Deploying Artificial Intelligence Models for Edge Computing on Drones

Dr. Rajiv Misra

Professor, **Dept. of Computer Science & Engg. Indian Institute of Technology Patna** **rajivm@iitp.ac.in**

INTRODUCTION

- With the increase in the number of devices connected to a Cloud system, Cloud Computing suffers from certain limitations regarding the high bandwidth requirements to transmit data to the centralized Cloud architecture, high computational power to process the data, and therefore high latency of data processing.
- To overcome these limitations a low latency network to process and return data faster to the request sender is required.
- As a result of this, we are witnessing a shift from Cloud Computing to Edge Computing.
- Edge Computing refers to the computations that take place at the 'outside Edge' of the internet, as opposed to Cloud Computing, where computation happens at a central location.
- Edge Computing typically executes close to the data source, for example onboard or adjacent to a connected camera.
- "Edge Computing market size is expected to reach USD 29 billion by 2025.
  -Grand View Research"

INTRODUCTION

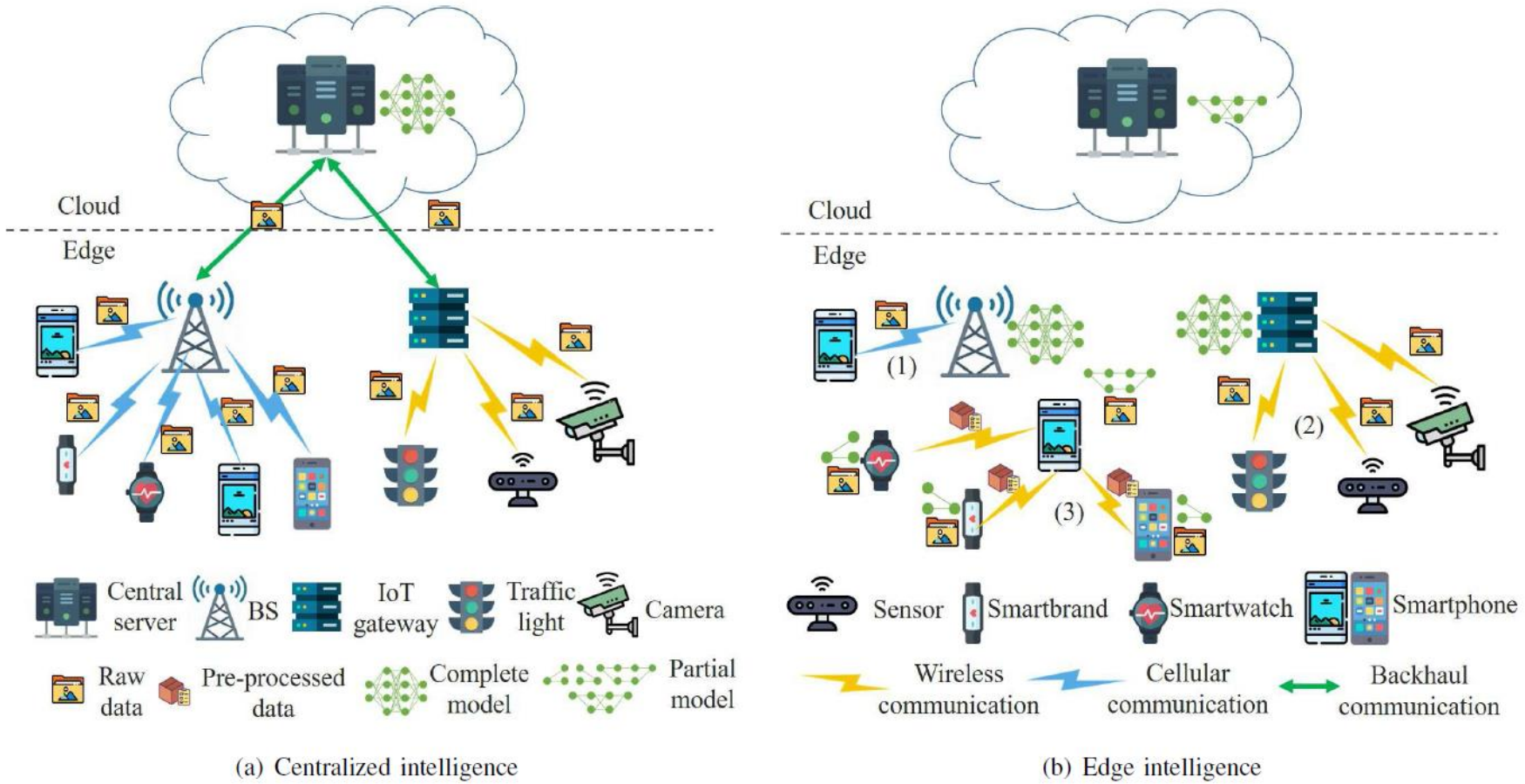
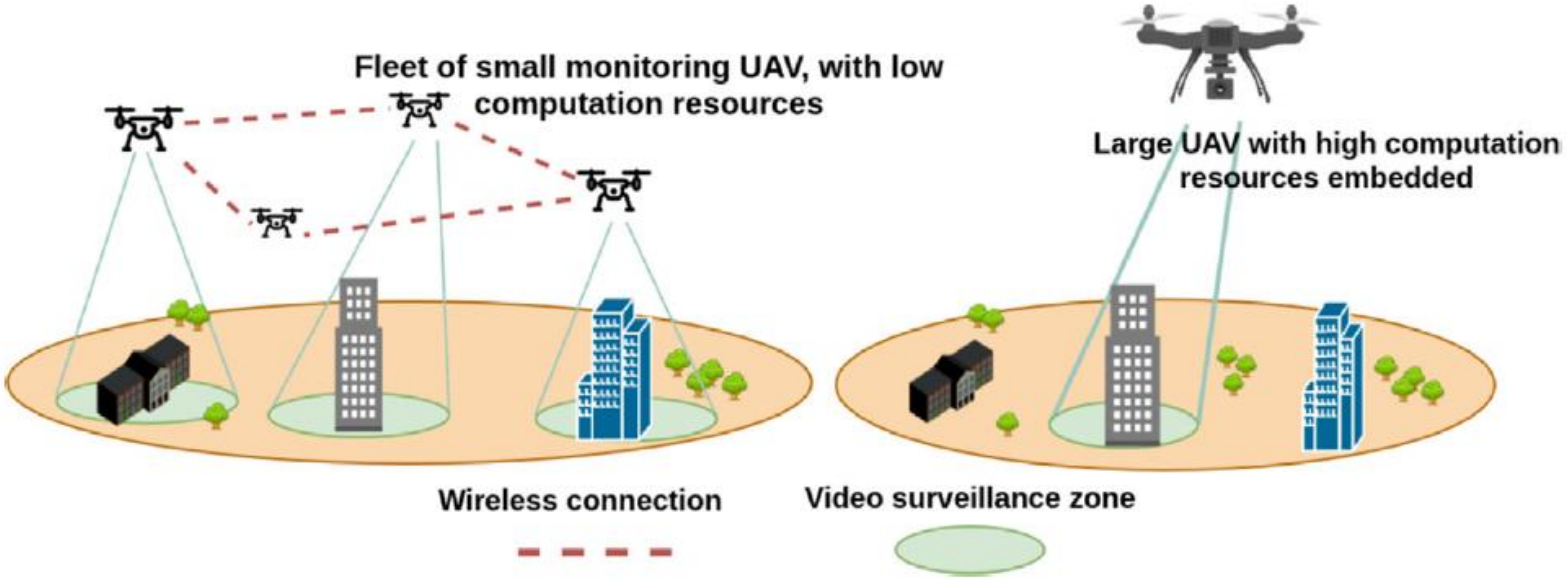


Fig. 1. The comparison of traditional intelligence and edge intelligence from the perspective of implementation. In traditional intelligence, all data must be uploaded to a central cloud server, whilst in edge intelligence, intelligent application tasks are done at the edge with locally-generated data in a distributed manner.

