Create Machine Learning projects with Python (HuggingFace & Gradio)

Dave Appadoo

Presentation for PYMUG

Note: The slides with recording are available at: https://daveappadoo.com/how-to-use-hugging-face-gradio-to-create-simple-machine-learning-ml-projects/

About me

- Software Developer at Accenture
- Consultant Data Scientist
- daveappadoo.com

Agenda

What is Hugging Face 🤗?

- Hugging Face 🤗 model zoo
- How to use the Hugging Face 🤗 library
- Hugging Face 🤗 demo

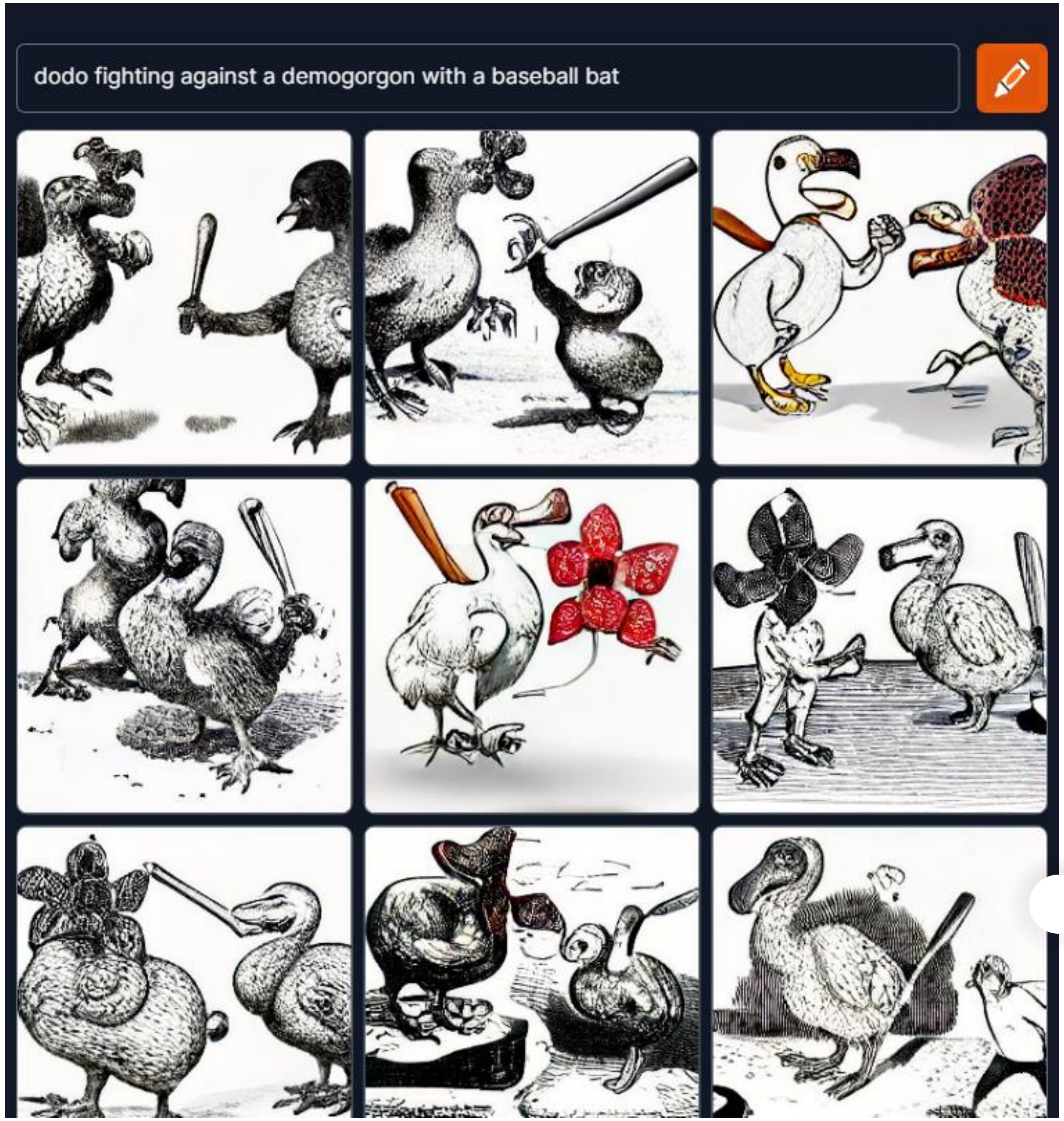
What's Gradio

- Using Gradio
- Gradio Demo

End

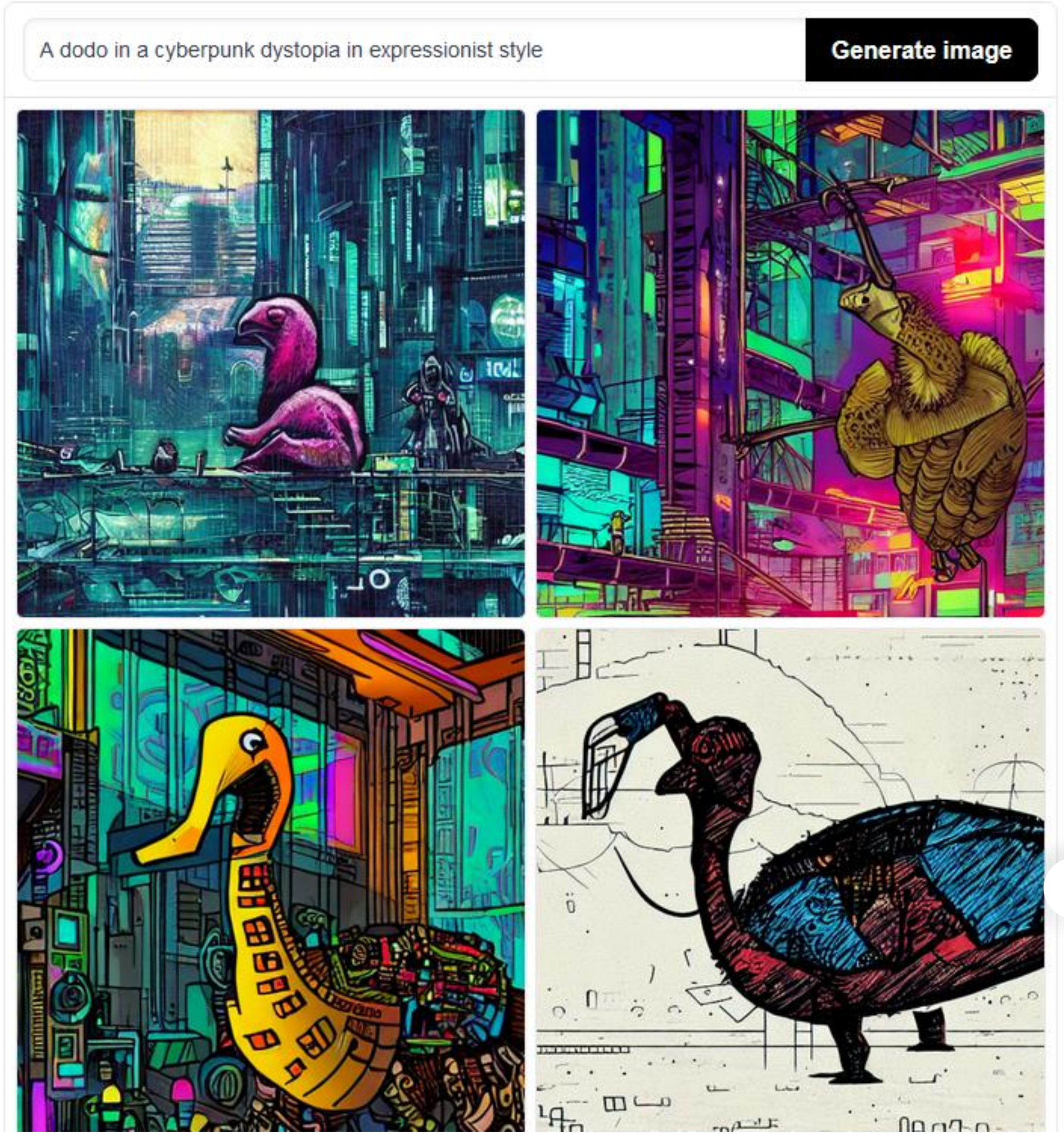
Have you seen this recently?

