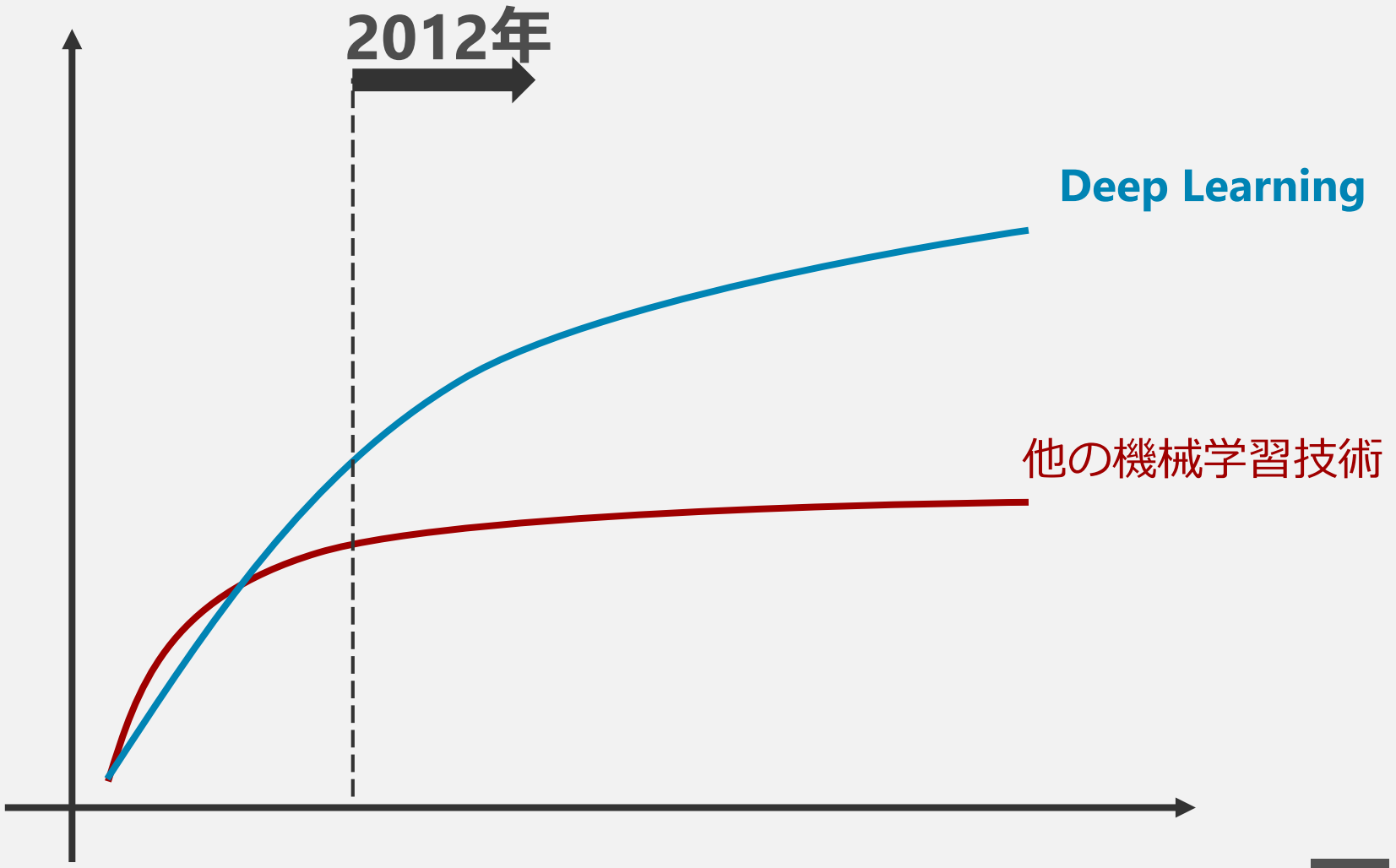
モバイルOS上での深層学習による画像認識システムの実装と比較分析

丹野 良介，柳井啓司 (電気通信大学)

はじめに

- **モバイル端末**の高性能化
  - スマートフォンで画像認識が可能に
- **深層学習**(Deep Learning)の応用
  - パターン認識分野で大きな注目

はじめに



はじめに

- **モバイル端末**の高性能化
  - スマートフォンで画像認識が可能に
- **深層学習**(Deep Learning)の応用
  - パターン認識分野で大きな注目

Deep Learningの応用: **モバイル上での画像認識**

モバイル＋Deep Learning

- あまり研究がされていない
  - 特に，**実装について(メモリ制限, CPU性能etc...)**

モバイルの特性

- Deep Learningには大規模演算が必須
  - **計算量がボトルネック** → **高速化の工夫は必須**

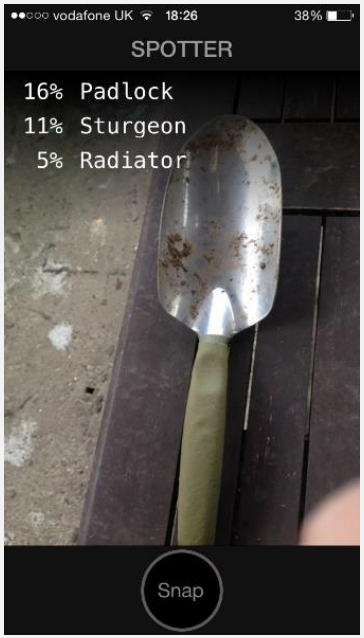
目的

- **モバイル上**で**Deep Learning**の実装
  - 画像認識システムを実装
- 画像認識システムの比較分析
  - 特に，**認識時間**に着目