Object Tracking for Edge Computing on Drones

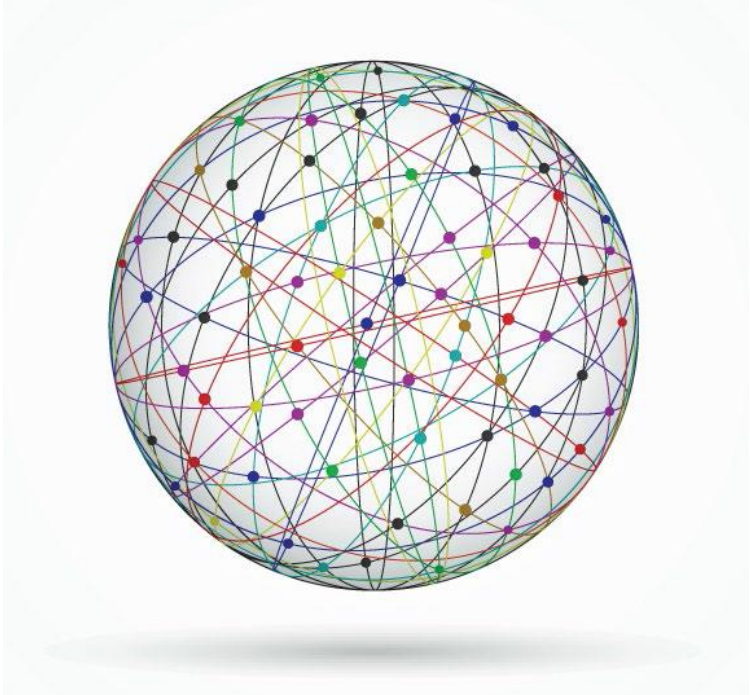


Search and Intelligence application scenario using multiple drones with diverse video cameras and sensing capabilities.

Edge Computing

- Idea is to push applications, data and computing power to the edge of the Internet, in close proximity to mobile devices, sensors, and end users

- An early example is Akamai, with servers around the world to distribute web site content from locations close to the user (content delivery networks, or CDNs)



Edge Computing: Drivers

Latency

- data processing close to where it originates avoids round-trip time to the cloud

Bandwidth

- optimization of communication to and from the cloud

Privacy/security

- sensitive data stays local

Connectivity

- continued processing (in some cases) despite lack of connectivity to the cloud

Local dependencies

- data processing close to points of interaction with end users and other system components

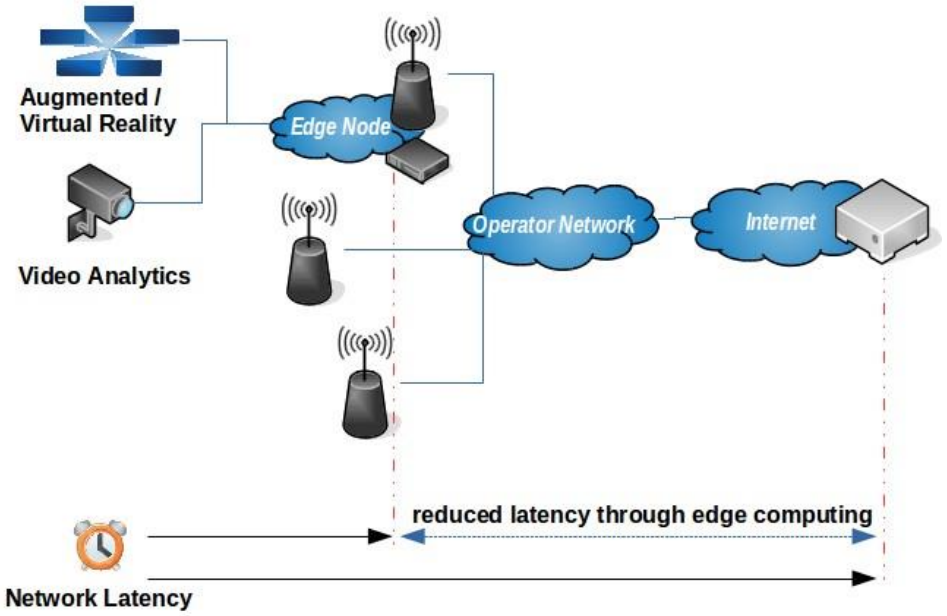
Edge Computing: The Telco View

Opportunity for providing edge computing devices in existing infrastructure

- e.g., micro data centers at the base of cellular towers

Multiple organizations seeking standardization: Multi-Access Edge Computing (MEC), Open Edge Computing (OEC), OpenFog Consortium, etc.

