DISTRIBUTED TRAINING ON EDGE

ENGR 295B, Spring 2020

Ahmed Hambaba, Farshid Marbouti

Shafie Mukhre, Pavani Somarouthu, Brian Hu

Table of Contents

PROBLEM STATEMENTS

1. **Privacy:** ML and DL utilizes lots user's data for model training. However, these data most often are personal and sensitive data such as images, health informations and so on.
2. **Decentralized data:** Most cosumer's data is decentralized in every edge devices, while the existing centralized training is necessary for complex model training, there are cases that this overkill and unnecessary.

MOTIVATIONS

1. To preserve the privacy of user data, and therefore increase the usage or implementation of ML and DL applications.

TECHNICAL OBJECTIVES

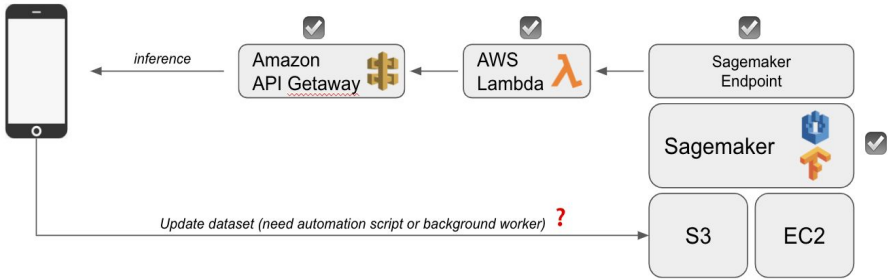
- The overall objective of the project is to create 2 ML Architectures:
    a. The first architecture is one based on Centralized Cloud Training
    b. The second is one that distributes the learning across IoT Devices

  The rest of this presentation will go into each of these approaches in vast detail.

CENTRALIZED TRAINING

Centralized training architecture

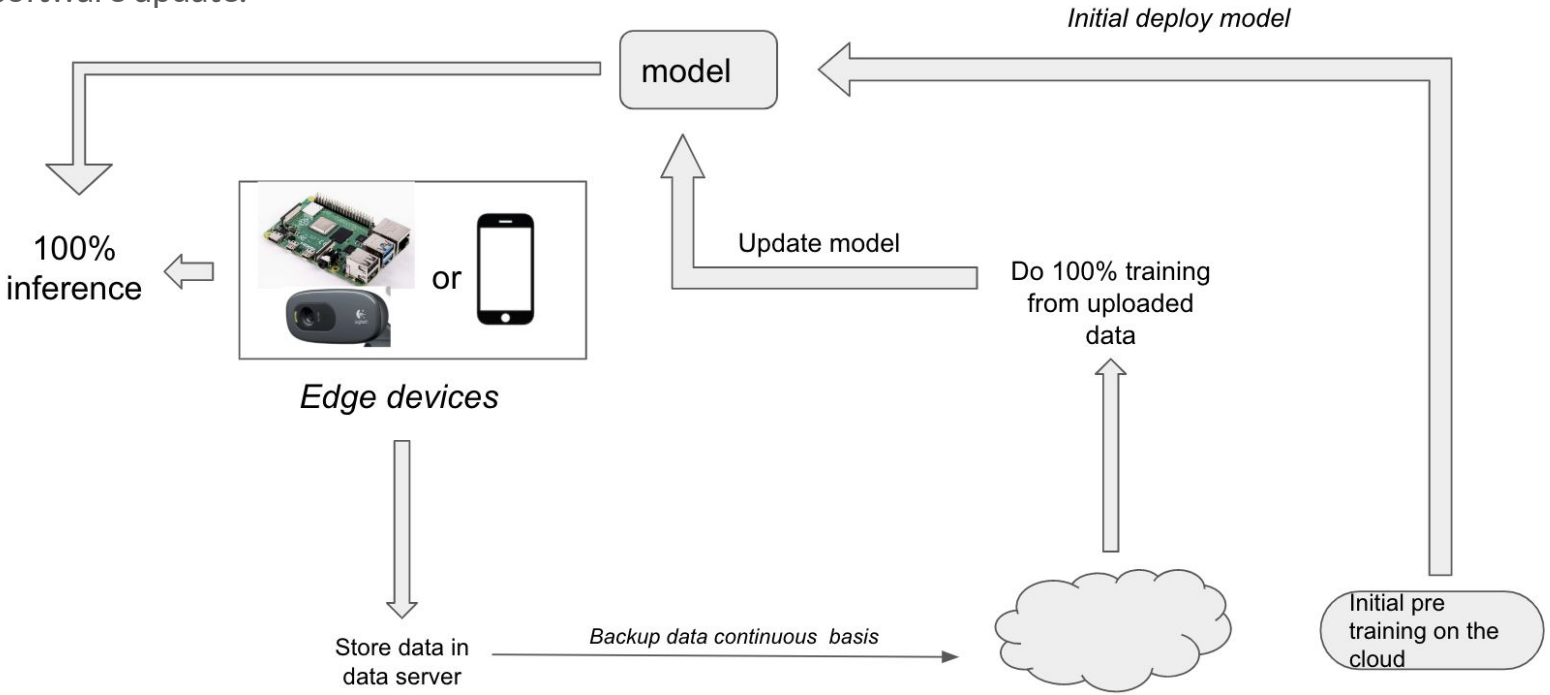
- Centralized Training is the the approach that most modern ML architectures and IoT applications use.
- The model is hosted and trained on a cloud.
- The application downloads a frozen version of the model and uses it to make inferences. The application can then send data to the cloud and download an updated retrained model.



