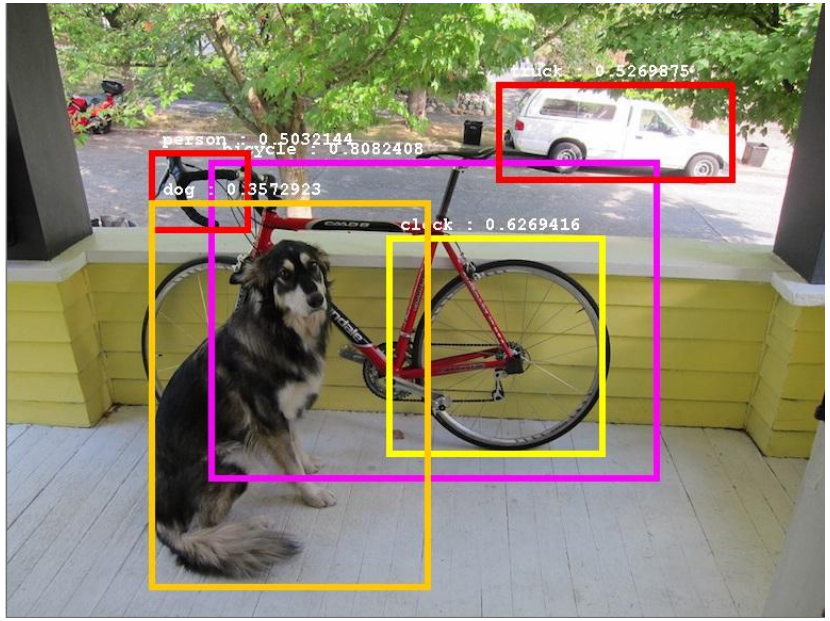
Advanced Topics

YOLO

YOLO is a **state-of-the-art** object **detection** model in **computer vision**.

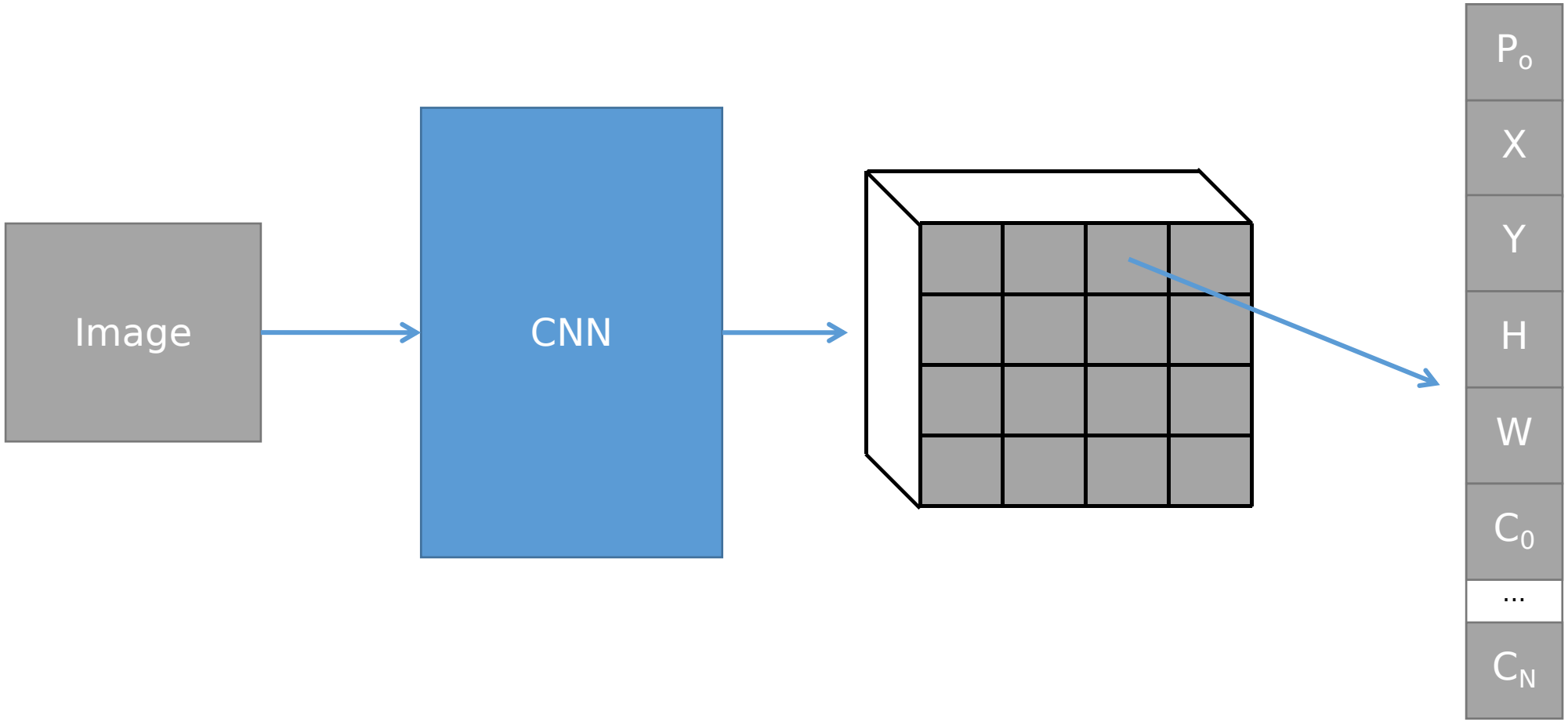
Unlike **traditional methods** that **require multiple passes** over an image, **YOLO performs** real-time object **detection** in a **single pass**, making it **incredibly fast**.

It **divides** an **image** into a **grid** and **predicts** bounding boxes and class probabilities for **objects** within **each grid cell**.

