, the inability to fit ViTs within these constraints severely limits their feasibility for deployment in resource-constrained environments.

Furthermore, physical space and weight restrictions further limit the feasibility of deploying large-scale ViTs in UAVs and other mobile platforms. These devices must balance computational performance with energy efficiency and heat dissipation, factors that can significantly impact flight duration and operational capabilities. To address these issues, lightweight alternative architectures such as *TinyViT* (K. Wu et al., 2022) have been created, but they often involve trade-offs in accuracy and generalizability. Hardware acceleration techniques could also be utilized to provide more computational power without adding additional computational units. However, this can be costly and complex due to dedicated hardware design and optimization.

In conclusion, the memory and hardware limitations of ViTs underscore the need for optimized architectures and deployment strategies. As real-world applications increasingly demand compact, low-power solutions, addressing these challenges will be crucial for making ViTs viable in constrained environments. Various optimization techniques and architectural improvements that try to address these issues will be explored in the upcoming chapters 4 and 5.

3.3 Dependency on Extensive Datasets

Another critical limitation of Vision Transformers is their dependence on large-scale labeled datasets. Standard ViT has shown impressive performance, often exceeding traditional CNNs in tasks such as image classification and object detection (Dosovitskiy et al., 2021). However, this success often comes at the cost of extensive pre-training on massive labeled datasets. Unlike CNNs, which can achieve competitive performance with relatively smaller datasets through strong inductive biases, ViTs require extensive pre-training on massive datasets (Dosovitskiy et al., 2021; Lu et al., 2022) such as ImageNet-21k with 14M images (Kolesnikov et al., 2020) or JFT-300M with 300M images (Sun et al., 2017). This requirement makes the training process highly data-intensive and raises concerns about

dataset availability and bias.

The required datasets are often task-specific and do not always exist in public domains. Manually annotating massive image datasets with millions of images is expensive and time-consuming, particularly for specialized fields such as medical imaging, where expert annotations are required. Acquiring labeled datasets of this magnitude is also often infeasible in specialized fields due to the limited availability of labeled data and the requirement for specialized knowledge. Although general sources, such as the internet, can sometimes be utilized in a large dataset collection, it can introduce biases and underrepresentation of certain categories. This may lead the model to fail to generalize the underrepresented domain, and thus limiting their real-world applicability due to the underperformance in real-world applications. In some cases, transfer learning can help mitigate the need for massive labeled datasets. However, it fails to perform effectively within specific domains that are significantly distant from the training dataset.

Another aspect of the dependency on large-scale datasets is the energy consumption associated with training ViTs. As discussed in the previous section, the training of the models often necessitates the use of high-performance computing clusters that operate for extended periods, particularly when dealing with large-scale datasets. The previously given example of the standard ViT training time with the JFT-300M dataset, which is almost 7 years (2.5k days) with a single TPUv3 core, highlights the issue. This leads to substantial energy consumption, contributing to high financial costs and environmental impact (Strubell et al., 2019; Fu et al., 2021). Strubell et al. reported in their study that the training of large-scale deep learning models can generate carbon emissions comparable to multiple times the lifetime emissions of a car. They also estimated the training costs of various NLP models, and even though training a single model is relatively cheap, the full cost required to build a working model can be extremely expensive. Thus, research and development can be unsustainable for many organizations. Although their paper focused on NLP models, the results apply also to ViTs but on a larger scale. The underlying Transformer architecture is the same in both fields, ViTs just scales quadratic compared to Transformers in NLP. As ViTs continue to scale, their associated energy consumption and carbon emissions increase accordingly, making sustainability also a key concern for upcoming research and development efforts.

To address these challenges, researchers are exploring data-efficient learning methods (Touvron, Cord, Douze, et al., 2021; Bao et al., 2022), which aim to reduce the dependency on massive labeled datasets while maintaining model performance. Furthermore, hybrid

approaches that combine ViTs with CNNs have shown promise in reducing dataset dependency by leveraging the strong inductive bias of CNN (H. Wu et al., 2021). These will be explored in more detail in the upcoming chapters.

4 Computational Complexity Reduction

Managing computational complexity is essential to effectively scale ViTs and apply the model to real-world visual tasks. ViTs require more computational resources compared to CNNs as they utilize a self-attention mechanism that scales quadratically with the number of input tokens. This design causes a significant computational burden, especially with high-resolution images as each token interacts with every other token in the self-attention process, or with substantial datasets which are often necessary to adequately train Vision Transformer models for precise predictions. Addressing these computational challenges has led to a variety of strategies, mainly focusing on reducing the number of tokens processed, optimizing attention mechanisms, and developing more efficient architectures.

This chapter explores various strategies that allow ViTs to achieve better scalability by reducing their computational complexity. Depending on the use case, the scalability needs can arise from various reasons, i.e. larger input images, real-time processing needs, or with hardware constraints which require performing the processing with cheaper and computationally less powerful hardware. By analyzing these scalability enhancements, we can better understand the future potential and limitations of Vision Transformers in large-scale computer vision tasks.

4.1 Enhancing Training Efficiency

In the previous chapter, we covered some scalability issues that traditional ViTs have with their extensive training method. Due to these issues, enabling effective training of ViTs on smaller datasets is essential to expand the range of AI applications. By reducing the dependence of large datasets, researchers and organizations with limited computational resources can train competitive models. This enables greater involvement in AI development within both industry and academic sectors due to the reduced investment costs. Many real-world applications require models tailored to specific domains, such as medical diagnostics, satellite imagery analysis, or industrial quality control. In these fields, access to large extensive labeled datasets can be limited, making data-efficient models essential.

Smaller datasets also allow faster training and experimentation cycles, which enable organizations to deploy and refine models more rapidly, responding to changing requirements and competition. Efficient training methods with smaller datasets also help respond to the growing calls for environmentally sustainable AI practices as fewer computational resources are used. Transformer-based AI models have gained significant popularity due to their superior performance, especially in the field of NLP. Unfortunately, these models need large datasets and long training times to outperform other models (Dosovitskiy et al., 2021). Like discussed in the previous chapter, building a large-scale deep learning model can be extremely expensive, consume a lot of energy, and at the same time produce a lot of carbon emissions (Strubell et al., 2019). Models that can be trained efficiently with fewer computational resources are not only economically more viable but also consume less energy within the training phase. This will also reduce their carbon footprints, which is essential for making ViT applications environmentally sustainable in the long run.

In this section, we will go through various studies which aim to enhance the training efficiency of ViT. We will start by going through popular *Data-efficient Image Transformer* (DeiT) model in a little more detail as a reference. The model was developed and published by Meta AI in 2021(Touvron, Cord, Douze, et al., 2021) and allows one to train accurate ViTs with smaller datasets, increasing the overall training speed while doing so. DeiT has been chosen as a reference due to its performance and popularity as a baseline model in various research studies that are introduced later in this chapter. After the introduction of DeiT, we will go through various other strategies that improve training efficiency, but these are explained only on a high level with their results.

4.1.1 Data-efficient Image Transformer

The data-efficient image transformer (DeiT)(Touvron, Cord, Douze, et al., 2021), is a specific approach to traditional ViT to improve its efficiency and performance, especially allowing training on smaller datasets without requiring large-scale labeled datasets such as ImageNet. DeiT introduced innovative techniques that address some of the limitations of standard ViTs and make ViT data efficient without compromising performance. DeiT achieves competitive accuracy when trained on datasets as small as ImageNet-1k(Deng et al., 2009), which contains only 1.28 million images. Its key innovations include the introduction of a *distillation token* for a *teacher-student strategy* that is specified to transformers, and the use of advanced *data augmentation* and *regularization techniques*. These innovations enable effective training on smaller datasets without requiring additional pre-

training or external data, making DeiT an ideal choice for applications where data availability is limited or resources are constrained.

Teacher-Student Strategy

The teacher-student strategy is a training method that transfers knowledge from the teacher to the student. The goal is to transfer the knowledge learned by the teacher model to the student model in a way that the student can achieve similar performance but with fewer parameters or computational requirements. The method is used particularly in knowledge distillation (Gou et al., 2021).

A *teacher model* is a high performing pre-trained network, typically a CNN, that has already learned to recognize patterns from a larger dataset. The teacher provides soft labels, which are probability distributions over the output classes rather than hard binary labels. These soft labels carry richer information about inter-class relationships, such as the similarity between classes. The *student model*, DeiT in this case, is trained to mimic the teacher's behavior while simultaneously learning from the labeled training dataset. Learning from soft labels, the student model develops a better understanding of the data distribution, which improves generalization, especially on smaller datasets.

This strategy minimizes the need for massive labeled datasets by using the knowledge of the already trained high-performing teacher, effectively 'borrowing' the generalization capabilities of larger datasets without directly requiring them.

