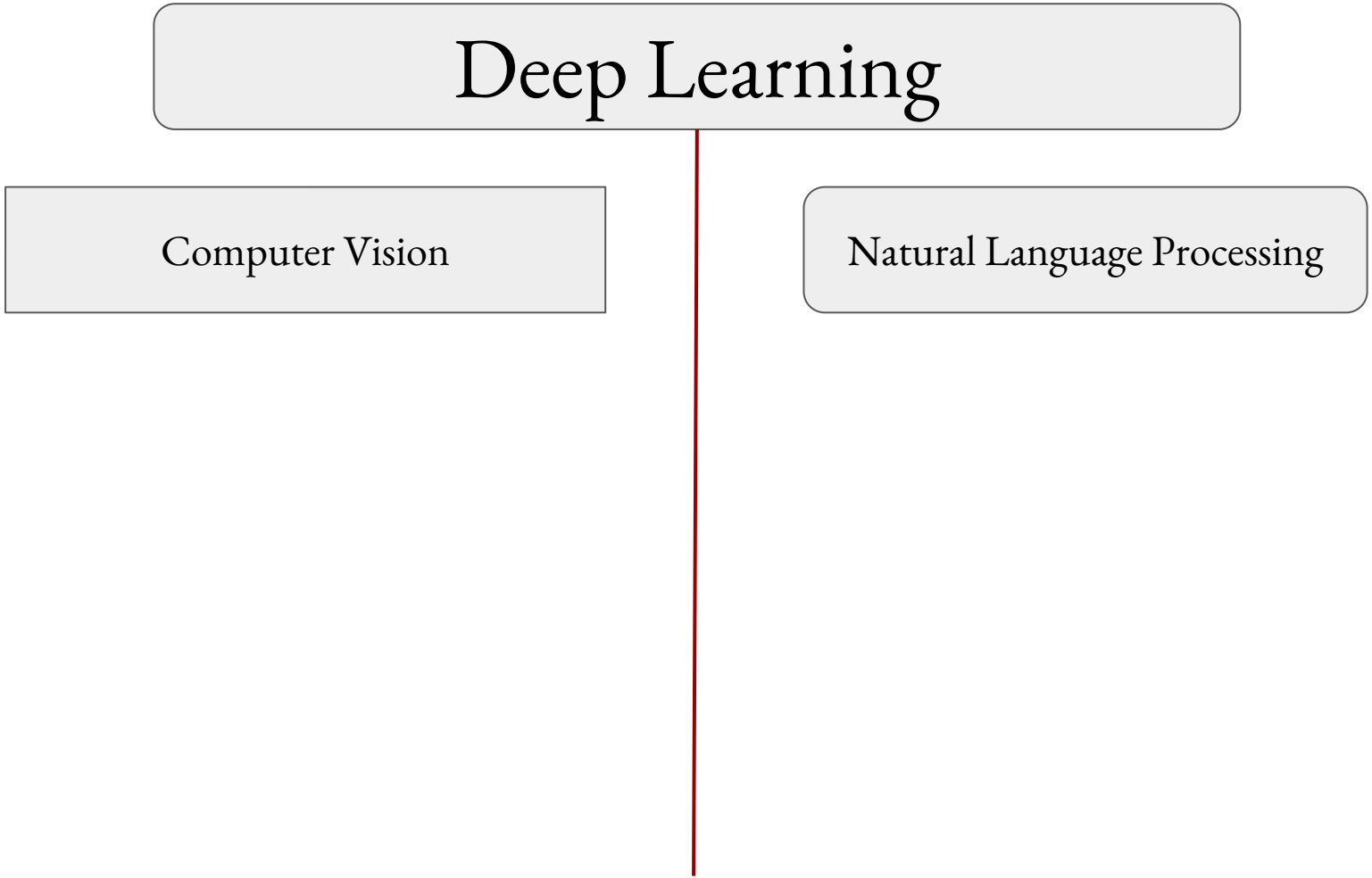
Getting started with transformers for computer vision

Divya Swaminathan & Tony Reina

JupyterCon , 5th Nov 2025

Deep Learning


Deep Learning
Computer Vision
Natural Language Processing

Deep Learning

Computer Vision

CNNs (2012 onwards)

- Image Classification
- Object Segmentation
- Semantic Segmentation

Natural Language Processing

Deep Learning

Computer Vision

CNNs (2012 onwards)

- Image Classification
- Object Segmentation
- Semantic Segmentation

Natural Language Processing

Transformers (2017 onwards)

- Text Summarization
- Translation
- Text Generation

What's so great about transformers?

