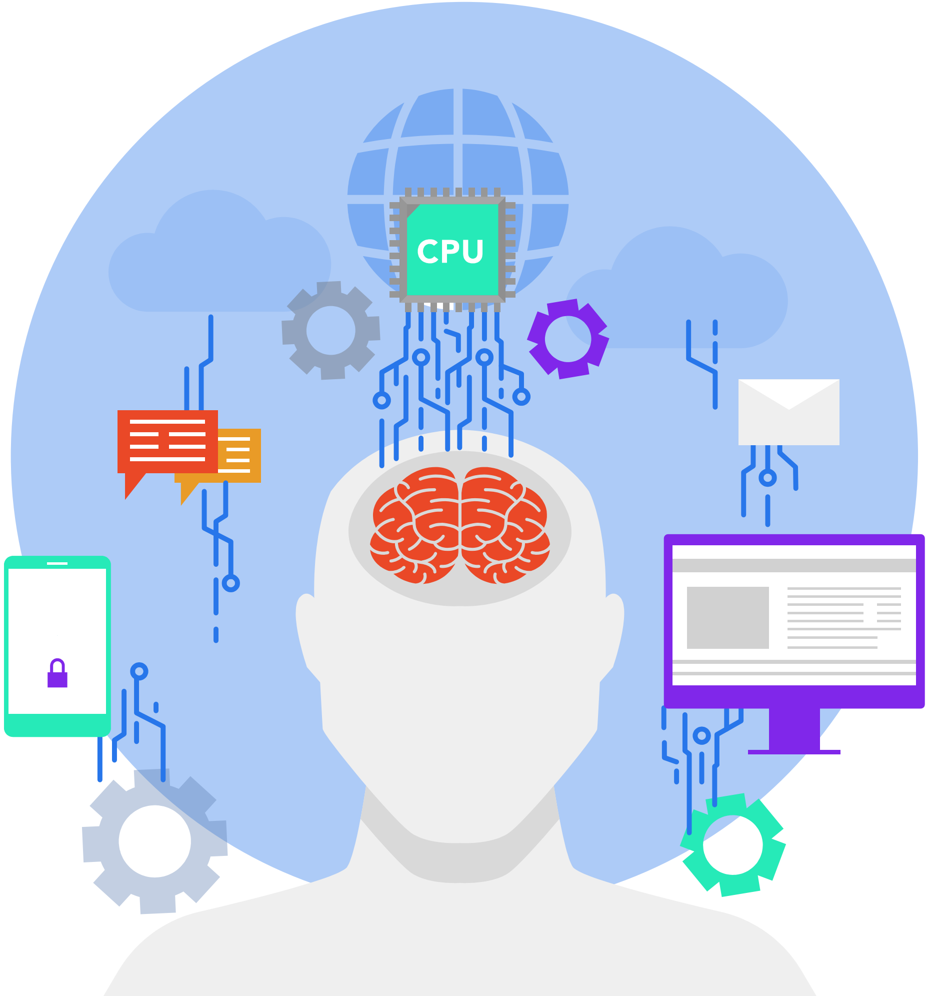
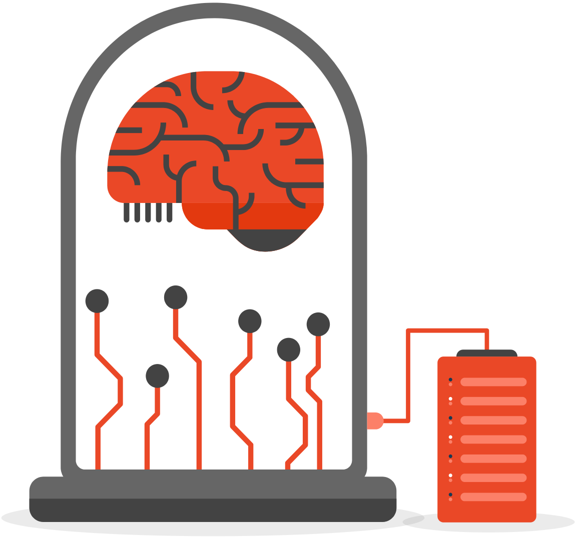


A basic ML pipeline

Sarper Okuyan
January 12, 2024

Object Detection Pipeline



01 Data Input

A JPEG image

02 Choosing model

Finding a suitable CNN-based algorithm

03 Making prediction

Feature extraction, object classification, Bounding box refinement

04 Containerization

Producing a Docker image with dependencies

05 Pushing image to cloud

Using Terraform, uploading Docker image to AWS ECR

06 Managing cloud infrastructure

Work-in-progress

01

Data Input (Extract)

```python
def download_image(image_url):
    try:
        # Send a GET request to the URL
        response = requests.get(image_url)

        # Check if the request was successful (status code 200)
        if response.status_code == 200:
            # Open the image using PIL
            image = Image.open(BytesIO(response.content))

            # Save the image
            image.save("./downloaded_image.jpg")
        else:
            print(f"Failed to retrieve the image. Status code: {response.status_code}")

    except Exception as e:
        print(f"An error occurred: {e}")
```