Business model is still not clear: Who pays for the service? Consumer? Content Provider?



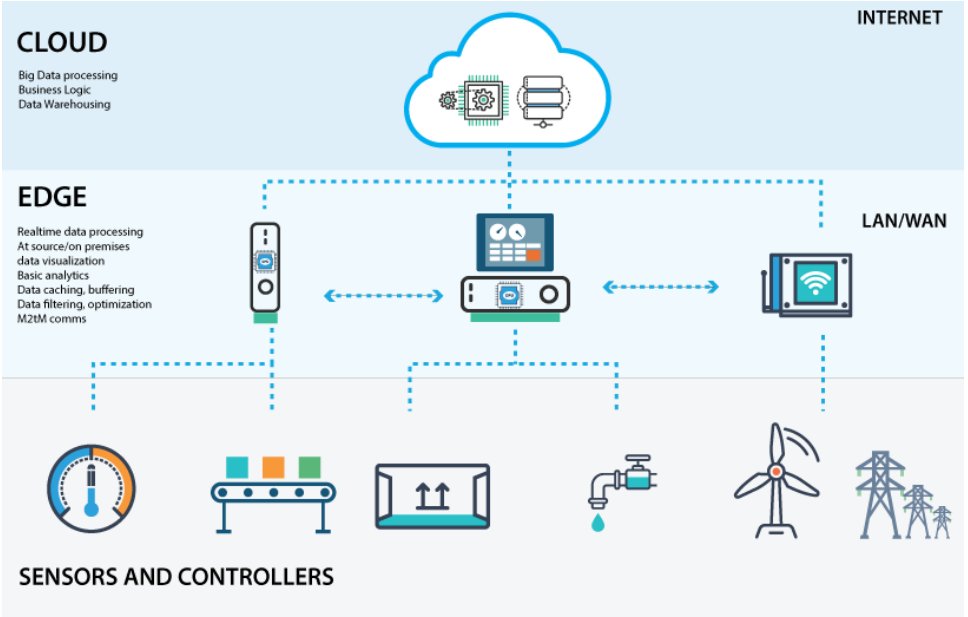
Edge Computing: The Cloud Provider View

Goal is mainly to provide

- Content Delivery Network (CDN) services
- IoT data processing and aggregation for data in transit to the cloud

Examples

- Azure IoT Edge —deploy business logic to edge devices and monitor from the cloud
- Amazon
  - AWS CloudFront —CDN Service, includes Lambda@Edge
  - AWS Greengrass —connected IoT devices can run AWS Lambda functions and other code on locally-collected data



Benefits of Edge Computing

- **Speed**

Where the response time is critical to the success of an application, interpreting input data close to the source is the better option if it is available.

When both input and output happen at the same location, such as in an IoT device, Edge Computing removes the network latency, and real-time becomes possible.

- **Bandwidth**

Computing at the Edge means less data transmission as most of the heavy-lifting is done by the Edge devices.

Instead of sending the raw data to the Cloud, most of the processing is done at the Edge, and only the result is sent to the Cloud, thus requiring less data transmission bandwidth.

- **Scale**

Edge Computing architecture is much more scalable because adding few additional IoT devices in the lot system seems less complex as there is no need to increase the Cloud compute power or network bandwidth.

Edge Computing: Scenarios

- A self-driving car is a perfect example of Edge Computing.
- In order for cars to drive down any road safely, it must observe the road in real-time and stop if a person walks in front of the car.
- In such a case, processing visual information and making a decision is done at the Edge, using Edge Computing.
- This example highlights one key motivation for Edge Computing — speed.
- All such real-time applications demand Edge Computing.
- Edge Computing is already being used in various applications ranging from self-driving cars and drones to home automation systems and retail.

