Deep Learning for Structured Outputs

A presentation by Tanishq Sardana, Qing Mu, Owais Shuja

Segment Anything

Deep Learning

Deep learning is a subset of machine learning that uses artificial neural networks with many layers to model complex patterns in data.

- **Neural Networks:** Inspired by the structure and function of the brain, neural networks consist of interconnected nodes (neurons) that process information.
- **Layers:** Deep learning models have multiple layers, including input, hidden, and output layers. Each layer learns different levels of abstraction from the data.
- **Training:** Deep learning models are trained using large datasets and optimization algorithms like gradient descent to minimize error.

Its ability to learn hierarchical representations of data makes it particularly effective for complex tasks.

Structured Outputs

Structured outputs refer to predictions that have a complex structure, such as sequences, trees, or graphs, rather than simple scalar values.

Examples:

- **Sequence Labeling:** Assigning a label to each element in a sequence (e.g., part-of-speech tagging in NLP).
- **Parsing:** Analyzing a sentence to identify its grammatical structure (e.g., syntactic parsing).
- **Image Segmentation:** Dividing an image into segments or regions based on pixel characteristics.

Importance:

- Many real-world problems involve structured outputs, where the goal is to predict a complex structure rather than a single value.
- Traditional machine learning methods often struggle with capturing the dependencies and constraints in structured outputs.

Deep Learning for Structured Outputs

- Deep learning models can capture intricate dependencies and structures in the data, making them well-suited for structured output prediction.
- The flexibility of deep learning architectures allows for tailored solutions to specific structured output problems.

Examples:

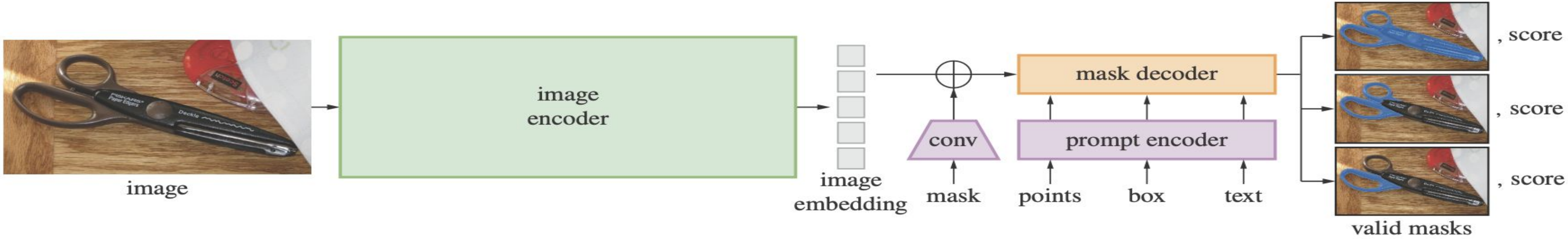
- **RNNs:** Designed to process sequential data by maintaining a hidden state that captures information from previous time steps.
- **LSTMs:** A type of RNN that addresses the vanishing gradient problem, allowing for better learning of long-term dependencies.
- **CNNs**: CNNs apply convolutional filters to capture spatial hierarchies and patterns and are effective for tasks like image segmentation, where the output is a structured grid of labels.

Segment Anything Model (SAM)

The Segment Anything Model (SAM) is a cutting-edge deep learning model developed by Meta for image segmentation tasks. It aims to segment any object within an image based on user prompts, such as points, boxes, or masks.

- **Purpose:** SAM is designed to be a foundational model for image segmentation, capable of generalizing to a wide variety of segmentation tasks without task-specific training.

In the context of SAM, the structured output is the segmentation mask, which delineates the boundaries of objects within an image. This mask is a structured grid where each pixel is labeled as part of an object or the background.

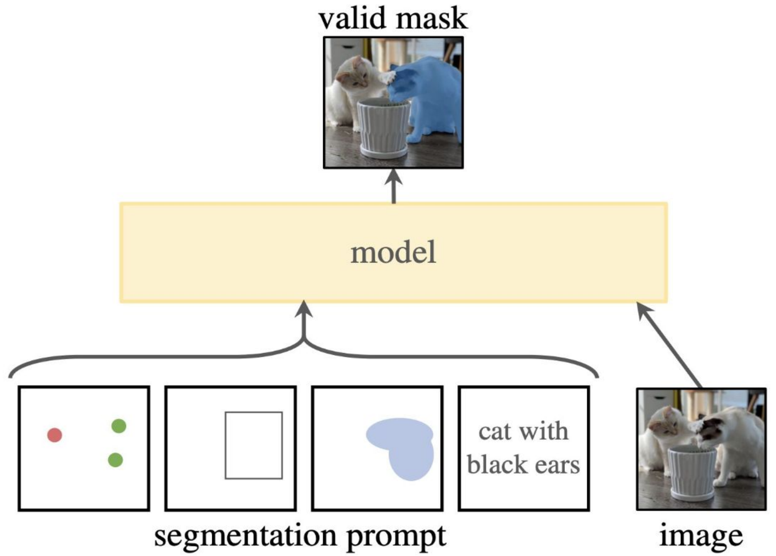


3 motivations behind implementing SAM

1. What task will enable zero-shot generalization?
2. What is the corresponding model architecture?
3. What data can power this task and model?

The Task

Given any segmentation prompt specifies what to segment in an image, the goal is to return a valid segmentation mask.



(a) **Task**: promptable segmentation

Why Promptable Segmentation

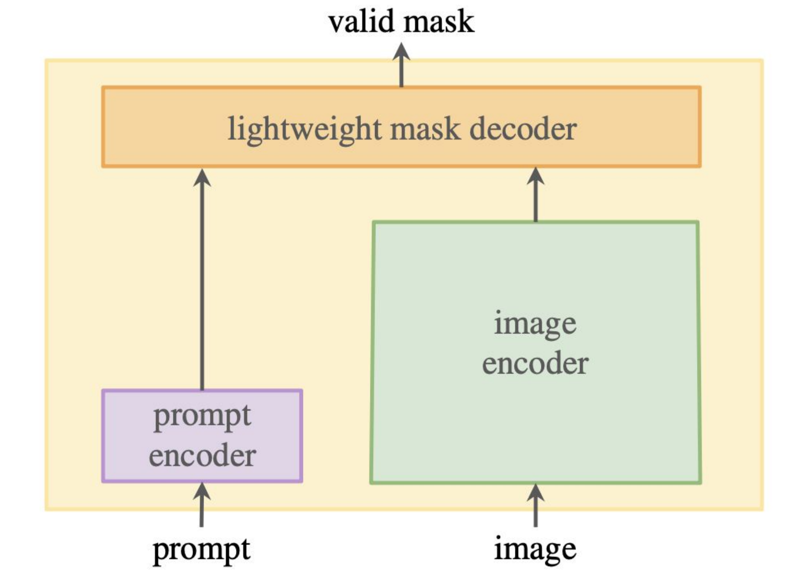
1. Inspired by **prompting in NLP** (e.g., GPT models respond to text prompts). SAM can take various types of user inputs (prompts) to guide the segmentation process.
2. This flexibility allows for interactive and customizable segmentation, making it adaptable to different applications.
3. Enables **zero-shot learning** – SAM can segment objects it has never seen before. SAM is trained on a diverse dataset of images and prompts, enabling it to generalize to new, unseen objects and scenarios without additional training.
4. This showcases the power of deep learning in capturing universal patterns in data.
5. Makes segmentation **interactive & adaptable** (works with different types of prompts)

Types of Prompts Used

1. **Point-based** - Click a point, and SAM segments the relevant object.
2. **Box-based** - Draw a bounding box, and SAM refines it into a segmentation mask.
3. **Mask-based** - Give a rough mask, and SAM improves it.
4. **Text-based** - Describe an object, and SAM segments it.

The Model

“a powerful image encoder computes an image embedding, a prompt encoder embeds prompts, and then the two information sources are combined in a lightweight mask decoder that predicts segmentation masks. We refer to this model as the Segment Anything Model, or SAM”



(b) **Model**: Segment Anything Model (**SAM**)

Image Encoder

- A **Vision Transformer (ViT)** pre-trained using **Masked Autoencoder (MAE)** techniques is used
- It converts an image into a **high-dimensional embedding** before any segmentation prompt is applied
- Runs **once per image**, making it computationally efficient when multiple prompts are applied

Prompt Encoder

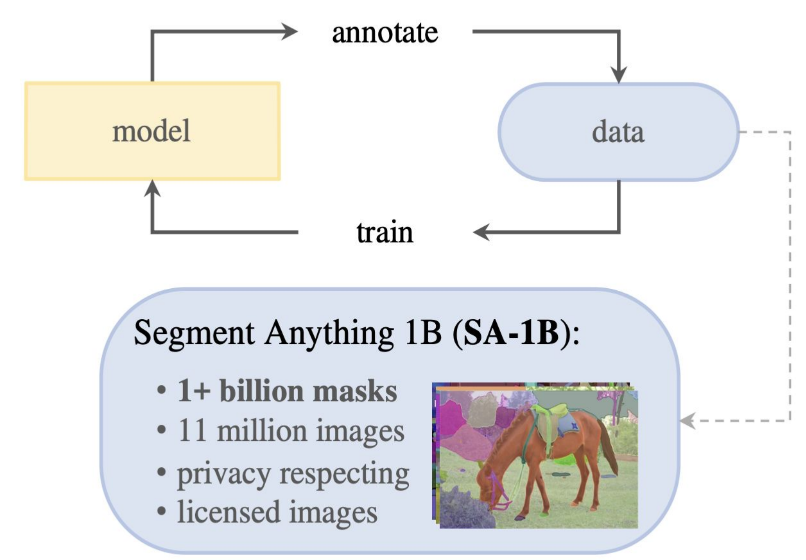
- Converts input prompts (points, boxes, masks, or text) into an **embedding representation**.
- **Sparse prompts** (points, boxes) are encoded using **learned embeddings**.
- **Dense prompts** (masks) are processed using a **convolutional network**.
- **Text prompts** use a **CLIP-based text encoder**.

Mask Decoder

- Merges **image embeddings and prompt embeddings** to predict segmentation masks.
- Uses a **Transformer-based decoder** that refines the segmentation mask iteratively.
- **Ambiguity-aware**: If a prompt is unclear (e.g., a point on a t-shirt), SAM predicts **multiple possible masks** and ranks them by confidence.
- Outputs segmentation masks in **real-time (~50ms per mask on a web browser!)**.

The Data Engine

To achieve strong generalization to new data distributions, it is necessary to train SAM on a large and diverse set of masks, beyond any segmentation dataset that already exists.



(c) **Data**: data engine (top) & dataset (bottom)

The Data Engine - SA-1B Dataset

<u>**Assisted Manual Annotation**</u>	<u>**Semi-Automatic Annotation**</u>	<u>**Fully Automatic Stage**</u>
Human annotators segment objects **with SAM's help**	SAM **pre-segments objects**; humans refine missing details	SAM **predicts masks completely automatically**.
Model suggestions speed up **mask creation**.	**Mix of automated and manual segmentation**.	Uses a **grid-based point prompting strategy** to find objects.
Collected 4.3M masks in this phase	**10.2M masks collected** in total at this stage	**Generated 1.1B high-quality masks**

Why is SA-1B a game changer?