+ Craiyon (formerly DALL·E mini)
+ an AI model that can draw images from any text prompt!


dodo fighting against a demogorgon with a baseball bat

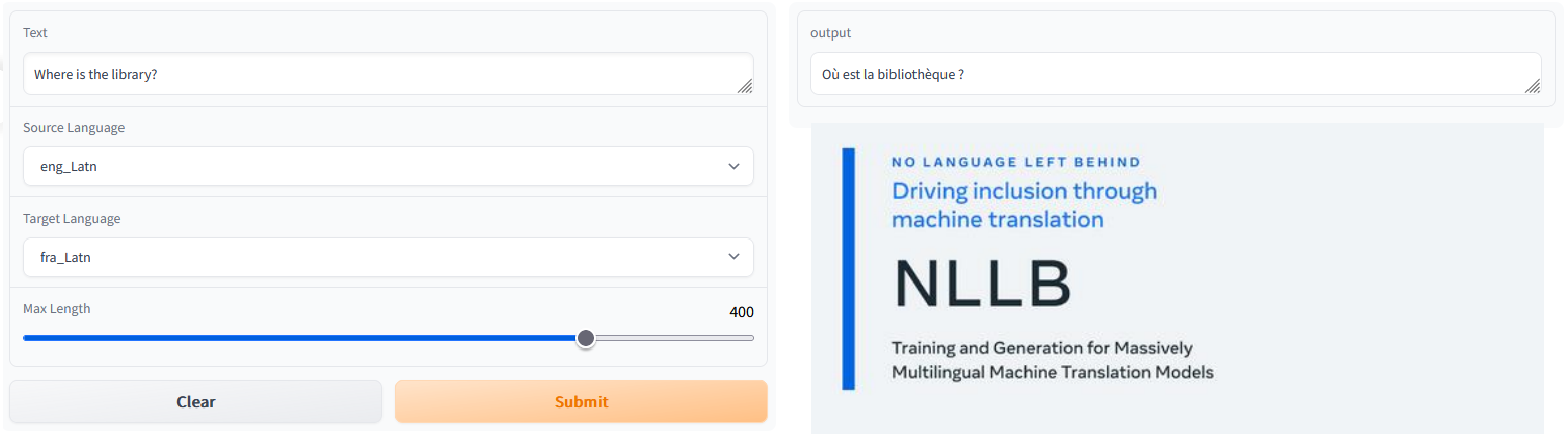
Have you seen this recently?

+ Stable Diffusion
+ state of the art text-to-image model that generates images from text!



Have you seen this recently?

- NLLB TRANSLATION Demo
- state of the art text translation between ~200 languages!


Text

Where is the library?

Source Language

eng_Latn

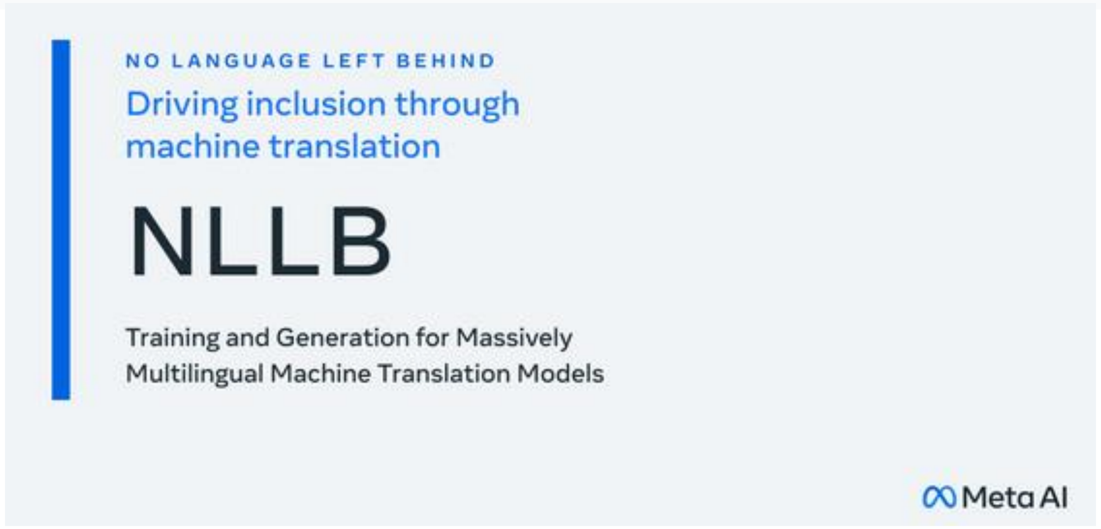
Target Language

fra_Latn

Max Length 400

output

Où est la bibliothèque ?


NO LANGUAGE LEFT BEHIND

Driving inclusion through machine translation

NLLB

Training and Generation for Massively Multilingual Machine Translation Models

Creating simple ML projects back then

Steps	Framework	Language
1. Train model or use pretrained models	TensorFlow PYTORCH	Python
2. Containerize and deploy the model	docker EC2 Flask web development, one drop at a time	Bash
3. Store incoming samples	MySQL	SQL
4. Build an interactive user interface	HTML JS CSS	HTML, JS, CSS

*Taken from https://docs.google.com/presentation/d/1TXw48MjZFvkVur6rE0tUrEECf76SgUrglTjv0Kb0SWQ/edit?usp=sharing

Creating simple ML projects now

Steps	Framework	Language
1. Train model or use pretrained models	TensorFlow PYTORCH	Python
2. Containerize and deploy the model	gradio	Python
3. Store incoming samples		Python
4. Build an interactive user interface		Python

*Taken from https://docs.google.com/presentation/d/1TXw48MjZFvkVur6rE0tUrEECf76SgUrglTjv0Kb0SWQ/edit?usp=sharing

Step 1 in creating a ML project

+ Train a model or use a pre-trained model
+ Pre-trained models come in either TensorFlow or PyTorch.

Step 1 in creating a ML project

- Train a model or use a pre-trained model
- Pre-trained models come in either TensorFlow or PyTorch.

- Murphy’s law guaranteed:
  - that PyTorch users would only find TensorFlow models, and vice versa.

Step 1 in creating a ML project

- Train a model or use a pre-trained model.
- Pre-trained models come in either TensorFlow or PyTorch.

- Murphy's law guaranteed:
  - that PyTorch users would only find TensorFlow models, and vice versa.

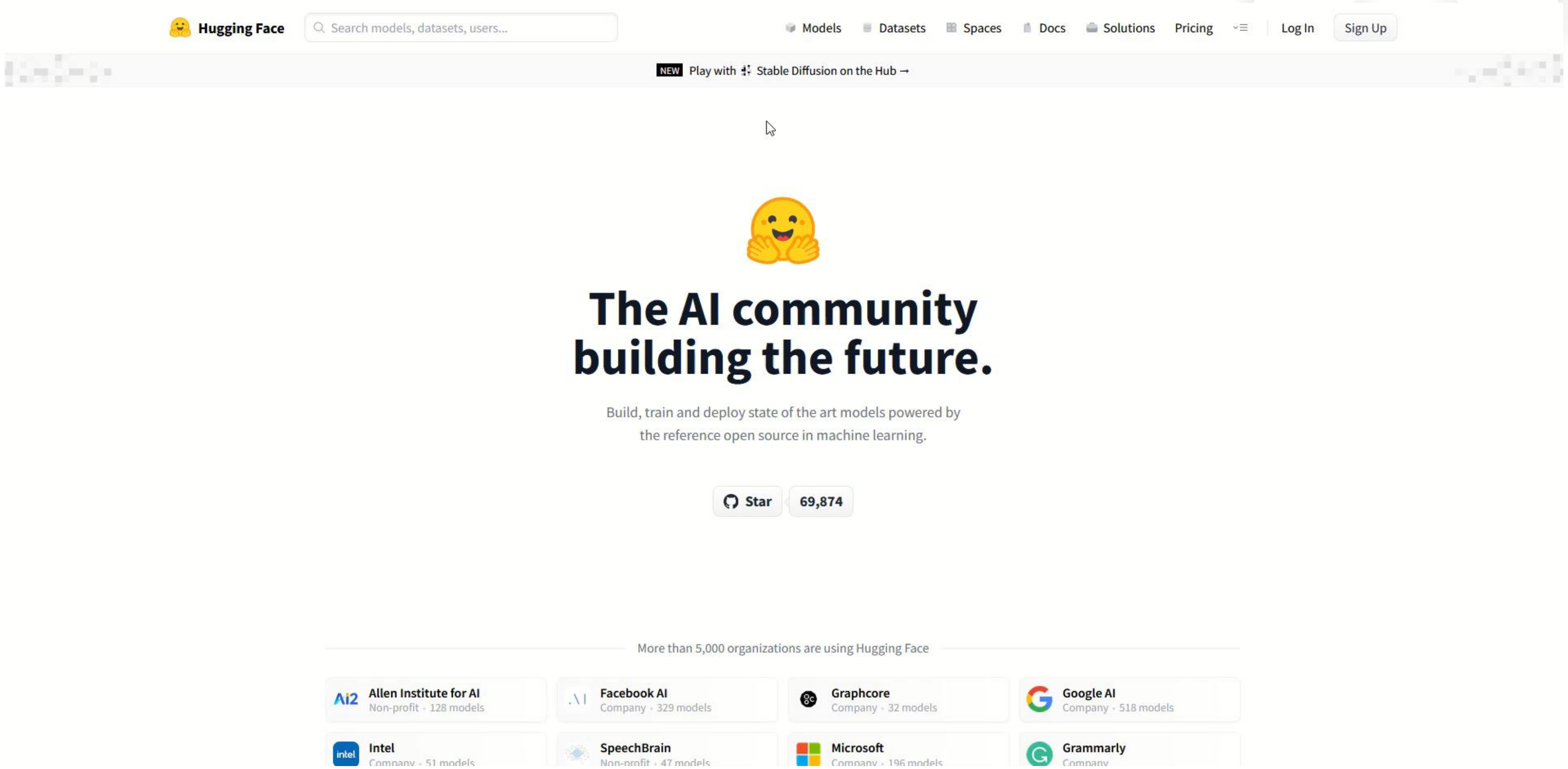
- Hugging Face 🤗 to the rescue with its model hub!

