Global Context Vision Transformers (GC ViT)

Sofiia Ilnytska, Wiebke Teetz

Contents

Understanding the Foundations of Transformers

What Came Before Transformers?

1986

Introduction of Recurrent Neural Networks (RNNs) for sequential data processing in machine learning.

1997

Long Short-Term Memory (LSTM) networks developed to address RNNs' vanishing gradient problem.

2014

Gated Recurrent Units (GRUs) introduced, simplifying LSTM architecture while maintaining performance.

2015

Recognition of RNNs' limitations in parallelization, prompting the need for more efficient models.

2017

Emergence of Transformers, utilizing self-attention mechanisms to eliminate recurrence entirely.

2018

Transformers revolutionize deep learning, enabling effective processing of long-range dependencies in data.

What Are Transformers?

A **sequence transduction model**: It turns one sequence into another (e.g., a sentence in English → a sentence in French).

Built entirely on Attention mechanisms: No Recurrent Neural Networks (RNNs), no Convolutions.

A **Transformer** model is built from two main parts:
➡️ **Encoder** (on the left side)
➡️ **Decoder** (on the right side)

Both of them are made up of **layers** that are **stacked** on top of each other—imagine building blocks stacked into a tower. Each layer does two important things:

1. **Self-Attention**
2. **Point-wise, Fully Connected Layers**

   This is a simple neural network that looks at each part (word, image patch, etc.) **on its own**, and processes it further.

   "Point-wise" just means it works on each piece individually, without looking at its neighbors.

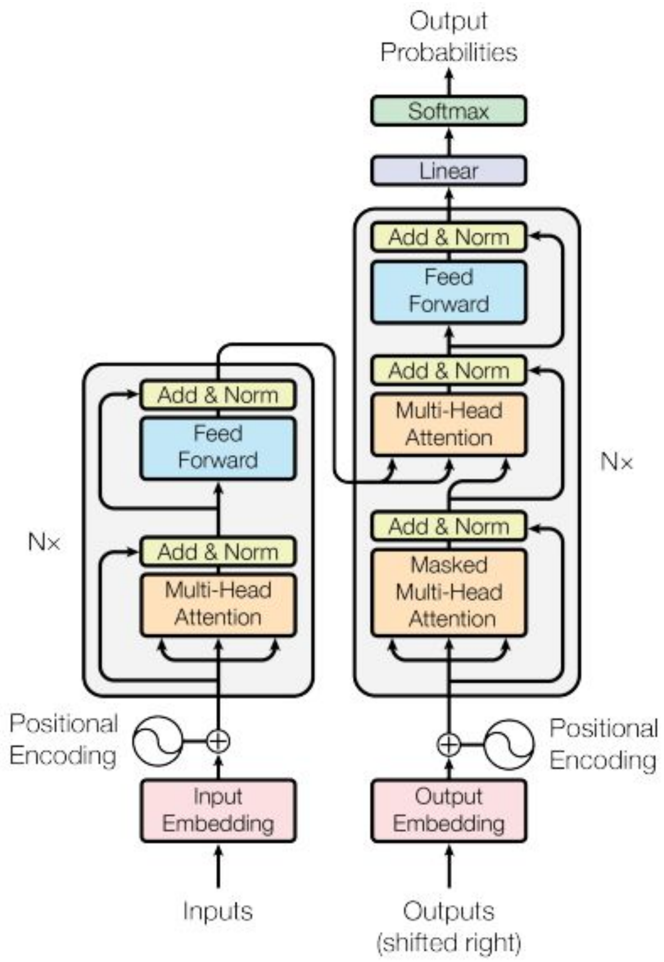


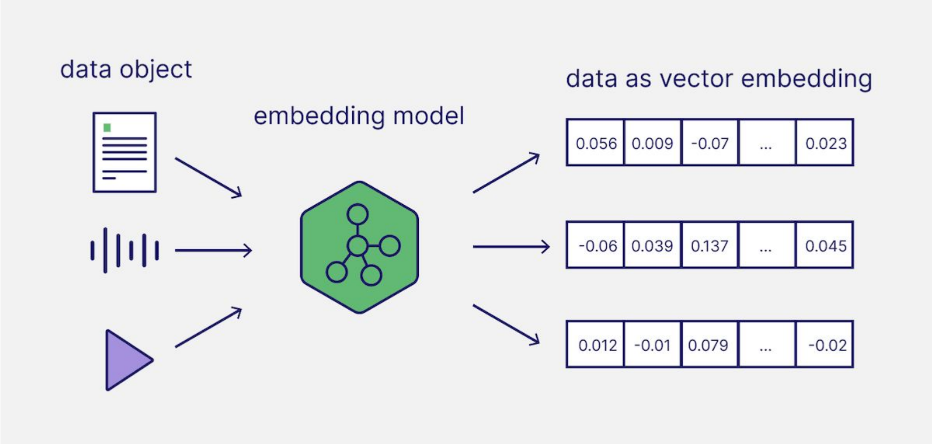
Figure 1: The Transformer - model architecture.

Input and Output Embedding in Transformers

Transformers don't work directly with words — they work with **numbers**!
So, before processing, words are converted into vectors called embeddings.

These vectors capture:

- Semantic Meaning: Similar words get similar vectors.
- Contextual Clues: Embeddings help the model grasp relationships between words.



Input Embedding:

- Converts words into dense numerical vectors that capture their meaning.
- Adds Positional Encoding to show word order.

Output Embedding:

- Converts the final vectors back into words or tokens.
- Uses a softmax layer to pick the most likely word.

Attention in transformers and its types

Attention is the mechanism that helps a model decide which parts of the input are important.
It weighs the relevance of each word when making predictions.