brian

Model training

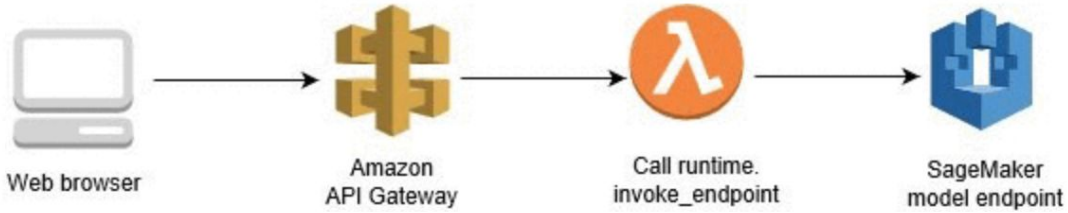
ML training is done exclusively on the cloud and the model is updated on regular basis via over-the-air software update.

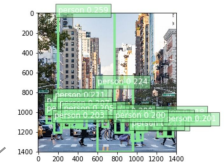


brian

CENTRALIZED TRAINING ON AWS

- Amazon Sagemaker: For access to jupyter notebook instances to build and train models.
- Amazon EC2 (Amazon Elastic Compute Cloud): Model Training is using accelerated computing provided by EC2
- Amazon S3 (Amazon Simple Storage Service): All files and datasets are download or uploaded to S3 storage
- AWS Lambda: To create a lambda function to handle the REST API request in JSON format
- Amazon API Gateway: As an intermediate communication between edge devices client and web server itself





