L-Ⅲ
準同型暗号アプリケーション開発ゼミ

2021/09/26

松本 直樹

事例紹介(Cheetah)

- Cheetah: Optimizing and Accelerating Homomorphic Encryption for Private Inference
  - Authors: Brandon Reagen et al.
  - Published at: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)
  - https://arxiv.org/abs/2006.00505

概略

- これまでの深層学習においては計算機の性能上で制約があった
- 加えて**プライバシー**という制約も出現
- 要件を達成可能な **Homomorphic Encryption(準同型暗号)** がある
- しかし、
  - 膨大な計算上の課題が存在する
  - 実用的でない

概略

- HE を利用した DNN 推論システムとして **Cheetah** を提案
  - アルゴリズムとハードウェアの最適化を実施
  - SoTA な手法に比べて **79 倍の高速化**を達成
  - **専用に設計されたアクセラレータ**によって平文に近い推論速度を達成

Introduction

- プライバシーを保護しながら DNN 推論を達成する手法は複数存在する
  1. ユーザーの手元のデバイスでの推論
  2. TEE(Trusted Execution Environment)での推論
  3. 理論的に安全性が担保された手法での推論
     1. 差分プライバシー(DP; Differential Privacy)
     2. Secure Multi-Party Compute(MPC)
     3. Homomorphic Encryption(HE)

Introduction

- 各手法にはそれぞれ制約が存在する
  - Local はモデルの漏洩
  - TEE はサイドチャネル攻撃
  - DP はプライバシと利用できるデータのトレードオフ
  - MPC は通信コスト
  - HE は計算コスト

TABLE I: Generalization of privacy-preserving techniques.

Solution	Security	Limitation
Local	System	Edge performance; leaks model
TEE	System	Performance; side-channels
DP	Statistical	Applications; utility-privacy tradeoff
MPC	Cryptographic	Communication bandwidth
HE	Cryptographic	Compute

Introduction

- HE を利用してシステムを構築することにした
- HE は平文に比して **10^6 ~ 10^7 のオーダーで遅い**
- 最新の手法(GAZELLE)でも MNIST の推論に 800ms かかる
  - GAZELLE: Packed Additively HE (加法準同型の一種) と GC(Garbled Circuit)を利用

Introduction

- ハイパフォーマンス HE 推論には3つの課題がある
  1. **HEのパラメータ選定**
     ノイズと計算速度はトレードオフ
     ※ ノイズが増大すると計算が誤る確率が増加する(LWE)
     　LWE 暗号などの細かい話は L-II のゼミ資料を参照
  2. **演算を HE で実現する手法**
     HE でできる計算処理は限られる(加算, 乗算)
  3. **HE 推論における膨大な計算処理**
     並列性を生かした並列計算

Introduction

- これらの課題について取り組んだ Cheetah を提案
  1. **HE-PTune**
     モデルの各層に応じた HE パラメータの自動選定
  2. **Sched-PA**
     HE 演算の順序が性能とノイズに大きな影響を与える研究結果を利用
  3. **ハードウェアアクセラレータ**
     ソフトウェア実装でプロファイリングを実施し、
     その結果に基づいてアクセラレータを設計

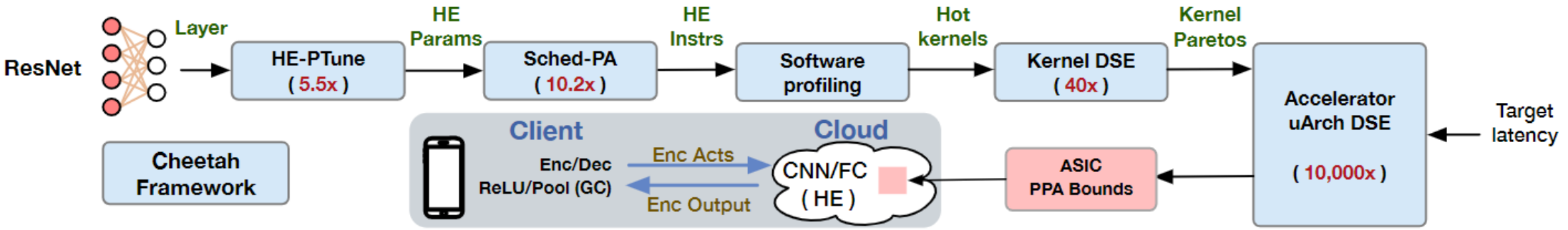


Fig. 1: The Cheetah framework and system design. Speedup achieved for ResNet50 is reported in red.

