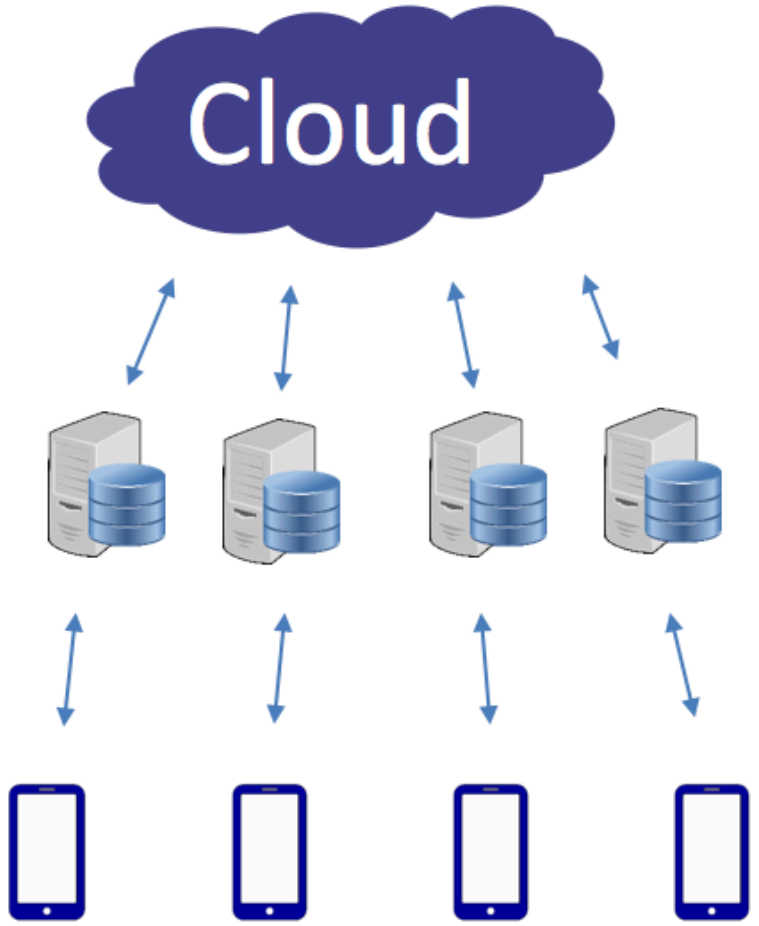
Core MLによる
iOS深層学習アプリの実装と性能分析

丹野 良介，泉 裕貴，柳井 啓司
電気通信大学 大学院情報理工学研究科
情報学専攻

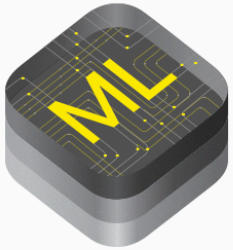
研究背景

- 深層学習をサーバ側で学習し，
  エッジ側で推論を行う
  Edge Computingによる利用の増加
  - サーバ側の負担軽減
  - セキュリティ面の向上



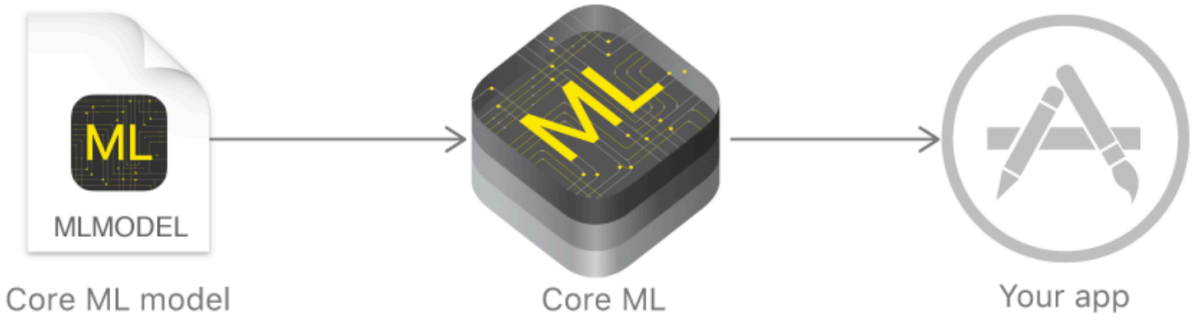
- 大量の畳込み演算を
  高速に処理する必要性
  - AlexNet: 約11億回

Core MLはご存知でしょうか？



Build more intelligent apps with machine learning.

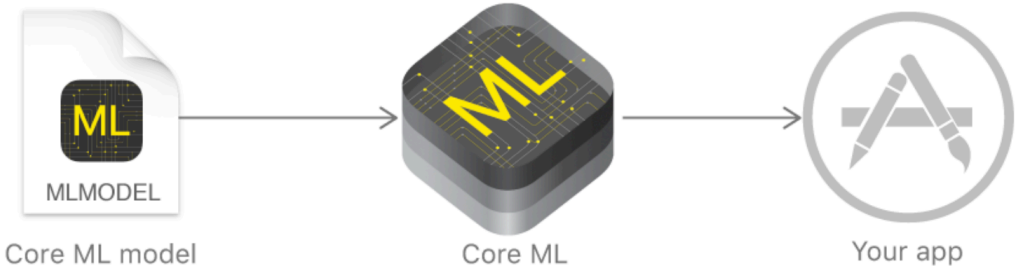
Take advantage of Core ML a new foundational machine learning framework used across Apple products, including Siri, Camera, and QuickType. Core ML delivers blazingly fast performance with easy integration of machine learning models enabling you to build apps with intelligent new features using just a few lines of code.