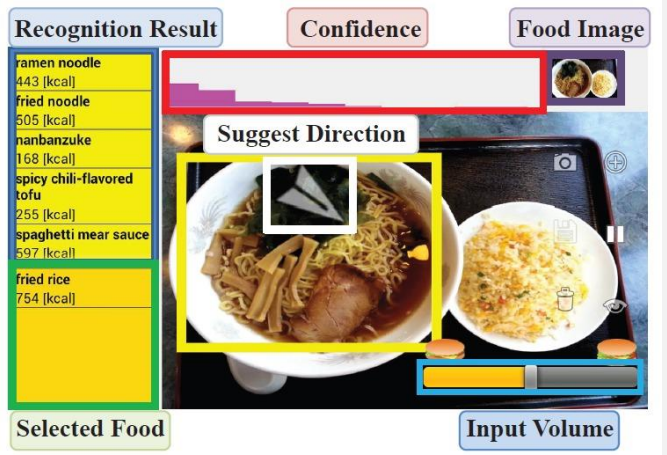
モバイルの特性を考慮したDeep Learningの実装

関連研究

- JetPac Spotter
  - スタンドアロン型1000種類物体認識
  - 認識手法: Deep Learning
  - 非公開(認識フレームワークはOSS)



関連研究

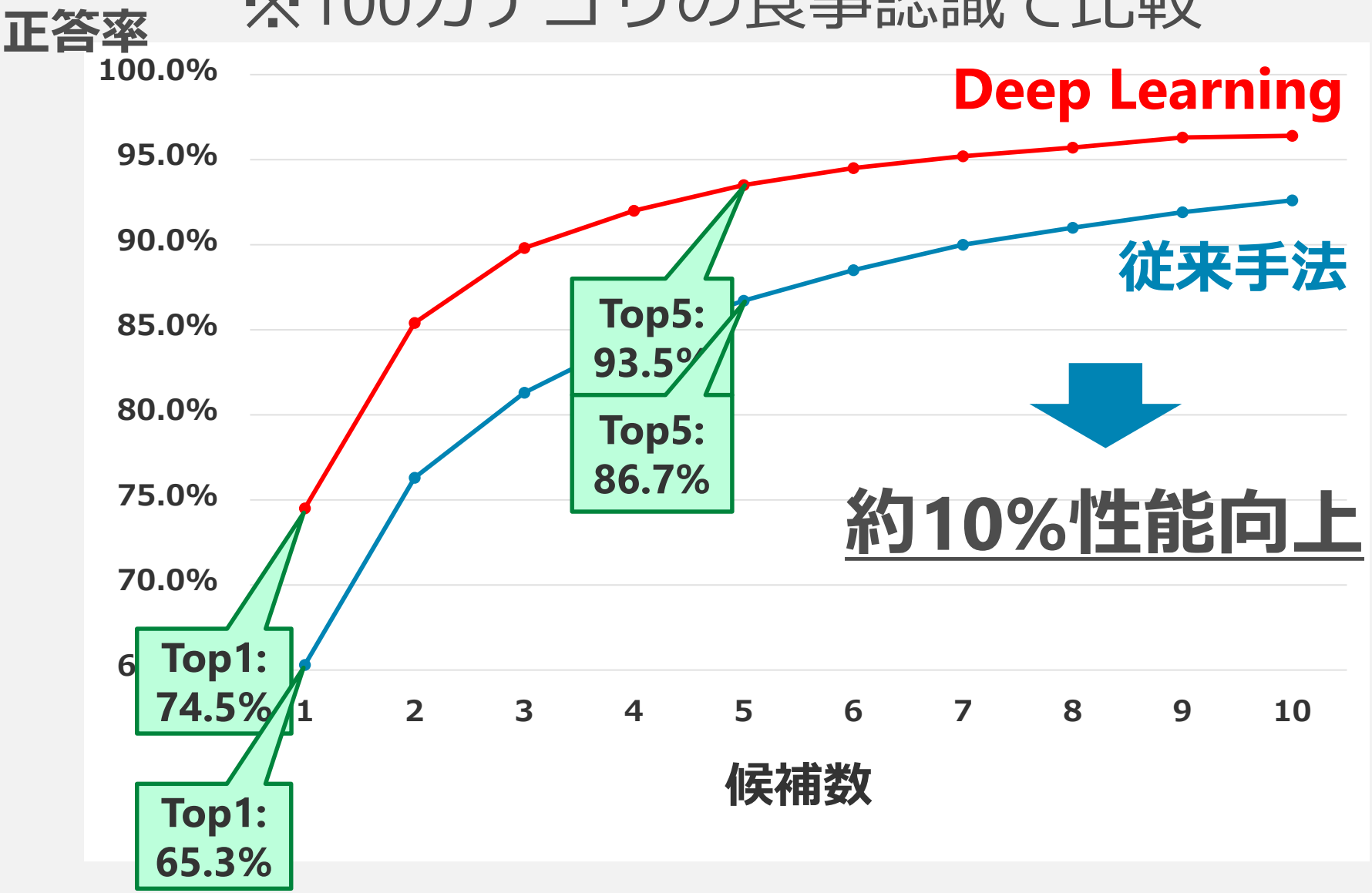


- 河野らの研究(IWMV2013)
  - Android版FoodCam
  - 認識手法：**Fisher Vector(HOG+Color)+線形SVM**
  - 100種類の食事認識

- 岡元らの研究(MIRU2015)
  - Android版DeepFoodCam
  - 認識手法: **Deep Learning**
  - 100種類の食事認識＋非食事

従来手法との比較(認識精度)