What is Hugging Face🤗 ?

- Was known mostly for its Transformers library
  - it’s open source, it supports both TensorFlow and PyTorch
- Now **a community and data science platform**
  - enable users to build, train and deploy ML models based on open-source code.

- Think of what GitHub is to software development.
- Hugging Face🤗 is like that but to ML.

The Hugging Face 🤗 model zoo



Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Docs | Solutions | Pricing | Log In | Sign Up

NEW Play with Stable Diffusion on the Hub →

The AI community building the future.

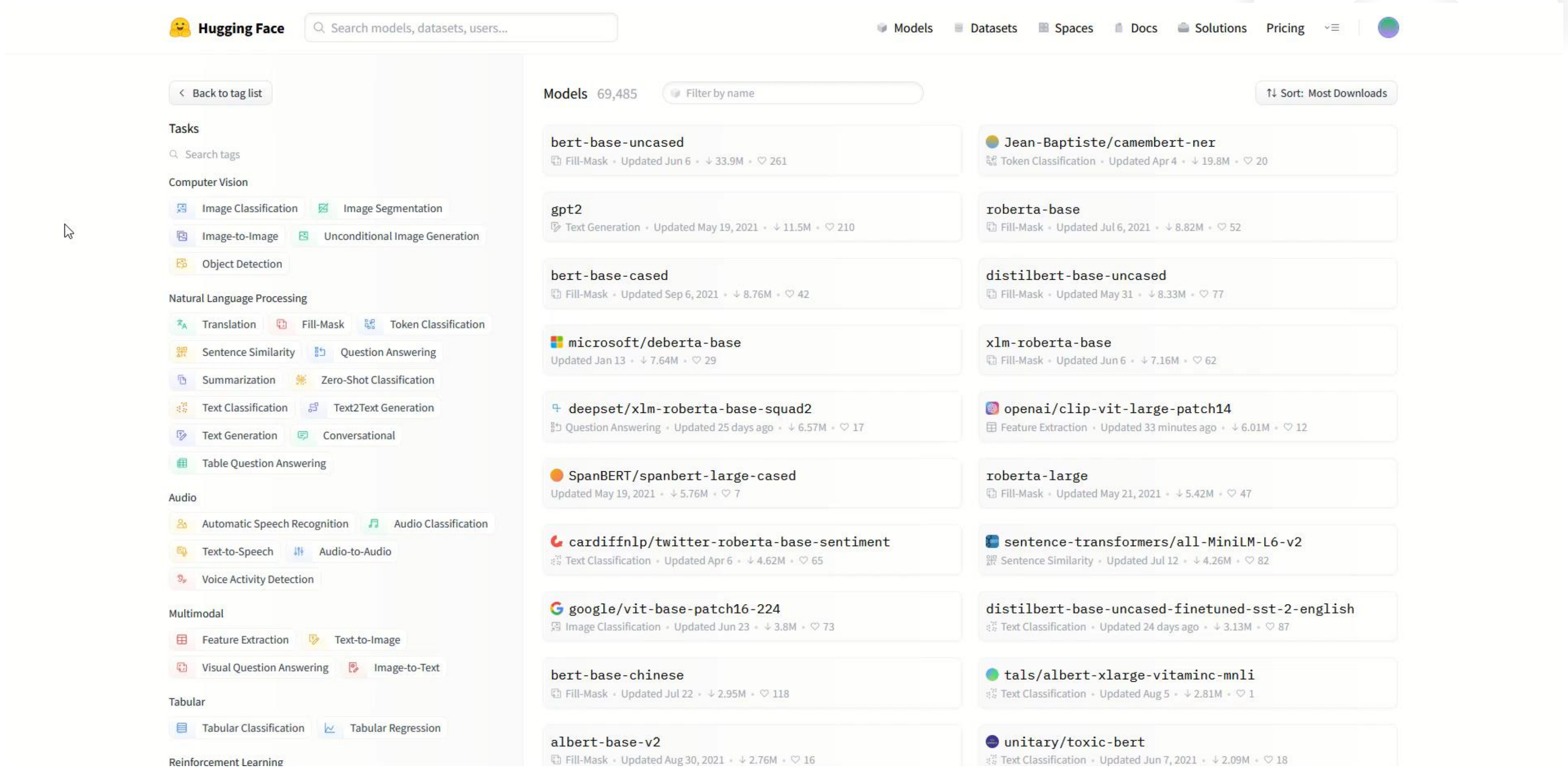
Build, train and deploy state of the art models powered by the reference open source in machine learning.

Star 69,874

More than 5,000 organizations are using Hugging Face

Allen Institute for AI Non-profit · 128 models	Facebook AI Company · 329 models	Graphcore Company · 32 models	Google AI Company · 518 models
Intel Company · 51 models	SpeechBrain Non-profit · 47 models	Microsoft Company · 196 models	Grammarly Company

Trying out a 🤗 Object Detection model



Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Docs | Solutions | Pricing

< Back to tag list

Tasks

Search tags

Computer Vision
- Image Classification
- Image Segmentation
- Image-to-Image
- Unconditional Image Generation
- Object Detection

Natural Language Processing
- Translation
- Fill-Mask
- Token Classification
- Sentence Similarity
- Question Answering
- Summarization
- Zero-Shot Classification
- Text Classification
- Text2Text Generation
- Text Generation
- Conversational
- Table Question Answering

Audio
- Automatic Speech Recognition
- Audio Classification
- Text-to-Speech
- Audio-to-Audio
- Voice Activity Detection

Multimodal
- Feature Extraction
- Text-to-Image
- Visual Question Answering
- Image-to-Text

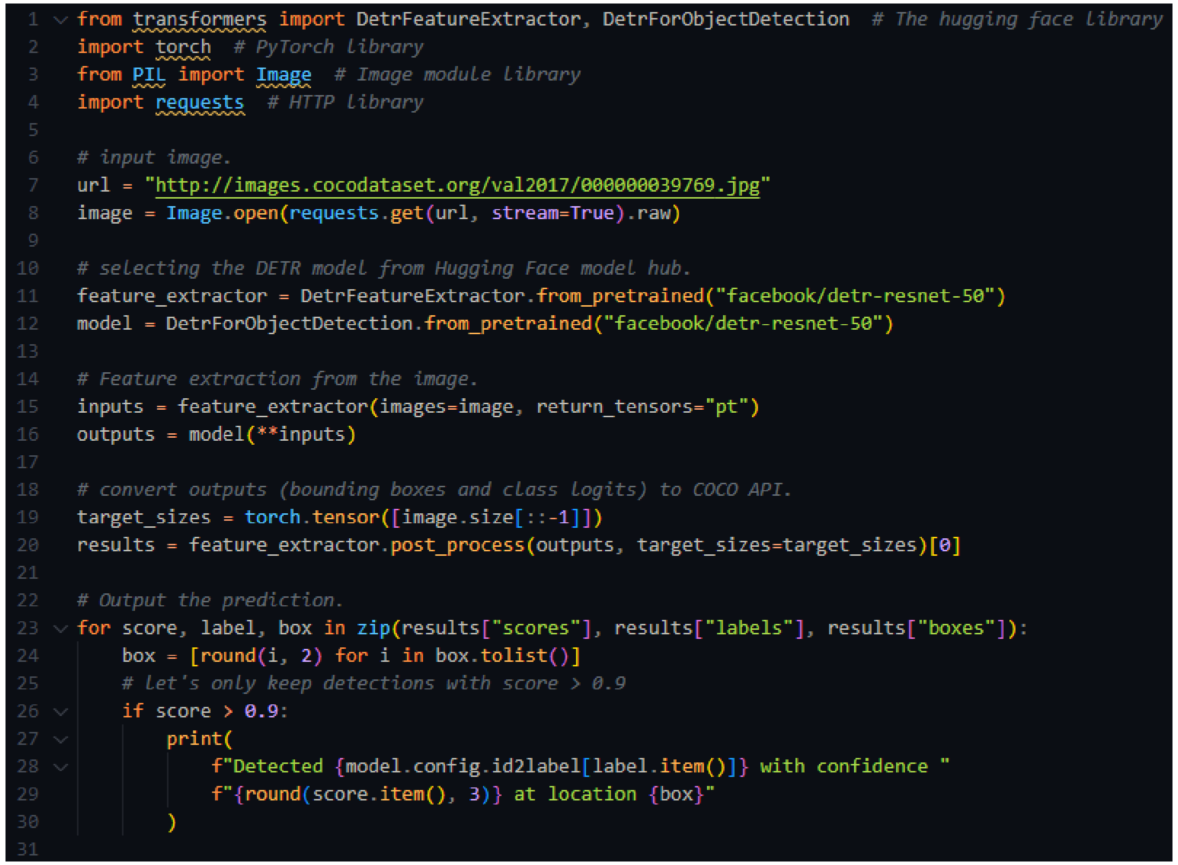
Tabular
- Tabular Classification
- Tabular Regression