- **400x more masks** than any previous dataset.
- Covers **diverse objects & scenarios** → Improves generalization.
- Enables **strong zero-shot performance** on new segmentation tasks.

SAM in the Structured Output Domain

Traditional methods struggle with capturing the complex dependencies and variations in object shapes and sizes, making deep learning approaches like SAM particularly valuable.

SAM employs a transformer-based architecture, which is adept at understanding global context and long-range dependencies in images. This capability is crucial for producing coherent and accurate segmentation masks.

SAM is designed to generalize across different segmentation tasks without task-specific training. This zero-shot capability highlights the model's robustness in handling diverse structured outputs.

SAM can be applied to a wide range of applications, from medical imaging to robotics, demonstrating the broad applicability of deep learning in structured output prediction.

The model's training on a large and diverse dataset enables it to transfer knowledge to new, unseen tasks, showcasing the power of deep learning in capturing universal patterns in structured data.

Evaluation and Results

SAM is tested on **23 segmentation datasets** across **five tasks** to assess its generalization:

Task	Description	Metric Used
Zero-shot Segmentation	Segment an Object from a single point	Mean IoU (mIoU)
Edge Detection	Identify Object Boundaries	ODS (F1 score), OIS (Optimized F1)
Object Proposal Segmentation	Find all objects in an image	Average Recall (AR@1000)
Instance Segmentation	Segment objects detected by a bounding box model	Mask AP (Average Precision)
Text-to-mask Segmentation	Generate masks from text prompts	Qualitative Evaluation

Zero Shot Segmentation

Goal: Predict a mask from a **single foreground point**
Results (on 23 datasets, compared to RITM, a strong interactive segmentation model):

- **SAM outperforms RITM on 16/23 datasets**
- Example dataset performance (Mean IoU at 1-point prompt):
  - **ADE20K:** SAM +9.1 over RITM
  - **LVIS:** SAM +18.5 over RITM
  - **STREETS:** SAM +41.1 over RITM

- **Human Study:** SAM's segmentation **rated higher than RITM** (Avg. score: **8/10** vs. **6/10** for RITM)

Edge Detection

Goal: Detect boundaries between objects
Dataset: BSDS500 (classic edge detection benchmark)

SAM performs significantly better than traditional methods and approaches **deep learning-based edge detectors** without explicit training on edges.

Model	ODS (F1 score)	OIS (Optimized F1)
Sobel	0.539	-
Canny	0.600	0.640
SAM (zero-shot)	0.768	0.786
HED (deep learning)	0.788	0.808
EDETR	0.840	0.858

Object Proposal Generation

Goal: Predict object masks without labeled bounding boxes

Dataset: LVIS v1 (large-scale instance segmentation benchmark)

Metric: AR@1000 (higher is better)

SAM is competitive with ViTDet-H, especially for **medium and large objects**, despite being **zero-shot**.

Model	AR@1000	Small Objects	Medium Objects	Large Objects
ViTDet-H (fully supervised)	63.0	51.7	80.8	87.0
SAM (zero-shot)	59.3	45.5	81.6	86.9

Instance Segmentation

Goal: Segment objects detected by a bounding box model

Dataset: COCO & LVIS

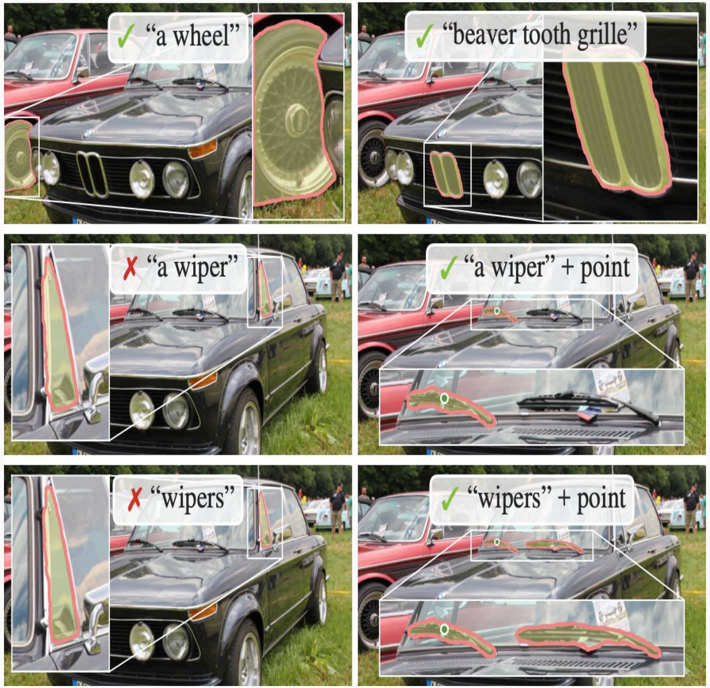
Metric: Mask AP (higher is better)

Human study: Annotators **preferred SAM's masks** over ViTDet's, despite ViTDet scoring higher on COCO AP.

Model	COCO AP	LVIS AP
ViTDet-H (fully supervised)	51.0	46.6
SAM (zero-shot)	46.5	44.7

Text to Mask Segmentation

SAM can segment objects based on simple text prompts like “a wheel” as well as phrases like “beaver tooth grille”. When SAM fails to pick the right object from a text prompt only, an additional point often fixes the prediction,



Applications of SAM

1. **Medical Imaging: Tumor Detection & Organ Segmentation**
2. **Autonomous Vehicles: Object Detection & Environment Perception**
    - **Object Detection:** Segment and identify objects on the road, such as vehicles, pedestrians, and traffic signs, to enhance navigation and safety.
    - **Environment Perception:** Understand the surrounding environment by segmenting different terrain types and obstacles.
3. **Robotics:**
    - **Object Manipulation:** Enable robots to identify and interact with specific objects in their environment by providing precise segmentation masks.
    - **Navigation:** Improve robotic navigation by segmenting pathways and obstacles in real-time.
4. **Augmented Reality (AR): Environment Mapping and Interactive Applications**
5. **Satellite Imagery: Land Use Analysis and Disaster Management**

Limitations

- While SAM performs well in general, it is not perfect.
- It can miss delicate structures, hallucinate small disconnected components at times, and does not produce boundaries as well as more computationally intensive methods.
- Dedicated interactive segmentation methods generally outperform SAM when many points are provided.
- SAM is designed for generality rather than high IoU interactive segmentation.

Future Works

1. **Enhancing Real-Time Performance:**
   - **Optimization Techniques:** Develop optimization techniques to reduce the computational overhead of SAM, enabling real-time segmentation on resource-constrained devices.
   - **Edge Computing:** Implement SAM on edge devices for real-time applications in autonomous vehicles, robotics, and mobile devices.
2. **Integration with Multi-Modal Systems:**
   - **Vision-Language Models:** Combine SAM with vision-language models to improve tasks like visual question answering and image captioning, where understanding both visual and textual information is crucial.
   - **Cross-Modal Learning:** Explore the integration of SAM with other sensory data, such as audio or tactile information, for more comprehensive scene understanding.
3. **Domain-Specific Fine-Tuning:**
   - **Medical Imaging:** Fine-tune SAM for specific medical imaging tasks, such as segmenting rare diseases or anomalies, to improve diagnostic accuracy.
   - **Industrial Inspection:** Adapt SAM for quality control in manufacturing, where precise segmentation of defects is essential.

Conclusion

- The **Segment Anything Model (SAM)** revolutionizes image segmentation by enabling **zero-shot generalization** across diverse tasks.
- SAM's **promptable segmentation** approach allows for **interactive and flexible** segmentation through various input types.
- The **SA-1B dataset**, with **1.1B high-quality masks**, significantly enhances model generalization and performance.
- SAM outperforms traditional and deep learning-based segmentation models on multiple benchmarks.
- While SAM has **limitations**, ongoing research aims to improve **real-time performance, multi-modal integration, and domain-specific fine-tuning**.
- The project **paves the way for foundation models in segmentation**, expanding applications across **medical imaging, robotics, AR, and beyond**.

🔗 **Try the Demo**: https://segment-anything.com

Understanding “DETR: End-to-End Object Detection with Transformers”

Qing Mu

Mar.5.2025

Main Problem

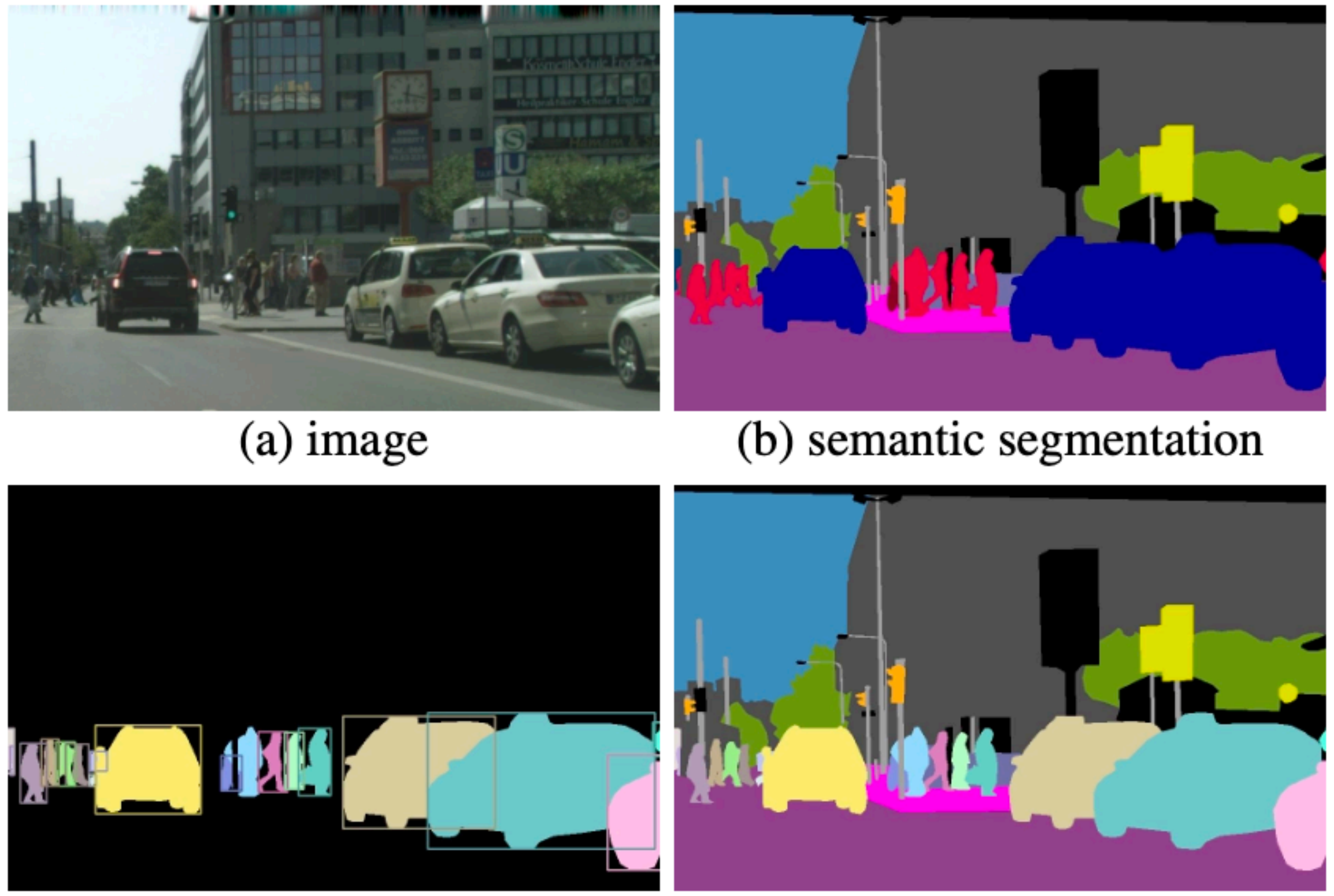
- Robot vision use object detection to get object information in the environment
- Majority of the object detection models today use hand-designed components
  - Encode prior knowledge about the object detection
- Prior end-to-end object detection works
  - Complex combination of hand-crafted components
  - Requiring manually adjustment (e.g., anchor size and NMS threshold) for specific datasets
  - Were not as competitive in results