Distillation Token

DeiT introduces a unique distillation token to facilitate the transfer of knowledge from a teacher model during training (Touvron, Cord, Douze, et al., 2021). In standard ViTs, the input of the Transformer consists of a sequence of image patch embeddings, along with a class token that aggregates information for classification (Dosovitskiy et al., 2021). In DeiT, an additional distillation token is added to this input sequence. Like the class token, the distillation token is a learnable vector that is passed through the self-attention layers, allowing it to gather information about the image and teacher outputs. It interacts with the other tokens through the self-attention mechanism in each layer and is specifically trained to capture the output distribution of the teacher model.

During training, two loss functions are used: one for the class token and one for the distillation token. The loss function of the class token uses ground-truth labels, and the

loss function of the distillation token uses the teacher's prediction distribution over the target classes. These soft labels carry information about inter-class relationships, helping the student model, DeiT in this case, learn the correct labels, as well as the confidence levels and the similarities among different classes.

The distillation token allows DeiT to achieve competitive accuracy on smaller datasets by leveraging a high-performing teacher model trained on existing data. This eliminates the need for large-scale pretraining on external datasets, reducing both the computational cost and data requirements.

Data Augmentation and Regularization

One of the primary challenges in training ViTs is their dependence on extensive datasets to acquire meaningful representations. Unlike CNNs, ViTs do not have built-in inductive biases that force them to learn these patterns directly from the training data (Dosovitskiy et al., 2021; Lu et al., 2022). This feature makes ViTs highly flexible, yet also demanding substantial amounts of training data. To address this, DeiT uses data augmentation and regularization techniques that enhance the learning process (Touvron, Cord, Douze, et al., 2021), allowing the model to perform well even with relatively limited data.

Data augmentation (Shorten and Khoshgoftaar, 2019) plays a crucial role in increasing the effective size of the dataset. Instead of collecting millions of additional labeled images, data augmentation applies transformations to existing images. These transformations prevent the model from overfitting by ensuring that it does not rely too heavily on specific pixel arrangements. This makes it more robust to variations in real-world data as it learns features from different image variations. DeiT rely on extensive data augmentation with strong data augmentation strategies such as RandAugment, Mixup, and CutMix (Touvron, Cord, Douze, et al., 2021). RandAugment (Cubuk et al., 2020) applies a series of random transformations to each image, ensuring that the model encounters a diverse set of variations during training. MixUp (Zhang et al., 2018) blends two different images by interpolating their pixel values and labels, forcing the model to make predictions based on mixed features. CutMix (Yun et al., 2019) replaces a portion of one image with a patch from another, encouraging the model to recognize partial visual patterns and make more holistic predictions.

Regularization further aids in mitigating overfitting by preventing the model from becoming excessively certain about its predictions. DeiT uses stochastic depth and label

smoothing as its regulation techniques together with MixUp and CutMix (Touvron, Cord, Douze, et al., 2021). The stochastic depth technique (Huang et al., 2016) randomly drops certain layers during training, forcing different parts of the network to learn independently, leading to better generalization. The label smoothing technique prevents the model from assigning too sharp probability distributions to class predictions. Label smoothing slightly distributes some probability mass to incorrect classes by converting the hard labels into soft labels, making the model less overconfident and more adaptable to unseen data. The approach assumes that the true label has a probability of $1 - \epsilon$, and the remaining probability ϵ is shared between the remaining classes. Although Mixup and CutMix are often classified as data augmentation techniques, they also serve as regularization methods by promoting smoother decision boundaries and improving model generalization.

Together, data augmentation and regularization form an essential component of the DeiT training strategy. Augmentation ensures that the model learns from a diverse set of images without requiring exponentially more data, while regularization techniques make the model more robust by reducing the risk of overfitting. These combined strategies are essential for scaling ViTs to scenarios where data availability is limited, ensuring that they remain competitive with CNNs even in constrained training environments.

Model Variants

The DeiT introduces three main model variants to balance computational efficiency and accuracy. These variants primarily differ in their architectural complexity, with the number of parameters and computational requirements adjusted to suit different resource constraints. The base model DeiT-B has the same architecture as ViT-Base with 86 million parameters. This model is designed to maximize accuracy and is comparable in size to the original ViT-Base (Dosovitskiy et al., 2021). The other two variants are DeiT-S (small) with 22 million parameters and DeiT-Ti (tiny) with 5 million parameters. The DeiT-S offers a mid-range trade-off between efficiency and accuracy, and the DeiT-Ti offers highly efficient training and inference with a cost of accuracy.

Results

To evaluate the effectiveness of the model variants, DeiT models were compared on a widely used ImageNet-1k dataset (Deng et al., 2009). The novel distillation token version of DeiT-B achieved top-1 accuracy 85.8 %, outperforming traditional ViT and state-of-the-

art CNN (Touvron, Cord, Douze, et al., 2021). DeiT-B without distillation had slightly lower accuracy than the state-of-the-art EfficientNets (Tan and Le, 2019) CNN models, but the addition of a novel token-based distillation technique made DeiT-B outperform EfficientNets in terms of accuracy and throughput. This accuracy and efficiency make DeiT a viable choice for various applications, offering a scalable alternative to traditional ViTs.

4.1.2 Other Related Studies

Beyond the Data-Efficient Image Transformer (Touvron, Cord, Douze, et al., 2021), several other studies have also explored techniques to improve the training efficiency of ViT. These approaches aim to reduce computational cost, improve convergence speed, and optimize model performance. In the following, some of the most relevant methods are highlighted.

Knowledge distillation DeiT successfully introduced knowledge distillation (Gou et al., 2021) for ViTs, but subsequent studies have refined this approach further.

Touvron, Cord, Sablayrolles, et al. (2021) introduced the *Class-Attention in Image Transformers* (CaiT) model, with the aim of enhancing the performance of ViTs by allowing the construction of deeper architectures. CaiT introduces a dedicated class-attention mechanism that operates after the standard transformer layers. This allows the model to focus on summarizing class-specific information more effectively. In addition, CaiT incorporates a technique, named LayerScale, which enhances the stability of training in deep networks. Specifically, LayerScale introduces learnable scalar parameters that are applied to the outputs of the attention and feedforward blocks within each transformer layer. These parameters are initialized with small values, allowing the network to initially down-weight the influence of each residual branch and gradually learn their optimal contributions during training. Together, these innovations enable CaiT to achieve state-of-the-art performance on ImageNet (Deng et al., 2009), achieving a top-1 accuracy of 86.5 % without external data, while utilizing fewer parameters and computational resources compared to previous models.

Touvron et al. (2022) revisited the supervised training of ViTs in their work “DeiT III: Revenge of the ViT”. They proposed a simplified training procedure that builds upon and streamlines a recipe initially developed to train ResNet-50 (He et al., 2016). This procedure includes a new straightforward data augmentation strategy (3-Augment), which uses

only three augmentations. The authors demonstrated that this approach leads to significant improvements in ViT performance in various tasks, including image classification, transfer learning, and semantic segmentation. Notably, their method outperforms previous fully supervised training techniques for ViTs and achieves performance comparable to more recent architectures. The results could serve as better baselines for the recent self-supervised approaches demonstrated on ViT.

Self-supervised learning In addition to supervised pretraining, self-supervised learning techniques have emerged as a powerful tool to enhance training efficiency. For example, BEiT (Bao et al., 2022), DINO (Caron et al., 2021), and MAE (He et al., 2022) demonstrated that ViTs could learn meaningful representations without labels, leading to more efficient downstream training. These approaches reduce reliance on labeled data and accelerate convergence during fine-tuning.

Adaptive computation Several studies propose adaptive computation frameworks that allocate more resources to informative regions of an image while skipping redundant computations in less significant areas. Token reduction techniques such as *token pruning* and *token pooling* dynamically adjust the number of tokens processed per image, leading to significant speed-ups while maintaining classification accuracy. In the upcoming section, we will go through these methods and their related studies in more detail.

Efficient Attention Mechanisms Several studies have aimed to reduce the quadratic complexity of the standard self-attention mechanism of the Transformer. Methods like *Linformer* (S. Wang et al., 2020), *Performer* (Choromanski et al., 2021) and *Nyströmformer* (Xiong et al., 2021) use approximation algorithms to scale the self-attention of the Transformer to be more efficient. Although these methods were originally developed for traditional transformers (NLP), Jeevan and Sethi (2022) explored the integration of these into ViT in their study. The study illustrated that replacing the standard self-attention mechanism in ViTs with these efficient attention variants can substantially decrease the use of computational resources while preserving or even improving image classification performance. These approaches reduce memory and computational overhead while preserving performance, making ViTs more practical for settings where resources are constrained. Later in this chapter, we will also go through various methods to improve the architecture of ViT and it's computational efficiency.

All of the methods highlighted collectively enhance the training efficiency of ViTs by optimizing computational cost, accelerating convergence, and reducing resource requirements. Future advancements in these areas could further improve the scalability and broaden the applicability of ViTs in real-world applications.