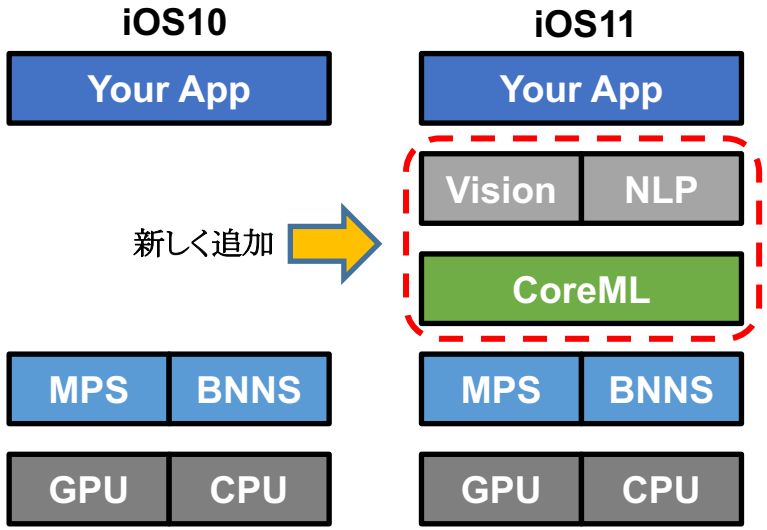


Core ML(ML: Machine Learning)とは

- iOS11から追加された機械学習API群(2017年9月一般公開)
  - 学習済みモデルをiOS上で推論できるようにモデルを変換して利用
  - MPS(GPU), BNNS(CPU)のラッパー

- iOS10では．．．(Apple提供)
  - MPS(Metal Performance Shaders)
    - GPU利用のニューラルネットワークAPI
  - BNNS(Basic Neural Network Subroutines)
    - CPU利用のニューラルネットワークAPI

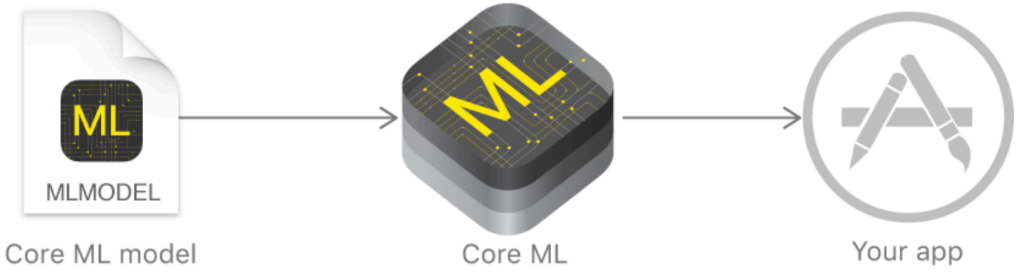


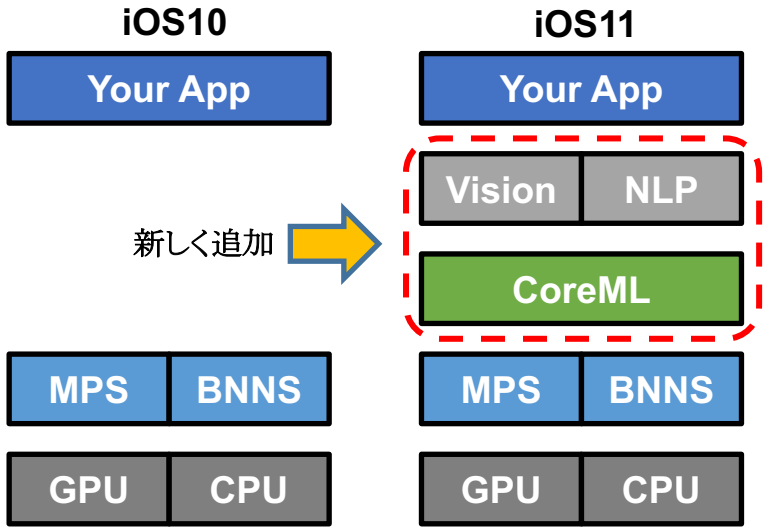


Core ML(ML: Machine Learning)とは

- 問題点(iOS10)
  - 独自で各API(MPS(GPU), BNNS(CPU))を利用したネットワーク構造を書く必要性
  - 学習したモデルをiOS用のフォーマットに書き出す必要性
  - **CPU, GPUを使うのは敷居が高かった**

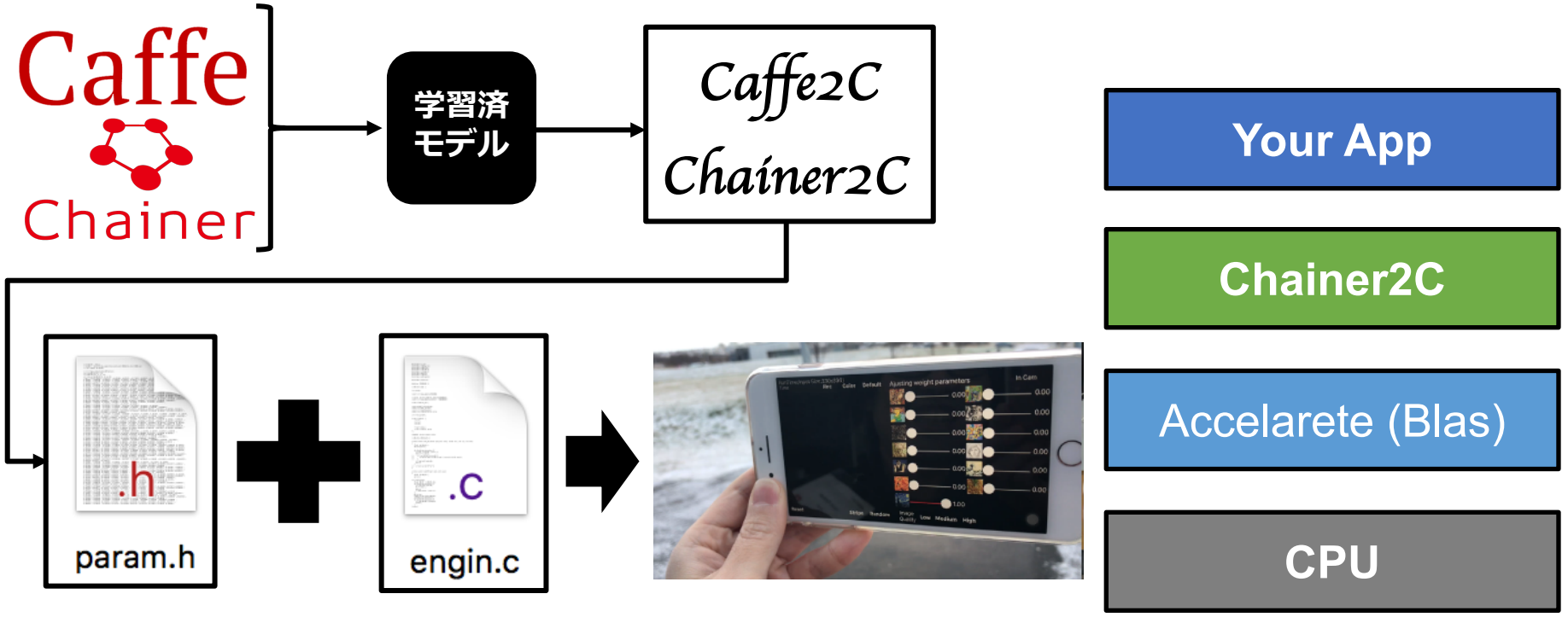
- 上記の問題点を解決したのがCore ML
  - **疑問**
  - **「パフォーマンスはどの程度？」**





