Computer Vision by Learning

Cees Snoek, University of Amsterdam

Efstratios Gavves, University of Amsterdam

With an invited tutorial by: Serge Belongie, University of Copenhagen

http://computervisionbylearning.info

Today's lectures

09:30-10:10	**Transformers**
10:10-10:30	*Break*
10:30-11:10	**Learning from limited data**
11:10-11:30	*Break*
11:30-12:10	**3D representation learning**

Subhransu Maji

Martin Oswald

Today's lab

ASCI CBL Practicals
latest
Search docs

PRACTICALS

» Practical 3: Vision Transformers

Edit on GitHub

Practical 3: Vision Transformers

Open notebook: Repo View On Github | Open in Colab
Authors: Phillip Lippe

In this practical, we will take a closer look at a recent new trend: Transformers for Computer Vision. Since Alexey Dosovitskiy et al. successfully applied a Transformer on a variety of image recognition benchmarks, there have been an incredible amount of follow-up works showing that CNNs might not be optimal architecture for Computer Vision anymore. But how do Vision Transformers work exactly, and what benefits and drawbacks do they offer in contrast to CNNs? We will answer these questions by implementing a Vision Transformer ourselves and train it on the popular, small dataset CIFAR10.

Let's start with importing our standard set of libraries.

[1]:
```
## Standard libraries
import os
import numpy as np
import random
import math
import json
from functools import partial
```

Abstract

Astonishing results from Transformer models on natural language tasks have intrigued the vision community to study their application to computer vision problems.

We start with an introduction to fundamental concepts behind the success of (language) Transformers. We then cover applications of transformers in vision for several popular recognition tasks.

Overview

1. **Transformer**, self-attention, multi-head attention, positional encoding
2. **Vision transformer**, patch token, classification token.
3. **Swin transformer**, shifted windows, vision backbone.
4. **Detector transformers,** DETR, box-attention, where-to-attend, 3D.

1. The Transformer

This chapter presents the Transformer network architecture. It is based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.

Many slides inspired by: https://jalammar.github.io/illustrated-transformer/

Vaswani et al. NIPS 2017

40,000+ citations

The (language) transformer

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*, †]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*, ‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Key insight: Attention suffices to derive input and output dependencies

Why are transformers so popular?

Pretrained Transformer models **adapt easily** and quickly to language tasks they have not been trained on.

Not limited to language related tasks, they quickly became useful for other modalities and problem domains.

Yields more **interpretable** models.

Hugging face **open sourced** transformers library (and raised 60M$).

Transformer family

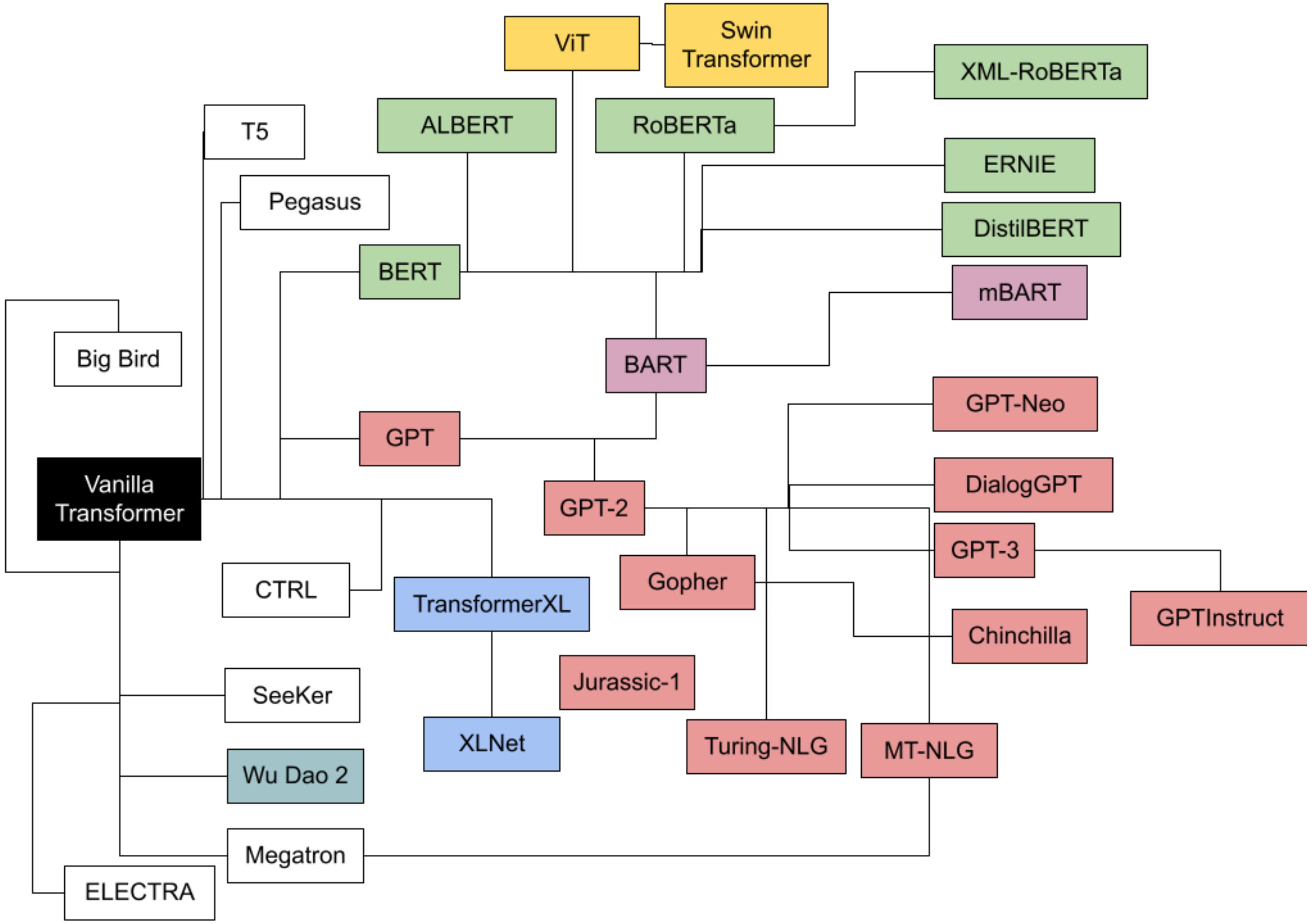
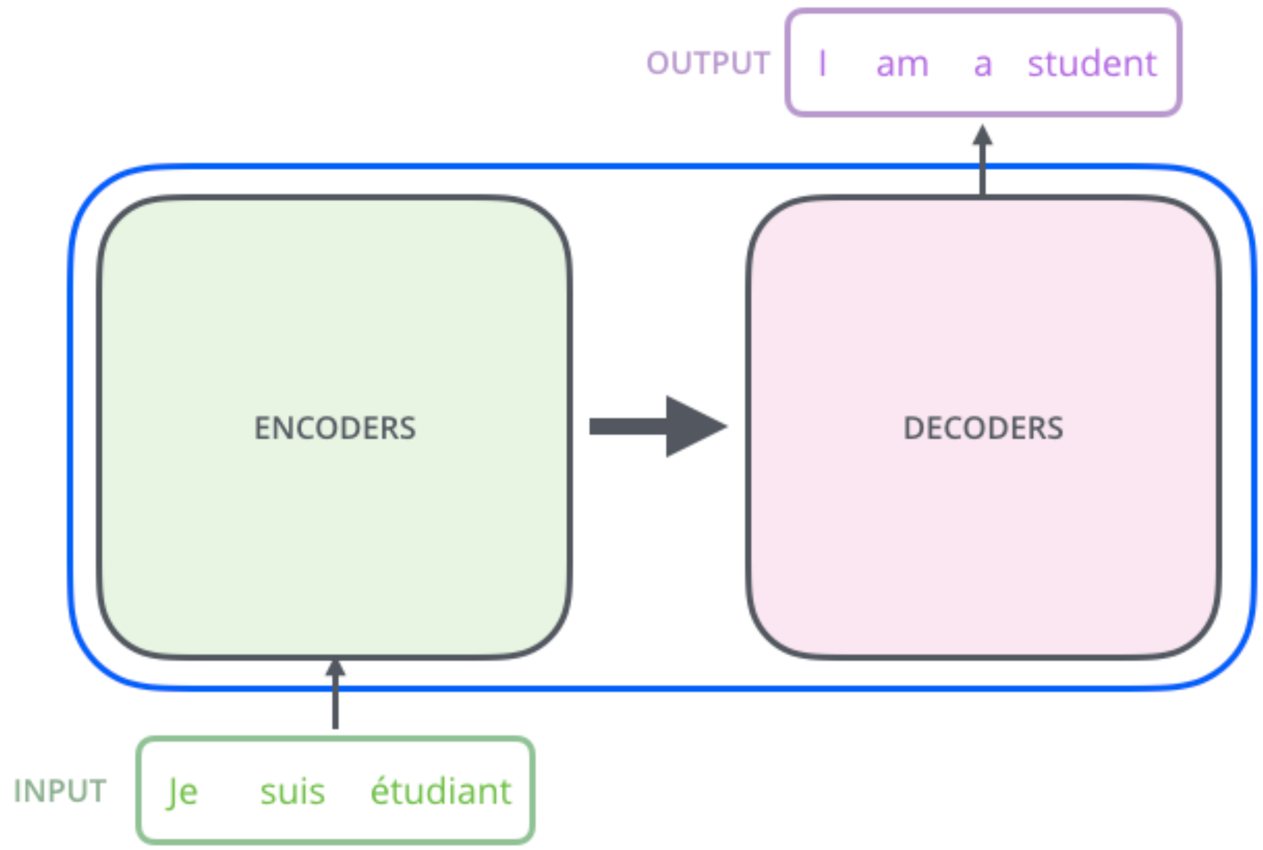


Figure credit: https://xamat.medium.com/transformers-models-an-introduction-and-catalogue-2022-edition-2d1e9039f376