※100カテゴリの食事認識で比較



Deep Learningによる認識エンジン

- DCNNの学習
- 畳込み層の高速化

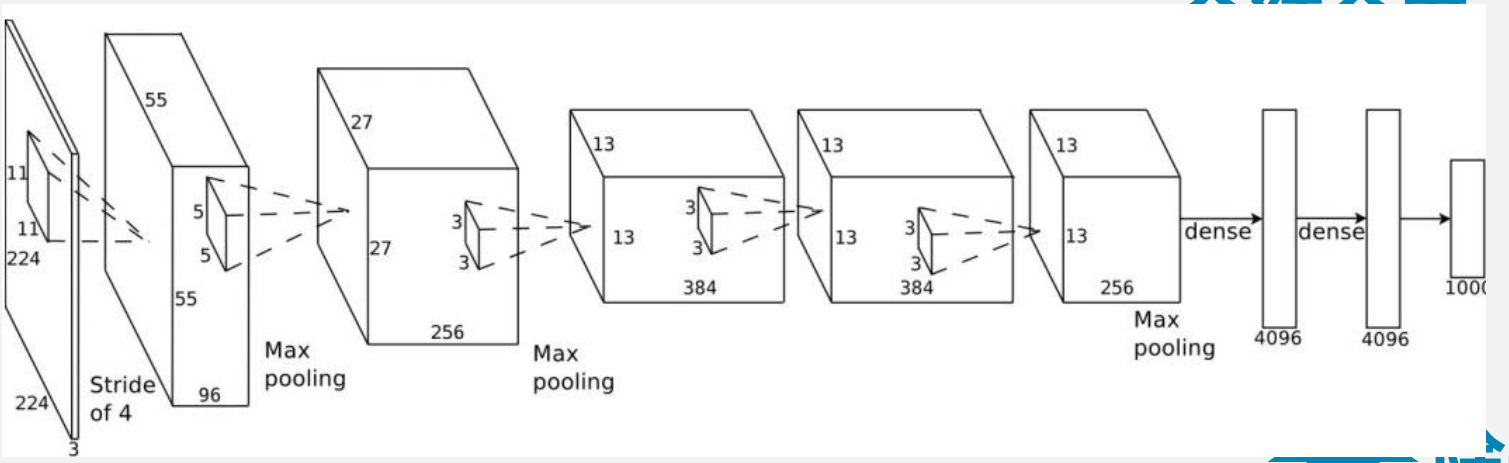
DCNNの学習

- 一般的にはAlexNetを学習モデルに利用
  - AlexNet: 約**6000万**パラメータ → **メモリ制限**

- Network-In-Network(NIN)を採用
  - Network-In-Network: 約**750万**パラメータ

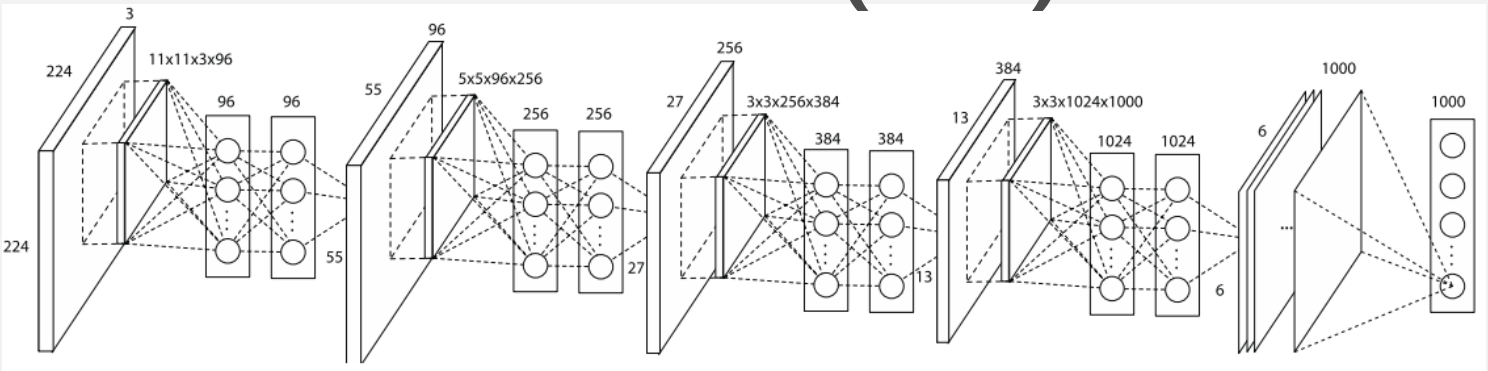
AlexNet vs. NIN

AlexNet

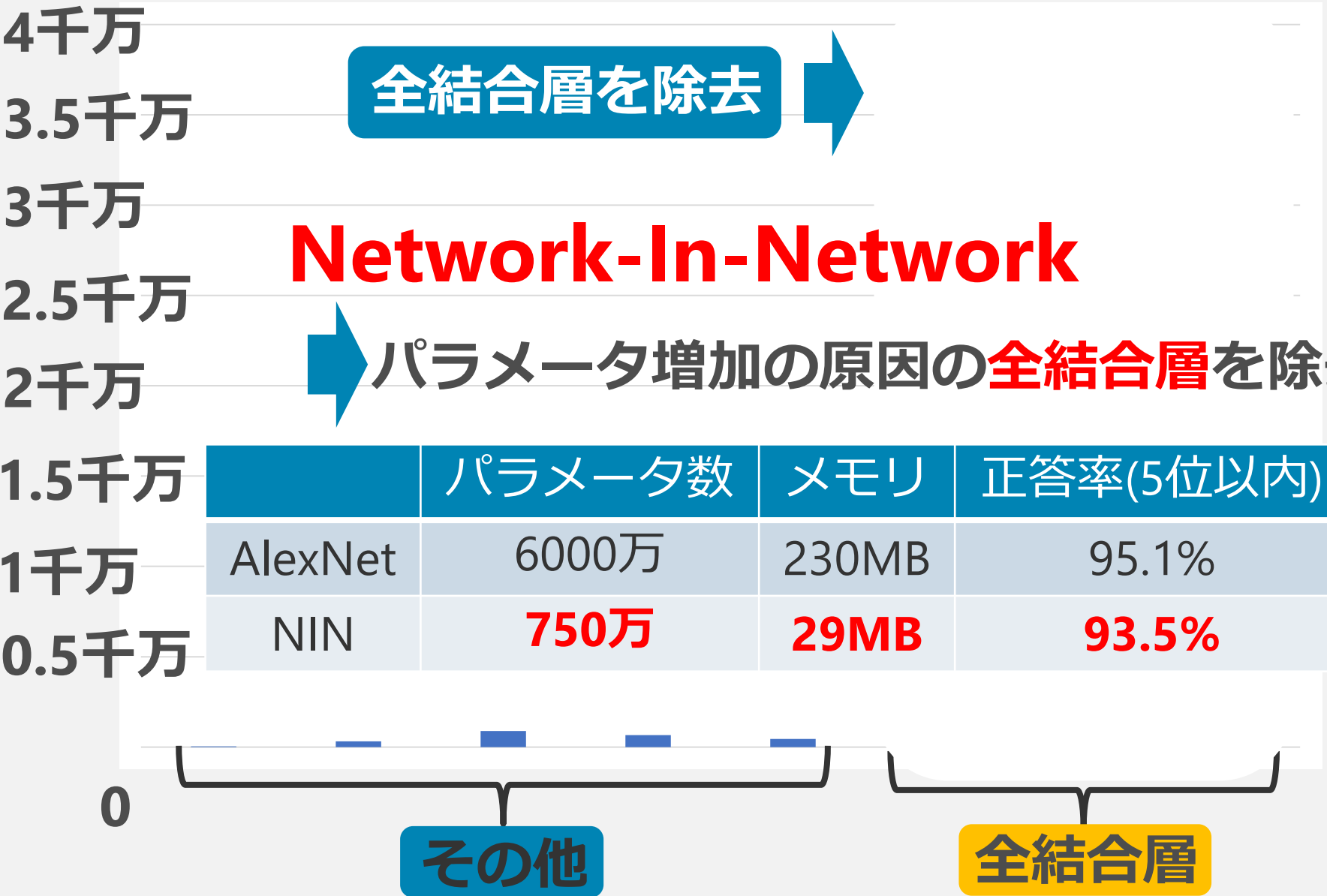


除去

Network-In-Network(NIN)



