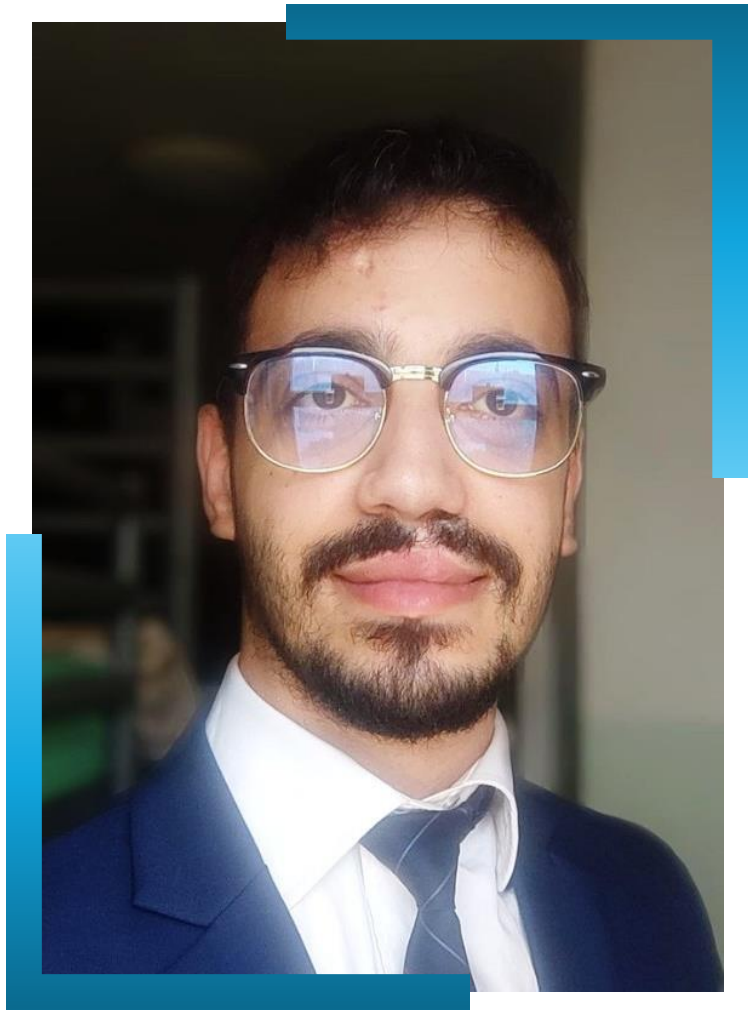
Transformers in Vision

AN INTRODUCTION

By:

Dr. Safouane El Ghazouali

BIO



Safouane El Ghazouali, Ph. D.

- Senior researcher in AI at TOELT
- Senior data scientist in Computer Vision
- Experienced in VLM / LLM inference and fine-tuning
- Buit a web app UI for fast annotation (VisioFirm)
- Experienced in training / fine-tuning of ML model for domain-specific application

Background

- Ph.D. in AI applied to 3D metrology, LNE, Paris
- Master in computer vision and embedded systems, Polytechnique Haut-de-France, Valenciennes

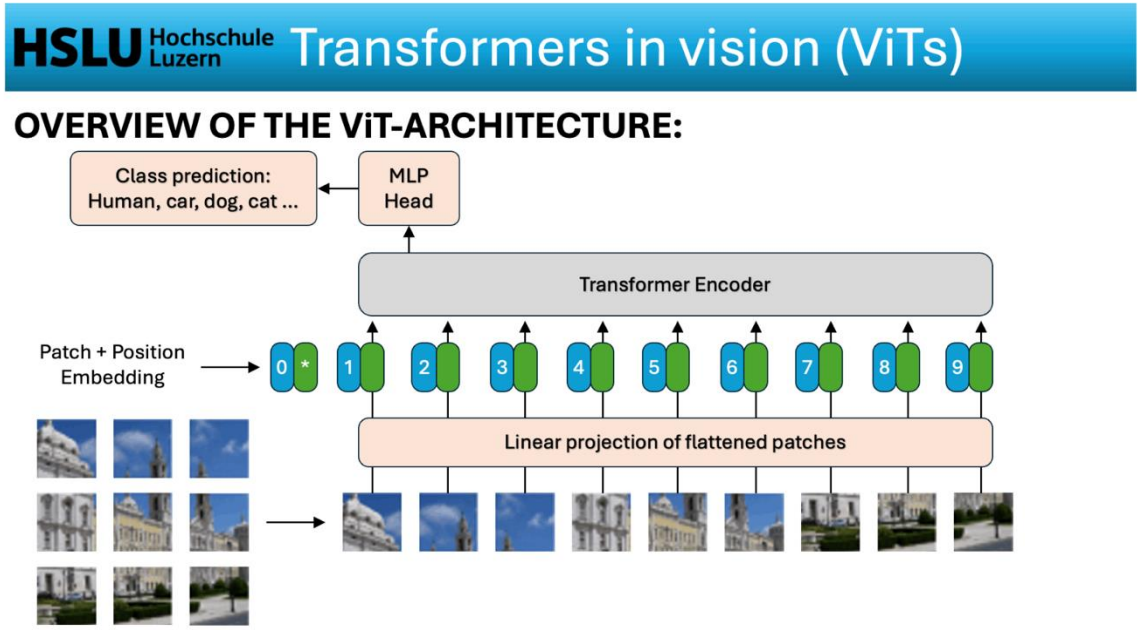
GitHub repo for the course :

https://github.com/safouaneelg/HSLU-Transformers-in-Vision

Transformers in Vision Course - HSLU

Welcome to the GitHub repository for the "Transformers in Vision" course at HSLU.

This repo contains materials and hands-on exercises for Day 3 and Day 4, from the bootcamp starting August 25, 2026.



AGENDA. DAY 1

- General Introduction
- Overview of neural architectures evolution
- Transformers in vision (ViTs)
- Divers ViTs architectures
  - Google ViT
  - Swin Transformer
- Hands-on (using “Torch” and ”timm” frameworks)
  - Pre-trained classification model
  - Visualization of features map
  - Finetuning ViT on classification task
- ViTs in object detection
- Hands-on Ultralytics (YOLO models)
  - Inferencing pre-trained model
  - Finetuning YOLOv10 on custom dataset
- Hugginface Transformers library
- Limitations of ViTs

AGENDA. DAY 2

- Background & Motivations
- Applications: image captioning, retrieval, grounding, generative tasks
- CLIP and Zero-Shot Learning
- Hands-on:
  - Inference CLIP for zero-shot classification
  - GroundingDINO for zero-shot detection
  - Prompt Engineering
- BLIP and Multimodal Generation
- Diffusion Model – introduction to Markov Chain
- Hands-on:
  - Stable-diffusion
  - Prompt engineering

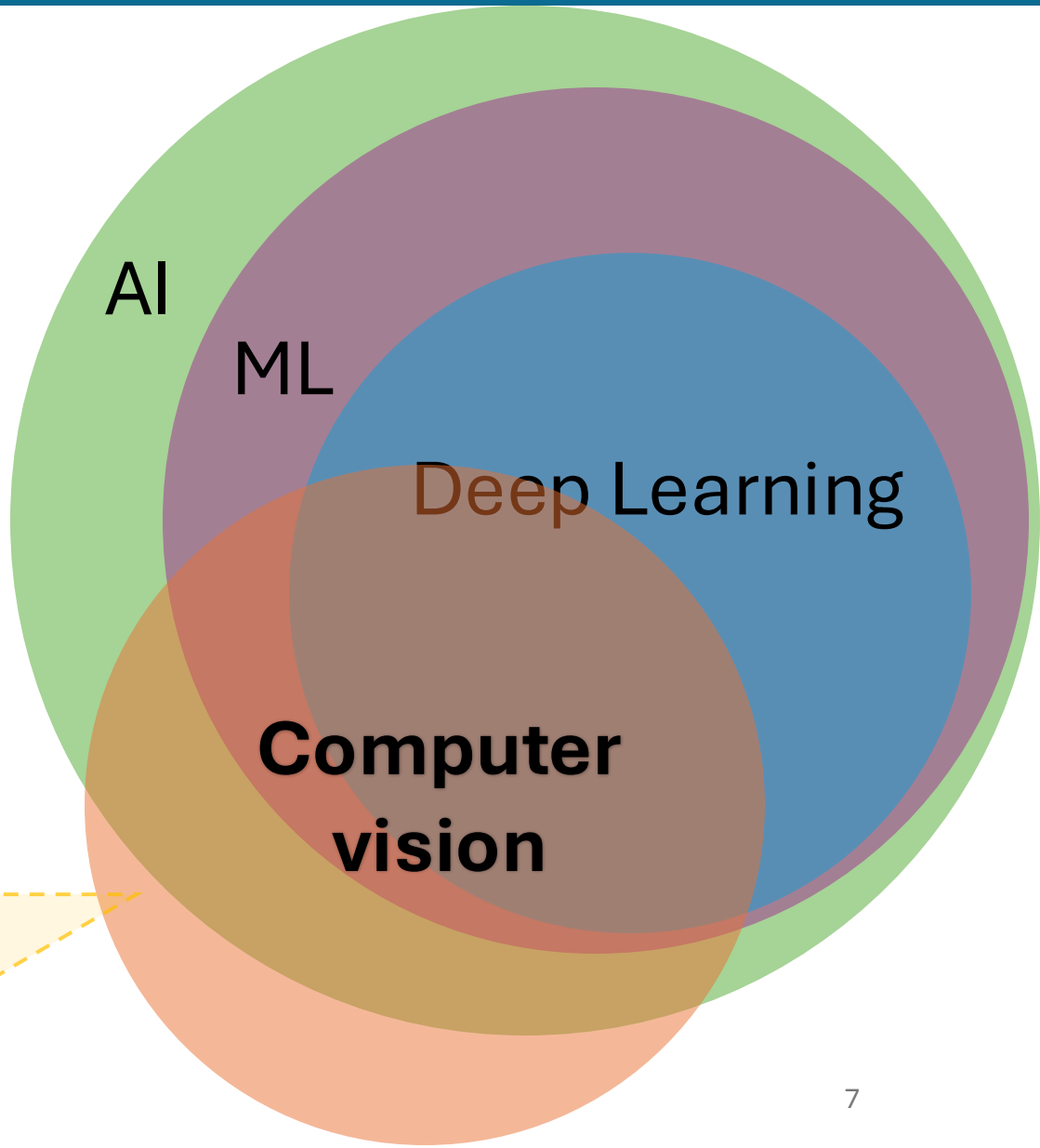
HSLU Hochschule Luzern

General introduction

General introduction

Computer vision includes:

1- Image/signal processing
2- 3D reconstruction
3- Feature extraction
4- Image restoration
5- Image classification
6- Object detection
7- Semantic/instance segmentation
8- Scene understanding
9- Image generation
10- Optical flow analysis
11- Pose estimation
12- Video tracking



Overview of neural architectures evolution

Computer vision evolution timeline

1950

Foundations in optics and early mathematical-based processing. Focus on understanding light, vision, pinhole and basic digital imaging.

1970

Shift to more structured NNs with hierarchical feature learning. Introduction of convolutional ideas for shift-invariant pattern recognition.

2017

Emergence of advanced modular architectures and powerful backbones able to extract deep features.
The most known backbones

- VGGNet 16–19 layers with uniform 3x3 convolutions for depth exploration
- AlexNet: 8-layer CNN winning ImageNet challenge
- DenseNet: Layer-wise connections for feature reuse

2020

Rise of pure vision transformers, outperforming CNNs on large datasets. Multimodal integration begins.
Vision Transformer: Patched images fed into transformers for classification.
DeiT (2021): Data-efficient ViT variant.
Swin Transformer (2021): Hierarchical shifted windows for efficiency.

HSLU Hochschule Luzern

Transformers in vision (ViTs)

Original Tranformer paper

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

First ViT paper

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

Transformers in vision (ViTs)

Objectives:

1. **Demonstrate the Applicability of Pure Transformers to Image Recognition**: Show that a standard Transformer architecture can be applied directly to sequences of image patches for classification tasks, without relying on convolutional neural networks (CNNs), thereby challenging CNN dominance in computer vision.
2. **Evaluate Performance on Large-Scale Pre-Training and Transfer Learning**: Assess the Vision Transformer (ViT) model's performance when pre-trained on massive datasets (e.g., ImageNet-21k or JFT-300M) and fine-tuned on various benchmarks (e.g., ImageNet, CIFAR-100, VTAB), comparing it to state-of-the-art CNNs.
3. **Investigate the Impact of Dataset Scale on Transformer Performance**: Explore how training dataset size influences ViT's effectiveness, emphasizing that large-scale data can compensate for Transformers' lack of built-in inductive biases like those in CNNs (e.g., translation equivariance and locality).
4. **Compare Computational Efficiency and Resource Requirements**: Highlight that ViT achieves strong results with significantly lower computational costs during training compared to leading CNNs, using metrics such as TPUv3-core-days.

Transformers in vision (ViTs)

Aspect	NLP Tokenization	Vision Tokenization
Definition	Process of breaking down text into smaller units (tokens) such as words, subwords, or characters.	Process of dividing an image into fixed-size patches or regions to process as tokens.
Normalization	- Converts text to a standard format (e.g., lowercasing, removing accents). - Removes punctuation or special characters. - Handles contractions (e.g., "don't" → "do not").	- Normalizes pixel values (e.g., scaling to [0,1] or standardizing with mean and std). - Adjusts color channels (e.g., RGB normalization). - Resizes images to a fixed resolution.
Segmentation/Splitting	- Splits text into tokens based on whitespace, punctuation, or subword units (e.g., WordPiece, BPE). - Example: "I am running" → ["I", "am", "running"]. - Subword tokenization for rare words (e.g., "unhappiness" → ["un", "##happiness"]).	- Divides image into fixed-size patches (e.g., 16x16 pixels). - Each patch is treated as a token. - Example: A 224x224 image with 16x16 patches → 196 patches (tokens).
Cleanup	- Removes irrelevant tokens (e.g., extra whitespace, stop words like "the"). - Filters out low-frequency tokens or replaces them with [UNK]. - Handles encoding issues (e.g., UTF-8 normalization).	- Removes or adjusts noisy patches (e.g., handling corrupted pixels). - Applies data augmentation (e.g., random cropping, flipping) to enhance robustness. - Filters out irrelevant regions (e.g., background suppression in some cases).
Token Representation	- Tokens are mapped to vocabulary indices or embeddings (e.g., Word2Vec, BERT embeddings). - Each token is a discrete unit in a vocabulary.	- Patches are flattened and projected to a fixed-dimensional embedding space (e.g., linear projection in ViT). - Each patch is a vector in a continuous space.
Handling Variability	- Handles linguistic variations (e.g., synonyms, misspellings, multilingual text). - Uses subword tokenization to manage out-of-vocabulary words.	- Handles variations in lighting, orientation

Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:

Transformers in vision (ViTs)

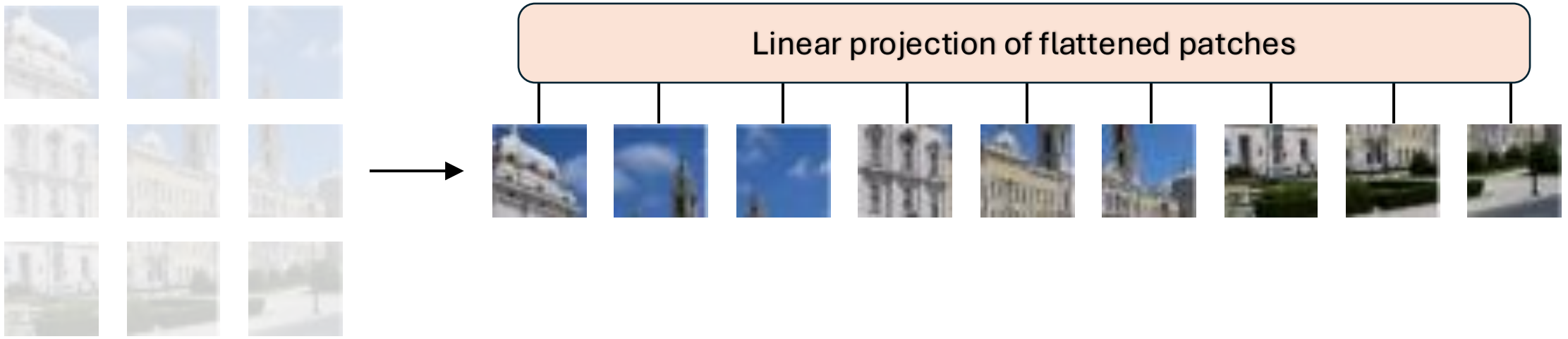
OVERVIEW OF THE ViT-ARCHITECTURE:

Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:

