Deploying Machine Learning Models on The Edge
Deploying Machine Learning Models on The Edge

Yan Zhang, Mathew Salvaris
Microsoft
Agenda

Concept and terminology
  IoT, AI, and ML
  Define “edge”
  Why AI/ML on edge?
  Docker

Deploy an object detection service on Azure IoT Edge
  Azure AI Platform
  Azure IoT Edge concepts
  Demo - Git repo: https://github.com/microsoft/deploy-MLmodels-on-iotedge
Device/Sensor Analytics

Edge Analytics

Cloud Analytics
AI/ML on Edge

Example: Healthcare with AI

Example: Early Prediction of Failures on Circuit Boards Assembly Line

Fault detection system makes “Pass” or “Fail” prediction on each circuit board. The goal is to minimize or remove the need for human intervention.

Why AI/ML on Edge?

It allows data is processed and analyzed near the data collection source.

Edge device sends data/message to centralized IoT platform in the Cloud.

IoT platform manages edge device including updating deployed ML model
Machine Learning in IoT

One type of analytics is to use the trained ML model to perform predictive analytics.
Machine Learning in IoT

One type of analytics is to use the trained ML model to perform predictive analytics.
Docker

https://www.docker.com

Docker is a software container service

*Instead of running the code we run the Container*

Application code, the libraries and dependencies needed to run the application

Portable, self sufficient, run anywhere

Source: https://cultivatehq.com/posts/docker/
A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

```bash
FROM python:3.5
ADD app /app
RUN pip install --upgrade pip
RUN pip install -r /app/requirements.txt

# Expose the port
EXPOSE 80

# Set the working directory
WORKDIR /app

# Run the flask server for the endpoints
CMD python -u app.py
```
Deploy an Object Detection service on **Azure IoT Edge**

- [object-detection-acv](https://github.com/microsoft/deploy-MLmodels-on-iotedge)
- [object-detection-azureml](https://github.com/microsoft/deploy-MLmodels-on-iotedge)

Link to repo: https://github.com/microsoft/deploy-MLmodels-on-iotedge
Azure IoT Edge

https://docs.microsoft.com/en-us/azure/iot-edge/

Components

- Virtual device in the Cloud (IoT Hub → IoT Edge → Device ID)
- Physical edge device
- Modules
ML Module Deployment

1. Create virtual device id in *Azure IoT Edge*
2. Configure physical edge device – install IoT runtime and associate with the virtual device id
3. Build docker image
4. Deploy
Deployment manifest file deployment.json

```json
{
  "modulesContent": {
    "$edgeAgent": { // required
      "properties.desired": {
        // desired properties of the Edge agent
        // includes the image URIs of all modules
        // includes container registry credentials
      }
    },
    "$edgeHub": { //required
      "properties.desired": {
        // desired properties of the Edge hub
        // includes the routing information between modules, and to IoT Hub
      }
    },
    "module1": { // optional
      "properties.desired": {
        // desired properties of module1
      }
    },
    "module2": { // optional
      "properties.desired": {
        // desired properties of module2
      }
    },
    ...
  }
}
```

Computer Vision Applications

Image Classification

Airplane: no
Bottle: yes
Can: no

Object Detection

Found 1 object:
Class: bottle
Position: (100,50, 200, 80)

Deep learning – neural network with multiple layers, examples include CNN, RNN, etc.

Source: https://github.com/microsoft/ComputerVision/tree/master/scenarios
Azure AI Platform

- Pre-built cognitive services
  - Pre-trained network
  - https://docs.microsoft.com/en-us/azure/cognitive-services/

- Azure custom vision service
  - Pre-trained network fine-tuned with user’s data
  - https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/

- Azure Machine Learning (AzureML)
  - Customized model architecture
  - Model management
  - Docker image creation and registration in Azure Container Registry (ACR)
Visual Intelligence Made Easy

Easily customize your own state-of-the-art computer vision models that fit perfectly with your unique use case. Just bring a few examples of labeled images and let Custom Vision do the hard work.