- Input: A JPEG image
- Similar in the implementation to extracting data using an API

01

Data Input (Transform)

Three main transformation functions:

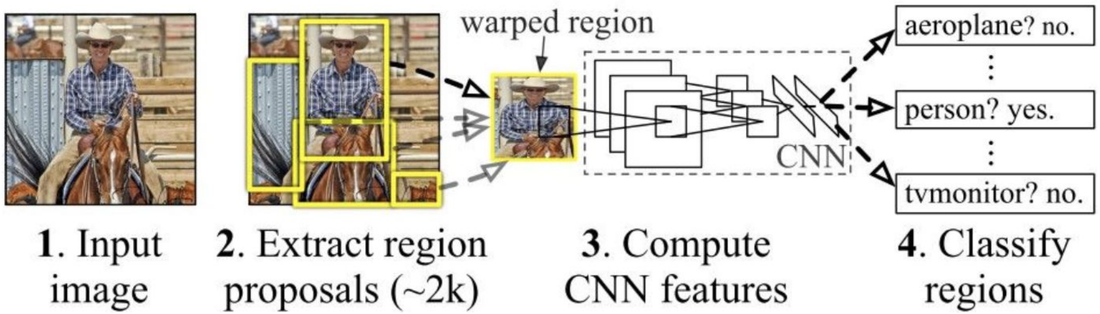
1. Converting from PIL to Tensor format
2. Adding Batch dimension
3. Converting to float

```python
# Transform

def convert_to_PIL(target_image):
    try:
        target_image_tensor_int = pil_to_tensor(target_image)
        #print(target_image_tensor_int.shape)
        return target_image_tensor_int
    except Exception as e:
        print(f"An error occurred: {e}")
        return None

def add_batchdim(target_image_tensor_int):
    try:
        target_image_tensor_int = target_image_tensor_int.unsqueeze(dim=0)
        #print(target_image_tensor_int.shape)
        return target_image_tensor_int

    except Exception as e:
        print(f"An error occurred: {e}")
        return None

def convert_imagerep_from_int_to_float(target_image_tensor_int):
    try:
        #print(target_image_tensor_int.min(), target_image_tensor_int.max())

        target_image_tensor_float = target_image_tensor_int / 255.0

        #print(target_image_tensor_float.min(), target_image_tensor_float.max())
        #print(target_image_tensor_float)

        return target_image_tensor_float
```

Choosing model

PyTorch

- Pre-trained object detection algorithm: Faster Region-based Convolutional NN
- Three main parts: Region Proposal Generator, Feature Extraction, Classification
- Why use RCNN? Relatively high accuracy in object detection tasks.
- Weights from COCO dataset (300K images - >200K labeled)