Motivation

- DETR
  - Try out transformer architecture for object detection
  - Transformer can predict multiple object in parallel
- Bipartite Matching
  - Unique matching
    - Invariant to permutations of predicted objects
    - No more autoregressive decoding to avoid duplicates
  - Bypass the need for NMS or anchors

Problem Setting

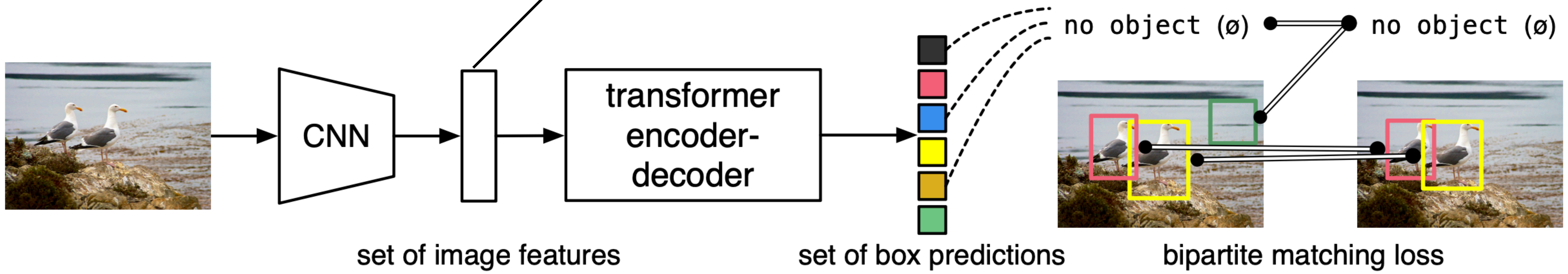
- **Object detection:**
  - For each object in the image:
    - Identify the bounding box of the object in the image
    - Classify the object
- **Panoptic Segmentation:**
  - Given a set of L semantic classes encoded by $S := \{0, \ldots, L-1\}$, for each pixel i of an image:
    - Identify l_i of the pixel, where $l_i \in S$ is the semantic class of pixel i
    - Identify z_i of the pixel, where z_i represents the pixel's instance id
      - Groups pixel of the same class into distinct segments



(a) image (b) semantic segmentation (c) instance segmentation (d) panoptic segmentation

DETR Overview

a d-dim cone channel



- DETR predicts in parallel the final set of detections.
- The CNN is used for feature learning, the Transformer is used to make predictions.
- During training, bipartite matching uniquely assigns predictions with ground truth boxes.
- No need for the annoying NMS

DETR - the loss function

It also contain empty ground-truth

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{\mathrm{match}}(y_i, \hat{y}_{\sigma(i)}),$$

Loss for the bounding-box

$$-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\mathrm{box}}(b_i, \hat{b}_{\sigma(i)})$$

Loss for the class

$$\mathcal{L}_{\mathrm{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\mathrm{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

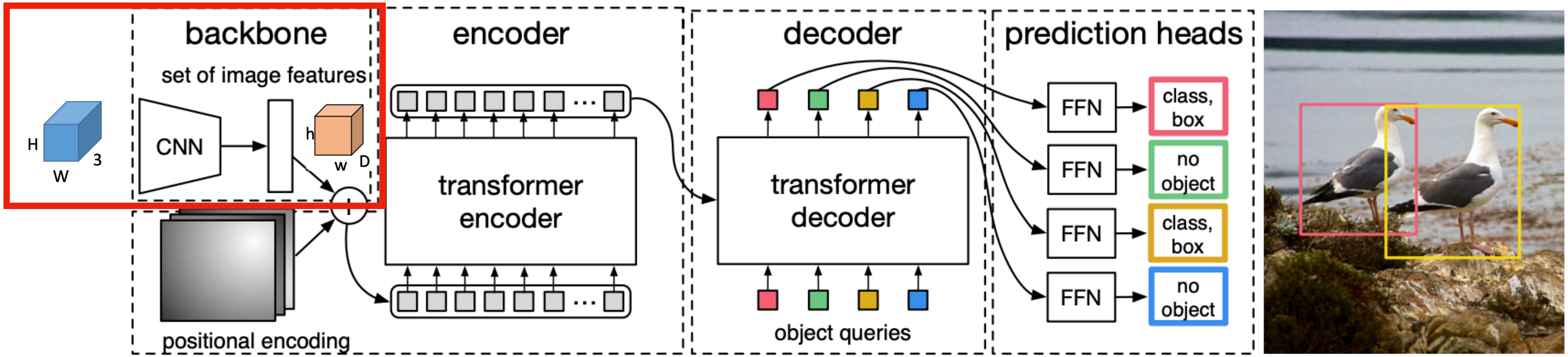
Optimal assignment computed in the first step

DETR - Bounding Box Loss

Bounding box loss. The second part of the matching cost and the Hungarian loss is $\mathcal{L}_{\rm box}(\cdot)$ that scores the bounding boxes. Unlike many detectors that do box predictions as a Δ w.r.t. some initial guesses, we make box predictions directly. While such approach simplify the implementation it poses an issue with relative scaling of the loss. The most commonly-used ℓ_1 loss will have different scales for small and large boxes even if their relative errors are similar. To mitigate this issue we use a linear combination of the ℓ_1 loss and the generalized IoU loss [38] $\mathcal{L}_{\rm iou}(\cdot,\cdot)$ that is scale-invariant. Overall, our box loss is $\mathcal{L}_{\rm box}(b_i, \hat{b}_{\sigma(i)})$ defined as $\lambda_{\rm iou}\mathcal{L}_{\rm iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\rm L1}||b_i - \hat{b}_{\sigma(i)}||_1$ where $\lambda_{\rm iou}, \lambda_{\rm L1} \in \mathbb{R}$ are hyperparameters. These two losses are normalized by the number of objects inside the batch.

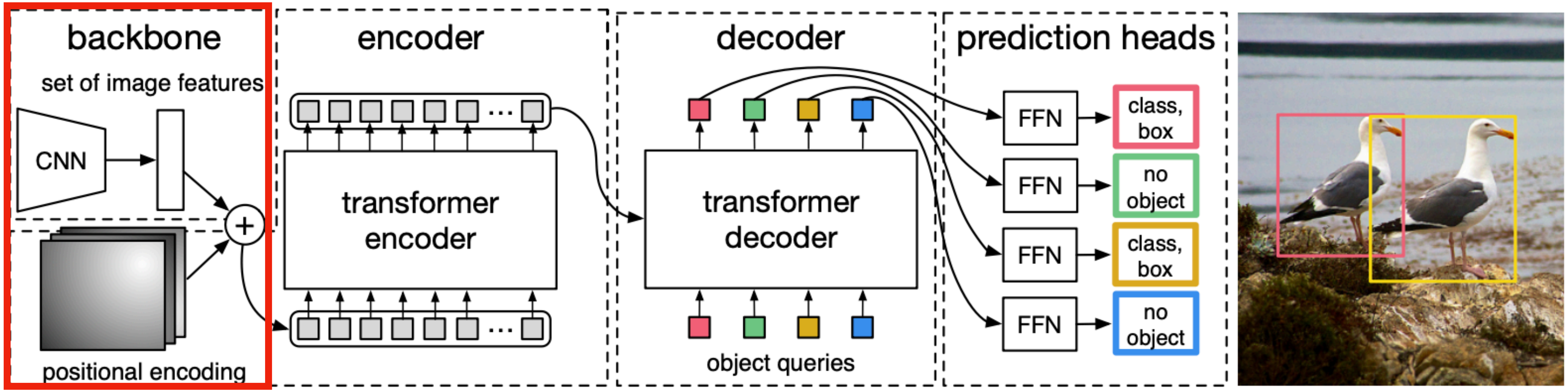
•

DETR - a closer look

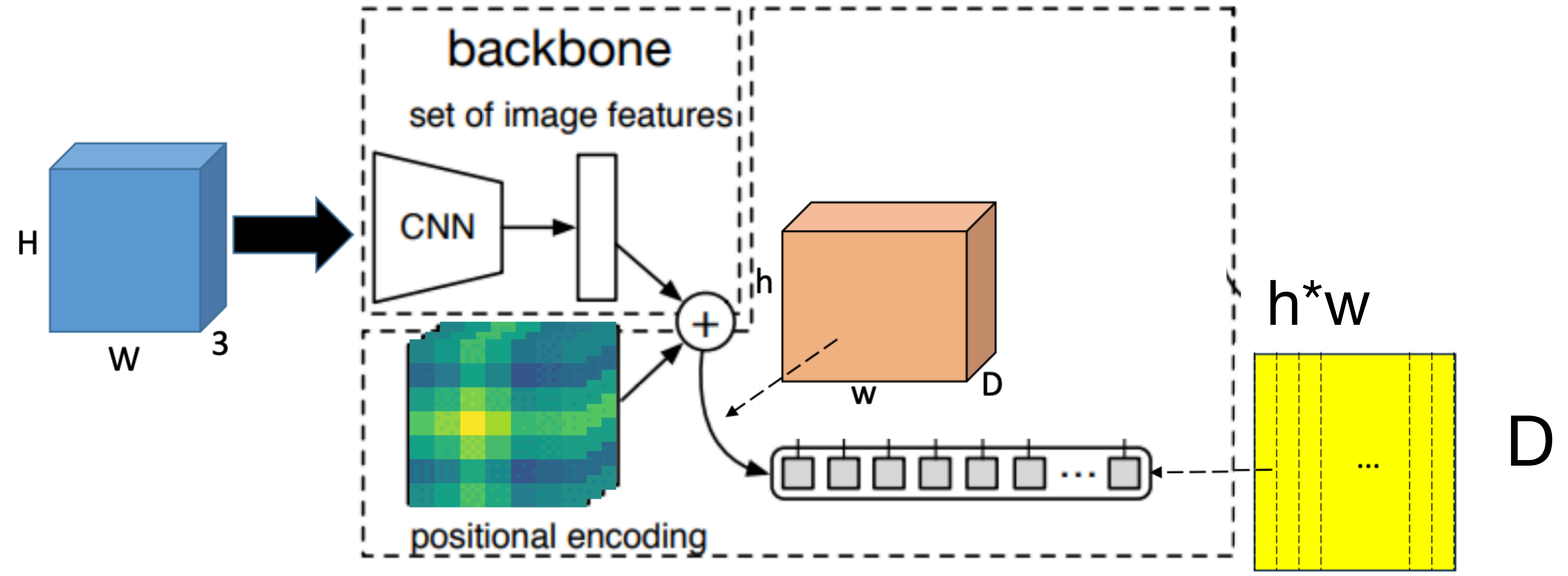


- DETR uses a conventional CNN backbone to learn a 2D representation of an input image.