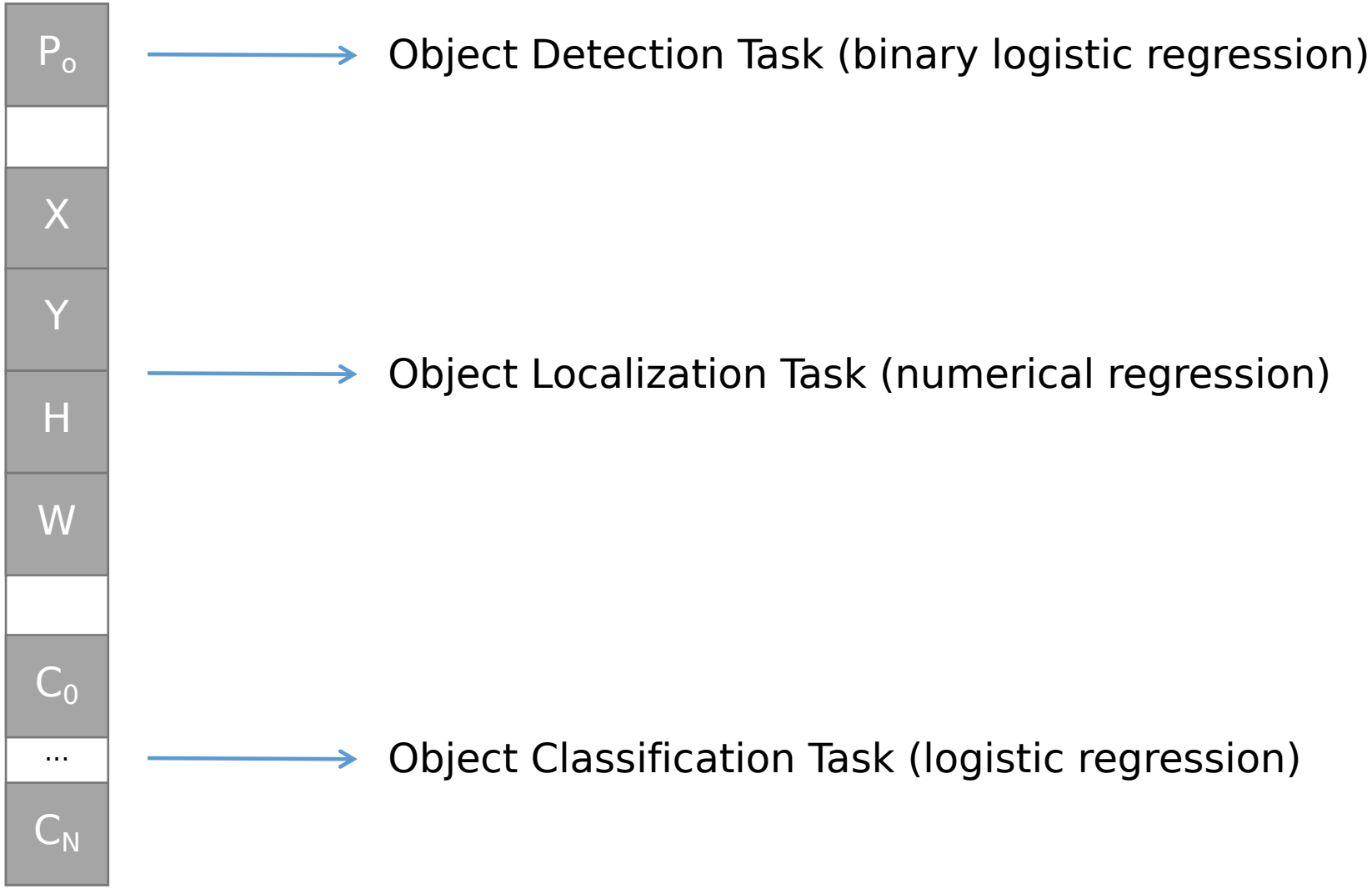


YOLO (1/3)

YOLO is **just** a **CNN** with a **specific depth** of the **output** tensor

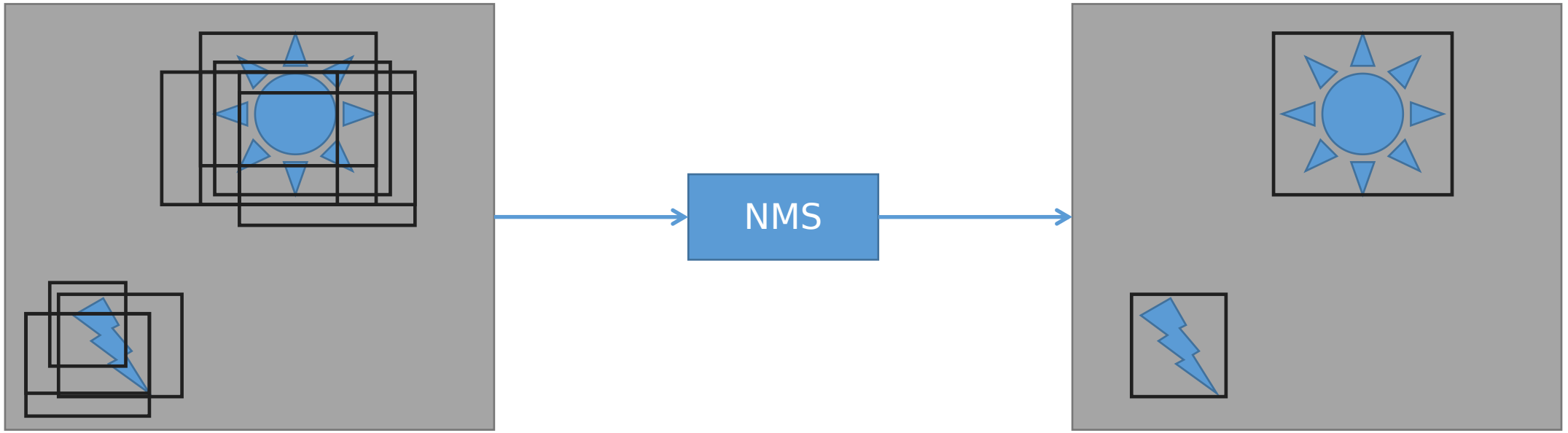


YOLO (2/3)


P_o
X
Y
H
W
C_0
...
C_N
Object Detection Task (binary logistic regression)
Object Localization Task (numerical regression)
Object Classification Task (logistic regression)