4.2 Token Reduction Techniques

The self-attention mechanism in standard ViT scales quadratically with the number of input tokens, causing a significant computational burden with larger images. In the ViT process, an image is split into fixed-size non-overlapping patches (e.g., 16×16 pixels), which are then flattened into 1D arrays and those are passed through an embedding layer, creating the tokens. Given the distinct patches that are translated into tokens, the number of tokens is directly proportional to the image size, which significantly increases computational load with high-definition images. Although some of the existing methods improve the quadratic complexity of the attention, the main computational bottleneck is the fully connected layers that spend over 80 % of the total computation(Marin et al., 2023). In comparison, softmax attention only takes less than 15 %. Given this fact about the total computation and the quadratic complexity of attention, the most direct approach to accelerate ViTs is to reduce the number of patches. This modification would not only accelerate the attention calculation but also enhance the computational complexity throughout all layers.

The most straightforward token reduction technique is to increase the patch size. This is an efficient way to reduce the computational complexity of the ViT, but unfortunately comes with additional disadvantages that we will discuss in the upcoming section. Instead of using the most straightforward solution, more advanced token reduction techniques can also be applied. Multiple studies have shown that the final prediction in vision transformers is derived exclusively from a subset of the most informative tokens, which is sufficient for accurate image recognition (Caron et al., 2021; Pan et al., 2021; Rao et al., 2021). Thus, it is possible to reduce the number of tokens by focusing mainly on the redundant ones. These techniques aim to reduce the number of tokens while retaining meaningful information without compromising the model's ability to capture spatial details. These methods allow the model to focus only on the most informative regions of an image. This section explores some of the most common token reduction techniques. It starts with the most straightforward one and gradually moves to more advanced techniques.

4.2.1 Patch Size Optimization

The most straightforward solution would be simply to increase the patch size as the number of tokens is directly proportional to the number of patches in the original image grid. By doubling the patch size, for example from 16 to 32 pixels squares, the computational load would drop to a quarter of the original due to the ViT's quadratic scaling factor of input patches. Unfortunately, this would significantly affect the model's ability to capture details from the image, as more data would be encoded into a single patch. This increase in patch size can lead to several significant disadvantages, which are described and explained below.

Loss of spatial details Larger patches cover more area of the image, decreasing the model's ability to capture spatial details. Subtle features within the image might be missed because each patch aggregates information over a larger region. This could be problematic especially for tasks that require high spatial resolution, such as image segmentation, fine-grained classification, or object detection.

Reduced positional encoding effectiveness There are fewer patches with larger patch sizes, which decreases positional granularity, since positional encodings are used in ViTs to provide spatial information. This can lead to less precise spatial relationships encoded and understood by the model.

Reduced information flow By reducing the number of patches, we also limit the amount of data that flows through the transformer processes. While this reduces computational costs, it also weakens the model's ability to learn nuanced patterns from the image.

Overlooking small objects Small objects or features smaller than the size of the patch may not be correctly represented. If a significant object is smaller than the patch, it can become diluted into a single patch, making it harder for the model to recognize.

Focusing to coarser features By aggregating over larger areas, the model can focus more on coarser features while ignoring finer details. This can lead to a bias toward high-level patterns, which might not generalize well across datasets requiring detailed understanding.

Less effective transfer learning Models that are trained on large datasets, such as ImageNet (Deng et al., 2009), with larger patch sizes, have less effective transfer learning capabilities. This is especially applicable to downstream tasks that require finer resolutions, since the learned features are less detailed.

Due to the possible disadvantages listed above, choosing the suitable patch size is always a trade-off between computational efficiency and the level of spatial detail required for the task. Thus, more advanced token reduction techniques, which are described in the following, are more favorable to reducing the number of tokens.

4.2.2 Token Pruning

Token pruning is a method of enhancing computational efficiency by reducing the number of tokens processed during inference or training. It selectively removes or "prunes" tokens that are estimated to be less important. This effectively reduces the quadratic complexity of self-attention mechanisms without significantly compromising model performance.

Importance Evaluation

Token pruning focuses on preserving tokens that capture attention, while eliminating those that do not, by designing importance evaluation strategies. Let the token embeddings be $\mathbf{X} \in \mathbb{R}^{N \times d}$,where N is the number of tokens, and d is the embedding dimension. The tokens are pruned according to an *importance score* s_i calculated for each token i based on the selected evaluation strategy. Common evaluation strategies are the following.

Attention scores Tokens receiving low attention on transformer heads are considered less critical. Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ represent the attention matrix of a given layer, where N is the number of tokens. The attention score s for token i can be calculated as:

$$s_i = \sum_{j=1}^{N} \mathbf{A}_{ij},$$

where $\mathbf{A}_{ij}$ is the attention weight from token i to j. Tokens with lower s_i are considered less significant and can be pruned. This strategy assumes that low attention weights indicate a lower contribution to the overall model output.

An alternative approach uses aggregated attention across multiple transformer heads. If

there are H heads, the aggregate score s for the token i is:

$$s_i^{(H)} = \frac{1}{H} \sum_{h=1}^{H} s_i^h,$$

where s_i^h is the attention weight of the head h:

$$s_i^h = \sum_{j=1}^{N} \mathbf{A}_{ij}^{(h)}.$$

Feature magnitudes Tokens with lower feature magnitudes are assumed to carry less semantic information. Let $\mathbf{X} \in \mathbb{R}^{N \times d}$ represent the embeddings, where N is the number of tokens, and d is the embedding dimension. The importance score s of a token i can be calculated as:

$$s_i = \|\mathbf{x}_i\|_p,$$

where $\mathbf{x_i}$ is the embedding vector i of the token and $\| \cdot \|_p$ is the p-norm. Typically, the l_2-norm is used ($p = 2$):

$$s_i = \sqrt{\sum_{k=1}^{d} x_{ik}^2}.$$

Tokens with a lower α_i score are pruned under the assumption that smaller magnitudes indicate less informative embeddings.

Learned scores A specialized sub-network predicts token importance scores dynamically. These scores are learned jointly with the main task during training, ensuring that token pruning aligns with the overall objective of the model. The sub-network is typically implemented as a lightweight neural network, which predicts the importance score for each token. Let $\mathbf{X} \in \mathbb{R}^{N \times d}$ represent the embeddings, where N is the number of tokens, and d is the embedding dimension. The learned importance scores $\mathbf{s}$ are calculated as

$$\mathbf{s} = f_\phi(\mathbf{X}),$$

where f_ϕ represents the transformation in the sub-network. This sub-network is typically a multilayer perceptron that operates on the token embeddings. *DynamicViT* is a good example of the Vision Transformer model that uses a lightweight prediction module to estimate the importance score of each token (Rao et al., 2021).

Pruning Rules

Pruning rules define the criteria for deciding which tokens are removed or "pruned" during computation. Once the importance of the token is evaluated, the less inattentive tokens are pruned with the desired pruning rule. Common pruning rules are as follows.

Threshold-based selection Tokens with importance scores above a certain threshold τ are retained, and others are pruned:

$$\mathcal{T} = \{i \mid s_i \geq \tau\}$$

Top-K selection Only the top k tokens with the highest scores s_i are preserved for further computation, and others are pruned:

$$\mathcal{T} = \{i \mid s_i \text{ is among the top } k \text{ values of } \mathbf{s}\},$$

where $\mathbf{s} = [s_1, s_2, ..., s_N]^T$ represents the token importance scores of the tokens.

Finally, the pruned tokens are either merged into neighboring tokens or completely discarded. The discard method reduces memory usage and computation cost, but critical information can be lost during the process. Merging is less optimal resource-wise, but ensures spatial continuity as information from pruned tokens can be reconstructed by merging. The merging method can also be considered part of the *path pooling* which is described later in this chapter. The tokens that are preserved constitute the pruned set:

$$\mathbf{X}' = [\, \mathbf{x}_i \mid i \in \mathcal{T} \,] \in \mathbb{R}^{N' \times d}, \quad \text{where } N' = |\mathcal{T}| < N.$$

Related studies

Token pruning has shown promising results in bridging the efficiency gap between ViTs and more traditional architectures like CNNs, especially in resource-constrained applications. By dynamically reducing the number of tokens processed at various stages of the model, token pruning enables more computationally efficient inference without significantly compromising performance. Recent studies in patch pruning have demonstrated its effectiveness in significantly reducing the computational cost of Vision Transformers while preserving equivalent performance levels.

Pan et al. (2021) introduced the *IA-RED*^{2} model, which progressively removes uncorrelated tokens at various stages, which significantly reduced computational costs while maintaining

performance. The framework achieved a speed increase of up to 1.4 times for state-of-the-art models such as DeiT (Touvron, Cord, Douze, et al., 2021), with a minimal accuracy reduction of less than 0.7 % in the ImageNet-1k dataset (Deng et al., 2009). The number of floating-point operations (FLOPs) was also reduced by $30 - 40$ % depending on which variant was used in the test.