一方，弊研究室のこれまでの研究

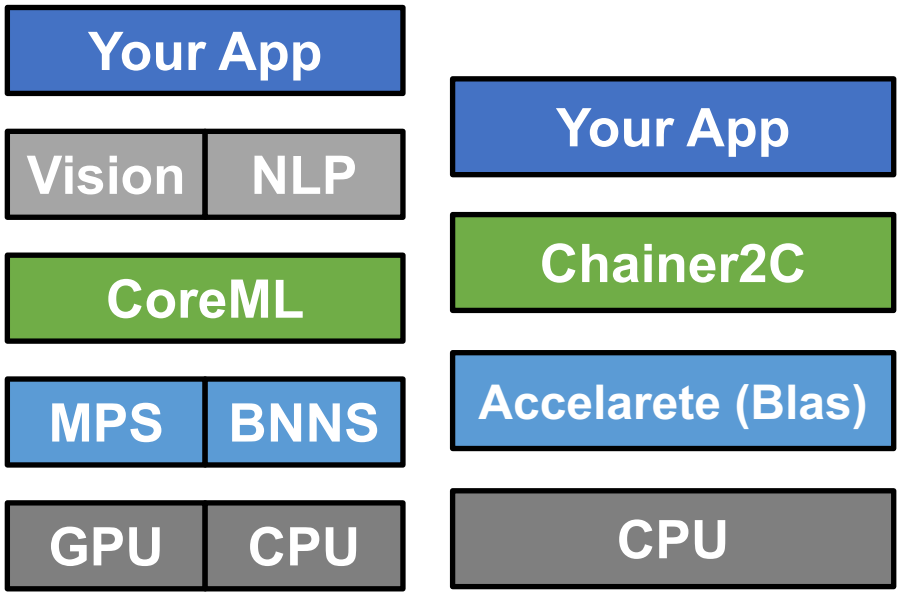
- Chainer2C(Caffe2C)
  - Chainer/Caffeで学習したネットワークを iOS/Androidで動作可能なC言語コードに**自動変換するモデルコンバータ**
  - オリジナルの順伝搬演算ライブラリとリンク → **高速動作可能**
  - **CPUのみ利用**



目的

- Core ML(GPU利用)とChainer2C(CPU利用)を比較
  - 両者の推論処理部分のベンチマークを取って，パフォーマンスを比較検討

- 本当にGPUって早いの？
- Core MLを使うと何が便利なの？
- どこまでの処理がiPhoneで実現可能なの？

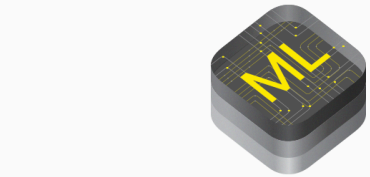


深層学習iOSアプリ実装レシピ

- 外部ライブラリ利用
  - TenforFlow for iOS, OpenCV
  - Caffe2など…

- GPU利用(Core ML)

- CPU利用(Chainer2C)



Build more intelligent apps with machine learning.

Take advantage of Core ML a new foundational machine learning framework used across Apple products, including Siri, Camera, and QuickType. Core ML delivers blazingly fast performance with easy integration of machine learning models enabling you to build apps with intelligent new features using just a few lines of code.

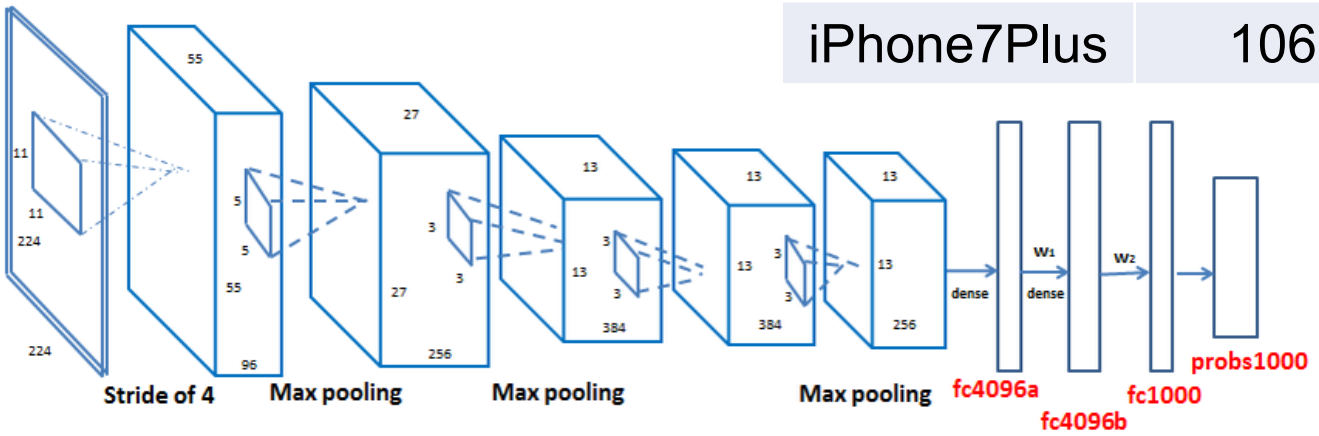
外部ライブラリ利用

- フレームワーク自体が iOS 上で動くように設計されている．または，学習済みモデルを利用するラッパーが用意されている．
  - 例：TensorFlow for iOS, OpenCV, Caffe2…

- 最適化が不十分なためパフォーマンス低い
  - Chainer2C(最適化)
  - OpenCV DNN(非最適化)
  - 16倍ほど遅い

AlexNetの推論結果[ms]

