

# Language Models in Software Development Tasks: An Experimental Analysis of Energy and Accuracy

---

Negar Alizadeh\*, Boris Belchev†, Nishant Saurabh\*  
\*Utrecht University, Utrecht, The Netherlands  
{n.s.alizadeh, n.saurabh}@uu.nl  
†University of Twente, Enschede, The Netherlands  
b.belchev@student.utwente.nl

Patricia Kelbert‡, Fernando Castor†  
‡Fraunhofer IESE, Kaiserslautern, Germany  
patricia.kelbert@iese.fraunhofer.de  
†University of Twente, Enschede, The  
Netherlands  
f.castor@utwente.nl

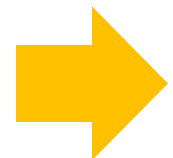
# 背景

## 生成AIの普及

- SOの調査では開発者の76%がコード支援ツールを使用
- サードパーティサービス（ChatGPT, GithubCopilot）が主流

└ 企業にとってはいくつかの不都合が

- ・ データプライバシー，セキュリティ  
機密情報，顧客情報などが外部に流出する可能性
- ・ お金  
シンプルにお金かかる

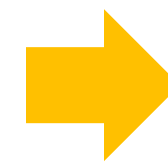


ローカル展開の需要が高まっている

# ローカル展開における問題

## 高性能GPUと大きな消費電力

ローカル展開には高性能なGPUが必要  
消費電力も大きい



費用  
環境負荷 大

LLMのエネルギー消費に関する研究が行われている

## 先行研究

NLPタスクの学習時におけるエネルギー消費を調査

推論時におけるエネルギー消費を調査した研究は少ない

SEタスクにおけるエネルギー消費は調査されていない

# 目的・アプローチ

---

## 目的

- SEタスクの実行時におけるLLMのエネルギー消費と精度を調査

## アプローチ

4つのSEタスク

2つのハードウェア環境

18のモデルファミリー

3つの量子化レベル

これらを組み合わせて実験

# RQs

---

## RQ1

- タスク間でエネルギー消費は異なるか？

## RQ2

- エネルギー効率と精度のトレードオフは存在するか？

## RQ3

- アーキテクチャの特性は消費エネルギーに影響を与えるか？

