Harvard CS 2420: Computing at Scale (Fall 2025) **(Due: 11/5/2025 at 11:59PM EST)**

Problem Set 4 (PSet 4): Federated Learning and Text-to-Image Generation

Instructor: H.T. Kung

Team: Name-1, Name-2, Name-3, Name-4

Please include the full names of all team members in your submission (See Section 7).
Please use Google Colab as your coding environment for Problem Set 4.

1 Exploring Federated Learning (FL)

Part 1.1 (5 points)

1. In *Code Cell 1.3*, implement the function `iid_sampler` to generate **IID** (independent and identically distributed) samples from the CIFAR-10 training set. We will use this function to generate training subsets for multiple devices in **PART 1.2**.
2. In *Code Cell 1.4*, create a single device using the function `create_device`. This device should have 10% of the CIFAR-10 training set, obtained using `iid_sampler`.
3. Train the model for 1000 epochs using the specified parameters in *Code Cell 1.4* (similar to Assignment 1). The number of epochs is 10× greater due to the single device having only 10% of the data. Plot the test accuracy (`device['test_acc_tracker']`) over and comment on the classification accuracy compared to using 100% of the dataset as in Assignment 2.

(Solution)
1. [See Colab.]
2. [See Colab.]
3.

Part 1.2 (10 points)

1. In *Code Cell 1.5*, implement `average_weights`. We have provided test code you can use to validate your implementation. This test code will also be useful for the full implementation of Federated Learning in **PART 1.3**.
2. In *Code Cell 1.5*, implement the `get_devices_for_round` function. Try multiple `device_pct` settings to ensure that it is working properly.

(Solution)
1. [See Colab.]
2. [See Colab.]

Part 1.3 (10 points)

1. In *Code Cell 1.6*, train a global model via federated learning. Much of the code has been given to you, but you will need to fill in the parts using calls to the functions you wrote above.

2. Graph the accuracy of the global model over 100 rounds. Discuss the accuracy difference between the global model trained here and the individual local model you trained in **PART 1.1**.

(Solution)

1. [See Colab.]
2.

2 Non-IID Federated Learning and Fairness

Part 2.1 (10 points)

1. In *Code Cell 2.1*, implement the `noniid_group_sampler` function to generate **non-IID** samples from the CIFAR-10 training set. As input, the function should take the dataset and number of training samples each device should be assigned. As in `iid_sampler` you implemented previously, the function should return a `dict` where each key is a device ID number and each value is a `set` of the indices of training samples in the dataset assigned to that device. You may want to have the function return other data as well, depending on how you implement functions in later parts of the assignment. We have provided the mapping that indicates which classes are mapped to each group. Within a given group, you should sample the data in an IID fashion.

(Solution)

1. [See Colab.]

Part 2.2 (5 points)

1. In *Code Cell 2.2*, implement the `get_devices_for_round_GROUP` function to generate a list of devices that will participate in each round of federated learning. The function should, at minimum, take as input 1) the list of all devices, and 2) how many devices from each group should participate in a given round. It should return a list of devices that will participate in a given round. You may want to add additional parameters to the function depending on your implementation strategy.

(Solution)

1. [See Colab.]

Part 2.3 (5 points)

1. In *Code Cell 2.3*, implement the `cifar_noniid_group_test` function to create a test dataset for each group. It should take the full CIFAR-10 test dataset as input, and return a dictionary where each key is a group ID, and each value is a `set` of the indexes for all test samples for that group.
2. In *Code Cell 2.3*, implement the `test_group` function to output the per-group classification accuracy of the global model.

(Solution)

1. [See Colab.]
2. [See Colab.]

Part 2.4 (10 points)

1. In *Code Cell 2.4*, train a global model via federated learning for the group-based non-IID setting. Much of the code has been given to you, but you will need to fill in the parts using calls to the group-based, non-IID functions you wrote above. You will likely be able to re-use parts of the code you wrote in **Part 1.3**.
2. Graph the per-group test accuracy over 100 rounds in the **Fair Device Participation** scenario. Each group should have its own line in the graph.