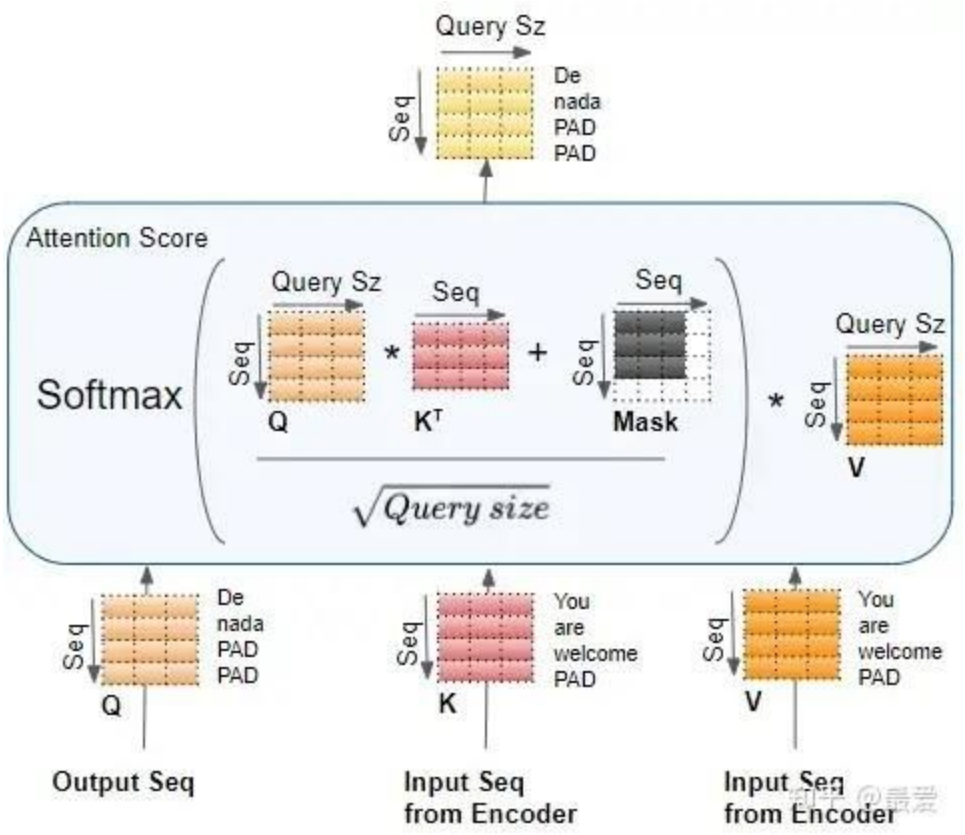
- Basic Attention
- Multi-Head Attention
- Masked Multi-Head Attention

Each word (or token) in the input is transformed into three vectors:

1. Query (Q): Represents what we're looking for.
2. Key (K): Represents what we have.
3. Value (V): Represents the actual information we use.

The model compares Queries with Keys to calculate attention scores, which determine how much Value to use from each word.

Multi-Head Attention helps the model look at different aspects of the input simultaneously, while Masked Attention makes sure the model doesn't cheat by looking ahead.



Query Sz
Seq
De
nada
PAD
PAD
Attention Score
Softmax
Query Sz
Seq
Seq
Seq
Seq
Q
K^T
Mask
$\sqrt{Query\ size}$
Query Sz
Seq
V
Seq
Q
De
nada
PAD
PAD
Seq
K
You
are
welcome
PAD
Seq
V
You
are
welcome
PAD
Output Seq
Input Seq
from Encoder
Input Seq
from Encoder

Encoder in Transformers: How It Works + BERT

- 6 stacked layers
- two important components (sub-layers):

  + Multi-Head Self-Attention
  + Feed-Forward Network (FFN)

- Each layer also has:

  + Residual Connections
  + Layer Normalization

BERT, (Bidirectional Encoder Representations from Transformers) a popular language model, only uses the encoder because it focuses on **understanding text** rather than generating it.

Decoder in Transformers: How It Works + GPT

- 6 stacked layers
- two important components (sub-layers):

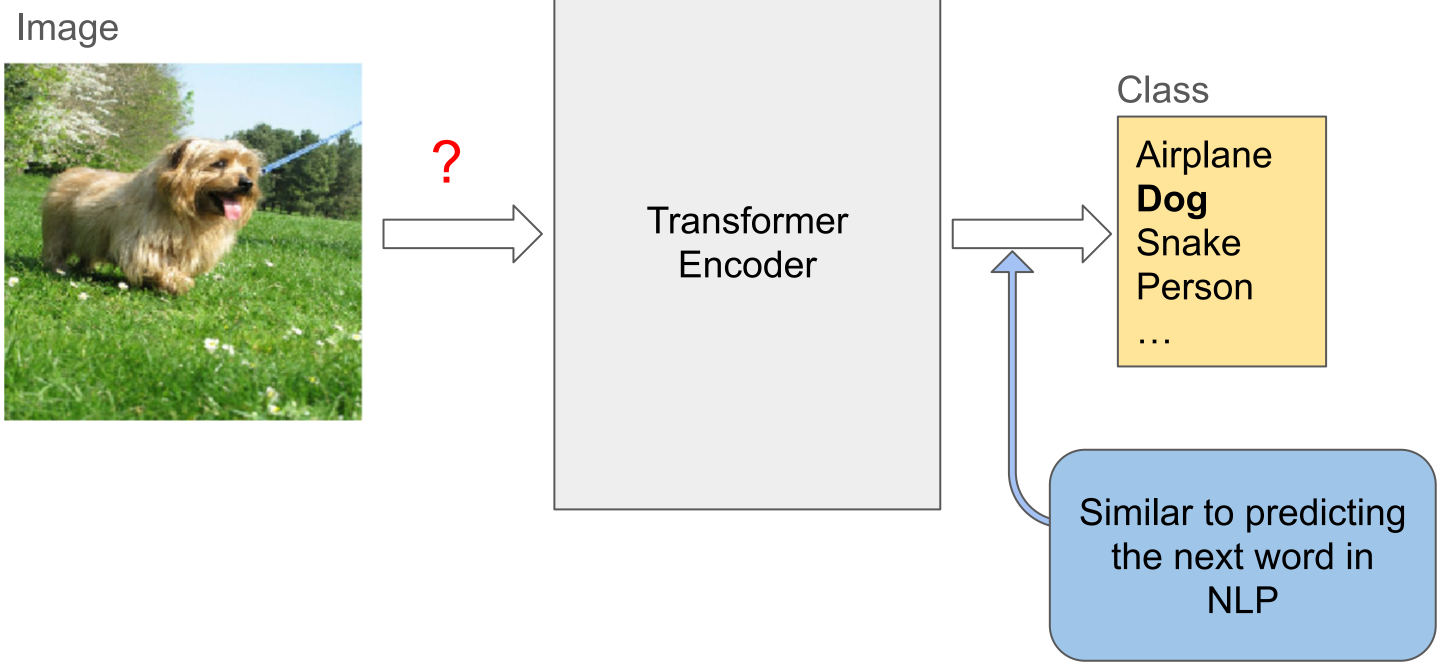
  + Multi-Head Self-Attention
  + Multi-Head Attention over Encoder Output
  + Feed-Forward Network (FFN)