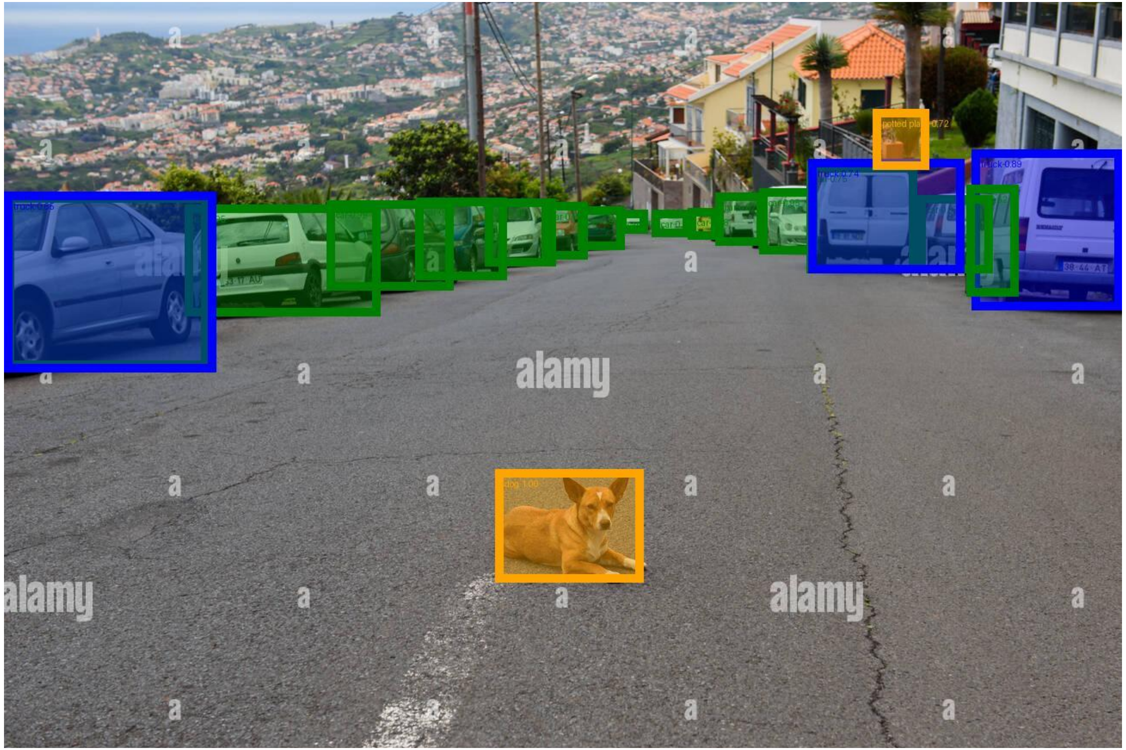
03

Making predictions

- Viewpoints: A major Computer Vision challenge

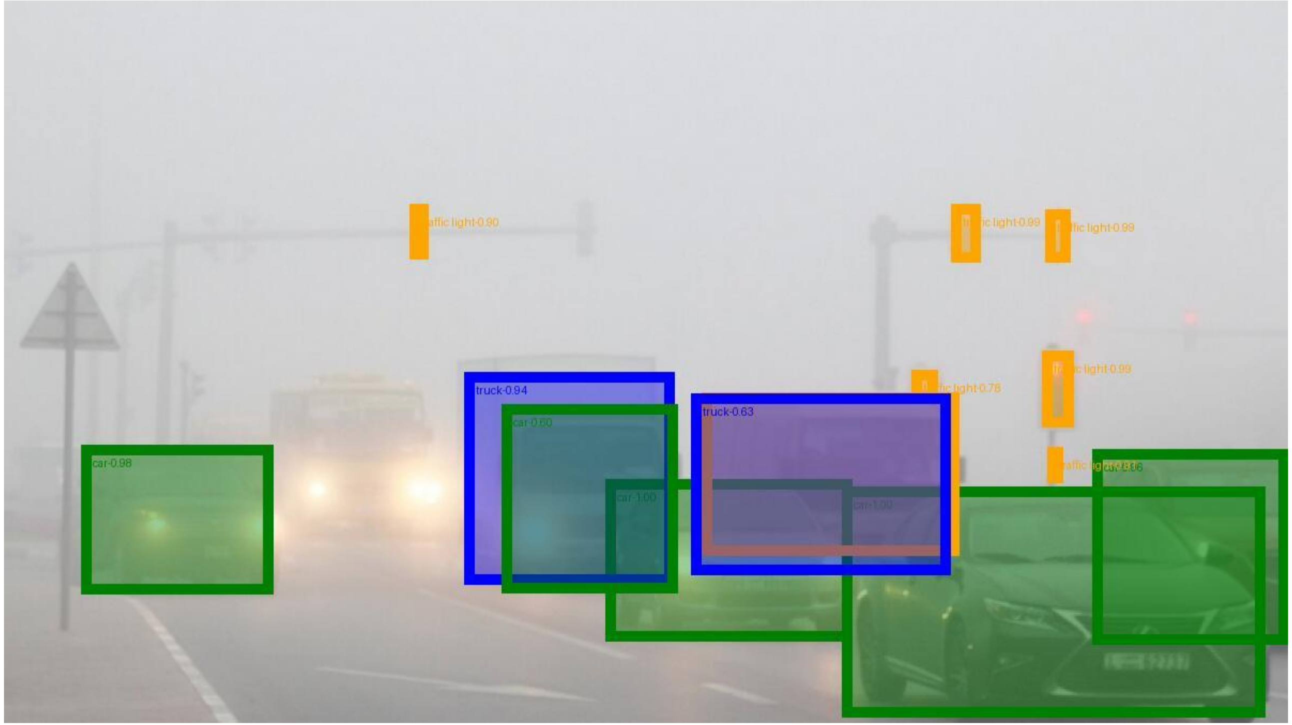
03

Making predictions



03

Making predictions



- Lightning: A major Computer Vision challenge

04

Containerization

```
sarperokuyan $ docker images
REPOSITORY                                         TAG      IMAGE ID      CREATED         SIZE
657403958857.dkr.ecr.eu-west-1.amazonaws.com/mlapp-repo  latest   72c38ab91e41  41 hours ago    7.67GB
objdetect-docker-app                               latest   72c38ab91e41  41 hours ago    7.67GB
my_image                                           latest   d195427a559a  3 weeks ago     5.62MB
python                                             latest   58a8f3dcd68a  5 weeks ago     1.02GB
postgres                                           latest   391a00ec7cac  5 weeks ago     425MB
ubuntu                                             latest   b6548eacb063  6 weeks ago     77.8MB
jupyter/scipy-notebook                             latest   ad65fcfebde3  2 months ago    4.14GB
```

- Containerization solves the “works-on-my-computer” problem.
- Room for Improvement:
  The object detection Docker image is very large in size. This may lead to increased costs in a cloud environment, scaling challenges and increased deployment time.

