Homework 1: System analysis of a simple mapping.

1 Problem statement and notation

Let $d \in \mathbb{N}$. For each layer index $j \in \{1, \dots, J\}$, let

$$f_j : \mathbb{R}^d \to \mathbb{R}^d$$

be a learnable map with parameters θ_j (e.g. an MLP, a convolutional block, or a transformer block). We study two families of *state-update* maps acting on a split state $(a_j, b_j) \in \mathbb{R}^d \times \mathbb{R}^d$:

$$\textbf{Type (a)} \qquad F_j(a_j, b_j) = (a_{j+1}, b_{j+1}) := \big(a_j + f_j(b_j),\ b_j\big), \tag{1}$$

$$\textbf{Type (b)} \qquad F_j(a_j, b_j) = (a_{j+1}, b_{j+1}) := \big(a_j + f_j(b_j),\ a_j\big). \tag{2}$$

Define the full network map by composition

$$\Phi := F_J \circ F_{J-1} \circ \cdots \circ F_1, \qquad (a_{J+1}, b_{J+1}) = \Phi(a_1, b_1).$$

Let $\ell : \mathbb{R}^{2d} \to \mathbb{R}$ be a (differentiable) loss, with potentially some trainable parameters, and define the objective

$$L(a_1, b_1; \theta_1, \dots, \theta_J) := \ell(a_{J+1}, b_{J+1}).$$

Remarks. For each question, clearly state any assumptions you use. Figures may be hand-drawn (scanned) or produced with TikZ, but they must be legible and properly labeled.

Submission format. Submit *one* self-contained PDF report of at most **3 pages** (including figures, excluding code listings if you place them in separate files, fontsize 11pt and margin 1.5cm). In addition, submit two code files: `a.py` and `b.py`, each **under 500 lines**, written in **pure PyTorch** (no external ML libraries). Your code should reproduce the numerical experiment(s) requested in the assignment and print the key quantities you report. Drop an email to `edouard.oyallon@cnrs.fr` with tag *[MVA] HW1* in the title.

Reproducibility. Your scripts must run end-to-end from the command line without manual intervention, set a random seed, and include all hyperparameters (e.g. d, J, learning rate, number of steps) at the top of the file or via simple CLI arguments.

2 Questions

Q1. Block-diagram and Jacobians.

(a) Draw a block diagram ("computational graph") for one layer of type (a) and one layer of type (b). Your diagram should explicitly show: the inputs (a_j, b_j), the computation of $f_j(b_j)$, the addition, and the outputs (a_{j+1}, b_{j+1}). How would you initialize (a_1, b_1)?

(b) Compute the Jacobian matrices $DF_j(a_j, b_j)$ for type (a) and type (b), written in 2×2 block form. Compute $\det(DF_j)$ and discuss what this implies for volume preservation.

Q2. Local backprop recursion for type (a). Fix $j \in \{1, \dots, J\}$ and define the *parameter gradient at layer j* by

$$g_j := \nabla_{\theta_j} L(a_1, b_1; \theta_1, \dots, \theta_J).$$

Show that for type (a) there exists an explicit mapping G_j such that, for all $j \in \{1, \dots, J-1\}$,

$$\big(g_j,\ a_j,\ b_j\big) = G_j\big(g_{j+1},\ a_{j+1},\ b_{j+1}\big).$$

Your answer should make explicit what information must be carried backward (e.g. adjoints of a_{j+1} and b_{j+1}, or equivalent quantities), and how (a_j, b_j) can be reconstructed from (a_{j+1}, b_{j+1}) in type (a).

Q3. Peak activation memory and numerical verification (type (a)). Assume the forward activations for one layer F_j occupy κ bytes (or units) of memory when stored for backward.

(a) Explain why standard backpropagation stores $\mathcal{O}(J\kappa)$ activation memory in a depth-J network.

(b) Using type (a) updates, show how to reduce peak activation memory from $\mathcal{O}(J\kappa)$ to $\mathcal{O}(\kappa)$ during the backward pass.

(c) Verify this numerically on a small example (choose d, J, and a simple f_j such as a J-layer MLP). Report: (i) peak memory with standard backprop, (ii) peak memory with reconstruction, and (iii) any time overhead. What are the pros and the cons?

Q4. Second-order recursion induced by type (b). For type (b), derive a second-order recursion satisfied by the sequence $(a_j)_{j \geq 1}$ alone. Discuss which additional constraint on f_j would allow to derive a mapping similar to Q2.

Q5. Decoupling computation and communication in a multi-GPU pipeline. Assume each f_j (hence each layer F_j) is placed on a different GPU in a chain (GPU j holds f_j).

(a) Describe what must be communicated between GPU j and GPU $j+1$ in the forward pass for type (a) and type (b).

(b) Show how, for type (b), communication (e.g., due to Tensor Parallelism) can be overlapped with computation by streaming only the necessary state components.

(c) Verify this again numerically on a small example (choose d, J, and a simple f_j such as a J-layer MLP). What are the pros and the cons?