- Each layer also has:

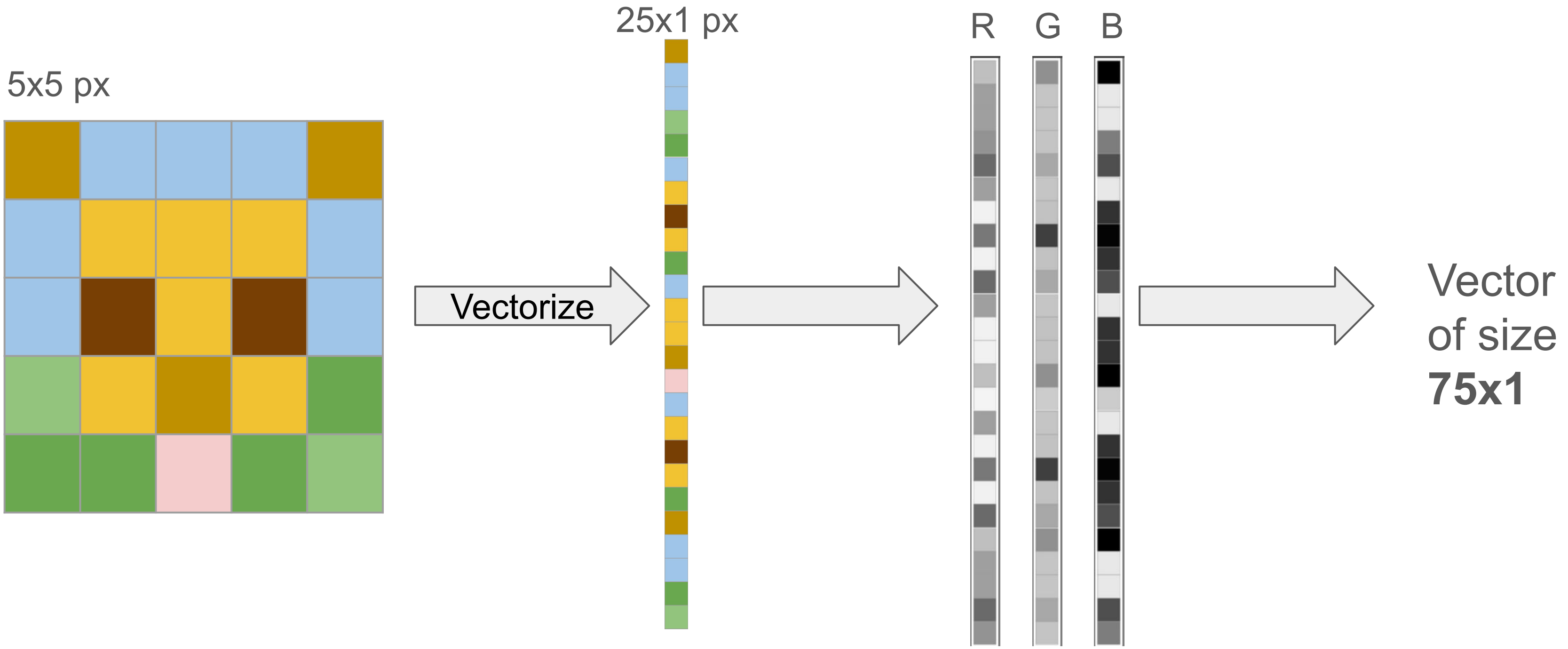
  + Residual Connections
  + Layer Normalization

GPT, (generative pre-trained transformer) a popular language model, only uses the decoder because it focuses on **text generation** rather than understanding.

Vision Transformers (ViT) for Image Classification



How does Embedding an Image work?



d1 x d2 x d3 x d4 tensor → d1*d2*d3*d4 x 1 tensor

How does Embedding an Image work at scale?

- Self-Attention is **expensive**!

  e.g. **224 x 224 px** →224*224*3 x 1 vector→ 150’528 x 150’528 Self-Attention Matrix
  = 150’528 x 1 ≈ **22.6 Billion**

- Transformers usually work with a **sequence of tokens** as input

How does Embedding an Image work <u>at scale</u>?

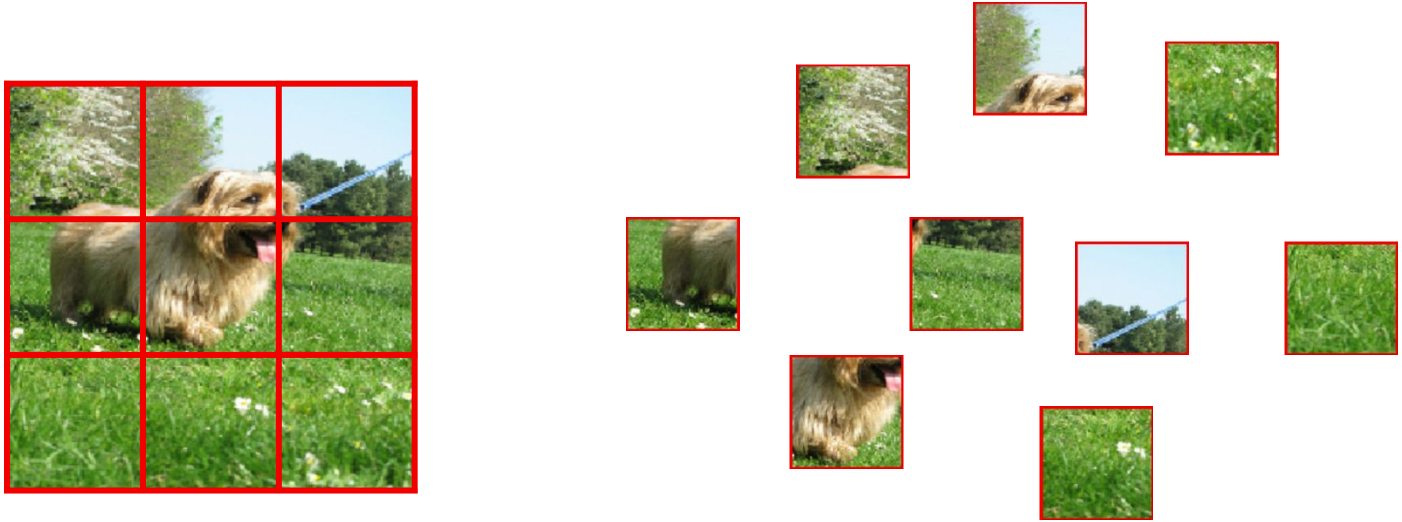
Published as a conference paper at ICLR 2021

AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*],
Xiaohua Zhai[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,**
Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby[*,†]
[*]equal technical contribution, [†]equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

How does Embedding an Image work <u>at scale</u>?