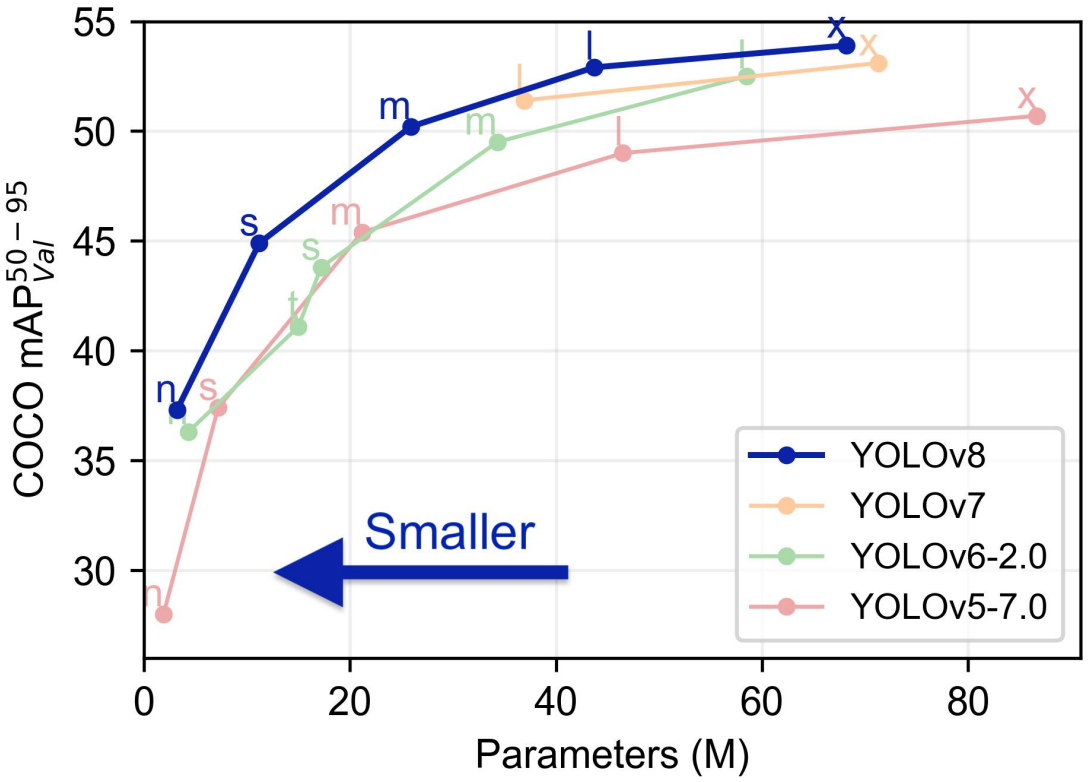
Non-Maximum Suppression

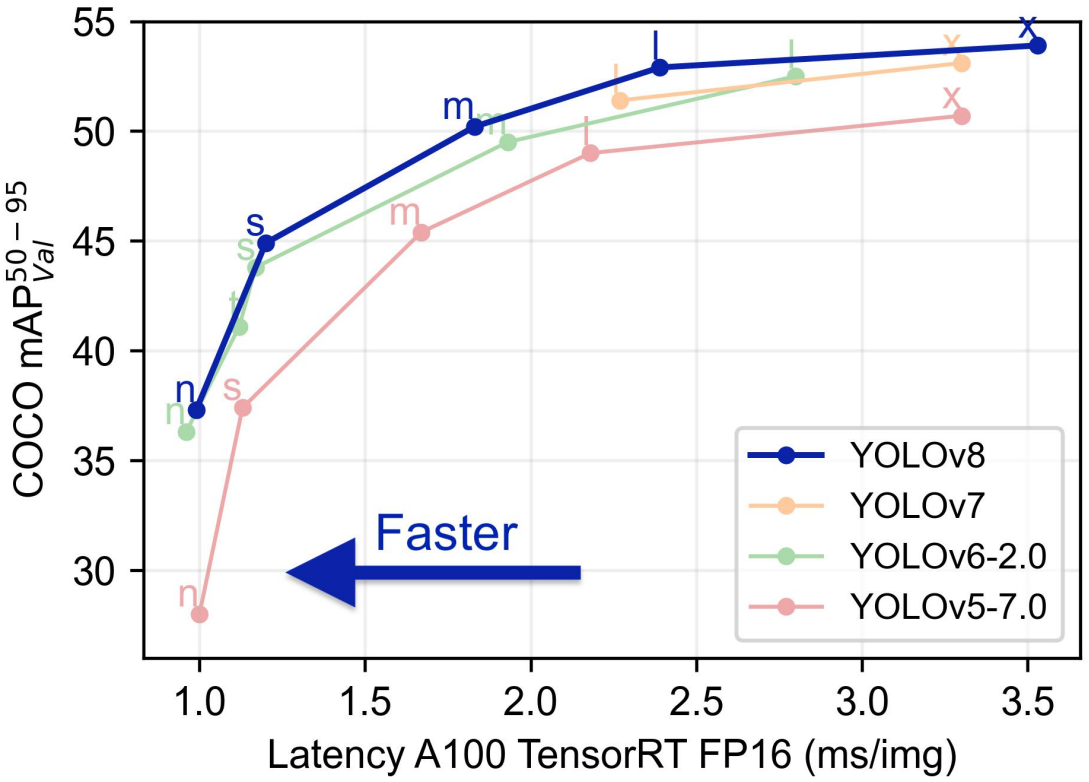
Non-Maximum Suppression (NMS) is a **technique** in computer vision used **to filter** out **redundant** bounding boxes in **object detection** tasks.
It **keeps** only the **most relevant** bounding boxes by **suppressing** those with **lower confidence** scores **or** significant **overlap** with others.



YOLOv8 (3/3)

Last version of the YOLO architecture released in 2023





YOLOv8: Implementation

A very famous and **well-maintained** implementation of **YOLOv8** is **ultralytics.** it has significantly contributed to the **YOLO** object detection framework by providing an **efficient** and **user-friendly implementation** of YOLOv8.

```python
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.yaml")  # build a new model from scratch
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
model.train(data="coco128.yaml", epochs=3)  # train the model
metrics = model.val()  # evaluate model performance on the validation set
results = model("https://ultralytics.com/images/bus.jpg")  # predict on an image
path = model.export(format="onnx")  # export the model to ONNX format
```

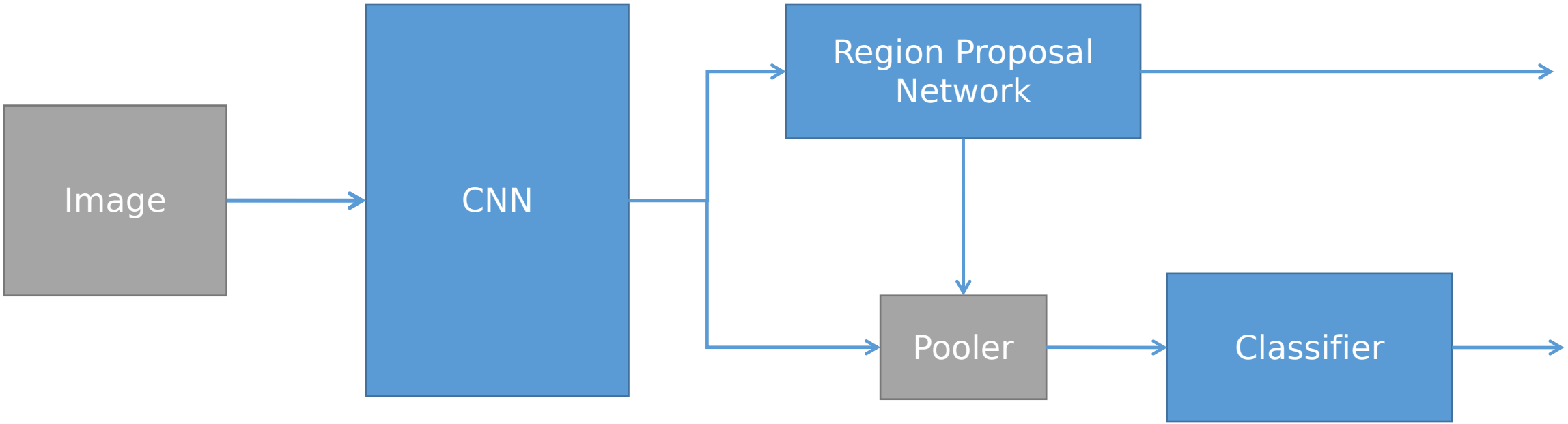
YOLOv8: Inference

```python
from ultralytics import YOLO
from PIL import Image
import cv2

model = YOLO("model.pt")
# accepts all formats - image/dir/Path/URL/video/PIL/ndarray. 0 for webcam
results = model.predict(source="0")
results = model.predict(source="folder", show=True) # Display preds. Accepts all YOLO predict arguments

# from PIL
im1 = Image.open("bus.jpg")
results = model.predict(source=im1, save=True)  # save plotted images

# from ndarray
im2 = cv2.imread("bus.jpg")
results = model.predict(source=im2, save=True, save_txt=True)  # save predictions as labels

# from list of PIL/ndarray
results = model.predict(source=[im1, im2])
```