AlexNetのパラメータ数のグラフ



	パラメータ数	メモリ	正答率(5位以内)
AlexNet	6000万	230MB	95.1%
NIN	**750万**	**29MB**	**93.5%**

畳込み層の高速化の必要性

- 畳込み層が計算時間のボトルネック
  - 畳込み層の**畳込み演算**を高速化
  - **Deep Learning全体の計算時間の削減**

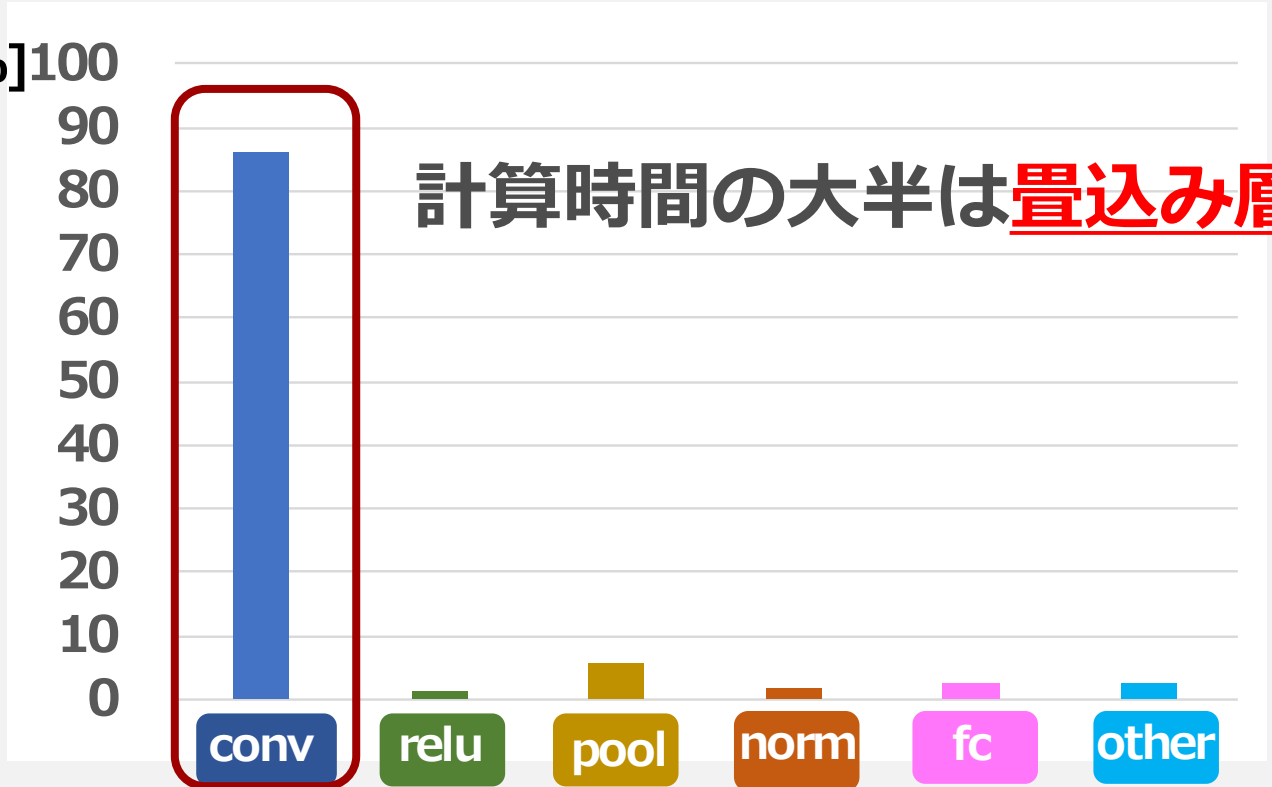
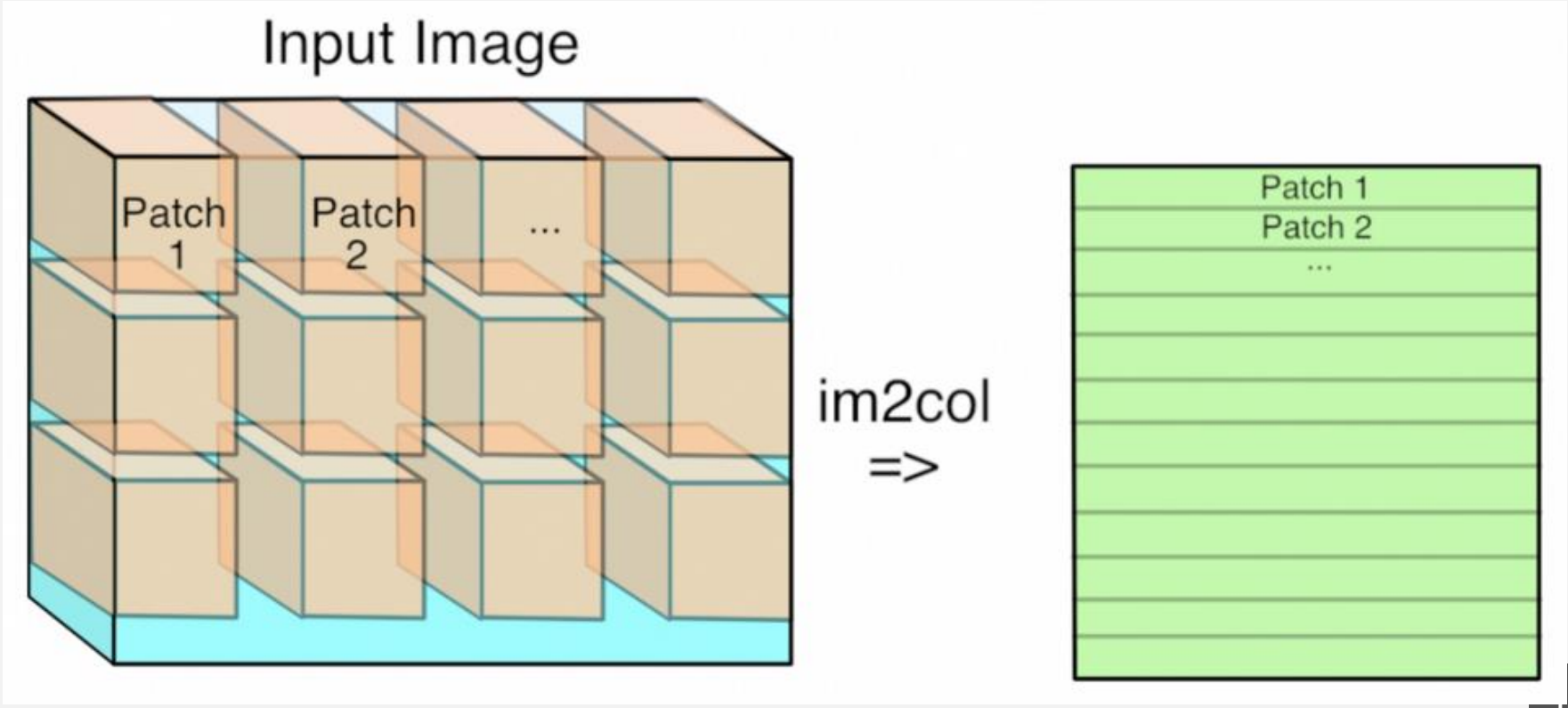
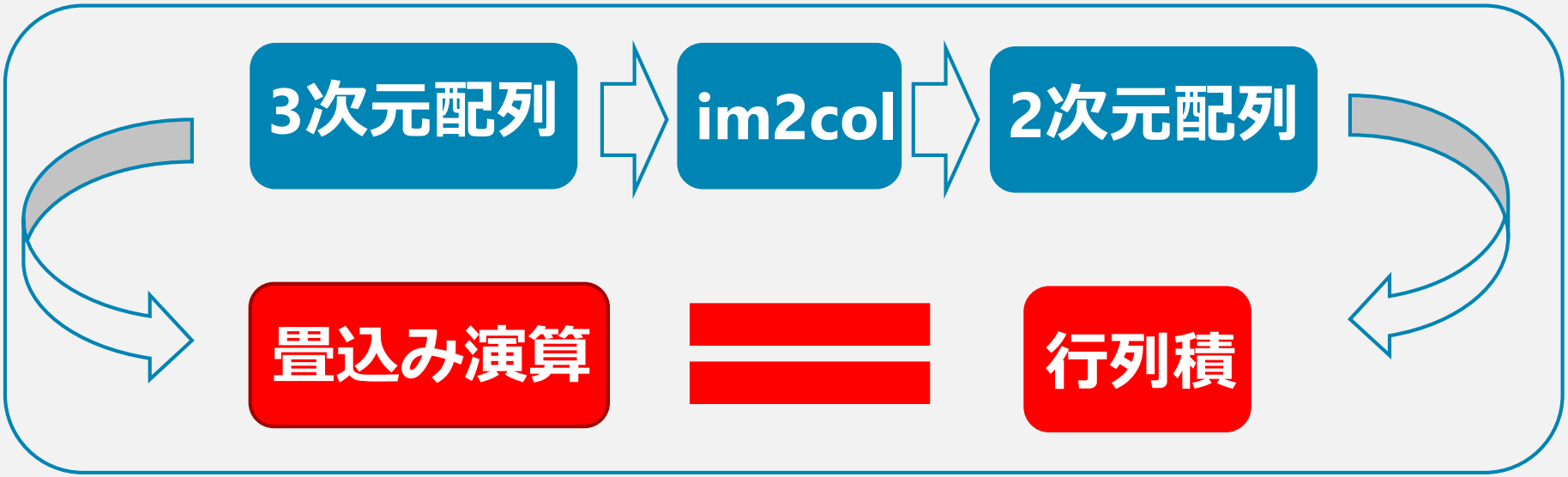