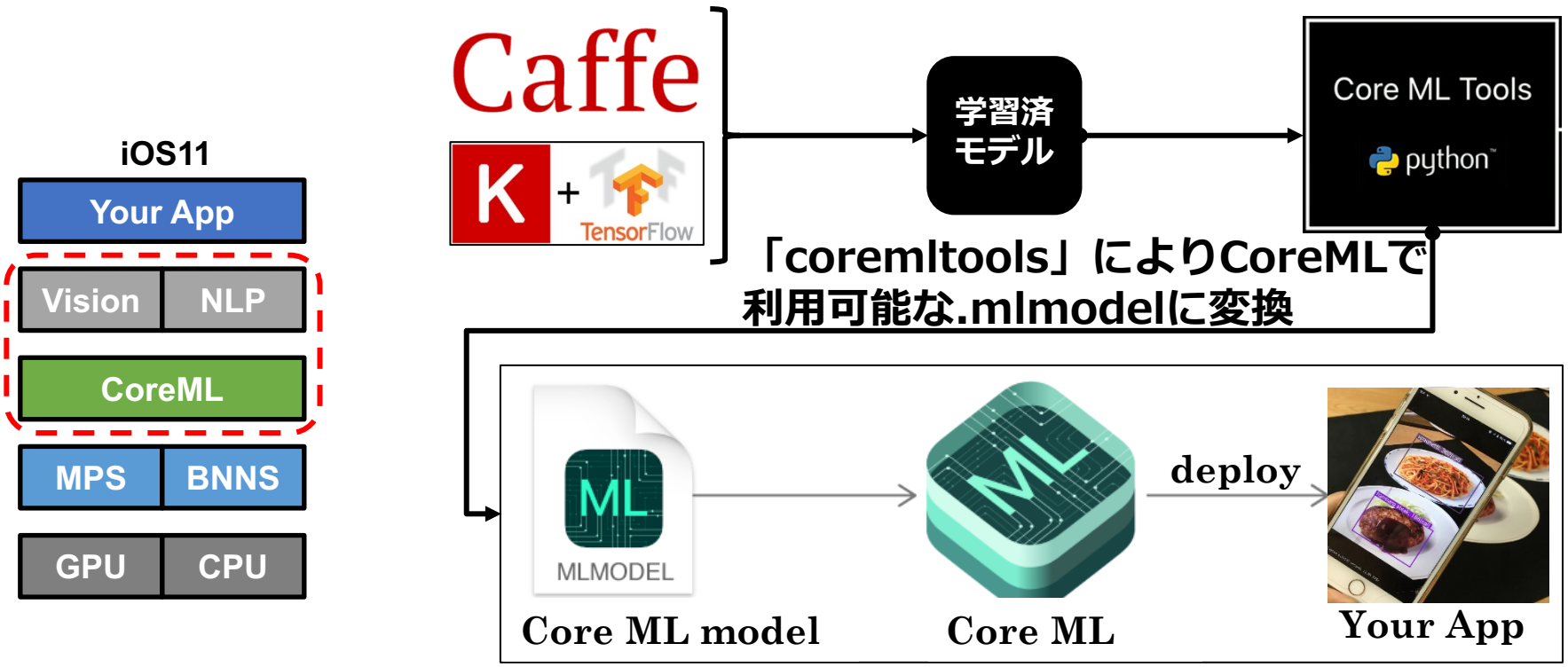
	Chainer2C	OpenCV DNN
	AlexNet	
iPhone7Plus	106.9	1663.8



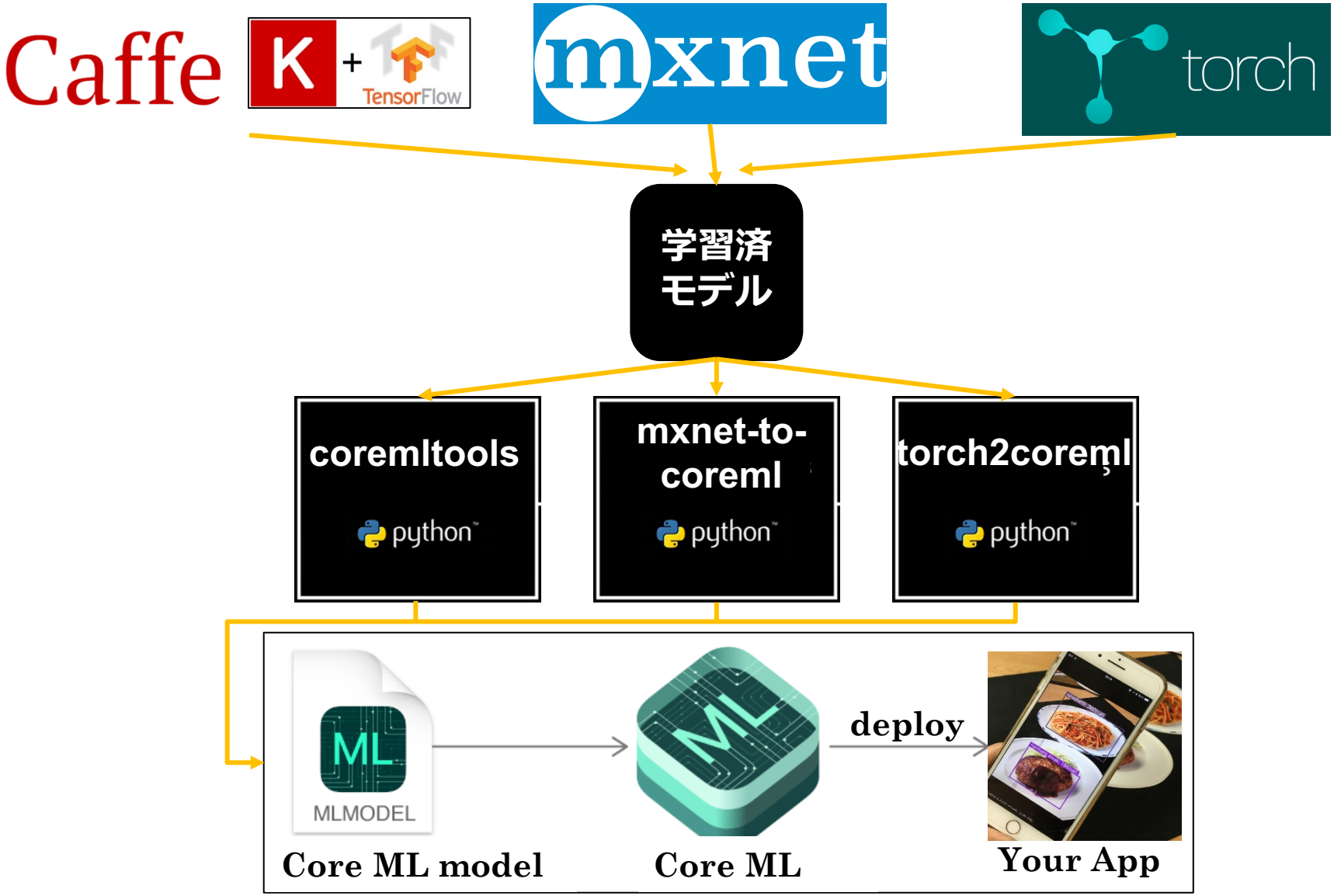
右図)AlexNet

GPU利用(Core ML)