Split the Input into **patches** of size 16 x 16 px

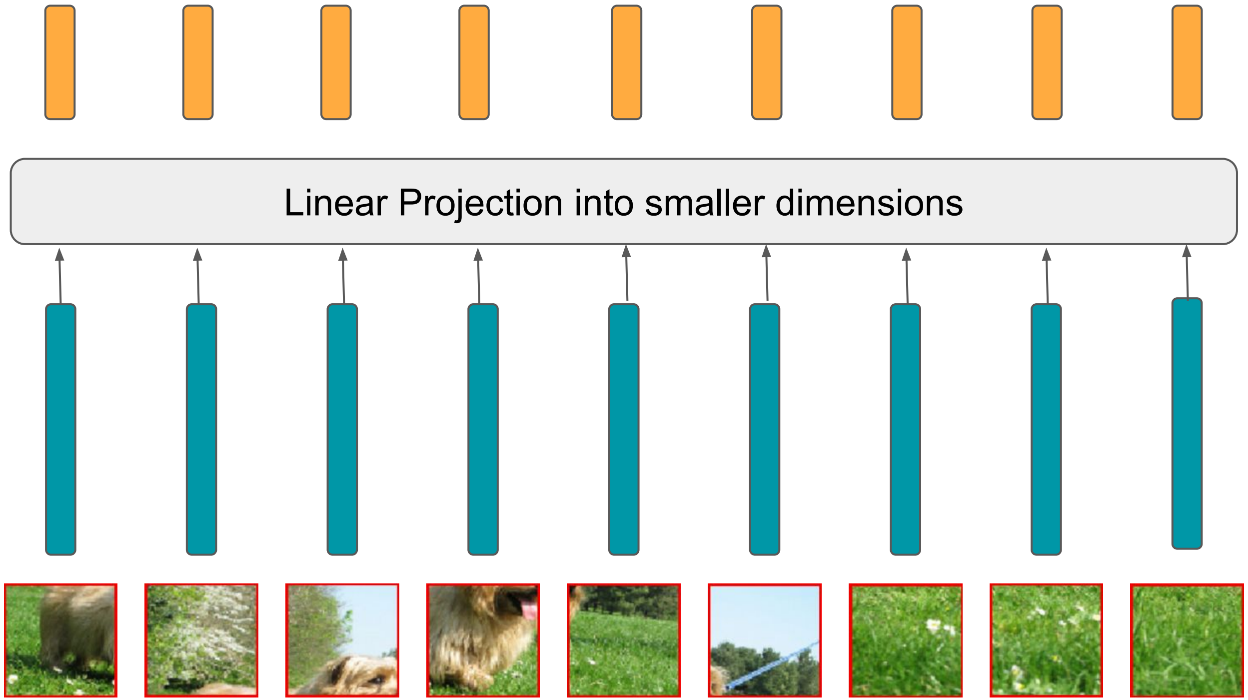


1 Patch (16x16) = 1 Token (= 1 Word)

How does Embedding an Image work at scale?

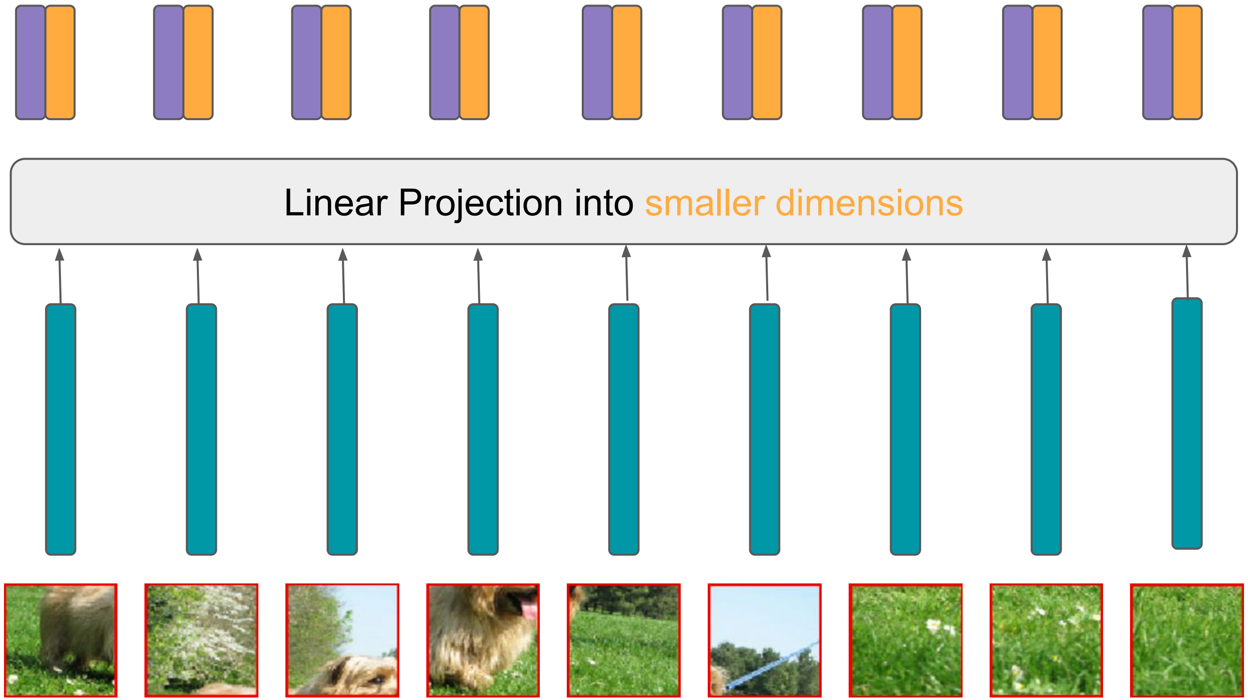
MatMul with
Embedding Matrix

Vectorize each patch



How does Embedding an Image work at scale?