Edge application

ie: web app

Take an image

Convert image to base64

API request

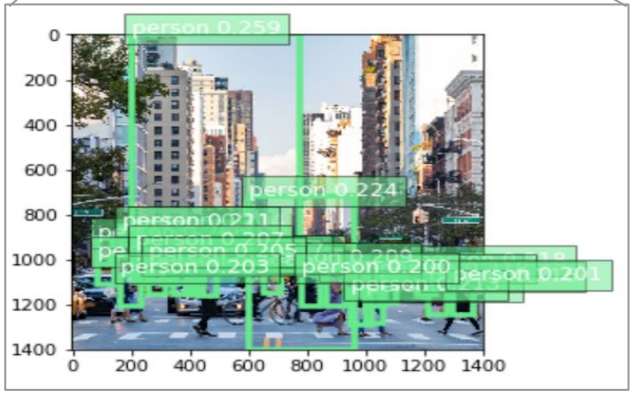
Plot detection on image

API response

Amazon API Getaway

AWS Lambda

Edge device



Example of prediction

base64

.jpg

Upload to S3 storage

Download to /tmp storage

.jpg

bytearray

Invoke Model endpoint

prediction

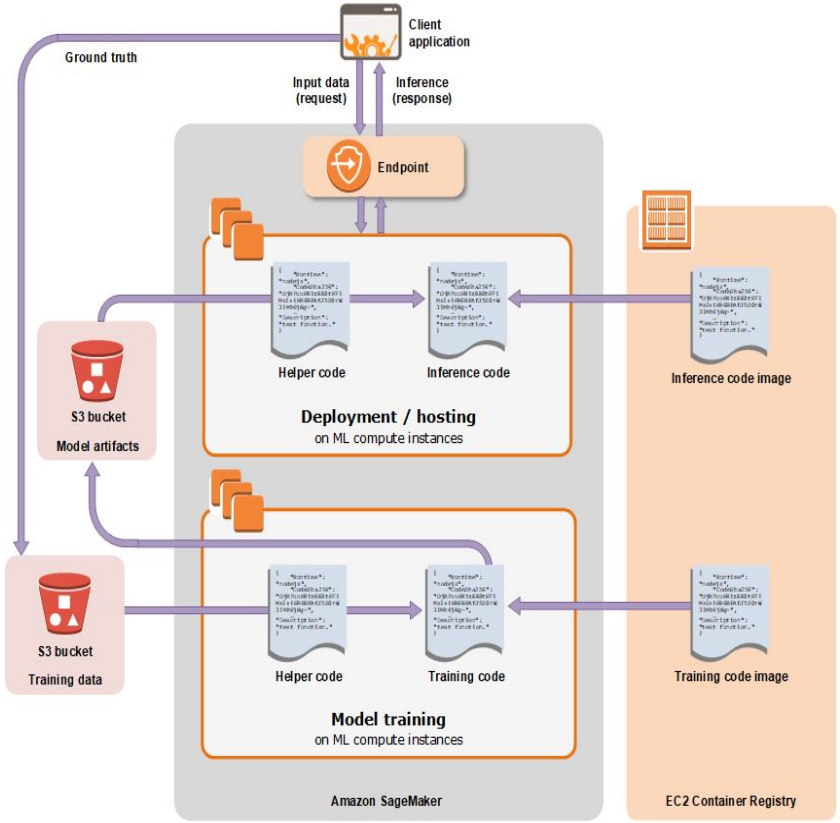
shafie

Tensorflow Training on AWS

- Sagemaker's own Object Detection Algorithm outputs image data in JSON format.
- It is difficult to deploy that output into applications so we had to do the exercise all over again in Tensorflow.

- Sagemaker has its own interface to train and deploy user-provided tensorflow code.
- Training code was taken from Tensorflow Object Detection API and adapted to Sagemaker.

- Sagemaker outputs a frozen pb model which can then be deployed to make predictions.



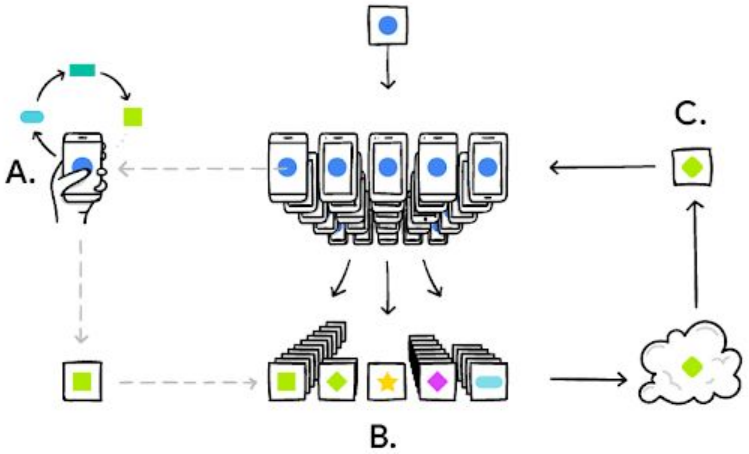
brian

DISTRIBUTED TRAINING

Distributed training options

Distributed Training is training a ML or DL model of decentralized data on multiple nodes.

- Compared distributed training technique:
  - Large Batch Training
  - Federated Learning
- Compared multiple options to do federated learning
  - Tensorflow Federated
  - Tensorflow.js
  - Trains on AWS