- **Parallelization.**
  - Before, text based tasks were run sequentially.
  - Models lost track of relationship b/w words.
  - Transformer models scale and can be trained on huge amounts of data.
- **Attention**.
  - Learn relationship between words.
  - In language position is important, the position of a word can alter context and meaning.
  - Opens the door to learning about context.

Parallelization + Attention

Opens the door to learning about context , fast !!

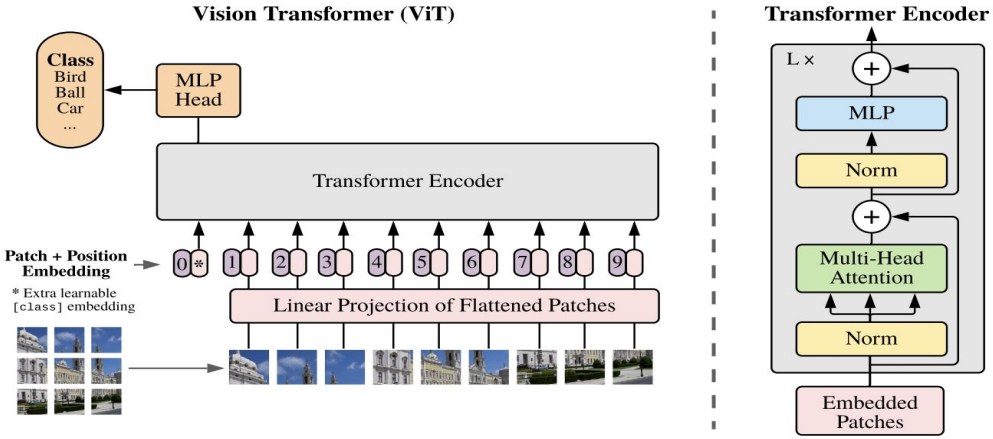


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

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

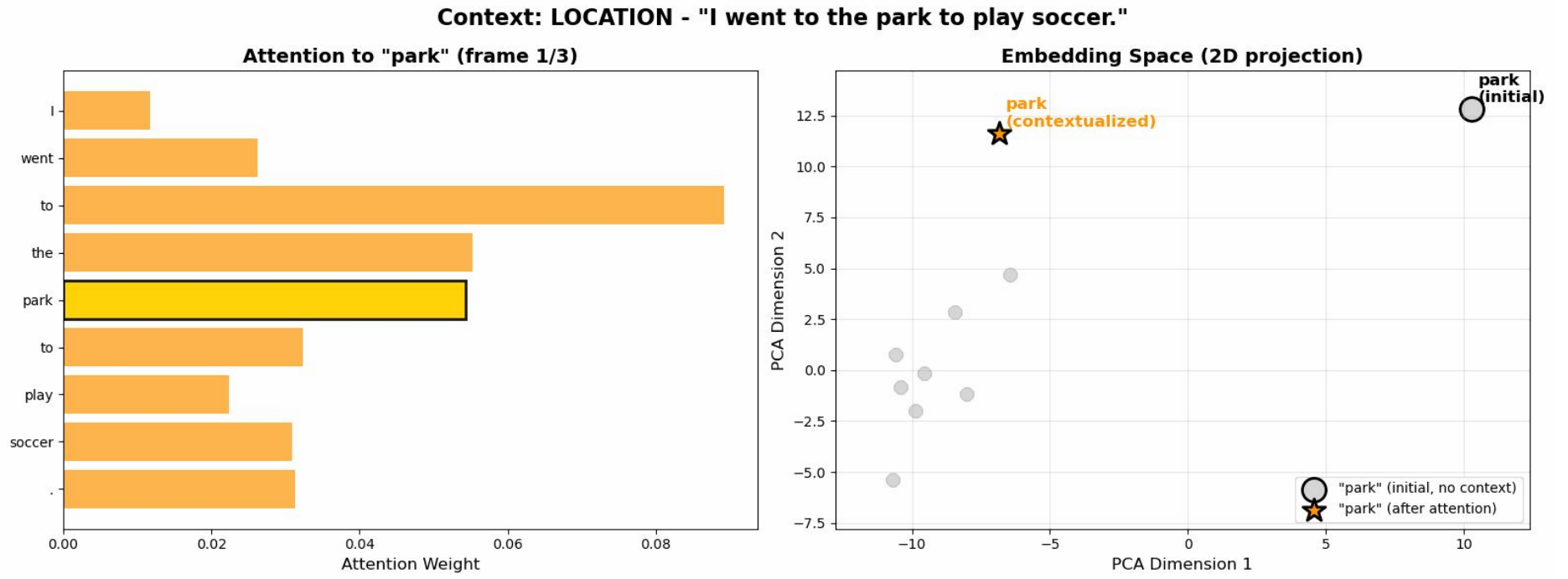
Google Research, Brain Team

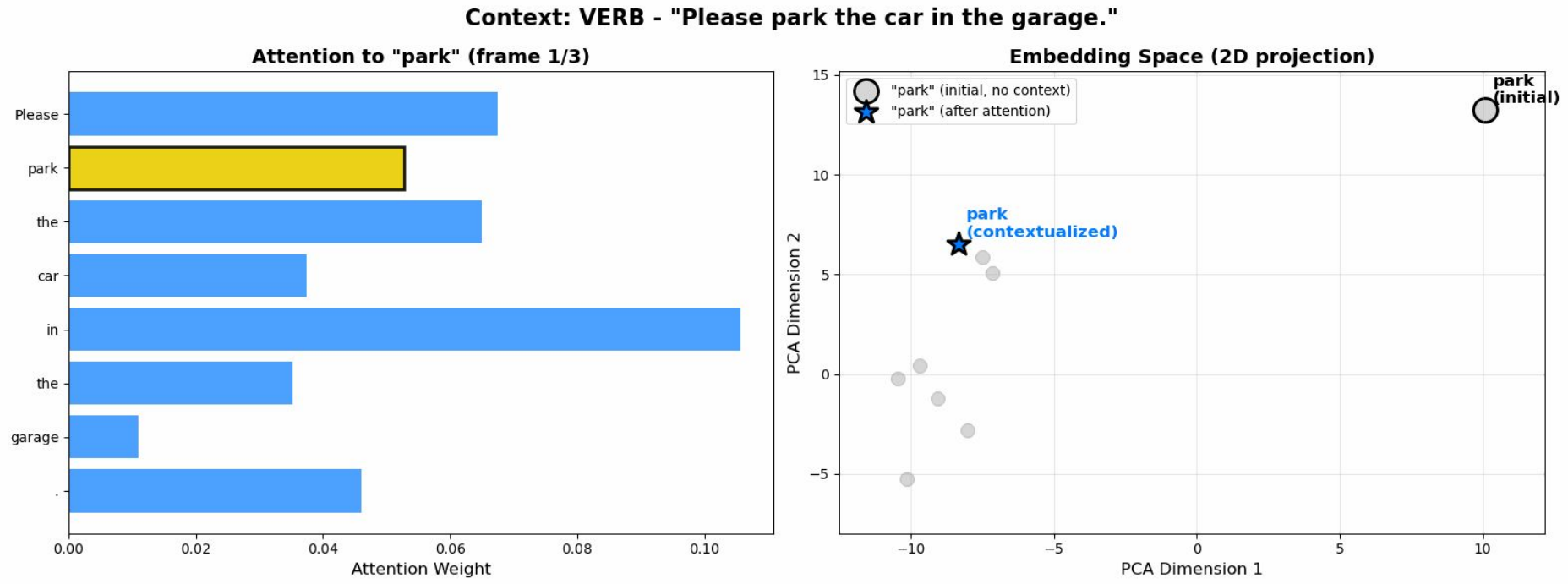
2020

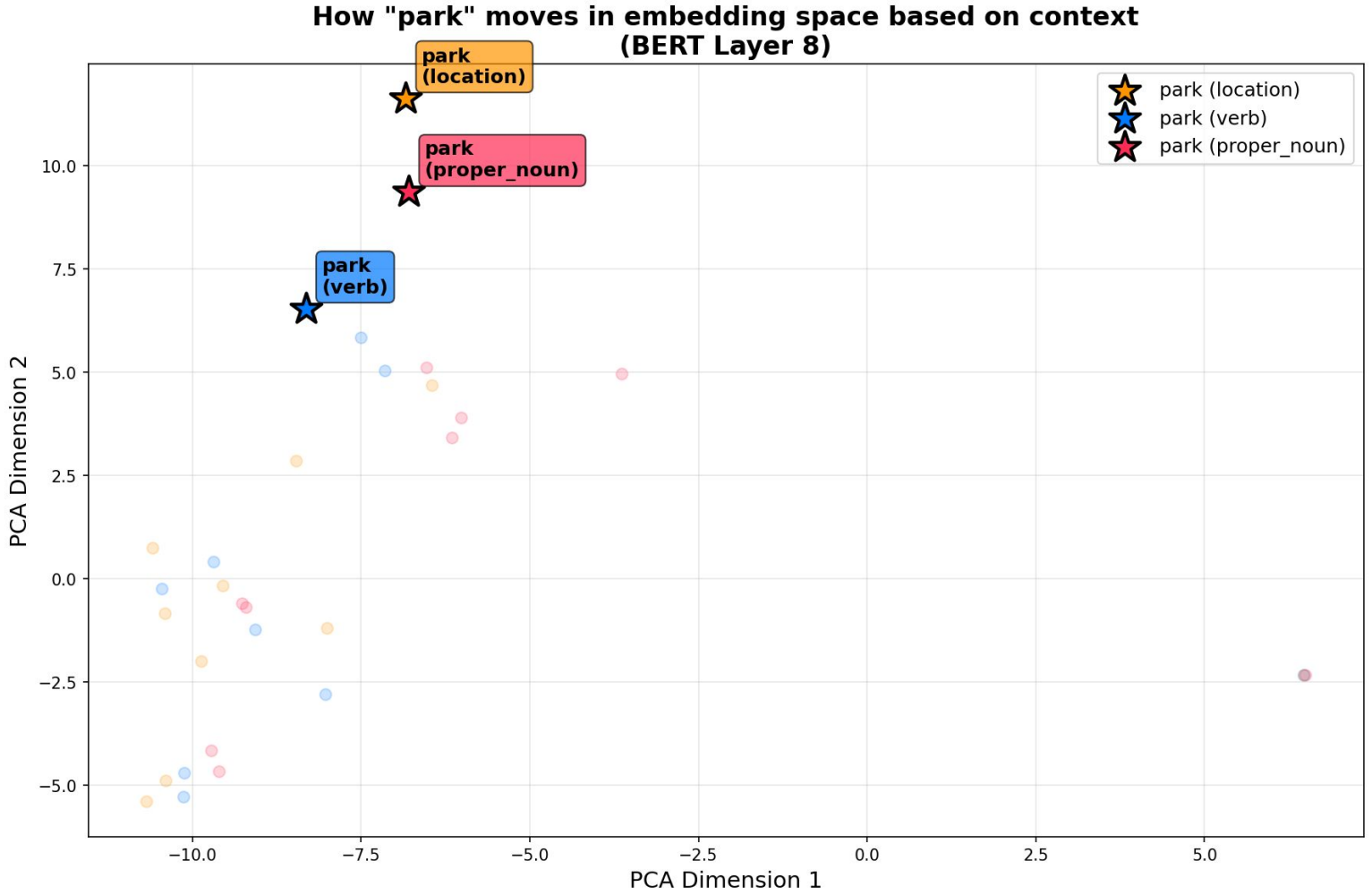
76,508 citations !!

Attention

Location	I went to the park to play soccer.
Verb	Please park the car in the garage.
Proper Noun	Central Park is in New York City.


Context: LOCATION - "I went to the park to play soccer."
Attention to "park" (frame 1/3)
I
went
to
the
park
to
play
soccer
.
0.00
0.02
0.04
0.06
0.08
Attention Weight
Embedding Space (2D projection)
park
(contextualized)
park
(initial)
"park" (initial, no context)
"park" (after attention)
PCA Dimension 1
PCA Dimension 2


Context: VERB - "Please park the car in the garage."
Attention to "park" (frame 1/3)
Please
park
the
car
in
the
garage
.
0.00
0.02
0.04
0.06
0.08
0.10
Attention Weight
Embedding Space (2D projection)
"park" (initial, no context)
"park" (after attention)
park
(initial)
park
(contextualized)
PCA Dimension 2
15
10
5
0
−5
−10
−5
0
5
10
PCA Dimension 1


How "park" moves in embedding space based on context
(BERT Layer 8)
park (location)
park (proper_noun)
park (verb)
park (location)
park (verb)
park (proper_noun)
PCA Dimension 2
PCA Dimension 1
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0

The Hugging Face Hub 🤗

- A platform for ML & AI development and deployment. Loads of open source, pretrained models and datasets are available to plug and play.
- Transformers library in python.
- Annotated Datasets - Audio, Video, Images and Text
- ~1000 models, ~250 datasets, ~600 space (model deployment)
- It is indeed easy to go set this up and start with computer vision projects.

Models	2,181,039	Pretrained. Open Source
Datasets	535,389	Audio, Video, Text, Images, Multimodal
Tasks	~ 50	
Space	~600	Deploy your code.