Rao et al. (2021) introduced the *DynamicViT* model that uses a lightweight prediction module to estimate the importance score of each token given the current features and prune redundant tokens progressively and dynamically based on the input. The module is added to different layers to prune redundant tokens hierarchically. This technique significantly lowers $31 \sim 37$ % of FLOPs and increases throughput by more than 40 % by systematically eliminating 66 % of the input tokens hierarchically. Despite these optimizations, the accuracy decreases by no more than 0.5 % for various vision transformers.

Xu et al. (2022) introduced the *Evo-ViT* model, which employs a self-motivated slow-fast token evolution approach to enhance the efficiency of vision transformers. This method dynamically selects informative tokens using global class attention and updates them through a slow-fast mechanism, maintaining spatial structure and information flow. Evo-ViT significantly reduces the computational cost of traditional ViT while preserving accuracy. For example, it accelerates the DeiT-S model (Touvron, Cord, Douze, et al., 2021) by more than 60 % in throughput with only a 0.4 % decrease in top-1 accuracy on ImageNet-1k (Deng et al., 2009). The model also outperformed existing token pruning methods, such as IA-RED^2 (Pan et al., 2021) and DynamicViT (Rao et al., 2021), both in precision and efficiency.

Tang et al. (2022) introduced a novel *patch sliming* technique, which employs a top-down approach to identify and remove redundant patches. The experimental findings illustrate that this technique can substantially reduce computational demands with minimal impact on accuracy. In particular, applying this method to the DeiT-Ti model (Touvron, Cord, Douze, et al., 2021) resulted in more than a 45 % reduction in FLOPs while incurring only a 0.2 % decrease in top-1 accuracy on the ImageNet dataset (Deng et al., 2009).

Yin et al. (2022) introduced the *A-ViT* model, which adaptively adjusts the inference cost of the vision transformers by dynamically reducing the number of tokens processed during inference. This is achieved through an adaptive halting mechanism that computes a halting probability for each token, allowing the model to discard redundant spatial tokens as processing progresses. In particular, A-ViT does not require additional parameters or modifications to the network architecture, as it leverages the original network parameters

for adaptive halting. With the ImageNet-1 dataset (Deng et al., 2009), A-ViT improves the throughput the tiny model variant of DeiT (Touvron, Cord, Douze, et al., 2021) by 62 % and its small model variant by 38 %, with only a decrease in the precision of 0.3 %, outperforming previous methods in terms of optimizing both efficiency and performance. The number of FLOPs was reduced by 38 % compared to the baseline model used in the tests.

Bonnaerens and Dambre (2023) introduced the *Learned Thresholds Token Merging and Pruning* (LTMP) method to improve the efficiency of vision transformers by dynamically reducing the number of tokens processed. LTMP employs learned threshold masking modules that determine which tokens to merge and which to prune, effectively balancing token merging and pruning across transformer layers. This approach allows for significant reductions in computational cost with minimal accuracy loss. For example, applying LTMP to the DeiT-S model (Touvron, Cord, Douze, et al., 2021) achieves state-of-the-art accuracy across various reduction rates while requiring only a single fine-tuning epoch, which is an order of magnitude faster than previous methods.

Challenges

Although token pruning has proven to be a powerful tool for enhancing the efficiency of Vision Transformers, it also poses some challenges, including the *risk of information loss*, *task dependency* and *computational overhead and design complexity*. Critical information can be lost by excessive pruning, which can significantly reduce the accuracy of the model, especially in tasks that require fine-grained spatial details. Token pruning is also task dependent, and its effectiveness varies across tasks, requiring meticulous fine-tuning for individual tasks. Additional dynamic scoring modules also add computational overhead and design complexity to the model.

4.2.3 Token Pooling

Token pooling is a method to reduce the number of tokens during processing by aggregating multiple tokens into a more representative single token. It is inspired by the pooling layers in CNNs. The method aims to condense the spatial information of less significant or redundant regions while preserving critical features. It compresses spatially or semantically similar tokens, ensuring that fewer but more meaningful tokens are forwarded to subsequent layers. The process involves *grouping* the tokens into clusters and *pooling*

their features into representative embeddings, effectively lowering the computational cost without a drastic performance loss.

Grouping

At first, tokens are assigned to groups based on predefined or dynamically learned criteria. Let the token embeddings be represented as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times d}$, where N is the number of tokens and d is the embedding dimension. The grouping process divides $\mathbf{X}$ into K groups $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_K\}$, where $\mathcal{G}_k$ contains tokens assigned to group k. Tokens are allocated to groups according to the *grouping criteria* that define how tokens are assigned to each group. Some commonly used grouping criteria are the following.

Spatial proximity Tokens are grouped according to their positions in the input image grid. For example, each 2×2 region from the image patch grid could be grouped into a single group. For tokens associated with patches located at positions $\mathbf{p}_i = (x_i, y_i)$, a group $\mathcal{G}_k$ is defined as:

$$\mathcal{G}_k = \{i \mid \mathbf{p}_i \in \mathcal{R}_k\},$$

where $\mathcal{R}_k$ is a predefined spatial region on the image patch grid.

Feature similarity Tokens are clustered based on their feature representations in the embedding space. Let $\mathbf{x}_i$ be the embeddings of the token i. A similarity function $f_s(x_i, x_j)$, such as cosine similarity or attention weight, assigns tokens to a group $\mathcal{G}_k$ with a group centroid $\mathbf{c}_k$:

$$\mathcal{G}_k = \{i \mid f_s(\mathbf{x}_i, \mathbf{c}_k) \geq \tau\},$$

where τ is a similarity threshold suitable for the pooling task at hand.

Learned grouping In this approach, grouping rules are learned during training. This is often done with clustering algorithms, such as K-means, that are integrated into the model architecture. With K-means clustering, tokens would be grouped as:

$$\mathcal{G}_k = \{i \mid \arg\min_j \|\mathbf{x}_i - \mathbf{c}_j\|^2 = k\},$$

where $\mathbf{c}_j$ is the centroid of the j-th group.

Pooling Operations

Once the tokens are grouped, each group $\mathcal{G}_k$ is pooled into a single representative token $\mathbf{y}_k \in \mathbb{R}^d$, where d is the embedding dimension. These tokens are then passed forward within the Vision Transformer model. The common pooling layers of CNNs, such as *max pooling* and *average pooling* (Chollet, 2018), can also be used with ViT. *Attention-based pooling* has also been applied effectively in ViTs to improve performance in various tasks (Xue et al., 2023; Psomas et al., 2023). These pooling operations are described below.

Max pooling Selects the maximum feature value across all tokens in the group:

$$\mathbf{y}_k = \max_{i \in \mathcal{G}_k} \mathbf{x}_i,$$

where the maximum is calculated over each embedding dimension:

$$\mathbf{y}_{k,j} = \max_{i \in \mathcal{G}_k} \mathbf{x}_{i,j},$$

Average pooling Computes the average feature value across all tokens in the group:

$$\mathbf{y}_k = \frac{1}{|\mathcal{G}_k|} \sum_{i \in \mathcal{G}_k} \mathbf{x}_i.$$

Attention-based pooling Attention-based pooling is a technique that integrates attention mechanisms into the pooling process, allowing models to assign varying levels of importance to different input features or tokens. It uses attention scores $\mathbf{A}_{ij}$ to weight each token in the group.

$$\mathbf{y}_k = \sum_{i \in \mathcal{G}_k} \frac{\mathbf{A}_{ij}}{\sum_{i \in \mathcal{G}_k} \mathbf{A}_{ij}} \mathbf{x}_i,$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents the attention matrix of a given layer and N is the number of tokens within that layer.

Related studies

Like token pruning, token pooling has emerged as an effective approach to improve the efficiency of Vision Transformers by reducing the computational inefficiencies encountered when processing extensive sets of tokens within the network. Through strategic aggregation or reduction of tokens, token pooling methods streamline computation while

preserving critical information necessary for accurate predictions. In various studies, token pooling methods have demonstrated the ability to reduce token counts by up to 60 % almost without any accuracy drop in baseline models. On benchmarks like ImageNet, token pooling has demonstrated significant potential in tasks that require efficient processing of high-resolution input, such as medical imaging, remote sensing with UAVs, and autonomous driving.

Marin et al. (2023) introduced a novel token down-sampling method, simply named *token pooling*, to enhance the computational efficiency of vision transformers. Unlike traditional pooling methods that operate on regular grids, Token Pooling treats tokens as discrete samples of an implicit continuous signal. Using K-Means-based clustering abroach, it identifies and aggregates redundant tokens, effectively reducing the number of tokens processed without a significant loss of information. This approach leads to a substantial reduction in computational cost without any further modifications to the architecture while maintaining accuracy. For example, Token Pooling enables the DeiT-Ti model (Touvron, Cord, Douze, et al., 2021) to achieve the same top-1 accuracy on ImageNet-1k (Deng et al., 2009) with 42 % fewer computations.