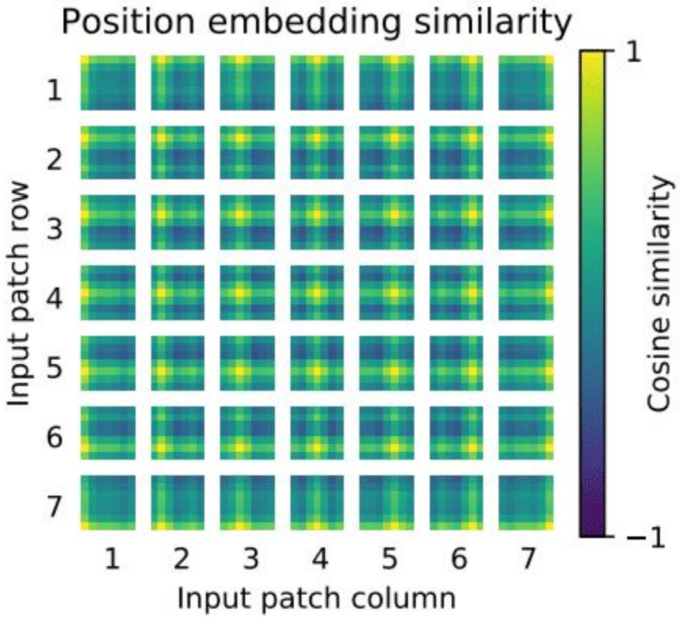
Add Positional Embeddings (trainable)



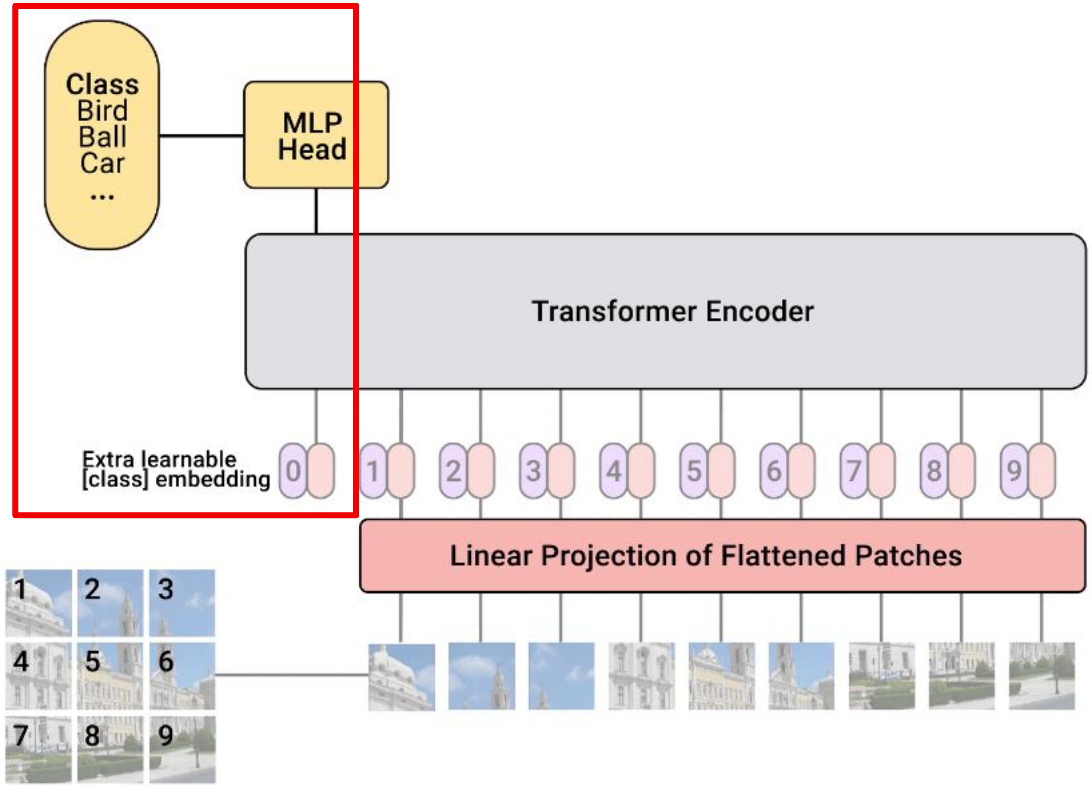
Vectorize each patch

Positional Embeddings

The transformer has no intrinsic understanding of where each patch is positioned in the original image



CLS Token

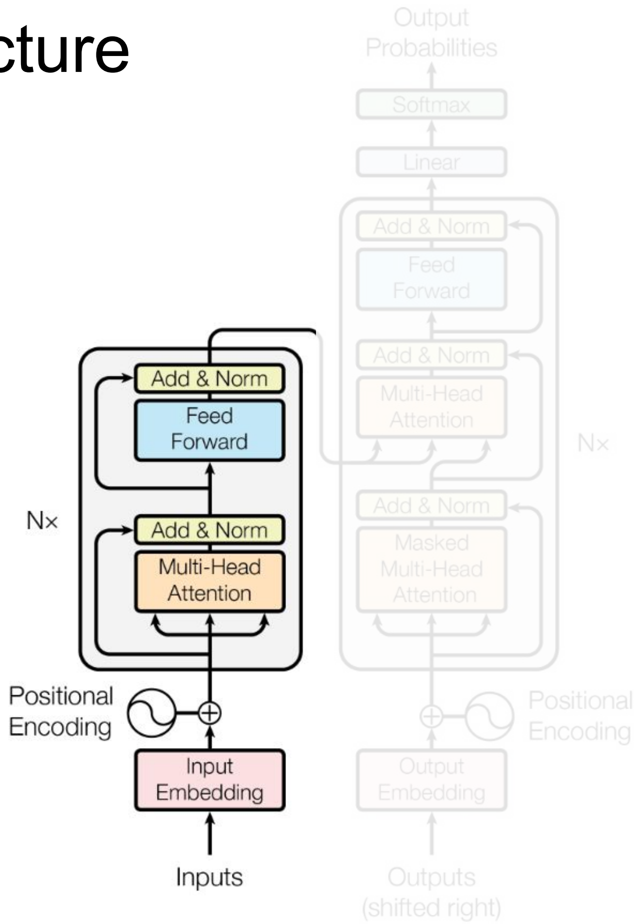


Class
Bird
Ball
Car
...
MLP
Head
Transformer Encoder
Extra learnable
[class] embedding
0
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
1
2
3
4
5
6
7
8
9