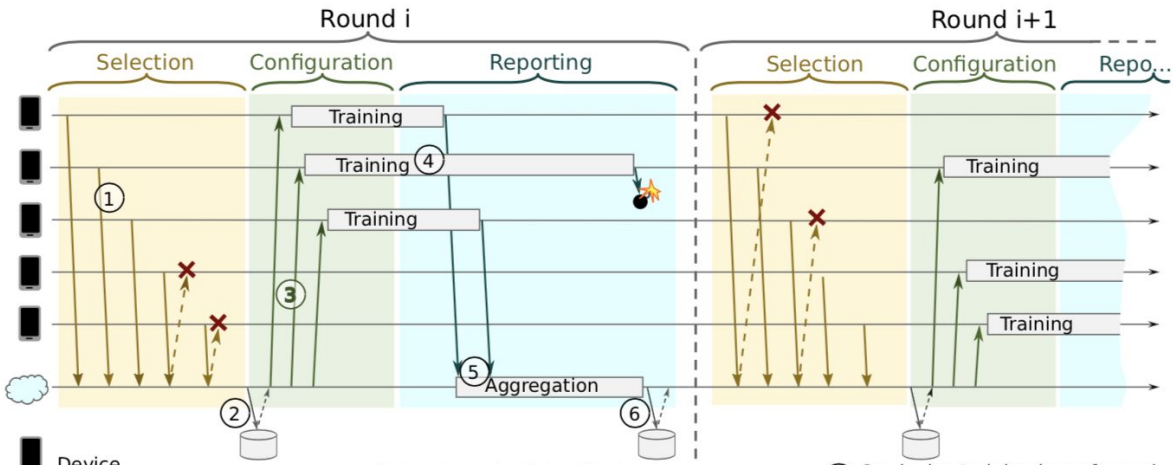
TENSORFLOW FEDERATED

TensorFlow Federated (TFF) is an open-source framework for machine learning and other computations on decentralized data.

TFF's interface is organized in two layers

- Federated Learning (FL) API - High level interface
  - Models
  - Federated Computation Builders
  - Datasets
- Federated Code(FC) API - Low level interface

Tensorflow Federated Architecture



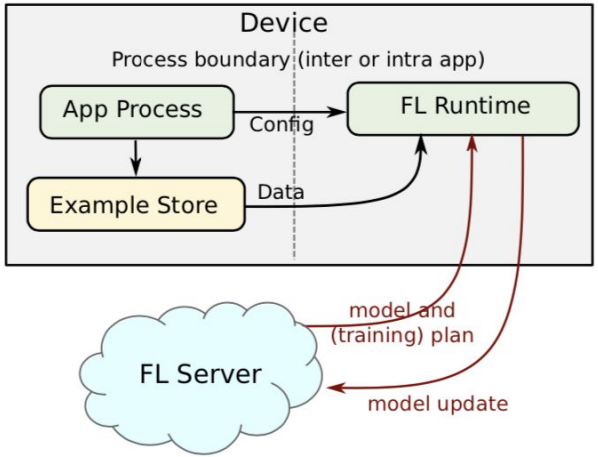


Figure 2: Device Architecture

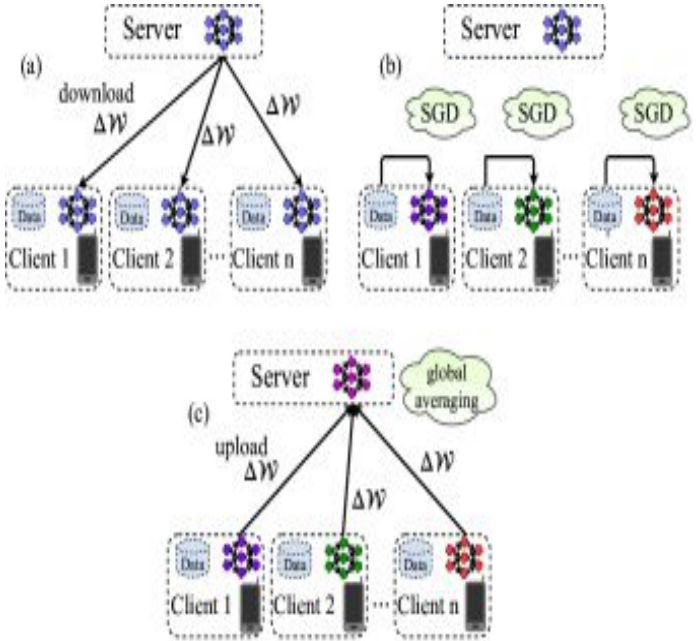
Federated Learning simulation implementation

Federated Learning Architecture

a) Clients synchronize with the server.

b) Clients compute a weight update independently based on their local data.

c) Clients upload their local weight updates to the server, where they are averaged to produce the new master model.