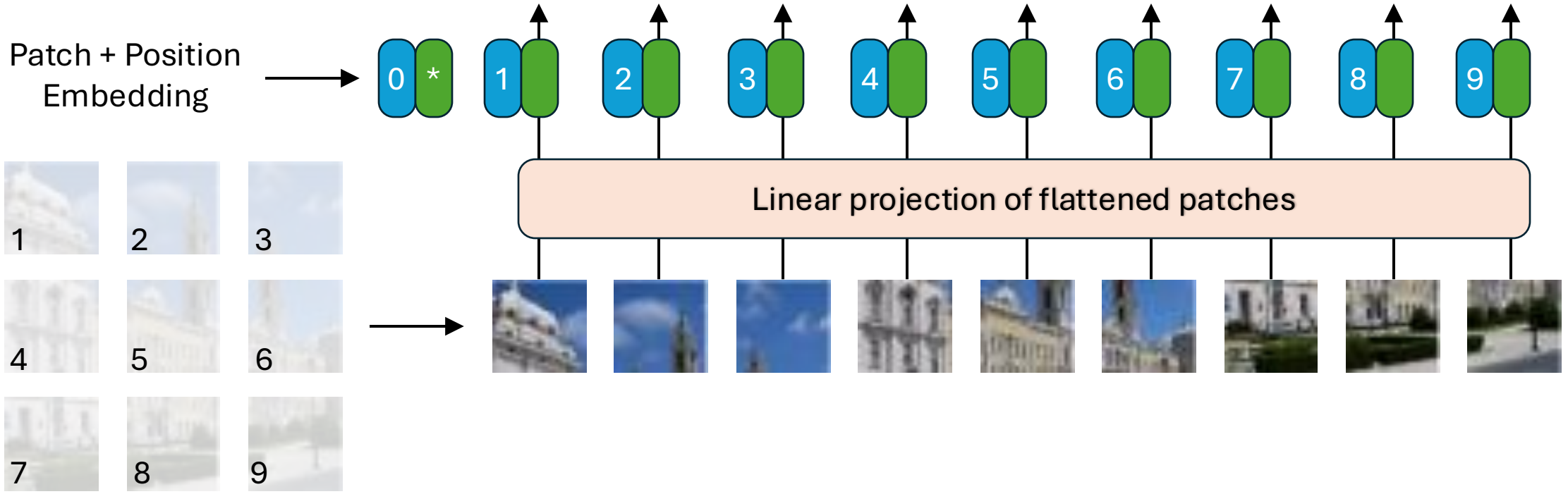
Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:



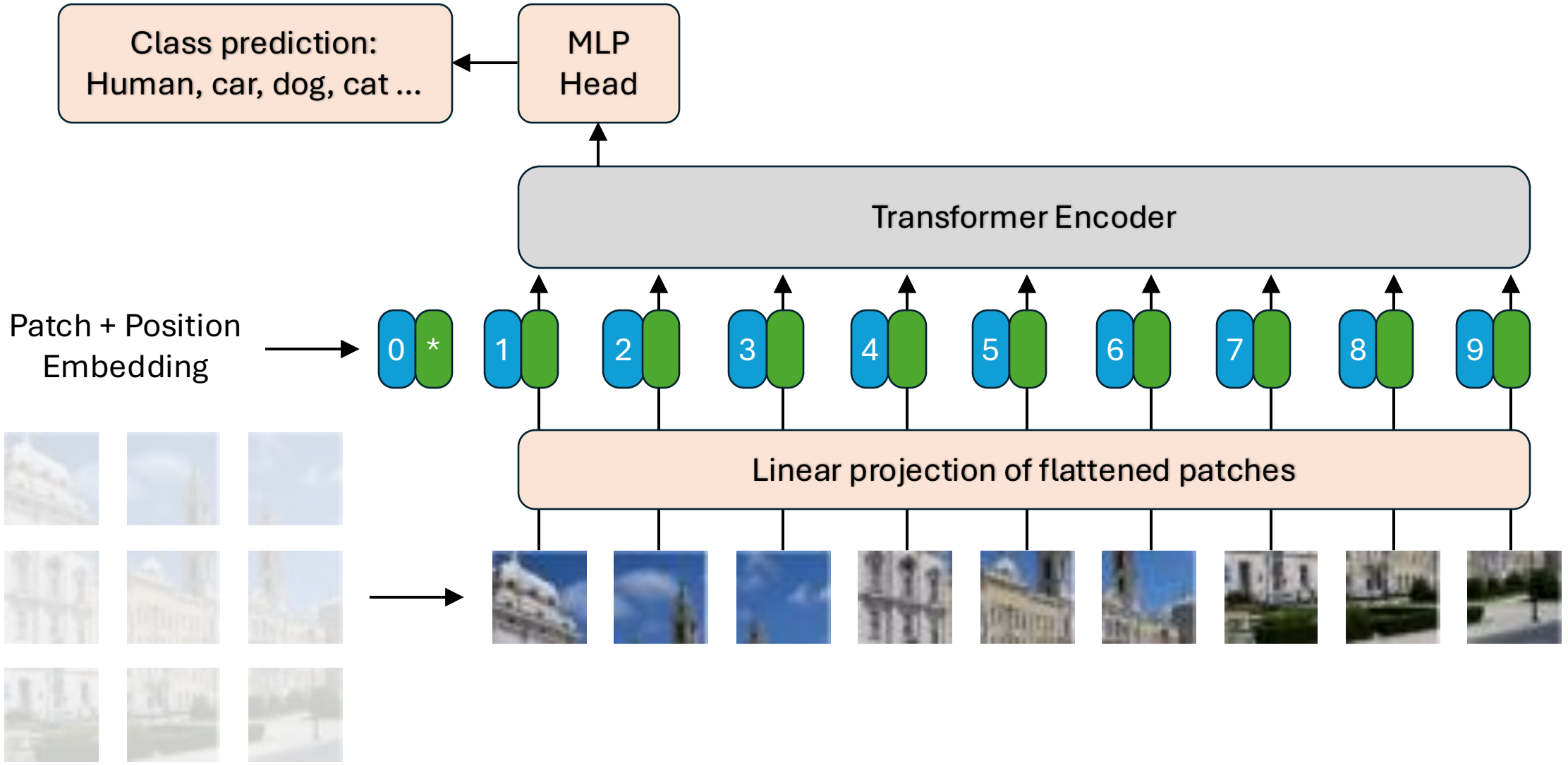
Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:



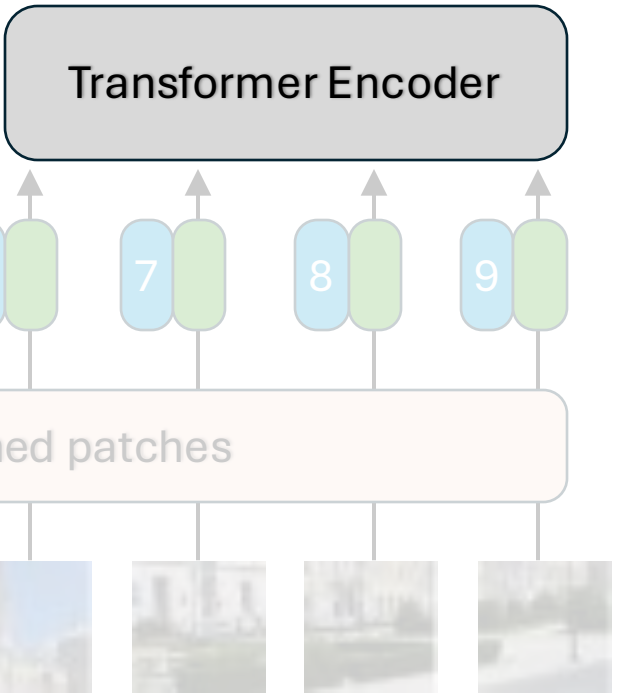
Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:



Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:



What is the "Transformer Encoder" how different / similar to NLP is it?

Both encoders are fundamentally similar in architecture and operation, but there are key differences due to the nature of the input data

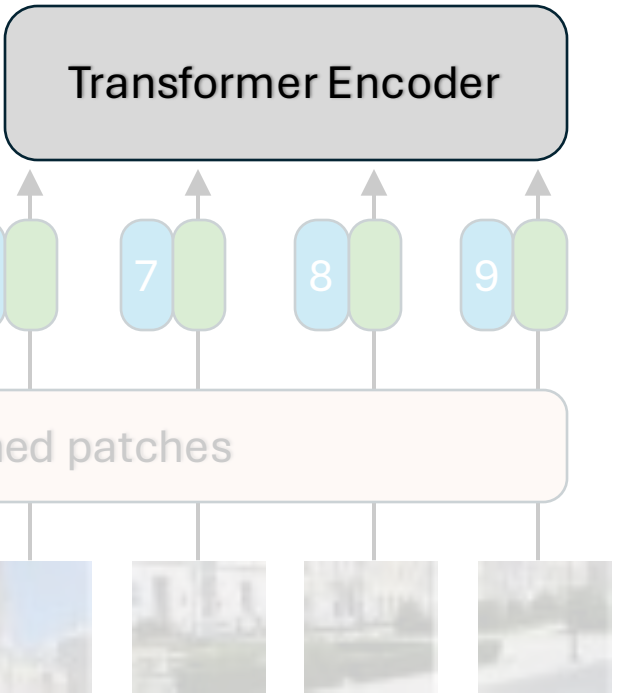
Core Similarity

- **Multi-Head Self-Attention**: Computes relationships between all input tokens/patches, capturing global dependencies.
- **Feed-Forward Networks (FFNs)**: Applied independently to each token/patch for feature transformation.
- **Layer Normalization and Residual Connections**: Stabilize training and improve gradient flow.
- **Positional Encoding/Embeddings**: Preserve sequence or spatial order.

The encoder processes a sequence of input tokens (or patches in ViTs) and produces contextualized representations by attending to all elements in the sequence simultaneously.

Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:



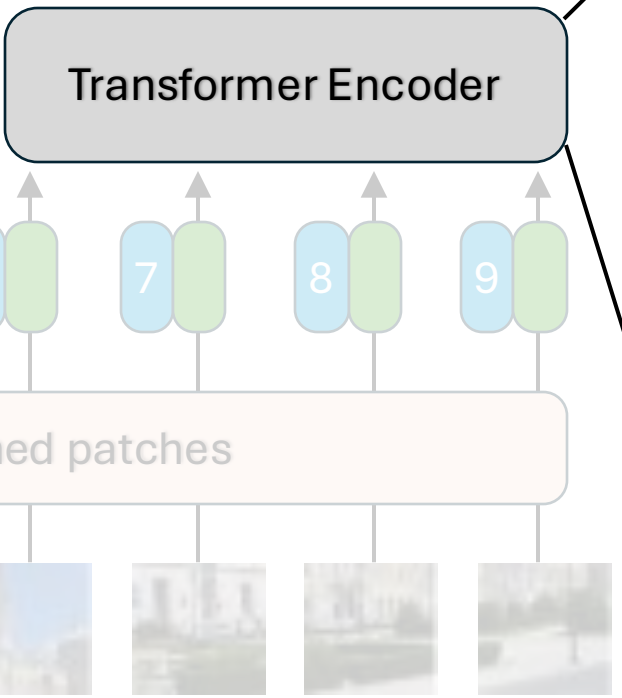
What is the "Transformer Encoder" how different / similar to NLP is it?

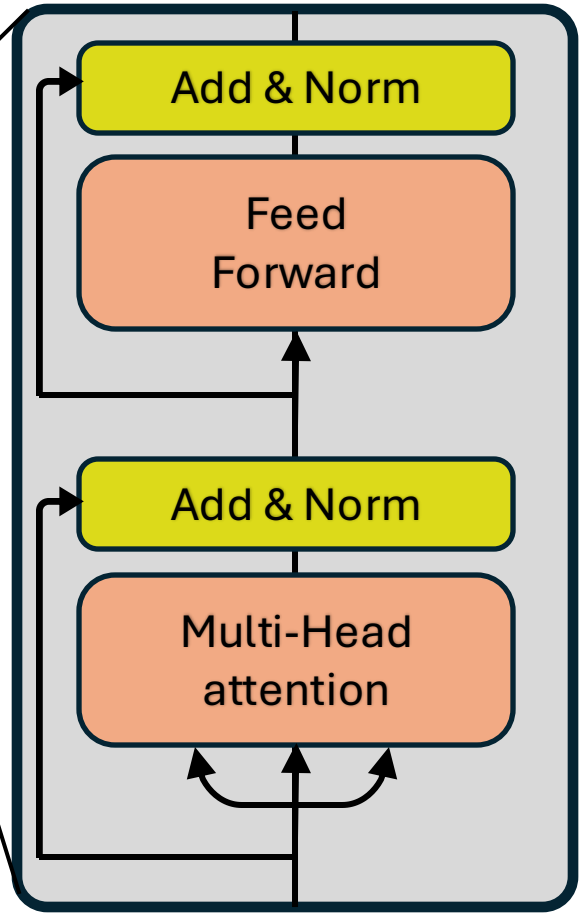
Differences

- **Input**
  - **NLP**: Takes a sequence of tokenized text (such as words, subwords) and tokens are mapped to embeddings via a vocabulary lookup (e.g., WordPiece)
  - **Vision**: Takes a sequence of image patches (e.g., 16x16 pixel patches). And the patches are flattened and linearly projected to a fixed-dimensional embedding space.
- **Positional Encoding**
  - **NLP**: In natural language, the input is a 1D sequence of words. The position can be represented by a single index, like 1, 2, 3, and so on. The positional encoding is therefore a 1D vector corresponding to each word's index.
  - **Vision**: Images are 2D. The position of a patch is defined by its row and column coordinates. While some ViTs use a flattened 1D sequence of patches (e.g., from top-left to bottom-right), more advanced methods might use a 2D encoding that explicitly models the row and column positions

Transformers in vision (ViTs)

OVERVIEW OF THE ViT-ARCHITECTURE:





Residual Connection & **Layer Normalization**
(Skip connections)

$$head_i = Attention(Q * W_i^Q, K * W_i^K, V * W_i^V)$$

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

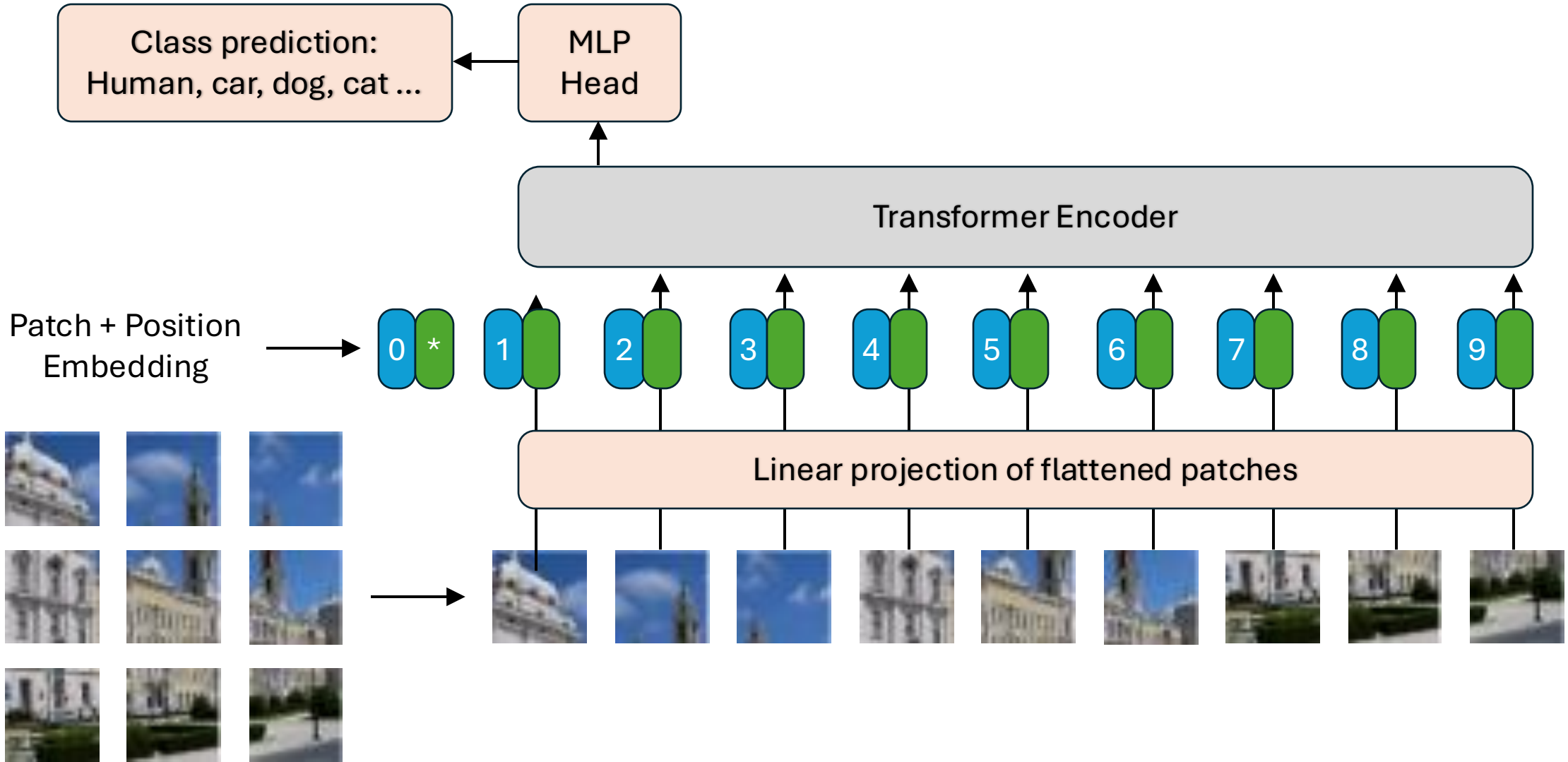
Q (Query), K (Key), and V (Value)