Getting started with image classification

- Install transformers library.
- Use pretrained, default models to predict classification on images from coco datasets.

-
```
from transformers import pipeline

pipe = pipeline("image-classification",
        model="microsoft/resnet-50")

result = pipe(image_macaw)

print('Image is classified as:-', result[0]['label'],
      with a probability score of', result[0]['score'])
```



Image is classified as:- ***macaw*** with a probability score of ***0.9999986886978149***

Link to image - "https://huggingface.co/datasets/huggingface/documentation-images/resolve/main/hub/parrots.png"

Getting started with image classification

- Install transformers library.
- Use pretrained, default models to predict classification on images from coco datasets.

-
```
from transformers import pipeline*

pipe = pipeline("image-classification",
        model="microsoft/resnet-50")

result = pipe(image_macaw)

print('Image is classified as:-', result[0]['label'],
      with a probability score of', result[0]['score'])
```



Image is classified as:- ***macaw*** with a probability score of ***0.9999986886978149***

* a pipeline is a high-level, user-friendly abstraction that simplifies the process of using pre-trained Transformer models for various tasks.

Getting started with image classification

- Install transformers library.
- Use pretrained, default models to predict classification on images from coco datasets.

- ```
  from transformers import pipeline

  pipe = pipeline("image-classification",
          model="microsoft/resnet-50")

  result = pipe(image_macaw)

  print('Image is classified as:-', result[0]['label'],
          with a probability score of', result[0]['score'])
  ```



Image is classified as:- ***macaw*** with a probability score of ***0.9999986886978149***
  ```

Getting started with image classification

- Install transformers library.
- Use pretrained, default models to predict classification on images from coco datasets.

-
```
from transformers import pipeline

pipe = pipeline("image-classification",
        model="microsoft/resnet-50")

result = pipe(image_macaw)

print('Image is classified as:-', result[0]['label'],
      with a probability score of', result[0]['score'])
```



Image is classified as:- ***macaw*** with a probability score of ***0.9999986886978149***

Getting started with Object Segmentation.

```
segment = pipeline( "object-detection",
                    model =  "facebook/detr-resnet-50")
result = segment(image_macaw)
for elem in result:
    print(elem)
```

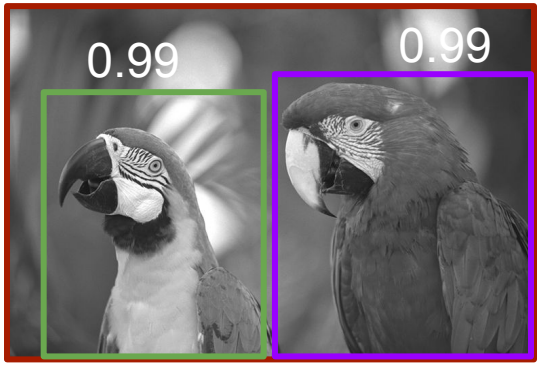


```
{'score': 0.9966394901275635, 'label': 'bird', 'box': {'xmin': 69, 'ymin': 171, 'xmax': 396, 'ymax': 507}}