Federated Averaging Algorithm

Federated Averaging algorithm, combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging.

Client (On Device) training

$$w^1 = model(x, y, b, e, \eta)$$

Where, w^1 is new model trained on device

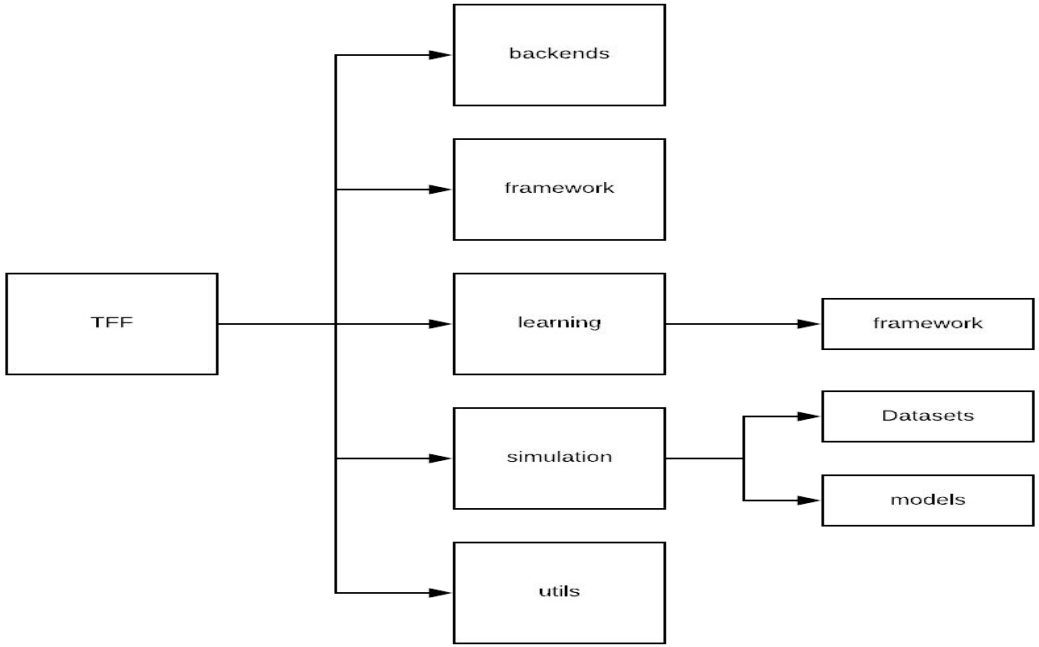
Server training

$$g=g+(n^k * w^{1k} / M)$$

Where, g is the Stochastic Gradient Descent

- n^k is the number of data points
- W^{1k} is the weights gathered from device k
- M is the sum of the number of data points