Vanilla transformer

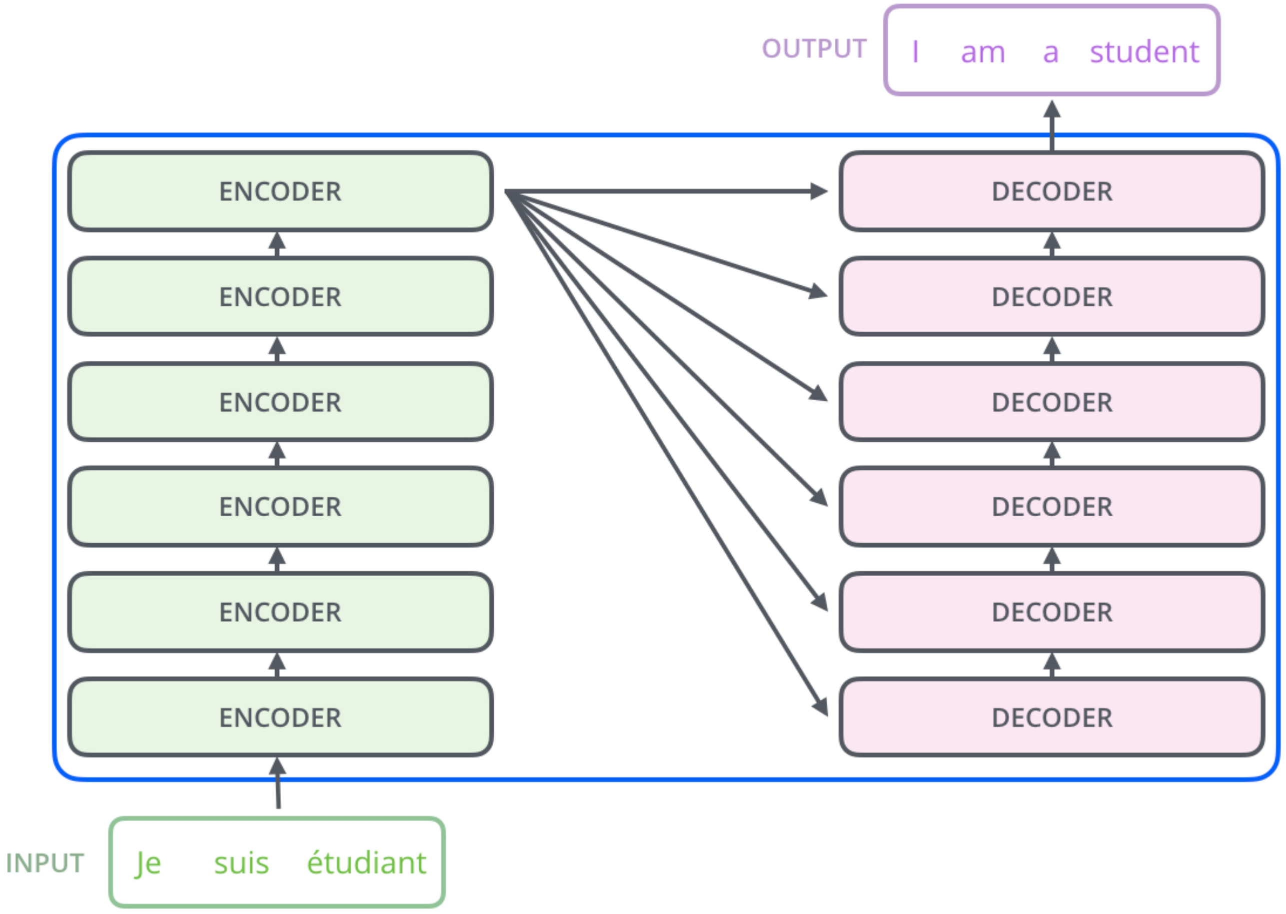


Encoder: input sequence to vector

Decoder: vector to output sequence

Jointly trained

Vanilla transformer

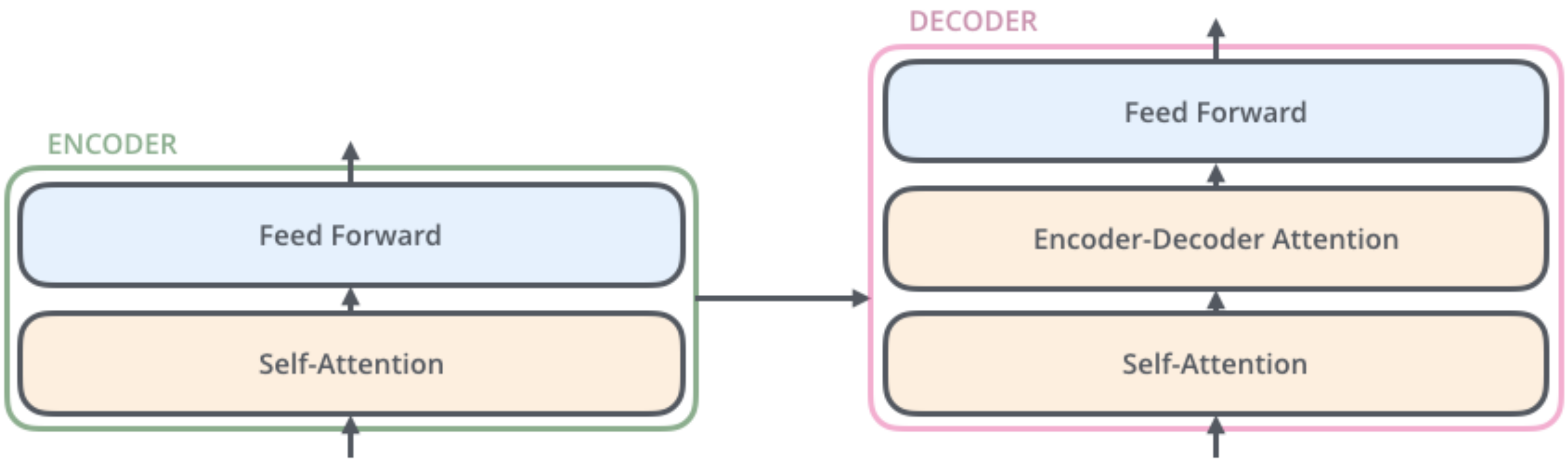


Encoders have similar 2-layer structure

Decoders have similar 3-layer structure

No weight sharing

Attention is all you need



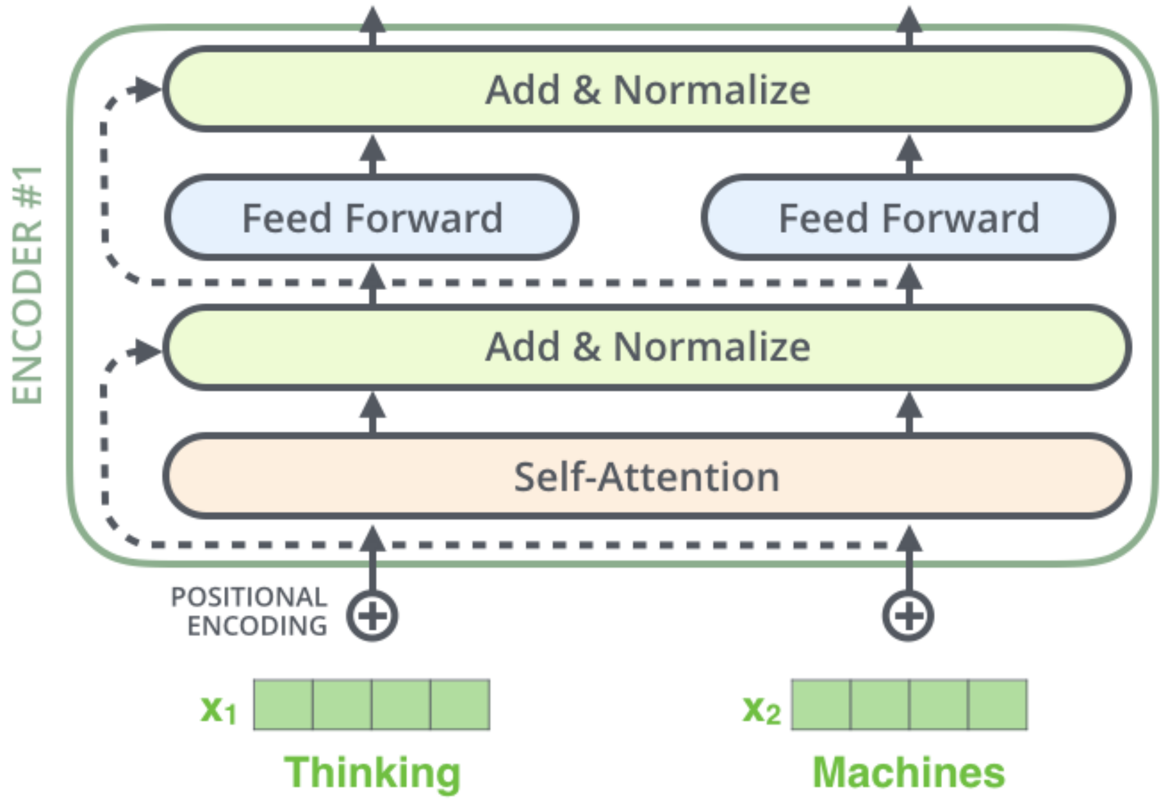
Self-attention helps the encoder look at other words in the input sentence

The same feed-forward neural network is applied to each position.

The encoder-decoder attention layer focuses on relevant parts of the input sentence

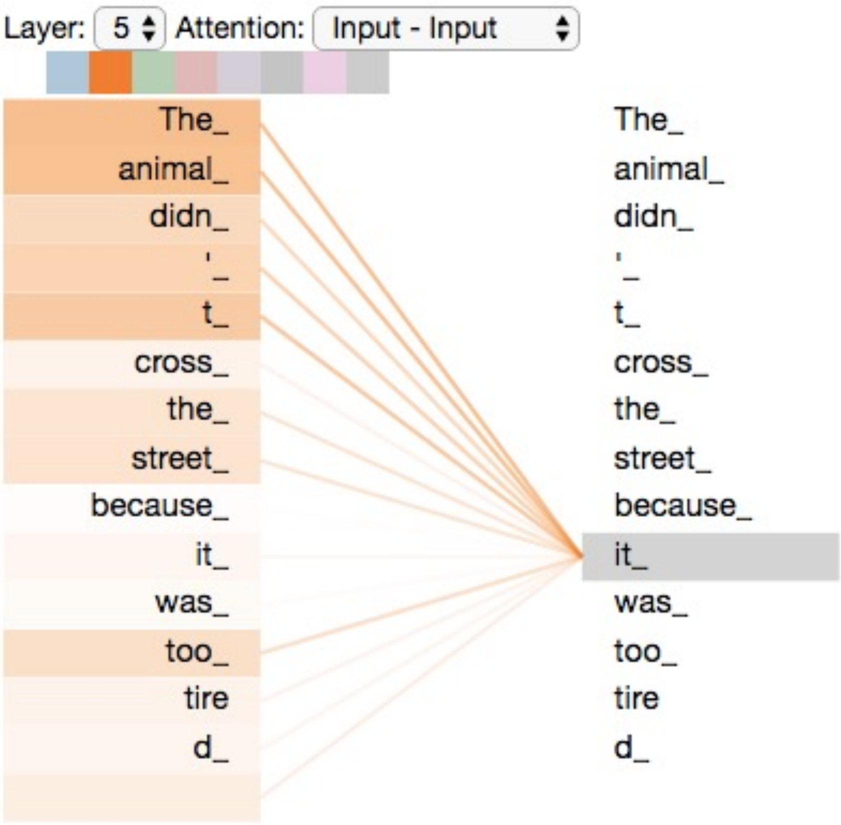
Why needed?

Each sublayer has a residual connection and a layer normalization



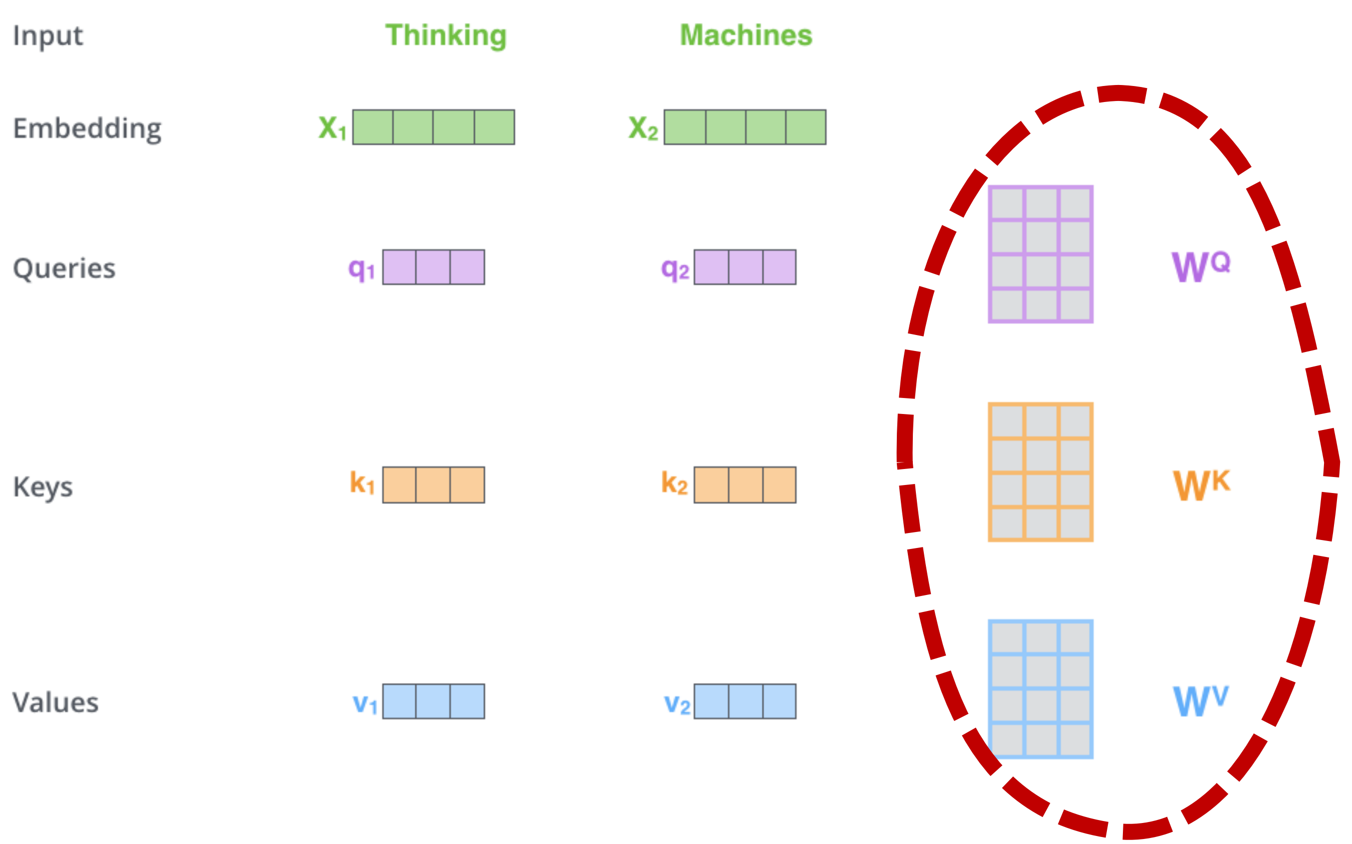
Self-attention example

"`The animal didn't cross the street because it was too tired`"



Self-attention in detail

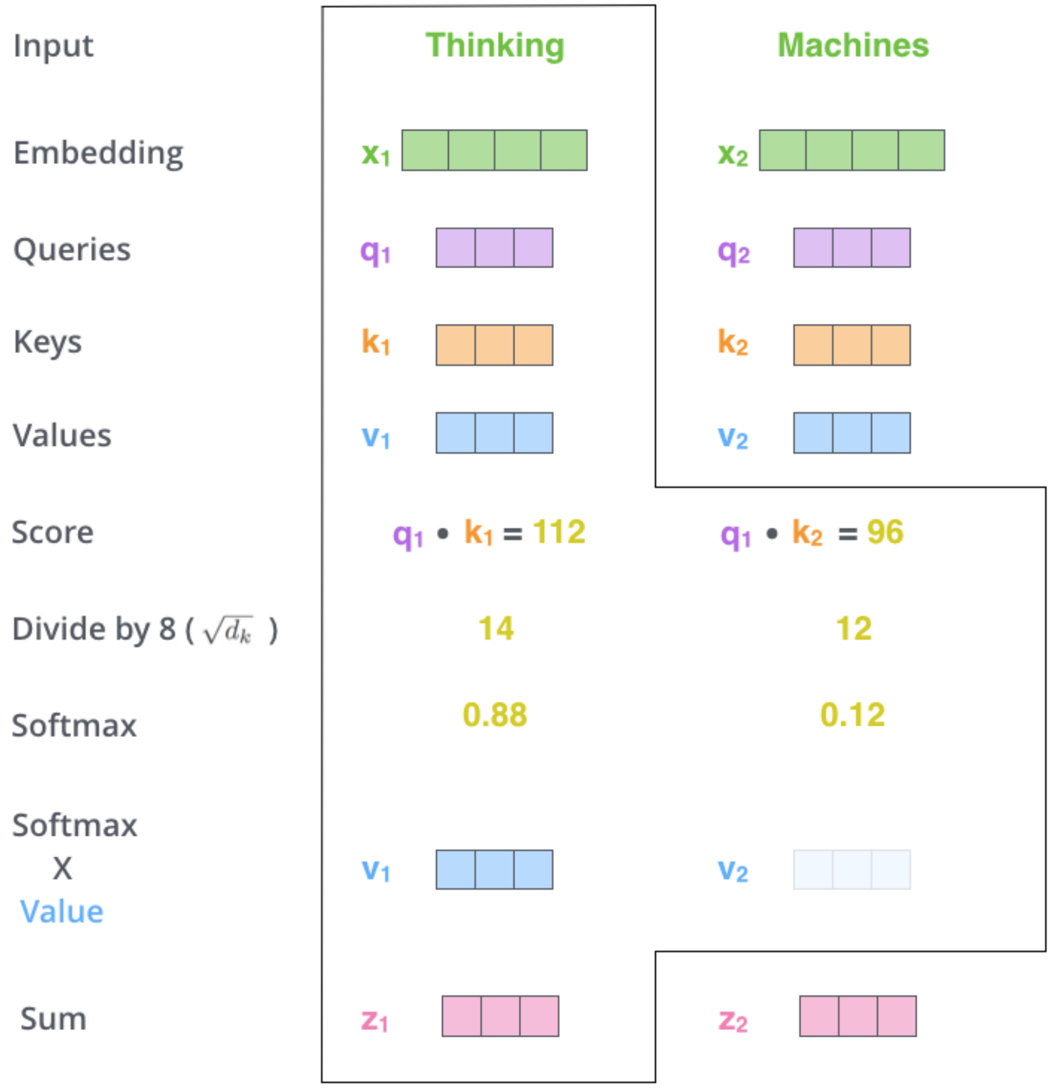
For each token, we create **a Query, a Key, and a Value vector** by multiplying the (word) embedding by three matrices that we optimize during training.