Wei et al. (2023) introduced a novel joint *Token Pruning & Squeezing* (TPS) module to compress ViTs with higher efficiency. The module is designed to enhance the efficiency of ViTs by jointly pruning and compressing tokens. Unlike traditional token pruning methods that may lead to significant information loss, TPS conserves the information from pruned tokens by redistributing their content into the remaining tokens. This is achieved through unidirectional nearest-neighbor matching and similarity-based fusion steps, which ensure that essential features are retained even after pruning. Experiments demonstrate that TPS outperforms existing methods in various token pruning intensities. In particular, when reducing the computational budgets of the DeiT-Ti and DeiT-S models (Touvron, Cord, Douze, et al., 2021) to 35 %, TPS improves the accuracy by $1 - 6$ % compared to the baselines on ImageNet classification (Deng et al., 2009). Furthermore, the TPS method can accelerate the throughput of the DeiT-S model beyond DeiT-Ti, while its accuracy surpasses DeiT-Ti by 4.78 %. Experiments on various transformers highlight the effectiveness of TPS compression without compromising performance.

Liang et al. (2022)introduced the *EViT* model, which improves the efficiency of ViTs by reorganizing image tokens during the feed-forward process. This method identifies attentive tokens based on class token attention and preserves them, while fusing inattentive tokens to expedite subsequent computations. By integrating this token reorganization into

the training process, EViT reduces computational costs without a significant accuracy loss. For example, incorporating EViT into the DeiT-S model (Touvron, Cord, Douze, et al., 2021) increases the inference speed by 50 %, with only a 0.3 % decrease in the top-1 accuracy on ImageNet classification (Deng et al., 2009). Additionally, EViT does not introduce additional parameters to the model and allows vision transformers to process higher-resolution images at the same computational cost, which leads to better recognition accuracy.

Bolya et al. (2023) introduced *Token Merging* (ToMe) method, which improves efficiency by gradually merging similar tokens during inference. ToMe employs a new proposed lightweight Bipartite Soft Matching algorithm to combine redundant tokens, effectively reducing the number of tokens without requiring additional training. Implementing ToMe can double the throughput of state-of-the-art ViT models with minimal accuracy loss. For example, applying ToMe to the ViT-L model (Touvron, Cord, Douze, et al., 2021) on ImageNet-1k (Deng et al., 2009) results in a 2× increase in throughput and a 2.2× increase on video, with only a $0.2 - 0.3$ % decrease in accuracy in each case. Furthermore, ToMe can also be easily applied during training to further enhance training speed and minimize accuracy loss.

Although the Evo-ViT (Xu et al., 2022) model was already introduced in the token pruning section, it also uses token pooling techniques when it merges the tokens it prunes into a single token to alleviate information loss. This ensures that the information of the pruned tokens is carried over to the next layer.

The Learned Thresholds token Merging and Pruning (LTMP) (Bonnaerens and Dambre, 2023), which was already introduced in the token pooling section, introduced a novel approach that takes advantage of the strengths of token merging and token pruning. It achieved state-of-the-art accuracy across reduction rates, while requiring only a single fine-tuning epoch.

Challenges

Although token pooling has proven to be a powerful technique for improving the efficiency of Vision Transformers, it also introduces several challenges, including the risk of information loss, difficulty in preserving spatial dependencies, and trade-offs between computational efficiency and task-specific requirements. Excessive pooling can lead to the loss of critical information, which can significantly impact model performance, especially

in tasks that require fine-grained spatial features. Token pooling also struggles with preserving spatial dependencies, as naive pooling methods can impair the model's ability to capture local and global relationships effectively. This can be especially tricky with static pooling techniques (e.g., fixed-size pooling) and can lead to suboptimal performance in heterogeneous datasets, as pooling may not adapt well to varying inputs. This can be partially solved by dynamic pooling strategies, but these can introduce additional computational overhead and complexity into the model architecture. The more advanced pooling strategies, which can solve many of the challenges mentioned above, can cause task dependency as fine-tuning might be needed based on the specific application or domain. For example, a task requiring high-resolution features (e.g., medical imaging) might need slightly different pooling than a task involving coarser classification.

5 Architectural Developments

In the previous chapter, we discussed how ViTs can be accelerated by reducing the number of tokens processed. This acceleration is based on the fact that the main computational bottleneck of traditional ViT is fully connected layers that consume more than 80 % of the total computation, while softmax attention takes a bit less than 15 % (Marin et al., 2023). Although various token reduction techniques help to accelerate ViT processing, while maintaining a high level of accuracy, there have also been improvements on the architectural level as well.

A straightforward approach to reduce computational complexity is to decrease the depth and width of ViT. The depth of ViT can be decreased by simply dropping transformer layers. However, this often causes a significant reduction in performance, as deeper layers play a crucial role in refining feature representations, modeling long-range dependencies, and improving robustness. Decreasing layers depth leads to less effective feature extraction and a reduced capacity to capture long-range dependencies, thereby lowering classification accuracy. For example, in the original ViT (Dosovitskiy et al., 2021), the early layers focus on extracting low-level features such as edges and textures, while the deeper layers gradually build high-level semantic understanding. The width of ViT can be reduced by decreasing its embedding dimension, which is also known as hidden size. However, this often has a notable impact on their performance by influencing the model's ability to learn and represent features. A larger hidden width allows the model to capture more fine-grained details and complex patterns, leading to better feature representations. A comparison (Dosovitskiy et al., 2021) between two variants of the original ViT: ViT-Large and ViT-Base highlights the performance effect of the depth and width reduction. ViT-Large achieved the 87.1 % top-1 accuracy on ImageNet (Deng et al., 2009), which is three percentage points higher compared to the accuracy of ViT-Base of 84.1 %, when models were trained with the JFT-300M dataset (Sun et al., 2017). In this comparison, both models had 16 attention heads, but ViT-Large had 24 transformer layers with a hidden size of 1024, and ViT-Base had 12 transformer layers with a hidden size of 768.

An additional straightforward strategy to reduce computational complexity involves decreasing the number of attention heads in the model, since the cost of self-attention scales with the number of heads, which also increases memory requirements. However, the num-

ber of attention heads in ViTs plays a crucial role in its accuracy. More attention heads enable the model to capture diverse relationships between image patches. Each head focuses on different aspects of the image, allowing the model to learn both detailed and overall contextual information. This improves the model's ability to recognize complex patterns and textures. In general, increasing the number of attention heads in ViT improves its accuracy. However, beyond a certain threshold, the improvements diminish due to redundancy among the heads. Although more heads improve the learning capacity of ViT, they also introduce a greater computational complexity.

Optimized architectures often balance the number of heads with the depth and width of the models. Yu et al. (2022) introduced a *Width & Depth Pruning* framework to improve the efficiency of ViTs by simultaneously reducing both their width and depth dimensions using learnable pruning parameters. Experimental results on benchmark datasets indicate that the framework can significantly reduce the computational costs of standard ViTs such as the DeiT (Touvron, Cord, Douze, et al., 2021) and the Swin Transformer (Liu et al., 2021) with minimal accuracy loss.

Instead of the most straightforward approaches described above, researchers have introduced various architectural modifications to improve the efficiency of ViTs while maintaining or even improving their performance. In this chapter, we will explore some key strategies and innovations in the architecture of ViT that reduce their computational complexity. We will start with hierarchical vision transformers and then move to sparse attention mechanisms. The chapter does not aim to introduce every possible architectural improvement out there, but rather to give a better understanding how architecture of ViTs can be improved and introduce some of the recent innovations at a higher level. There are many other studies and architectural techniques to improve ViT efficiency and performance, for example hybrid models such as *Convolutional Vision Transformer* (H. Wu et al., 2021), *Convolution-enhanced Image Transformer* (Yuan et al., 2021) and *BoTNet* (Srinivas et al., 2021), but those are scoped out.

5.1 Hierarchical Vision Transformers

Although ViT demonstrates robust performance, it has a significant limitation due to its high computational demands, particularly due to the quadratic complexity associated with self-attention. This makes ViTs computationally expensive, especially for high-resolution images. Furthermore, ViT has much less image-specific inductive bias than CNNs (Doso-

vitskiy et al., 2021). As a result, ViTs often require extensive pre-training to achieve competitive performance on downstream tasks.

Hierarchical Vision Transformers (HVTs) have been developed to address these issues with the efficiency and scalability concerns of ViTs. Unlike the original ViT, which maintains a fixed-scale tokenization throughout the network, hierarchical models introduce a multi-stage approach that progressively reduces the number of tokens processed in the deeper transformer layers of ViT while increasing feature richness. This architecture mimics the feature extraction mechanism found in CNNs, improving computational efficiency and the suitability of the model for *dense prediction* tasks. Dense prediction refers to a task in which the model generates an output for every pixel or region of an input image, rather than a single global prediction. Object detection and segmentation are examples of dense prediction tasks. HVTs operate in multiple stages, where each stage processes a progressively lower resolution representation while increasing the number of feature channels. This hierarchical representation allows for better semantic understanding at different levels. Hierarchical models often introduce windowed self-attention mechanism to improve locality and structural information. Several HVTs models have been developed to improve the efficiency and scalability of ViT. In the following, we explore a few of them to give a better understanding of how HVTs work in practice.