HSLU Hochschule Luzern

Divers ViTs architectures

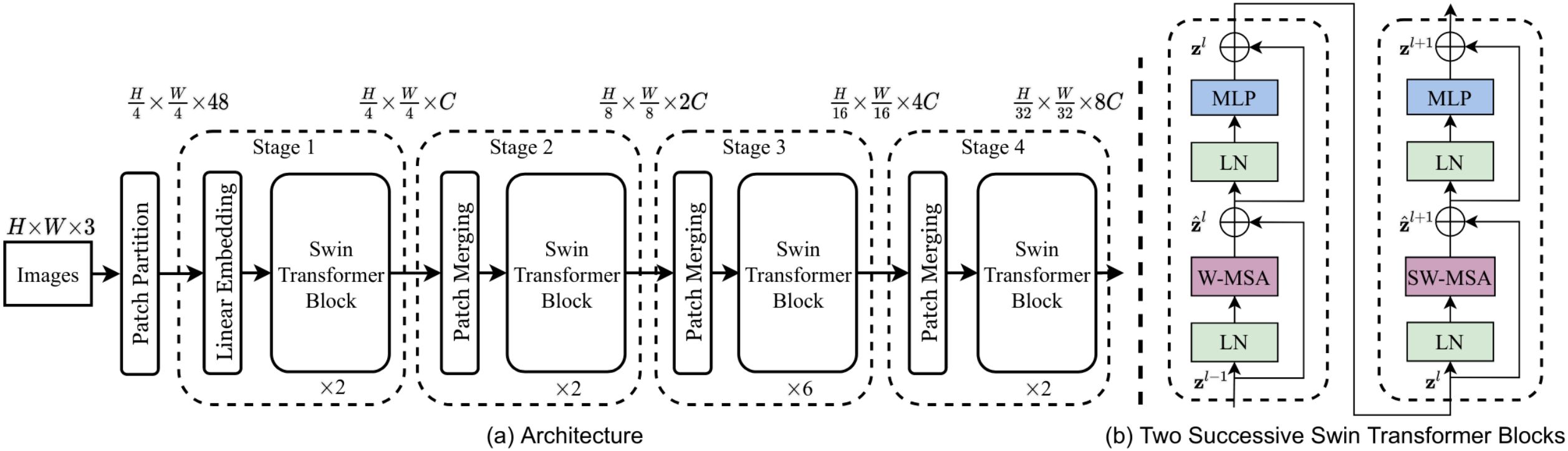
Divers ViTs architectures

ViT : Google - Vision Transformer (paper)



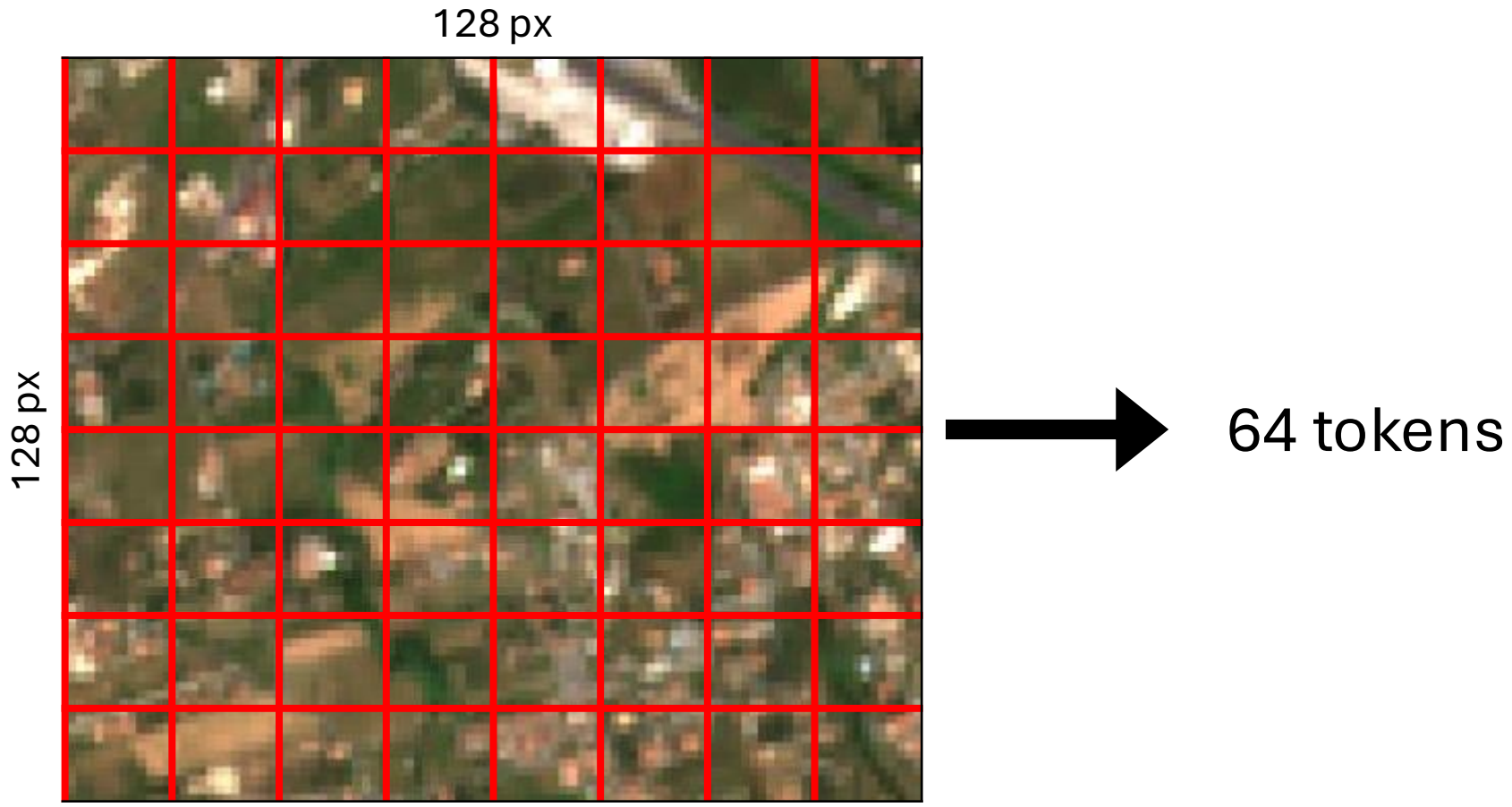
Divers ViTs architectures

Microsoft – Swin-Transformer (paper)



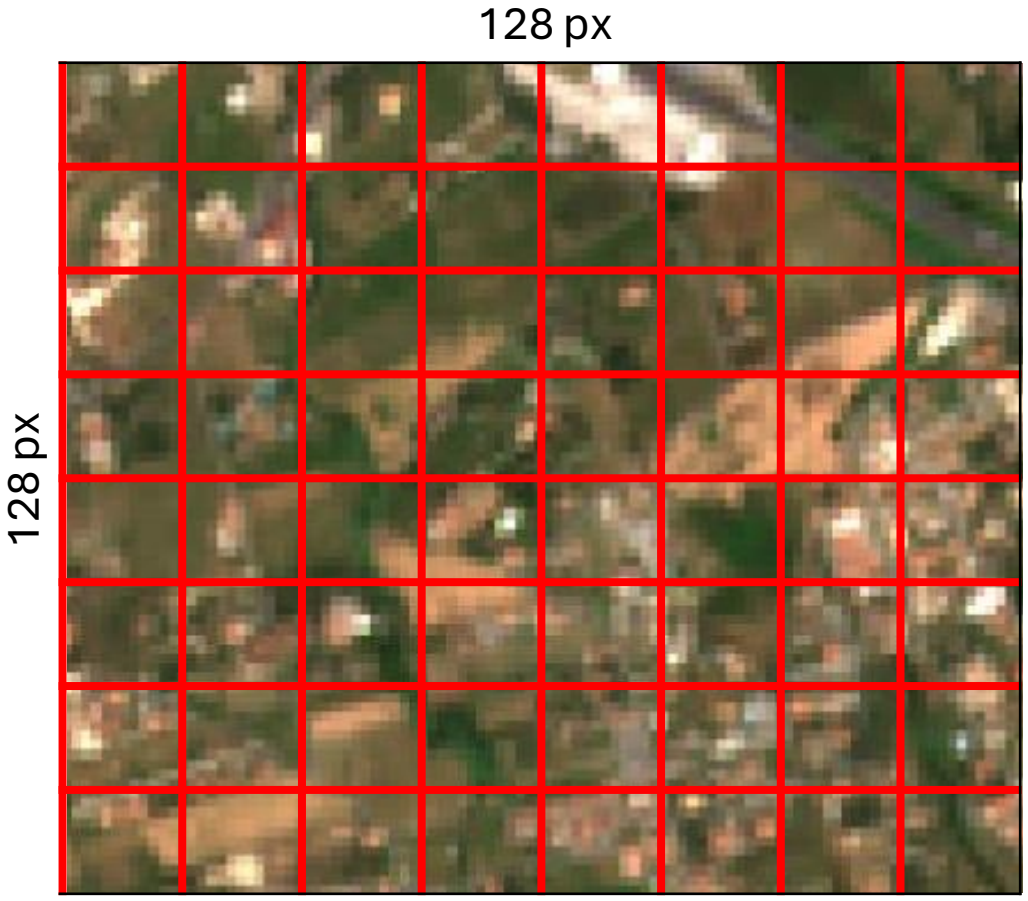
Divers ViTs architectures

Microsoft – Swin-Transformer (paper)

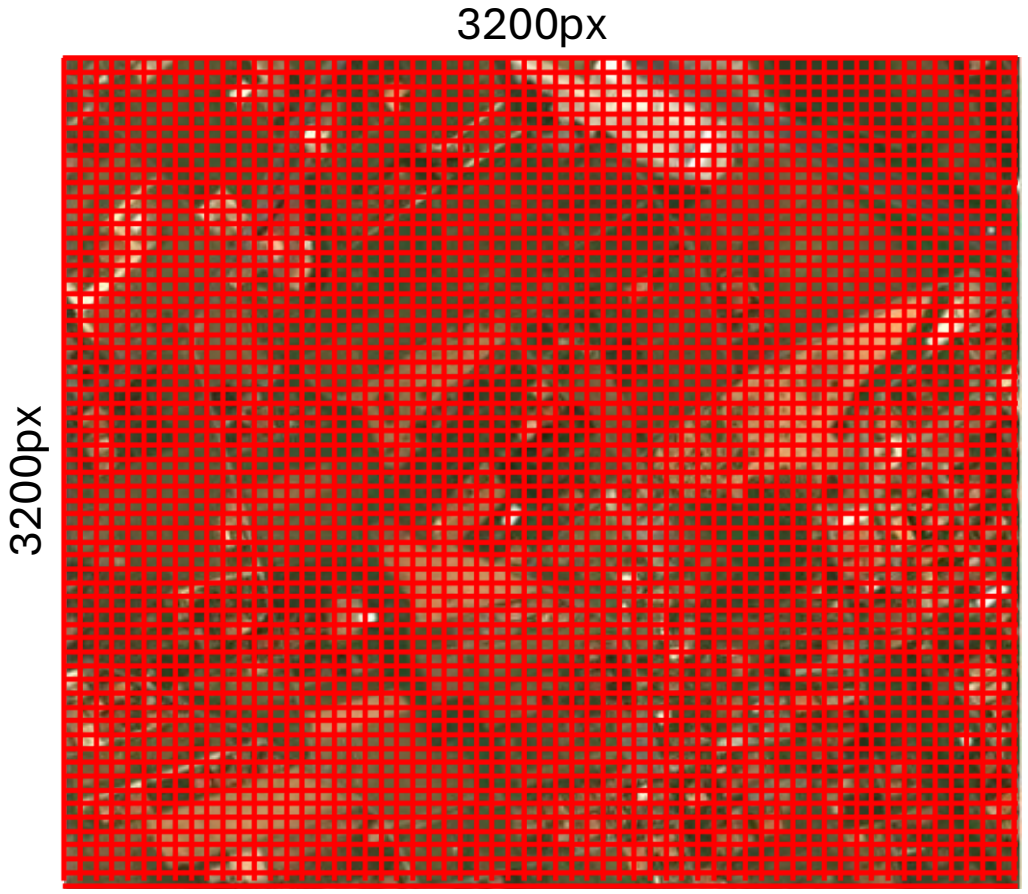


Divers ViTs architectures

Microsoft – Swin-Transformer (paper)



(128 / 16 x 128 /16) = 64 tokens



(3200 / 16 x 3200 /16) = 40000 tokens

Divers ViTs architectures

Microsoft – Swin-Transformer (paper)

The Swin Transformer was developed to solve two main problems with the original Vision Transformer (ViT):

quadratic computational complexity

The original ViT treats an image as a sequence of non-overlapping patches and applies a standard Transformer encoder to them. A key operation within the Transformer is **self-attention**, which calculates the relationship between every patch and every other patch. The computational cost of this operation grows quadratically with the number of patches ($O(n^2)$), where n is the number of patches.

- A high-resolution image needs more patches
- The quadratic cost makes the model very slow and memory-intensive, impractical for tasks like object detection or semantic segmentation

lack of hierarchical feature representation.

The ViT architecture processes patches at a single, fixed resolution throughout the network. It doesn't have a hierarchical structure like a Convolutional Neural Network (CNN), which builds a pyramid of features from low-level details (edges, textures) to high-level semantic concepts. This makes ViT less suitable for "dense prediction" tasks like object detection and segmentation, where objects of different sizes need to be detected and a multi-scale representation is crucial.

Divers ViTs architectures

Microsoft – Swin-Transformer (paper)

The Swin Transformer solutions

quadratic computational complexity

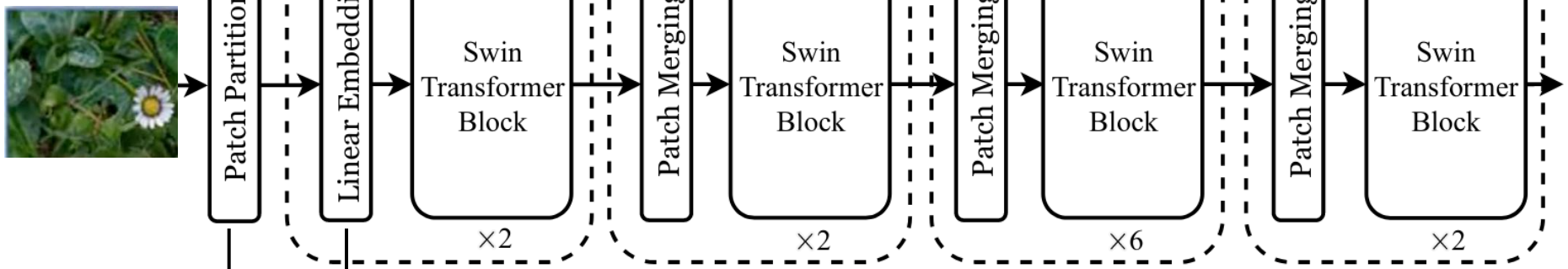
The Swin Transformer limits the self-attention computation to **non-overlapping local windows**. This changes the complexity from quadratic to **linear** with respect to the number of patches. To ensure information can still flow between these windows, Swin introduces a **shifted window** mechanism. In alternating layers, the windows are shifted, allowing patches to interact with different neighbors and learn global context over the network's depth.

lack of hierarchical feature representation.

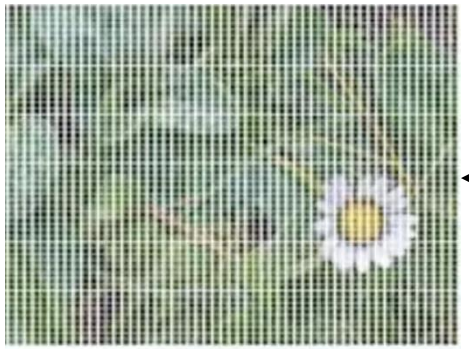
The Swin Transformer introduces a **hierarchical architecture** that produces multi-scale feature maps, just like a CNN. It starts with small patches and then gradually merges adjacent patches in deeper layers. This downsampling process creates a feature pyramid that can capture both fine-grained details for small objects and high-level semantic information for large objects.