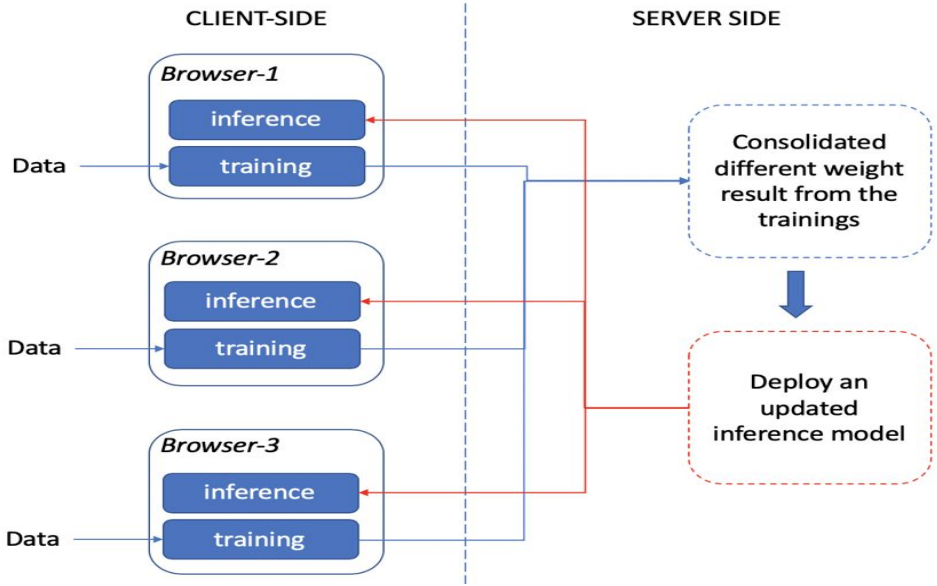
TFF Library


backends
framework
TFF
learning
framework
Datasets
simulation
models
utils

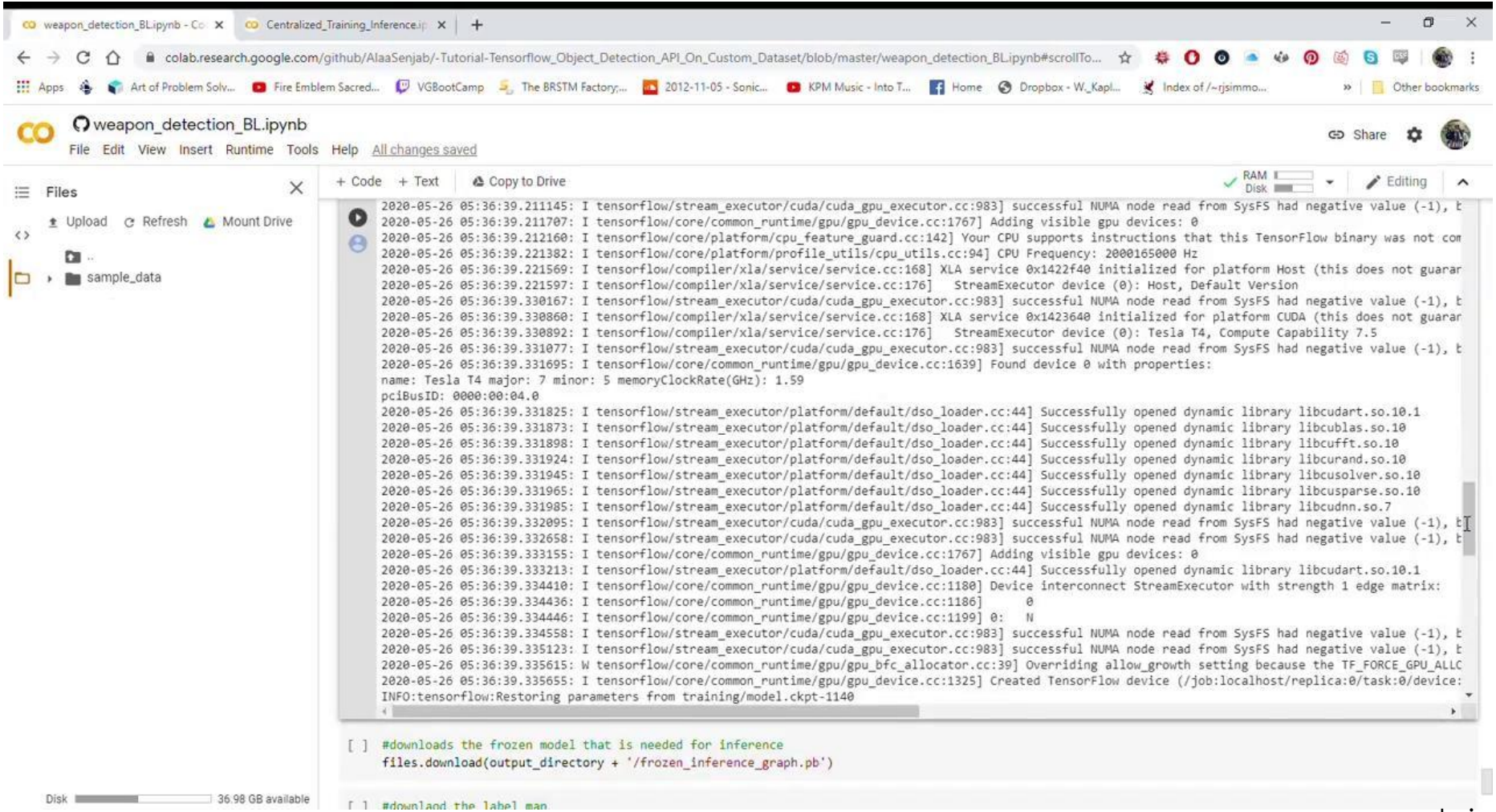
Obstacles we are facing with TFF

- Tensorflow federated is the only framework available for federated learning that that have good documentations, but it is not for production.
- Tensorflow federated simulation is limited only to LEAF dataset which is designed or orchestrated for federated learning.