Swin Transformer Liu et al. (2021) introduced a new ViT model, called the *Swin Transformer*, which is a hierarchical vision transformer that can serve as a general-purpose backbone for computer vision. Unlike traditional self-attention mechanisms, Swin Transformer computes it's representation with shifted windows. The shifted window scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also facilitating connections between these windows. This process is illustrated in Figure 5.1.

In addition to the shifted windows, the Swin Transformer builds hierarchical feature maps by merging neighboring tokens at deeper layers, which is illustrated in Figure 5.2. This method achieves linear computational complexity with respect to the image size, thereby significantly enhancing efficiency when dealing with high-resolution input.

The performance of the model exceeds the previous state-of-the-art by a large margin of +2.7 box Average Precision (AP) and +2.6 mask AP on COCO (Lin et al., 2014) and +3.2 mean Intersection over Union (mIoU) on ADE20K (Zhou et al., 2017). It also achieves competitive 87.3 % top-1 accuracy on the ImageNet-1k classification (Deng et al., 2009).

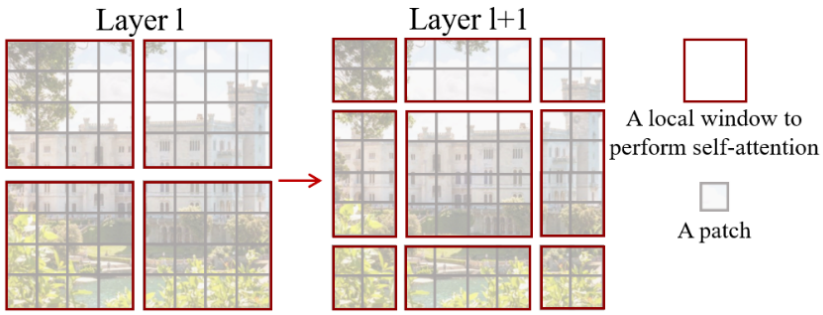

Figure 5.1: An illustration of the shifted window approach for computing self-attention in the proposed Swin Transformer architecture (Liu et al., 2021). In layer l, a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$, the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

These results demonstrate the potential of Transformer-based models, especially the Swin Transformer, as a general-purpose backbone for computer vision tasks.

Pyramid Vision Transformer W. Wang et al. (2021) introduced the *Pyramid Vision Transformer* (PVT), which is an improved ViT architecture designed to address some of the limitations of the original ViT. It is designed to serve as a versatile backbone for dense prediction tasks without relying on convolutional operations. PVT combines the strengths of both CNN and Transformer, making it a unified backbone in various vision tasks without convolution by simply replacing CNN backbones. Instead of processing all patches at a fixed resolution, PVT progressively reduces the spatial size of feature maps, forming a pyramid structure. This hierarchical method allows PVT to efficiently capture both local and global details.

The entire model is divided into four stages, each of which consists of a patch embedding layer and a L_i-layer Transformer encoder. The overall architecture of the model is illustrated in Figure 5.3. PVT employs a systematic reduction approach to manage the scales of the feature map through patch embedding layers. In the first stage, an input image of size $H \times W \times 3$ is divided into $HW/4^2$ patches, each of size $4 \times 4 \times 3$. Then, the flattened patches are fed to a linear projection and the embedded patches of size $HW/4^2 \times C1$ are obtained. Finally, the embedded patches along with a position embedding are passed through a Transformer encoder with $L1$ layers, and the output is re-shaped to a feature map $F1$ of size $H/4 \times W/4 \times C1$. Using the feature map from the previous stage as input, the following feature maps are obtained for each stage, whose strides are 8, 16, and 32

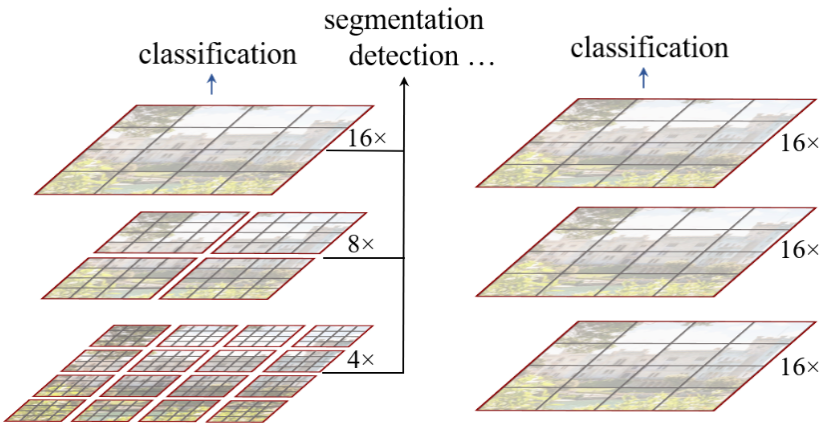

Figure 5.2: Left) The Swin Transformer builds hierarchical feature maps by merging image patches, shown in gray, in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window, which are shown in red (Liu et al., 2021). Right) The original ViT with quadratic computation complexity due to computation of self-attention globally.

pixels with respect to the input image.

Instead of using a traditional multi-head attention (MHA) layer in the encoder (Dosovitskiy et al., 2021), a spatial-reduction attention (SRA) layer is used. Similarly to MHA, the proposed SRA receives a query Q, a key K, and a value V vectors, but reduces the spatial scale of K and V before the attention operation. This optimization largely reduces computational and memory overhead while maintaining good performance, making the model more scalable, especially for high-resolution images.

Extensive experiments demonstrate PVT's effectiveness across various tasks. For example, when integrated with the RetinaNet (Lin et al., 2020) framework for object detection, PVT achieved an Average Precision (AP) of 40.4 on the COCO dataset (Lin et al., 2014). This surpasses the performance of the ResNet50 (He et al., 2016) + RetinaNet backbone by 4.1 absolute AP. These results highlight PVT's potential as a robust and efficient alternative to traditional convolutional backbones in vision applications.

5.2 Sparse Attention Mechanisms

The traditional self-attention mechanism computes attention scores across all pairs of tokens, leading to a quadratic scaling of the attention mechanism with respect to the number of input tokens. This results in excessive memory usage and computational cost when dealing with high-resolution images and large datasets. To solve the quadratic scaling

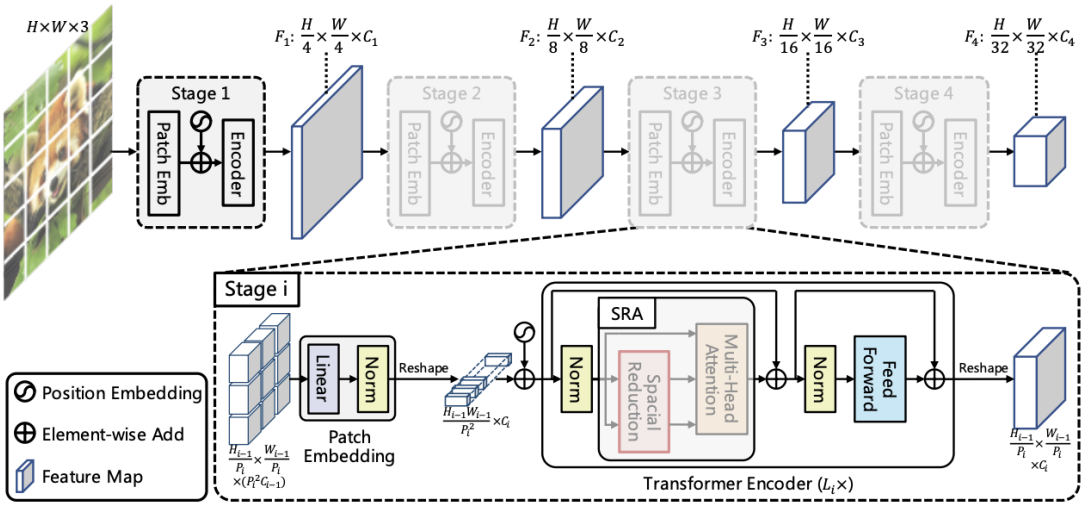

Figure 5.3: Overall architecture of PVT (W. Wang et al., 2021). The model is divided into four stages, each of which is comprised of a patch embedding layer and a L_i-layer Transformer encoder. Following a pyramid structure, the output resolution of the four stages progressively shrinks from high (4-stride) to low (32-stride).

issue with the self-attention mechanism, researchers have developed various techniques for the sparse attention mechanism, which focus on either restricting attention to specific token subsets or approximating the full attention matrix. Sparse attention mechanisms approximate or constrain attention computation to improve efficiency while maintaining strong performance. By reducing the number of token interactions, these mechanisms allow ViTs to scale effectively to larger images and datasets.