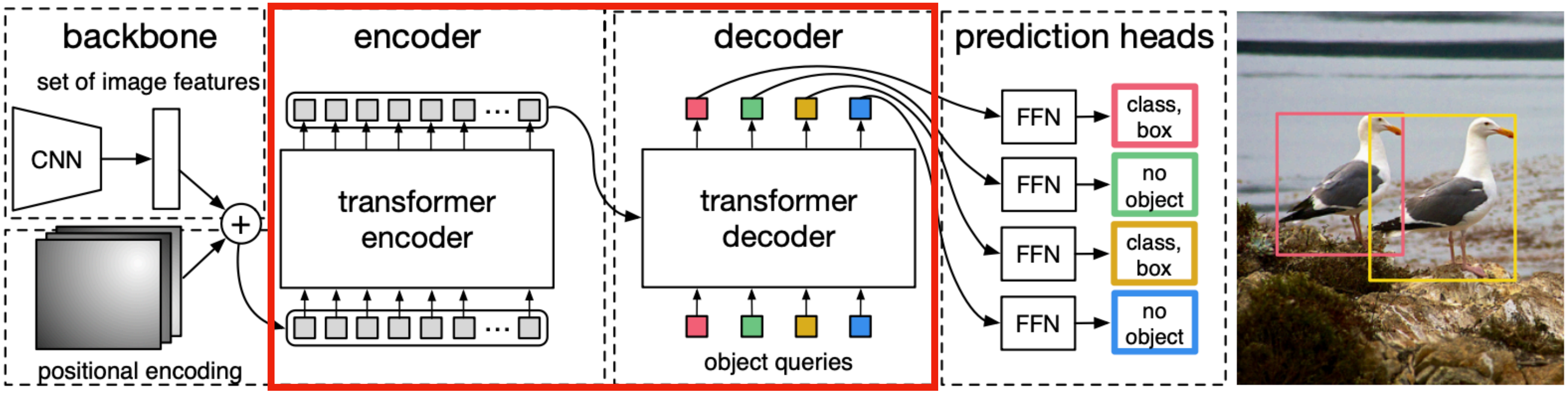
DETR - a closer look



- The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder.

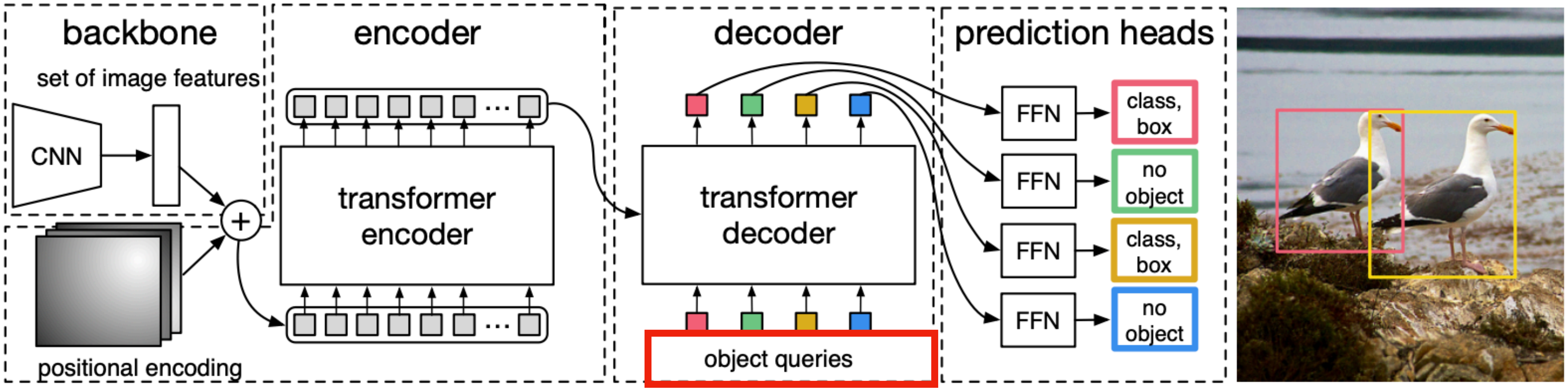


DETR - a closer look

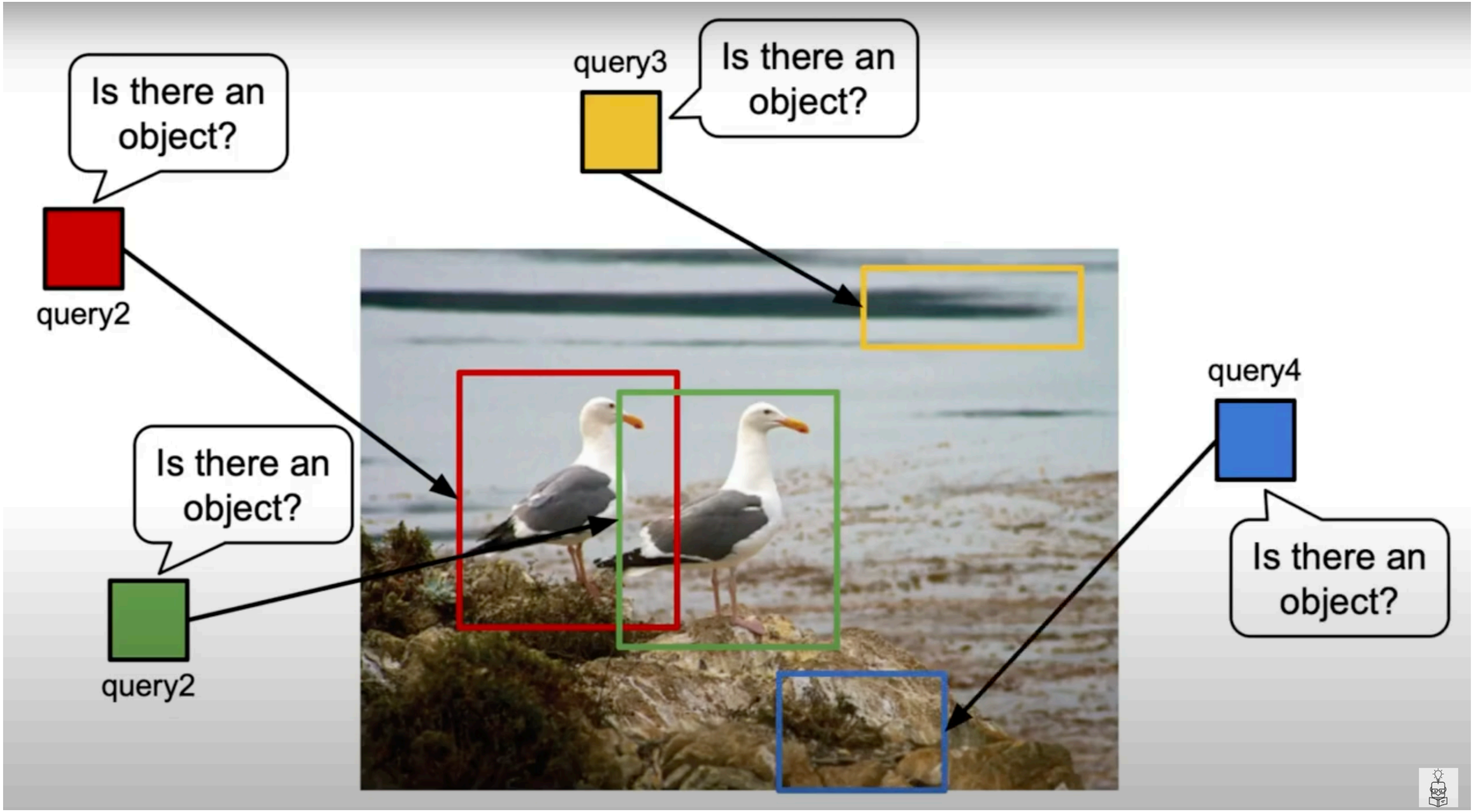


- A transformer first encodes the given input, and then the decoder takes as input a small fixed number of learned positional embeddings, which we call object queries, and additionally attends to the encoder output.

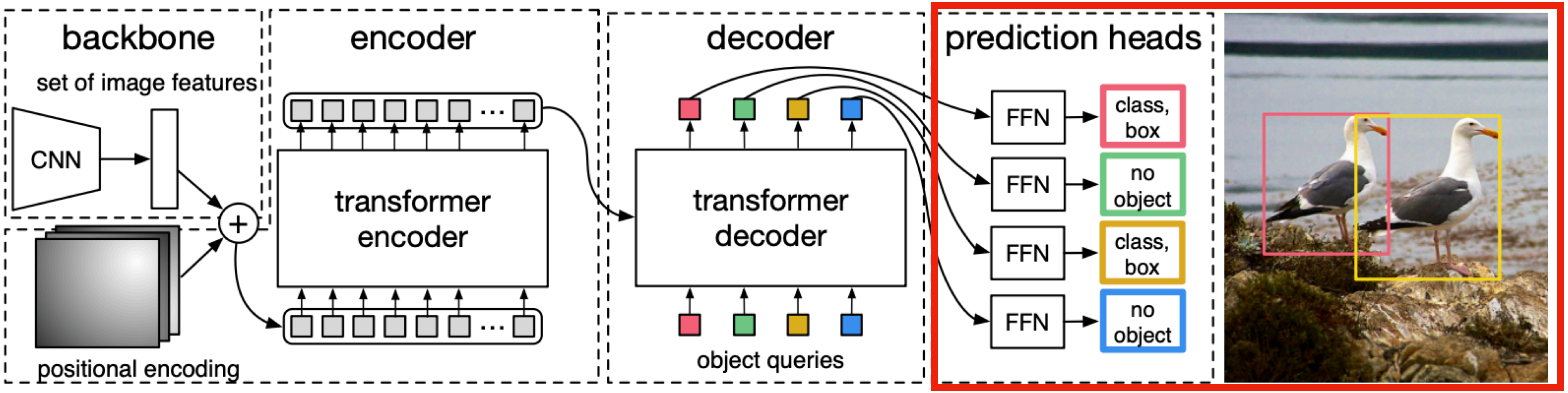
DETR - a closer look



- are randomly initialized embeddings
- refined through the course of training, and
- then fixed for evaluation.