Reinforcement Learning

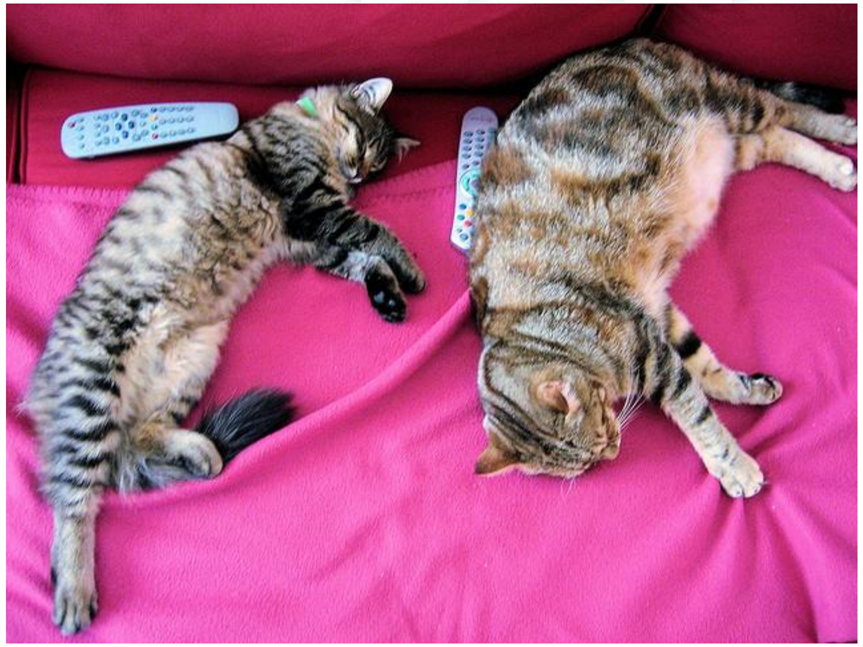
Models 69,485 | Filter by name | Sort: Most Downloads

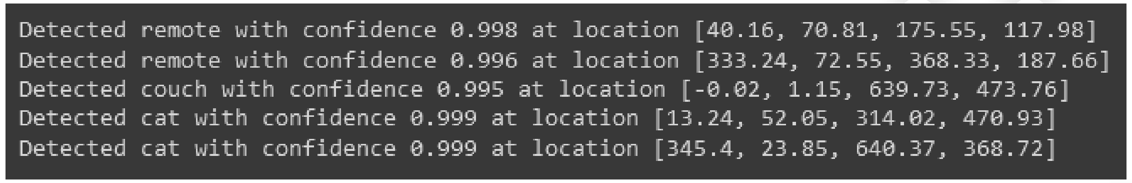
- bert-base-uncased — Fill-Mask • Updated Jun 6 • ↓33.9M • ♡ 261
- Jean-Baptiste/camembert-ner — Token Classification • Updated Apr 4 • ↓19.8M • ♡ 20
- gpt2 — Text Generation • Updated May 19, 2021 • ↓11.5M • ♡ 210
- roberta-base — Fill-Mask • Updated Jul 6, 2021 • ↓8.82M • ♡ 52
- bert-base-cased — Fill-Mask • Updated Sep 6, 2021 • ↓8.76M • ♡ 42
- distilbert-base-uncased — Fill-Mask • Updated May 31 • ↓8.33M • ♡ 77
- microsoft/deberta-base — Updated Jan 13 • ↓7.64M • ♡ 29
- xlm-roberta-base — Fill-Mask • Updated Jun 6 • ↓7.16M • ♡ 62
- deepset/xlm-roberta-base-squad2 — Question Answering • Updated 25 days ago • ↓6.57M • ♡ 17
- openai/clip-vit-large-patch14 — Feature Extraction • Updated 33 minutes ago • ↓6.01M • ♡ 12
- SpanBERT/spanbert-large-cased — Updated May 19, 2021 • ↓5.76M • ♡ 7
- roberta-large — Fill-Mask • Updated May 21, 2021 • ↓5.42M • ♡ 47
- cardiffnlp/twitter-roberta-base-sentiment — Text Classification • Updated Apr 6 • ↓4.62M • ♡ 65
- sentence-transformers/all-MiniLM-L6-v2 — Sentence Similarity • Updated Jul 12 • ↓4.26M • ♡ 82
- google/vit-base-patch16-224 — Image Classification • Updated Jun 23 • ↓3.8M • ♡ 73
- distilbert-base-uncased-finetuned-sst-2-english — Text Classification • Updated 24 days ago • ↓3.13M • ♡ 87
- bert-base-chinese — Fill-Mask • Updated Jul 22 • ↓2.95M • ♡ 118
- tals/albert-xlarge-vitaminc-mnli — Text Classification • Updated Aug 5 • ↓2.81M • ♡ 1
- albert-base-v2 — Fill-Mask • Updated Aug 30, 2021 • ↓2.76M • ♡ 16
- unitary/toxic-bert — Text Classification • Updated Jun 7, 2021 • ↓2.09M • ♡ 18

Creating a simple UI for the model



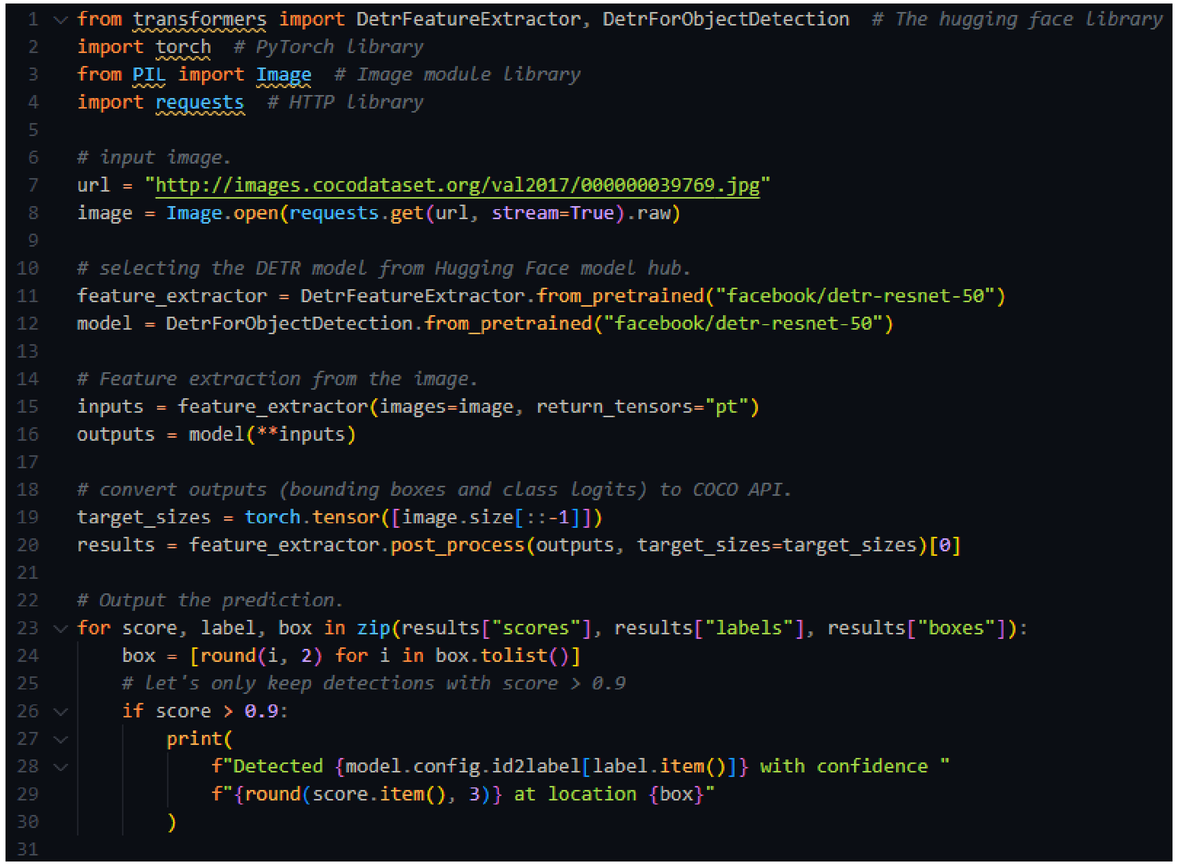
```python
from transformers import DetrFeatureExtractor, DetrForObjectDetection  # The hugging face library
import torch  # PyTorch library
from PIL import Image  # Image module library
import requests  # HTTP library

# input image.
url = "http://images.cocodataset.org/val2017/000000039769.jpg"
image = Image.open(requests.get(url, stream=True).raw)

# selecting the DETR model from Hugging Face model hub.
feature_extractor = DetrFeatureExtractor.from_pretrained("facebook/detr-resnet-50")
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")

# Feature extraction from the image.
inputs = feature_extractor(images=image, return_tensors="pt")
outputs = model(**inputs)

# convert outputs (bounding boxes and class logits) to COCO API.
target_sizes = torch.tensor([image.size[::-1]])
results = feature_extractor.post_process(outputs, target_sizes=target_sizes)[0]

# Output the prediction.
for score, label, box in zip(results["scores"], results["labels"], results["boxes"]):
    box = [round(i, 2) for i in box.tolist()]
    # Let's only keep detections with score > 0.9
    if score > 0.9:
        print(
            f"Detected {model.config.id2label[label.item()]} with confidence "
            f"{round(score.item(), 3)} at location {box}"
        )
```





```
Detected remote with confidence 0.998 at location [40.16, 70.81, 175.55, 117.98]
Detected remote with confidence 0.996 at location [333.24, 72.55, 368.33, 187.66]
Detected couch with confidence 0.995 at location [-0.02, 1.15, 639.73, 473.76]
Detected cat with confidence 0.999 at location [13.24, 52.05, 314.02, 470.93]
Detected cat with confidence 0.999 at location [345.4, 23.85, 640.37, 368.72]
```