3. Graph the per-group test accuracy over 100 rounds in the **Unfair Device Participation** scenario. Each group should have its own line in the graph.
4. Describe the differences you see between the two scenarios. How can you explain what you are seeing?

(Solution)

1. [See Colab.]
2.
3.
4.

3 Quantization of Local Models for Reduced Communication Cost

Part 3.1 (5 points)

1. In *Code Cell 3.1*, implement a function that converts full-precision numbers in the range $[0, 1]$ into n-bit fixed-point numbers. If your implementation is correct, it should return *'Output of Quantization Matches!'*.

(Solution)

1. [See Colab.]

Part 3.2 (10 points)

1. In *Code Cell 3.2*, implement `dorefa_g(w, nbit, adaptive_scale=None)` using the **stochastic quantization function** shown above. Again, if your implementation is correct, it should return *'Output of Quantization Matches!'*.

(Solution)

1. [See Colab.]

Part 3.3 (10 points)

1. In *Code Cell 3.3*, run federated learning with the following two quantization settings (bit widths): `nbit=16` and `nbit=4`. Graph the accuracy of the global models over 100 rounds for the different bit widths: 32-bit (the full-precision baseline you ran previously), 16-bit, and 4-bit.
2. Discuss the accuracy difference between the global models across the three different bit width settings: 32-bit, 16-bit, and 4-bit. (100 words maximum)

(Solution)

1. [See Colab.]
2.

4 Gradient Inversion Attacks to Federated Learning

Part 4.1 (15 points)

1. Implement three defense strategies (i.e., noise injection, pruning, and quantization) defense_method in *Code Cell 4.1*.
2. Run the 4 experiments (i.e., Baseline, Noise, Pruning, and Quantization). For each experiment, you should report: (1) the final inversion result measured by $\|\text{image} - \text{recovered\_image}\|_2$ and (2) the optimization process of inversion using *Code Cell 4.2*.
3. Compare and discuss the effectiveness of the 3 different defense strategies. (100 words maximum)

(Solution)

1. [See Colab.]
2.
3.

5 Fine-Tune a Large Pre-Trained Stable Diffusion Model to Learn a New Concept

Part 5.1 (15 points)

1. In *Code Cell 5.17*, fine-tune a small subset of U-Net layers in Stable Diffusion v1.5 (Q, K, V projections and the `to_out` layer).
2. Report the number of *tunable parameters* from your implementation (use the value printed by `print_params()`).

(Solution)

1. [See Colab.]
2.

Part 5.2 (10 points)

1. From *Code Cell 5.23* to *Code Cell 5.30*, generate comparison images with the `sks` token and without it using the provided prompts. Then include those images in Figure 1 by replacing each placeholder with the corresponding image you generated.

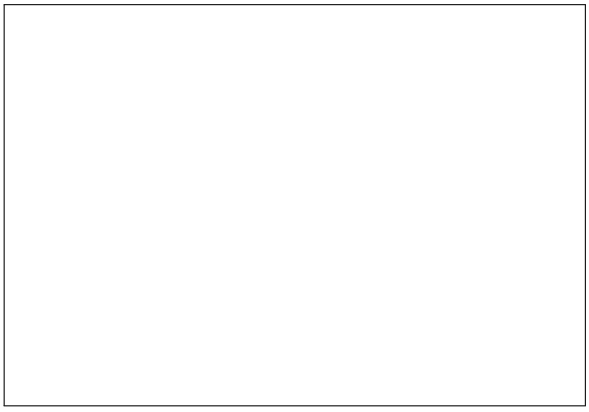
(Solution)