{'score': 0.999381422996521, 'label': 'bird', 'box': {'xmin': 398, 'ymin': 105, 'xmax': 767, 'ymax': 507}}
```

Getting started with Object Segmentation.

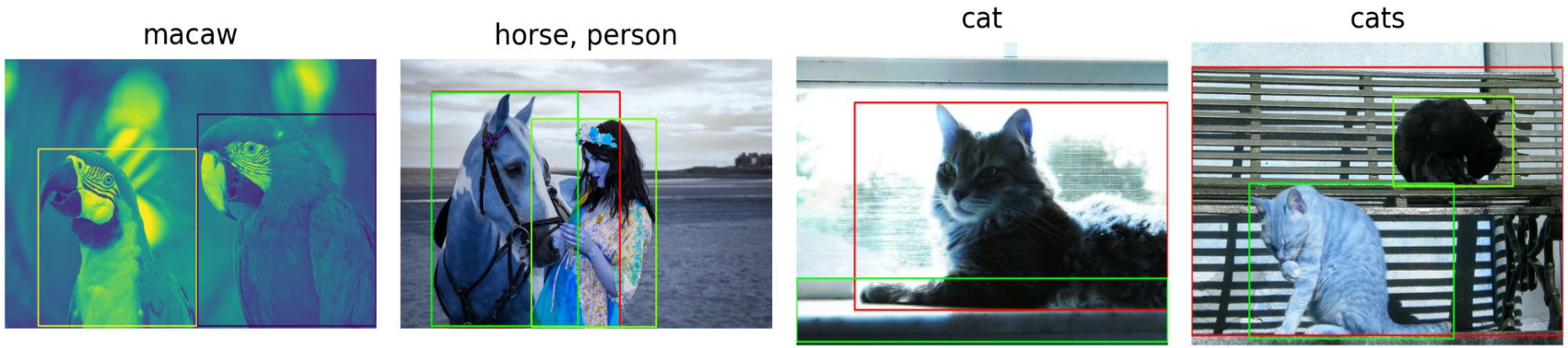
```
segment = pipeline( "object-detection",
                    model =  "facebook/detr-resnet-50")
result = segment(image_macaw)
for elem in result:
    print(elem)
```



{'score': 0.9966394901275635, 'label': 'bird', 'box': {'xmin': 69, 'ymin': 171, 'xmax': 396, 'ymax': 507}}

{'score': 0.999381422996521, 'label': 'bird', 'box': {'xmin': 398, 'ymin': 105, 'xmax': 767, 'ymax': 507}}

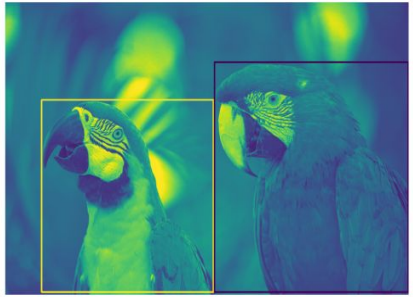
Segmentation



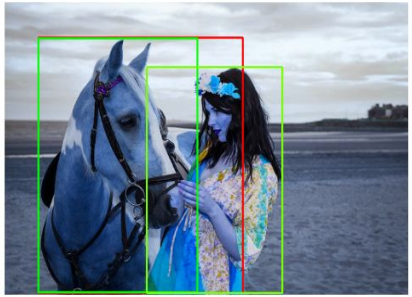
Images from COCO dataset.

Segmentation

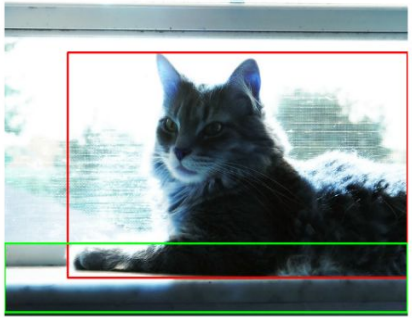
macaw



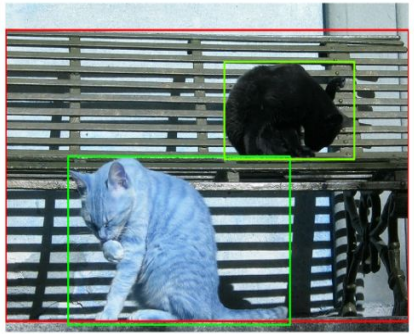
horse, person



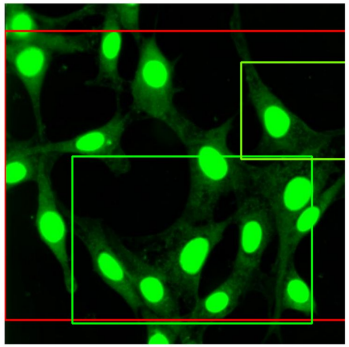
cat



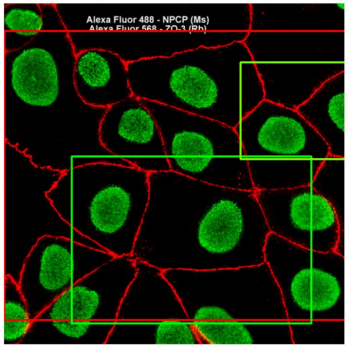
cats



GFP tagged cells

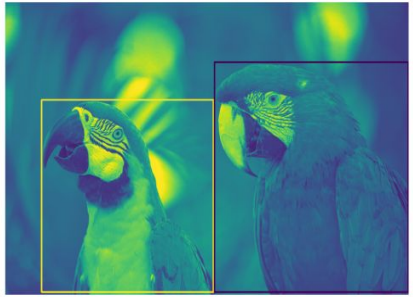


GFP tagged kidney cells

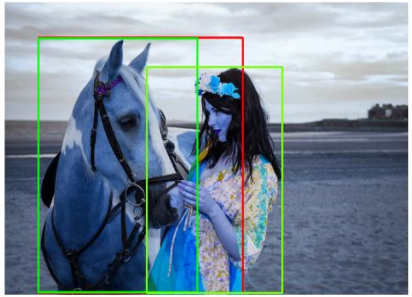


Segmentation

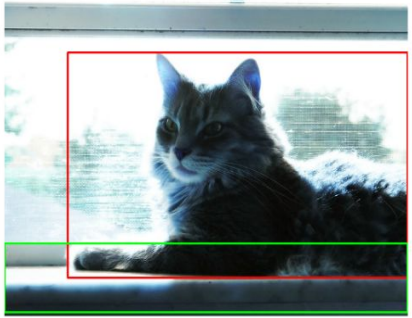
macaw



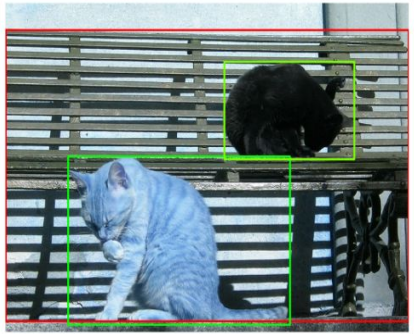
horse, person



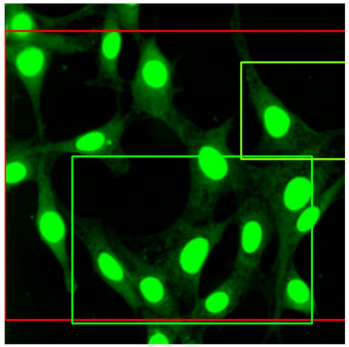
cat



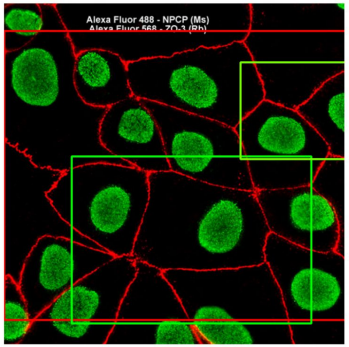
cats



GFP tagged cells