Divers ViTs architectures

Microsoft – Swin-Transformer (paper)



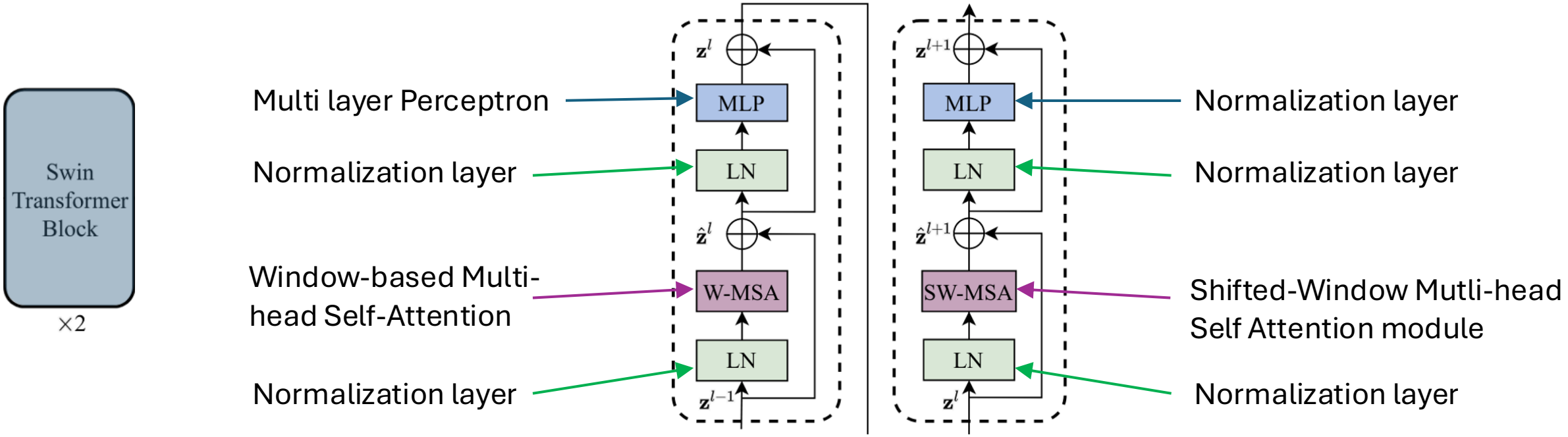
4x4 px



```
First patch embedding vector: tensor([ 0.0872,  0.0503, -0.1719,  0.1464,  0.1546, -0.0358,  0.0016, -0.3824,
       -0.2313, -0.1955,  0.1172, -0.0174, -0.1295, -0.0316,  0.0104, -0.0479,
        0.0975, -0.0499, -0.0125, -0.1404, -0.2292,  0.1697, -0.2276, -0.0672,
        0.0015, -0.0895,  0.1411, -0.0210,  0.0128,  0.1669,  0.0921, -0.1980,
       -0.2184,  0.0075,  0.1579,  0.0842,  0.0635, -0.0160,  0.2636,  0.1215,
        0.1866, -0.2069,  0.0335,  0.0802,  0.1717, -0.1648,  0.2139,  0.1257,
       -0.1655, -0.0129, -0.1318, -0.1114,  0.1215, -0.0470,  0.0056, -0.2260,
       -0.0460,  0.0295, -0.0118, -0.4155,  0.0999, -0.0171,  0.1431, -0.0716,
       -0.0708,  0.0945, -0.1236,  0.0800, -0.1344,  0.0319,  0.0184,  0.1027,
        0.0112, -0.0439, -0.2414,  0.0518,  0.1925, -0.0069,  0.0818,  0.0732,
        0.1824,  0.1271,  0.0092,  0.0964, -0.0937,  0.1673, -0.2281, -0.0448,
       -0.1030, -0.0190,  0.1538, -0.0324,  0.2129,  0.1216,  0.0420, -0.2114],
```

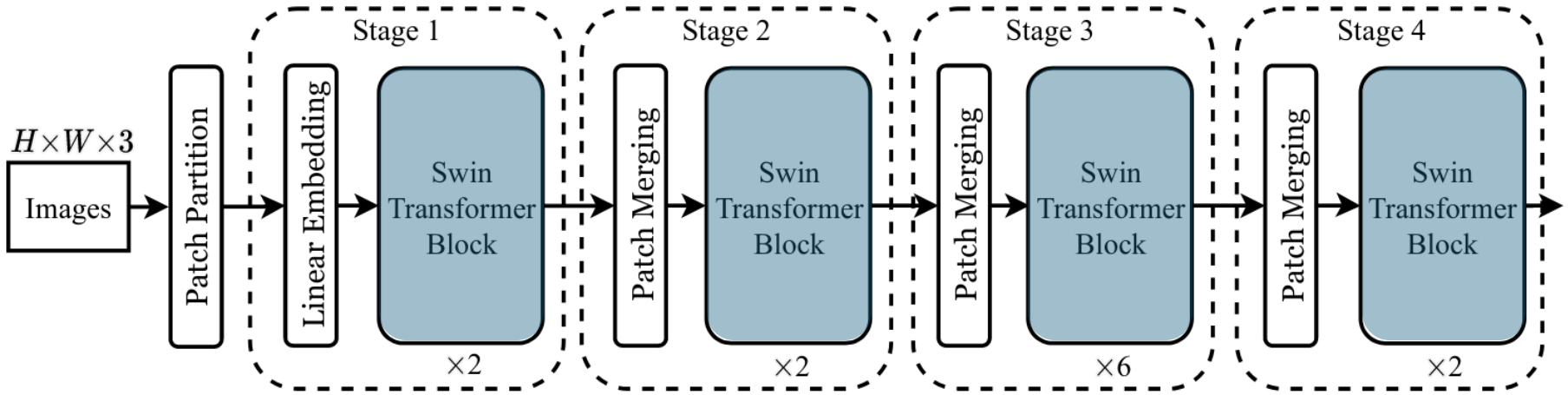
Divers ViTs architectures

Microsoft – Swin-Transformer (paper)



Divers ViTs architectures

Microsoft – Swin-Transformer (paper)



HSLU Hochschule Luzern

Hands-on:

(1) Pre-trained classification model

HSLU Hochschule Luzern

Hands-on:

(2) Visualization of features map

HSLU Hochschule Luzern

Hands-on:

(3) Finetuning ViT on classification task

HSLU Hochschule Luzern

ViTs in object detection

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

Backbone Encoder Decoder Prediction heads

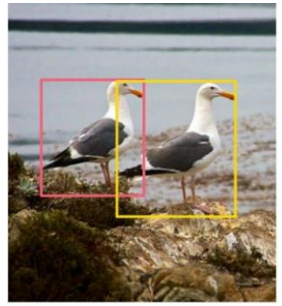


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

Backbone Encoder Decoder Prediction heads

Pretrained CNN: VGG – ResNet ...etc

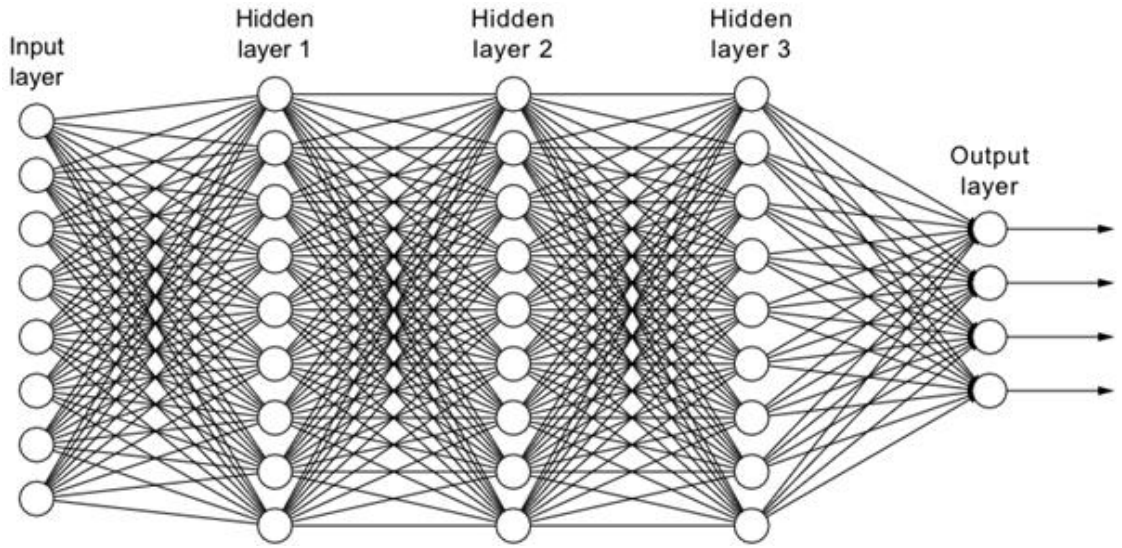


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

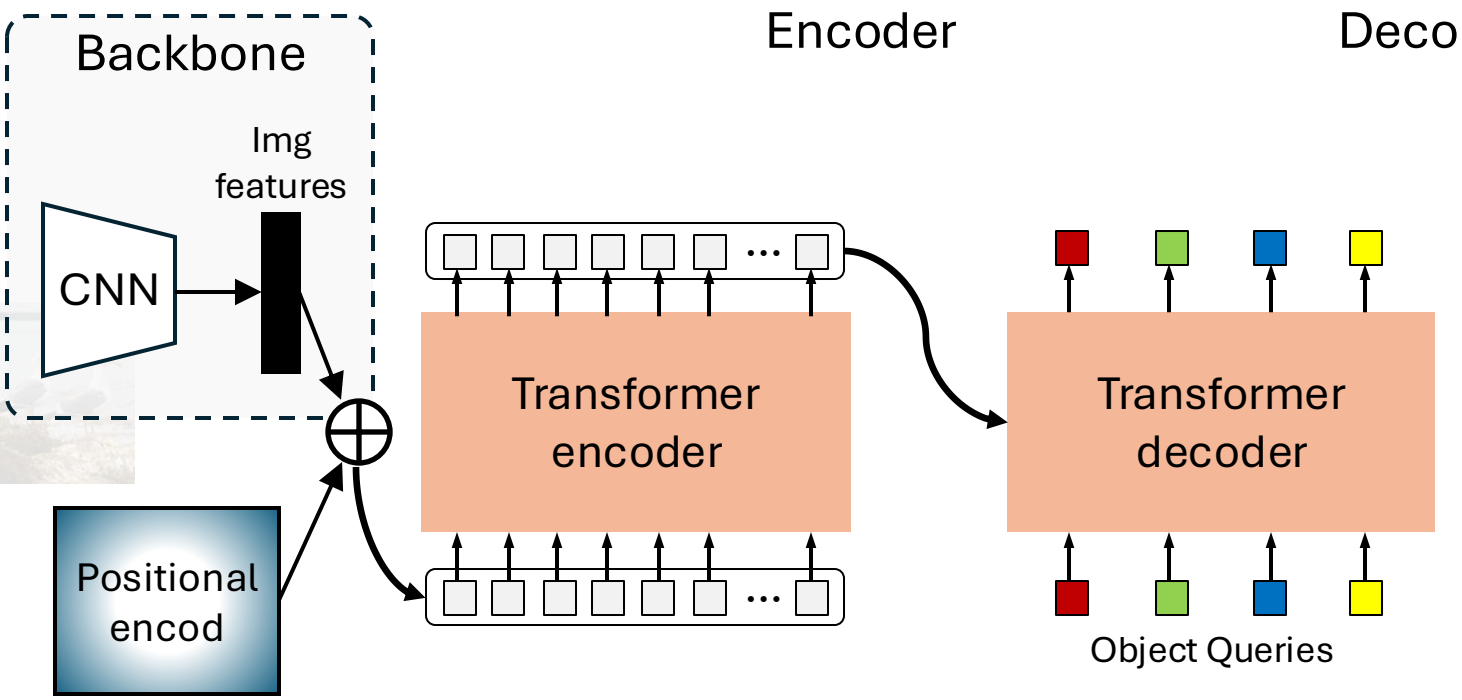


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

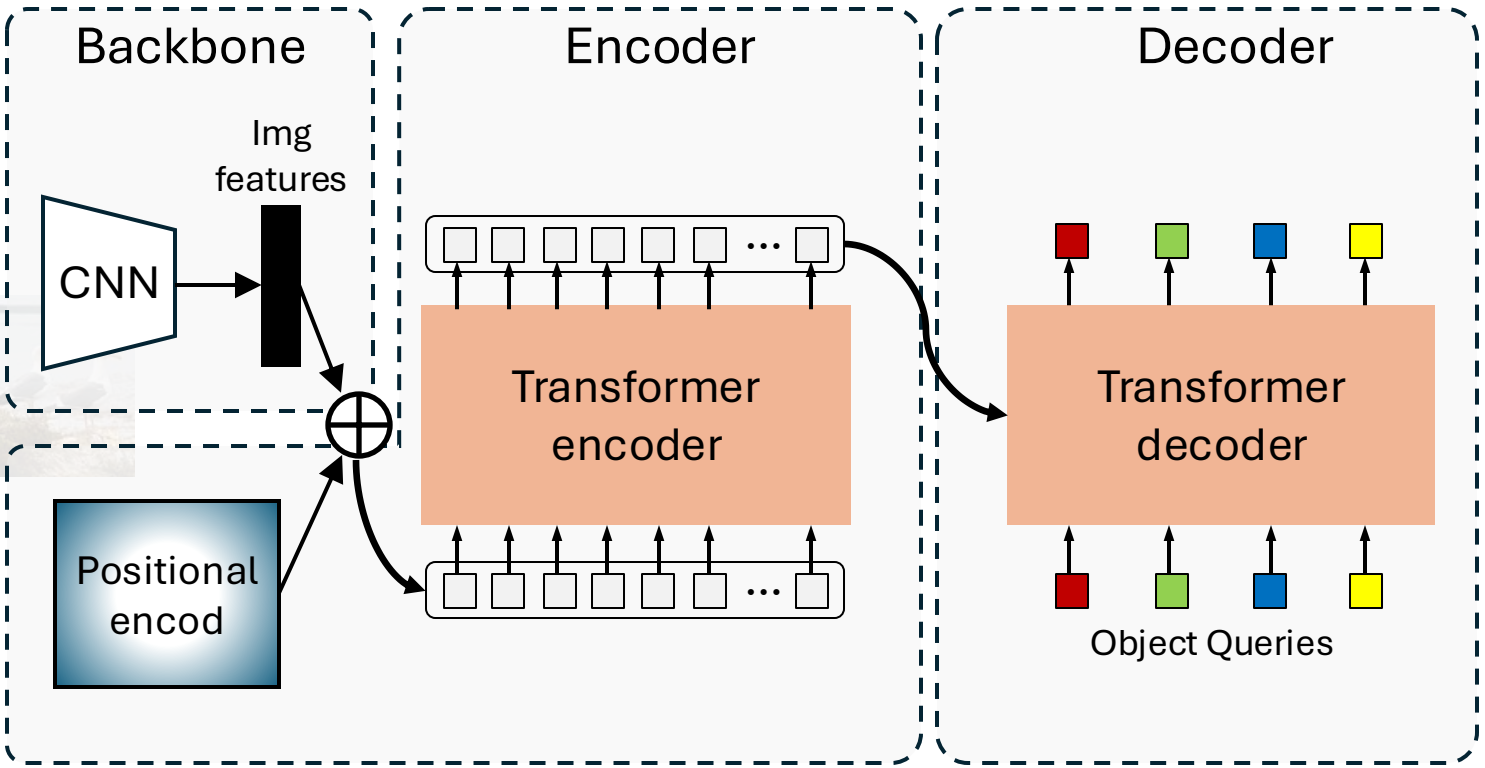
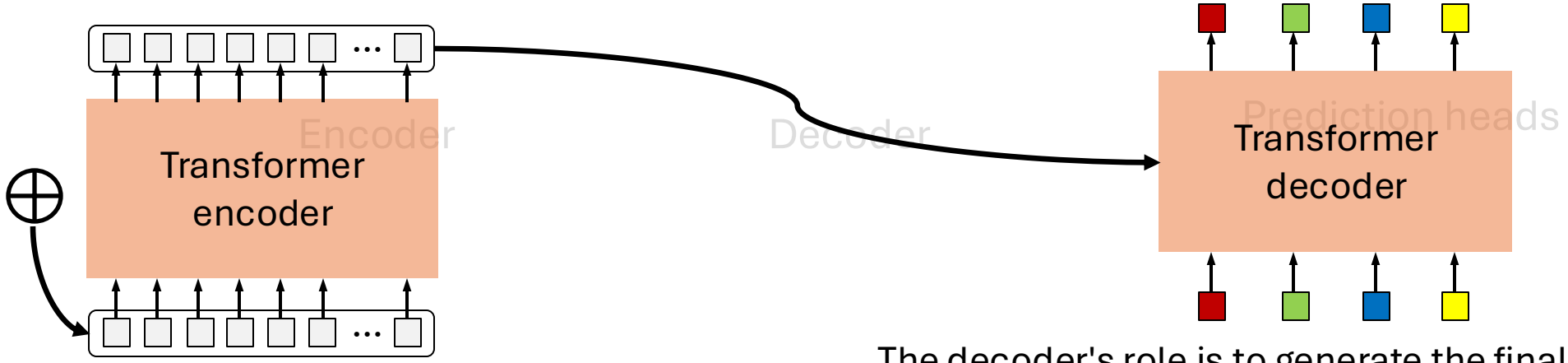


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)



The encoder consists of multiple layers, each with a multi-head self-attention mechanism. This mechanism allows the model to reason about the global context of the image by enabling each feature vector to attend to all other feature vectors. This helps the encoder understand the relationships between objects, their parts, and the background, regardless of their distance from each other. The output is a set of contextualized feature representations.