Overview of ViT Architecture

Combine infos of the patches

Self Attention on each individual patch



Original Transformer Architecture

Limitations of ViT

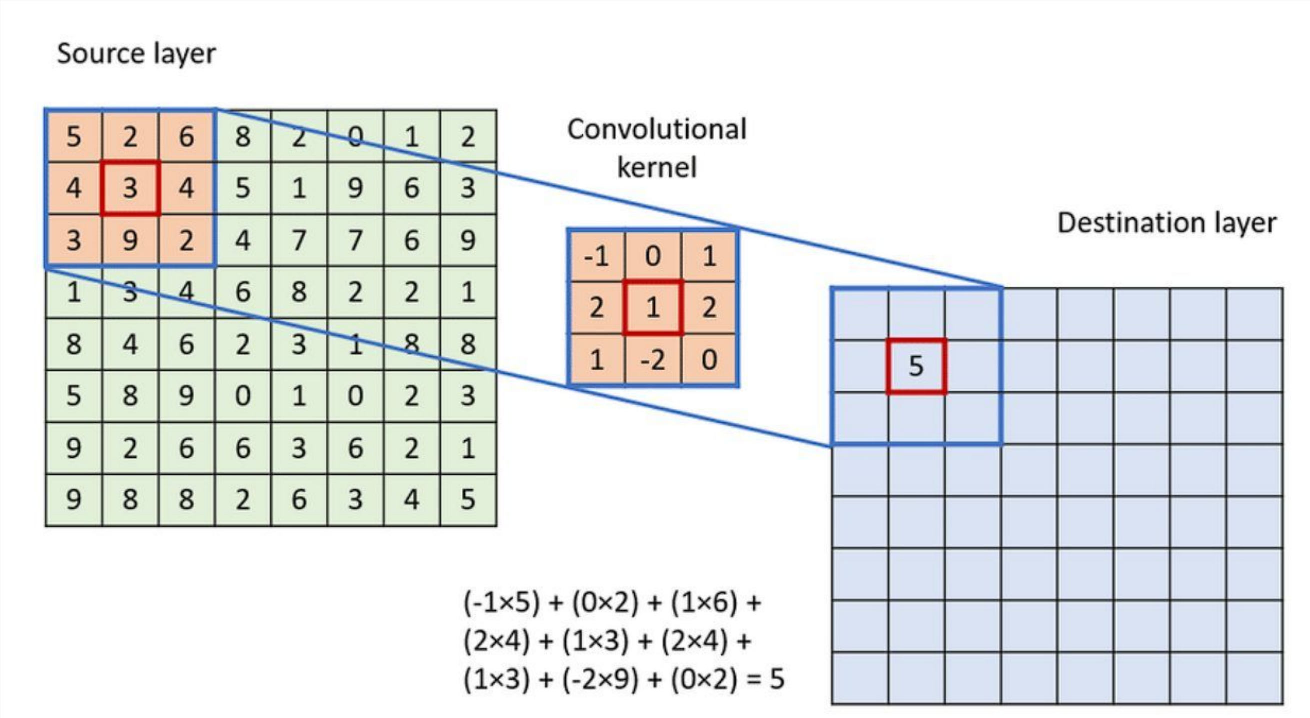
- relative computational inefficiency (scales quadratically with image size)
- ViT lack the inductive bias of convolutional neural networks (CNN)
  - locally restricted inductive fields
  - translation invariance

ViTs treat **all patches equally**

→lack of local feature emphasis (e.g. edges, texture)

→large-scale training data needed to outperform older models (e.g. CNNs)

What is convolution?



Limitations of ViT

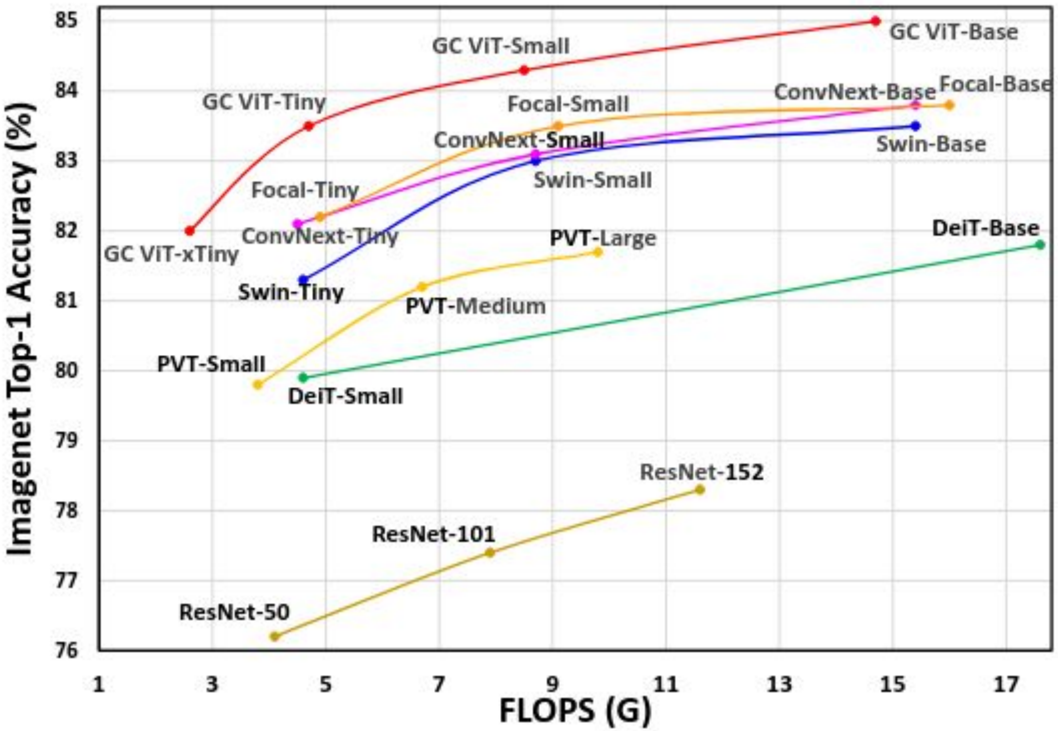
- relative computational inefficiency (scales quadratically with image size)
- ViT lack the inductive bias of convolutional neural networks (CNN)
    - locally restricted inductive fields
    - translation invariance