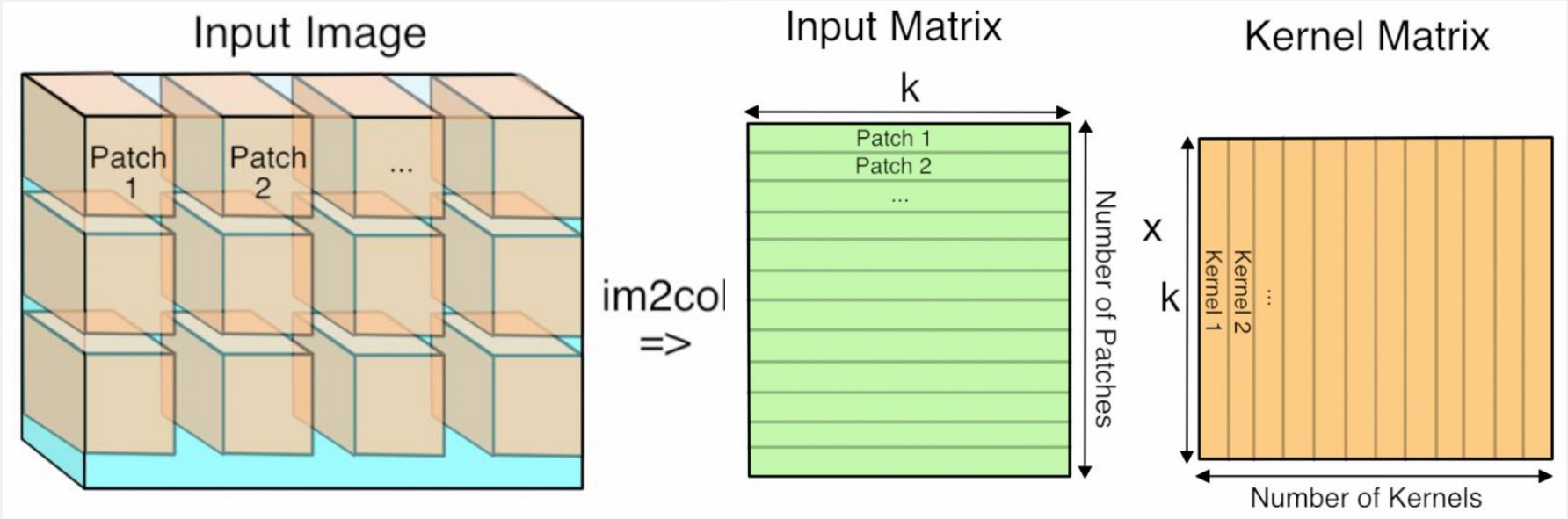
Image to column(Im2col)

- 画像の3次元配列を2次元配列に変換
  - 畳込み演算を行列積に落としこむ



畳込み演算の行列積表現





畳込み演算の高速化

① BLASライブラリ

② NEON命令

Basic Linear Algebra Subprograms(BLAS)

- BLASライブラリ
  - APIを利用
  - Level3 BLAS
    - $C \leftarrow \alpha AB + \beta C$ 行列と行列の積
  - iOS: **Accelerate Framework**,
  - Android: **OpenBLAS**

NEON命令

- NEON命令
  - 同時に4つの32bit単精度浮動小数点の演算
  - iOS: 2Core*4 = **8演算**
  - Android: 4Core*4 = **16演算**