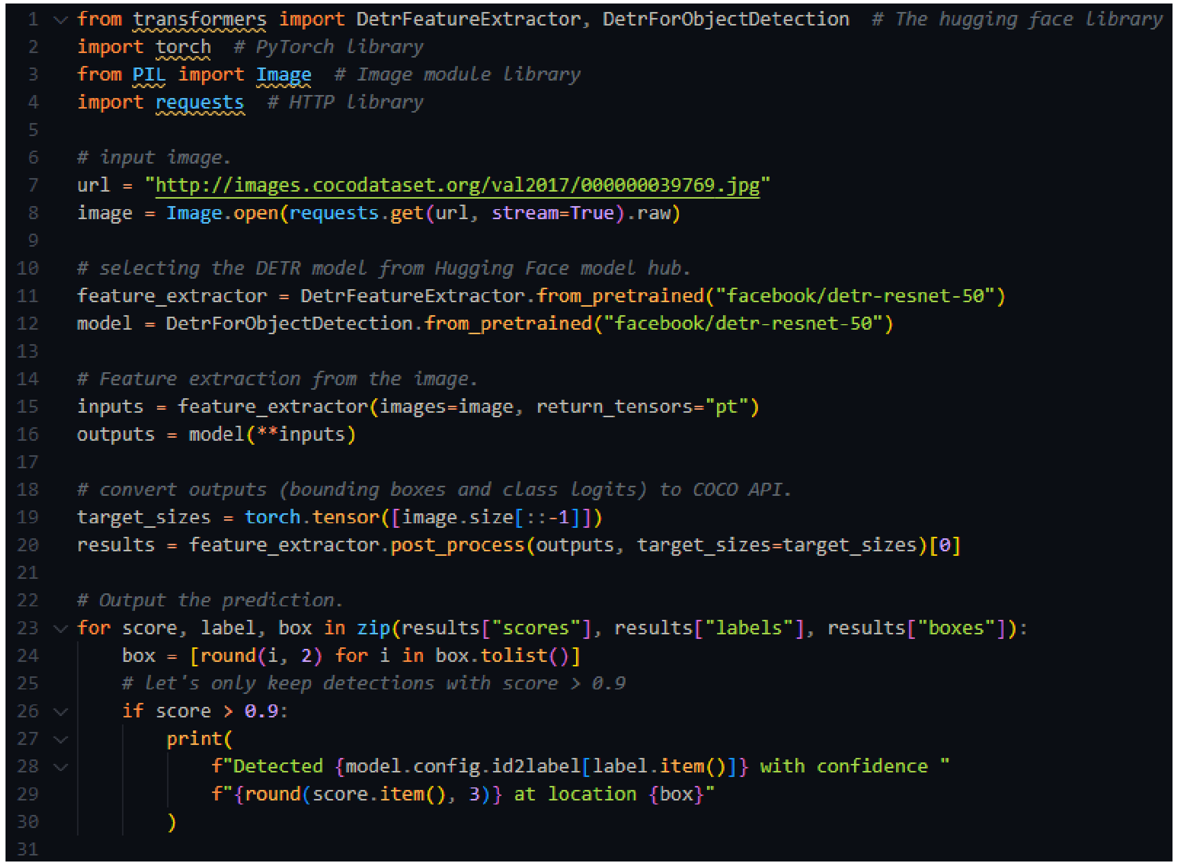
Creating a simple UI for the model



```python
from transformers import DetrFeatureExtractor, DetrForObjectDetection  # The hugging face library
import torch  # PyTorch library
from PIL import Image  # Image module library
import requests  # HTTP library

# input image.
url = "http://images.cocodataset.org/val2017/000000039769.jpg"
image = Image.open(requests.get(url, stream=True).raw)

# selecting the DETR model from Hugging Face model hub.
feature_extractor = DetrFeatureExtractor.from_pretrained("facebook/detr-resnet-50")
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")

# Feature extraction from the image.
inputs = feature_extractor(images=image, return_tensors="pt")
outputs = model(**inputs)

# convert outputs (bounding boxes and class logits) to COCO API.
target_sizes = torch.tensor([image.size[::-1]])
results = feature_extractor.post_process(outputs, target_sizes=target_sizes)[0]

# Output the prediction.
for score, label, box in zip(results["scores"], results["labels"], results["boxes"]):
    box = [round(i, 2) for i in box.tolist()]
    # Let's only keep detections with score > 0.9
    if score > 0.9:
        print(
            f"Detected {model.config.id2label[label.item()]} with confidence "
            f"{round(score.item(), 3)} at location {box}"
        )

```

+ Now we have the model
+ How to serve it to a user?
+ The user should not be editing source code to change image file.

Creating a simple UI for the model



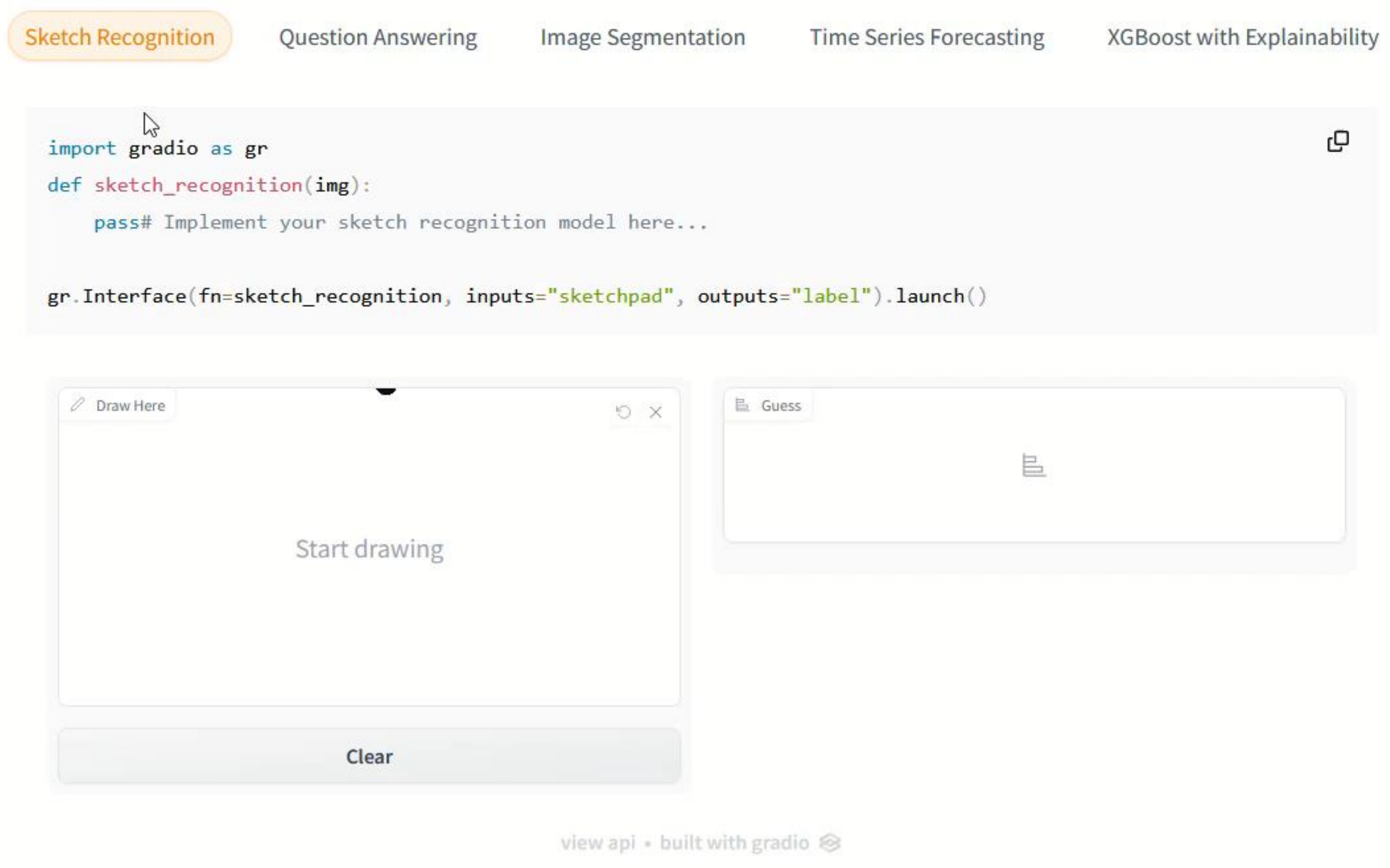
```python
from transformers import DetrFeatureExtractor, DetrForObjectDetection  # The hugging face library
import torch  # PyTorch library
from PIL import Image  # Image module library
import requests  # HTTP library

# input image.
url = "http://images.cocodataset.org/val2017/000000039769.jpg"
image = Image.open(requests.get(url, stream=True).raw)

# selecting the DETR model from Hugging Face model hub.
feature_extractor = DetrFeatureExtractor.from_pretrained("facebook/detr-resnet-50")
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")

# Feature extraction from the image.
inputs = feature_extractor(images=image, return_tensors="pt")
outputs = model(**inputs)

# convert outputs (bounding boxes and class logits) to COCO API.
target_sizes = torch.tensor([image.size[::-1]])
results = feature_extractor.post_process(outputs, target_sizes=target_sizes)[0]

# Output the prediction.
for score, label, box in zip(results["scores"], results["labels"], results["boxes"]):
    box = [round(i, 2) for i in box.tolist()]
    # Let's only keep detections with score > 0.9
    if score > 0.9:
        print(
            f"Detected {model.config.id2label[label.item()]} with confidence "
            f"{round(score.item(), 3)} at location {box}"
        )

```

- Now we have the model
- How to serve it to a user?
- The user should not be editing source code to change image file.
- Gradio as a solution!