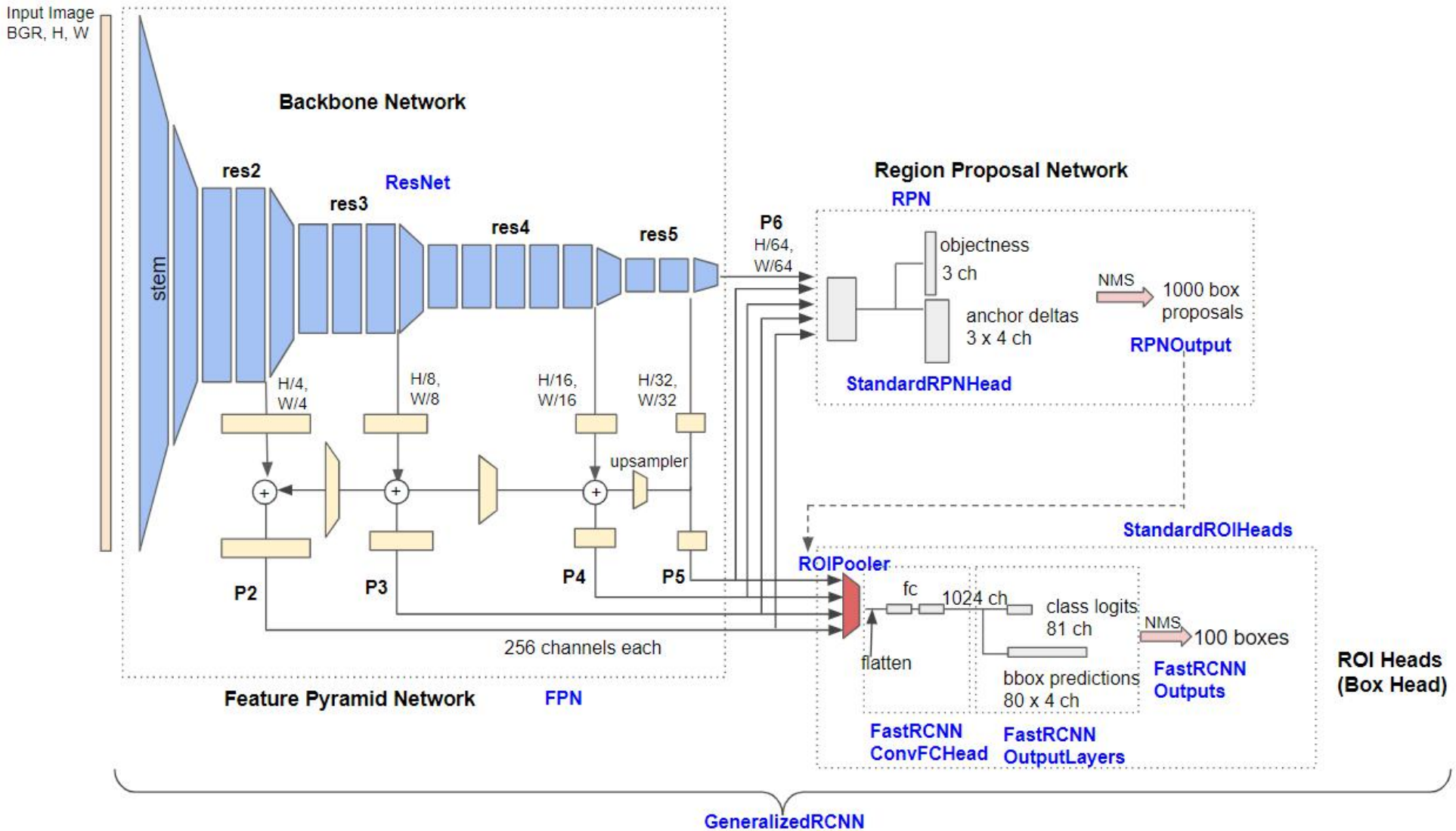
Faster R-CNN

Faster R-CNN is a deep learning framework for **object detection**.

It **combines** a **Region Proposal Network** (RPN) with a **CNN** to **efficiently** generate **region proposals** and **classify objects** within those regions.



Faster R-CNN: Architecture



Faster R-CNN vs YOLO

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Faster R-CNN: Detectron2

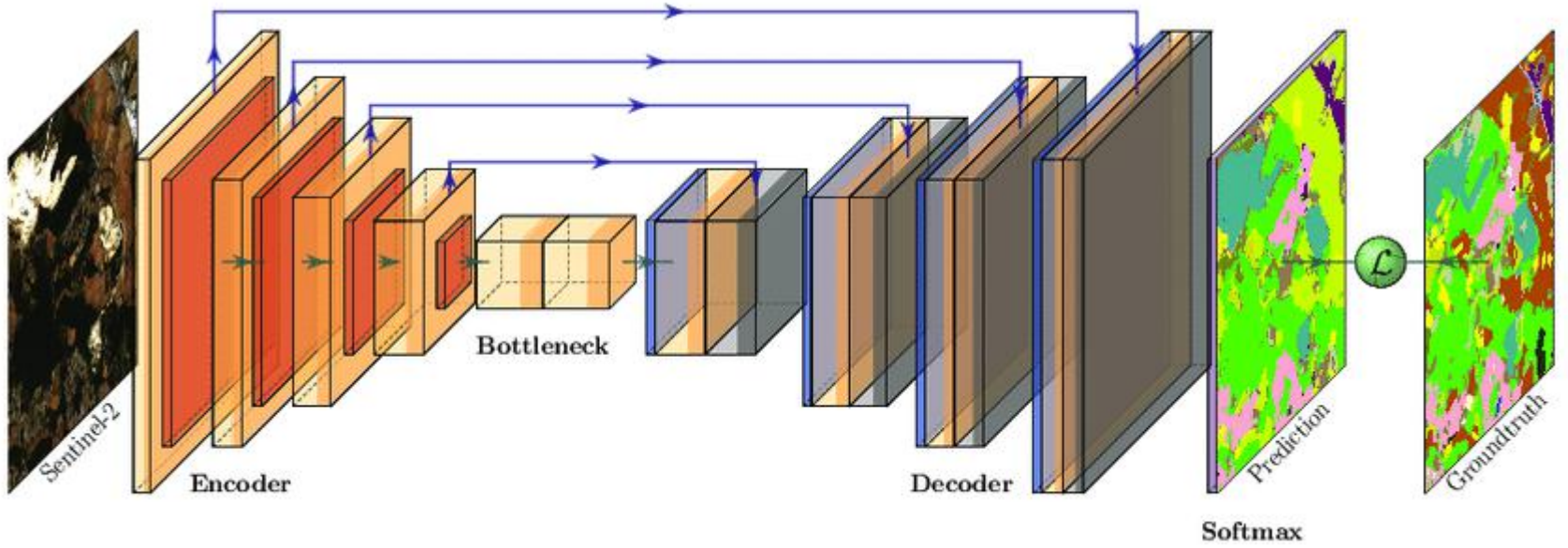
Detectron2 is a popular **deep learning framework** for **object detection** and **segmentation tasks**. Developed by **Facebook AI Research (FAIR)**, it builds upon the original Detectron framework and provides a more **modular** and **flexible** architecture.