- **学習済みモデル**をiOS上で推論できるように**モデルを変換して利用**
  - OSSモデルコンバータ「coremltools」を利用

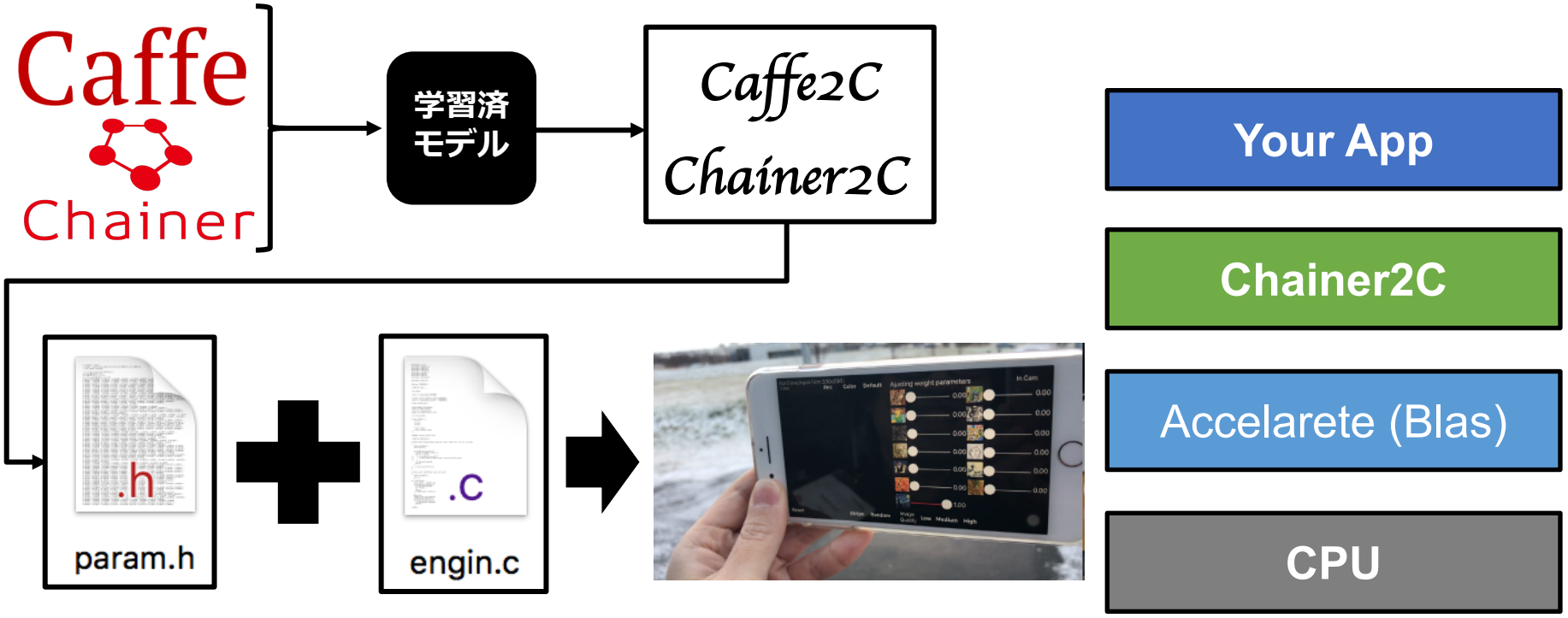


CoreMLがサポートするフレームワーク群



CPU利用(Chainer2C)

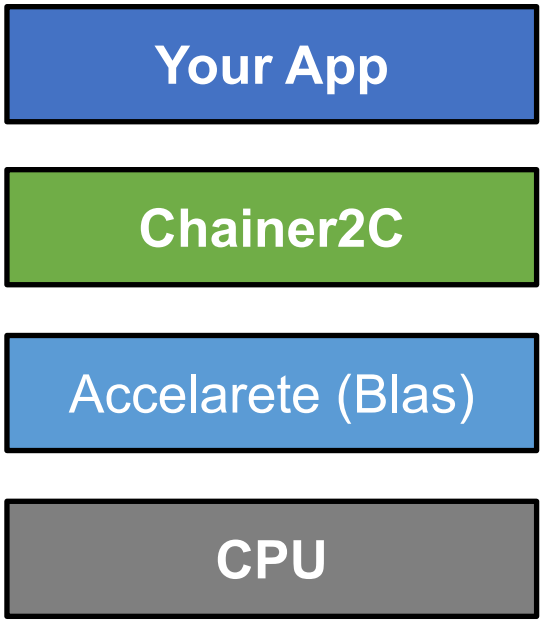
- Chainer2C(Caffe2C)
  - Chainer/Caffeで学習したネットワークを iOS/Androidで動作可能なC言語コードに**自動変換するモデルコンバータ**
  - オリジナルの順伝搬演算ライブラリとリンク → **高速動作可能**
  - **CPUのみ利用**