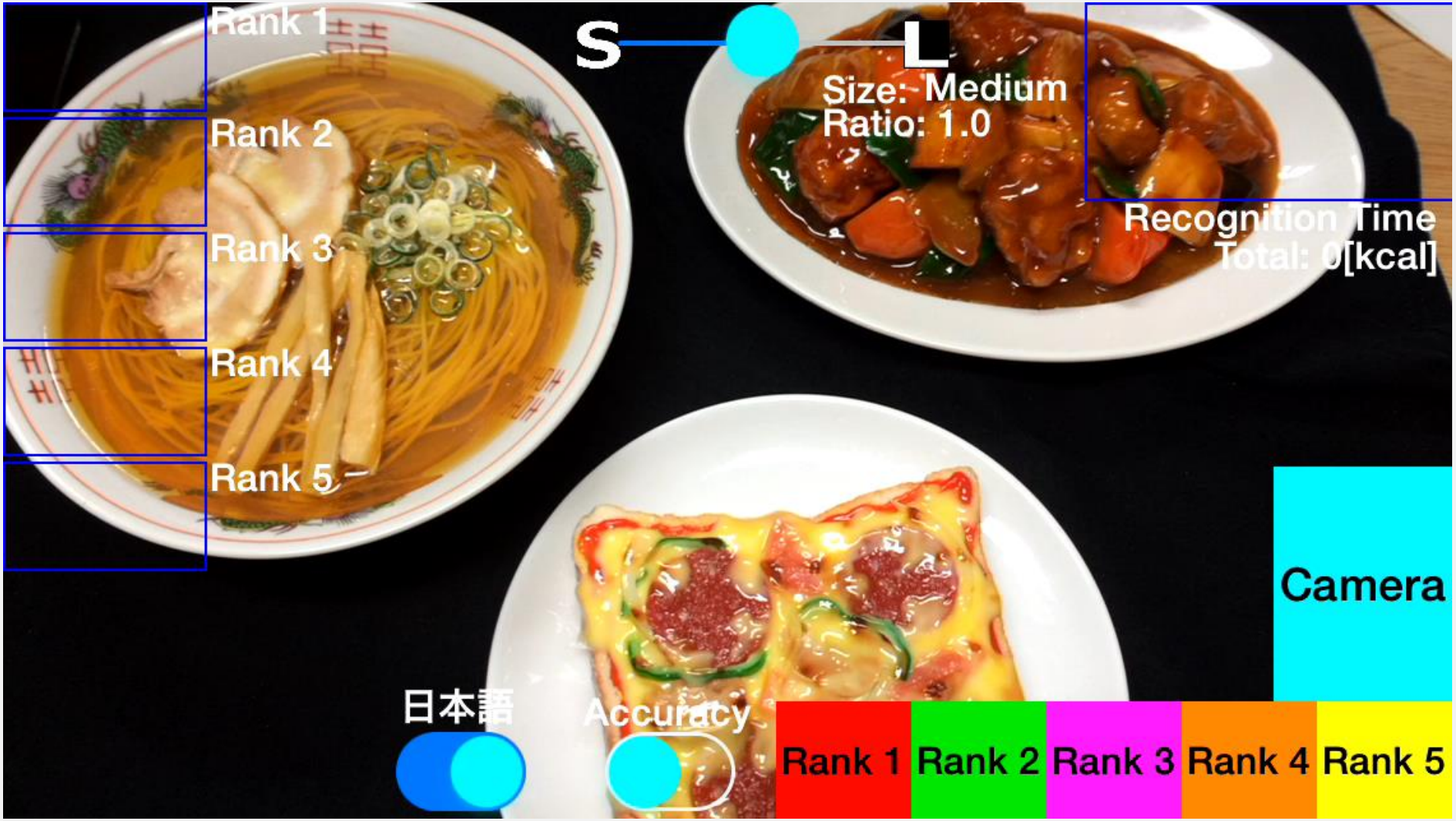
作成したアプリ

- 2000種類(非食事1000+食事1000)を認識することができるアプリ
  - **2000CategoryCam**
- 101種類(非食事+食事100種類)を認識することができるアプリ
  - **iOS版DeepFoodCam**

