Machine Learning for IoT

Homework 3

*****DUE DATE: 21 Jan (h23:59)*****

Submission Instructions:

Each group will send an e-mail to andrea.calimera@polito.it and valentino.peluso@polito.it (in cc) with subject <ML4IOT21 GroupN> (N is the group ID). Attached with the e-mail the following files:

1. One single .zip file for each exercise, titled <HW3_exM_GroupN.zip>, where M is the exercise number and N is the group ID, containing the code and the models (more details provided later in the text). The code must use only the packages that get installed with *requirements.txt* and *rpi_requirements.txt*.
2. One-page pdf report, titled <GroupN_Homework3.pdf>, organized in different sections (one for each exercise). Each section should motivate the main adopted design choices and discuss the outcome of the exercise.

Late messages, or messages not compliant with the above specs, will be automatically discarded.

be compliant with SenML Json format in every comunication, with the exception of the output label of the big_service in ex1

Exercise 1: Big/Little Inference

Develop a Big/Little model for Keyword Spotting composed by a *Big* neural network running on your notebook and a *Little* neural network running on your Raspberry Pi. Both models must be deployed in *tflite* format.

The overall system must be composed by:

- A web service (named *big_service.py*) that receives a **raw** audio signal (sampling frequency is 16 kHz) in SenML+JSON format, runs inference using the *Big* model, and returns the output label (in JSON format).

  in the senMl I need to specify the deviceId/IP and other things, I can't omit them. Like http://xxx.xxx.xxx.xxx/, timestamp (integer, I just need seconds, not milliseconds)

  SYNOPSIS:
  ```
  python big_service.py
  ```

- A client application (*little_client.py)* that sequentially reads an audio signal from the Speech Command test-set (the signals are listed in *test_files.txt* from the *Portale*), runs inference using the *Little* model and, if needed, invokes the web service to get the prediction from the *Big* model. Define and implement the "success checker" policy that decides whether to accept the prediction of the *Little* model or invoke the web service.
  The client must print on the screen the accuracy and the communication cost measured on the Speech Command test-set. The communication cost is the sum among the sizes of the SenML+JSON strings sent to the web-service.

  a strategy for the score margin is to try to define different score margin thresholds and look at what happens with my data how to improve accuracy in big model? Adding layers (depth) and the number of filters (width) are knobs to play with

how the communication should be impelemted? When I invoke the big service, I eed to prepare a senml string (create a disctionary, and with dumps generate a string), measure its size with len() and sum the string sizes of all the data sent to the big web service

SYNOPSIS:

```
python little_client.py
```

EXAMPLE OUTPUT:

```
Accuracy: 93.125%
Communication Cost: 4.405 MB
```

this is of the big_little model (with the success checker policy)

Identify and adopt the most suitable protocol (between REST and MQTT) to implement the communication among the different applications. Motivate your choice.

- Write a Python script to train the *Big* and *Little* models. For the training, use the signals listed in *train_files.txt* and the labels order in *labels.txt* from the *Portale*.

  the biggest part in the senML is the audio array)

  SYNOPSIS:

  ```
  python train.py --version <big|little>
  ```

  there is a tradeoff on the number of times I can call the big model becasue I need to respect the communicaiton cost

  The models must be compliant with the following constraints (measured on the **test-set**):
  - ✓ *Big/Little* Accuracy > 93%
  - ✓ *Little* TFLite Size < 20 kB — inferent time is measured with the same script of hw2
  - ✓ *Little* Total Inference Time < 40 ms — including preprocessing
  - ✓ Communication cost < 4.5 MB — measure size of the senMl string (when I have to send it). Make a dictionary, dump with json, then measure size using len() method of python. Measure size everytime I invoke the big service

  do not consider the size (that is negligeble) of the big model response. To get MB (from bytes) divide by 2^20

- The final submission should contain the following files:
  1. The *big_service.py* script.
  2. The *little_client.py* script.
  3. The *train.py* script.
  4. The *Big/Little* models in *tflite* format (*big.tflite* and *little.tflite)*. If you have compressed the TFLite files with *zlib,* append .zlib to the filenames.
  5. Any other .py file needed for the correct execution of the previous commands (if any).

  Organize the files in two different folders, named *notebook* and *rpi* respectively, depending on the target device where the commands should run.

- In the report, briefly describe the *Big/Little* models and the “success checker” policy adopted.

Exercise 2: Cooperative Inference

- Develop a client application that receives a **pre-processed** audio signal, classify its content, and send the output of the last neural network layer (in JSON format). Run different *N* (*N* is an optimization parameter, *N* > 1) instances of the same application, each of them running a different prediction model (in *tflite* format). The application instances could potentially run on different devices (suggestion: launch them on different shells on your notebook). To run the application, develop a Python command with the following specifications:

In ensambles I use different models for predicitons instead of one, this is similar. In this case, all neural networks are fed with the same data and get multiple output predictions. Define a policy that combines partial prediciotns from the different models in order to get the final lables.

for SenML organize data in a list of events and write and read data in a proper way

when I send the preprocessed audio, since base.64.b64encode(sample) flattens the tensor to 1d, I can add an additional field to the SenML json so that I can mitigate this problem (but the other mandatory fields must be there) the inference code is the same for each model

SYNOPSIS:

```
python inference_client.py --model <model_path>
```

where *model_path* is the path to the *tflite* model.

- On your Raspberry Pi, develop a client application that sequentially read an audio signal from the Speech Command test-set (the signals are listed in *test_files.txt* from the *Portale*), apply some pre-processing, send the pre-processed signal (in SenML+JSON format) to the inference clients, and retrieve the neural networks outputs. Define and implement a cooperative inference policy to compute the final prediction combining the outcomes of the different inference clients. To run the application, develop a Python command with the following specifications:

  SYNOPSIS:

  ```
  python cooperative_client.py
  ```

  The command should print on the screen the classification accuracy measured on the Speech Command test-set.

  EXAMPLE OUTPUT:

  ```
  Accuracy: 94.375%
  ```

  print accuracy

  Identify and adopt the most suitable protocol (between REST and MQTT) to implement the communication among the different clients. Motivate your choice.

  The application must reach an Accuracy > 94% (measured on the **test-set**).

- The final submission should contain the following files:
  1. The *inference_client.py* script.
  2. The *cooperative_client.py* script.
  3. The *N tflite* models (named as *1.tflite*, *2.tflite*, …, *N.tflite*).
  4. The training script used to generate the *tflite* models (use the signals listed in *train_files.txt* and the labels order in *labels.txt* from the *Portale*). The training command should meet the following specifications:

     SYNOPSIS:

     ```
     python train.py --version <ID>
     ```

     where <*ID*> is an integer number ranging from 1 to *N*.

  5. Any other .py file needed for the correct execution of the previous commands (if any).

  Organize the files in two different folders, named *notebook* and *rpi* respectively, depending on the target device where the commands should run.

- In the report, briefly describe the models used for inference and the cooperative policy adopted.

  also describe the strategy to differentiate the models

I need to provide more than 1 model and to develop the copeerative policy, not possible to provide a single models that makes the prediction.
Suggerimento: N is much less than 10

how can we differentiate the neural netowkrs of a cooperative sysem? Type and number of layers (naive), search for the state of the art techniques from the machine learning world.