In the following, we explore a few techniques that have aimed to reduce the quadratic complexity of the standard self-attention mechanism (Vaswani et al., 2017) of the Transformer. Although these methods were originally developed for traditional transformers (NLP), Jeevan and Sethi (2022) explored the integration of these into ViT in their study. The study illustrated that replacing the standard self-attention mechanism in ViTs with these efficient attention variants can substantially decrease the use of computational resources while preserving or even improving image classification performance. These sparse attention mechanisms significantly reduce memory and computational demands, making ViTs more feasible for real-world applications that require high-resolution image processing, such as medical imaging, autonomous driving, and video processing in resource-constrained environments.

Linformer S. Wang et al. (2020) introduced the *Linformer* model, which significantly improves the efficiency of self-attention in Transformers. Instead of computing full atten-

tion maps, Linformer employs a low-rank projection strategy to approximate self-attention. This is achieved by learning smaller projection matrices that compress the key and value representations, reducing the space and time complexity from $O(n^2)$ to $O(n)$, with respect to the sequence length n. This reduction in complexity allows efficient self-attention calculations for high-definition images without significant accuracy loss. Experiments demonstrate that Linformer maintains performance comparable to standard Transformers while requiring significantly fewer resources (Jeevan and Sethi, 2022).

Performer Choromanski et al. (2021) introduced the *Performers* architecture, which improves the scalability of Transformers by estimating full-rank softmax attention in linear time and space complexity. To achieve this, Performers use a novel Fast Attention Via positive Orthogonal Random features (FAVOR+) approach, which approximates the standard softmax attention (Vaswani et al., 2017) using kernel methods. FAVOR+ can also be used to efficiently model kernelizable attention mechanisms beyond softmax. Linear time and space complexity enables Performers to handle longer sequences more efficiently without sacrificing model accuracy. Empirical tests across various NLP tasks demonstrate that Performers achieve performance comparable to traditional Transformers while offering significant improvements in computational efficiency.

Nyströmformer Xiong et al. (2021) introduced the *Nyströmformer* model, which takes advantage of the Nyström method (Williams and Seeger, 2000) to approximate the standard self-attention mechanism (Vaswani et al., 2017). This approximation reduces the computational complexity from quadratic to linear with respect to the sequence length. Just like the other two approaches mentioned above, this approach enables the processing of longer sequences more efficiently, expanding the applicability of Transformers to tasks involving extensive input. In the model, the self-attention mechanism is approximated using a subset of landmark points, which are selected to construct a low-rank approximation of the original attention matrix. This method significantly reduces memory and computational requirements while maintaining performance levels comparable to those of traditional Transformers. Empirical tests demonstrate that the Nyströmformer performs comparably, or in some cases even slightly better, than standard self-attention on the GLUE benchmarks (A. Wang et al., 2019) and the IMDB reviews (Maas et al., 2011). In extended sequence tasks within the Long Range Arena benchmark (Tay et al., 2021), the Nyströmformer demonstrates competitive performance compared to other efficient self-attention techniques. These findings suggest that the Nyströmformer effectively balances

efficiency and accuracy, making it a viable solution for scaling Transformers to handle longer sequences.

6 Conclusions

Vision Transformers have emerged as a promising alternative to traditional CNNs, offering enhanced global context understanding through self-attention mechanisms. However, despite their advantages, they face scalability challenges due to high computational complexity, memory and hardware constraints, and their dependence on extensive training datasets. The thesis has explored the scalability aspects of ViTs, starting with their scalability limitations and concluding with potential solutions to enhance their efficiency. To address the first research question: *"What are the scalability limitations of Vision Transformers in real-time applications?"*, the thesis highlighted three primary scalability limitations of ViTs:

Computational complexity: The self-attention mechanism in traditional ViTs scales quadratically with respect to the number of input tokens. This makes high-resolution image processing computationally expensive, as the number of input tokens is directly proportional to the resolution of the image. This quadratic scaling poses a significant challenge in real-time applications, especially in resource-constrained environments.

Hardware and memory constraints: Beyond computational cost, ViTs face significant hardware and memory constraints, particularly during training. Training a traditional ViT model requires high-performance computing clusters, often composed of hundreds of GPUs or TPUs that run for extended periods. These are not always available to all researchers or smaller organizations because of their availability and investment costs. Physical space and weight restrictions further limit the feasibility of deploying large-scale ViTs in UAVs and other mobile platforms. These devices often need to balance computational performance with energy efficiency and heat dissipation, making it harder to choose hardware only based on memory and computational needs.

Dependence on large-scale datasets: Unlike CNNs, which take advantage of inductive biases to learn effectively from smaller datasets, ViTs require extensive labeled datasets for training. This dependence increases training costs together with energy consumption and raises concerns about data availability and bias.

Although ViTs have been shown to be a promising advancement in computer vision, offering robust performance for image recognition and other vision-related applications, the scalability challenges must be addressed to enable widespread adoption. By optimizing computational efficiency, fine-tuning architectures, and improving data efficiency, ViTs can become a viable solution for a wide range of applications, from real-time processing to their usage in resource-constrained environments.

To address the second research question: *"How can Vision Transformer architectures be optimized to reduce computational costs while maintaining high accuracy in large-scale image recognition tasks?"*, the thesis has explored various strategies that enhance the scalability of ViTs. It started by going through various training strategies that improve the efficiency of training and reduce ViT's dependence on large datasets. Improved models and methods such as the Data-efficient Image Transformer, knowledge distillation, advanced data augmentation techniques, and hybrid approaches that integrate CNN features contribute to reducing dataset dependency and improving training efficiency. Token reduction techniques, such as token pruning, token pooling, and patch size optimization, have also been investigated to minimize the number of tokens processed per image. These techniques help to reduce ViT's computational complexity while balancing accuracy and efficiency, making it more viable for real-life applications. Additionally, architectural improvements, including sparse attention mechanisms and hierarchical ViTs, offer also promising solutions to reduce computational overhead while preserving model accuracy.

The studies explored within the thesis have shown promising results, but none of them alone can solve all the scalability limitations of ViTs. To see the full potential of the optimization methods explored, more combined studies will be needed. Those studies would tell whether these methods could be combined effectively without sacrificing too much accuracy compared to the original studies. Although the thesis primarily focused on the scalability factors of ViTs, it is also crucial to emphasize the importance of the prediction accuracy of the ViT model. Some of the ViT variants have been able to beat previous state-of-the-art models on predictions, but we have not yet seen a model that outperforms all existing models on all computer vision tasks. Therefore, it is also important to invest in improving the prediction accuracy of ViTs in various computer vision tasks if we want ViT to serve as a general-purpose model in computer vision. The continuous research and development of ViTs will define whether they can surpass all CNN models in accuracy and applicability, potentially leading to the next generation in computer vision.

Acknowledgments

Large language models and AI tools such as ChatGPT and Writefull have been used to support writing process of this thesis. These tools were used for grammar and style checks, synonym suggestions, and paraphrasing, all focused on enhancing the text's clarity and coherence. Additionally, I have also used them for brainstorming ideas related to chapter structure, obtaining early-stage feedback on content organization and identifying potential gaps or missing viewpoints. Although these tools were valuable in improving the quality and presentation of the thesis, all substantive content, analysis, and conclusions remain my own.

Bibliography

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). "Layer Normalization". In: *CoRR* abs/1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations (ICRL)*.

Bao, H., Dong, L., Piao, S., and Wei, F. (2022). "BEiT: BERT Pre-Training of Image Transformers". In: *10th International Conference on Learning Representations (ICRL)*.

Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. (2021). "Understanding Robustness of Transformers for Image Classification". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10211–10221. DOI: 10.1109/ICCV48922.2021.01007.

Bolya, D., Fu, C., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. (2023). "Token Merging: Your ViT But Faster". In: *11th International Conference on Learning Representations (ICRL)*.

Bonnaerens, M. and Dambre, J. (2023). "Learned Thresholds Token Merging and Pruning for Vision Transformers". In: *Transactions on Machine Learning Research (TMLR)* 2023.

Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). "Emerging Properties in Self-Supervised Vision Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9630–9640. DOI: 10.1109/ICCV48922.2021.00951.

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. (2021). "Rethinking Attention with Performers". In: *9th International Conference on Learning Representations (ICRL)*.

CNN (2025). *Biden unveils last round of AI chip curbs aimed at China, Russia*. URL: https://edition.cnn.com/2025/01/13/tech/china-us-biden-chips-ai-curbs-hnk-intl/index.html (visited on 02/23/2025).