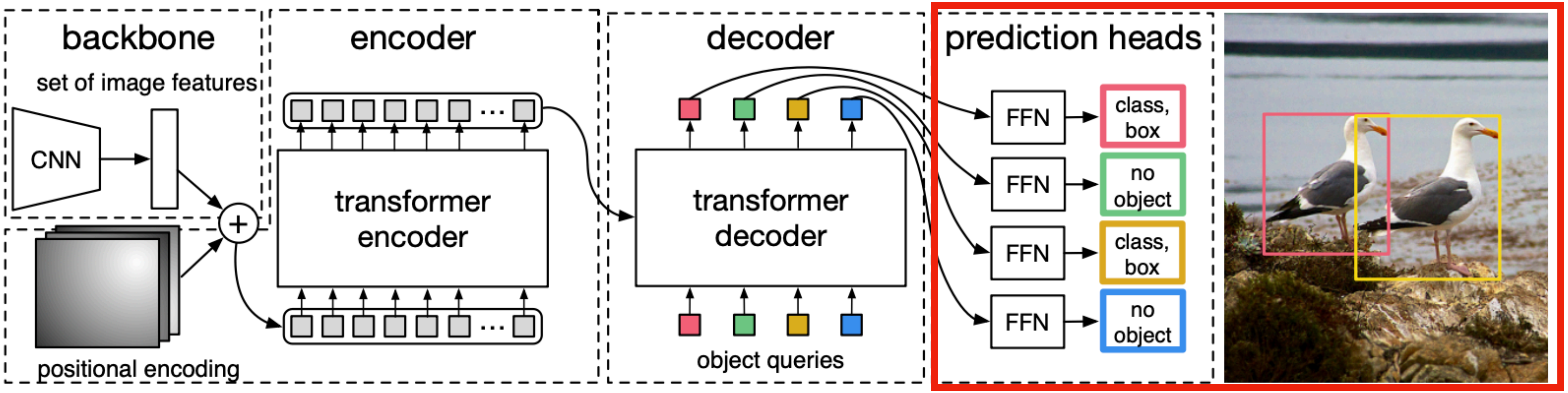

query3
Is there an object?
Is there an object?
query2
query4
Is there an object?
Is there an object?
query2
Is there an object?

DETR - a closer look



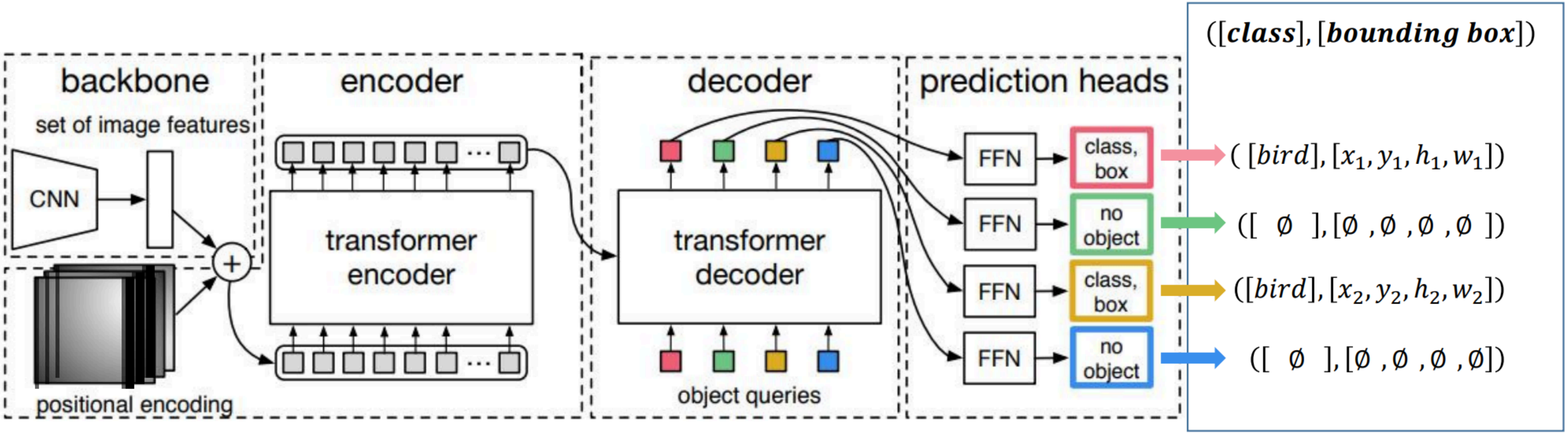
- Each output embedding of the decoder is passed to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “no object” class.

DETR - a closer look



- Each output embedding of the decoder is passed to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "no object" class.

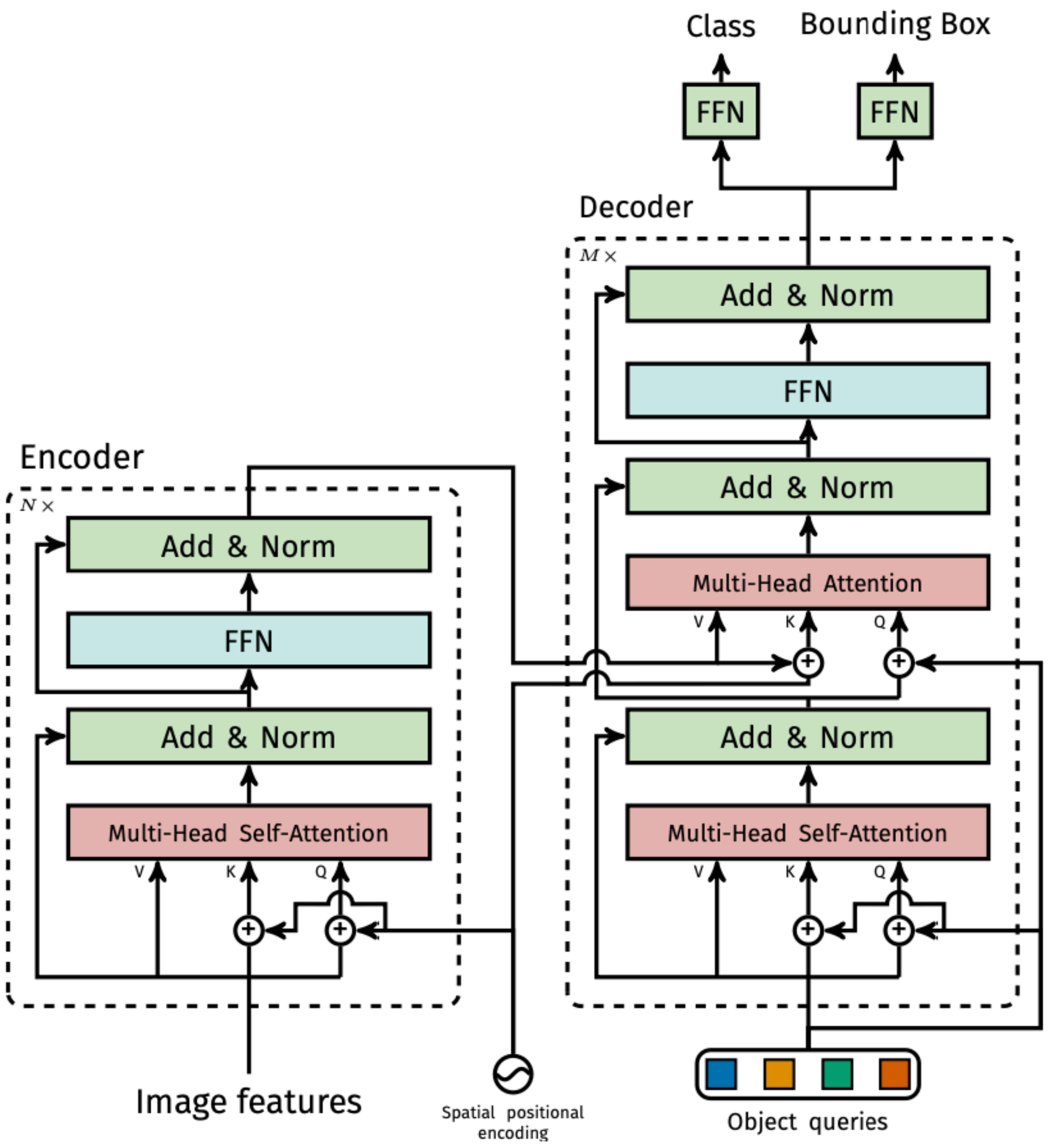
DETR - a closer look



- Each output embedding of the decoder is passed to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “no object” class.

DETR - Transformer Architecture

- Very similar to Attention is All you need architecture, with just a few addition made to work for this particular problem.