GFP tagged kidney cells



Default models will fail on custom image datasets.

Case Study - Breast Cancer Classification

- Identify dataset of cancer images on hub.
  - ~ 130,000 images across several cancer types ("Aliounethegoat/classification-medicale-multi-cancer")

Case Study - Breast Cancer Classification

- Identify dataset of cancer images on hub. ("Aliounethegoat/classification-medicale-multi-cancer")
  - ~ 130,000 images across several cancer types
- Take a subset of images , i.e. all breast cancer images.
  - ~10,000 images , 2 categories
  - Push to hub, generating a breast cancer dataset anyone can use.
- Loading and exploring datasets is easy.
  - path_to_dataset = "divyaswamy/breast_cancer_practice"
  - ds = load_dataset(path_to_dataset)

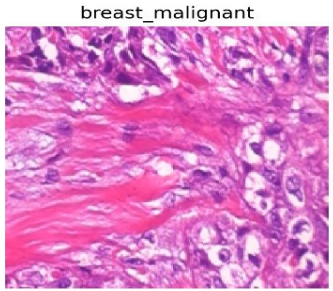
Case Study - Breast Cancer Classification

- Identify dataset of cancer images on hub. ("Aliounethegoat/classification-medicale-multi-cancer")
  - ~ 130,000 images across several cancer types
- Take a subset of images , i.e. all breast cancer images.
  - ~10,000 images , 2 categories
  - Push to hub, generating a breast cancer dataset anyone can use.
- Loading and exploring datasets is easy.
  - path_to_dataset = "divyaswamy/breast_cancer_practice"
  - ds = load_dataset(path_to_dataset)