UNET

U-Net is a **convolutional neural network** architecture designed for semantic **image segmentation** tasks.

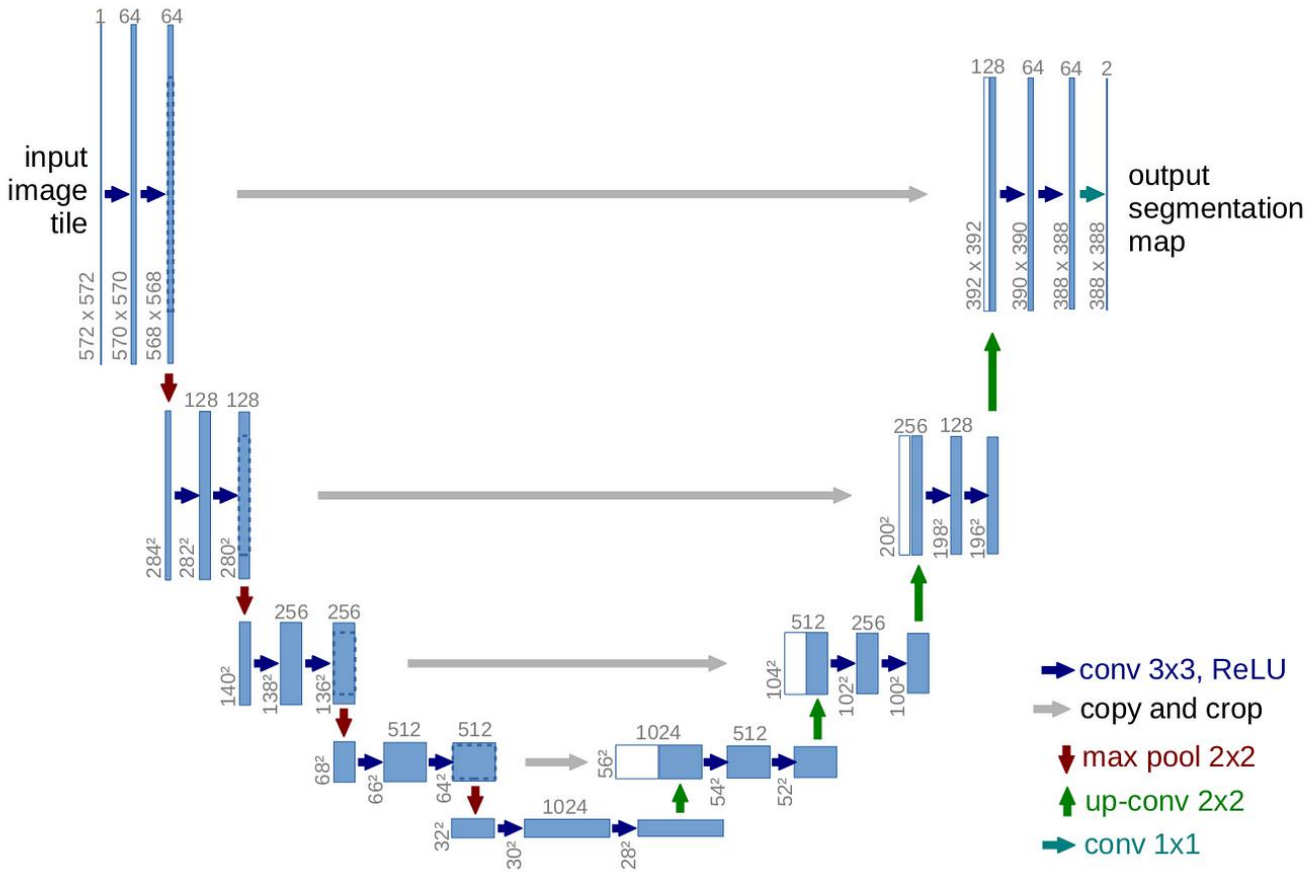
Its distinctive **U-shaped** architecture allows it to **capture fine-grained** details while **maintaining spatial** information.

U-Net has been **widely** used in **medical image** analysis and other fields where **precise image segmentation** is required.



UNET: Architecture

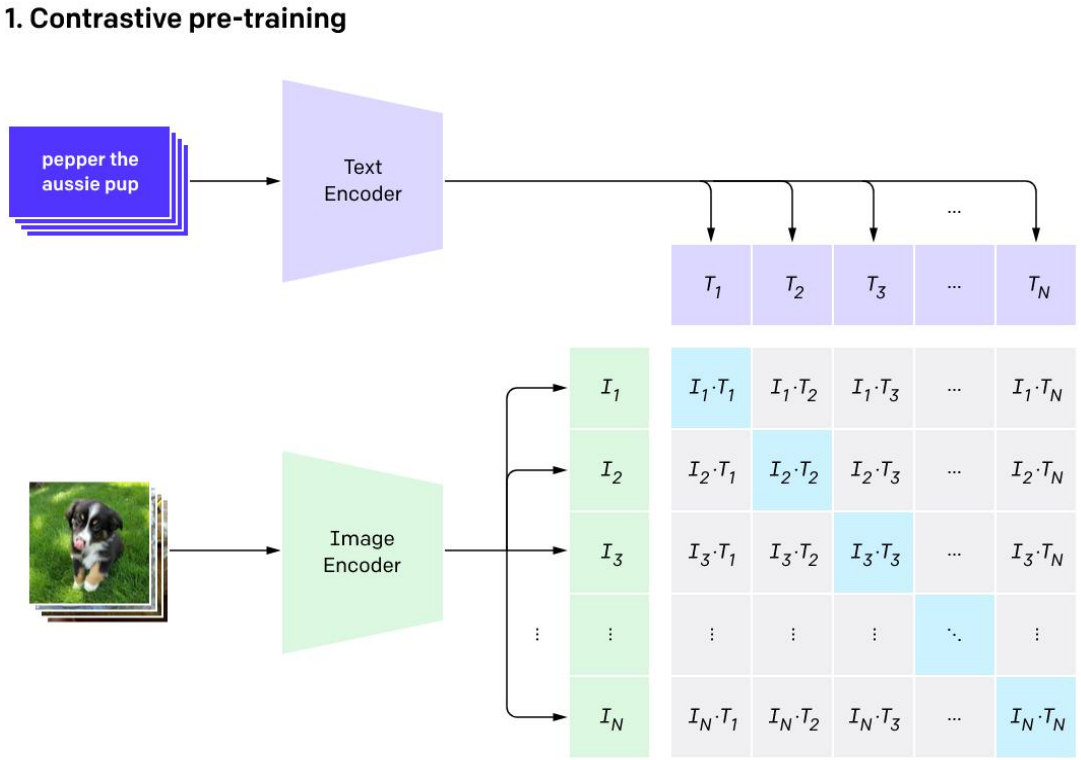
It consists of a **U-shaped** structure with two main parts: an **encoder** (the contracting path) and a **decoder** (the expansive path)