Edge Computing: Scenarios

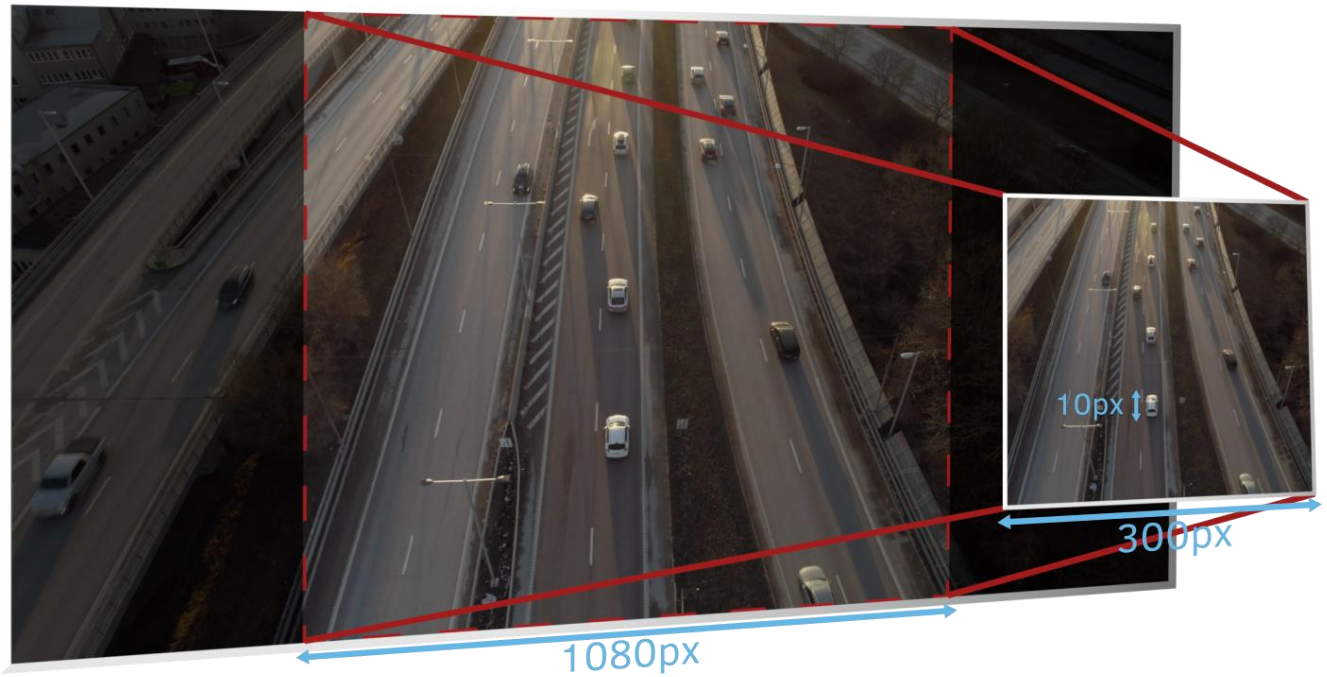
- Different organisations are working together to push the boundaries of industrial operators to integrate autonomous robots, drones, and advanced AI tools into their daily operations.
- Many cloud providers provides an enterprise-grade software platform in order to benefit from the most complex AI models in edge computing and to insure industrial operators are fully sovereign with their data.
- It enables any industry/academia to acquire data, organize their datasets, design their models, train, and deploy an optimized variant of their AI for edge computing, allowing drones and robots to detect, inspect, and track a series of objects of interest through time.
- One of the recent hot topic in this field include, automatic detection and tracking of the unmanned aerial vehicles in the most hardware-optimized way.

AI Models for Edge Computing on Drones

- A lot of progress has been made in recent years on object detection due to the use of convolutional neural networks (CNNs).
- Modern object detectors based on CNN network includes – Faster R-CNN, R-FCN, Multibox, SSD and YOLO.
- However, it can be difficult for practitioners to decide what architecture is best suited to their application.
- Standard accuracy metrics, such as mean average precision (mAP), do not tell the entire story, since for real deployments of computer vision systems, running time and memory usage are also critical.
- For example, mobile devices often require a small memory footprint, and drones require real time performance. Server-side production systems, like those used in Google, Facebook or Snapchat, have more leeway to optimize for accuracy, but are still subject to throughput constraints.