Overview and Assumptions

- 設計上の欠点が存在
  1. 非線形関数を処理できない
     → 先行研究は複数の暗号方式を組み合わせて実現
     FC, CNN は HE 上で処理し、 **ReLU, Pool は GC で処理**することにした
  2. DNN における膨大な計算はパラメータ選定とHEの処理速度に悪影響を与える
     → アルゴリズムとハードウェアアクセラレータで何とかした

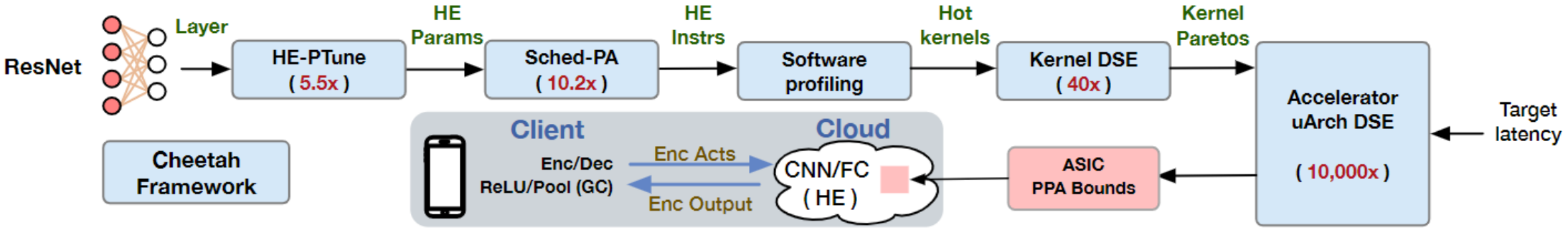


Fig. 1: The Cheetah framework and system design. Speedup achieved for ResNet50 is reported in red.

Overview and Assumptions

- 脅威モデル
  - 2-party compute モデル
  - クライアント(ユーザー)とクライドは **honest but curious**
  - 2者はプロトコルには従うが、情報を抜き取ろうとする可能性がある
  - Cheetah は**クライアント側のデータ**と**クラウド側のモデル**を保護する

Background

- 割愛
  - HE, LHE, RLWE, BFV に関する基本的な内容

HE-PTune: Models & Parameter Tuning

- HE のパラメータ選定は難しい問題
- ノイズを許容すればするほどHEの計算コストは増大する
- 従来の手法はできる限りノイズを許容しようとする
  → 結果として HE の計算速度が悪化する
- パラメータの適切な選定を行う HE-PTune を提案
  → SoTA な手法に比して **11.7 倍の高速化を達成**

HE-PTune: Performance Modeling

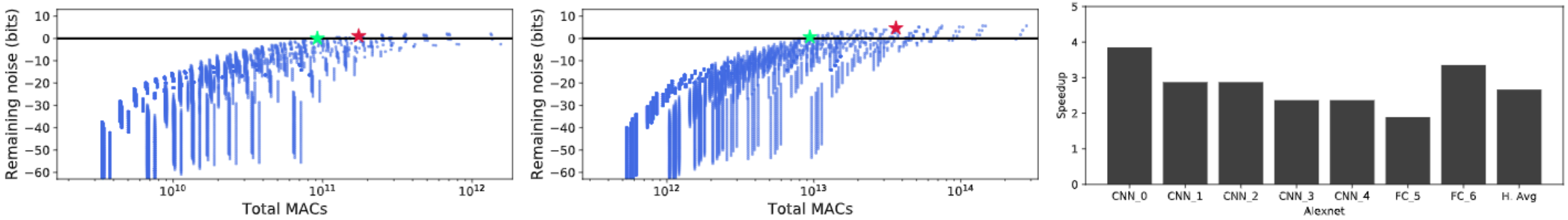
- モデルの各層における**HE演算内の整数乗算**の総数によりパフォーマンスモデルを構築
  - ほとんどの HE 演算は整数(多項式)乗算に基づき、実行時間と強い相関がある
  - CNN と FC のパフォーマンスモデルを構築するために、まず全ての HE 演算と NTT 処理を集計し、それから整数乗算の総数を割り出す
  - CNN と FC それぞれについて HE_Mult や HE_Rotate の数を集計し、整数乗算の総数を計算(具体的な計算式は割愛)

HE-PTune: Noise Modeling

- モデルの各層が HE 演算として実装されればノイズも集計可能
  - 各HE演算で増加するノイズはモデル化されている
  - HEのパラメータと CNN, FC のパラメータからノイズの増加を算出
  - 算出されるノイズは最悪の場合であり、ほとんど場合は生じない
  - **復号の誤り率から妥当なパラメータを算出する手法**を提案
    - ノイズはIBDG(Independent Bounded Discrete Gaussian distribution)に従う (TFHE のモジュラー正規分布と同義, L-II の資料を参考)
    - IBDGには再生性があるため複数のHE演算後のエラーの確率分布もIBDGに従う
    - 復号の誤り率からノイズの閾値を計算可能