SIGN IN
Create an Object Detection Project
Iteration 1

Finished training on 8/2/2019, 5:43:48 PM using General domain
Iteration id: 35ca5854-5399-4d1c-8871-09154d8595a9

<table>
<thead>
<tr>
<th>Tag</th>
<th>Precision</th>
<th>Recall</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>83.7%</td>
<td>92.3%</td>
<td>96.0%</td>
</tr>
</tbody>
</table>

Choose your platform

- iOS
  - CoreML
    - iOS 11
- TF
  - TensorFlow
- ONNX
  - ONNX
    - Windows ML
- DF
  - Dockerfile
    - Azure IoT Edge, Azure Functions, AzureML
- VAIDK
  - Vision AI Dev Kit
Git repo overview

https://github.com/microsoft/deploy-MLmodels-on-iotedge

01_AzureSetup.ipynb
02_IoTEdgeConfig.ipynb
03_BuildRegisterImage.ipynb
04_DeployOnIOTedge.ipynb
05_TestPeopleDetectionModule.ipynb
06_TearDown.ipynb
Why use this tutorial?

- From the angle of data scientist – where and how the model are trained
- Jupyter notebooks → no other IDE or tools needed
- Multiple physical edge device options
- Ramp up with basic concepts - how Azure IoT Edge work with ML models
- Minimal user intervention
  - Use python-dotenv - Initialize the first notebook, and “run all” for remaining notebooks
  - Use papermill – provide option to run programmatically
Pipeline 1: object-detection-acv

Objective

- Build docker image from Dockerfile
- Register docker image in ACR
- Deploy both Image-Capture module and People-Detection-Service module
Check resources from Azure portal

After executing 01_AzureSetup.ipynb notebook
python-dotenv

For information required by the user such as subscription names, keys, passwords, resource group names, etc. Avoid manual interruption across notebooks.

```
set_key(env_path, "subscription_id", subscription_id)
set_key(env_path, "resource_group", resource_group)
set_key(env_path, "workspace_name", workspace_name)
set_key(env_path, "workspace_region", workspace_region)
set_key(env_path, "image_name", image_name)
```

mylogin@amlaksdeploy:~/notebooks/yanzrepo/demo/yanzdemo/Keras_Tensorflow$ cat .env
subscription_id="************"
resource_group="yanzdemo"
workspace_name="workspace"
workspace_region="eastus"
image_name="kerasimage"
model_version="1"

```
resource_group = get_key(env_path, \"resource_group\")
model_name = \"resnet_model\"
image_name = get_key(env_path, \"image_name\")
```
For parameterization of notebooks use `papermill`.

```
image1_name = '<image1_name>'  # e.g. image1_name = 'imgcaptureod'
image2_name = '<image2_name>'  # e.g. image2_name = 'peopledetector'
```

Example command

```bash
source activate deployment_env
echo 03_BuildRegisterImage.ipynb
papermill 03_BuildRegisterImage.ipynb out_03_BuildRegisterImage.ipynb \
    -- log-output \
    -- no-progress-bar \
    -k python3 \
    -p image1_name "img1"
    -p image2_name "img2"
```
Pipeline 2: **object-detection-azureml**

Objective
- Illustrate AzureML workspace
**AzureML**

object-detection-azureml

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>fslstnameiothub</td>
<td>IoT Hub</td>
</tr>
<tr>
<td>workspace</td>
<td>Machine Learning service workspace</td>
</tr>
<tr>
<td>workspacee87976e9</td>
<td>Container registry</td>
</tr>
<tr>
<td>workspacinsights84f105a8</td>
<td>Application Insights</td>
</tr>
<tr>
<td>workspacekeyvault89384c92</td>
<td>Key vault</td>
</tr>
<tr>
<td>workspacstorageeccda03ac7</td>
<td>Storage account</td>
</tr>
</tbody>
</table>
AzureML

object-detection-azureml

Models

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>resnet_model</td>
<td>1</td>
<td>resnet 152 model</td>
</tr>
</tbody>
</table>
Architecture

1. Docker Image
   Azure Container Registry

2. Request With Payload
   Microsoft Data Science Virtual Machine

3. Model Registry
   Azure Machine Learning

- HTTP Request
- HTTP Response
- IoT Hub
- IoT Messages
- Azure IoT Edge runtime
- GPU (for deep learning model)
- Edge Device

Input Data
Azure IoT Edge Deployment Options

Deployment process

Language: Python
IDE: Jupyter notebooks

Azure ML Python SDK - access Azure ML workspace, build image, etc.

CLI commands – deploy on IoT edge

Other deployment options

Deploy Azure IoT Edge modules from the Azure portal
Deploy Azure IoT Edge modules from Visual Studio Code
tutorial: deploy image classification model on Raspberry Pi
Summary

Edge is a concept comparing to Cloud in IoT context.

We build ML model into docker image and deploy a docker container into target compute.

We can use Azure IoT Edge to deploy a docker container on an edge device.
Thank you!