Self-attention in detail

Calculate scaled dot-product attention

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



Multi-head attention

Allows the model to jointly attend to information from different representation subspaces at different positions.

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

Simply learn multiple versions of W^Q, W^K and W^V and concat output.

Positional encoding

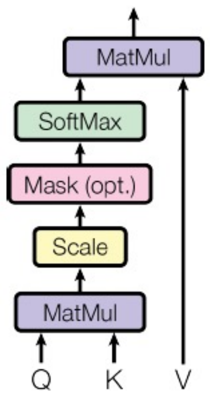
The attention operator is **permutation invariant**

Positional encoding added to input embedding accounts for word order.

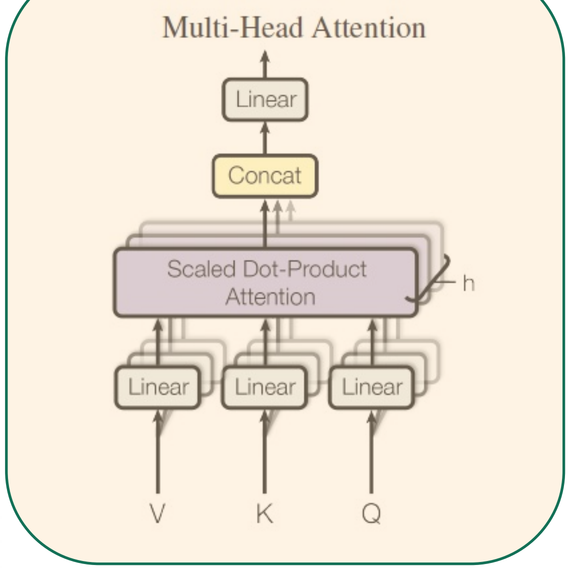
Vaswani et al. use **sine and cosine functions** of different frequencies.

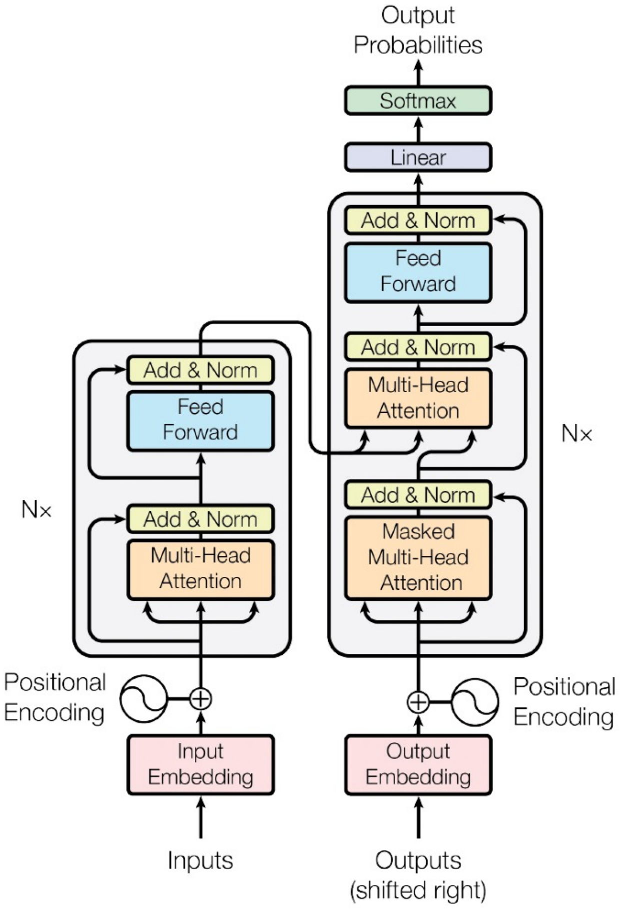
Overall architecture

Scaled Dot-Product Attention



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





2. Vision Transformer

We explain how transformers can be adapted for visual recognition. Reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks.

A vision transformer for classification

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

ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

Introduce **16x16 patch as token** to avoid computational complexity

Good speed/accuracy tradeoff

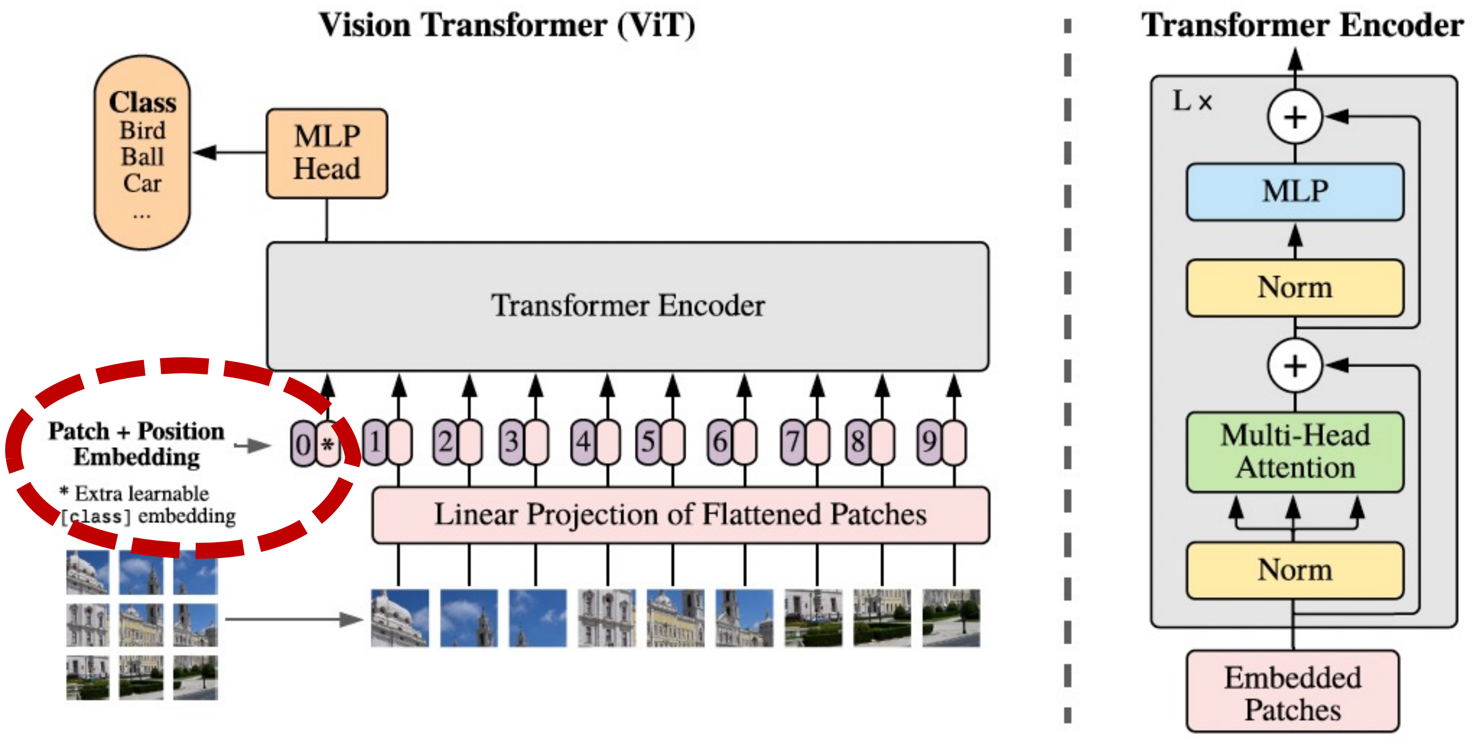
Competitive with CNN on **huge datasets**

Only suited for **classification**

Dosovitskoy et al. ICLR 2021

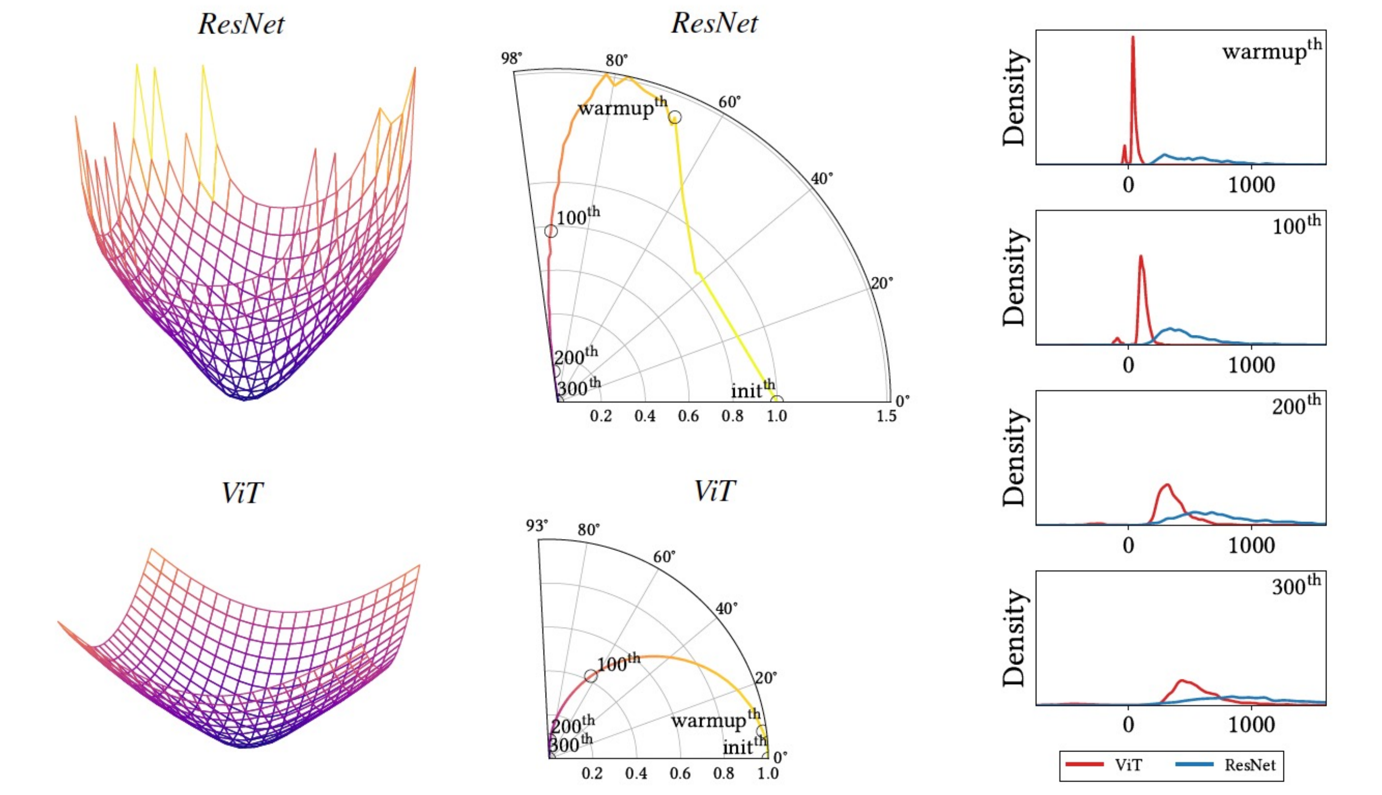
Model overview

Follows standard transformer encoder, adds learnable classification token



Why do vision transformers work?

Multi-scale soft attention flattens the loss landscapes



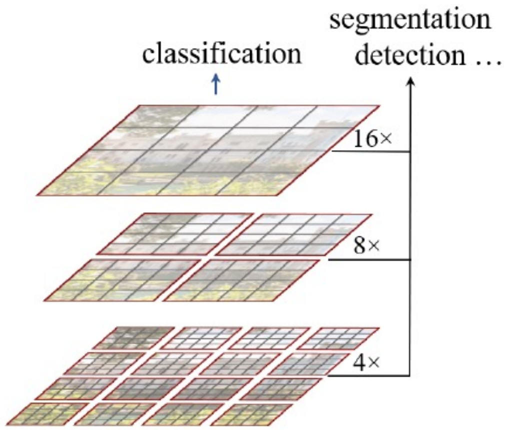
(a) Loss landscape visualizations (b) Trajectories in polar coordinates (c) Hessian max eigenvalue spectra

Park & Kim. ICLR 2022

Liu et al. ICCV 2021

Best paper award

3. Swin Transformer



This chapter presents the Swin Transformer, a general-purpose backbone for computer vision. It addresses specific vision challenges related to large variations in the scale of visual entities and the high resolution of pixels in images. It proposes a hierarchical Transformer whose representation is computed with shifted windows.

Recap: How to classify an image with an MLP?