HE-PTune: HE Parameter Space Exploration

- HE-PTune によるパラメータの選定はDNNのモデルに基づいて限られたパラメータ空間から選定できる
- 実際にHE-PTune でパラメータを選定した結果、計算速度が改善した



(a) Layer5 (FC) (b) Layer0 (CNN) (c) Alexnet Speedup

Fig. 3: Comparison of HE-PTune and Gazelle using AlexNet. Blue dots are HE configurations modeled by HE-PTune. The red star is Gazelle's configuration and the green star is the optimal found by HE-PTune. Layer5 and Layer0 show the best and worst configuration for Gazelle with respect to utilized noise budget. HE-PTune's speedup for all layers on the right.

Partial-Aligned Scheduling

- Sched-PA: 内積に関するスケジューリング
- HE 演算はそれぞれ実行時間と増加するノイズが異なる
  - HE_Add, HE_Rotate は**加算的に**ノイズが増える
  - HE_Mult は**乗算的に**ノイズが増える
- Sched-PA によって **10.2 倍の高速化**を達成

Sched-PA: Partial-Aligned Dot Products

- 内積に利用するHE演算は3つ
  - HE_Mult で部分積、 HE_Add, HE_Rotate で部分積の足し上げ
  - BFV では SIMD 的な演算を行う(同じインデックス同士でしか計算できない)
  - 計算例
    1. ベクトルの各項同士を HE_Mult で積を取る
    2. HE_Rotate で部分積の第n+1項を第n項にずらす
    3. HE_Add で部分積の第0項を結果ベクトルの第0項に足す
    4. すべての部分積の項を足し上げるまで 2 に戻る
- ベクトルに要素を詰め込む順番を工夫すればノイズを減らせる

Sched-PA: Partial-Aligned Dot Products

- Sched-PA では HE_Mult が乗算的、HE_Rotate が加算的にノイズが増えることを利用

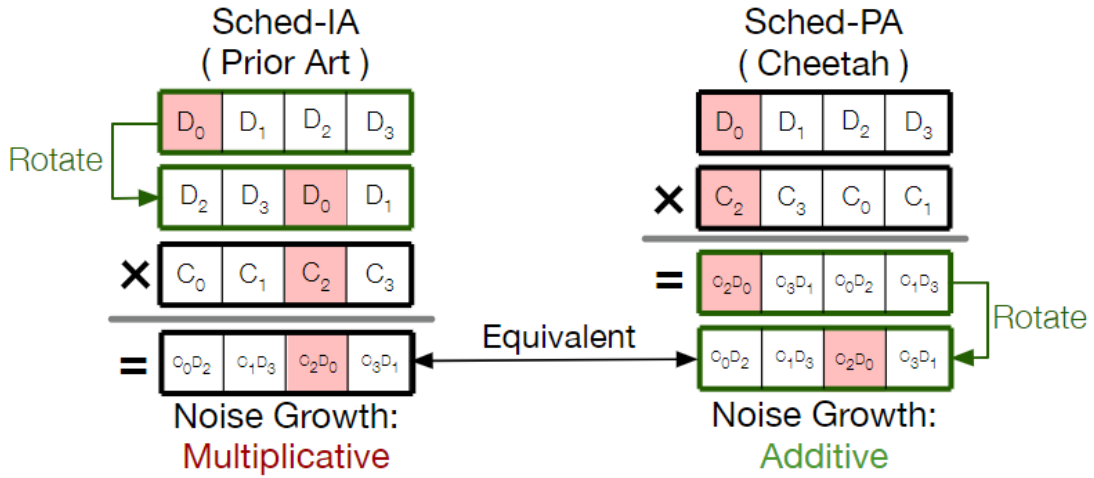


Fig. 5: Sched-IA (input-aligned) versus Sched-PA (partial-aligned) dot product schedules. Cheetah uses Sched-PA to improve performance of CNN and FC layers.