What is Gradio?

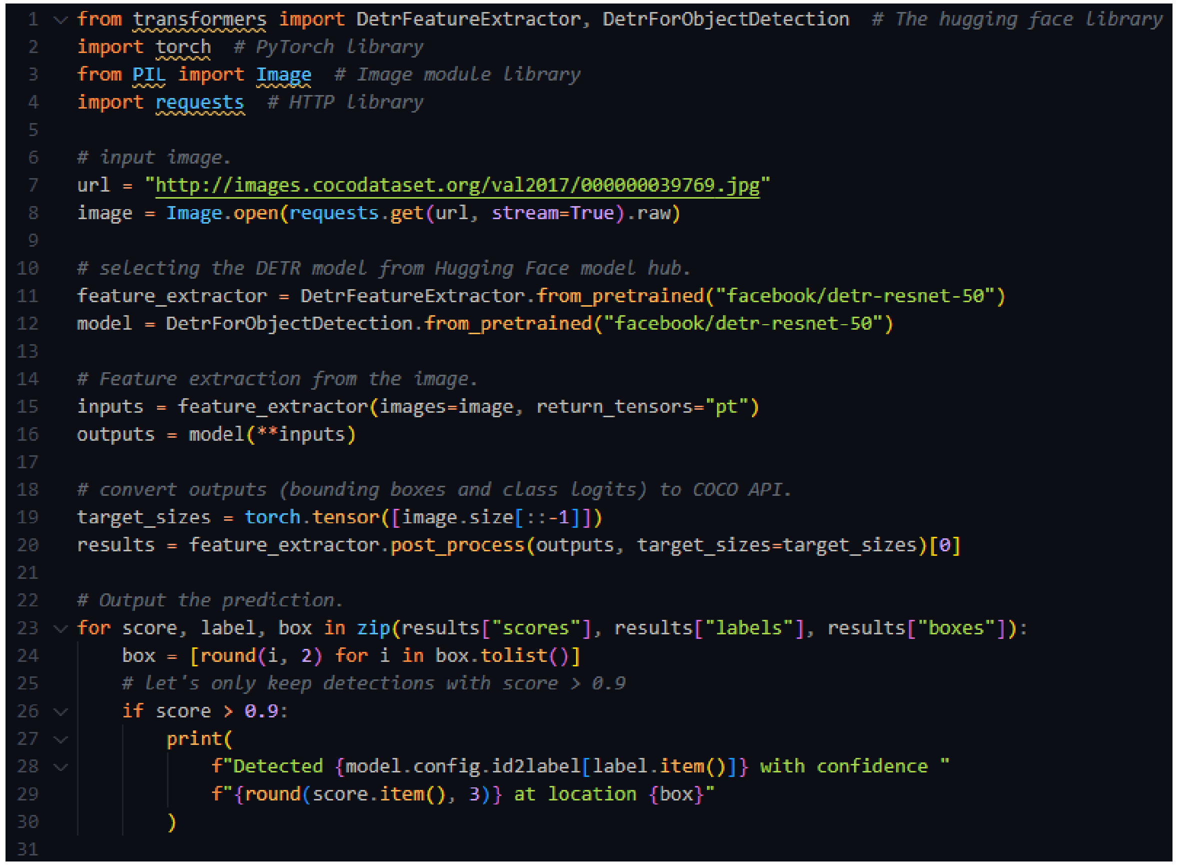
+ Gradio is the fastest way to demo a machine learning model.
+ It provides a friendly web interface.
+ Gradio can be installed with pip.
+ Gradio can be embedded:
    - in Python notebooks or
    - presented as a webpage.
+ It can be permanently hosted on Hugging Face.

Some Gradio Examples

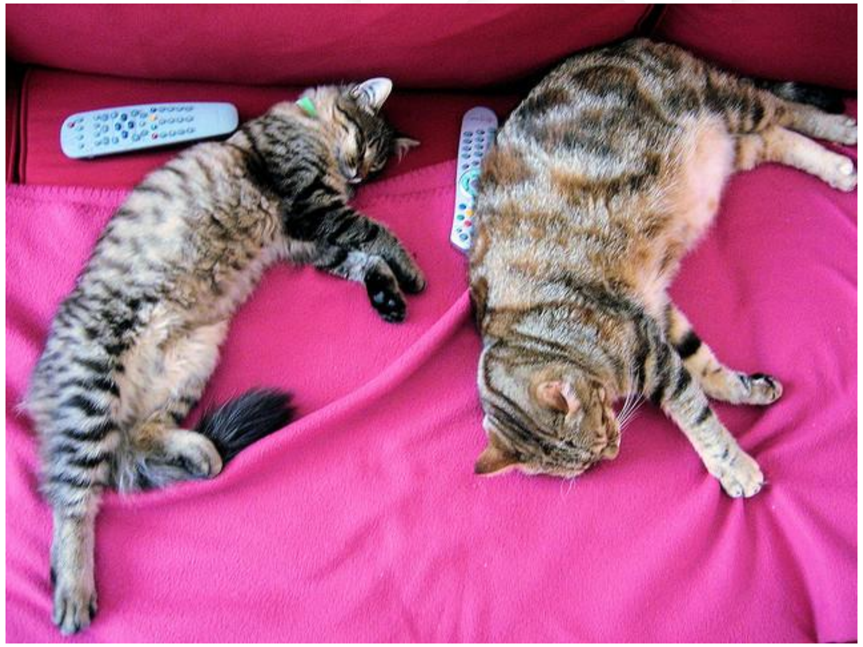


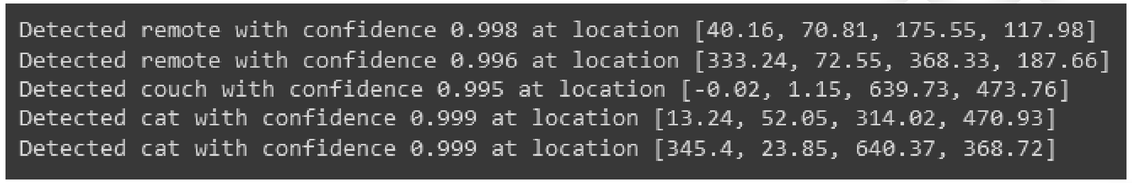
```
import gradio as gr
def sketch_recognition(img):
    pass# Implement your sketch recognition model here...

gr.Interface(fn=sketch_recognition, inputs="sketchpad", outputs="label").launch()
```