## RQ4

- 汎用モデルとコード特化モデルに違いはあるか

# 実験設定

---

## タスク

HumanEvalを拡張したデータセット<sup>[20]</sup>を利用

コード生成・バグ修正・ドキュメント生成・テストケース生成

## ハードウェア環境

- NVIDIA A100（データセンター想定）

全てのモデルを実行

- RTX 3070（エンドユーザー想定）

小さいモデルのみを実行

# 実験設定

---

## 使用モデル

HuggingFace上の人気上位モデルを利用

コード特化モデル・汎用モデルどちらも利用

例) コード特化モデル

granite-code starcoder2 codellama など

例) 汎用モデル

phi3 llama2 deepseek-llm など

## 量子化

タイプ：学習後量子化

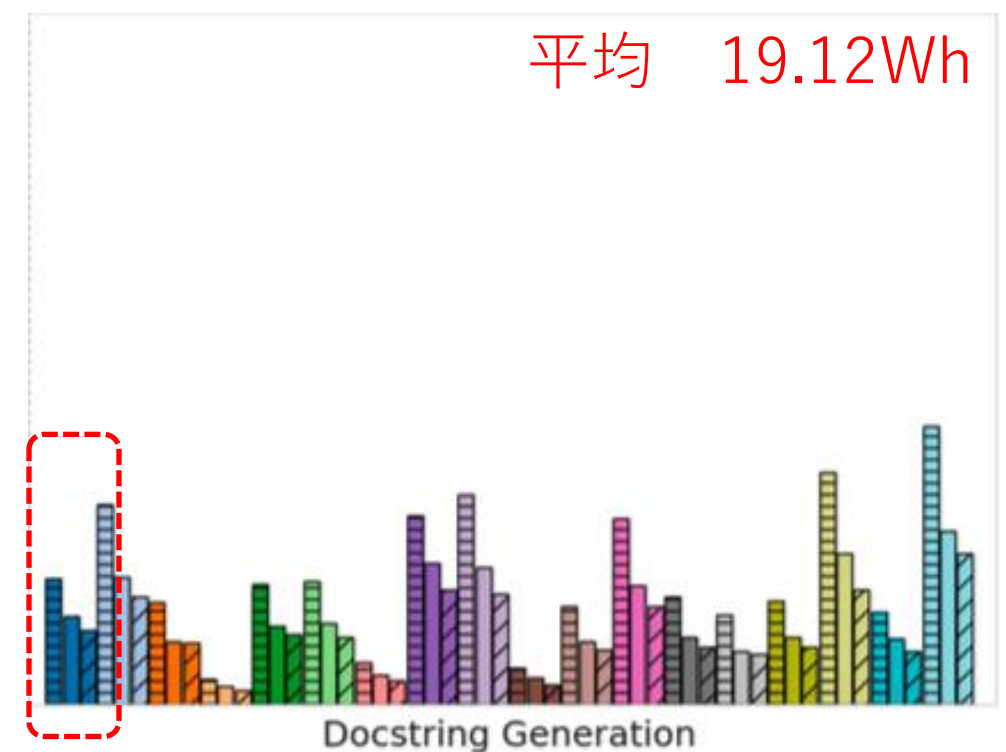
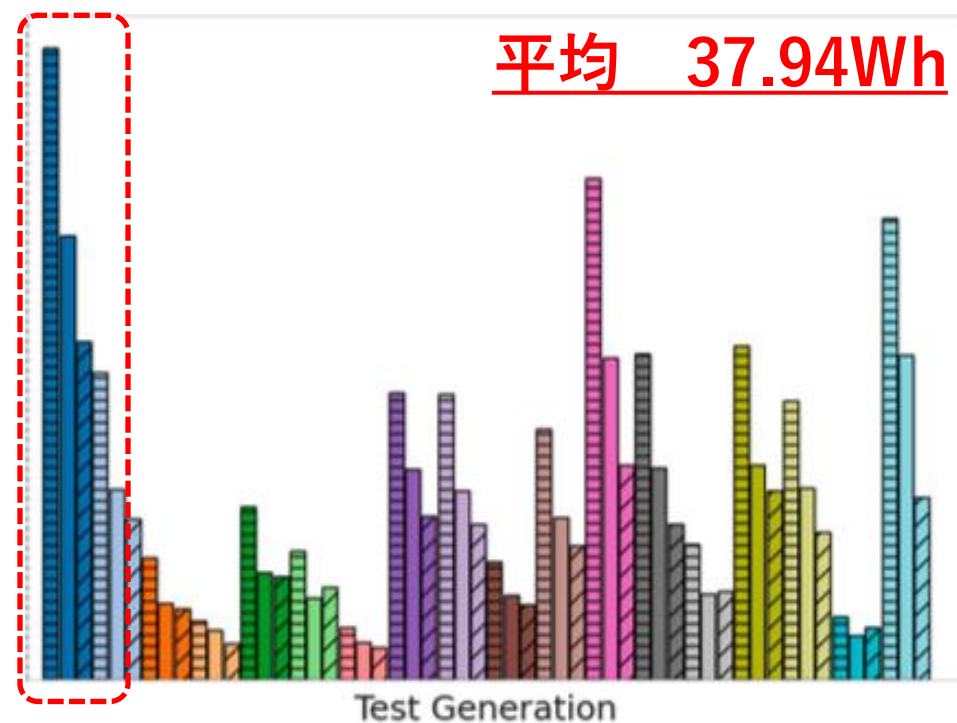
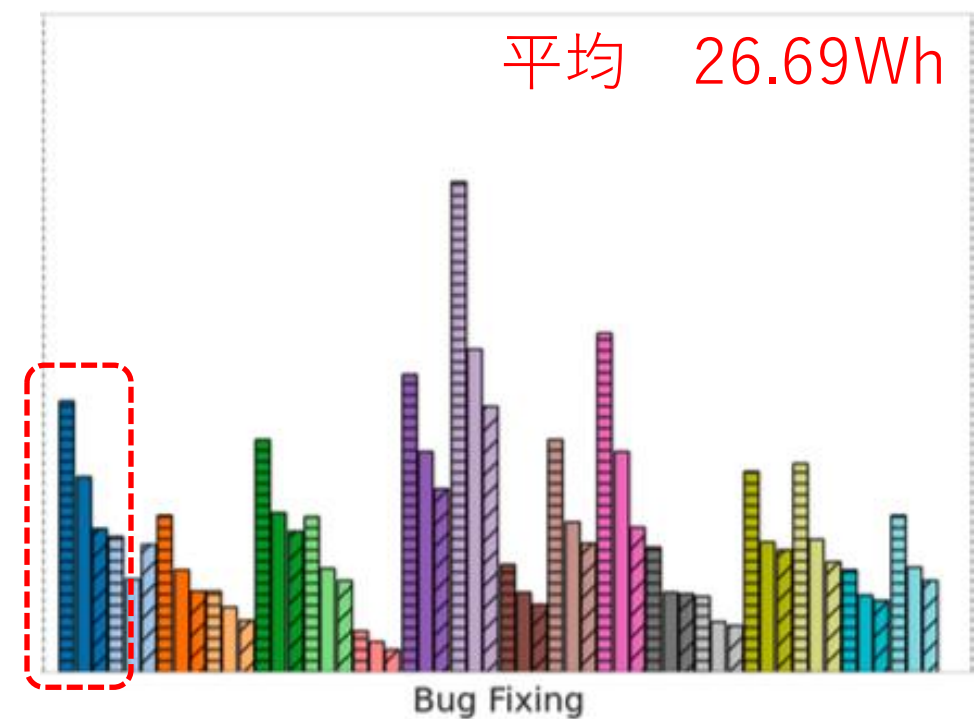
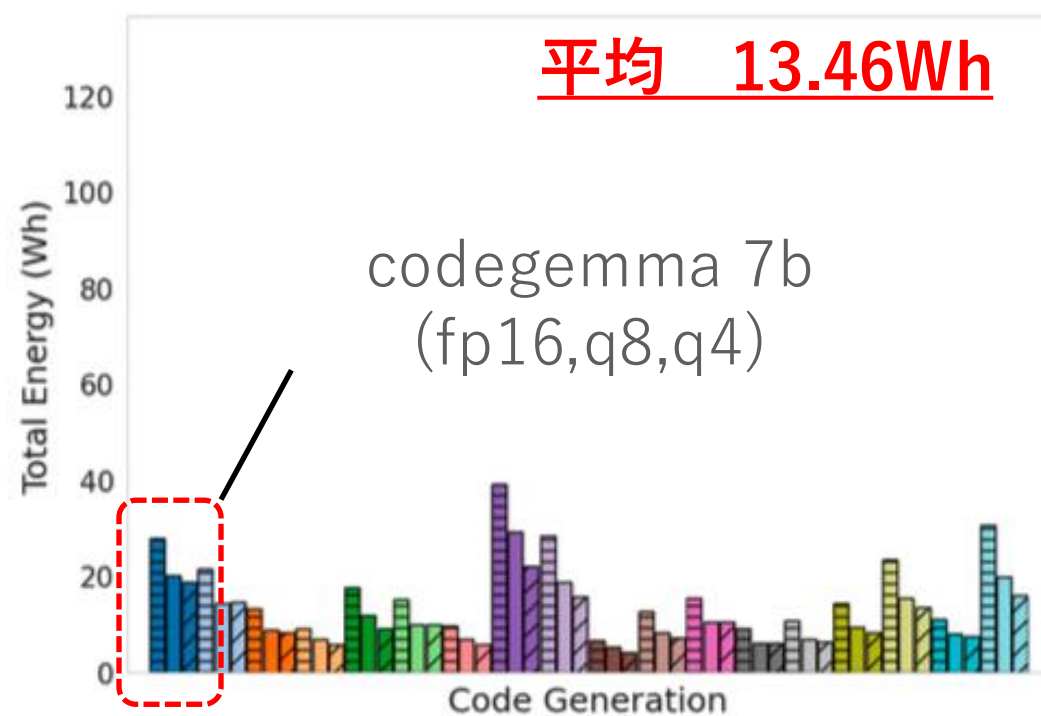
形式：GGUF (GPT-Generated Unified Format)