Implementing Low-Noise Convolution

- Sched-PA を利用した CNN の計算
  - 入力画像は順番にベクトルに埋め込み
  - 重み(フィルター)の各項を取り出しベクトルに適切に埋め込み
  - HE_Mult で部分積を計算し、 HE_Rotate と HE_Add で足し上げ

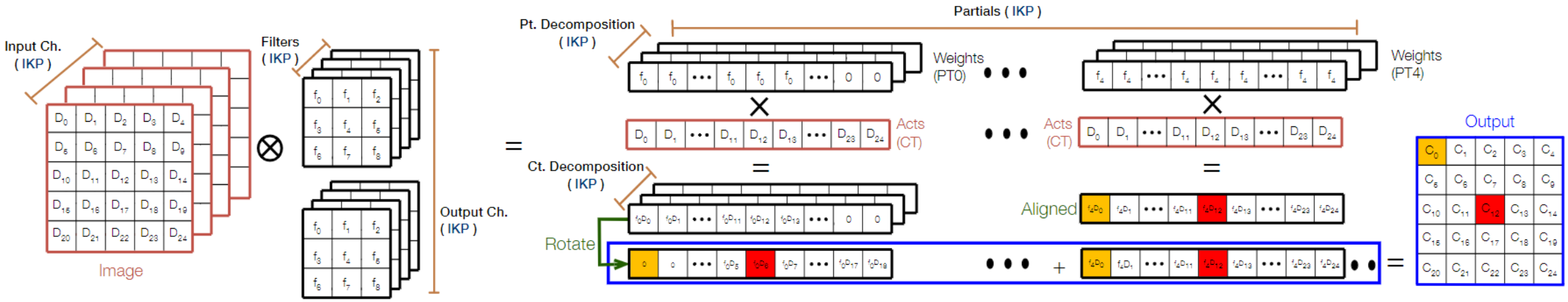


Fig. 4: How Cheetah implements CNNs using Sched-PA. Sources of inter-kernel parallelism (IKP) are labeled

Evaluation Results

- Sched-PA により HE-PTune 単体より2.98倍高速化を達成
- HE-PTune + Sched-PA により Gazelle の 13.5倍高速化を達成
- Sched-PA は 5.20倍の高速化を達成したことになる

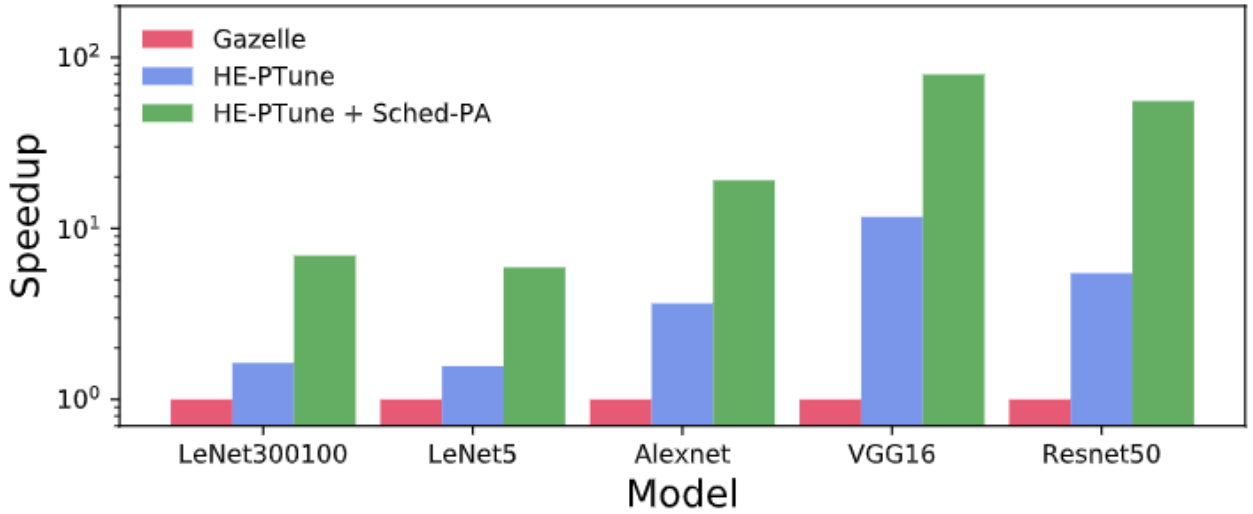
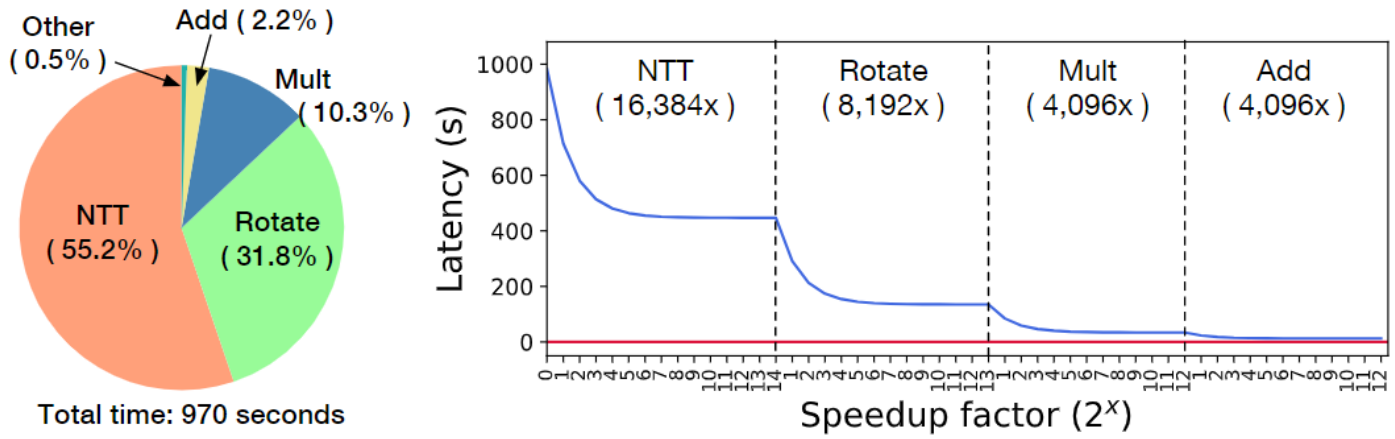