A **256x256 RGB image** requires 200 000 input values

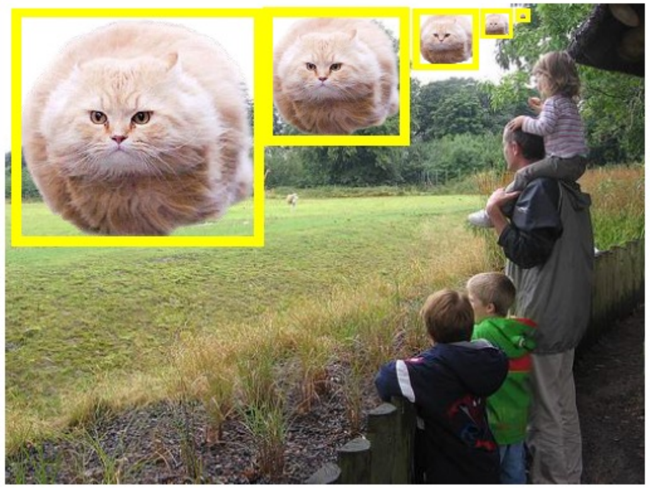
MLP with a single hidden layer with 500 units already implies
100 million parameters

Clearly we need to incorporate an **inductive bias** into the architecture

From language to vision

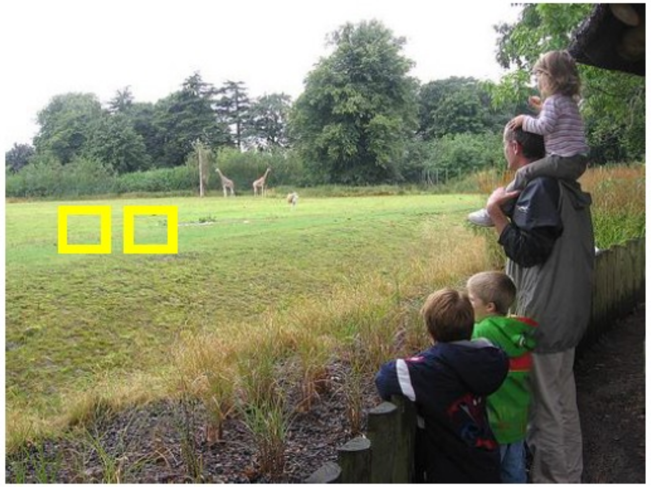
Differences between visual and text signals

Multi-scale
(scale invariance)



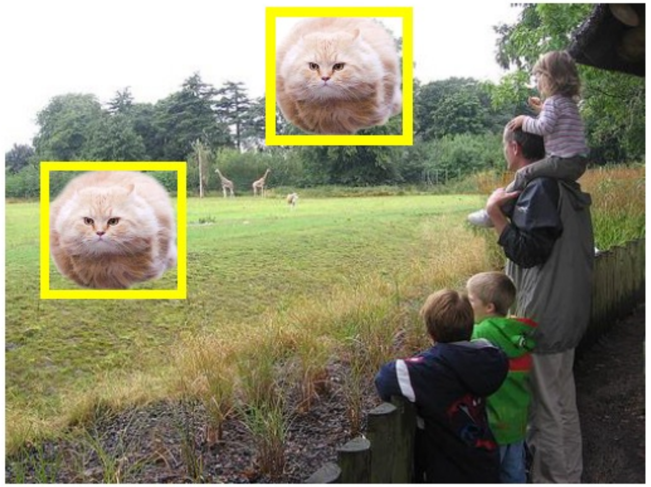
I am a **fat cat.**
I am a **fat fat cat cat. (invalid)**

Locality
(spatial smoothness)



I like the **green grass**.

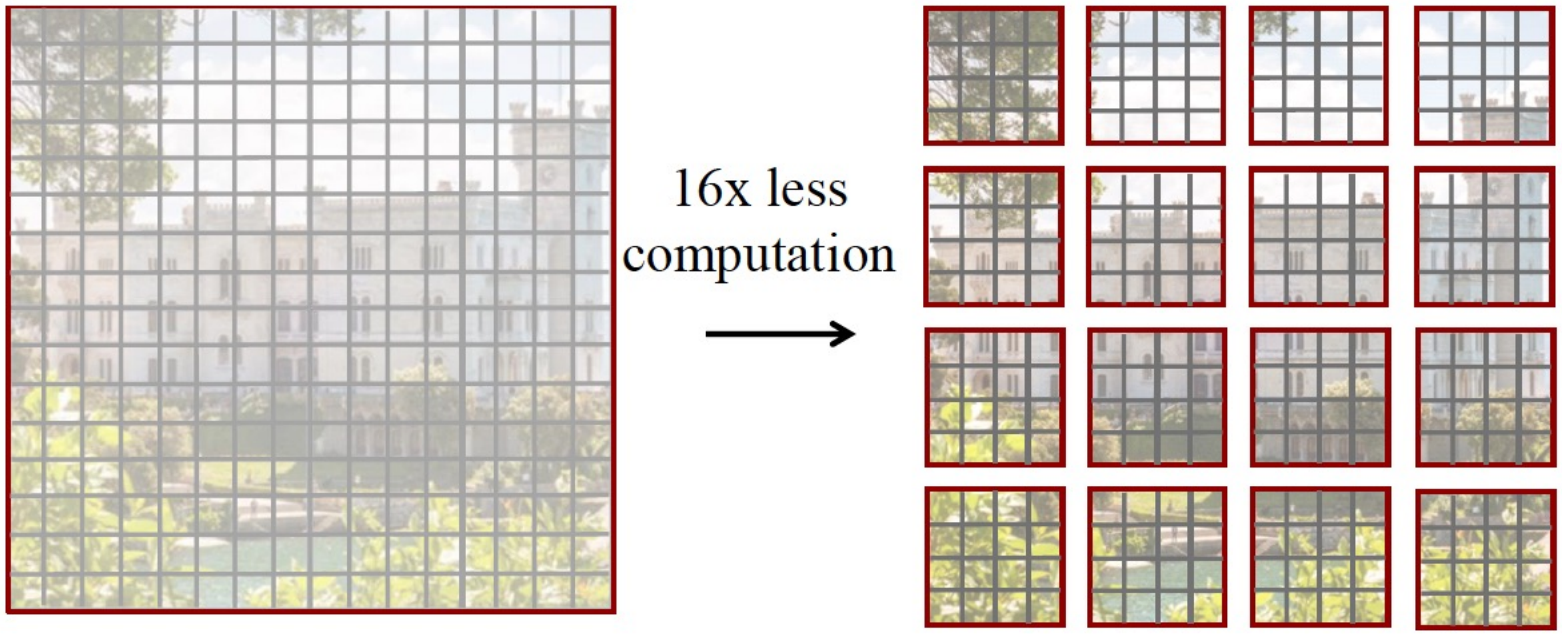
Translation invariance



I am a **fat cat**.
Fat cat is me.

Key idea: Shifted windows

Linear computation complexity with image resolution: from O(n2) to O(n)

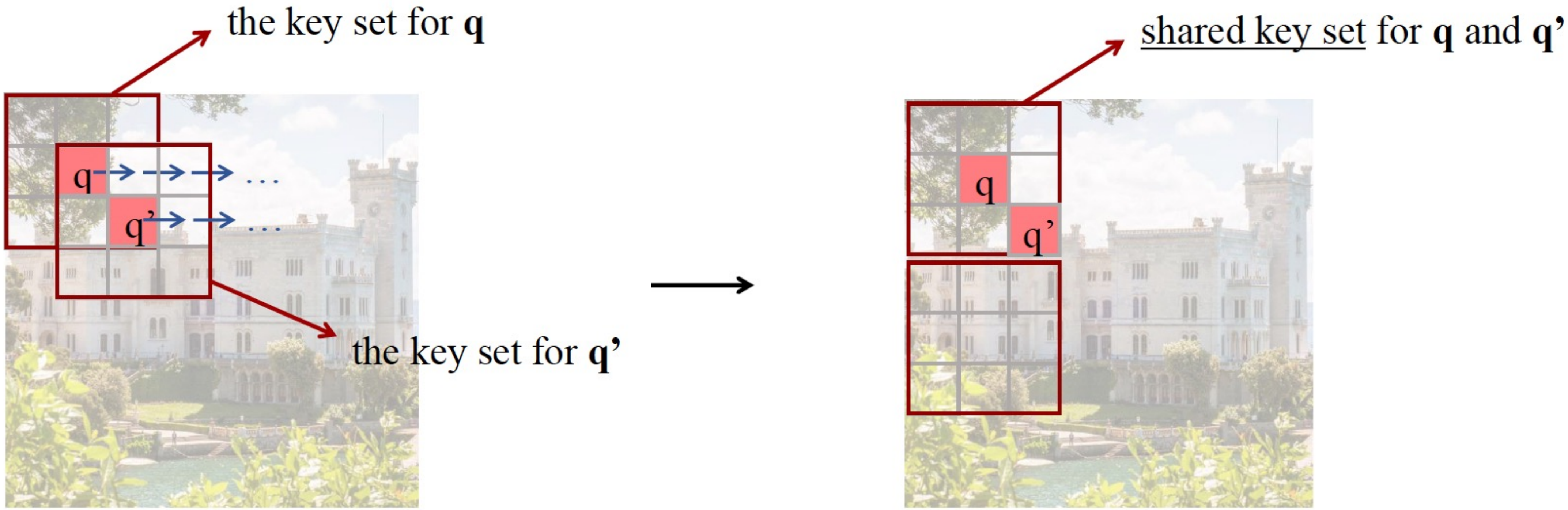


Vanilla vision Transformer (ViT):
256^2=65536 (Global)

Swin Transformer:
16x16^2=4096 (Local)

Key idea: Shifted windows

Shared key set in same window enables friendly memory access

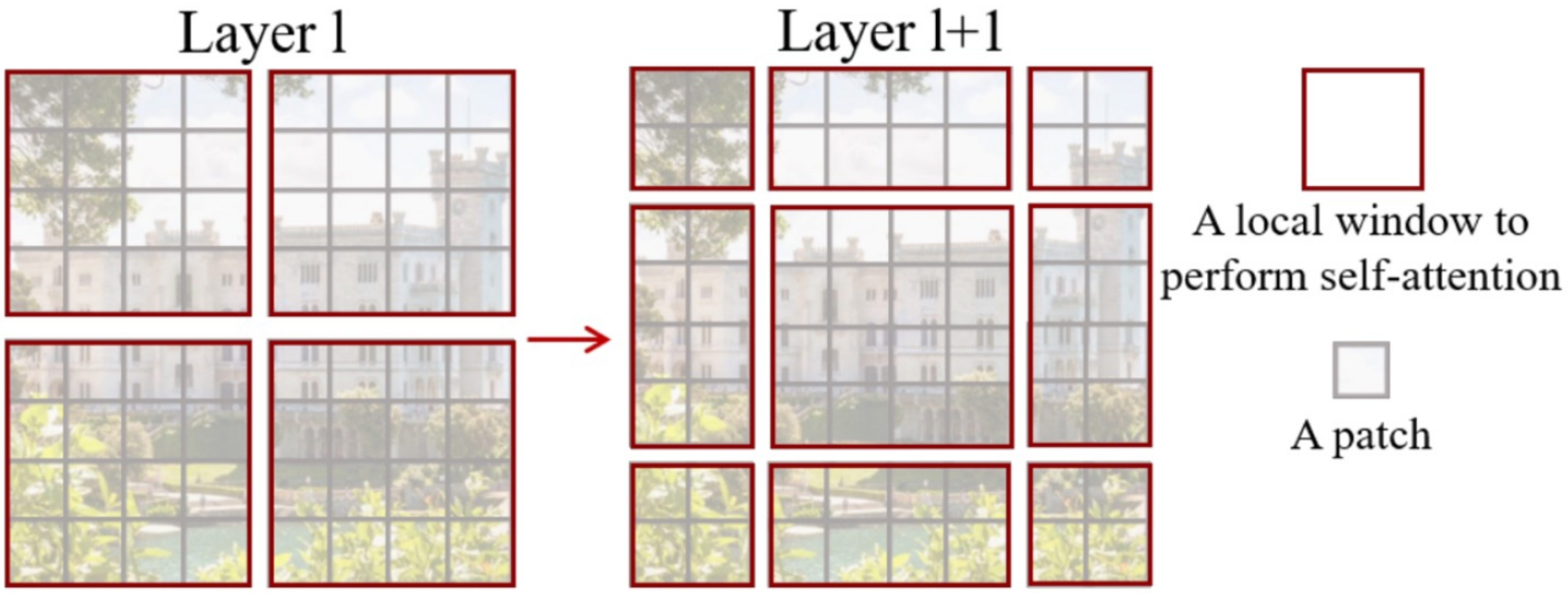


Traditional sliding window

Non-overlapping window (Swin Transformer)

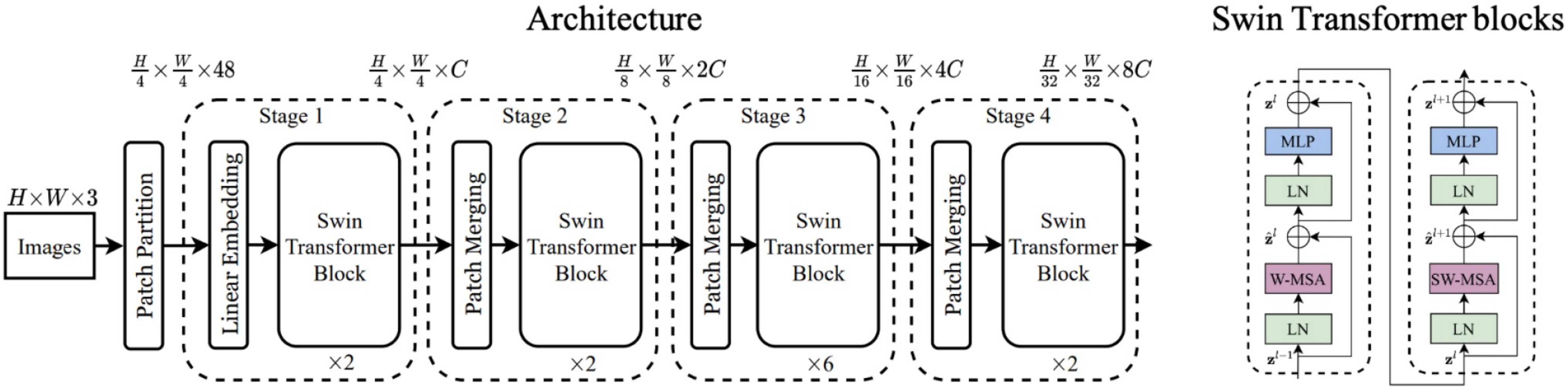
Key idea: Shifted windows

Shifted non-overlapping windows enable cross-window connections



Swin Transformer architecture

An hierarchical transformer



Swin Transformer is solid vision backbone

ImageNet Classification

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [34]	384^2	388M	204.6G	-	84.4
R-152x4 [34]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Coco Detection and Segmentation

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

ADE20K Segmentation

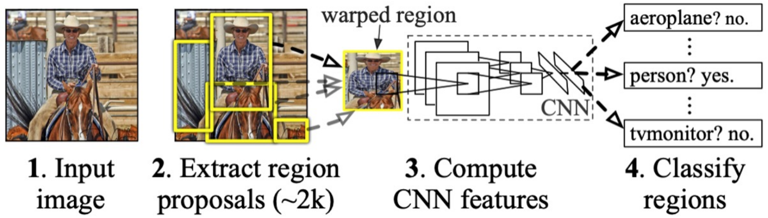
ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
DNL [65]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [67]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [63]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [67]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [73]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

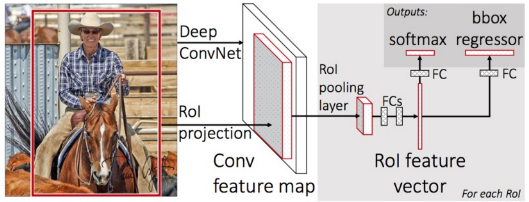
4. Detector Transformers

In this chapter, we cover transformers for object detection. They effectively remove the need for many hand-designed components like a non-maximum suppression procedure or anchor generation. We also cover a simple box-attention mechanism that enables spatial interaction between grid features, as sampled from boxes of interest, and improves the learning capability of transformers for several 2D and 3D detection and segmentation tasks.