(a) a photo of `sks` person next to the lake

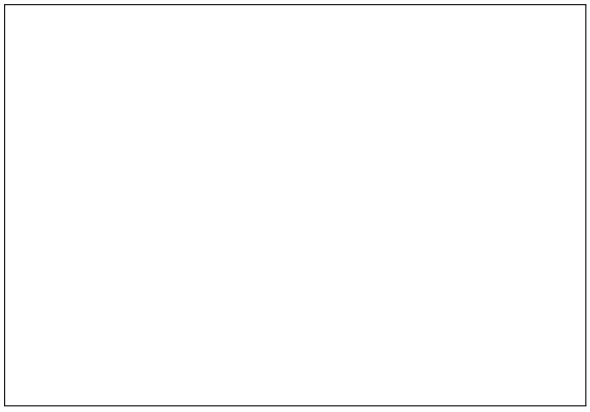
(b) a photo of person next to the lake

(c) a photo of `sks` person in the office

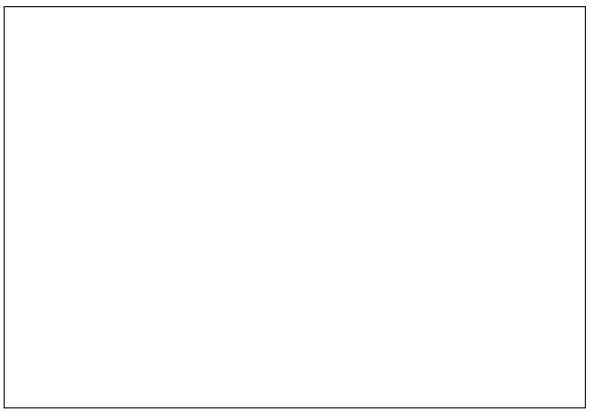
(d) a photo of person in the office



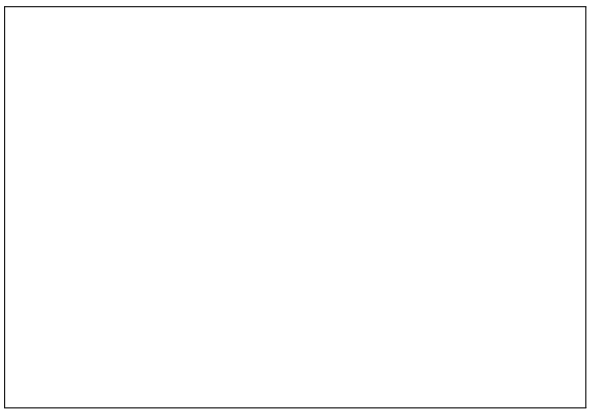
(e) a photo of `sks` person at cafe



(f) a photo of person at cafe



(g) a photo of `sks` person on the yard



(h) a photo of person on the yard

Figure 1: Part 5.2: Comparison with and without the `sks` token on the fine-tuned Stable Diffusion model.

6 Training-Free Multi-Prompt Generation with Varied Resolutions (Inference-Time)

Part 6.1 (10 points)

1. In *Code Cell 6.9*, implement PART 6.1 (a) - (h) to modify the behavior of the cross-attention layers.

(Solution)
1. [See Colab.]

Part 6.2 (5 points)

1. Include the generated images from *Code Cells 6.11 to 6.13* here.

(Solution) [Include generated images here]

Part 6.3 (10 points)

1. In *Code Cell 6.14*, implement PART 6.3 (a) - (h) to modify the behavior of the cross-attention layers.

(Solution)
1. [See Colab.]

Part 6.4 (5 points)

1. Include the generated images from *Code Cells 6.16 to 6.18* here.

(Solution) [Include generated images here]

7 What to Submit

Your submission should be a `.zip` archive with a `CS2420_FA25_PSet4` prefix followed by your full name. The archive should contain:

- PDF write-up
- Assignment code
- Text files or PDFs containing the complete outputs (e.g., ChatGPT logs) of all generative AI tools used.

Example filename: `CS2420_FA25_PSet4_Name1__Name2__Name3__Name4.zip`

Write-up

Written responses should be contained within a single PDF document. (LaTeX is highly recommended!) Each response or figure should clearly indicate which problem is being answered.

Code

You should include **all** files that were provided, but with the changes you made.