Fig. 6: Per-benchmark speedup achieved by Cheetah using HE-PTune and Sched-PA. Speedup is relative to Gazelle [32].

Profiling HE Inference

- HE-PTune と Sched-PA により ResNet 50 で 55.6倍高速化を達成
- まだ平文と比べて 10^3 ~ 10^4 程度遅い
- Ceetah を SEAL を使って実装したところ、 ResNet 50 の推論に 970秒かかった (平文は 100ミリ秒)



(a) Time breakdown (b) Speedup Needed

Fig. 7: Profiling results for ResNet50 and speedup needed by each kernel to match plaintext inference latency.

Profiling HE Inference

- GPUによる高速化を模索
- ベクトル長やバッチサイズを変えて試行したが、120倍でサチった
- ネイティブサポートされていない整数長や分岐が原因と考えられる
- 目標速度に全く届かない

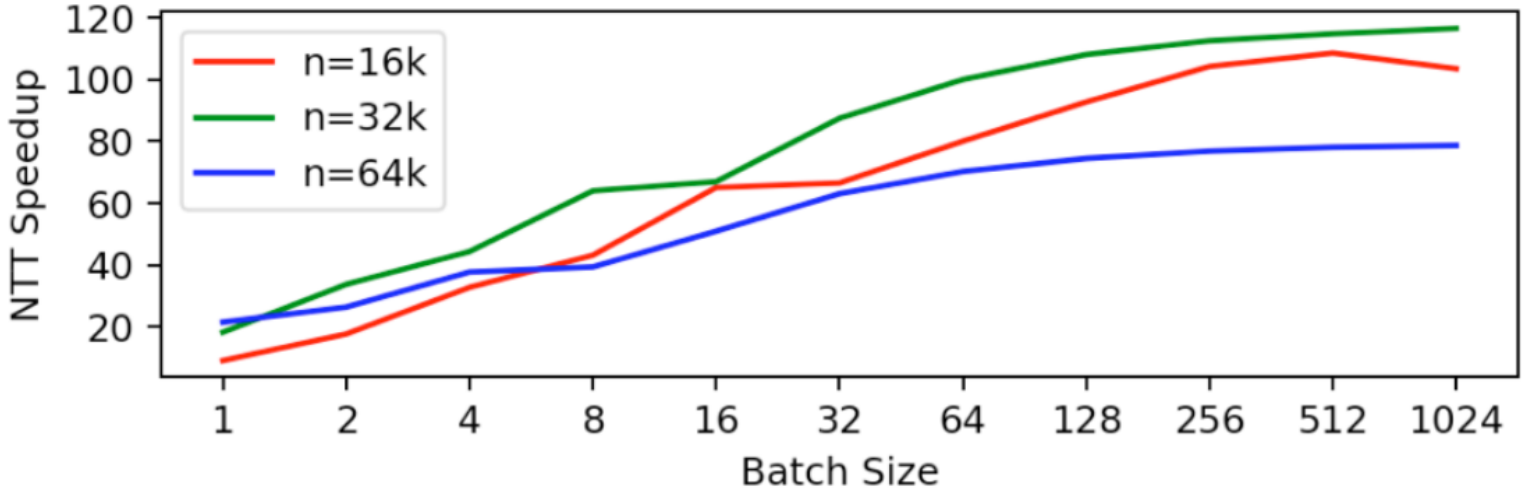


Fig. 8: NTT GPU speedup over CPU.