```
DatasetDict({
    train: Dataset({
        features: ['image', 'label', 'label_name'],
        num_rows: 7500
    })
    validation: Dataset({
        features: ['image', 'label', 'label_name'],
        num_rows: 1250
    })
    test: Dataset({
        features: ['image', 'label', 'label_name'],
        num_rows: 1250
    })
})
```

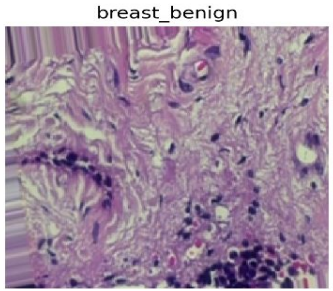
Case Study - Breast Cancer Classification

- Identify dataset of cancer images on hub.
  - ~ 130,000 images across several cancer types
- Take a subset of images , i.e. all breast cancer images.
  - ~10,000 images , 2 categories
  - Push to hub, generating a breast cancer dataset anyone can use.
- Loading and exploring datasets is easy.
  - path_to_dataset = "divyaswamy/breast_cancer_practice"
  - ds = load_dataset(path_to_dataset)









Case Study - Breast Cancer Classification

- Pre-trained model will not work on these images.
- Image is classified as:- ***nematode*** with a probability score of ***0.028.***

