Efficient Computing for Artificial Intelligence

Homework 2

DUE DATE: 7 Jan (h23:59)

Submission Instructions:

Each group will send an e-mail to `daniele.jahier@polito.it` and `valentino.peluso@polito.it` (in cc) with subject **ECAI26 Group**`N` (replace `N` with the group ID).

Attach a single ZIP archive named **HW2_Group**`N`**.zip** (replace `N` with your group ID) containing:

1. The code deliverables specified in the text of each exercise.
2. A one-page PDF report named **report.pdf**.

Late messages or messages not compliant with the above specifications will be **automatically discarded**.

Smart Hygrometer – Version 2.0

Goal: Develop a more advanced prototype of the **smart hygrometer** by integrating a Voice User Interface (VUI) based on an **efficient KWS pipeline**.

System Description

The system includes:

- A **Raspberry Pi** connected to a **DHT-11 sensor** to measure temperature and humidity.
- Data storage in a **Redis Cloud database** using the **redis-py API**.
- Voice commands **"up"** and **"stop"** to enable or disable data collection.
- A **custom KWS classification** pipeline running locally on the Raspberry Pi for command recognition.

1. Training & Deployment of a "Up/Stop" Keyword Spotter

Implementation Steps

On **Deepnote**, create a Python notebook to train and optimize a classification model for "up/stop" keyword spotting on the the *Mini Speech Commands* dataset.

The Python notebook must include the following steps:

- Develop a data pipeline that processes of "stop" and "up" keywords from the MSC train, validation, and test splits, mapping "stop" to 0 and "up" to 1.
- Define the model architecture and the training, optimization, and testing flow.
- Generate and save the ONNX version of the feature extraction module and the trained model.

The model must meet all the constraints below:

1. Accuracy measured on the test set > **99.4%**
2. Total ONNX Size (Feautre Extraction + Model) < **300 KB**
3. Total Median Latency (Feautre Extraction + Model) on the RPI < **5 ms**

The ONNX model can be provided in ONNX format or ZIP format. The Total ONNX Size must be measured on the selected format (1 KB = 1024 bytes).

Instructions:

- On **Deepnote**, run the notebook `8.1 - HW2 - Evaluation of Accuracy and Size` to measure Accuracy and Total ONNX Size.
- On the **RPI**, run the script `hw2_inference_latency.py` to measure Total Media Latency.

2. System Integration

On the RPI, you will implement and test the integration of the Smart Hygrometer system. The system will continuously collect temperature and humidity data from the DHT-11 sensor, store the measurements in a cloud Redis database, and control the data collection process using a VUI powered by the custom KWS pipeline.

1. **Create a script named `hygrometer.py`.** This script will control the full system workflow: reading temperature and humidity data, sending it to Redis, and enabling or disabling data collection through voice commands. The script must be executable from the command line and accept the following arguments using `argparse`:
   - `--host (str)`: Redis Cloud host.
   - `--port (int)`: Redis Cloud port.
   - `--user (str)`: Redis Cloud username.
   - `--password (str)`: Redis Cloud password.
2. **Initialize the system.** At startup:
   - Initialize the DHT-11 sensor to collect temperature and humidity data.
   - Establish a connection to the Redis Cloud database using the `redis-py` API.
   - Load the ONNX files of the feature extraction module and the neural model for voice command recognition.
   - Set the system state to *disabled* (data collection off).
3. **Configure the audio acquisition.** The VUI must continuously record audio in the background using the USB microphone. Configure the recording parameters as follows:
   - # Channels: 1
   - Bit depth: 16 bits (`int16`)
   - Sampling rate: 48 kHz
4. **Implement command recognition.** Every second, the VUI must analyze the most recent one-second audio window using the custom KWS pipeline to detect command words. The processing pipeline should include the following steps:
   - Convert the recorded audio to a PyTorch tensor of type `float32`.
   - Change the data layout from channel-last to channel-first format.
   - Normalize the waveform values to the range $[-1, 1]$.
   - Downsample the signal to 16 kHz.
   - Convert the downsampled signal to a `numpy` array.
   - Insert the batch axis to the array.
   - Feed the resulting array to the custom KWS pipeline.
   - Compute the softmax of the model output to translate the prediction into a probability distribution.
5. **Implement the control logic.**
   - If the predicted keyword is `''up''` with probability $> 99.9\%$, enable data collection.
   - If the predicted keyword is `''stop''` with probability $> 99.9\%$, stop data collection.
   - If the top-1 probability (regardless of the predicted label) is $\leq 99.9\%$, remain in the current state.

   The VUI must run continuously in the background.

6. **Implement data collection and upload.** When data collection is enabled:
   - Measure temperature and humidity every 5 seconds using the DHT-11 sensor.
   - Upload the collected values to Redis following the naming rules defined in *LAB2 – Exercise 1c*.
   - Send data to Redis.
7. **Testing and validation.** Before submission:
   - Verify that the script starts correctly from the terminal and that all arguments are parsed without errors.
   - Confirm that the DHT-11 sensor measures temperature and humidity, and Redis receives data only when the system is enabled.
   - Test multiple `‘‘up’’` and `‘‘stop’’` commands to ensure correct state transitions.

3. Report

Prepare a one-page PDF report using the LaTeX template `Homework2/report_template.tex` available in the Material section of the course portal. The report should present your results and answer the discussion questions as described below:

1. Describe the methodology adopted to build the KWS pipeline compliant with the target constraints.
2. Add a table reporting the feature extraction hyper-parameters of the final solution. Motivate why the selected configuration enables meeting both accuracy and latency constraints.
3. Add a table reporting the training hyper-parameters of the final solution.
4. Describe the model architecture and the adopted optimizations.
5. Add a table reporting Accuracy (%), Total ONNX Size (KB), and Total Median Latency (ms) of the final solution.
6. Comment on the reported results.

4. Evaluation

Each homework part contributes separately to the final grade as reported in Table 1. Maximum points differ for students enrolled in the **Data Science and Engineering (DSE)** and **Computer Engineering (CE)** programs.

Exercise Part	**DSE (max points)**	**CE (max points)**
1. Training & Deployment	3*	6†
2. System Integration	1	1
3. Report	2	2
Total	**6**	**9**

* 1 point per constraint met
† 2 points per constraint met

Table 1: Evaluation scheme for DSE and CE students.

5. Deliverables

Submit a single ZIP archive named: **HW2_GroupN.zip** containing:

1. `training.ipynb` – Python notebook with the training, optimization and conversion code to generate the ONNX files of the KWS pipeline.
2. `hygrometer.py` – Smart hygrometer code for Raspberry Pi.

3. `GroupN_frontend.onnx` – ONNX file of the feature extraction module (replace *N* with the group ID).
4. `GroupN_model.onnx` or `GroupN_model.onnx.zip` – The ONNX neural model of 1. The ONNX model must be named `GroupN_model.onnx`, if provided in ONNX format, or `GroupN_model.onnx.zip`, if provided in ZIP format (replace *N* with the group ID).
5. `report.pdf` – One-page report.

During the evaluation, Accuracy, Total ONNX Size, and Total Median Latency will be measured on the submitted files.

Note: Submissions that do not follow the required filenames or format **will be discarded without evaluation**.