HE Inference Accelerator Architecture

- 暗号文ごとにPEを持ち並列に処理
- PE は Lane を複数持ち、それぞれで並列に部分積を計算する

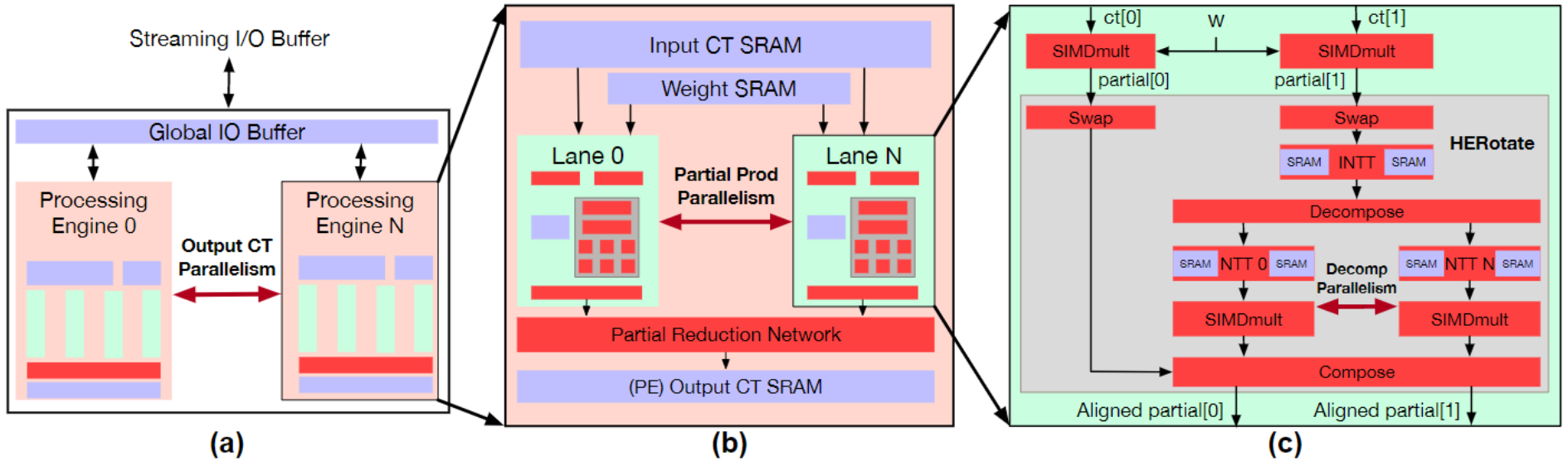
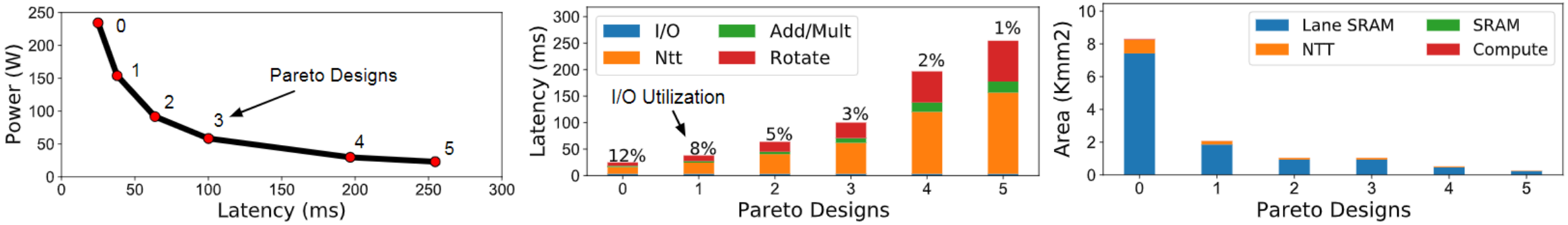


Fig. 9: Cheetah accelerator architecture. (a) The accelerator is composed of parallel PEs operating in output stationary fashion. Off-chip data is communicated via a PCIe-like streaming interface, and data is buffered on-chip using global PE SRAM. (b) Each PE contains Partial Processing Lanes which compute the HE dot product. (c) Lanes comprise individual HE operators.

Experimental Results

- Catapult HLS を用いて回路を記述
- 消費電力と処理速度のトレードオフを調査しつつ最適なデザインを決定
- SRAM は別途商用のSRAMコンパイラを利用



(a) ResNet50 DSE (b) Time Breakdown (c) Area Breakdown

Fig. 11: (a) Power-latency Pareto for ResNet50 DSE. (b) Run time breakdown for each Pareto design point. (c) Energy and area breakdown for each Pareto design point.

Experimental Results

- 単体のカーネルで 40 倍の高速化を達成
- 様々なモデルを扱えるようにある程度汎用性を持たせている

TABLE VI: Performance of running VGG16 and AlexNet on PT-ResNet50 accelerator. Prt is partials per output CT.