05

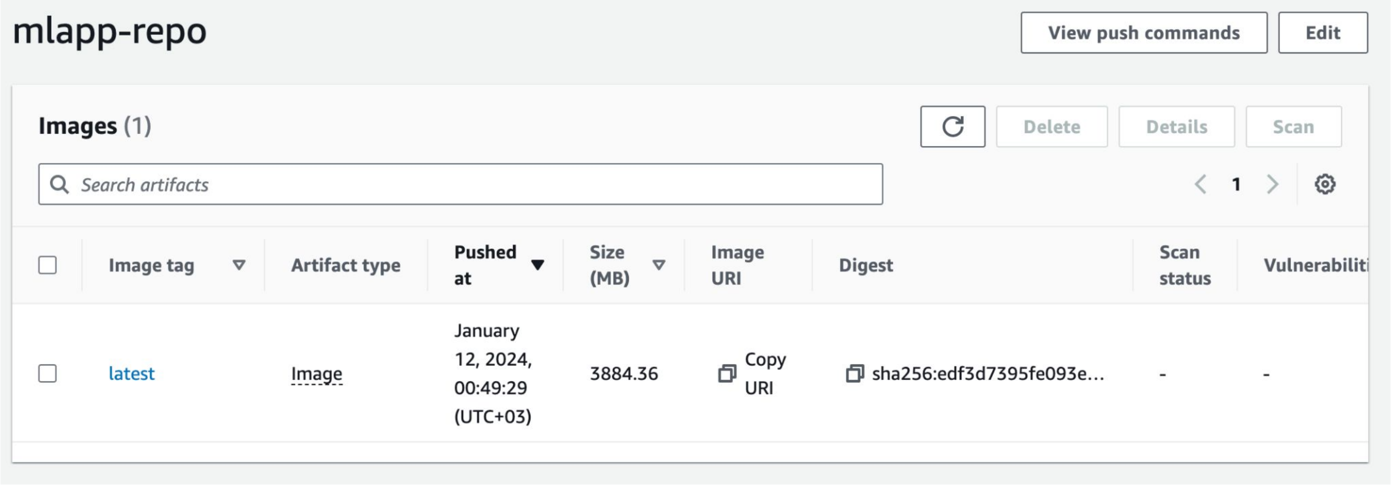
Pushing image to the cloud

- Why transferring to the cloud? Scalability, Compute Power and Cost Efficiency.
- Cloud of choice: AWS
- Infrastructure-as-Code softwares facilitate managing and provisioning cloud infrastructure.
- Terraform is useful because of the declarative and human-readable language.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider and Authentication
provider "aws" {
  region     = "eu-west-1"
  access_key = "A[redacted]7"
  secret_key = "3[redacted]"
}
```

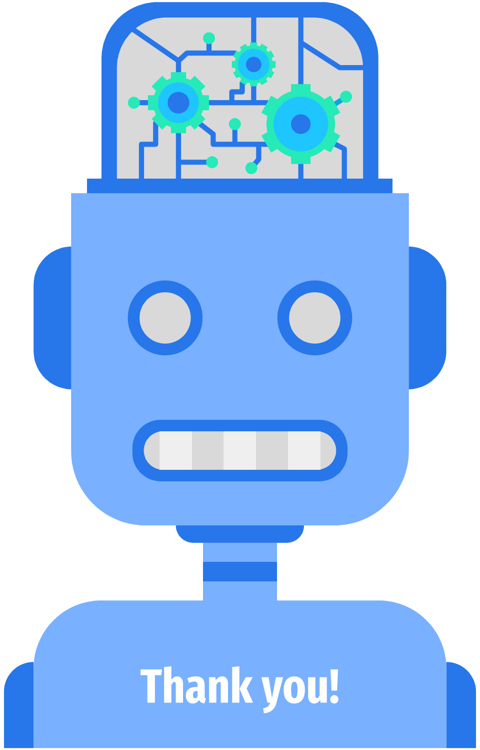
06

Work-in-progress: Managing Cloud Infrastructure



Next steps:

1. Creating a AWS ECS Cluster
2. Creating a ECS Service Task
3. Deploying Docker containers on ECS


Thank you!