Object Tracking for Edge Computing on Drones

To illustrate some of these difficulties, consider a drone flying at a height of about 40 meters. For this drone, everything on the ground would look pretty small. Combine that with input images of size 300x300 pixels and your objects might be only ten pixels long, as can be seen in the following “camera versus resized” comparison:



This makes the task very challenging. Objects easily go undetected (false negatives), while the risk of wrong detections increases (false positives).

Single Shot Detector (SSD)

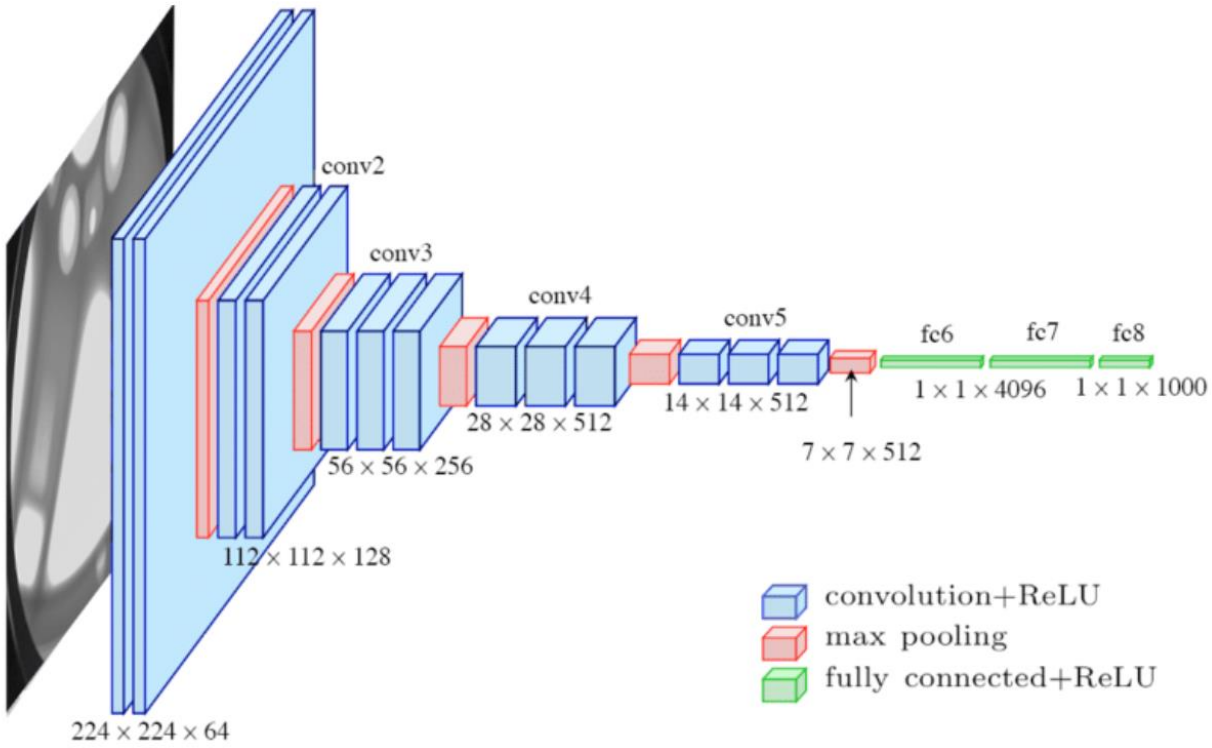
- We saw some improvements for time inference with Faster R-CNN, it is usually not enough to score a live video. Two fast object detection models came to the rescue: YOLO (You Only Look Once) and SSD.
- YOLO or SSD are preferable over Faster R-CNN for the application of this partnership, given their compromise between inference speed (FPS) and accuracy (mAP).
- It is known that the performance between the two models are similar in terms of speed and mAP. We decided on SSD due to its easy deployment on drone architecture.
- The different algorithms seen in the previous article are called two-shot detector models, meaning they have two stages:
  - Region Proposal Stage: Extract a few regions of the image where an object is probably present.
  - Classification: Classify the object inside each proposed region and rectify the final prediction location.