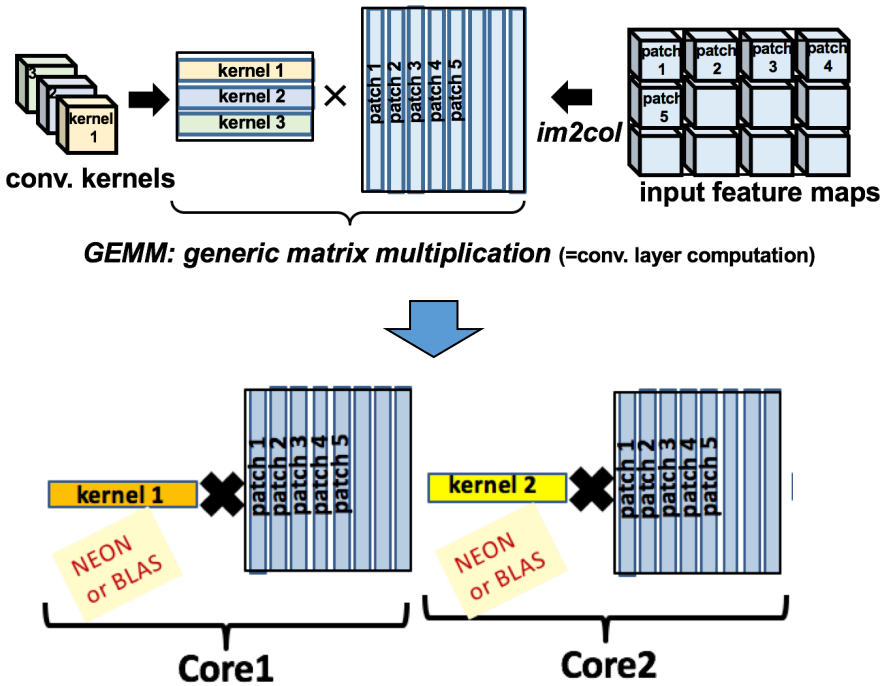


Chainer2Cの高速動作可能な理由

- Multithread 化による NEON/BLAS の効率的な利用
  - Acceralate (内部的にはNeon命令(ARM のSIMD命令)を利用
  - 32bit浮動小数点演算が各スレッドについて4並列実行
    = 8並列演算

- CNN に掛かる演算の可能な限りの事前計算の実行
  - CONV＋BNの事前計算によるBN計算の省略



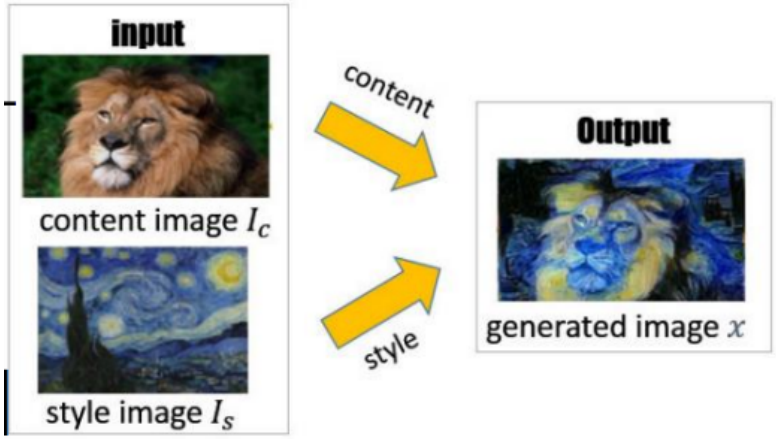


ベンチマーク

- 認識タスク(iPhone8/7)
  - AlexNet
  - AlexNet+BN
  - Network-In-Network(NIN)
  - **NIN+BN**
  - GoogLeNet
  - Inception-v3(299x299)
  - ResNet50
  - VGG16
  - VGG19
  - MobileNet
  - SqueezeNet(227x227)

- 基本的に元論文の実装のまま
- 入力サイズは224x224
(例外有)

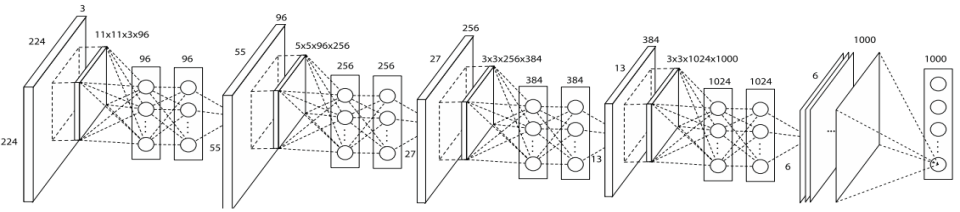
- 変換タスク(iPad Pro2017)
  - Fast Neural Style Transfer
  - 入力サイズは640x480



Network-In-Network + BN

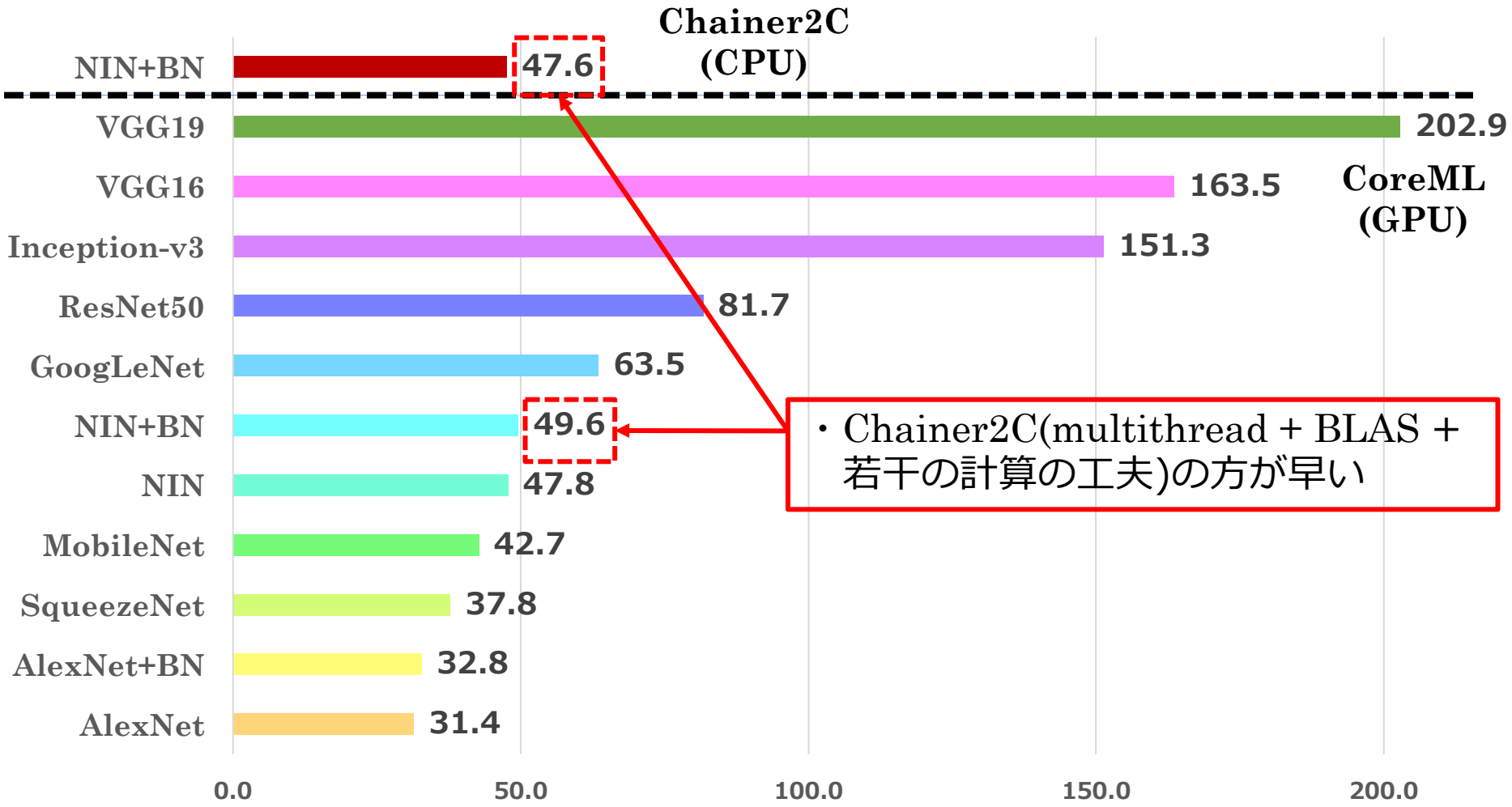
- 元論文との主な差異
  - ReLU関数の直前にBNを配置
    - Batch Normalization
  - 5x5 conv -> 3x3 conv
  - conv4のfeature mapsを 1024 -> 768に下げる