Let’s create a simple UI for this model



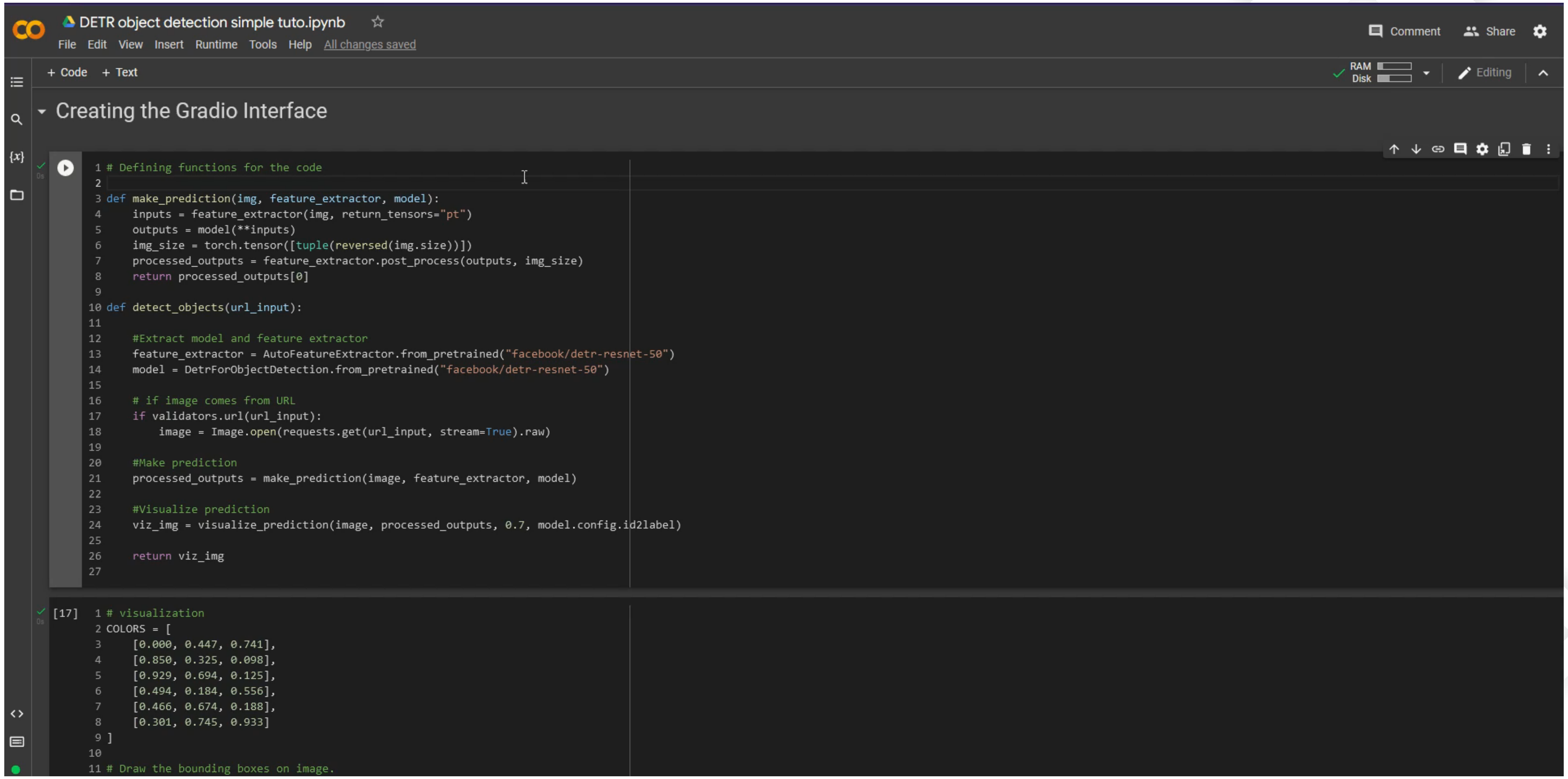
```python
from transformers import DetrFeatureExtractor, DetrForObjectDetection  # The hugging face library
import torch  # PyTorch library
from PIL import Image  # Image module library
import requests  # HTTP library

# input image.
url = "http://images.cocodataset.org/val2017/000000039769.jpg"
image = Image.open(requests.get(url, stream=True).raw)

# selecting the DETR model from Hugging Face model hub.
feature_extractor = DetrFeatureExtractor.from_pretrained("facebook/detr-resnet-50")
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")

# Feature extraction from the image.
inputs = feature_extractor(images=image, return_tensors="pt")
outputs = model(**inputs)

# convert outputs (bounding boxes and class logits) to COCO API.
target_sizes = torch.tensor([image.size[::-1]])
results = feature_extractor.post_process(outputs, target_sizes=target_sizes)[0]

# Output the prediction.
for score, label, box in zip(results["scores"], results["labels"], results["boxes"]):
    box = [round(i, 2) for i in box.tolist()]
    # Let's only keep detections with score > 0.9
    if score > 0.9:
        print(
            f"Detected {model.config.id2label[label.item()]} with confidence "
            f"{round(score.item(), 3)} at location {box}"
        )

```





```
Detected remote with confidence 0.998 at location [40.16, 70.81, 175.55, 117.98]
Detected remote with confidence 0.996 at location [333.24, 72.55, 368.33, 187.66]
Detected couch with confidence 0.995 at location [-0.02, 1.15, 639.73, 473.76]
Detected cat with confidence 0.999 at location [13.24, 52.05, 314.02, 470.93]
Detected cat with confidence 0.999 at location [345.4, 23.85, 640.37, 368.72]
```

Let’s create a simple UI for this model


DETR object detection simple tuto.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing

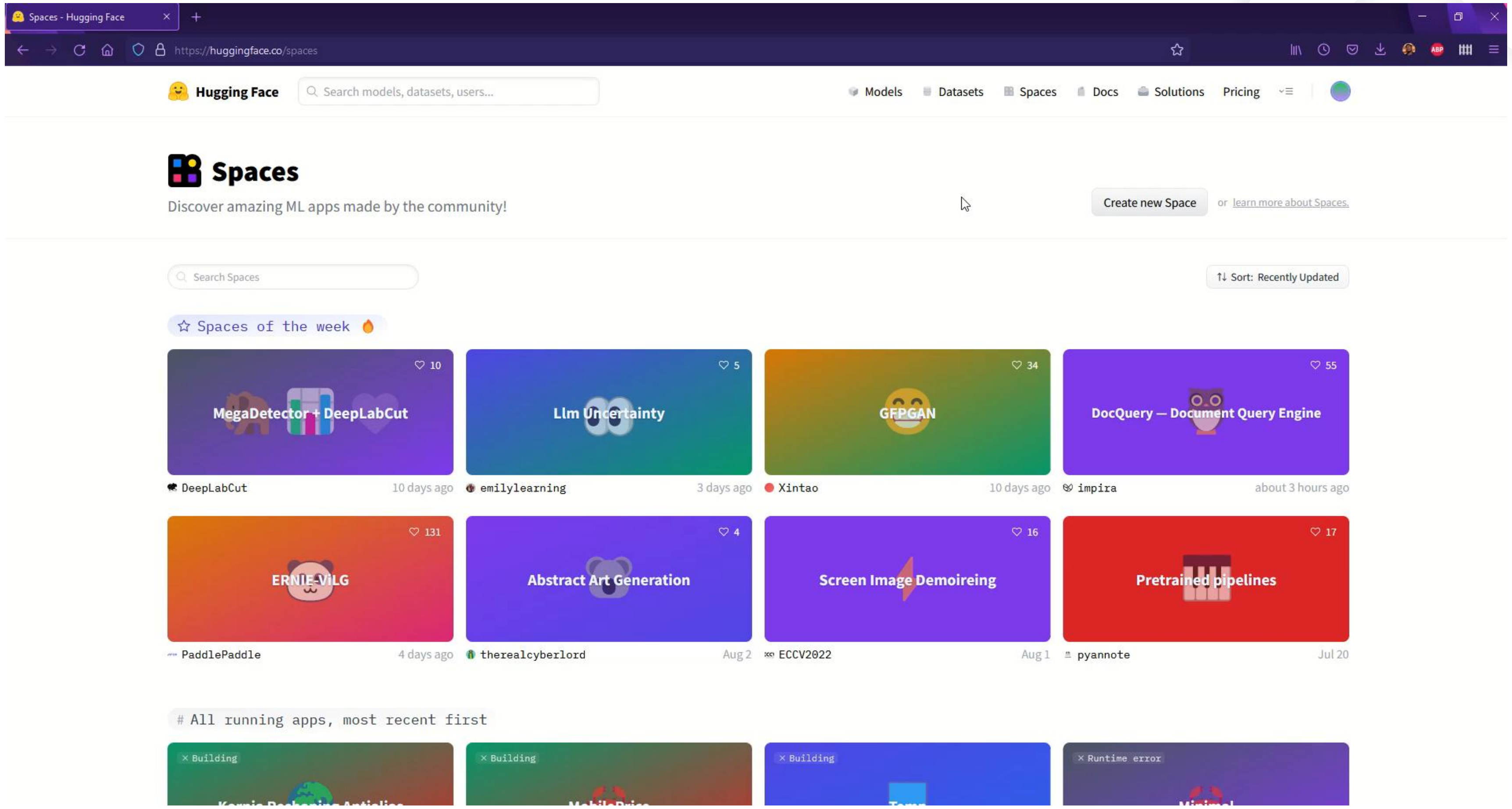
Creating the Gradio Interface

```python
# Defining functions for the code

def make_prediction(img, feature_extractor, model):
    inputs = feature_extractor(img, return_tensors="pt")
    outputs = model(**inputs)
    img_size = torch.tensor([tuple(reversed(img.size))])
    processed_outputs = feature_extractor.post_process(outputs, img_size)
    return processed_outputs[0]

def detect_objects(url_input):

    #Extract model and feature extractor
    feature_extractor = AutoFeatureExtractor.from_pretrained("facebook/detr-resnet-50")
    model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")

    # if image comes from URL
    if validators.url(url_input):
        image = Image.open(requests.get(url_input, stream=True).raw)

    #Make prediction
    processed_outputs = make_prediction(image, feature_extractor, model)

    #Visualize prediction
    viz_img = visualize_prediction(image, processed_outputs, 0.7, model.config.id2label)

    return viz_img

```

[17]

```python
# visualization
COLORS = [
    [0.000, 0.447, 0.741],
    [0.850, 0.325, 0.098],
    [0.929, 0.694, 0.125],
    [0.494, 0.184, 0.556],
    [0.466, 0.674, 0.188],
    [0.301, 0.745, 0.933]
]

# Draw the bounding boxes on image.
```