ViTs treat **all patches equally**

→lack of local feature emphasis (e.g. edges, texture)

→large-scale training data needed to outperform older models (e.g. CNNs)

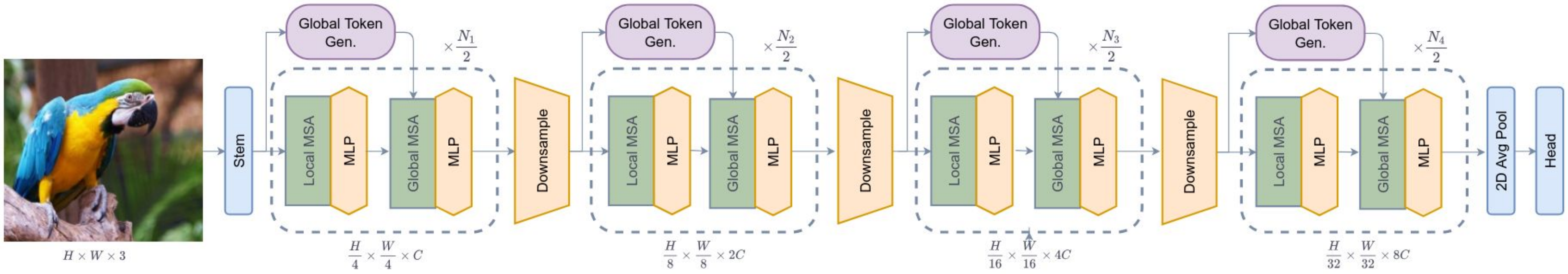
Global Context Visual Transformer (GC ViT)



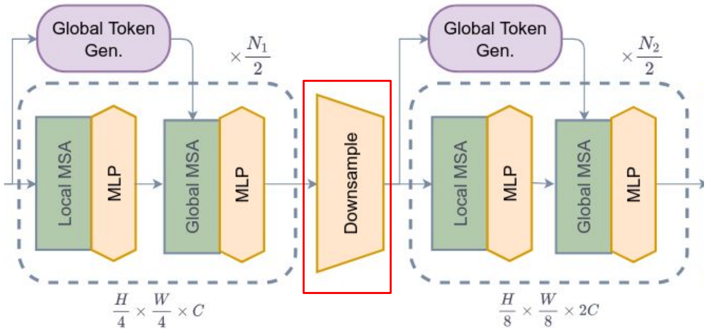
Global Context Visual Transformer (GC ViT)

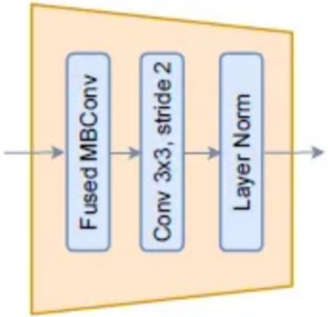
Main Adjustments:

- optimized number of parameters
- CNN like token generator for global queries
- downsampling module that integrates inductive bias



GC ViT - Downsampler

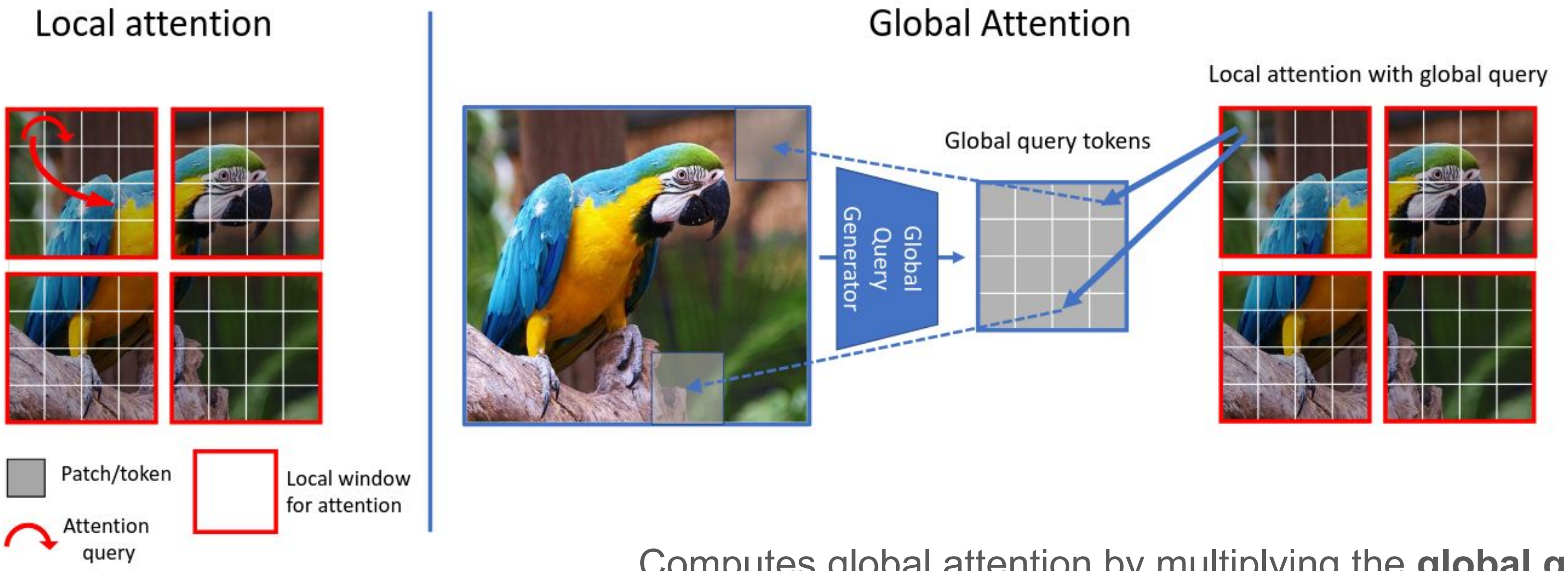




$$
\begin{aligned}
\hat{\mathbf{x}} &= \text{DW-Conv}_{3\times3}(\mathbf{x}), \\
\hat{\mathbf{x}} &= \text{GELU}(\hat{\mathbf{x}}), \\
\hat{\mathbf{x}} &= \text{SE}(\hat{\mathbf{x}}), \\
\mathbf{x} &= \text{Conv}_{1\times1}(\hat{\mathbf{x}}) + \mathbf{x},
\end{aligned}
$$