2000種類物体認識アプリ

2000種類物体認識アプリ

101種類(食事100+非食事)物体認識アプリ



画像認識システムの比較分析

Android

岡元らの研究(MIRU2015)
Android版: DeepFoodCam

Deep Learning
フレームワーク

DeepBeliefSDK

DeepFoodCam

- BLAS
- NEON

iOS上に実装

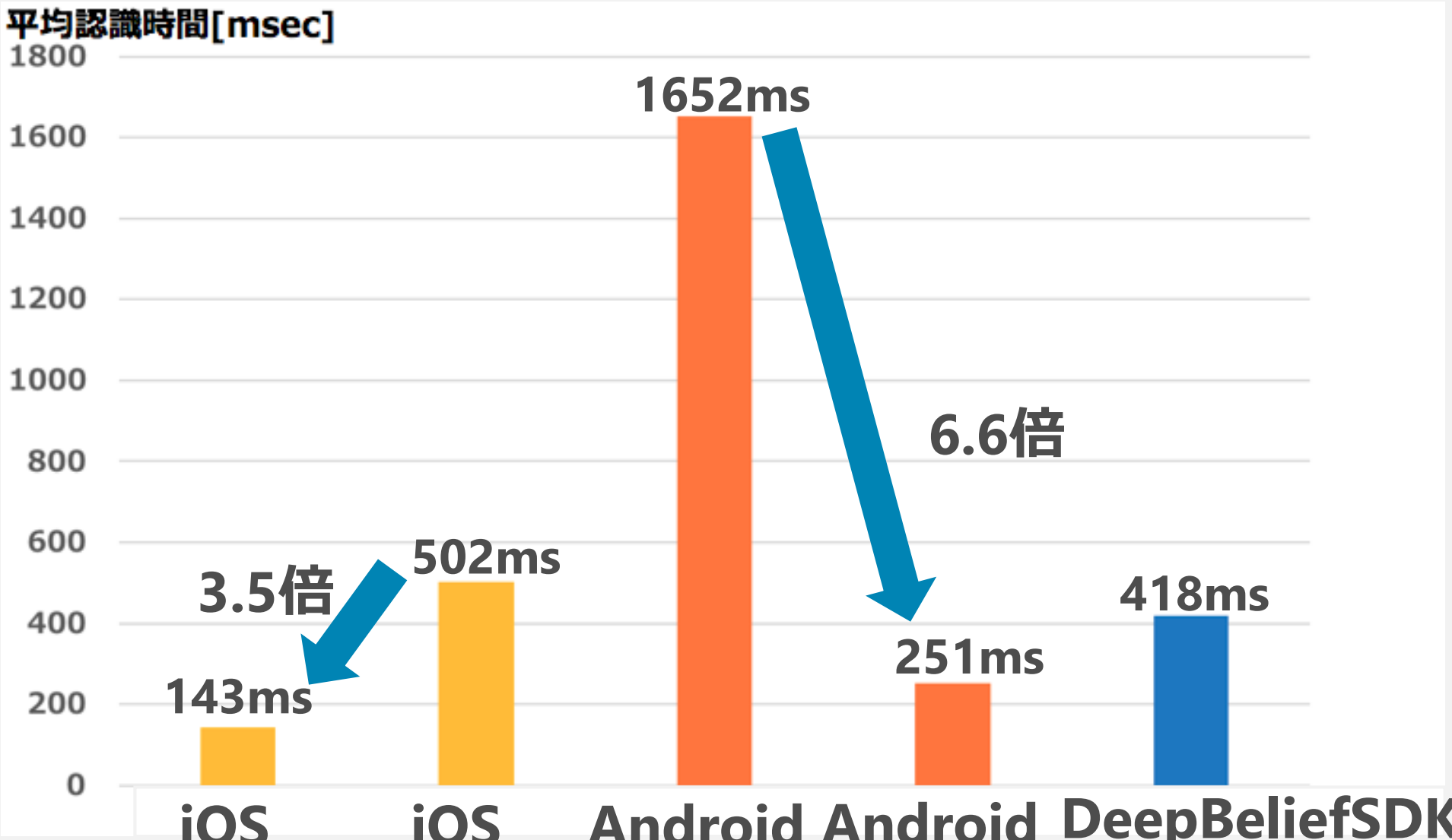
比較分析

実験設定

OS	実験番号	高速化手法	フレームワーク
iOS	(1)	BLAS	iOS(BLAS)
iOS	(2)	NEON	iOS(NEON)
Android	(3)	BLAS	Android(BLAS)
Android	(4)	NEON	Android(NEON)
iOS	(5)	BLAS	DeepBeliefSDK

- 実験目的
  - モバイル上での認識時間の計測
- 評価デバイス(2台)
  - iOS: iPhone 5s(1.3GHz, RAM1GB, 2Core)
  - Android: GALAXY Note3(2.3GHz, RAM3GB, 4Core)
- 評価方法
  - 5パターンの実験
  - 認識時間を20回計測して平均を算出

結果


平均認識時間[msec]
1800
1600
1400
1200
1000
800
600
400
200
0
143ms
502ms
1652ms
251ms
418ms
3.5倍
6.6倍
iOS (BLAS)
iOS (NEON)
Android (BLAS)
Android (NEON)
DeepBeliefSDK

考察

- iOS: Accelerate framework
  - Appleが提供
  - デバイスに最適化
- Android: OpenBLAS
  - 様々なCPUに対応
  - デバイスに最適化されていない

まとめ

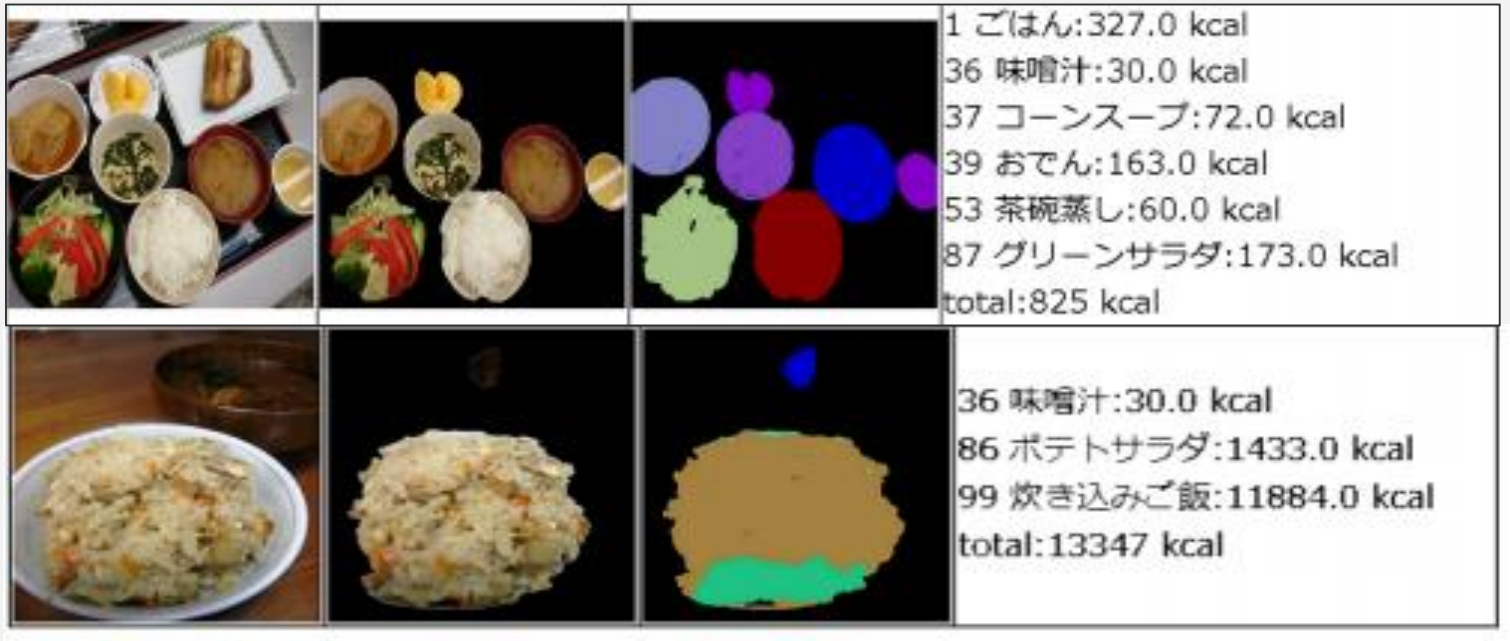
- モバイル上でDeep Learningの実装
  - 画像認識システムを実装

	Top 5	認識時間[ms]
Deep Learning	**93.5%**	**143[ms]**
従来手法	**86.7%**	**96[ms]**

- 画像認識システムの比較分析
  - **演算の最適な高速化工夫はモバイルOSで異なる**
  - iOS: Accelerate Framework(BLAS)
  - Android: NEON命令

今後の課題

- CNNによる，より複雑な処理の実装
  - 領域分割
  - 動画像のリアルタイム処理



- GPGPU

質疑応答

まとめ

- モバイル上でDeep Learningの実装
  - 画像認識システムを実装
- 画像認識システムの比較分析
  - 演算の高速化工夫はモバイルOSで異なる

今後の課題

- CNNによる，より複雑な処理の実装
  - 領域分割
  - 動画像のリアルタイム処理
  - GPGPU