The decoder's role is to generate the final object predictions (class labels and bounding boxes) by using the contextualized features from the encoder.

- **Object Queries**: The decoder's input is a fixed, small set of learned embeddings called **object queries**. These queries are abstract, learnable vectors that can be thought of as "proposals" for potential objects in the image.
- **Cross-Attention**: The decoder uses a multi-head cross-attention mechanism to "interact" with the encoder's output.
- **Self-Attention**: The decoder also uses SA among the object queries themselves.

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

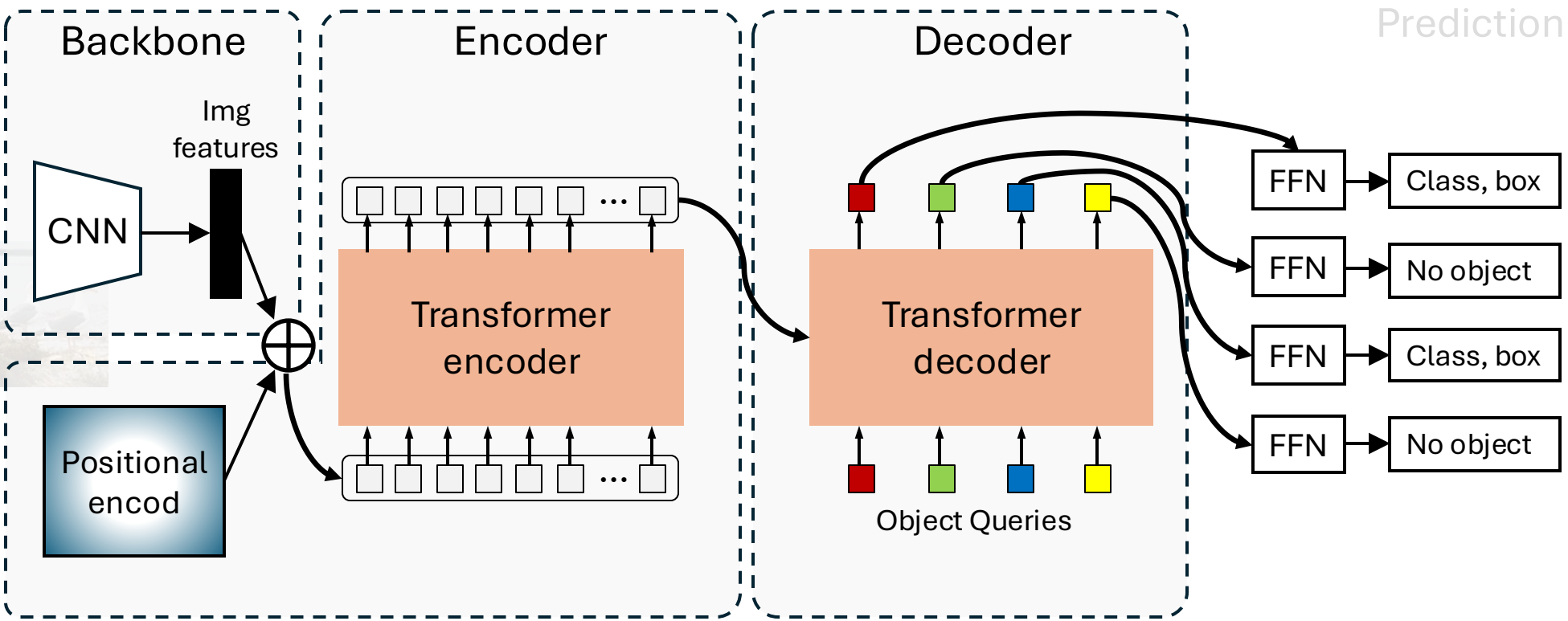


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

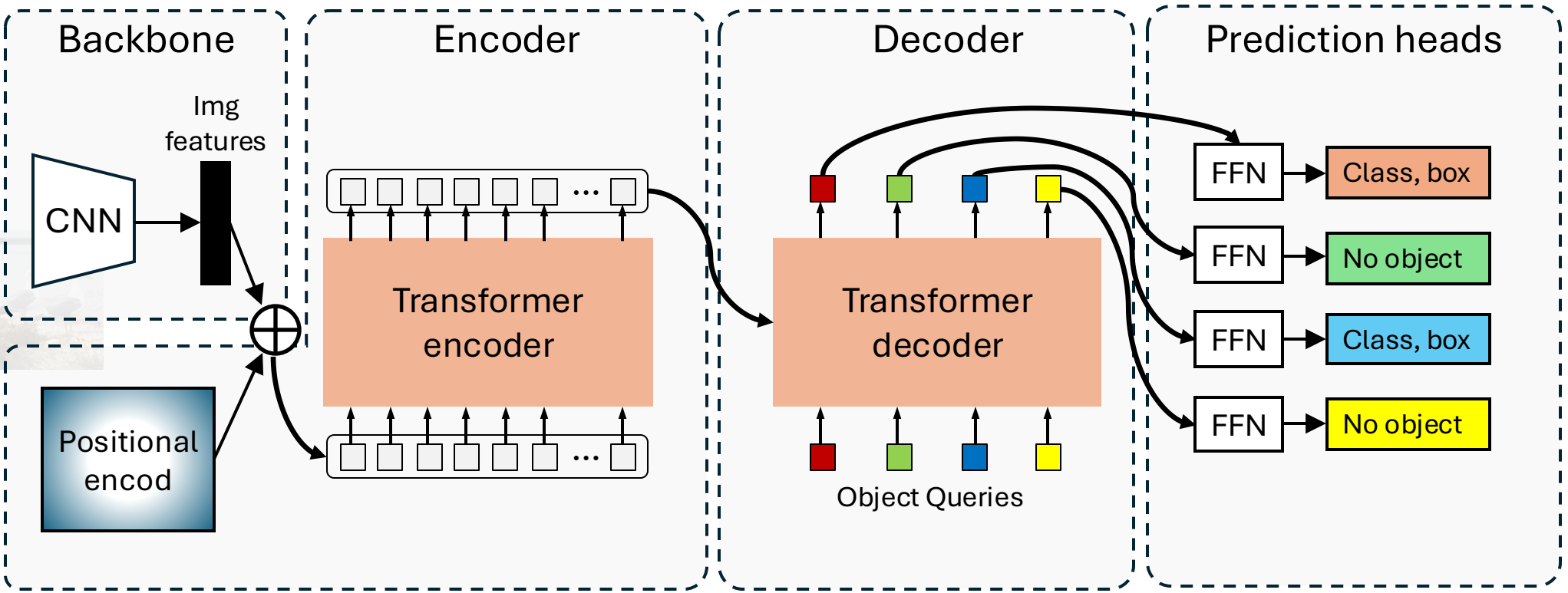


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

DETR: End-to-end Detection TRansformer (paper)

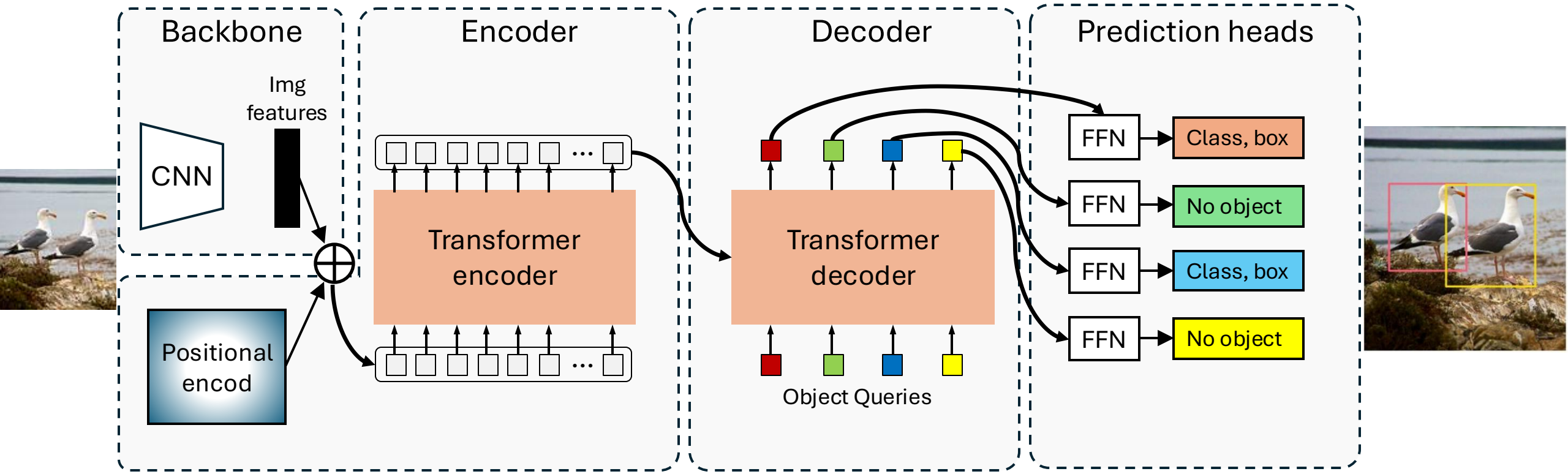


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

YOLO series - v10: YOLOv10: Real-Time End-to-End Object Detection

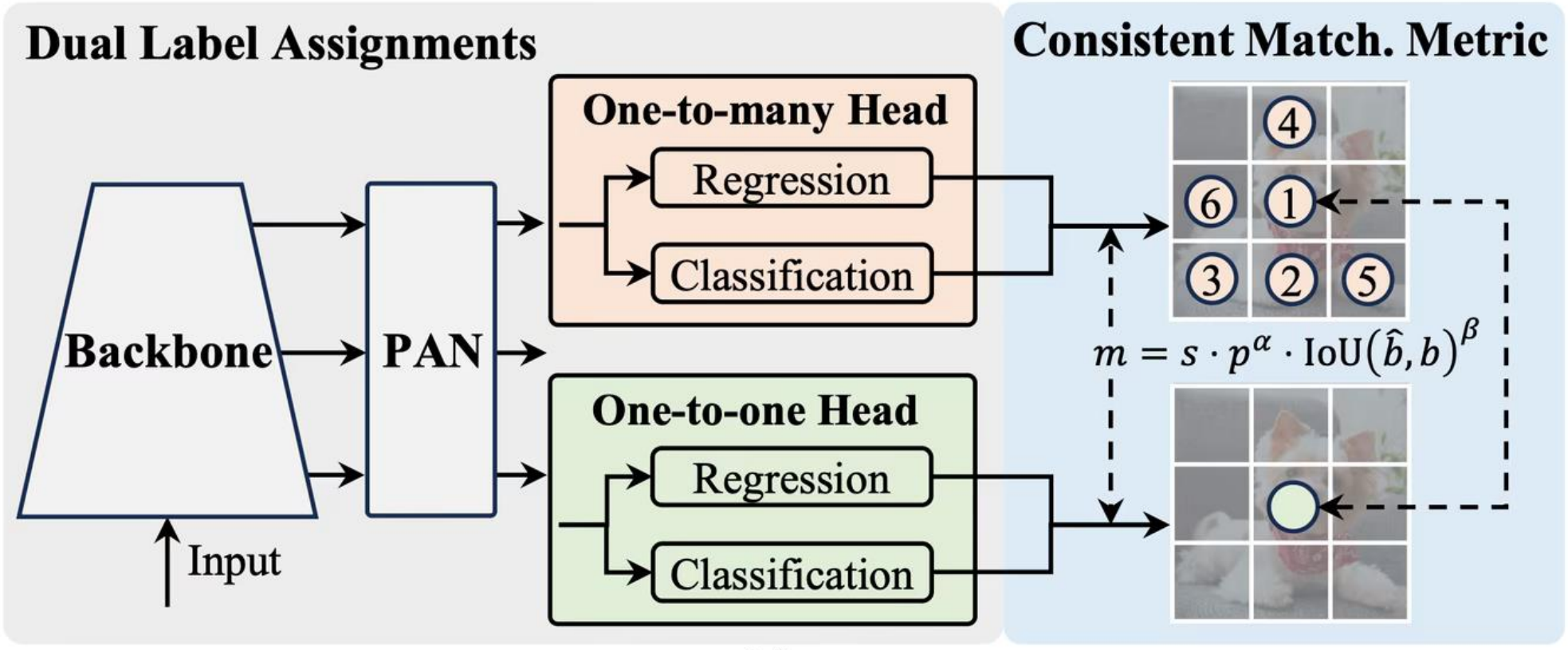
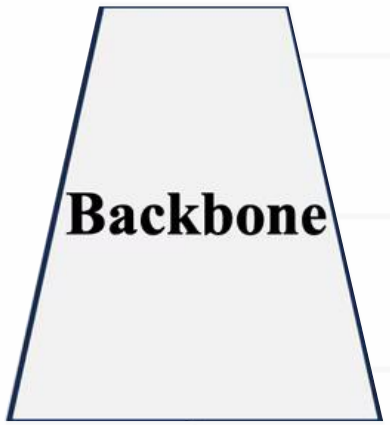


Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

YOLO series - v10: YOLOv10: Real-Time End-to-End Object Detection



Pretrained image features extraction model:

- VGG
- ResNet
- MobileNet
- …etc

Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

YOLO series - v10: YOLOv10: Real-Time End-to-End Object Detection

PAN - path aggregation network :

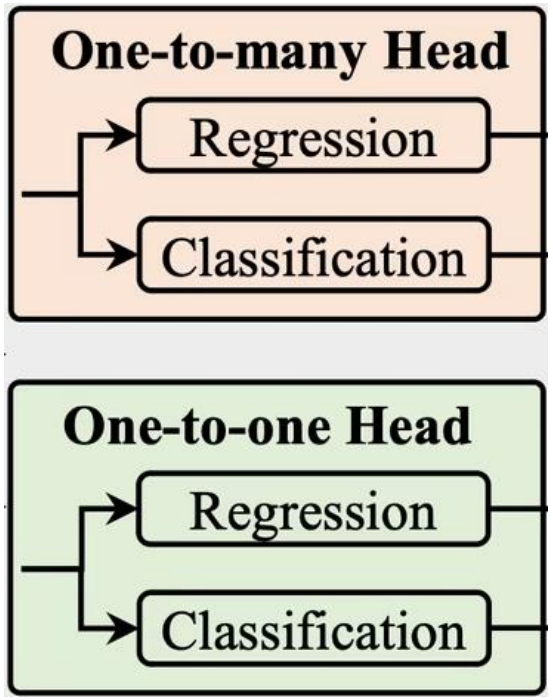
sits between the backbone and the detection heads.

Its primary purpose is to collect feature maps from different layers of the backbone and fuse them to create a richer, multi-scale representation.

Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

YOLO series - v10: YOLOv10: Real-Time End-to-End Object Detection



YOLOv10 also incorporates large-kernel convolution and self-attention mechanisms to enhance accuracy with minimal computational cost:

1.Large-Kernel Convolution: Employing large-kernel depthwise convolution effectively enlarges the receptive field and enhances the model's capability. However, using them in all stages can contaminate shallow features needed for detecting small objects and introduce significant I/O overhead and latency in high-resolution stages.

2.Partial Self-Attention (PSA): Self-attention is known for its global modeling capability but comes with high computational complexity and memory footprint. YOLOv10 introduces an efficient partial self-attention (PSA) module to mitigate these issues. After a 1×1 convolution, the features are evenly partitioned across channels into two parts. PSA is placed after Stage 4 with the lowest resolution, minimizing the overhead from the quadratic computational complexity of self-attention. This design incorporates global representation learning into YOLOv10 with low computational costs, enhancing the model's capability and improving performance.

Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

ViTs in object detection

YOLO series - v10: YOLOv10: Real-Time End-to-End Object Detection

Model	#Param.(M)	FLOPs(G)	AP^{val}(%)	Latency(ms)	$Latency^{f}$(ms)
YOLOv6-3.0-N [27]	4.7	11.4	37.0	2.69	1.76
Gold-YOLO-N [54]	5.6	12.1	39.6	2.92	1.82
YOLOv8-N [20]	3.2	8.7	37.3	6.16	1.77
YOLOv10-N (Ours)	**2.3**	**6.7**	**38.5 / 39.5**†	**1.84**	**1.79**
YOLOv6-3.0-S [27]	18.5	45.3	44.3	3.42	2.35
Gold-YOLO-S [54]	21.5	46.0	45.4	3.82	2.73
YOLO-MS-XS [7]	4.5	17.4	43.4	8.23	2.80
YOLO-MS-S [7]	8.1	31.2	46.2	10.12	4.83
YOLOv8-S [20]	11.2	28.6	44.9	7.07	2.33
YOLOv9-S [59]	7.1	26.4	46.7	-	-
RT-DETR-R18 [71]	20.0	60.0	46.5	4.58	4.49
YOLOv10-S (Ours)	**7.2**	**21.6**	**46.3 / 46.8**†	**2.49**	**2.39**
YOLOv6-3.0-M [27]	34.9	85.8	49.1	5.63	4.56
Gold-YOLO-M [54]	41.3	87.5	49.8	6.38	5.45
YOLO-MS [7]	22.2	80.2	51.0	12.41	7.30
YOLOv8-M [20]	25.9	78.9	50.6	9.50	5.09
YOLOv9-M [59]	20.0	76.3	51.1	-	-
RT-DETR-R34 [71]	31.0	92.0	48.9	6.32	6.21
RT-DETR-R50m [71]	36.0	100.0	51.3	6.90	6.84
YOLOv10-M (Ours)	**15.4**	**59.1**	**51.1 / 51.3**†	**4.74**	**4.63**
YOLOv6-3.0-L [27]	59.6	150.7	51.8	9.02	7.90
Gold-YOLO-L [54]	75.1	151.7	51.8	10.65	9.78
YOLOv9-C [59]	25.3	102.1	52.5	10.57	6.13
YOLOv10-B (Ours)	**19.1**	**92.0**	**52.5 / 52.7**†	**5.74**	**5.67**
YOLOv8-L [20]	43.7	165.2	52.9	12.39	8.06
RT-DETR-R50 [71]	42.0	136.0	53.1	9.20	9.07
YOLOv10-L (Ours)	**24.4**	**120.3**	**53.2 / 53.4**†	**7.28**	**7.21**
YOLOv8-X [20]	68.2	257.8	53.9	16.86	12.83
RT-DETR-R101 [71]	76.0	259.0	54.3	13.71	13.58
YOLOv10-X (Ours)	**29.5**	**160.4**	**54.4 / 54.4**†	**10.70**	**10.60**

NANO (very lightweight)

Small (lightweight)

Medium

Base

Large

Extra large

Diagram: YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions, Nidhal Jegham1,2, Chan Young Koh 2 Marwan Abdelatti2,3, and Abdeltawab Hendawi2 1 Tunis Business School, University of Tunis. (https://arxiv.org/html/2411.00201v3)

HSLU Hochschule Luzern

ultralytics

Hands-on:

(1) Exploring models – inferencing

HSLU Hochschule Luzern

ultralytics

Hands-on:

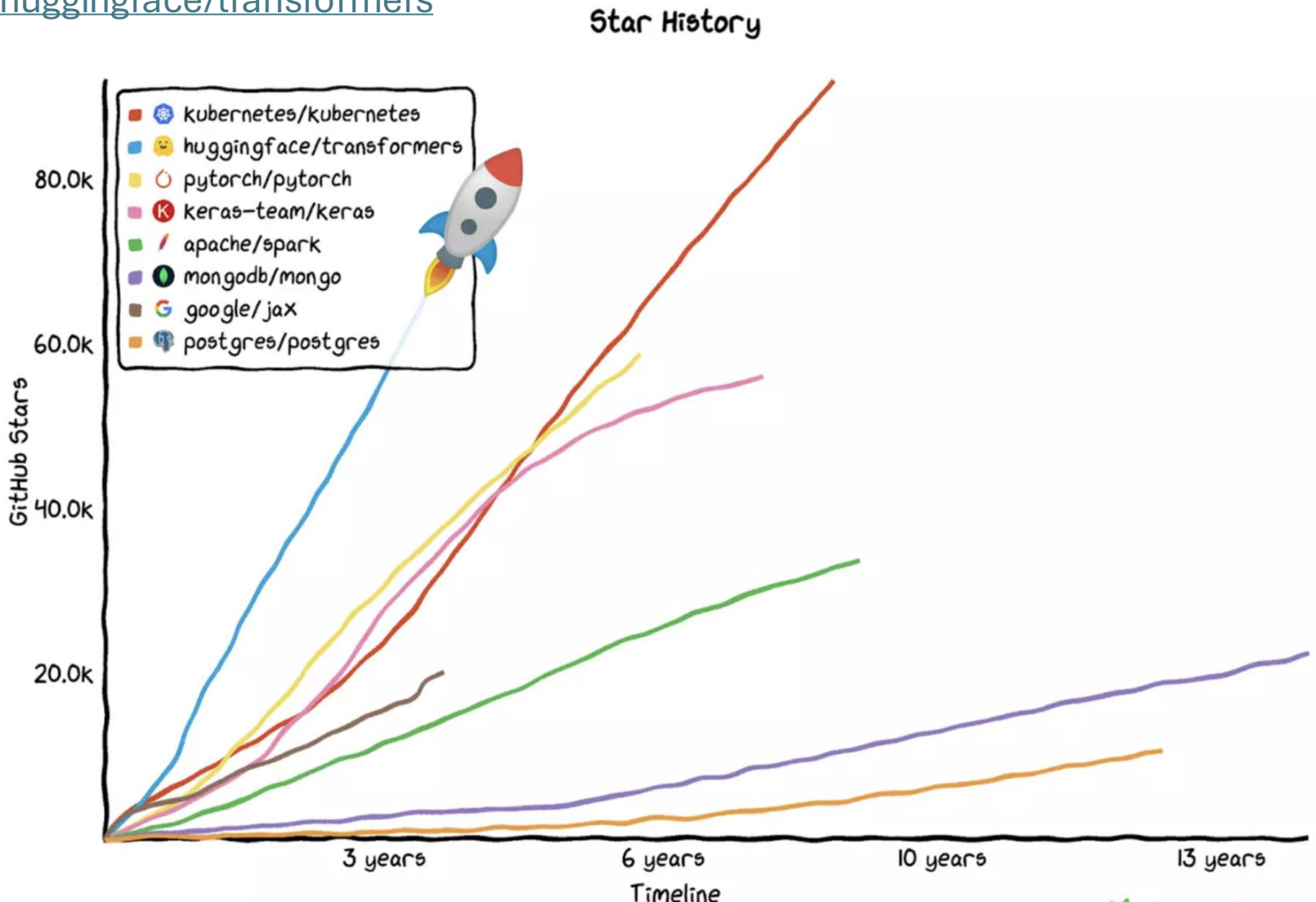
(2) Finetuning object detection model on custom dataset

Hugging Face Transformers library

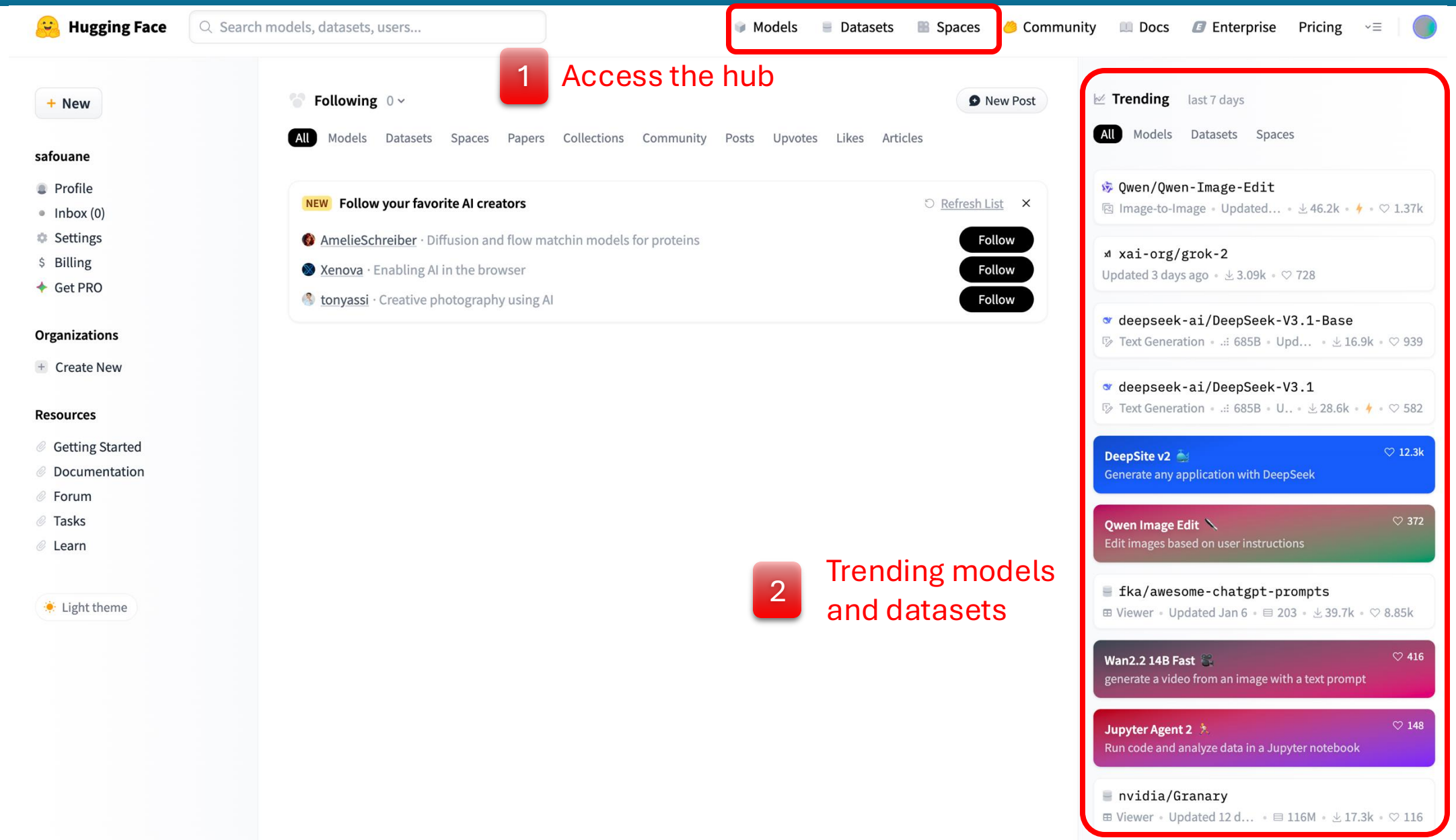
Transformers library

Transformers is one of the fastest growing libraries for open source projects

https://github.com/huggingface/transformers



Transformers library



HSLU Hochschule Luzern

Limitations of ViTs

Limitations of ViTs

Some researchers have noticed that:

- **High Data Requirements**: ViTs are highly data-hungry and usually need massive datasets for effective training.

- **Computational and Memory Intensity**: The self-attention mechanism in ViTs is computationally expensive and memory-intensive, particularly for high-resolution images, leading to higher processing demands and potential scalability issues.

- **Challenges with Local Textures and High-Frequency Details**: ViTs often struggle to capture high-frequency components or local textures in images, where CNNs excel due to their convolutional operations.

- **Prone to Overfitting on Small Datasets**: Without sufficient data, ViTs are susceptible to overfitting, limiting their applicability in scenarios with limited training samples.

AGENDA. DAY 2

- Background & Motivations
- Applications: image captioning, retrieval, grounding, generative tasks
- CLIP and Zero-Shot Learning
- Hands-on:
  - Inference CLIP for zero-shot classification
  - GroundingDINO for zero-shot detection
  - Prompt Engineering
- BLIP and Multimodal Generation
- Diffusion Model – introduction to Markov Chain
- Hands-on:
  - Stable-diffusion
  - Prompt engineering

HSLU Hochschule Luzern

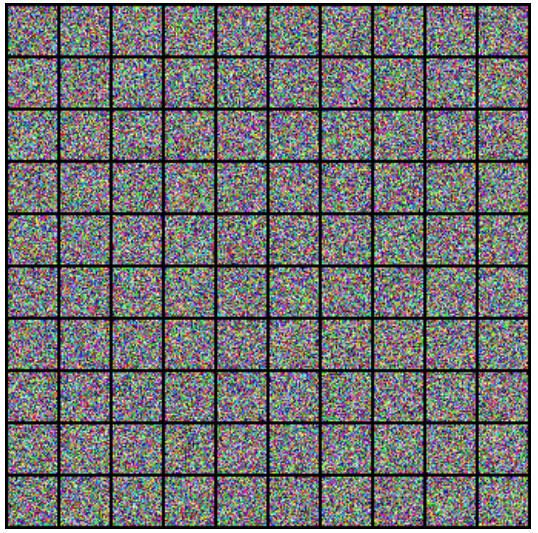
Background & Motivations

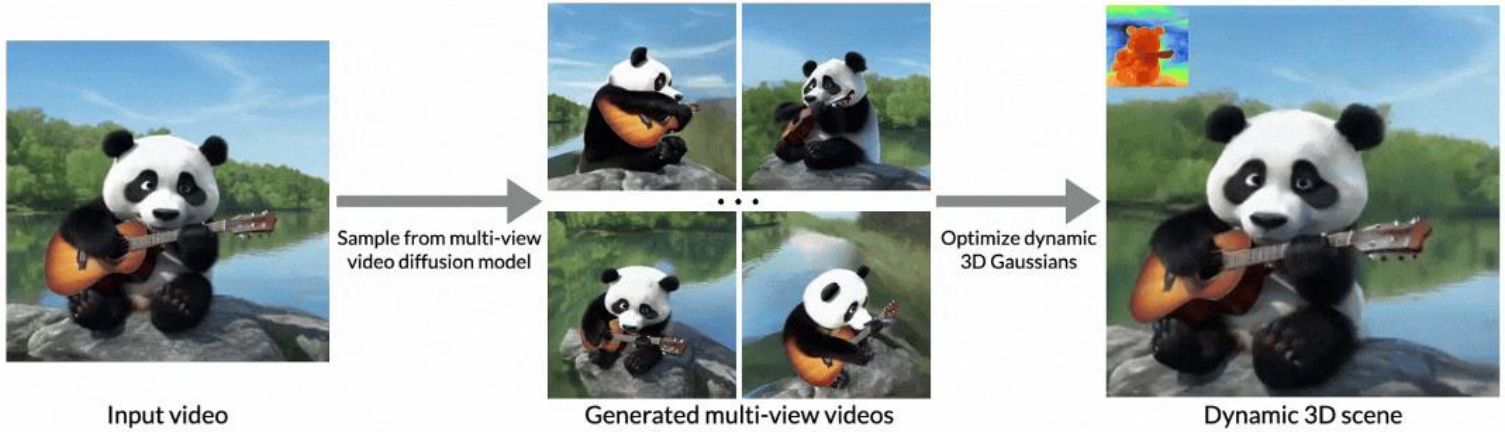
Background & Motivations

Introduction to Generative AI Fundamentals: Previous generative AI models rely on GAN models. Attention-Generative AI as models that create new data (e.g., images, text) from learned patterns in training data, contrasting it with discriminative models.

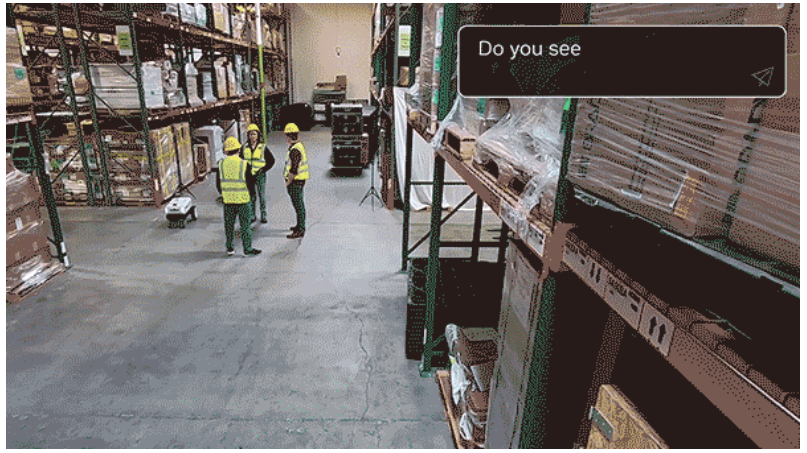
Motivations: solving data scarcity (e.g., synthetic data for training), creativity (art, design), personalization (e.g., custom content), and efficiency (automating tasks like content creation). Highlight real-world drivers like computational power growth, large datasets (e.g., LAION-5B), and ethical considerations (e.g., bias in generated outputs).