layer no.	(1) original NIN	(2) 4layers+BN
1	11x11x96 **conv1**	11x11x96 **conv1**
2	1x1x96 cccp1_1	1x1x96 cccp1_1
3	1x1x96 cccp1_2	1x1x96 cccp1_2
4	5x5x256 **conv2**	3x3x256 **conv2_1**
5	1x1x256 cccp2_1	3x3x256 **conv2_2**
6	1x1x256 cccp2_2	1x1x256 cccp2_1
7	3x3x384 **conv3**	1x1x256 cccp2_2
8	1x1x384 cccp3_1	3x3x384 **conv3**
9	1x1x384 cccp3_2	1x1x384 cccp3_1
10	3x3x1024 **conv4**	1x1x384 cccp3_2
11	1x1x1024 cccp4_1	3x3x768 **conv4**
12	1x1xN cccp4_2	1x1x768 cccp4_1
13	avg. pool	1x1xN cccp4_2
14	softmax	avg. pool
15		softmax
16		
17		
18		
weights	7.6Million	5.5Million
computation	1.1Billion	1.2Billion



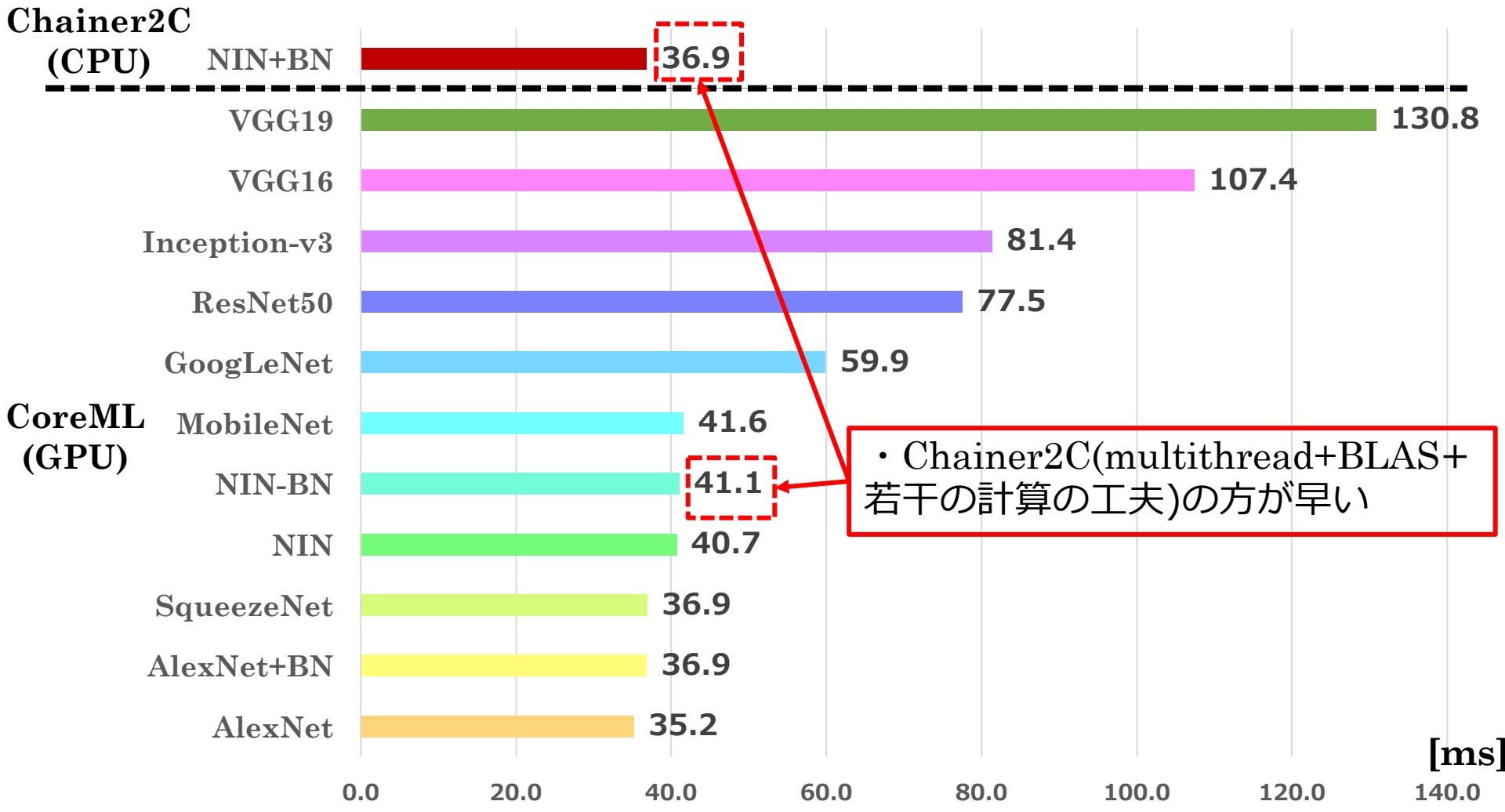
認識タスクベンチマーク（iPhone7Plus）

- CoreML vs. オリジナル順伝搬ライブラリ
  - GPU vs. CPU only

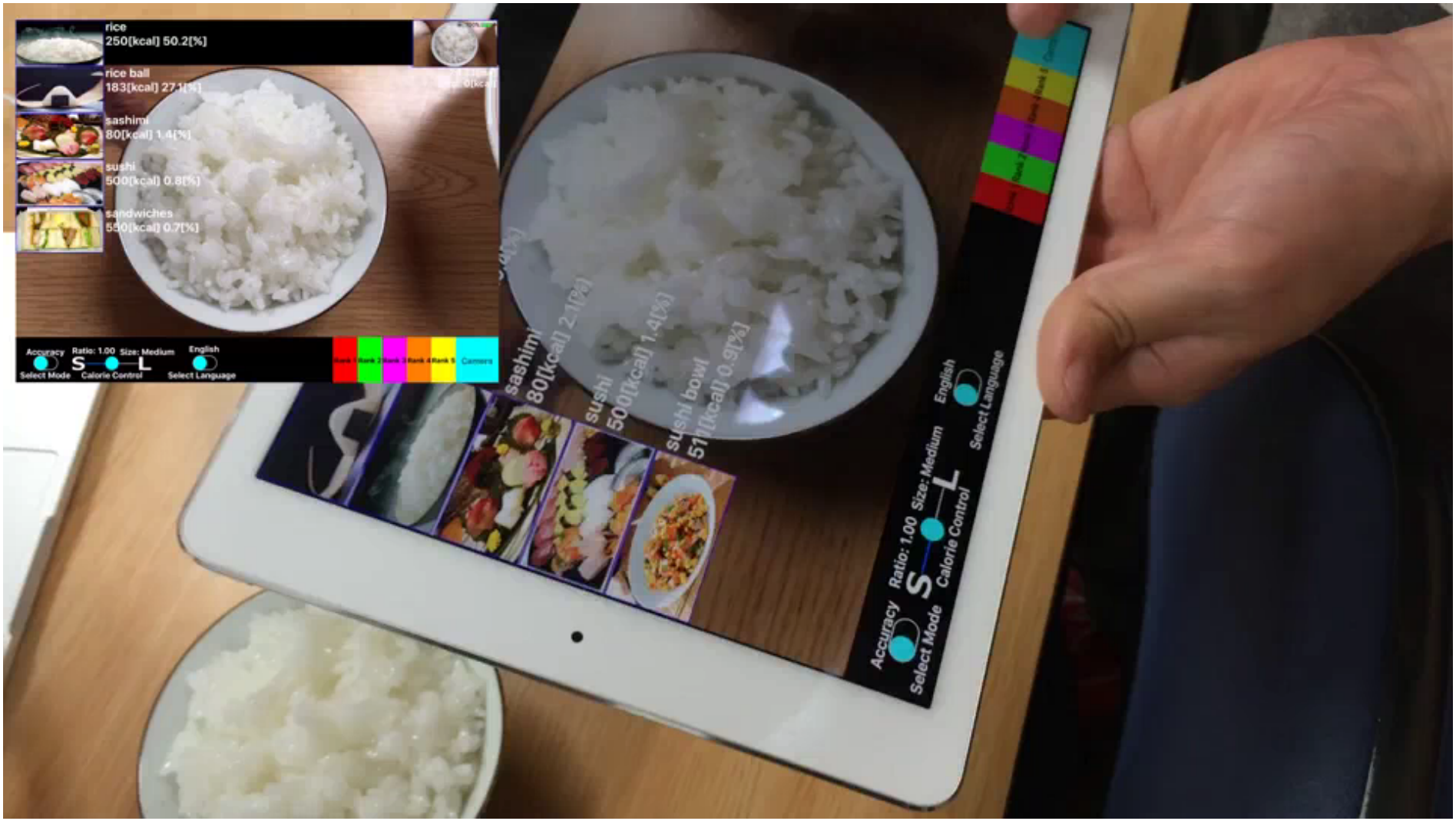


実行速度ベンチマーク（iPhone8Plus）

- CoreML vs. オリジナル順伝搬ライブラリ
  - GPU vs. CPU only

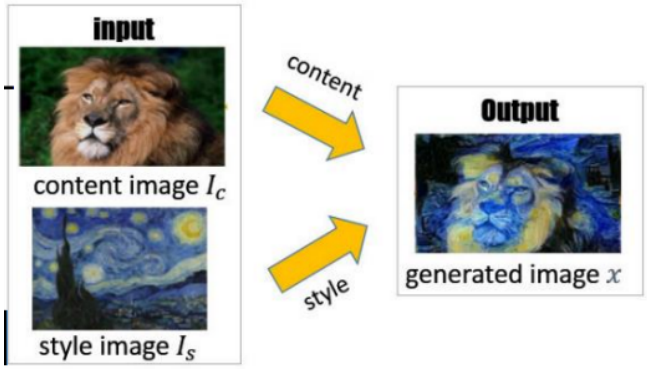


NIN+BN(Chainer2C)を利用したデモ動画



Fast Neural Style Transfer

- 特定の画風変換を順伝搬で行うCNNを学習



- 元論文との差異