Single Shot Detector (SSD)

- SSD, on the contrary, has only one stage.
- There is no region proposal stage, that is why it's called Single-Shot Detector. Indeed, it predicts the boundary boxes and the classes at once, directly from the feature maps.
- To be more precise, SSD makes use of a series of convolutions to predict objects directly from the feature maps of its micro-architecture.
- Let's take an example with an input image of dimension 300x300x3 (width x height x number of channels, which are the three primary colors in our case).
- SSD can be implemented using different "backbones" or micro-architectures that serve as a feature extractor.
- In practice, backbones that make use of separable convolutions, such as MobileNetV2, score better than models using full convolutions in real-time performance. However, for the sake of this explanation, let the simpler VGG-16 be the pretrained model from which the SSD network will extract the features maps. Figure 1 shows the VGG-16 architecture:

Single Shot Detector (SSD)

- Below Figure shows the VGG-16 architecture:



- Since we need an image-like output to localize any possible object over an actual image, the last fully connected layers of the VGG-16 become unnecessary and are, therefore, removed from the model.
- SSD may then use the remaining image-like outputs on different levels of VGG-16 to perform the object detection task.

Single Shot Detector (SSD)

- An object detection model needs to learn two things:
  - which object we are facing and
  - where it is located.
- In this sense, SSD loss function composes the information into a number by a weighted sum of two different losses:
  - **The localization loss:** Measures the precision between the ground truth and the predicted boxes. This function considers only the “positive matches,” meaning that it looks at a predicted box only when it has an IoU (Intersection Over Union) greater than 50% with their associated ground truth.
  - **The confidence loss:** Measures the precision of the classification task. This loss encodes how close the prediction class distribution was from categorizing the given object in the right class.

Building SSD Model on Edge

We can use an open software platform to train our object detection model using the TensorFlow Object Detection API.

Pretrained Model

- For the sake of this presentation, use MobileNetV2 which has been trained on the COCO dataset. For task of dealing with ML on drone, MobileNetV2 seems to be a good fit since it is fast and does not lose too much accuracy.

Model Settings

- Label map (JSON file mapping your ID to the name of your class)
- Training configuration file. You will have to fill the number of classes you are using and the paths to your pretrained model, labeled dataset, and images.