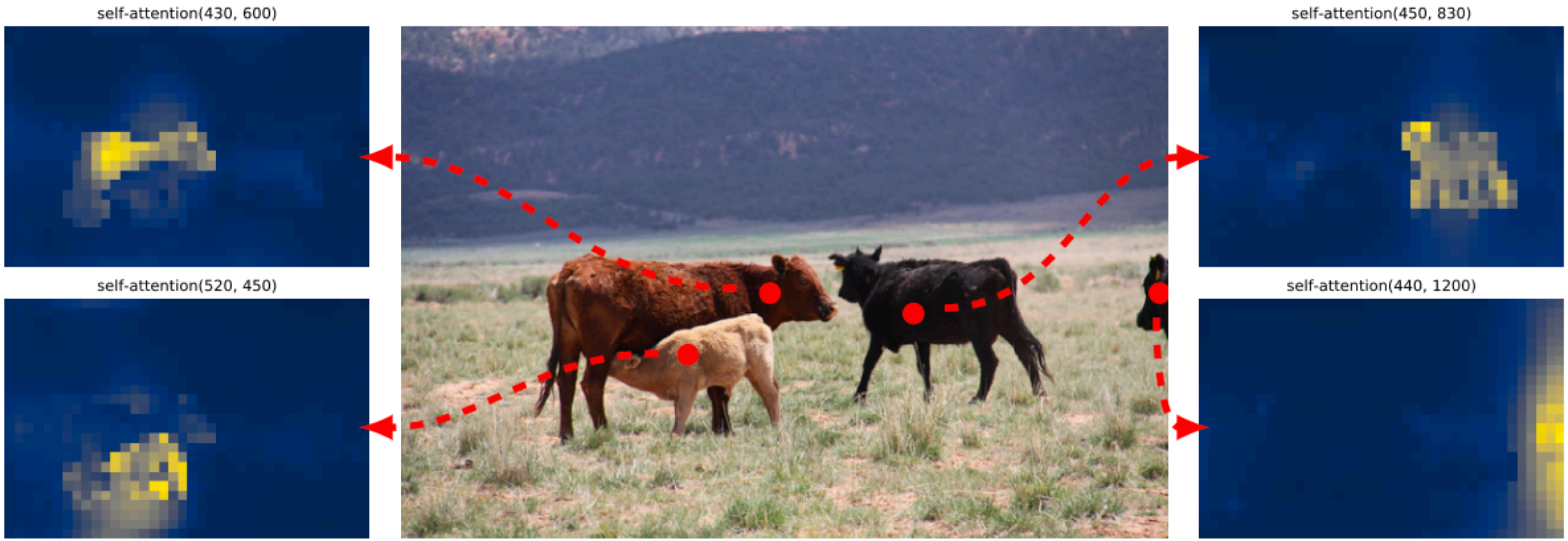


Experimental Results

DETR - Detection Results

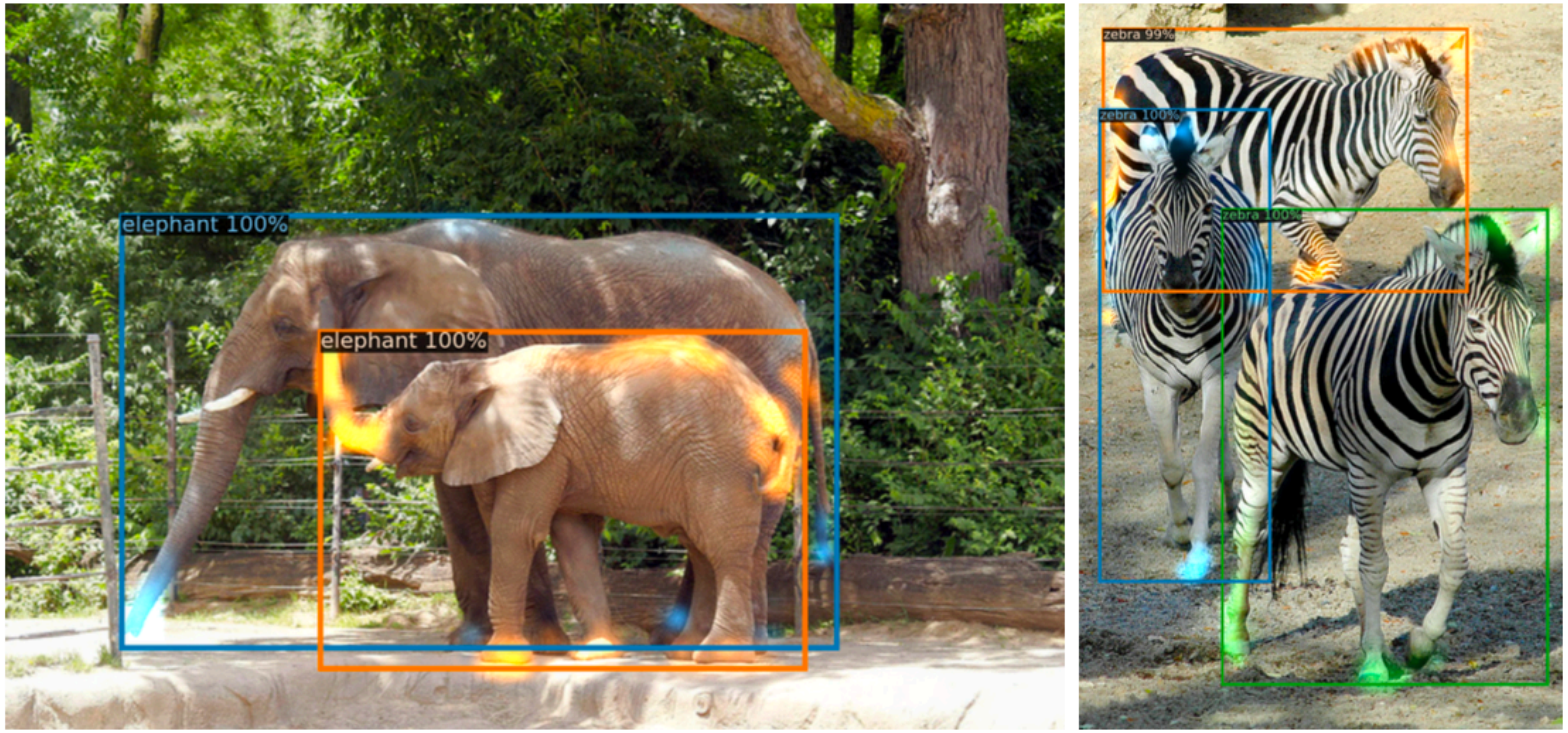
Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

DETR - Quantitative results



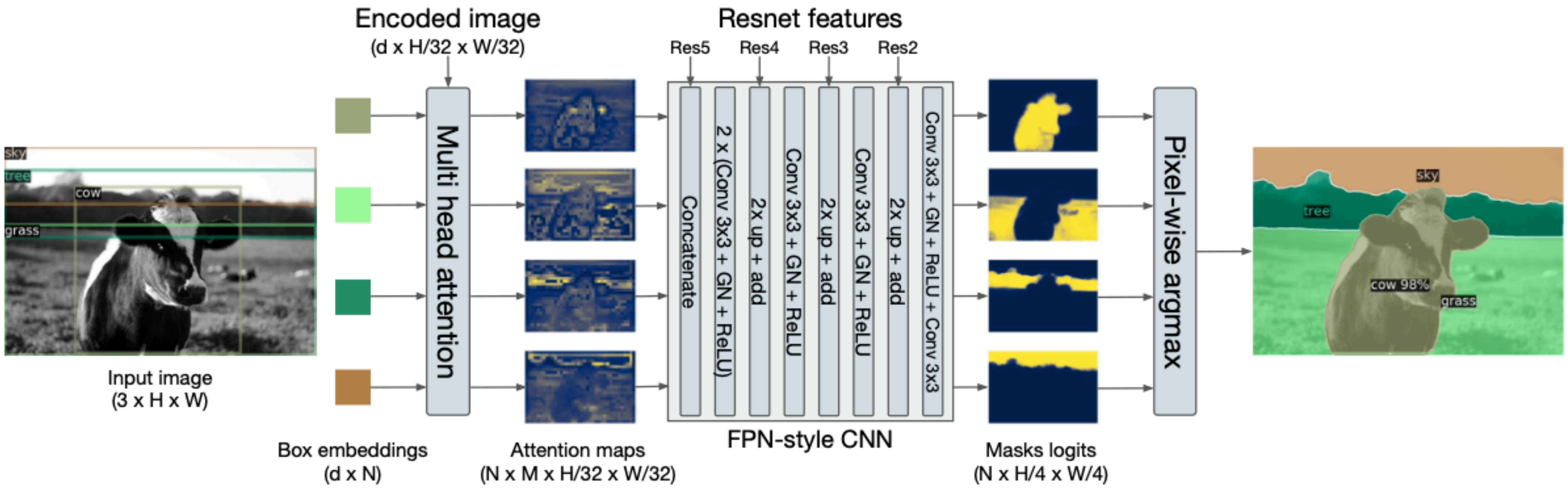
- The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

DETR - Quantitative results



- Visualizing decoder attention for every predicted object. Attention scores are coded with different colors for different objects. Decoder typically attends to object extremities, such as legs and heads.

DETR used for panoptic semantic segmentation



- A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax
  - Compute multi-head attention heatmap of decoder output over encoder output
  - Use an FPN-like architecture to increase the resolution of the mask
  - Mask is supervised independently using DICE/F-1 loss and Focal loss

DETR Panoptic Segmentation - results

Model	Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}	AP
PanopticFPN++	R50	42.4	79.3	51.6	49.2	82.4	58.8	32.3	74.8	40.6	37.7
UPSnet	R50	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7	34.3
UPSnet-M	R50	43.0	79.1	52.8	48.9	79.7	59.7	34.1	78.2	42.3	34.3
PanopticFPN++	R101	44.1	79.5	53.3	**51.0**	**83.2**	60.6	33.6	74.0	42.1	**39.7**
DETR	R50	43.4	79.3	53.8	48.2	79.8	59.5	36.3	78.5	45.3	31.1
DETR-DC5	R50	44.6	79.8	55.0	49.4	80.5	60.6	**37.3**	**78.7**	**46.5**	31.9
DETR-R101	R101	**45.1**	**79.9**	**55.5**	50.5	80.9	**61.7**	37.0	78.5	46.0	33.0

Summary

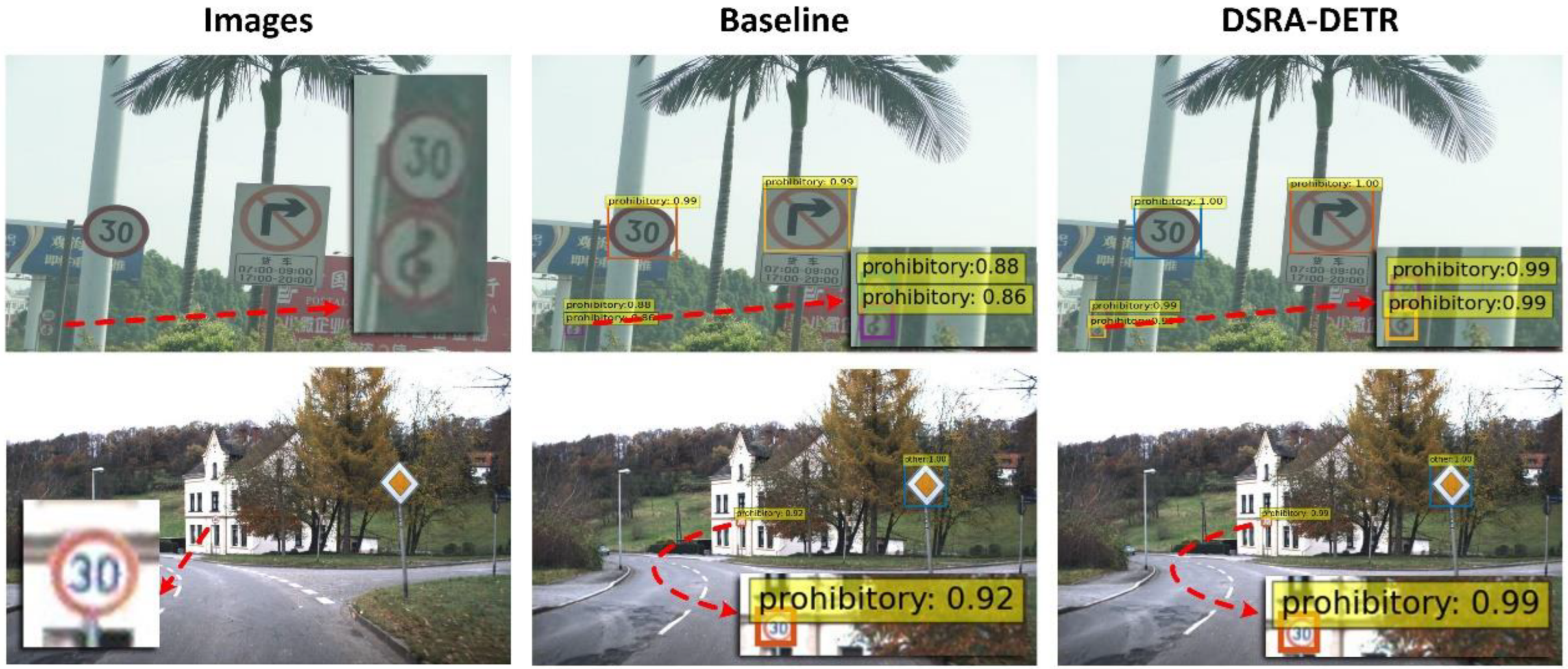
- A **fresh design** for object detection systems based on transformersand bipartite matching loss for direct set prediction.
- Significantly better performance on **large objects** than Faster R-CNN
- **Global information** performed by the self-attention.
- With a minor modification can be used for panoptic segmentation.
- It under performs on a **smaller objects** compared to other object detectors of same magnitude.
- It takes **long training hours** and is not real time.
- The transformer architecture leads to significant overhead in training/inference

Extended Research

- A Survey on Vision Transformer
  - Addresses that DETR has a slow convergence and other limitations of DETR.
  - Proposed several papers that improved DETR’s training time and AP.
- DEFORMABLE DETR: DEFORMABLE TRANSFORMERS FOR END-TO-END OBJECT DETECTION
  - Use a deformed attention module instead of self-attention, which attends to a small sample of feature maps instead of all, and this improves both time complexity and AP.
- Points as Queries: Weakly Semi-supervised Object Detection by Points
  - Encode object centers (points) as object queries to DETR instead of learnt positional encodings. This is done by using a point encoder on predicted points on an image.
- Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions
  - A backbone that uses a transformer to generate feature pyramids, and the features are compatible with DETR.