Hosting our Object Detection app on Hugging Space 🤗



Spaces

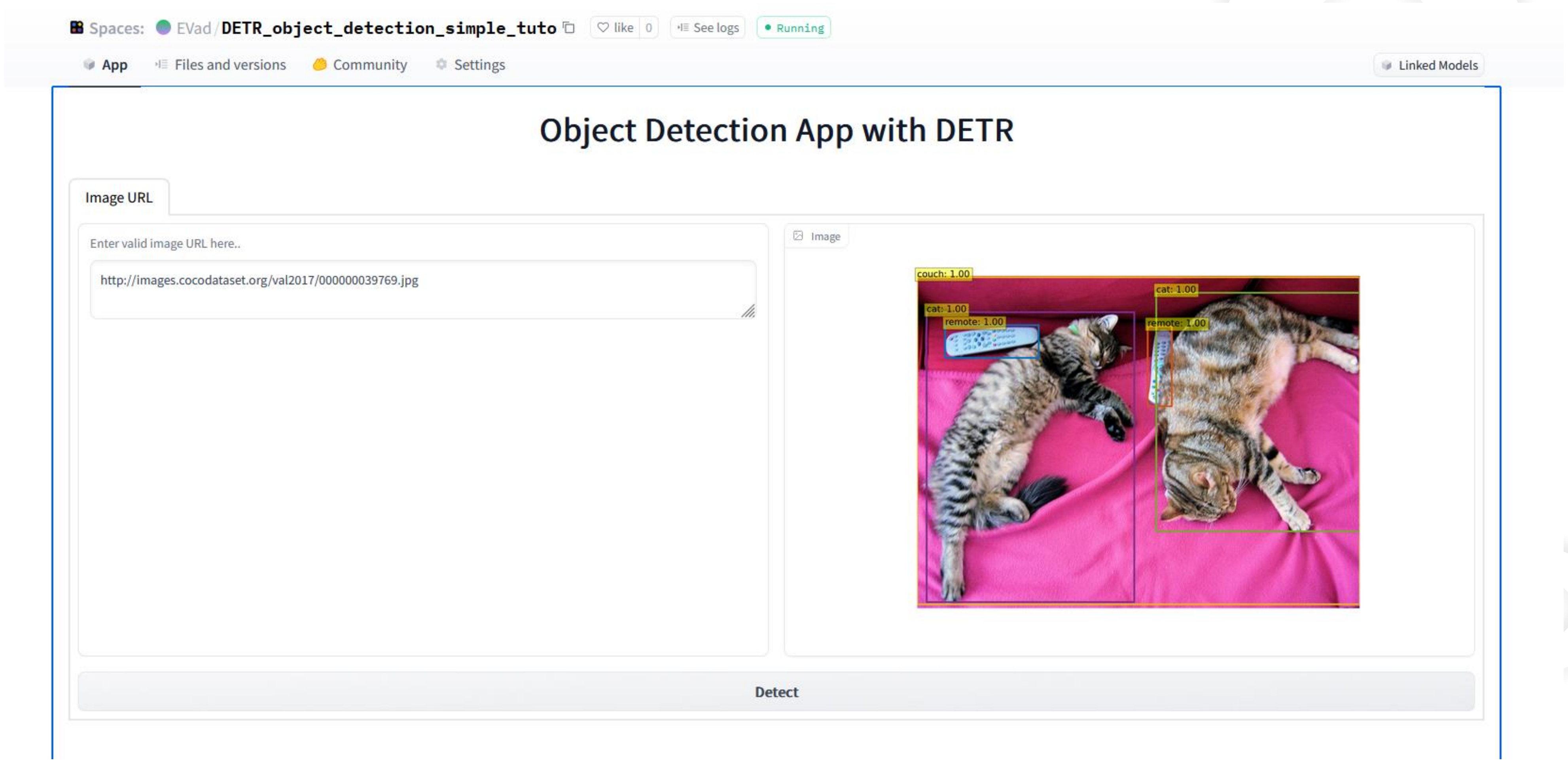
Discover amazing ML apps made by the community!

Spaces of the week 🔥

Space	Likes	Owner	Updated
MegaDetector + DeepLabCut	10	DeepLabCut	10 days ago
Llm Uncertainty	5	emilylearning	3 days ago
GFPGAN	34	Xintao	10 days ago
DocQuery — Document Query Engine	55	impira	about 3 hours ago
ERNIE-ViLG	131	PaddlePaddle	4 days ago
Abstract Art Generation	4	therealcyberlord	Aug 2
Screen Image Demoireing	16	ECCV2022	Aug 1
Pretrained pipelines	17	pyannote	Jul 20

All running apps, most recent first

Hosting our Object Detection app on Hugging Space 🤗



Spaces: EVad / DETR_object_detection_simple_tuto

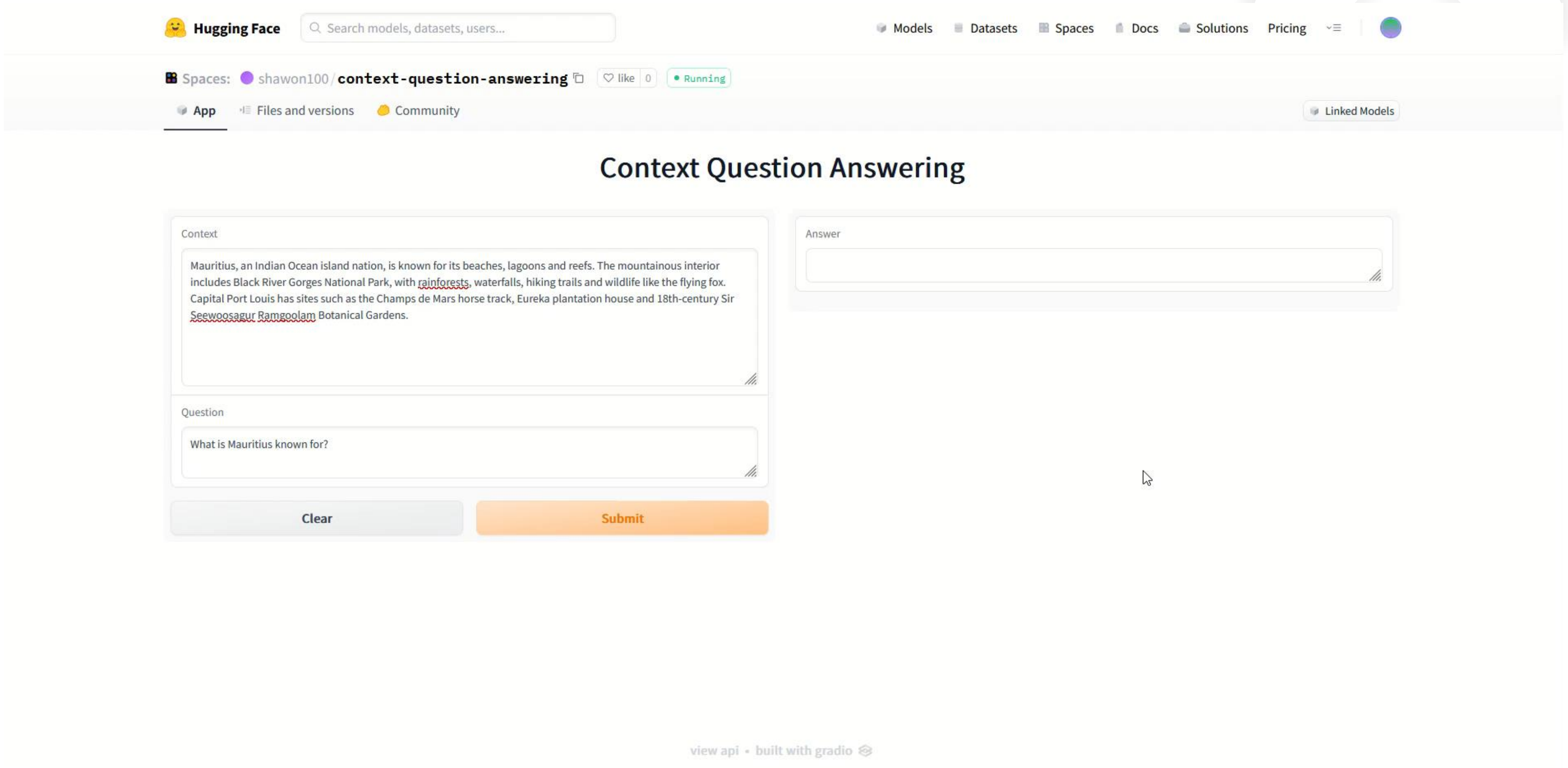
Object Detection App with DETR

Image URL

Enter valid image URL here..

http://images.cocodataset.org/val2017/000000039769.jpg

What else does 🤗 Spaces offer?



Context Question Answering

Context

Mauritius, an Indian Ocean island nation, is known for its beaches, lagoons and reefs. The mountainous interior includes Black River Gorges National Park, with rainforests, waterfalls, hiking trails and wildlife like the flying fox. Capital Port Louis has sites such as the Champs de Mars horse track, Eureka plantation house and 18th-century Sir Seewoosagur Ramgoolam Botanical Gardens.

Question

What is Mauritius known for?

Answer

Resources

- https://github.com/azezezaaa/DETR-object-detection-tutorial
- https://huggingface.co
- https://gradio.app/
- https://huggingface.co/spaces/anaxagoras7/gauravgs-text-summarizer
- https://huggingface.co/spaces/Narrativaai/NLLB-Translator
- https://huggingface.co/spaces/stabilityai/stable-diffusion
- https://huggingface.co/spaces/dalle-mini/dalle-mini

End

Thank you!

Connect with me