Synthetic image creation





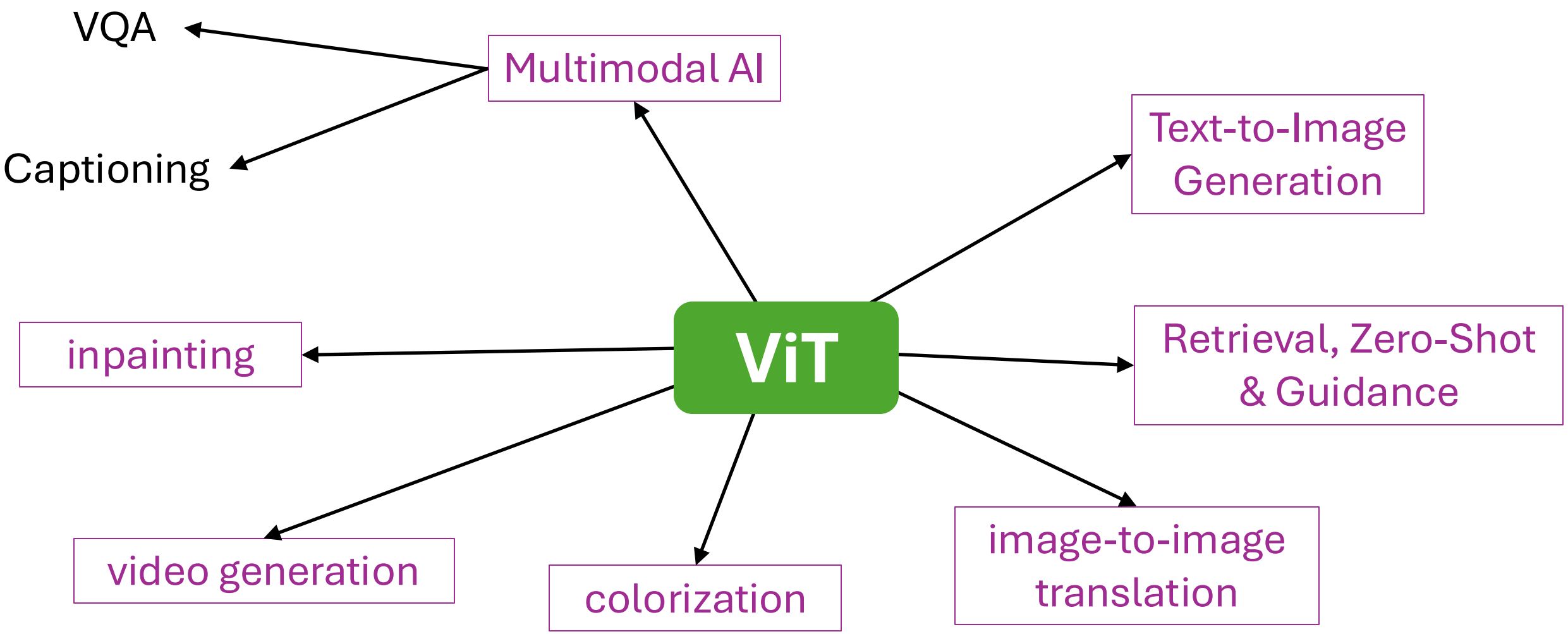
Captioning/VQA



HSLU Hochschule Luzern

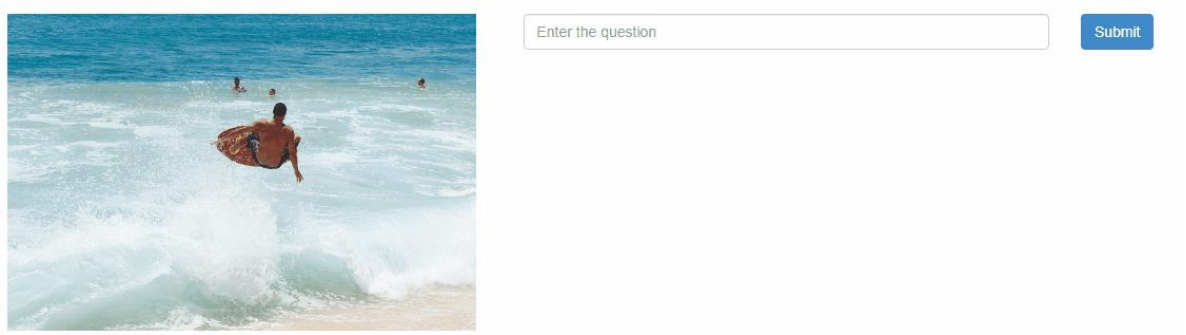
Applications of Vision Transformers in Gen AI

Applications in Generative AI



HSLU Hochschule Luzern Applications in Generative AI

Multimodal AI: VQA



Text-to-Image Generation

inpainting

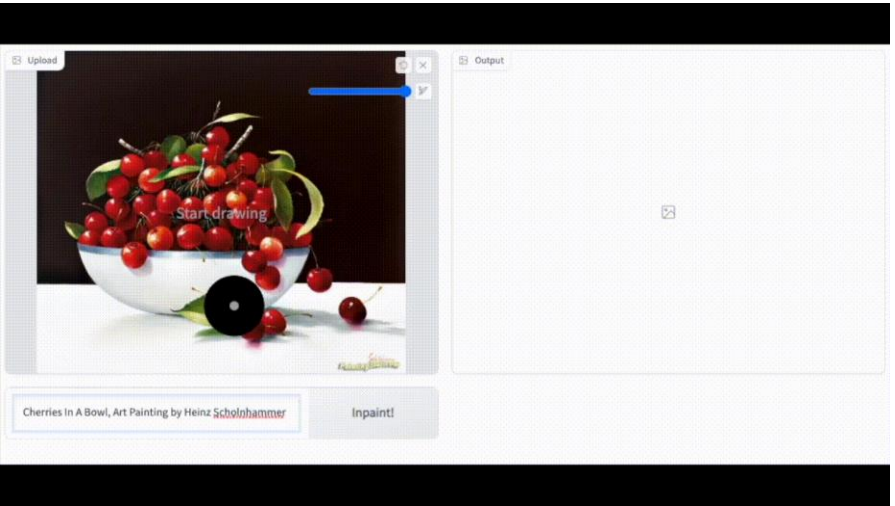
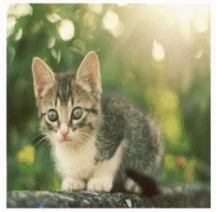


image-to-image translation

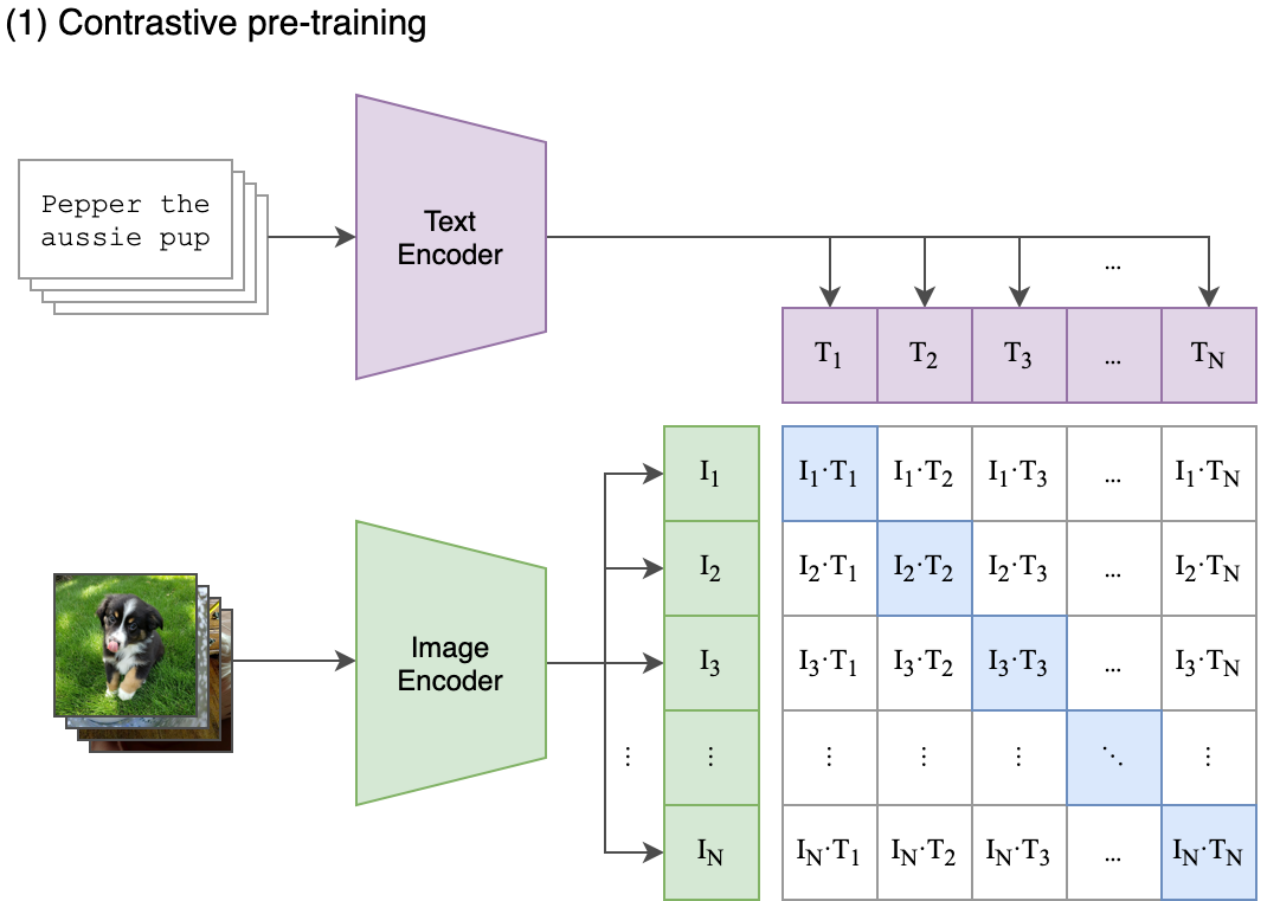


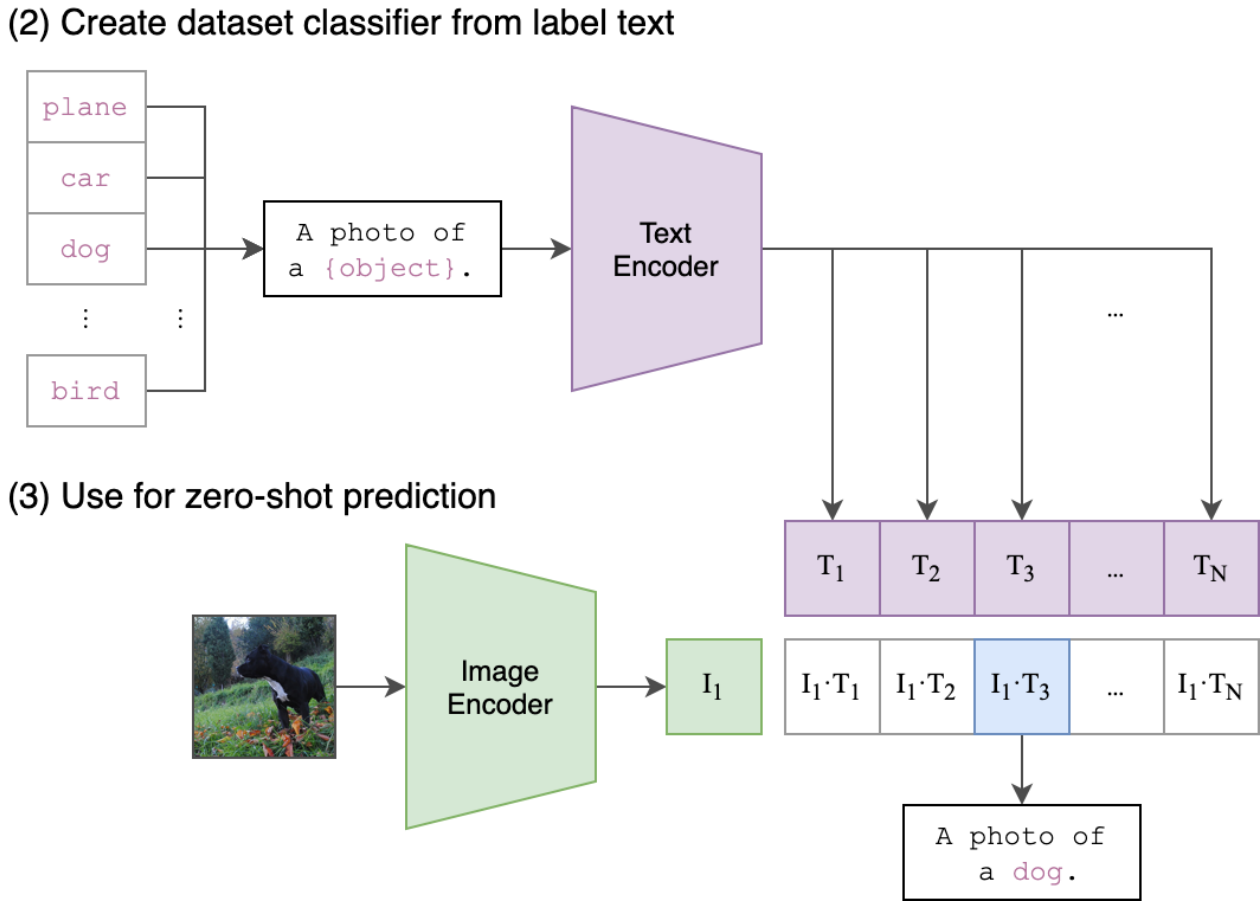
HSLU Hochschule Luzern

CLIP and zero-shot Learning

CLIP and zero-shot Learning

CLIP: Learning Transferable Visual Models From Natural Language Supervision (Paper)