CLIP

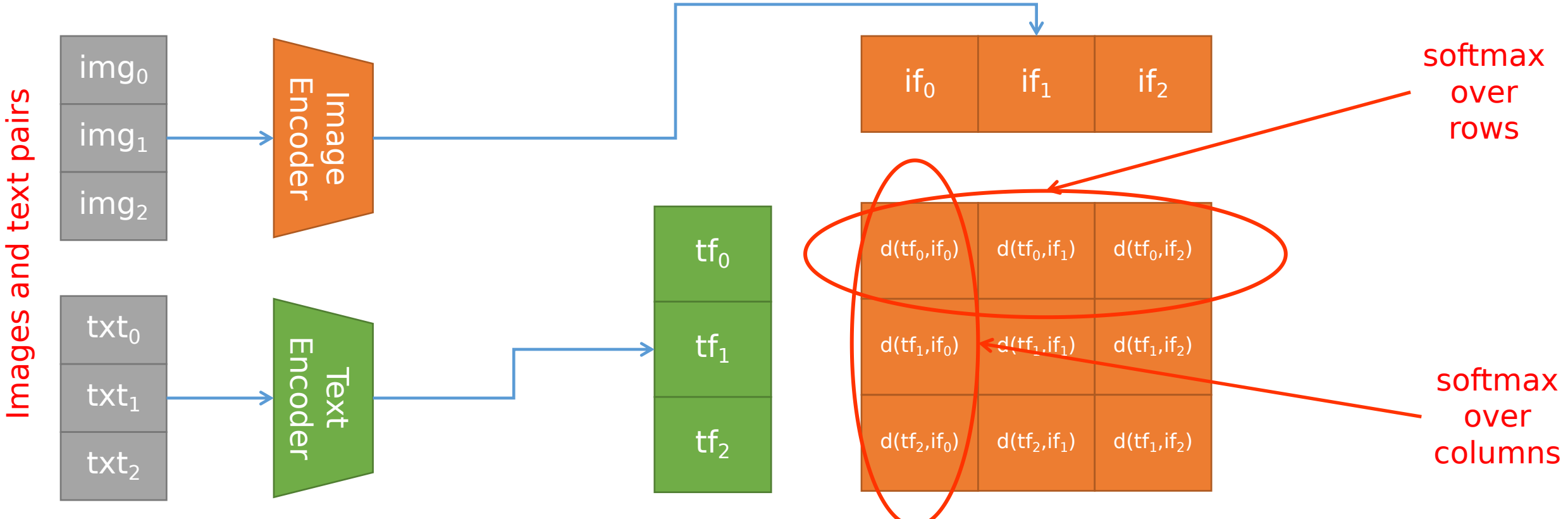
CLIP is a deep learning **model** developed by **OpenAI** that stands for "**Contrastive Language-Image Pretraining**."

It **combines vision** and **language** by learning to **associate images** and **text** in a way that allows it to perform a **wide range of tasks**, including image classification and natural language understanding, **without task-specific training**.



CLIP: Architecture

CLIP is composed of **two** neural networks: an **image encoder** and a **text encoder**. These two encoders **work together** to enable the model's **cross-modal** understanding of **images** and **text**.



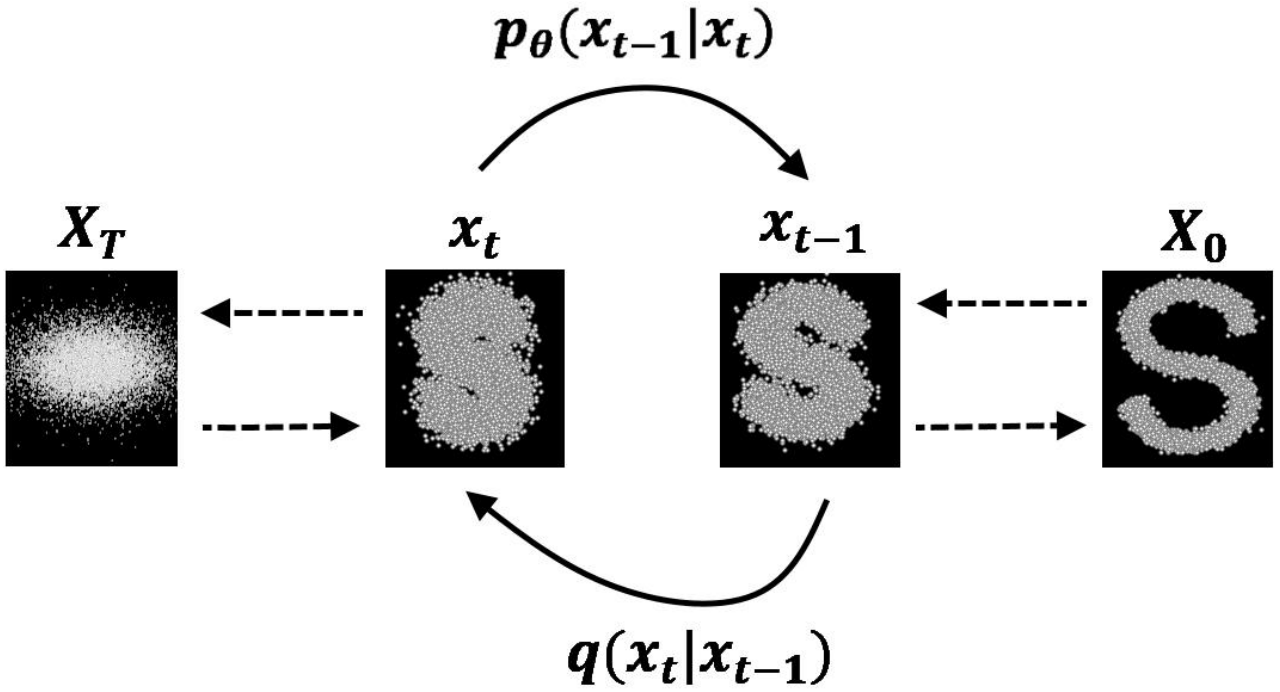
CLIP: Implementation

The **official CLIP** repository from **OpenAI** provides a **robust implementation** of CLIP, along with a **wide variety** of **pretrained models**.

```python
import torch
import clip
from PIL import Image

device = "cuda" if torch.cuda.is_available() else "cpu"
model, preprocess = clip.load("ViT-B/32", device=device)

image = preprocess(Image.open("CLIP.png")).unsqueeze(0).to(device)
text = clip.tokenize(["a diagram", "a dog", "a cat"]).to(device)

with torch.no_grad():
    image_features = model.encode_image(image)
    text_features = model.encode_text(text)

    logits_per_image, logits_per_text = model(image, text)
    probs = logits_per_image.softmax(dim=-1).cpu().numpy()

print("Label probs:", probs)  # prints: [[0.9927937  0.00421068 0.00299572]]
```