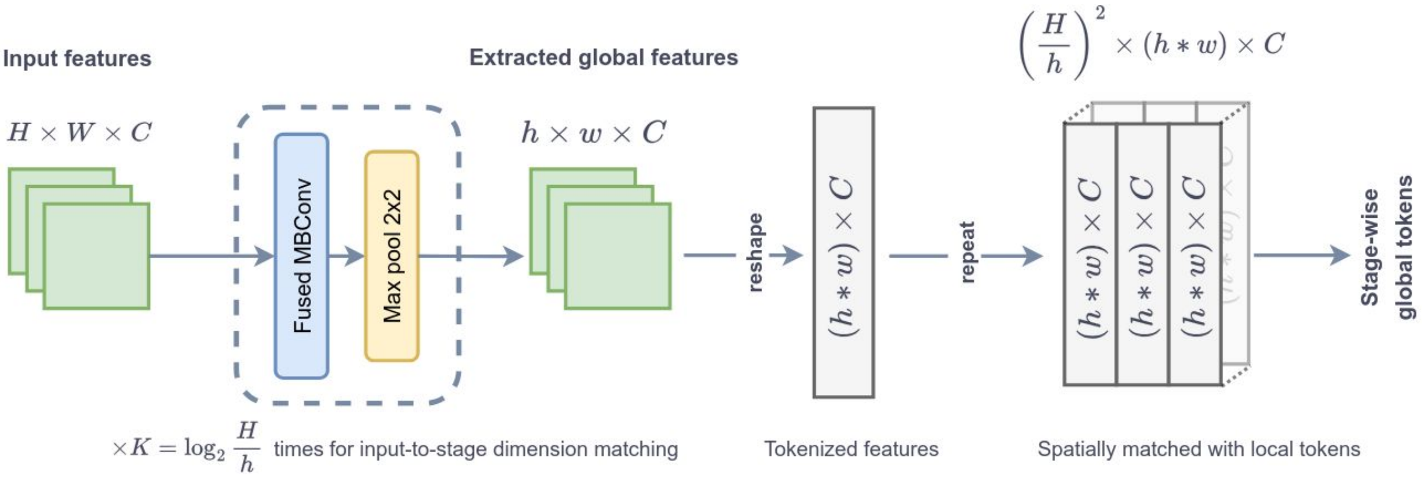
- applies convolution to efficiently capture local features
    - enforcing inductive biases
- reduces spatial resolution

GC ViT - Global and Local Attention

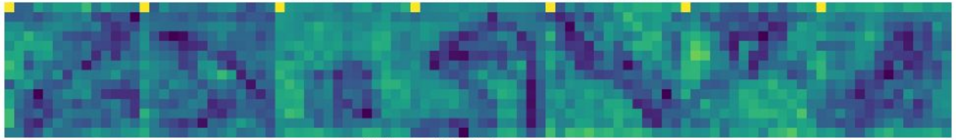


Computes global attention by multiplying the **global query** with each **local key** matrix

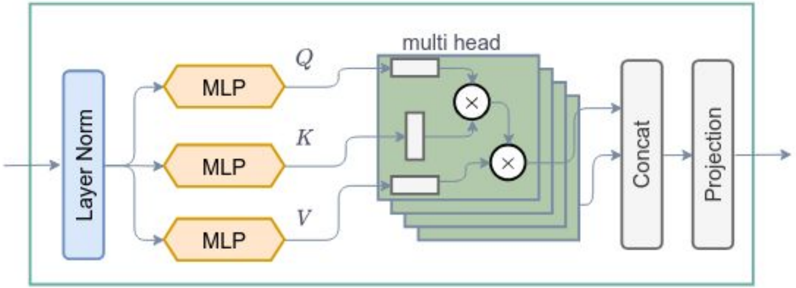
GC ViT - Global Query Generator



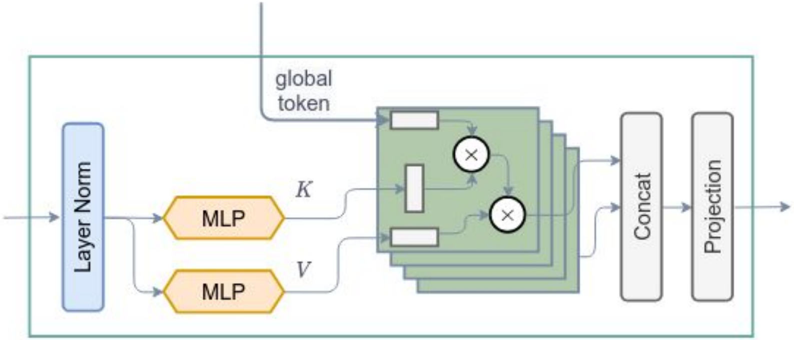
(a) Original images from ImageNet-1K validation set.



GC ViT - Local and Global Self-Attention

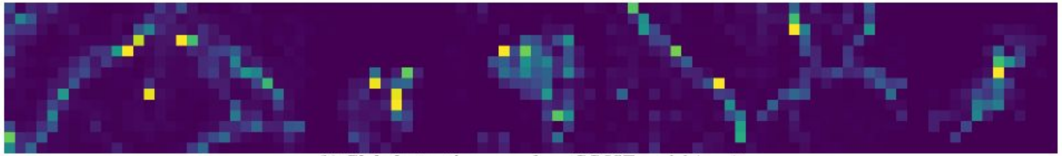


Local MSA



Global MSA

(a) Original images from ImageNet-1K validation set.



(b) **Global attention** maps from GC ViT model (ours).

Thank you

Any questions?