最もエネルギー効率が高い<sup>[16]</sup>

量子化レベル：4or5ビット, 8ビット, 16ビット (元々と同じ)

# 結果 | RQ1

## タスクごとの消費電力



# 結果 | RQ1

エネルギー効率 (tokens/J) 表 2 より

| モデル名                     | コード生成 | バグ修正 | ドキュメント生成 | テストケース生成 |
|--------------------------|-------|------|----------|----------|
| granite-code<br>20b-fp16 | 0.13  | 0.12 | 0.13     | 0.12     |

全タスクにわたってエネルギーあたりの出力トークン数は同じ

→ タスク間の消費エネルギー差は出力トークン数の違い

最もエネルギー効率が高かったモデル

コード生成：granite-code:3b-q4

バグ修正：gemma:2b-q4

ドキュメント生成：deepseek-coder:1.3b-q4

テスト生成：gemma:2b-q4

最もエネルギー効率がいいモデルはタスクごとに異なる

# 結果 | RQ1

---

## まとめ

- タスクによって消費エネルギーは異なる

テスト生成はコード生成の約2.8倍

- タスク間の消費エネルギー差は出力トークン数の違い

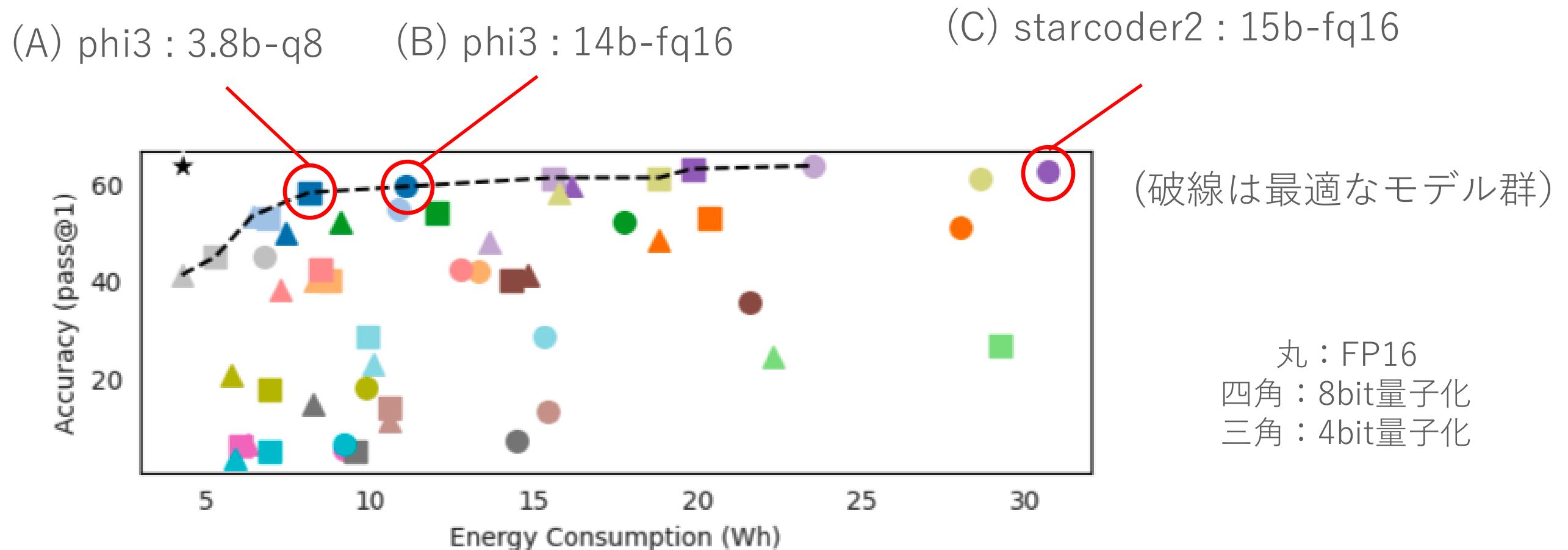
モデル自体のエネルギー効率は変わっていない

- 最もエネルギー効率のいいモデルはタスクごとに異なる

タスクに応じたモデル選択が重要

# 結果 | RQ2

## 消費エネルギーと精度の関係

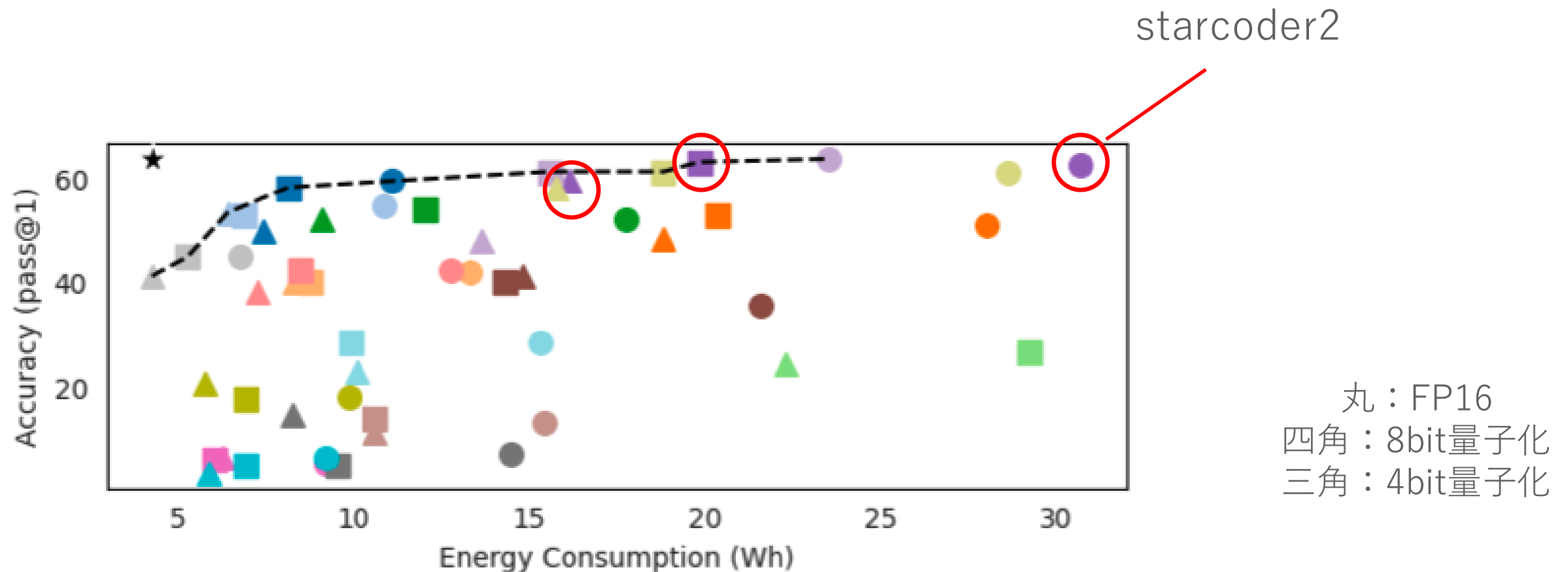


(a) Code Generation

(A) : 8.16Wh, 58.53%      (B) : 23.52Wh, 64.02%      (C) : 30.66Wh, 62.80%

パラメータ数を **2倍以上**に増やし, エネルギーを**約3倍**消費しても  
精度の向上は**4-5%**

# 結果 | RQ2



(a) Code Generation

量子化しても精度の低下は**限定的**

# 結果 | RQ2

---

## まとめ

- ・ エネルギーと精度は必ずしもトレードオフではない

小型・量子化モデルが大型・FPモデルを上回る場合がある

- ・ 量子化は効果的

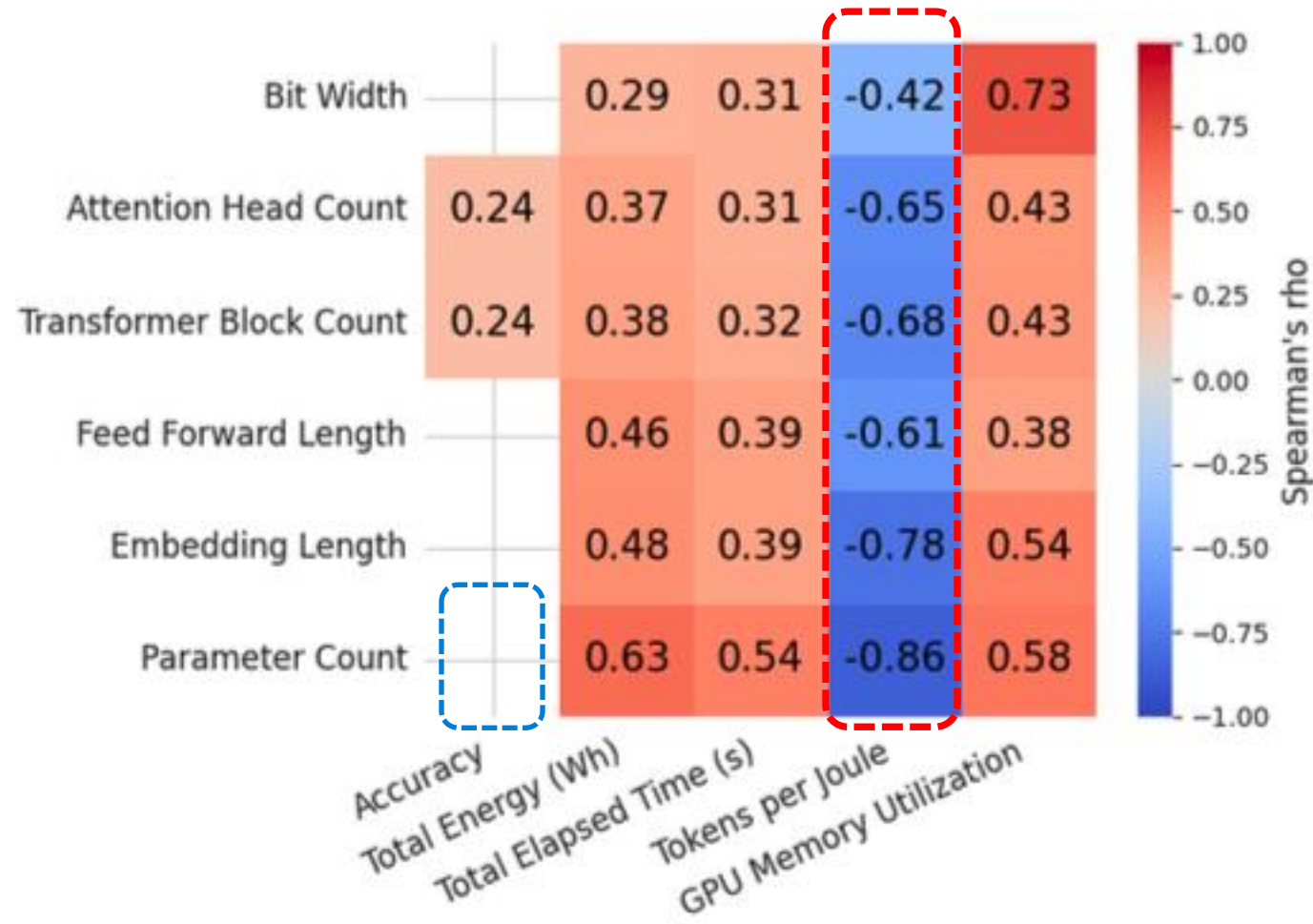
4-bit量子化が最も効率的

精度の低下は最小限

- ・ ある時点以降, エネルギー増加による精度向上は限定的

