APPENDIX

IMPLEMENTATION DETAILS OF THE BAGGING PROCESS IN VFPU

As shown in Algorithm 1 of this paper, assuming that the time complexity of the Base_Estimator_Learning function (line 10) is $\mathrm{O}(B)$. M and T denote the number of executions of the outer loop (line 2) and inner loop (line 5), respectively. The inner loop is actually a bagging process, which has two ways of implementation: serial execution and parallel execution. The implementation details and the experimental comparison of these two ways will be described as follows.

(1) Serial execution

Serial execution refers to the process where one task is completed before proceeding to the next. As shown in Fig. A.1, party C executes sampling, distributing ID and calling Base_Estimator_Learning function in a serial manner, where each subsequent round of sampling, distributing ID, and Base_Estimator_Learning commences only upon the completion of the previous round. If we use serial execution in our method, the total time complexity of our method is $\mathrm{O}(MTB)$

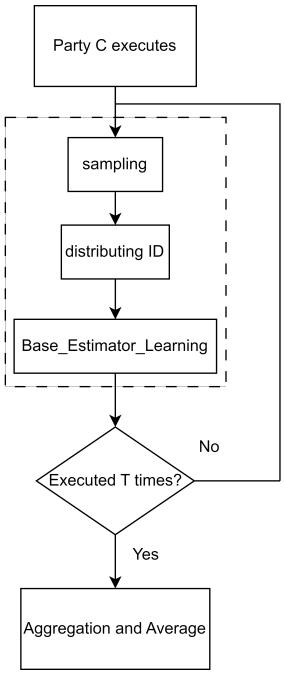

Fig. A.1: Design of serial execution scheme

(2) Parallel execution

Parallel execution refers to the process where multiple tasks are executed simultaneously rather than one by one. As shown in Fig. A.2, each round of the process, consisting of sampling, distributing ID, and calling the Base_Estimator_Learning function, is executed in a linear sequence. However, multiple rounds can be executed concurrently using multithreading techniques. If we use parallel execution, in an ideal scenario, disregarding thread creation and management, time complexity could be reduced to $\mathrm{O}(MB)$

(3) Experimental comparison

To compare the two designs of execution schemes, we first recorded the runtime of our method based on the two

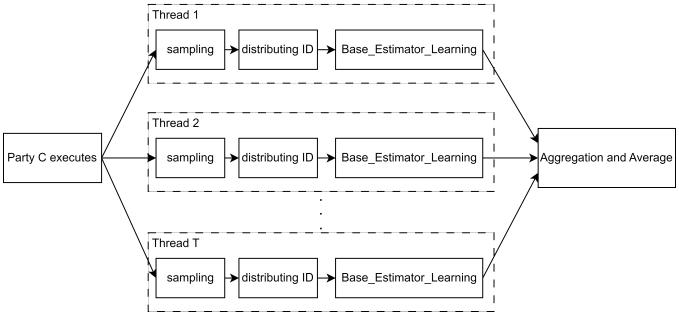

Fig. A.2: Design of parallel execution scheme

execution schemes without federation (No_Fed). We found that the parallel execution indeed reduces the time overhead significantly. Then, we experimented with our method based on the two execution schemes with federation. The experimental results are shown in Table A.1. We observed that parallel execution significantly reduces the runtime on all three datasets. For instance, on the Bank Marketing (Bank) dataset, the runtime of No_Fed_VFPU_GBDT_Serial is 62.77s, while the runtime of No_Fed_VFPU_GBDT_Parallel is only 15.41s, which is an improvement of about 4.07 times. Similarly, VFPU_GBDT_Serial runs in 106922.20s while VFPU_GBDT_Parallel runs in 29803.25s, an efficiency improvement of 3.59 times.

Finally, as vertical federated learning necessitates encryption algorithms, the data employed for computation are encrypted values. This leads to a significantly greater runtime in vertical federated learning than in non-federated scenarios. This is corroborated by experiments in [37], [38], and [51]. Despite our efforts to minimize time overhead, it remains at this order of magnitude.

TABLE A.1
Runtime(s) of our method - 'Serial' vs 'Parallel' execution in bagging process

Method	Execution scheme	Datasets		
		Bank	Credit	Census
No_Fed_VFPU_GBDT	Serial	62.77	63.37	123.89
	Parallel	15.41	17.85	30.29
	Speedup	4.07	3.55	4.09
VFPU_GBDT	Serial	106922.2	107075.9	107089.9
	Parallel	29803.25	30086.33	29996.25
	Speedup	3.59	3.56	3.57