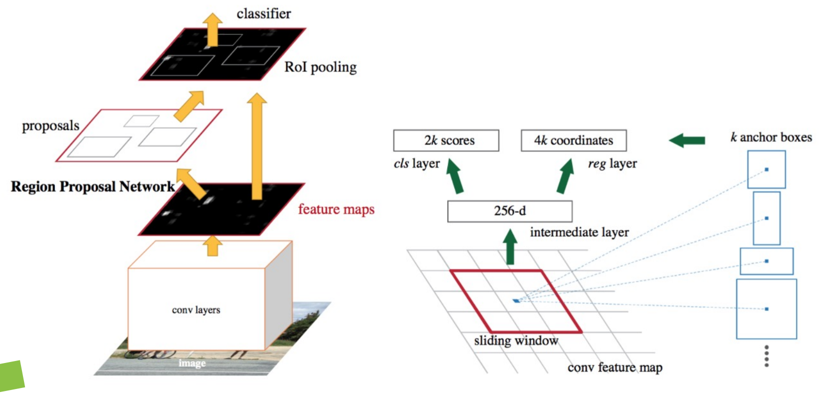
Recap: Modern Detectors



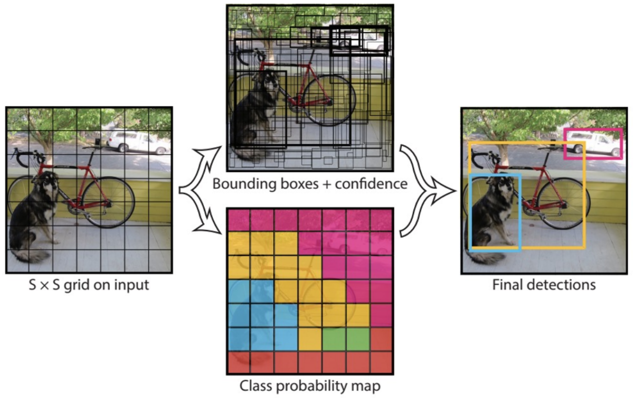
RCNN



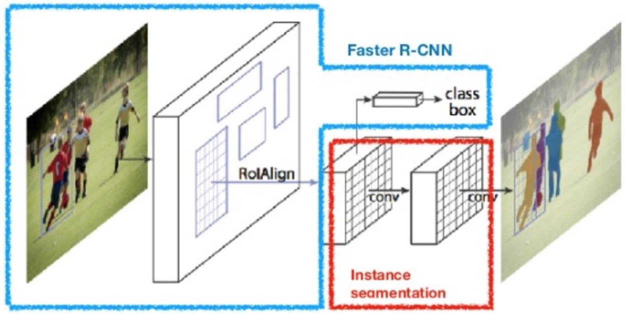
Fast RCNN



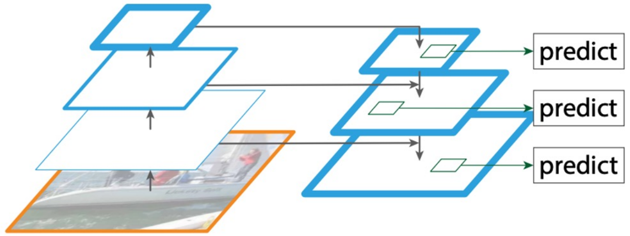
Faster RCNN



YOLO



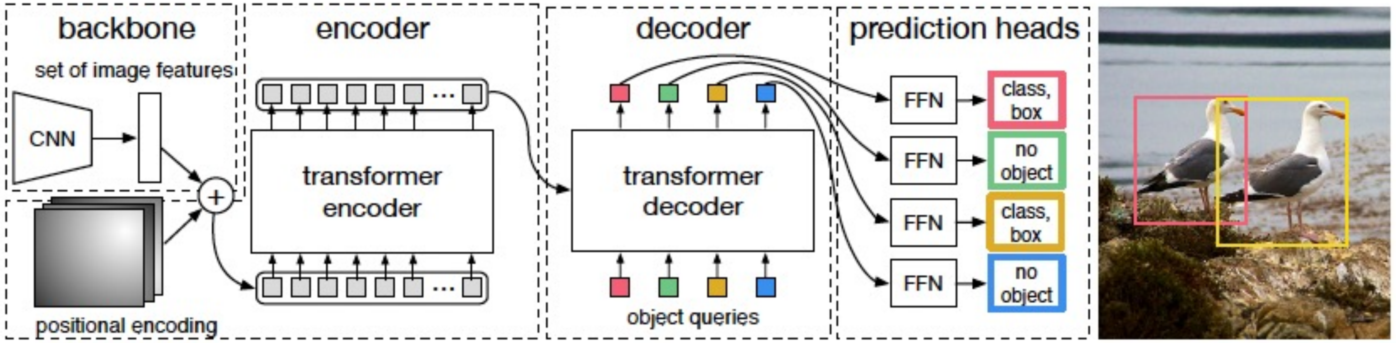
Mask RCNN



Feature Pyramid Network

DETR: First vision transformer for detection

Models detection as **set prediction problem** using Hungarian loss and

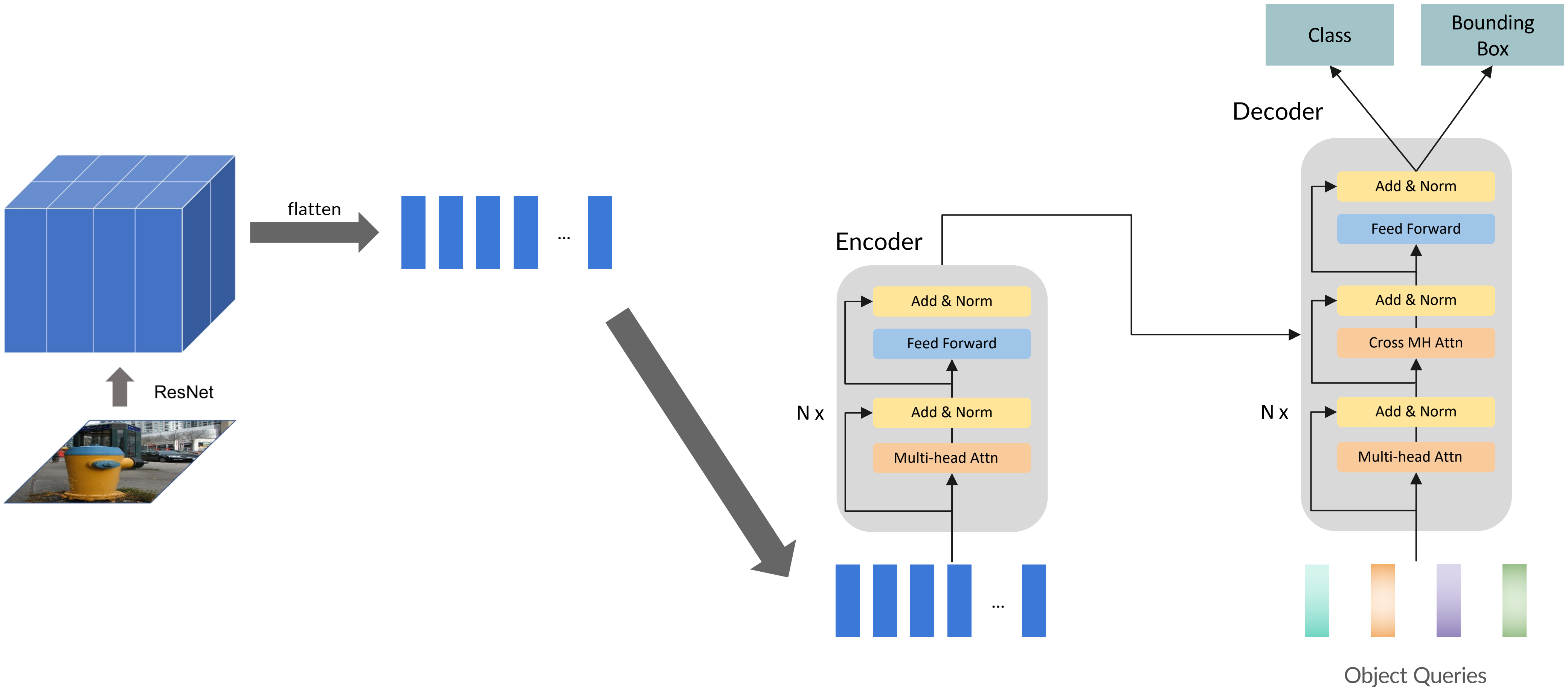


uses transformer to **encode relationship** between set elements

Removes the need for **hand-crafted modules**:

non-maximum suppression, anchor generation, …

DETR: more detailed look



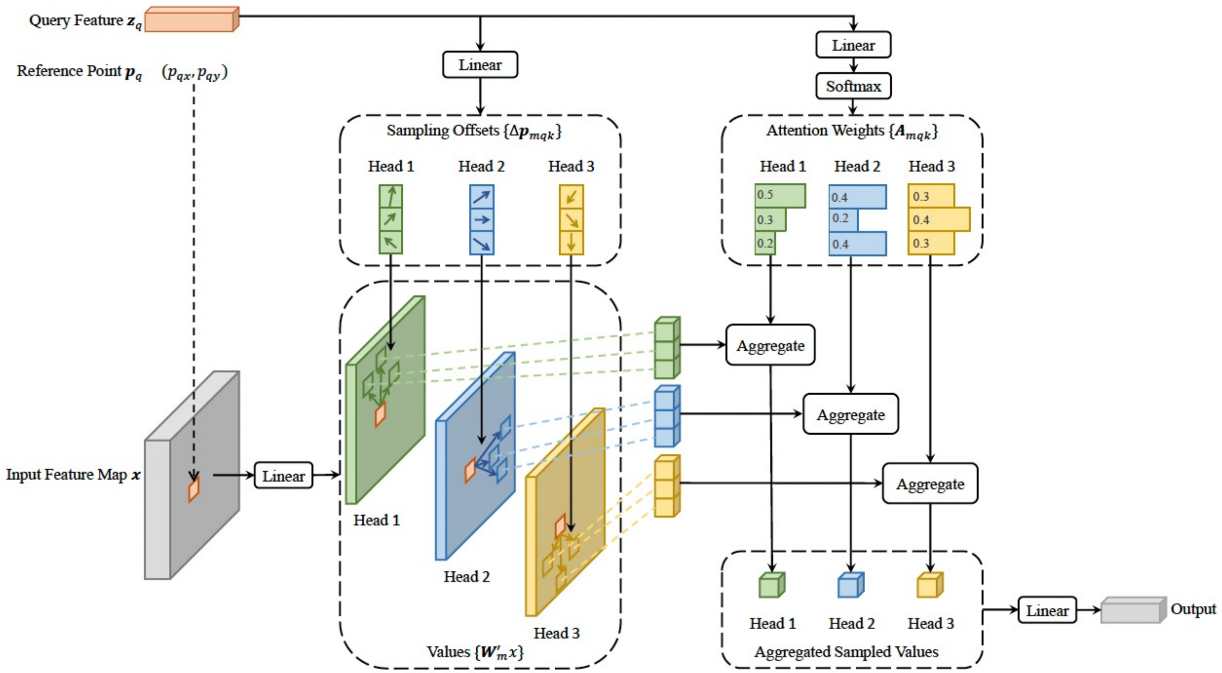
Zhu et al. ICLR 2021

Deformable-DETR: for detection and segmentation

Introduces **deformable-attention** to attend to sparse set of elements from whole feature map, regardless of spatial size.

Adds **multi-scale** variant

Faster convergence.

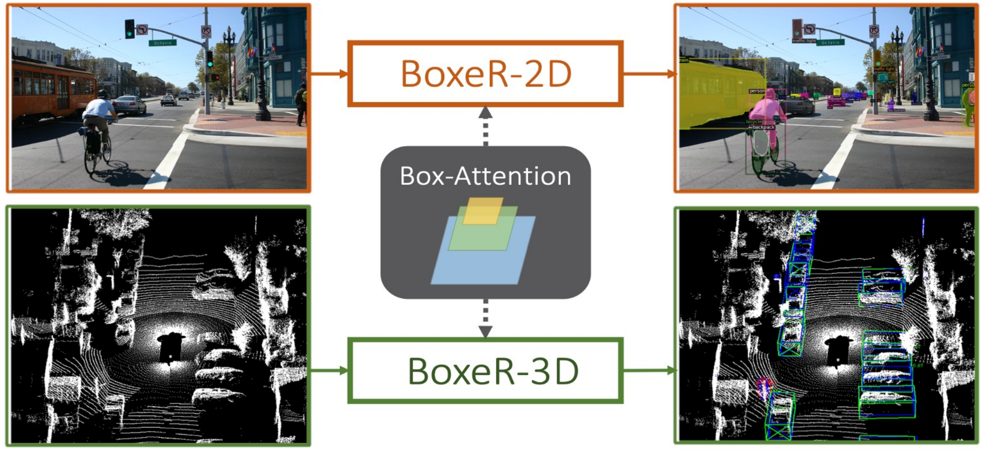


Nguyen et al. CVPR 2022

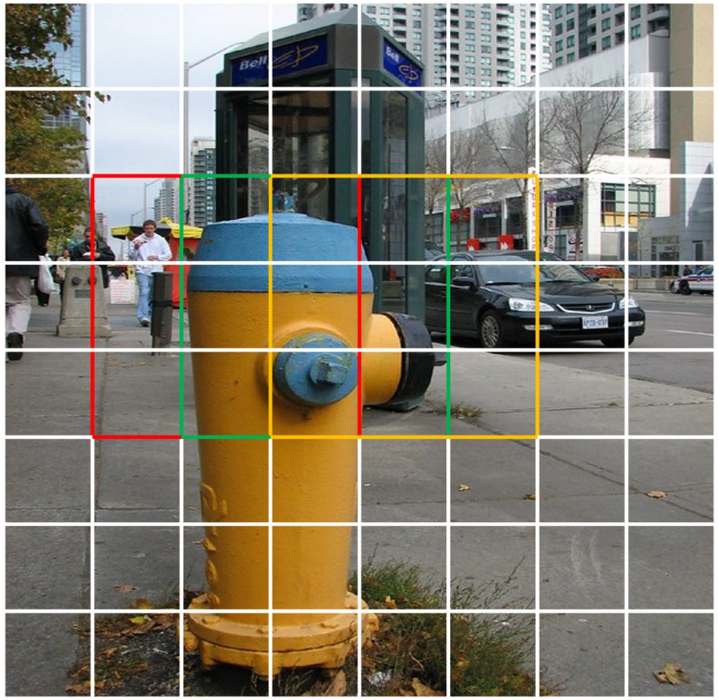
BoxeR: Box-Attention for 2D and 3D Transformers

Key observation: existing detector transformers ignore the inherent regularities of the vision modality.