Cordonnier, J., Loukas, A., and Jaggi, M. (2020). "On the Relationship between Self-Attention and Convolutional Layers". In: *8th International Conference on Learning Representations (ICRL)*.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). "Randaugment: Practical automated data augmentation with a reduced search space". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3008–3017. DOI: 10.1109/CVPRW50498.2020.00359.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *9th International Conference on Learning Representations (ICRL)*.

Fu, A., Hosseini, M. S., and Plataniotis, K. N. (2021). "Reconsidering co2 emissions from computer vision". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2311–2317.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). "Convolutional sequence to sequence learning". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 1243–1252. DOI: 10.5555/3305381.3305510.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press. ISBN: 9780262035613.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). "Knowledge distillation: A survey". In: *International Journal of Computer Vision* 129, pp. 1789–1819. DOI: 10.1007/s11263-021-01453-z.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). "Masked Autoencoders Are Scalable Vision Learners". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988. DOI: 10.1109/CVPR52688.2022.01553.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). "Deep networks with stochastic depth". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, pp. 646–661. DOI: 10.1007/978-3-319-46493-0_39.

Ioffe, S. and Szegedy, C. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr, pp. 448–456. DOI: 10.5555/3045118.3045167.

Jeevan, P. and Sethi, A. (2022). "Resource-efficient Hybrid X-formers for Vision". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3555–3563. DOI: 10.1109/WACV51458.2022.00361.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2022). "Transformers in vision: A survey". In: *ACM computing surveys (CSUR)* 54.10s, pp. 1–41. DOI: 10.1145/3505244.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). "Big transfer (BiT): General visual representation learning". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, pp. 491–507. DOI: 10.1007/978-3-030-58558-7_29.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Liang, Y., Ge, C., Tong, Z., Song, Y., Wang, J., and Xie, P. (2022). "EViT: Expediting Vision Transformers via Token Reorganizations". In: *10th International Conference on Learning Representations (ICLR)*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2020). "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2, pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). "Microsoft coco: Common objects in context". In: *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*. Springer, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *2021*

IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9992–10002. DOI: 10.1109/ICCV48922.2021.00986.

Lu, Z., Xie, H., Liu, C., and Zhang, Y. (2022). "Bridging the gap between vision transformers and convolutional neural networks on small datasets". In: *Advances in Neural Information Processing Systems* 35, pp. 14663–14677.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). "Learning word vectors for sentiment analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pp. 142–150. DOI: 10.5555/2002472.2002491.

Marin, D., Chang, J.-H. R., Ranjan, A., Prabhu, A., Rastegari, M., and Tuzel, O. (2023). "Token Pooling in Vision Transformers for Image Classification". In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 12–21. DOI: 10.1109/WACV56688.2023.00010.

NVIDIA Corporation (2019). *NVIDIA Jetson Nano Developer Kit*. URL: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (visited on 04/14/2025).

OpenAI (2022). *ChatGPT: Optimizing Language Models for Dialogue*. URL: https://openai.com/blog/chatgpt (visited on 09/15/2024).

Pan, B., Panda, R., Jiang, Y., Wang, Z., Feris, R., and Oliva, A. (2021). "IA-RED^2 : $Interpretability-awareredundancyreductionforvisiontransformers$". In: *Advances in neural information processing systems* 34, pp. 24898–24911.

Psomas, B., Kakogeorgiou, I., Karantzalos, K., and Avrithis, Y. (Oct. 2023). "Keep It SimPool:Who Said Supervised Transformers Suffer from Attention Deficit?" In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, pp. 5327–5337. DOI: 10.1109/ICCV51070.2023.00493.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). "Improving language understanding by generative pre-training". In: *OpenAI blog*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *Journal of machine learning research* 21.140, pp. 1–67.

Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., and Hsieh, C.-J. (2021). "Dynamicvit: Efficient vision transformers with dynamic token sparsification". In: *Advances in neural information processing systems* 34, pp. 13937–13949.

Raspberry Pi Foundation (2019). *Raspberry Pi 4 Model B*. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b (visited on 04/14/2025).

Shorten, C. and Khoshgoftaar, T. M. (2019). "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1, pp. 1–48. DOI: 10.1186/s40537-019-0197-0.

Srinivas, A., Lin, T.-Y., Parmar, N., Shlens, J., Abbeel, P., and Vaswani, A. (2021). "Bottleneck Transformers for Visual Recognition". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16514–16524. DOI: 10.1109/CVPR46437.2021.01625.

Strubell, E., Ganesh, A., and McCallum, A. (2019). "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 3645–3650. DOI: 10.18653/v1/P19-1355.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 843–852. DOI: 10.1109/ICCV.2017.97.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). "Sequence to sequence learning with neural networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. Advances in Neural Information Processing Systems (NeurIPS). MIT Press, pp. 3104–3112. DOI: 10.5555/2969033.2969173.

Tan, M. and Le, Q. (2019). "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR, pp. 6105–6114.

Tang, Y., Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., and Tao, D. (2022). "Patch Slimming for Efficient Vision Transformers". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12155–12164. DOI: 10.1109/CVPR52688.2022.01185.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. (2021). "Long Range Arena : A Benchmark for Efficient Transformers". In: *9th International Conference on Learning Representations (ICLR)*.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). "Training data-efficient image transformers & distillation through attention". In: *International conference on machine learning*. PMLR, pp. 10347–10357.

Touvron, H., Cord, M., and Jégou, H. (2022). "DeiT III: Revenge of the ViT". In: *European conference on computer vision*. Springer, pp. 516–533. DOI: 10.1007/978-3-031-20053-3_30.

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jégou, H. (2021). "Going deeper with Image Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42. DOI: 10.1109/ICCV48922.2021.00010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Curran Associates Inc., pp. 6000–6010. DOI: 10.5555/3295222.3295349.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *7th International Conference on Learning Representations (ICRL)*.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). "Linformer: Self-Attention with Linear Complexity". In: *CoRR* abs/2006.04768.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. (2021). "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 548–558. DOI: 10.1109/ICCV48922.2021.00061.

Wei, S., Ye, T., Zhang, S., Tang, Y., and Liang, J. (2023). "Joint Token Pruning and Squeezing Towards More Aggressive Compression of Vision Transformers". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2092–2101. DOI: 10.1109/CVPR52729.2023.00208.

Williams, C. K. I. and Seeger, M. (2000). "Using the Nyström method to speed up kernel machines". In: *Proceedings of the 14th International Conference on Neural Information Processing Systems*. NIPS'00. MIT Press, pp. 661–667. DOI: 10.5555/3008751.3008847.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. (2021). "CvT: Introducing Convolutions to Vision Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22–31. DOI: 10.1109/ICCV48922.2021.00009.

Wu, K., Zhang, J., Peng, H., Liu, M., Xiao, B., Fu, J., and Yuan, L. (2022). "Tinyvit: Fast pretraining distillation for small vision transformers". In: *European conference on computer vision*. Springer, pp. 68–85. DOI: 10.1007/978-3-031-19803-8_5.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. (2021). "Nyströmformer: A nyström-based algorithm for approximating self-attention". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 16, pp. 14138–14148. DOI: 10.1609/aaai.v35i16.17664.

Xu, Y., Zhang, Z., Zhang, M., Sheng, K., Li, K., Dong, W., Zhang, L., Xu, C., and Sun, X. (2022). "Evo-vit: Slow-fast token evolution for dynamic vision transformer". In:

Proceedings of the AAAI conference on artificial intelligence. Vol. 36. 3, pp. 2964–2972. DOI: 10.1609/aaai.v36i3.20202.

Xue, F., Wang, Q., Tan, Z., Ma, Z., and Guo, G. (2023). "Vision Transformer With Attentive Pooling for Robust Facial Expression Recognition". In: *IEEE Transactions on Affective Computing* 14.4, pp. 3244–3256. DOI: 10.1109/TAFFC.2022.3226473.

Yin, H., Vahdat, A., Alvarez, J. M., Mallya, A., Kautz, J., and Molchanov, P. (2022). "A-ViT: Adaptive Tokens for Efficient Vision Transformer". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10799–10808. DOI: 10.1109/CVPR52688.2022.01054.

Yu, F., Huang, K., Wang, M., Cheng, Y., Chu, W., and Cui, L. (2022). "Width & depth pruning for vision transformers". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 3, pp. 3143–3151. DOI: 10.1609/aaai.v36i3.20222.

Yuan, K., Guo, S., Liu, Z., Zhou, A., Yu, F., and Wu, W. (2021). "Incorporating Convolution Designs into Visual Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 559–568. DOI: 10.1109/ICCV48922.2021.00062.

Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. (2019). "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6022–6031. DOI: 10.1109/ICCV.2019.00612.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). "mixup: Beyond Empirical Risk Minimization". In: *6th International Conference on Learning Representations (ICRL)*. OpenReview.net.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). "Scene Parsing through ADE20K Dataset". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5122–5130. DOI: 10.1109/CVPR.2017.544.