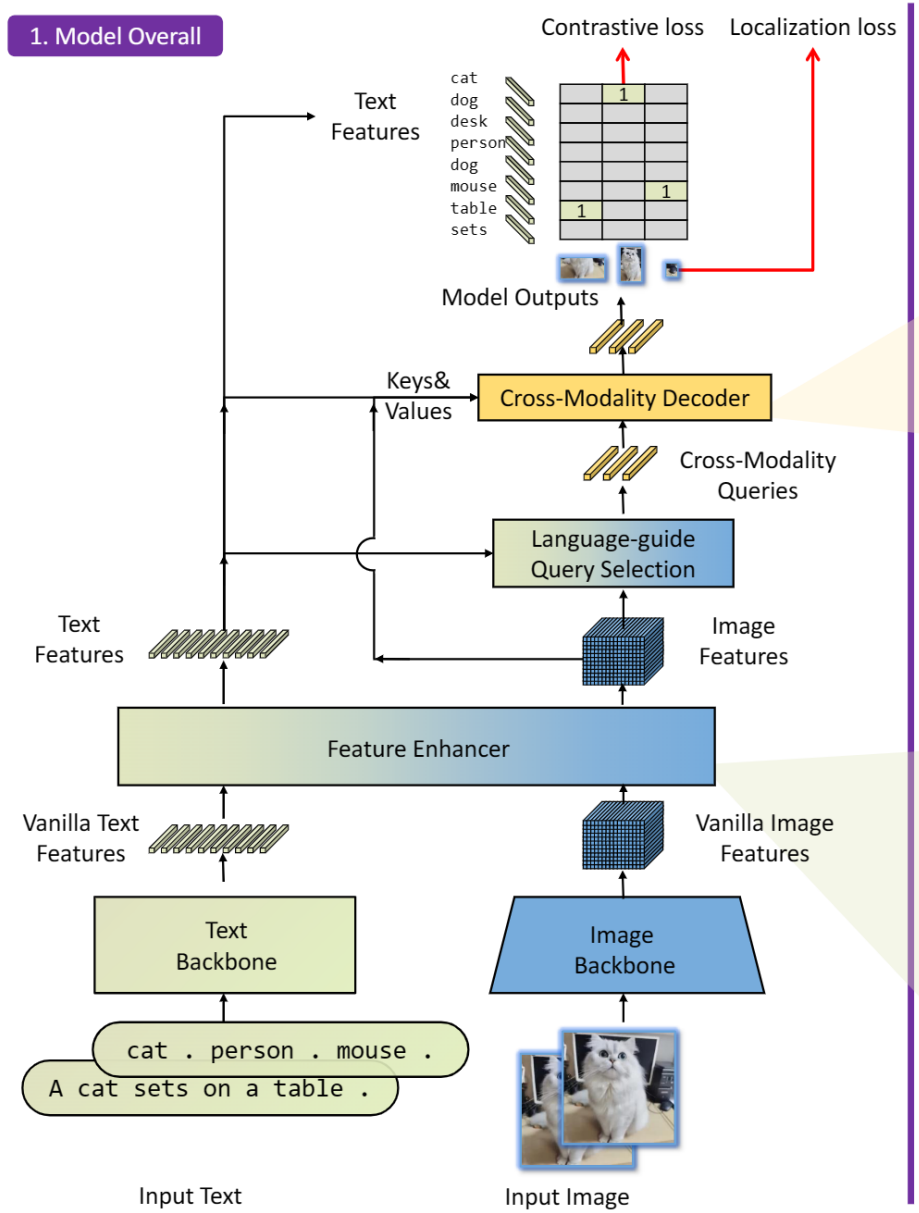


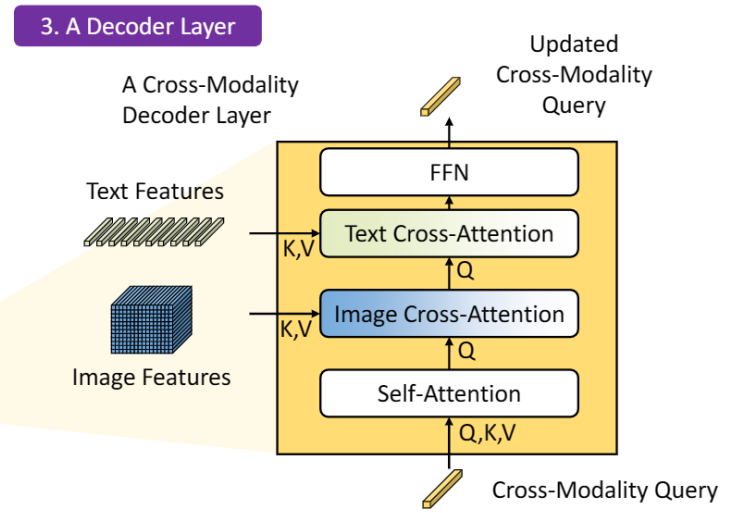


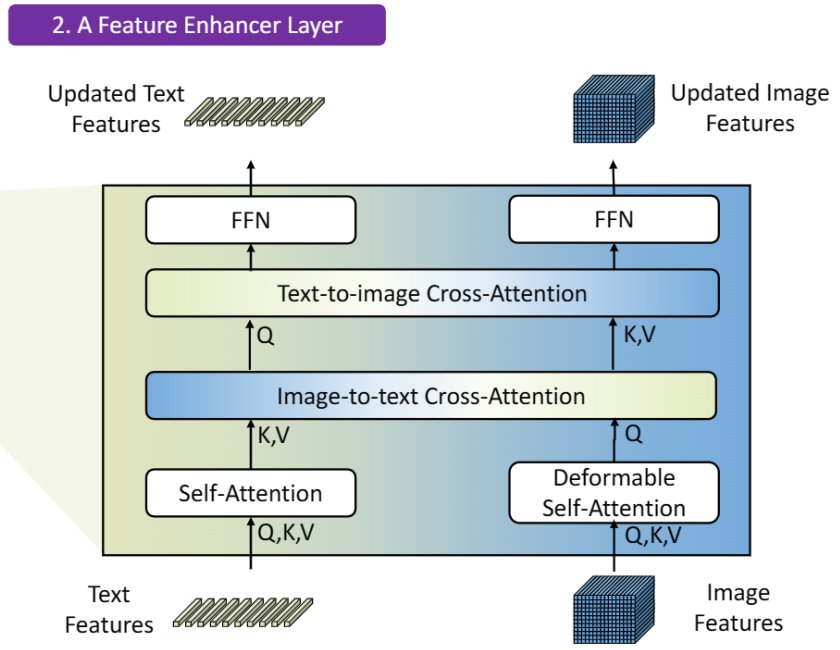
HSLU Hochschule Luzern

CLIP and zero-shot Learning

GroundingDINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection ([Paper])







HSLU Hochschule Luzern

Hugging Face

Hands-on:

(1) CLIP for zero-shot classification

OpenAI

HSLU Hochschule Luzern

Hugging Face

Hands-on:

(2) GroundingDINO for zero-shot detection

OpenAI

HSLU Hochschule Luzern

Hugging Face

Hands-on:

(3) Prompt Engineering

OpenAI

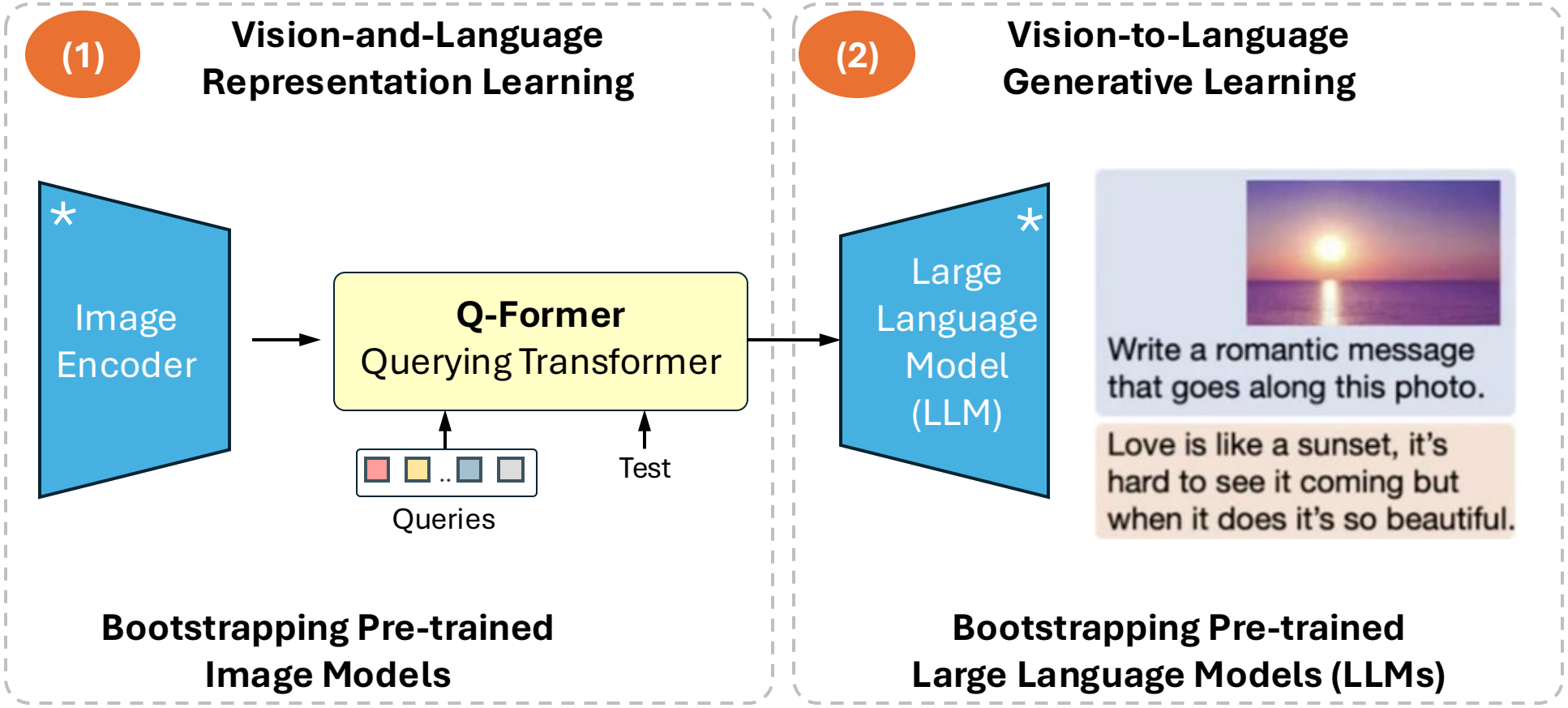
HSLU Hochschule Luzern

BLIP and multimodal generation

BLIP and Multimodal generation

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models (Paper)

Pre-train a light weight Q-Former in two stages:



HSLU Hochschule Luzern BLIP and Multimodal generation

	Concept	Paper
BLIP	A vision-language pretraining framework that uses a bootstrapped approach to generate and filter captions from noisy web data, enabling unified vision-language understanding and generation tasks like image captioning. It combines a vision encoder and text decoder for efficient captioning.	BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation
IDEFICS	A multimodal model designed for vision and language tasks, including image captioning, built by fine-tuning open-access models like LLaVA and CLIP-ViT. It processes images and text jointly, leveraging a large-scale pretraining dataset for robust performance.	IDEFICS: An Open-Source Visual Language Model
GIT	A generative image-to-text transformer that excels in image captioning by directly generating text from image inputs. It uses a single vision transformer and a text decoder, optimized for lightweight and efficient captioning with performance comparable to BLIP-2.	GIT: A Generative Image-to-text Transformer for Vision and Language
InstructBLIP	An enhanced version of BLIP-2, fine-tuned with instruction-tuning datasets to improve zero-shot and few-shot performance in image captioning and other vision-language tasks. It leverages BLIP-2's architecture with additional task-specific optimizations.	InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning
COCA	A contrastive captioning model that combines contrastive learning with generative captioning. It uses image-text alignment to produce high-quality captions, offering performance close to BLIP-2 with lower computational requirements.	CoCa: Contrastive Captioners are Image-Text Foundation Models

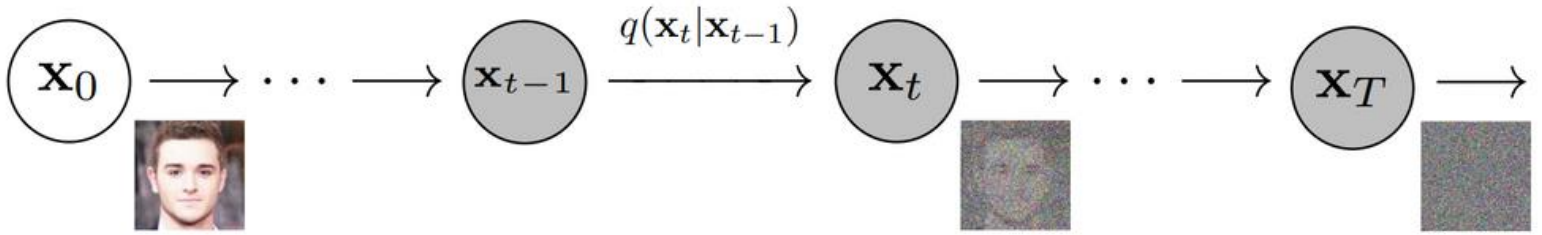
HSLU Hochschule Luzern

Diffusion Models

HSLU Hochschule Luzern

Diffusion Models

Diffusion Models work by destroying training data through the successive addition of Gaussian noise, and then learning to recover the data by reversing this noising process.



Forward diffusion process

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$$

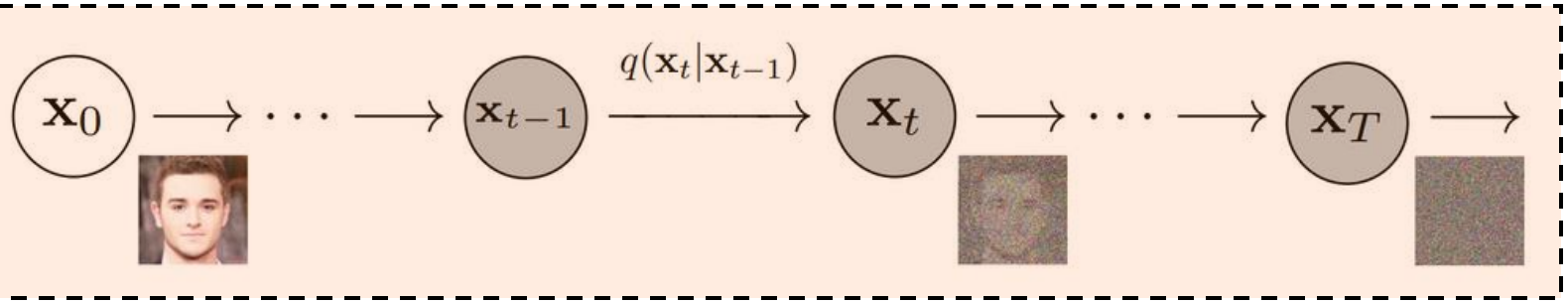
distribution of the noised images | Output | Mean μ_t | Variance Σ_t

- ***t***: This represents the **time step**. It goes from a clear image at t=0 to a completely noisy image at t=T.
- ***x_0***: This is your **original, clean data sample**. It's the starting point of the whole process.
- ***β_t***: This is the **variance schedule**. It's a list of small numbers that control how much noise is added at each step. (1) It's a fixed sequence (not learned). (2) it typically starts with a very small number ($\beta_0 \approx 0$) so the first few steps add very little noise. (3) It gradually increases to a larger number ($\beta_t \approx 1$), so the later steps add a lot of noise. (4) The values of βt are always between 0 and 1.
- **I**: This is the **identity matrix**. It ensures that the noise is added independently to each pixel of the image, without any correlation between pixels.

Diffusion Models

Diffusion Models work by destroying training data through the successive addition of Gaussian noise, and then learning to recover the data by reversing this noising process.

Markov chain



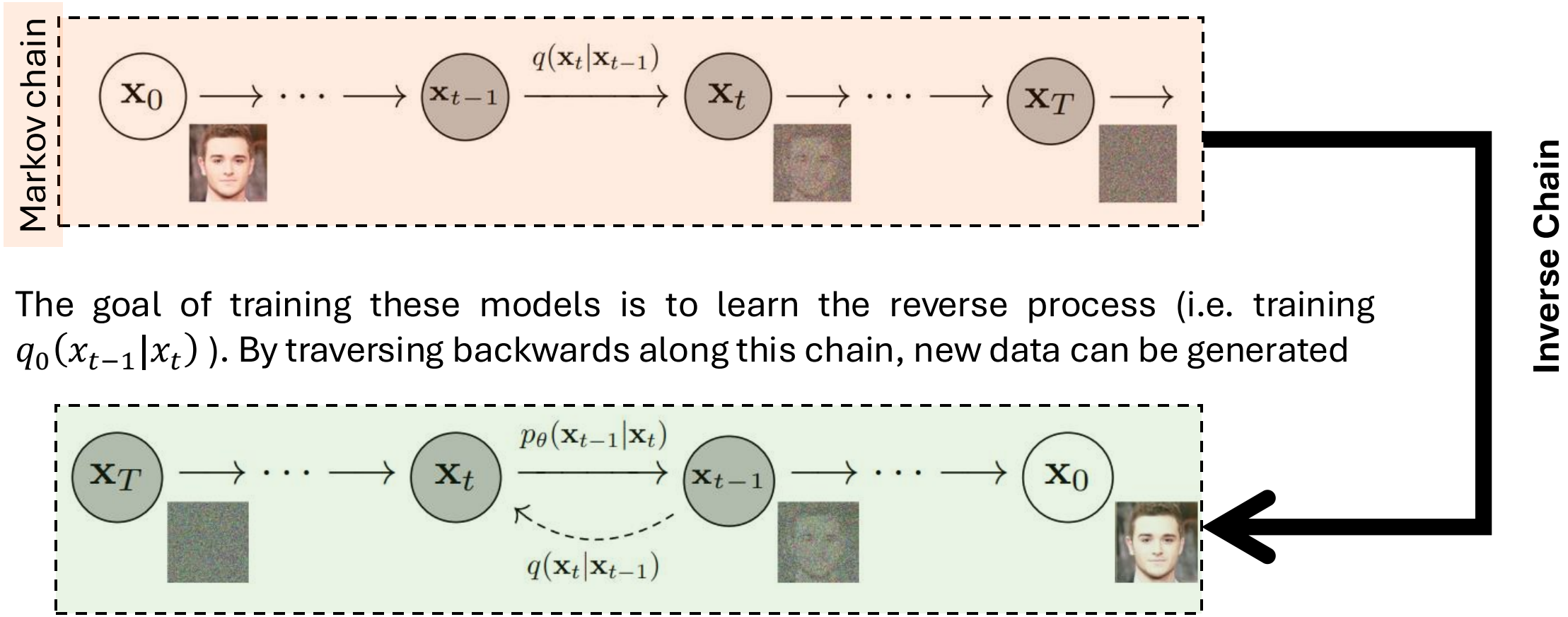
Ultimately, the image is asymptotically transformed to pure Gaussian noise.

The goal of training these models is to learn the reverse process (i.e. training $q_0(x_{t-1}|x_t)$). By traversing backwards along this chain, new data can be generated

Diffusion Models:A Comprehensive Survey of Methods and Applications. LING YANG *et. al. Feb. 2024* arXiv:2209.00796

Diffusion Models

Diffusion Models work by destroying training data through the successive addition of Gaussian noise, and then learning to recover the data by reversing this noising process.



The goal of training these models is to learn the reverse process (i.e. training $q_0(x_{t-1}|x_t)$). By traversing backwards along this chain, new data can be generated

Diffusion Models:A Comprehensive Survey of Methods and Applications. LING YANG *et. al. Feb. 2024* arXiv:2209.00796

HSLU Hochschule Luzern

Hugging Face

Hands-on:

(1) BLIP image captioning

OpenAI

HSLU Hochschule Luzern

Hugging Face

Hands-on:

(2) Stable diffusion

OpenAI

Computer Vision – Fundamentals

Introduction To Tranformers In Vision

Program directors:

Dr. Prof.
Umberto Michelucci

Dr.
Aygul Zagidullina

Dr.
Safouane El Ghazouali

HSLU Lucerne University of Applied Sciences and Arts

BOOTCAMP AI-VISION

Starting date: **25 August 2026**

DAY 1

- …

DAY 2

- …

DAY 3

- Introduction to transformers in vision
- Exploring ViT architecture
- Other vision tranformers alternatives
- Transformers in object detection
- Transformers library for ViT application

DAY 4

- Use of transformers if vision generative AI
- Fields of applications
- Zero-shot learning for classification and detection tasks
- Multimodal AI models
- Image Diffusion Models

DAY 5

Projects realisation