# 結果 | RQ3

## アーキテクチャと消費エネルギーの関係



有意な相関のみを記述

- ・ パラメータ数と精度に有意な相関なし
- ・ パラメータ数が多いほどエネルギー消費が増加

# 結果 | RQ4      コード特化モデル   VS   汎用モデル

---

## 精度面での比較

ドキュメント生成以外のタスクでコード特化モデルが優位

## エネルギー効率を考慮した分析

- ・ バグ修正

最適モデル群の多くは汎用モデル

- ・ テスト生成,   ドキュメント生成

最適モデル群に汎用モデルとコード特化モデルが混在

- ・ コード生成

最適モデル群は全てコード特化モデル

コード特化モデルがSEタスクで最適とは限らない

# まとめ

---

| SEタスクにおけるLLMの精度とエネルギー消費の  
トレードオフを調査

| エネルギー消費はタスクによって大きく異なる

| 大型モデルは必ずしも小型モデルより高精度ではない

| 大型モデルの量子化版は中型モデルの完全版よりも効率的

| モデルアーキテクチャからエネルギー効率を予測可能

# 所感

---

パラメータ数と精度に相関が見られないのは意外だった  
少なくとも相関はあると思っていた

(モデルによるが) 量子化してもたいして精度が落ちないのが  
意外だった