Practical Applications of DETR

- Autonomous vehicles
    - (DSRA-DETR: An Improved DETR for Multiscale Traffic Sign Detection)
- Temporal action localization
- Robotics
- Segmentation/ Super-resolution
- ……



High-Resolution Image Synthesis with Latent Diffusion Models

Owais Saad Shuja
March 5, 2025

Introduction

What problem does this paper solve ?

- Generating high-quality images efficiently using generative models.
- Standard diffusion models work in pixel space, making them computationally expensive.
- The paper proposes a more efficient alternative: **Latent Diffusion Models (LDMs)**.

- Main contributions
  - Introduces **LDMs**, which perform diffusion in a lower-dimensional latent space.
  - Balances efficiency and high-resolution image synthesis.

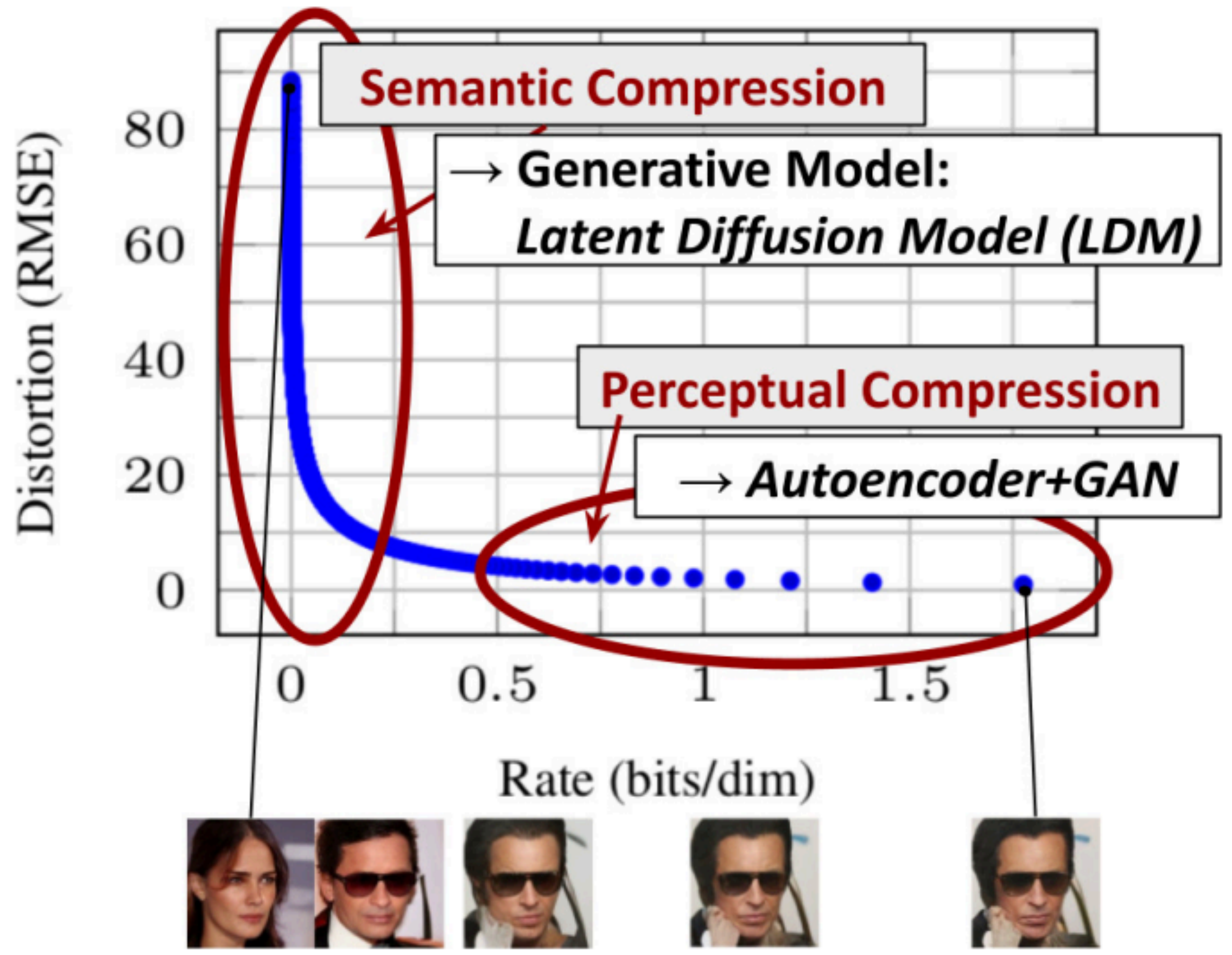
Motivation

- What are Diffusion Models ?
  - Generative models that incrementally denoise Gaussian noise to generate data (images).
  - Popular due to their high-quality image synthesis

- What are the limitations of traditional Diffusion Models?
  - Operate in pixel space, leading to high computational costs.
  - Inefficient for high-resolution images.

Motivation (contd.)

- How do Variational Autoencoders (VAEs) and GANs relate?
  - VAEs encode images into a latent space for **compressed representations**.
  - GANs are used for high-quality generation but suffer from **mode collapse**.
  - LDMs leverage VAE-based latent representations to improve efficiency.



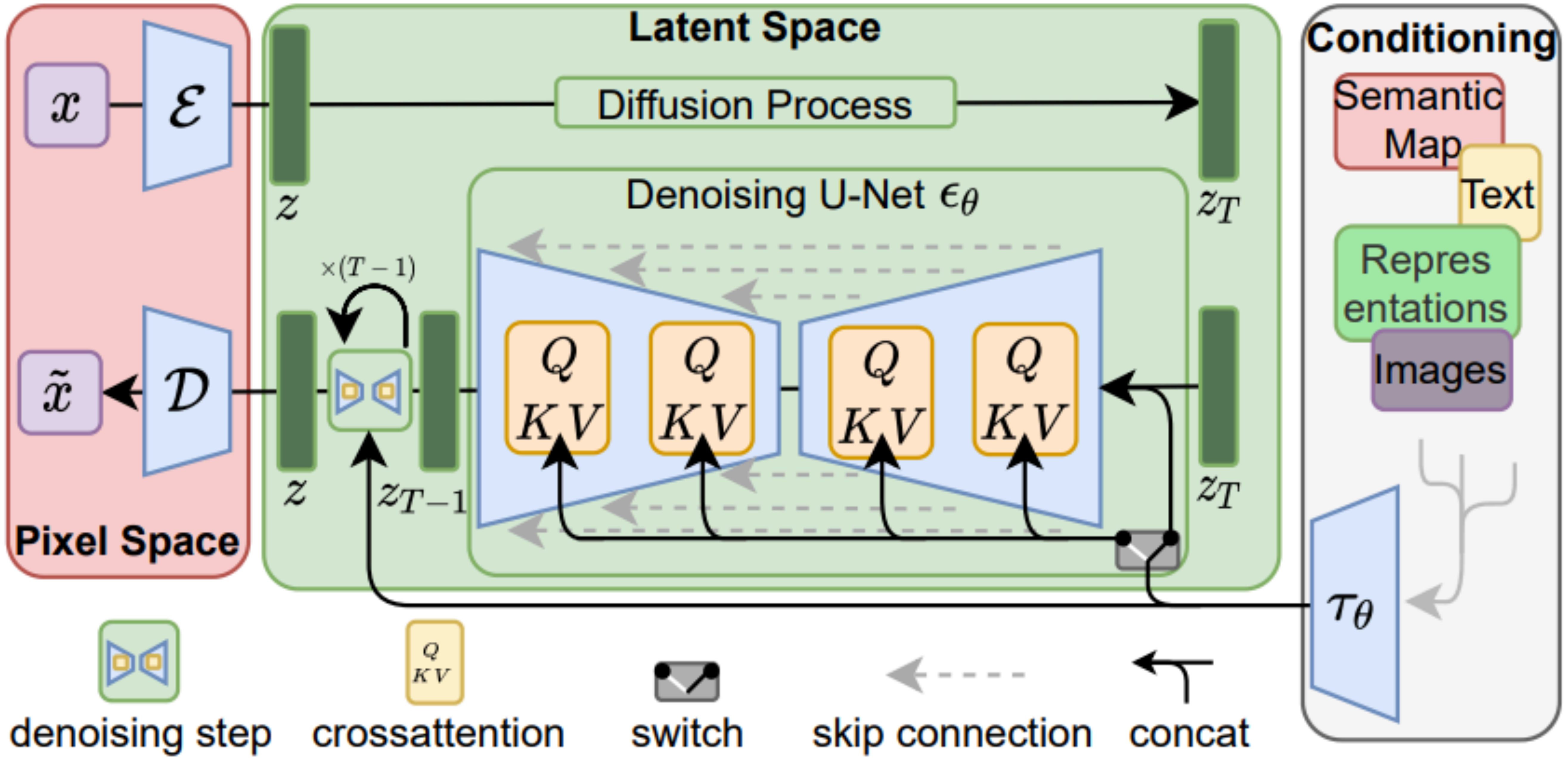
Latent Diffusion Models

- What is the key idea behind LDMs?
  - Instead of applying diffusion in **pixel space**, LDMs perform diffusion in a **latent space**.
  - This significantly reduces computational cost without sacrificing quality.

- How does LDM work?
  - Uses a **pretrained VAE** to encode an image into a **latent representation**.
  - The diffusion process is applied in this latent space.
  - After diffusion, the latent representation is **decoded back into an image**.

LDM (contd.)

- Why is Latent Space Better?
  - **Lower dimensionality** → Less computational power required.
  - Still retains important semantic features of the image.