Tensorflow.js



DEMO


weapon_detection_BL.ipynb - Co
Centralized_Training_Inference.ip
colab.research.google.com/github/AlaaSenjab/-Tutorial-Tensorflow_Object_Detection_API_On_Custom_Dataset/blob/master/weapon_detection_BL.ipynb#scrollTo...
Apps
Art of Problem Solv...
Fire Emblem Sacred...
VGBootCamp
The BRSTM Factory;...
2012-11-05 - Sonic...
KPM Music - Into T...
Home
Dropbox - W_Kapl...
Index of /~rjsimmo...
Other bookmarks
weapon_detection_BL.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
Files
Upload
Refresh
Mount Drive
..
sample_data
+ Code
+ Text
Copy to Drive
RAM
Disk
Editing
2020-05-26 05:36:39.211145: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.211707: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1767] Adding visible gpu devices: 0
2020-05-26 05:36:39.212160: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not com
2020-05-26 05:36:39.221382: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2000165000 Hz
2020-05-26 05:36:39.221569: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x1422f40 initialized for platform Host (this does not guarar
2020-05-26 05:36:39.221597: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
2020-05-26 05:36:39.330167: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.330860: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x1423640 initialized for platform CUDA (this does not guarar
2020-05-26 05:36:39.330892: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Tesla T4, Compute Capability 7.5
2020-05-26 05:36:39.331077: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.331695: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1639] Found device 0 with properties:
name: Tesla T4 major: 7 minor: 5 memoryClockRate(GHz): 1.59
pciBusID: 0000:00:04.0
2020-05-26 05:36:39.331825: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.1
2020-05-26 05:36:39.331873: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10
2020-05-26 05:36:39.331898: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcufft.so.10
2020-05-26 05:36:39.331924: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcurand.so.10
2020-05-26 05:36:39.331945: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusolver.so.10
2020-05-26 05:36:39.331965: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusparse.so.10
2020-05-26 05:36:39.331985: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
2020-05-26 05:36:39.332095: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.332658: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.333155: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1767] Adding visible gpu devices: 0
2020-05-26 05:36:39.333213: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.1
2020-05-26 05:36:39.334410: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1180] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-05-26 05:36:39.334436: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1186]   0
2020-05-26 05:36:39.334446: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0:   N
2020-05-26 05:36:39.334558: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.335123: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had negative value (-1), t
2020-05-26 05:36:39.335615: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:39] Overriding allow_growth setting because the TF_FORCE_GPU_ALLC
2020-05-26 05:36:39.335655: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhost/replica:0/task:0/device:
INFO:tensorflow:Restoring parameters from training/model.ckpt-1140
[ ] #downloads the frozen model that is needed for inference
files.download(output_directory + '/frozen_inference_graph.pb')
[ ] #download the label map
Disk
36.98 GB available

brian

Conclusion

1. We have implemented Centralized Cloud Training architecture using Tensorflow framework and MS COCO dataset on AWS.
2. We have implemented a simulation of Distributed Training architecture using Federated Learning using Tensorflow Federated framework and LEAF's FEMNIST dataset on Google Colab notebook.
3. We have created federated learning environment using flask as the node server and tfjs to train the model.

Appendix

An example of Lambda function python script

lambda_function

```python
import os
import io
import boto3
import json
import csv
import base64

s3 = boto3.client('s3')

ENDPOINT_NAME = os.environ['ENDPOINT_NAME']
runtime= boto3.client('runtime.sagemaker')

def lambda_handler(event, context):
    print(event)
    # if event['httpMethod'] == 'POST' :
    data = json.loads(json.dumps(event))
    name = data['name']
    image = data['file']
    image = image[image.find(",")+1:]
    dec = base64.b64decode(image + "===")
    s3.put_object(Bucket='objectdataset', Key=name, Body=dec)
    s3.download_file('objectdataset', name, '/tmp/'+name)

    file_name = '/tmp/' + name
    with open(file_name, 'rb') as image:
        f = image.read()
        b = bytearray(f)
    payload = b
    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       ContentType='image/jpeg',
                                       Body=payload)
    print(response)
    detections = json.loads(response['Body'].read().decode())
    print(detections)
    return detections

```

Execution Result

Testing API endpoint with postman