Denoising Diffusion Probabilistic Models

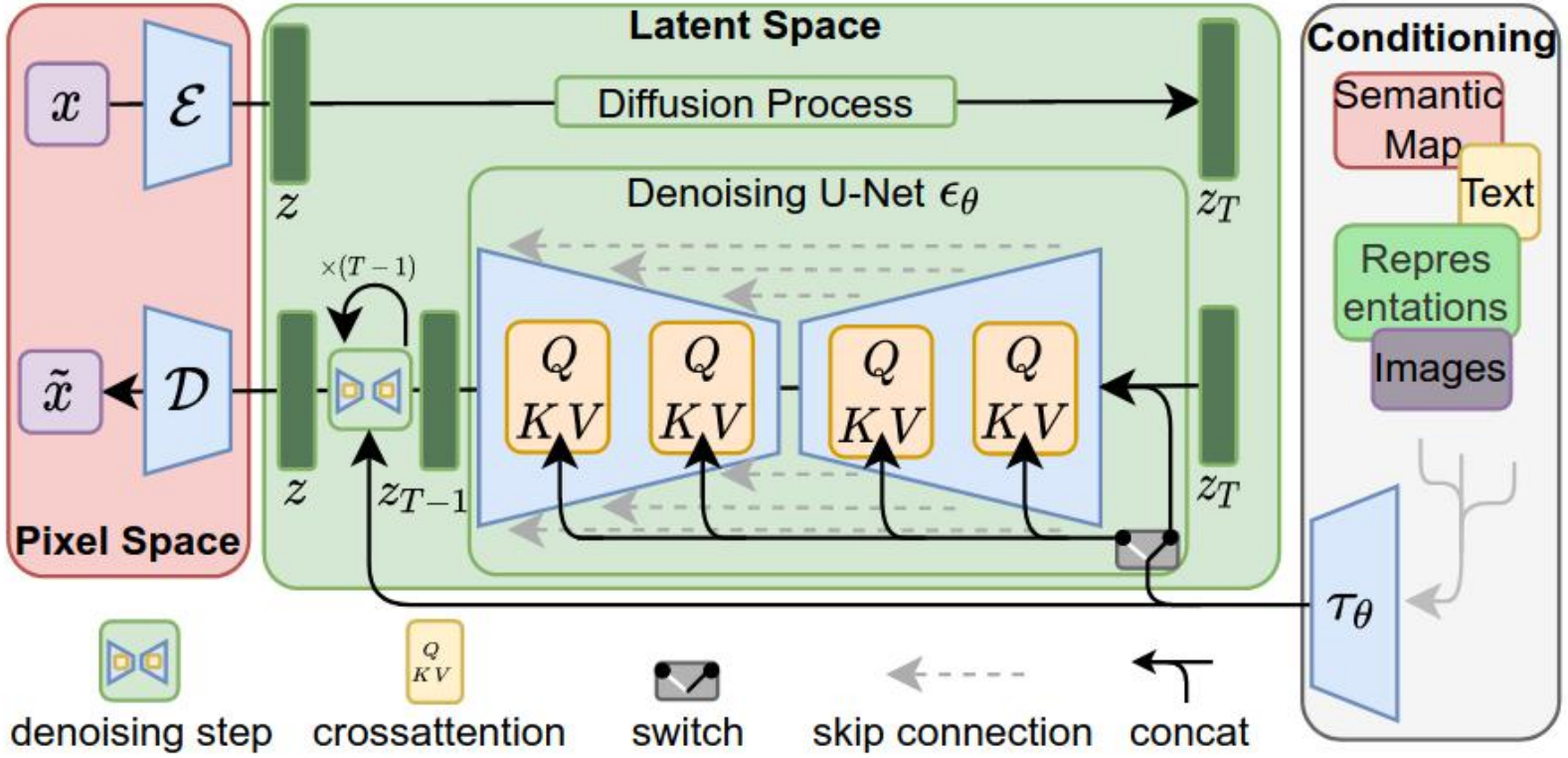
DDPM are a method of **generating** high-quality **samples** from a **probabilistic model**. It involves **iteratively refining** an **initially noisy** or **low-quality sample** by applying a **sequence** of **invertible transformations.** This process **gradually** makes the **sample more coherent** and **realistic.**



Latent Diffusion

Latent Diffusion is a **variant** of the **diffusion models (DDPM)** that **operates** in the **latent space** of a generative model.

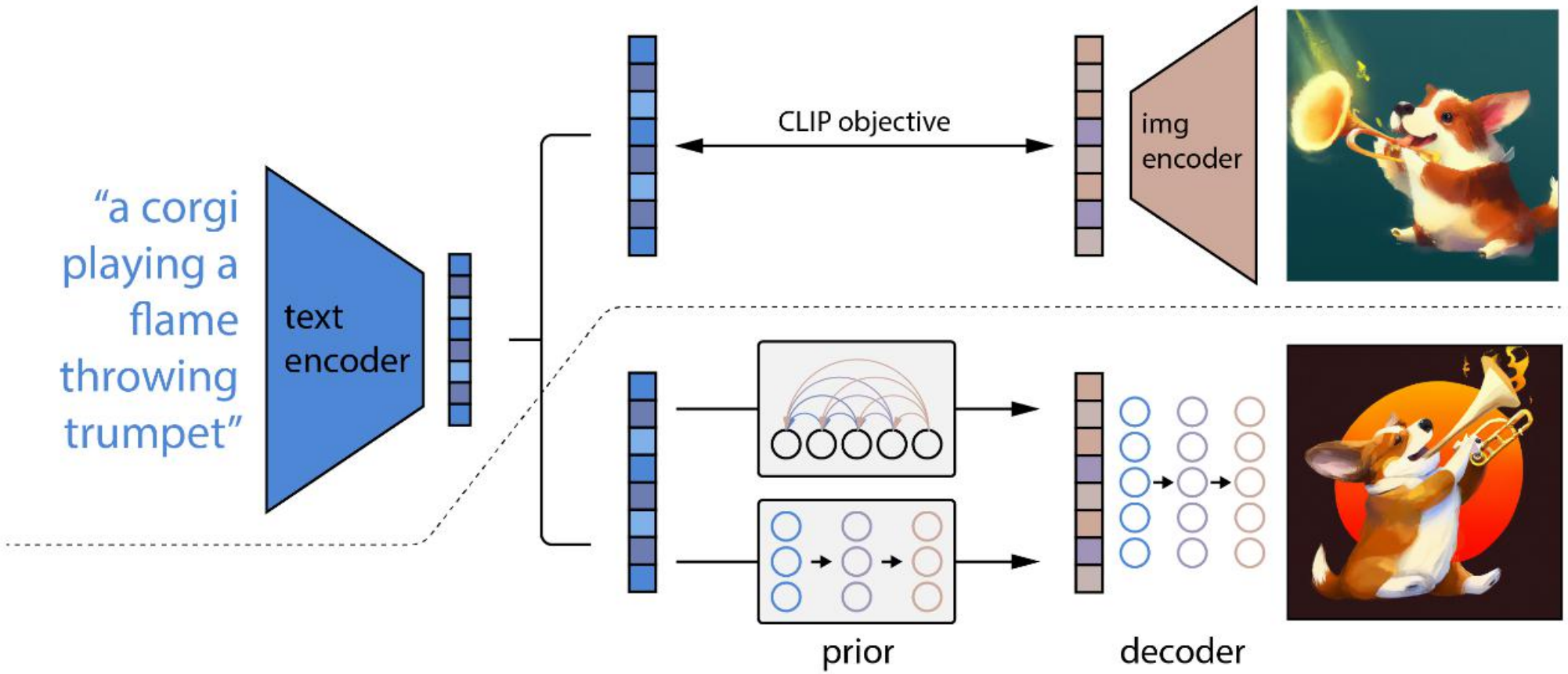
In this approach, a **diffusion process** is **applied** to the **latent** representations of data, allowing for **more efficient generation** of high-quality samples.