```
from transformers import pipeline
classifier = pipeline('image-classification',  model='google/vit-base-patch16-224')
result = classifier(ds['test'][0]['image'])
print('Image is classified as:-', result[0]['label'],  'with a probability score of', result[0]['score'])
```

Case Study - Breast Cancer Classification

- Pre-trained model will not work on these images.
- Image is classified as:- ***nematode*** with a probability score of ***0.028.***

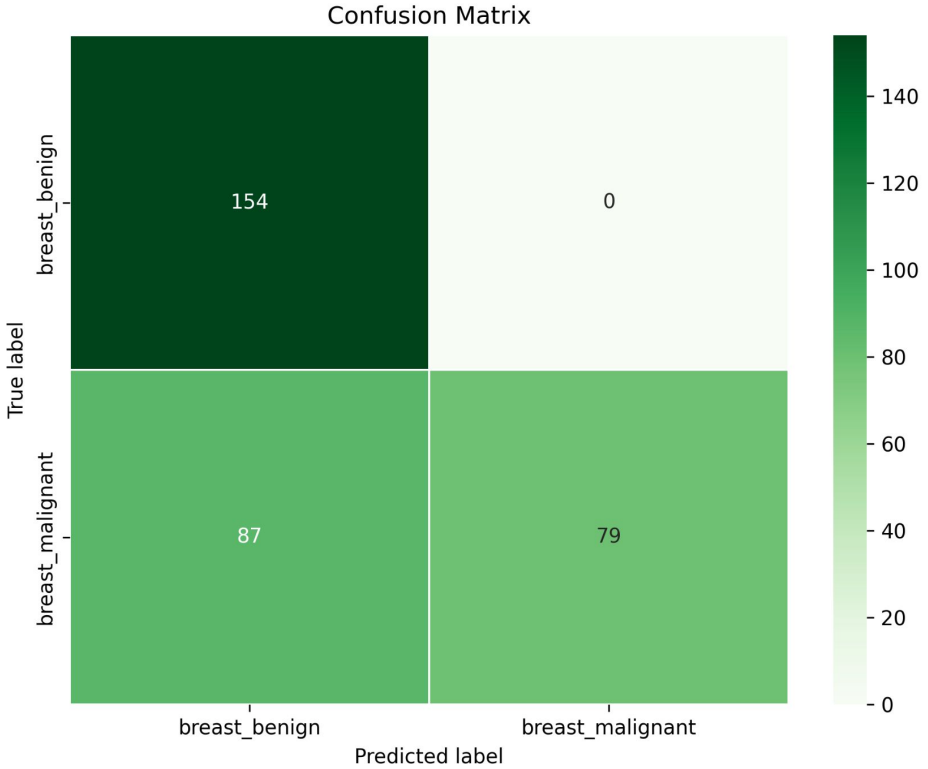
```
from transformers import pipeline
classifier = pipeline('image-classification',  model='google/vit-base-patch16-224')
result = classifier(ds['test'][0]['image'])
print('Image is classified as:-', result[0]['label'],  'with a probability score of', result[0]['score'])
```

- Fine tuning a model - Of all the model layers, train the last classification layer - freeze weights on the others.

```
model = ViTForImageClassification.from_pretrained(
                    'google/vit-base-patch16-224', num_labels=len(labels),
                     id2label=id2label, label2id=label2id,
                    ignore_mismatched_sizes = True)

for name, weights in model.named_parameters():  # weights per layer.
        if not name.startswith('classifier'):          # If the layer is not the classification layer
                weights.requires_grad = False       # Do not update weights in this layer
```

Case Study - Breast Cancer Classification



- Trained model for 3 epochs. ~ 40 mins, 96% accuracy on validation dataset.
- Recall for malignant cells is only 48% !!
- Over all a nice little project to familiarize oneself with the hugging face platform, transformers library and a basic computer vision task.

Pain points

- ***Annotated data*** - if one has a very specific image based problem one would like to solve - the appropriate annotated dataset is likely the most significant blocker.
- It takes time , effort and $$ to generate and annotate data.
- Get clarity on how the model will ingest the data.
- Training hardware.
    - On my laptop I quickly run into issues.
    - Paid subscription on google colab, AWS etc if you are working independently and want to work with complex tasks and/ or large datasets.

The Wins

- Easy to set up and get started.
- One can plug and play, learning about different models, datasets and tasks.
  - Your curiosity is the limiting factor.
- A lot of resources are available on the hub and online to help along.
  - Hugging Face platform
  - Youtube
  - https://github.com/DivyaSwamy/transformers_tutorials

Thank you !!