Image features are vectorized the same way as language tokens, resulting in **loss of local connectivity** among pixels.



Motivation of Box-Attention



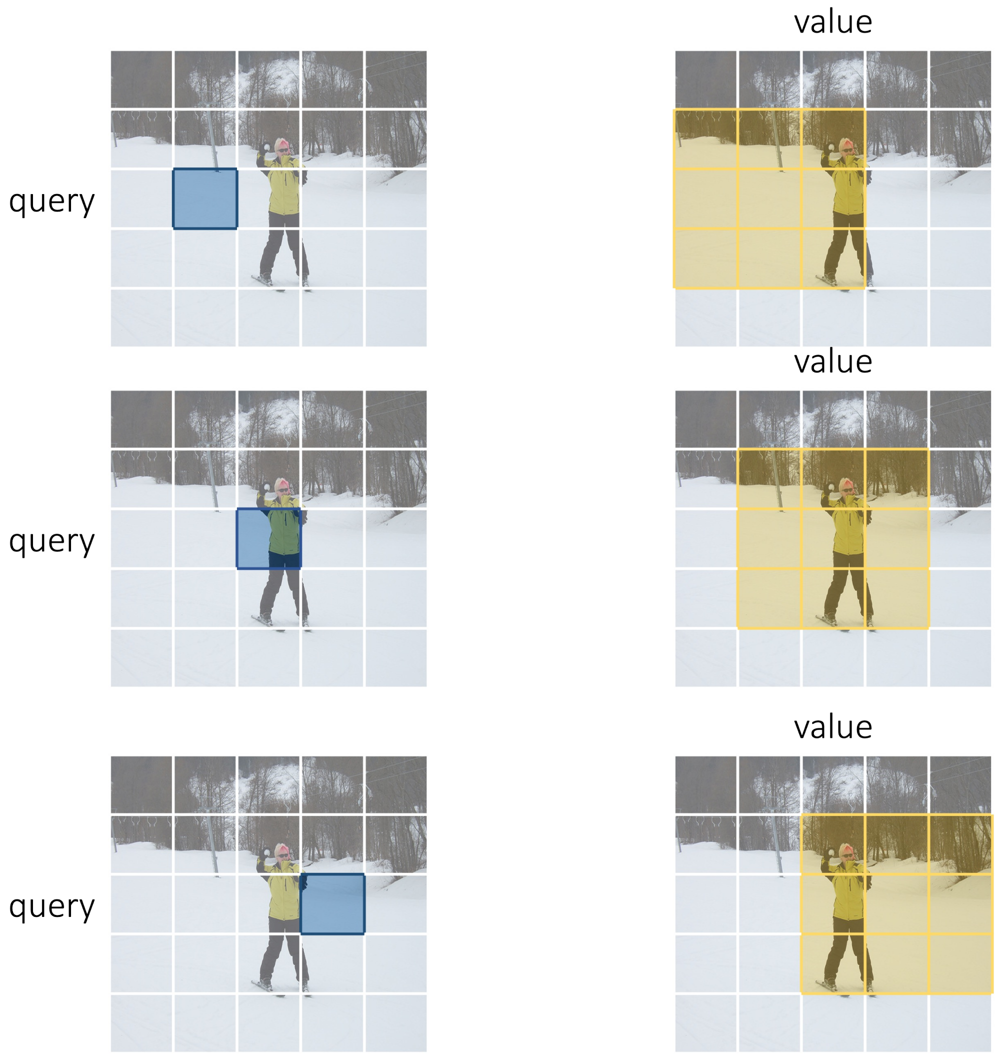
Success of **sliding window** in modern detectors

Use of **grid structure** within boxes in attention computation

Enable **2D inductive bias** on multi-scale features

Box-Attention

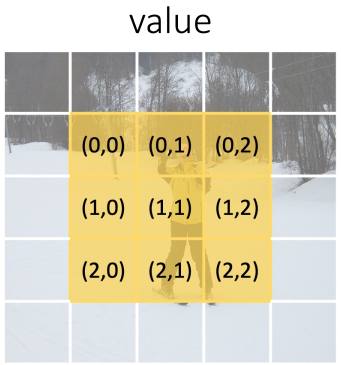
Each **query** vector with a **reference window**

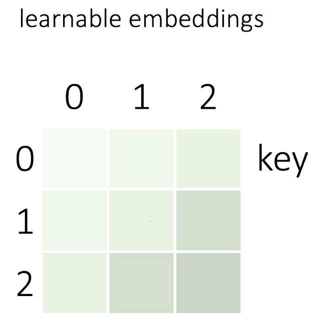


Box-Attention

Each **query** vector with a **reference window**

Key as **learnable** vectors of **relative positions**



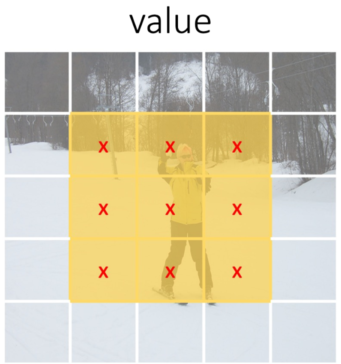


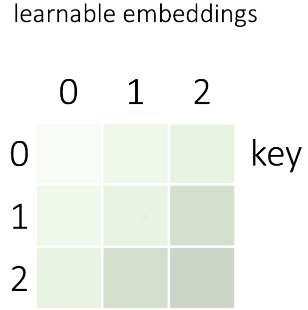
Box-Attention

Each **query** vector with a **reference window**

Key as **learnable** vectors of **relative positions**

Value vectors are sampled from the window



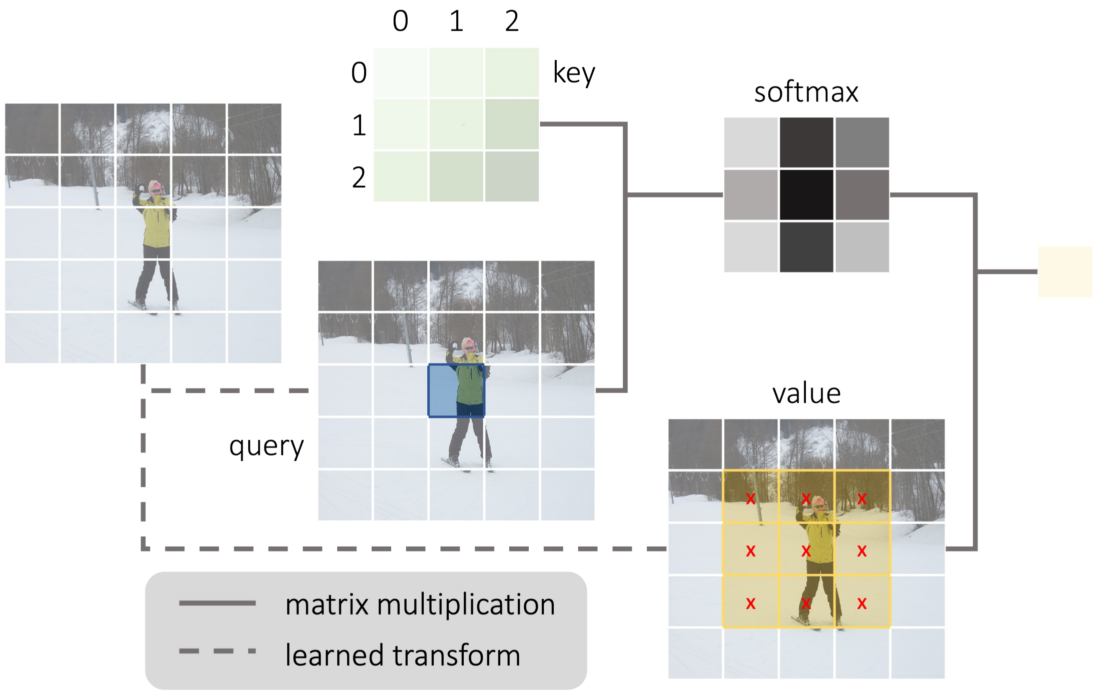


Box-Attention

Each **query** vector with a **reference window**

Key as **learnable** vectors of **relative positions**

Value vectors are sampled from the window



Attention computation

Where-to-attend module

Learn transformation functions: **translation** + **scaling**

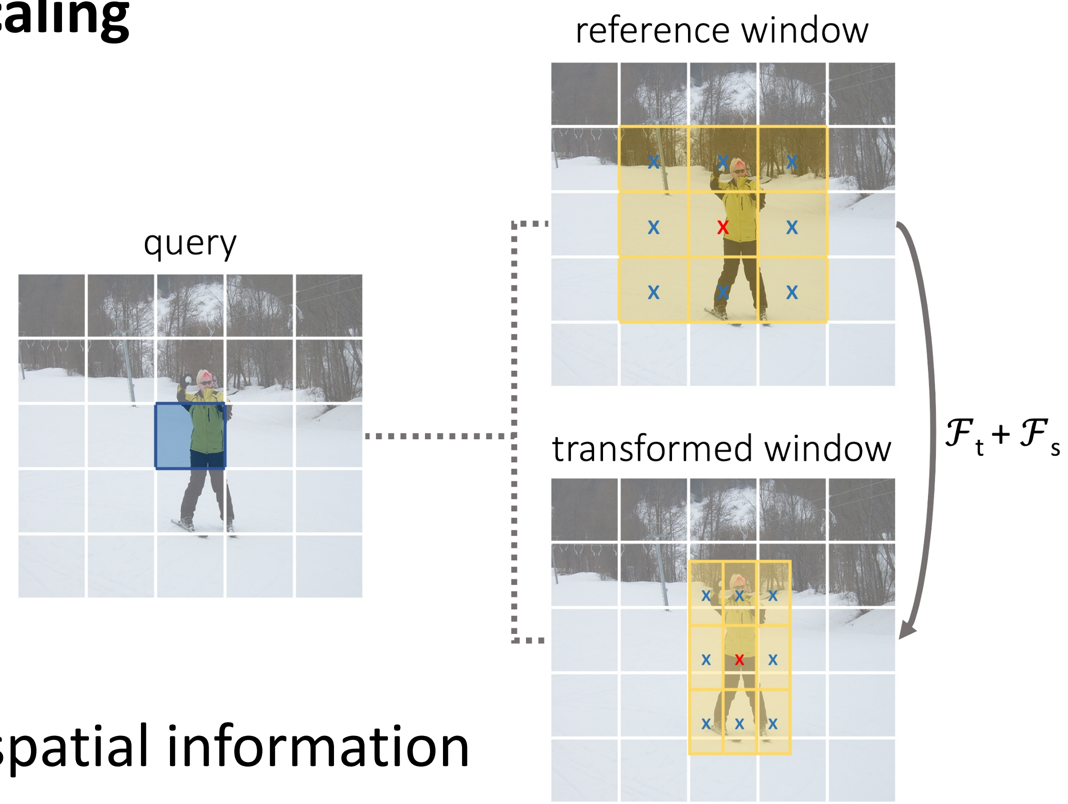
A reference window: b = [x, y, wx, wy] and query q

Translation: $\mathcal{F}t$ (b, q) = [x + Δx, y + Δy, wx, wy]

Scaling: $\mathcal{F}s$ (b, q) = [x, y, wx + Δwx, wy + Δwy]

Sample **a grid of features** from the transformed box

Extend Box-Attention to **Instance-Attention**
which predicts an instance mask by preserving spatial information