Model	Lat(ms)	Increase	PEs-Lanes	Out CT μ	(K)Prt μ
ResNet50	100	0%	8-512	147	50.5
VGG16	215	59%	16-256	422	595
AlexNet	77	28%	16-128	475	337

Related Work

- 割愛

Conclusion

- HE-PTune により 11.7 倍の高速化を達成した
- Sched-PA により 10.2 倍の高速化を達成した
- カーネル間、カーネル内の並列性を活用したアクセラレータを開発した(30W and 545mm^2 in 5nm)
- 平文と同等の性能を達成した

- **本気で HE を使おうと頑張った結果、平文と同じ速度になった**

事例紹介

- GateNet: Bridging the Gap between Binarized Neural Network and FHE Evaluation
  - Authors: Cheng Fu et al.
  - Published at ICLR 2021 Workshop on Security and Safety in Machine Learning Systems
  - https://aisecure-workshop.github.io/aml-iclr2021/papers/9.pdf

GateNet

- FHE(TFHE) で BNN を評価する話
- Projected Time (h) なことに注意

Table 2: Comparison between GateNet with other BNN baselines under TFHE inference.

	Models	Type	Gate ops (Million)	popcount	XNOR	Other ops†	Projected Time (h)*	Accuracy ‡ (Top-1 %)
MNIST	XNOR-LeNet-5	A	953.26	944.16	5.69	3.40	4236.7	99.2
	TAPAS	A	825.71	745.80	79.62	0.30	3669.8	98.6
	GateNet-A	A	73.56	20.43	2.94	50.19	326.9	98.3
	GateNet-S	$S_{f,l}$	9.92	1.40	0.73	7.79	44.1	98.8
CIFAR-10	ReActNet (Bi-real) (Liu et al., 2020b)	$S_{f,l}$	68187.73	67462.87	610.27	114.58	303056.6	85.8
	DSQ (Gong et al., 2019)	$S_{f,l}$	68117.27	67462.87	610.27	44.12	302743.4	84.1
	DoReFaNet (Zhou et al., 2016)	$S_{f,l}$	68084.23	67462.87	610.27	11.08	302596.6	79.3
	FracBNN-1-bit (Zhang et al., 2021)	S_f	5296.66	5205.50	51.6096	39.55	23540.7	85.9
	GateNet-S-1.0×	$S_{f,l}$	882.13	669.36	54.69	158.08	3920.6	80.5
	GateNet-S-1.5×	$S_{f,l}$	2388.38	2028.19	123.07	237.12	10615.0	84.6
	GateNet-S-1.5×	S_f	3454.48	3076.75	137.21	240.51	15353.2	84.1
ImageNet	XNOR-AlexNet (Rastegari et al., 2016)	$S_{f,l}$	$2815.85_{\times 10^3}$	$2773.84_{\times 10^3}$	$28.32_{\times 10^3}$	$13.70_{\times 10^3}$	$12514.9_{\times 10^3}$	48.6
	BENN-SB-3 (Zhu et al., 2019)	$S_{f,l}$	$8447.56_{\times 10^3}$	$8321.51_{\times 10^3}$	$84.96_{\times 10^3}$	$41.10_{\times 10^3}$	$37544.7_{\times 10^3}$	53.6
	ReActNet (Bi-real) (Liu et al., 2020b)	$S_{f,l}$	$209.05_{\times 10^3}$	$206.78_{\times 10^3}$	$1.88_{\times 10^3}$	$0.40_{\times 10^3}$	$929.1_{\times 10^3}$	65.9
	Bi-RealNet-18 (Liu et al., 2020a)	$S_{f,l}$	$208.77_{\times 10^3}$	$206.78_{\times 10^3}$	$1.88_{\times 10^3}$	$0.12_{\times 10^3}$	$927.8_{\times 10^3}$	56.4
	GateNet-S-1.0×	$S_{f,l}$	$1.01_{\times 10^3}$	$0.45_{\times 10^3}$	$0.08_{\times 10^3}$	$0.48_{\times 10^3}$	$4.5_{\times 10^3}$	49.8
	GateNet-S-2.0×	$S_{f,l}$	$3.15_{\times 10^3}$	$1.94_{\times 10^3}$	$0.29_{\times 10^3}$	$0.92_{\times 10^3}$	$14.0_{\times 10^3}$	61.8

* We report the total execution time through linear projecting based on the evaluation time of average gate (16*ms* on a 2-core CPU).
† Other ops includes pooling / residual / Non-linear function / Batch Normalization.
‡ The accuracy of other BNN methods do not quantize the β in affine functions.

