Efficient Computing for Artificial Intelligence

Homework 1

DUE DATE: 26 Nov (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 **HW1_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 1.0

Goal: Develop the first prototype of a **smart hygrometer** controlled through a **Voice User Interface (VUI)**.

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 **Speech-to-Text (STT)** model from the *Whisper* family running locally on the Raspberry Pi for command recognition.

1. Model Evaluation

Evaluate the **accuracy** and **latency** of different Whisper model versions (*tiny*, *base*, *small*, *medium*, *large*, *largev2*).

Implementation Steps

On Deepnote

Follow the steps below to evaluate the classification accuracy of different Whisper model versions using the *Mini Speech Commands* dataset.

This dataset contains a total of 8,000 audio samples distributed across eight spoken keywords: `down`, `no`, `go`, `yes`, `stop`, `up`, `right`, and `left`, with 1,000 samples per keyword. Each audio sample is recorded at a sampling rate of 16 kHz using 16-bit integer (`int16`) resolution. The duration of the samples varies, with the longest sample lasting up to 1 second. The dataset is split into 6,400 training samples, 800 validation samples, and 800 test samples.

1. **Open the evaluation notebook** `5.1 - HW1 - Accuracy Evaluation.ipynb`**.** The notebook already contains the evaluation loop for measuring the model accuracy, but

it does not include the code that loads and preprocesses the dataset. You will complete this part by implementing a custom dataset class.

2. **Implement the dataset class `MSCDataset`.** Create a new Python file named `msc_dataset.py` and define a class `MSCDataset` that inherits from `torch.utils.data.Dataset`. This class must:
   - Take as input:
     - **root folder path** (`str`) where the dataset is stored.
     - **ordered list of labels** (`list of str`) specifying the target keywords. This list will be used to map textual labels to integer indices consistently.
   - Load each audio file (raw WAV data) along with its corresponding sampling rate and label.
   - Convert the waveform into a PyTorch tensor.
   - Map the textual label to an integer using the provided ordered label list.
   - Return, at each iteration, a dictionary with the following structure:

```
{
    "x": <audio data (torch.Tensor)>,
    "sampling_rate": <sampling rate (int)>,
    "label": <integer label (int)>
}
```

   The goal is to make the dataset class fully compatible with the evaluation loop already implemented in `5.1 - HW1 - Accuracy Evaluation.ipynb`.
3. **Test your implementation.** Open and run the notebook `HW1 Dataset Test.ipynb` to verify that your `MSCDataset` class correctly loads, processes, and returns samples in the expected format.
4. **Run the model evaluation.** Once your dataset class works correctly, return to the notebook `5.1 - HW1 - Accuracy Evaluation.ipynb`. Execute the cells to evaluate the classification accuracy of the available Whisper model versions on the provided test set located in the folder `/tmp/msc-test`. Record the accuracy values (in %) for each model version.

On the Raspberry Pi

- Evaluate the inference latency of the different Whisper model versions on the Raspberry Pi using the script `4.5-lab2_edge_inference.py`.
- Record the median and standard deviation (in seconds, with two decimal digits) for each model version.

2. System Integration

On the Raspberry Pi, you will implement and test the integration of the Smart Hygrometer system. The system will integrate multiple components: it 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 a pretrained STT model.

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 pretrained Whisper `tiny` 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 Whisper `tiny` model 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.
    - Remove the channel dimension.
    - Feed the resulting tensor to the Whisper pipeline.
    - Transcribe the output, removing spaces and punctuation.
5. **Implement the control logic.**
    - If the keyword `‘‘up’’` is detected, enable data collection.
    - If the keyword `‘‘stop’’` is detected, disable data collection.
    - Otherwise, keep 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 `Homework1/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. Present the results in a table including:
    - **Model Version** (tiny, base, small, medium, large, large-v2)
    - **Memory** (in MB)
    - **Accuracy** (% with two decimal digits)
    - **Latency** (median and standard deviation in seconds with two decimal digits)

2. **Discussion:**
   - Evaluate the suitability of the Whisper-based pipeline for real-time, command-based VUIs.
   - Discuss pros and cons of replacing Whisper with a custom, task-specific model.
   - Describe the **training pipeline design** you would implement for such a custom model. Identify all the key steps involved in the process and briefly describe each step.

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. Model Evaluation	2	3
2. System Integration	2	3
3. Report	2	3
Total	**6**	**9**

Table 1: Evaluation scheme for DSE and CE students.

5. Deliverables

Submit a single ZIP archive named: **HW1_GroupN.zip** containing:

1. `msc_dataset.py` – PyTorch dataset implementation (used in Deepnote)
2. `hygrometer.py` – Smart hygrometer code for Raspberry Pi
3. `report.pdf` – One-page report

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