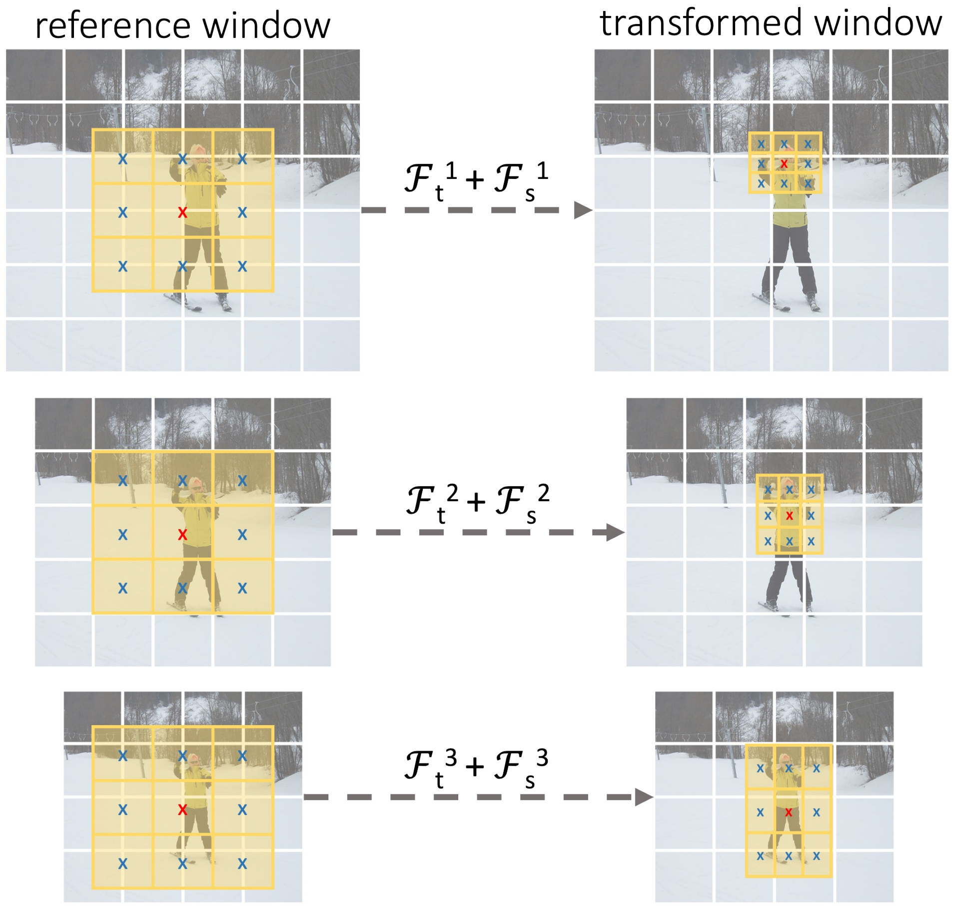


Multi-scale variant

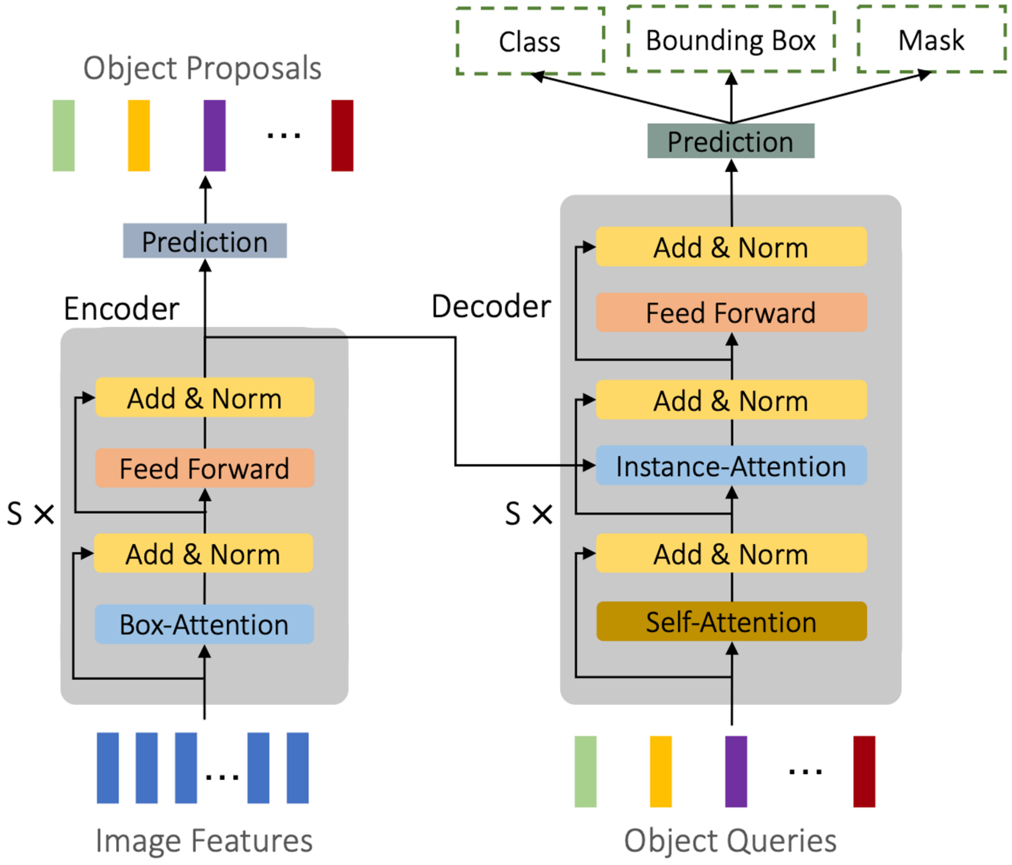
Each box with a **separate where-to-attend** module

Query vector of **each multi-scale feature** maps with **different reference window size**

Key vectors **correspond** to **transformed boxes**



BoxeR-2D: object detection and instance segmentation



Utilize **multi-scale feature** maps of **ResNet**

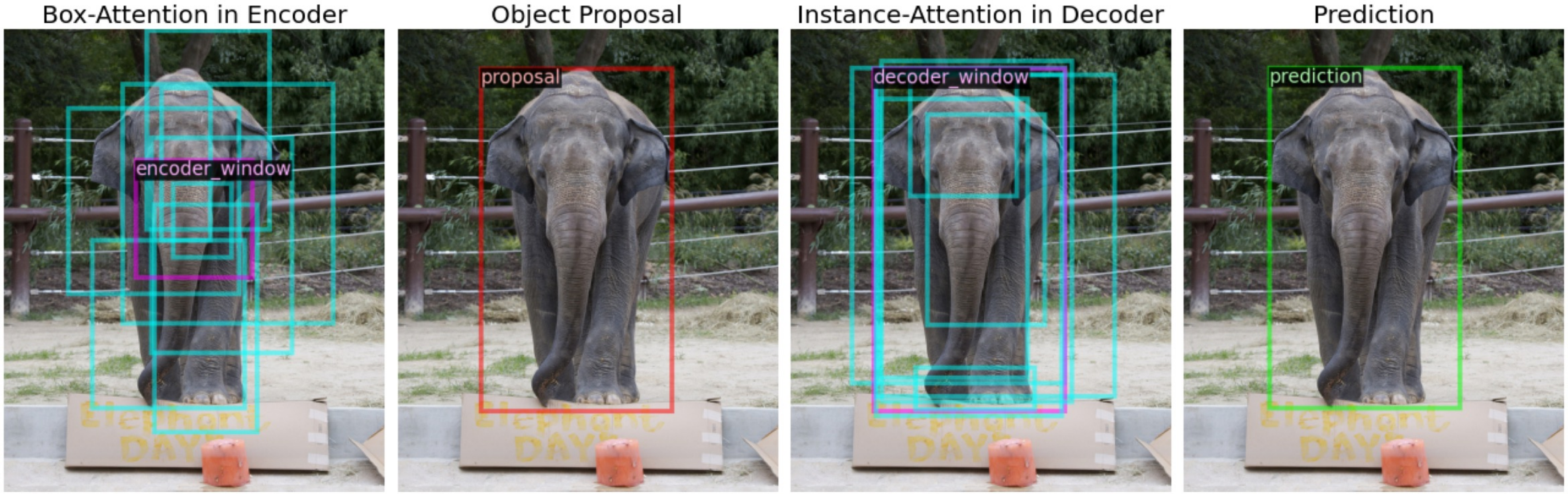
Each **query** with a **reference window**

A **bounding box** is predicted w.r.t **reference window**

BoxeR-2D behavior

High-quality object proposals from encoder overlap with prediction

Predicted boxes from attention module **capture regions with multiple aspect ratios**



Comparisons on COCO 2017 test-dev

BoxeR outperforms CovNets and Transformers by 2 AP points on all metrics

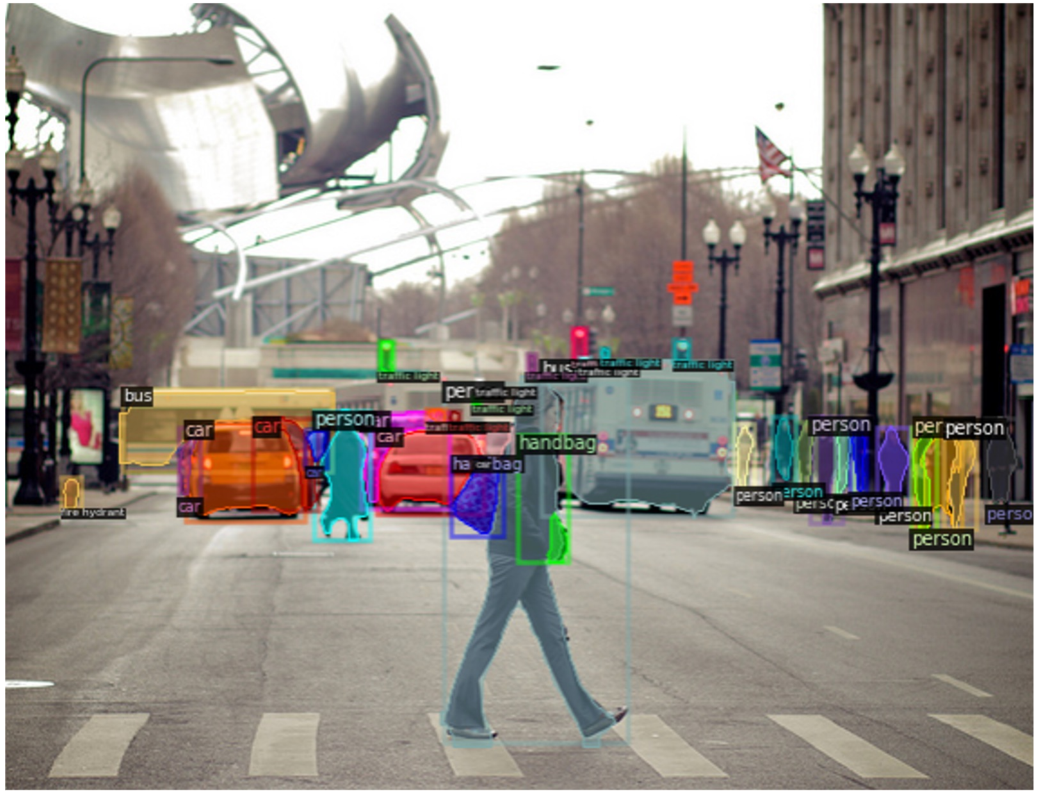
	Method	Backbone	Epochs	end-to-end	AP↑	AP_{50}↑	AP_{75}↑	AP_S↑	AP_M↑	AP_L↑
ConvNet	Faster RCNN-FPN	R-101	36	✗	36.2	59.1	39.0	18.2	39.0	48.2
	ATSS	R-101	24	✗	43.6	62.1	47.4	26.1	47.0	53.6
	Sparse RCNN	X-101	36	✓	46.9	66.3	51.2	28.6	49.2	58.7
	VFNet	R-101	24	✗	46.7	64.9	50.8	28.4	50.2	57.6
Transformer	Deformable DETR	R-50	50	✓	46.9	66.4	50.8	27.7	49.7	59.9
	Deformable DETR	R-101	50	✓	48.7	68.1	52.9	29.1	51.5	62.0
	Dynamic DETR	R-50	50	✓	47.2	65.9	51.1	28.6	49.3	59.1
	TSP-RCNN	R-101	96	✓	46.6	66.2	51.3	28.4	49.0	58.5
BoxeR-2D	**BoxeR-2D**	R-50	50	✓	50.0	67.9	54.7	30.9	52.8	62.6
	BoxeR-2D (3× schedule)	R-50	36	✓	49.9	68.0	54.4	30.9	52.6	62.5
	BoxeR-2D (3× schedule)	R-101	36	✓	**51.1**	**68.5**	**55.8**	**31.5**	**54.1**	**64.6**

Same for segmentation

BoxeR outperforms CovNets and Transformers by 2 AP points on all metrics

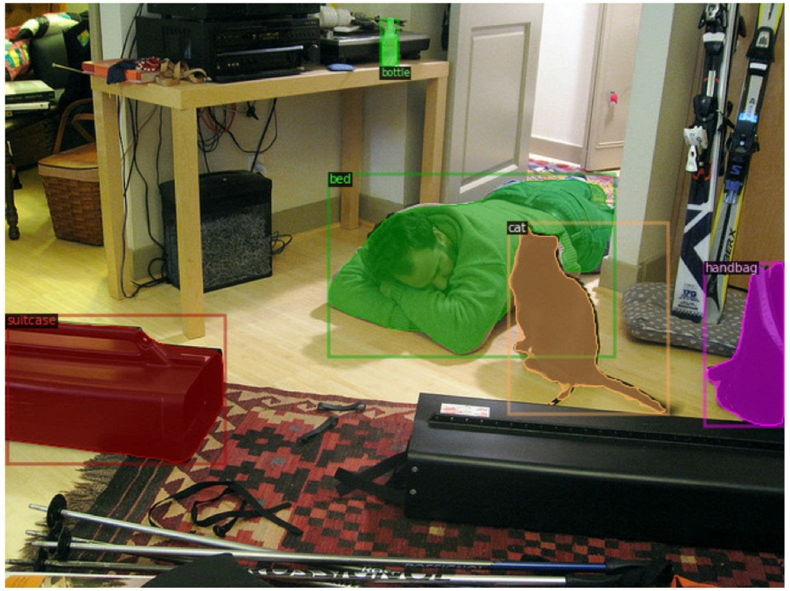
		Epoch	end-to-end	AP↑	AP_S↑	AP_M↑	AP_L↑	AP^m↑	AP^m_S↑	AP^m_M↑	AP^m_L↑
ConvNet	Mask R-CNN	36	✗	43.1	25.1	46.0	54.3	38.8	21.8	41.4	50.5
	QueryInst	36	✗	48.1	-	-	-	42.8	24.6	45.0	55.5
Transformer	SOLQ	50	✓	48.7	28.6	51.7	63.1	40.9	22.5	43.8	54.6
BoxeR-2D	**BoxeR-2D** (3× schedule)	36	✓	**51.1**	**31.5**	**54.1**	**64.6**	**43.8**	**25.0**	**46.5**	**57.9**