Packaging the Model for Deployment

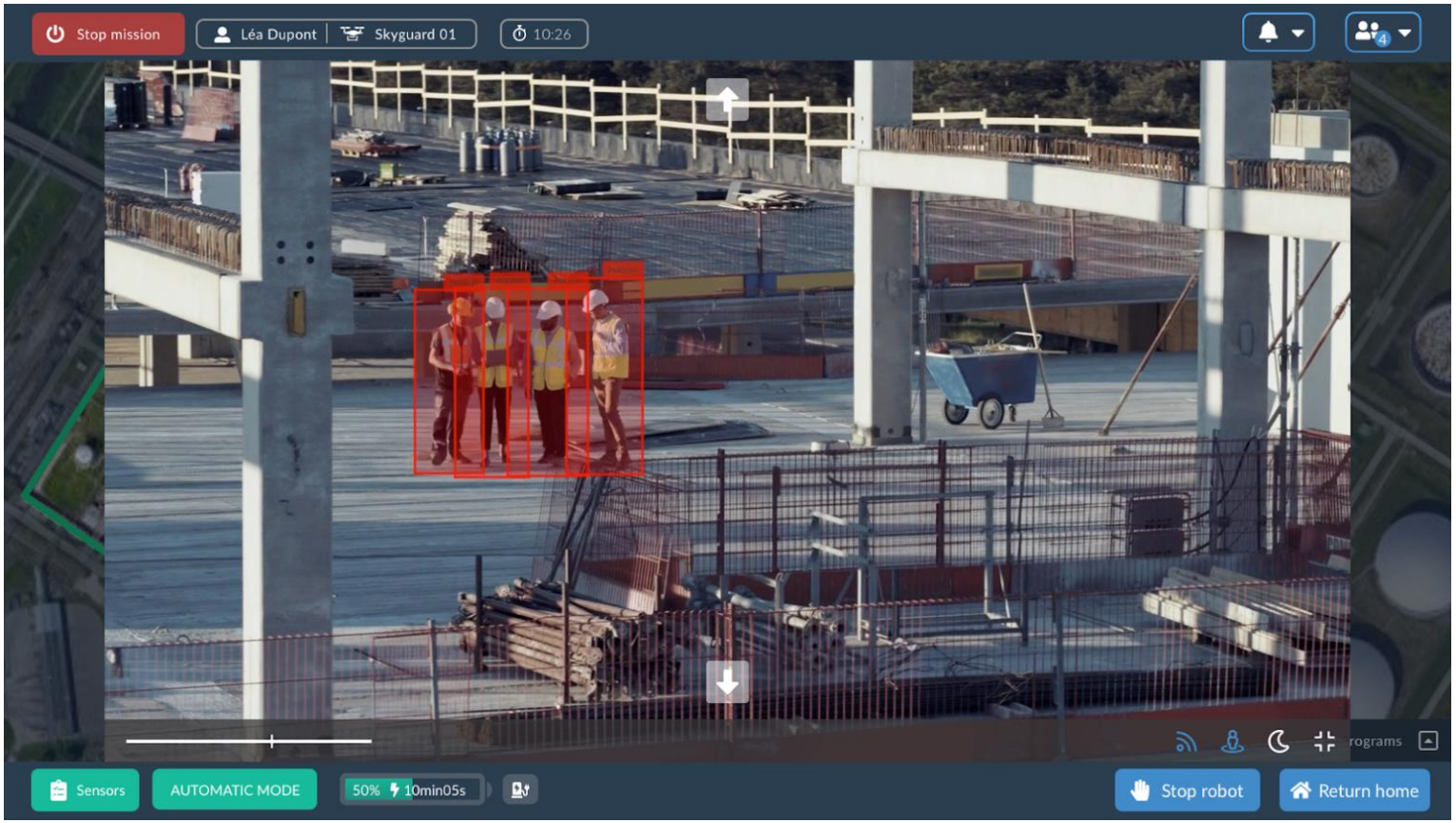
- Once the model is trained and ready to be deployed, we package all the model files together and upload them to a shared storage server. Among the different possible ways to store and transfer data, we use a Docker registry.

Make the Model Readable by the Drone

- With the object detection model trained and provisioned to software Platform, we are ready to fly, right? Well... not just yet.
- *In order to squeeze as much performance as possible out of the embedded SOC, the models first must be converted and optimized.*
- *TensorFlow and PyTorch are great frameworks for training neural networks. But when it comes to inference, they are not the best tools for the job, particularly if you want to use hardware capabilities at their maximum. For instance, specialized libraries such as Nvidia's TensorRT allow us to optimize neural networks for inference on GPU.*
- From an AI perspective, drones are flying GPUs.
- The world of machine learning is in constant evolution.
- New neural network architectures, new layers, and new needs are emerging everyday. Unfortunately, TensorRT is lagging a little bit behind TensorFlow and PyTorch.
- However, it is still possible to convert your cutting edge TensorFlow object detection model to TensorRT if you are willing to put some work into that.

Real-Time Inference by UAVIA's DroneOS

- Once the TensorRT engine model is on a drone and ready to be used, performing inference is only one click away. In any UAV application, whether on a computer, tablet, or phone, the video feed from the drone's camera is displayed in real time and features a button to toggle object detection on and off.



Thank You

Edge Computing: The “Appliance” View

Goal is to provide a “data center in a box” to push cloud computing capabilities to the edge

- Often combined with networking capabilities such as edge gateways and smart routers

Many players in this space, such as Amazon, Cisco, Dell EMC, HPE, etc.

Disconnected Operations

AWS Snowball Edge — large-scale data transfer service with an embedded computing platform (based on AWS Greengrass plus Lambda functions)

Snowball Edge Device [3]