事例紹介

- FDFB: Full Domain Functional Bootstrapping Towards Practical Fully Homomorphic Encryption
  - Authors: Kamil Kluczniak et al.
  - Submitted on 6 Sep 2021 at arxiv
  - https://arxiv.org/abs/2109.02731

FDFB

- TFHE の Circuit Bootstrapping と LUT を拡張して高速化した話
- log_2(t) は整数ビット幅
- 表の Evaluation Time は **hour 単位**なことに注意

Param. \ $\log_2(t)$	Evaluation Time MNIST-1-f						Accuracy MNIST-1-f					
	6	7	8	9	10	11	6	7	8	9	10	11
FDFB:80:6	0.027						0.80	0.95	0.95	0.95	0.80	0.60
FDFB:100:6	0.11						0.75	0.90	0.90	0.90	0.80	0.75
FDFB:80:7	0.08						0.90	0.90	0.95	0.95	0.90	0.70
FDFB:100:7	0.35						0.90	0.90	0.95	0.90	0.90	0.80
FDFB:80:8	0.17						0.85	0.90	0.95	0.90	0.95	0.90
FDFB:100:8	0.76						0.80	0.95	0.90	0.95	0.95	0.90
TFHE:80:2	99.24	135.88	177.28	225.84	289.82	392.14	0.82	0.94	0.95	0.95	0.95	0.95
TFHE:100:2	122.18	165.9	215.44	274.44	351.42	460.94	0.82	0.94	0.95	0.95	0.95	0.95
	Evaluation Time MNIST-2-f						Accuracy MNIST-2-f					
FDFB:80:6	0.14						0.10	0.20	0.45	0.85	0.90	0.85
FDFB:100:6	0.23						0.10	0.20	0.70	0.95	0.95	0.90
FDFB:80:7	0.37						0.15	0.25	0.85	0.90	0.90	0.90
FDFB:100:7	0.65						0.15	0.30	0.85	0.95	0.90	0.85
FDFB:80:8	1.1						0.15	0.30	0.85	0.90	0.95	0.90
FDFB:100:8	1.8						0.15	0.30	0.90	0.90	0.95	0.95
TFHE:80:2	24.56	33.64	43.88	55.9	71.74	97.08	0.14	0.22	0.57	0.93	0.93	0.93
TFHE:100:2	33.64	41.06	53.3	67.94	86.98	114.1	0.14	0.22	0.57	0.93	0.93	0.93

TABLE IV: Time to evaluate the MNIST-1-f and MNIST-2-f neural networks. The time to evaluate the neural networks is given in hours. Due to extremely high execution times, we only present estimates for TFHE, which we base on Table VI. For accuracy we show values for $f = \text{ReLU}$, but stress that we can use arbitrary univariate functions without impacting execution time. We discretized the models using a discretization factor $\delta = 6$ for MNIST-1-f and $\delta = 4$ for MNIST-2-f.

事例紹介

- Virtual Secure Platform: A Five-Stage Pipeline Processor over TFHE
  - Authors: Kotaro Matsuoka et al.
  - Published at: 30th USENIX Security Symposium
  - https://www.usenix.org/conference/usenixsecurity21/presentation/matsuoka

Virtual Secure Platform

- プログラムとデータを暗号化した状態で処理する話
- Iyokan がCPUとGPUを混ぜた計算処理をサポート
- CMUX ROM などの最適化

Table 5: Performance Evaluation Using Hamming

Case #	Machine	# of V100	Pipelining?	CMUX Memory?	# of cycles	Runtime [s]	sec./cycle	# of tries
1	AWS c5.metal	0	No	Yes	936	2342.0 ± 13.3	2.502 ± 0.014	3
2			Yes	Yes	1216	2773.0 ± 2.8	2.280 ± 0.002	3
3	Sakura Koukaryoku	1	No	No	936	5919.0 ± 33.1	6.324 ± 0.035	5
4			No	Yes	936	2232.1 ± 1.7	2.385 ± 0.002	5
5			Yes	No	1216	7809.0 ± 45.8	6.422 ± 0.038	4
6			Yes	Yes	1216	2045.0 ± 4.6	1.682 ± 0.004	5
7	AWS p3.8xlarge	4	No	Yes	936	1455.7 ± 0.3	1.555 ± 0.000	3
8			Yes	Yes	1216	979.0 ± 12.5	0.805 ± 0.010	3
9	AWS p3.16xlarge	8	No	No	936	1627.0 ± 4.2	1.739 ± 0.004	3
10			No	Yes	936	1440.0 ± 2.5	1.538 ± 0.003	3
11			Yes	No	1216	1566.0 ± 9.7	1.288 ± 0.008	3
12			Yes	Yes	1216	**965.9 ± 3.4**	**0.794 ± 0.003**	3

事例紹介まとめ

- LHE を使った手法はすでに実用段階
- FHE はまだまだ発展途上