DALLE-2

DALL·E 2 is an advanced version of the **DALL·E** model by **OpenAI**, which is a generative model capable of creating images from text descriptions.

It is based on a **pretrained CLIP**, a **diffusion prior** and a **decoder**.



Stable Diffusion: Implementation

Diffusers from **Hugging Face** offers a comprehensive suite of tools for building and utilizing diffusion models, in addition to providing access to thousands of pretrained models.

```
import torch
from diffusers import StableDiffusionPipeline, EulerDiscreteScheduler

model_id = "stabilityai/stable-diffusion-2"

scheduler = EulerDiscreteScheduler.from_pretrained(model_id, subfolder="scheduler")
pipe = StableDiffusionPipeline.from_pretrained(model_id, scheduler=scheduler, torch_dtype=torch.float16)
pipe = pipe.to("cuda")
```

```
prompt = "oil painting of the Leaning Tower of Pisa at night with the moon"
image = pipe(prompt).images[0]
image.save("night.png")
```

Stable Diffusion: Implementation

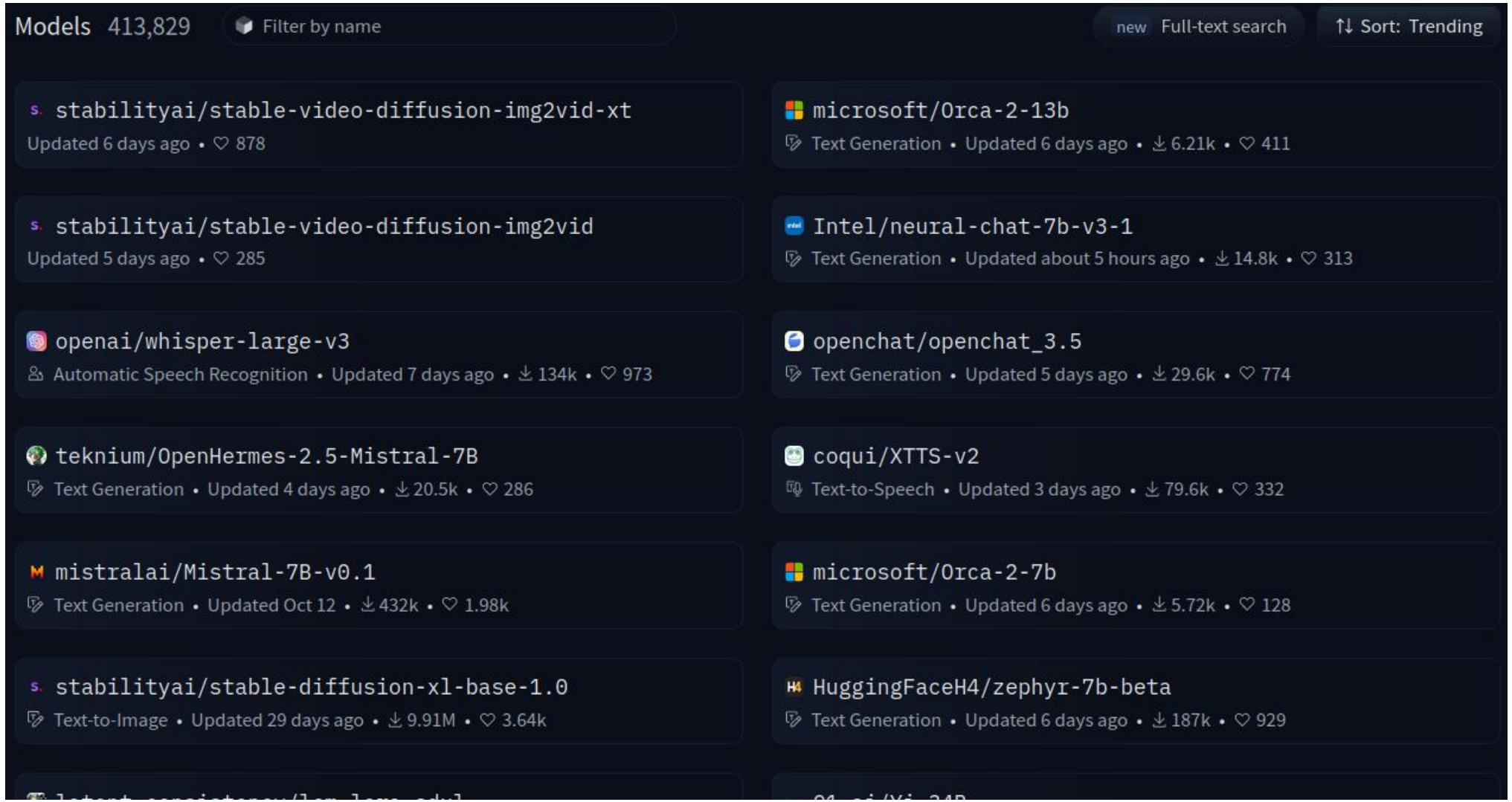
Huggingface

French-American company
(we can do AI in Europe)

Hugging Face is a **leading organization** and **platform** in the field of **natural language processing** (NLP) and **machine learning**. They are known for their contributions to the development and **democratization** of **state-of-the-art NLP models** and **tools**. Hugging Face provides a **wide range** of **open-source libraries** and **pre-trained models.**

- **Transformers**
  - Hugging Face's Transformers library is a comprehensive framework for working with state-of-the-art natural language processing (NLP) models, including BERT, GPT, and many others. It provides easy access to pretrained models and tools for fine-tuning them on specific NLP tasks.
- **Diffusers**
  - Hugging Face's Diffusion library is designed for working with diffusion models, which use diffusion processes to generate and denoise data. It offers tools for creating, training, and using diffusion models, making it easier to apply these advanced generative models.
- **Datasets**
  - Hugging Face's Datasets library is a collection of high-quality, preprocessed datasets for a wide range of NLP and machine learning tasks. It simplifies data loading and preprocessing, making it convenient for researchers and developers to work with diverse datasets.

Huggingface: models



Huggingface: datasets