Success cases



Failure cases

Small objects in low-light conditions still hard



Classification failure

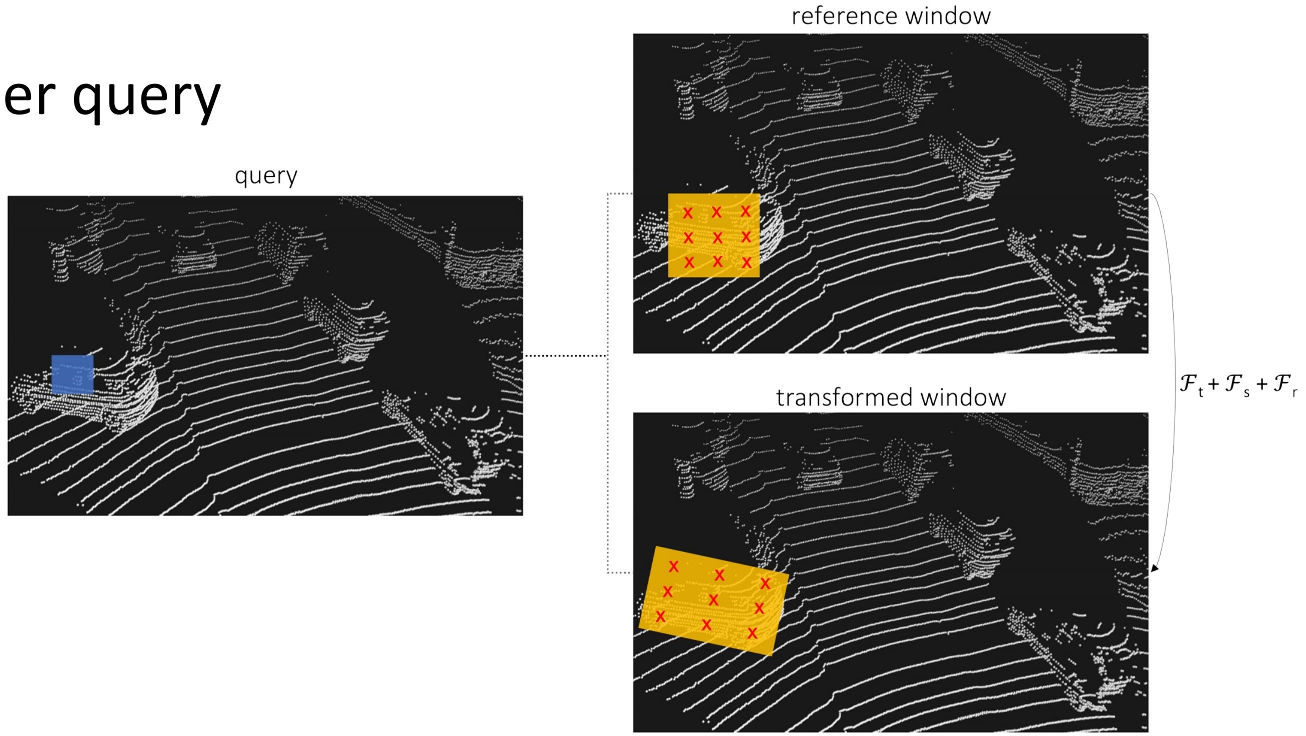
BoxeR-3D

Learn transformation function: **rotation**

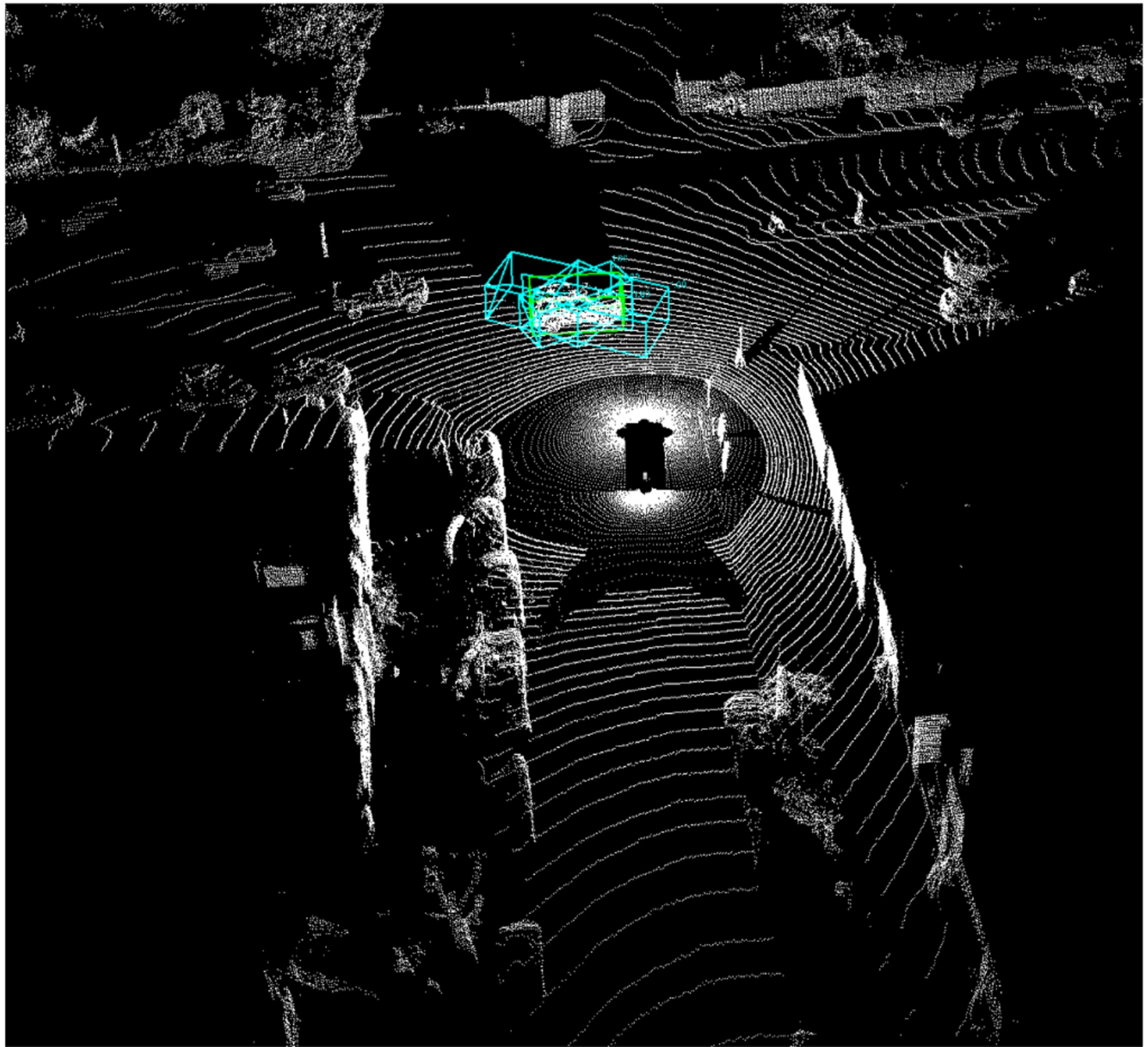
A reference window: b = [x, y, wx, wy, θ] and query q

Rotation: $\mathcal{F}r$ (b, q) = [x, y, wx, wy, $\theta + \Delta\theta$]

Use **multi-angle reference windows** per query



BoxeR-3D behavior



Multiple heads capture boxes of **different angles** and one is well-aligned with the groundtruth

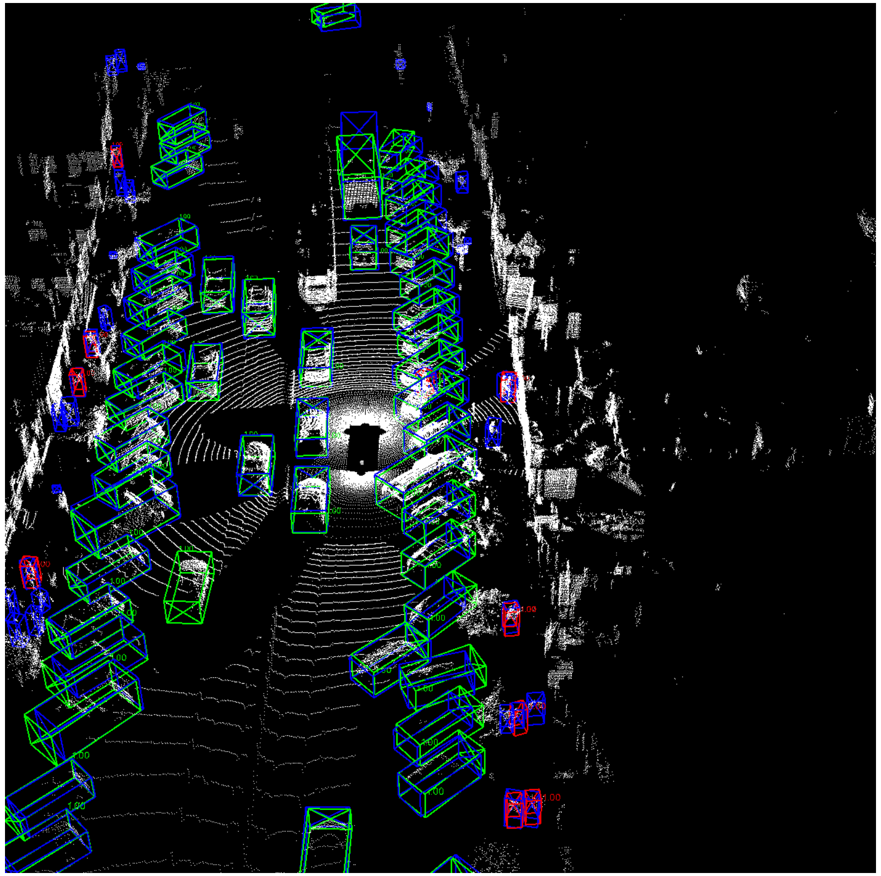
Comparisons on Waymo Open val set

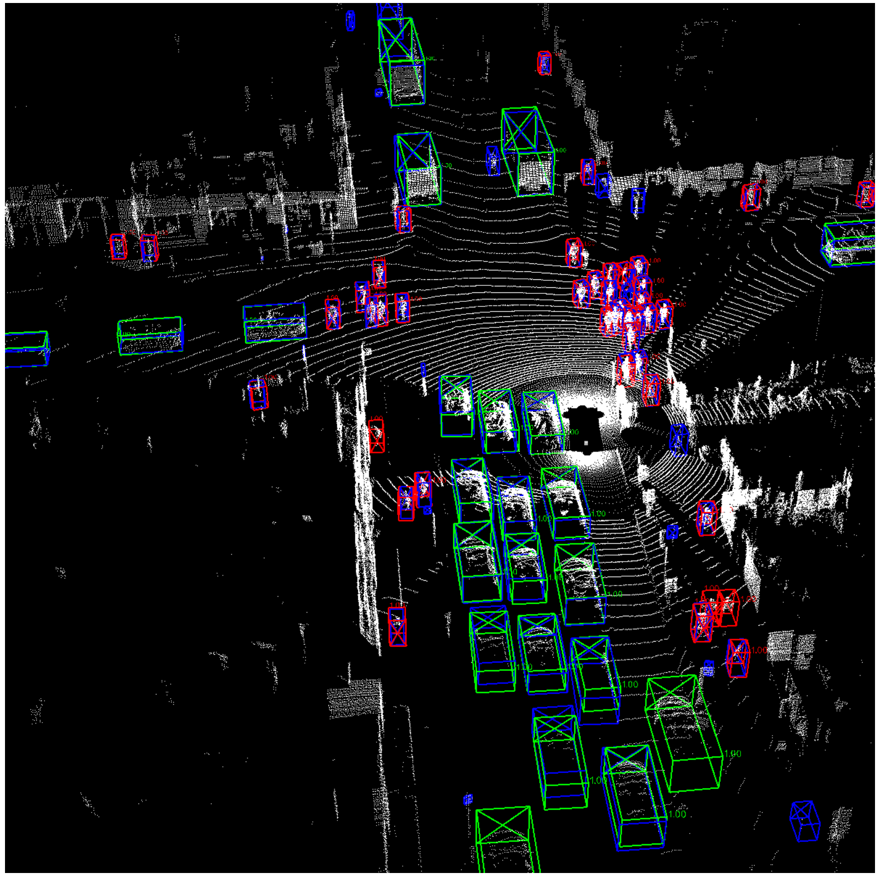
Much better than vanilla transformer, bit behind on dedicated solutions.

		end-to-end	Vehicle AP↑	Vehicle APH↑	Pedestrian AP↑	Pedestrian APH↑
ConvNet	PointPillar	✗	55.2	54.7	60.0	49.1
	PV-RCNN	✗	**65.4**	**64.8**	-	-
	RSN S_1f	✗	63.0	62.6	**65.4**	**60.7**
Transformer	Deformable DETR	✓	59.6	59.2	45.8	36.2
BoxeR-3D	**BoxeR-3D**	✓	63.9	63.7	61.5	53.7

Success & Failure

Ground-truth Vehicle prediction Pedestrian prediction





Khan et al., in press.

Further reading

Transformers in Vision: A Survey

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir,
Fahad Shahbaz Khan, and Mubarak Shah

Abstract—Astounding results from Transformer models on natural language tasks have intrigued the vision community to study their application to computer vision problems. Among their salient benefits, Transformers enable modeling long dependencies between input sequence elements and support parallel processing of sequence as compared to recurrent networks *e.g.*, Long short-term memory (LSTM). Different from convolutional networks, Transformers require minimal inductive biases for their design and are naturally suited as set-functions. Furthermore, the straightforward design of Transformers allows processing multiple modalities (*e.g.*, images, videos, text and speech) using similar processing blocks and demonstrates excellent scalability to very large capacity networks and huge datasets. These strengths have led to exciting progress on a number of vision tasks using Transformer networks. This survey aims to provide a comprehensive overview of the Transformer models in the computer vision discipline. We start with an introduction to fundamental concepts behind the success of Transformers i.e., self-attention, large-scale pre-training, and bidirectional feature encoding. We then cover extensive applications of transformers in vision including popular recognition tasks (*e.g.*, image classification, object detection, action recognition, and segmentation), generative modeling, multi-modal tasks (*e.g.*, visual-question answering, visual reasoning, and visual grounding), video processing (*e.g.*, activity recognition, video forecasting), low-level vision (*e.g.*, image super-resolution, image enhancement, and colorization) and 3D analysis (*e.g.*, point cloud classification and segmentation). We compare the respective advantages and limitations of popular techniques both in terms of architectural design and their experimental value. Finally, we provide an analysis on open research directions and possible future works. We hope this effort will ignite further interest in the community to solve current challenges towards the application of transformer models in computer vision.