Model Architecture

- What are the primary components of the LDM architecture?
  - VAE Encoder-Decoder:
    - Encodes images into a latent space and decodes latent representations back to images.
  - U-Net-based Diffusion Model:
    - Applies noise in the latent space and learns to denoise iteratively to reconstruct high-quality latent representations.
  - Cross-Attention Mechanisms:
    - Incorporated to allow conditioning on various inputs, such as text descriptions or bounding boxes, enhancing the model's flexibility.

Model Arch. (Contd.)

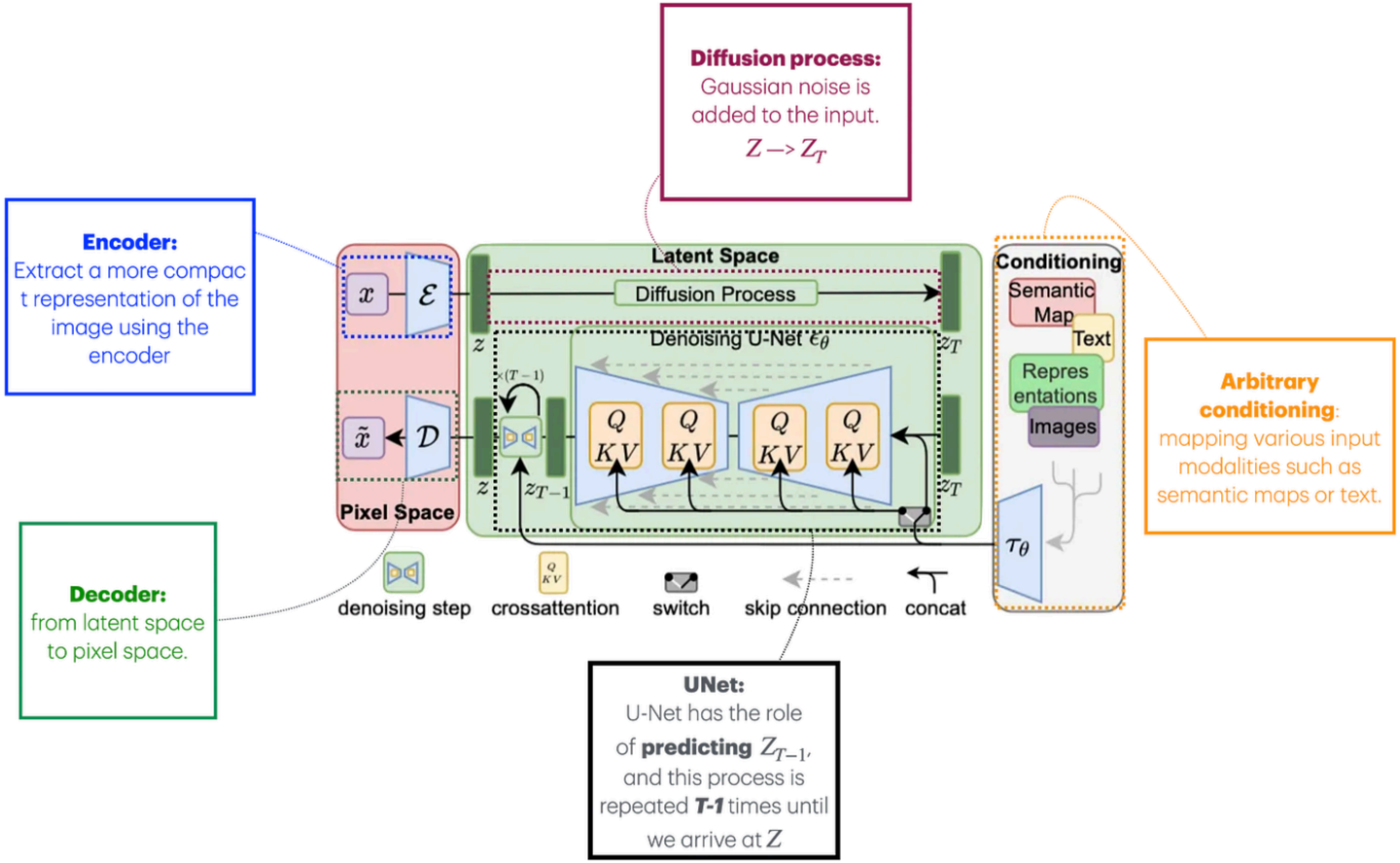
- How does the cross-attention mechanism enhance the model's capabilities?
  - It enables the model to integrate additional information (e.g., text prompts) seamlessly, guiding the image generation process to produce outputs aligned with the provided conditions.

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right], \qquad (1)$$

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right]. \qquad (2)$$

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right], \qquad (3)$$

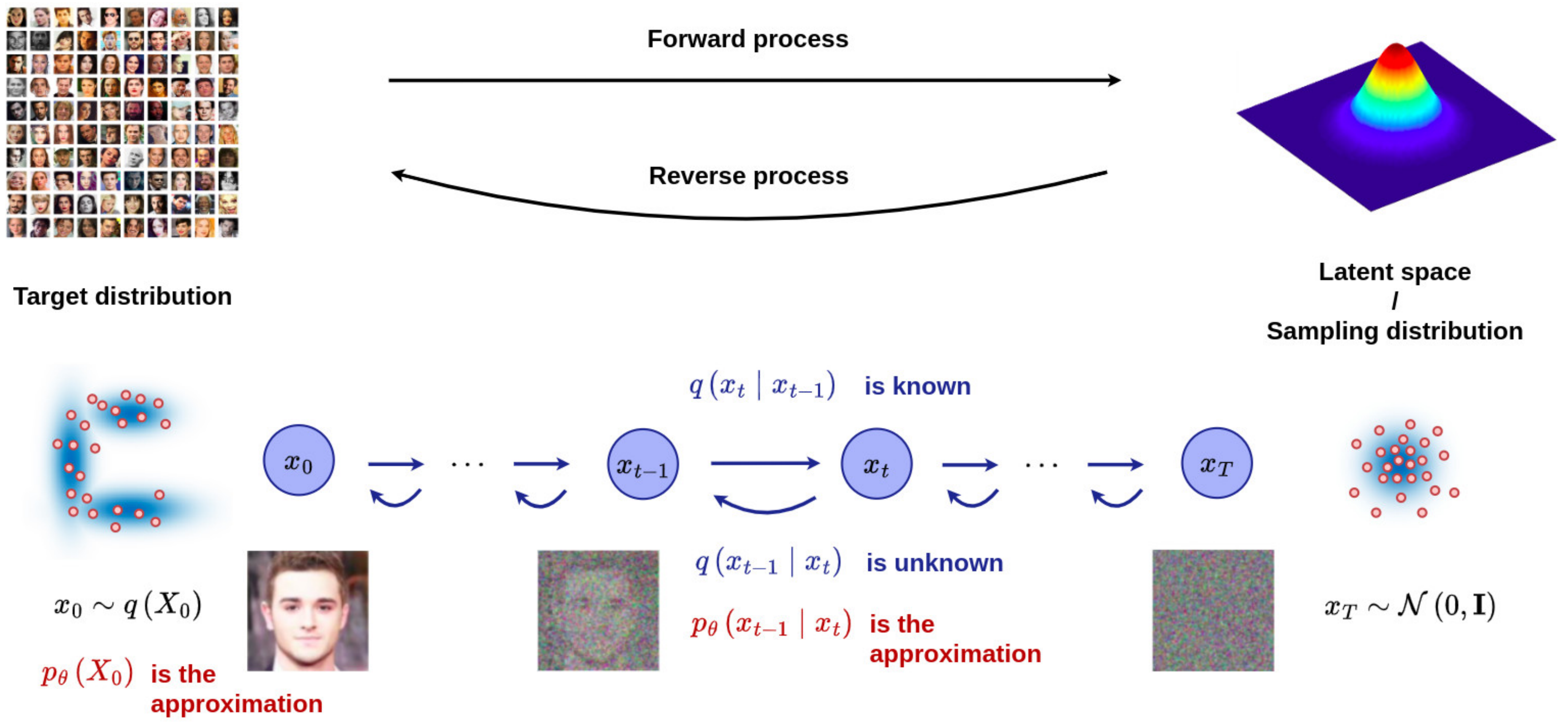
Overview of the Latent Diffusion Architecture



Overview of the latent diffusion architecture.

Training Procedure

- How is the training of LDMs structured?
  - **Stage 1**: Train a VAE to learn an effective latent space representation of images.
  - **Stage 2**: Train the diffusion model within this latent space to learn the denoising process, effectively modeling the data distribution



Training Procedure (contd.)

- What loss functions are utilized during training?
  - **VAE Training**: Combines reconstruction loss (to ensure accurate decoding) and a regularization term (to enforce a structured latent space).
  - **Diffusion Model Training**: Employs a denoising score matching loss, training the model to predict and remove noise effectively at each diffusion step.
- Why is a two-stage training process beneficial?
  - Separating the training allows the VAE to focus on learning a compact and informative latent representation, while the diffusion model specializes in the generative process within this efficient latent space, leading to improved performance and reduced computational requirements.

Results

- How do LDMs perform compared to traditional diffusion models and GANs?
  - LDMs achieve comparable or superior image quality with significantly reduced computational resources.
  - They demonstrate state-of-the-art performance in tasks like image inpainting and class-conditional image synthesis.

Figure 4. Samples from *LDMs* trained on CelebAHQ [39], FFHQ [41], LSUN-Churches [102], LSUN-Bedrooms [102] and class-conditional ImageNet [12], each with a resolution of 256 × 256. Best viewed when zoomed in. For more samples *cf*. the supplement.

Text-to-Image Synthesis on LAION. 1.45B Model.

'A street sign that reads "Latent Diffusion" '	*'A zombie in the style of Picasso'*	*'An image of an animal half mouse half octopus'*	*'An illustration of a slightly conscious neural network'*	*'A painting of a squirrel eating a burger'*	*'A watercolor painting of a chair that looks like an octopus'*	*'A shirt with the inscription: "I love generative models!" '*

Figure 5. Samples for user-defined text prompts from our model for text-to-image synthesis, *LDM-8 (KL)*, which was trained on the LAION [78] database. Samples generated with 200 DDIM steps and $\eta = 1.0$. We use unconditional guidance [32] with $s = 10.0$.

Summary

- **Latent Diffusion Models (LDMs)**: The paper introduces a new diffusion-based generative model that **operates in latent space** rather than pixel space, significantly reducing computational costs while maintaining high-quality image synthesis.
- Two-Stage Training:
  - A **Variational Autoencoder (VAE)** is first trained to compress images into a lower-dimensional latent space.
  - A **diffusion model** then learns the generative process in this latent space
- **U-Net Architecture**: The diffusion model uses a **U-Net-based architecture** with **cross-attention mechanisms**, enabling flexible conditioning on different inputs (e.g., text, class labels).

Summary (contd.)

- **Efficiency Gains**: LDMs require significantly **less computation** than traditional diffusion models and **outperform GANs** in terms of sample diversity and quality.
- **Multimodal Applications**: The model successfully generates images **conditionally** based on textual prompts (text-to-image), class labels, and even performs tasks like **inpainting** and **super-resolution**.