  - down-sampling layerと up-sampling layerの追加
  - 最初と最後のconvのカーネルを 9x9 conv -> 5x5 conv

CoreML	オリジナル順伝搬ライブラリ
Reflection Padding (40x40)	5x5x16 conv, stride 1
9x9x32 conv, stride 1	4x4x32 conv, stride 2
3x3x64 conv, stride 2	4x4x64 conv, stride 2
3x3x128 conv, stride 2	4x4x128 conv, stride 2
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	Residual block, 128 filters
3x3x64 conv, stride 1/2	4x4x64 conv, stride 1/2
3x3x32 conv, stride 1/2	4x4x32 conv, stride 1/2
9x9x3 conv, stride 1	4x4x16 conv, stride 1/2
	5x5x3 conv, stride 1
512[msec]	841[msec]

- 推論処理時間(640x480)
  - Core ML: 512 [ms]
  - Chainer2C: 841 [ms]
  - iPad Pro 2017

スタイル変換(デモ動画)Chainer2C

23rd MMM 2017

Tanno, R. and Yanai, K.: DeepStyleCam: A Real-time Style Transfer App on iOS, Proc. of International MultiMedia Modeling Conference (MMM) (2017)

ベンチマークを終えて

- Chainer2Cの場合(CPU利用)
  - **2つの高性能コア**と4つの高効率コアの合計6コア(iPhone8)
  - 2core * 4 = 最大8並列演算
- Core ML(GPU利用)
  - Apple独自設計 3 コア GPU * 4 = 最大12並列演算
  - 16bit浮動小数点演算

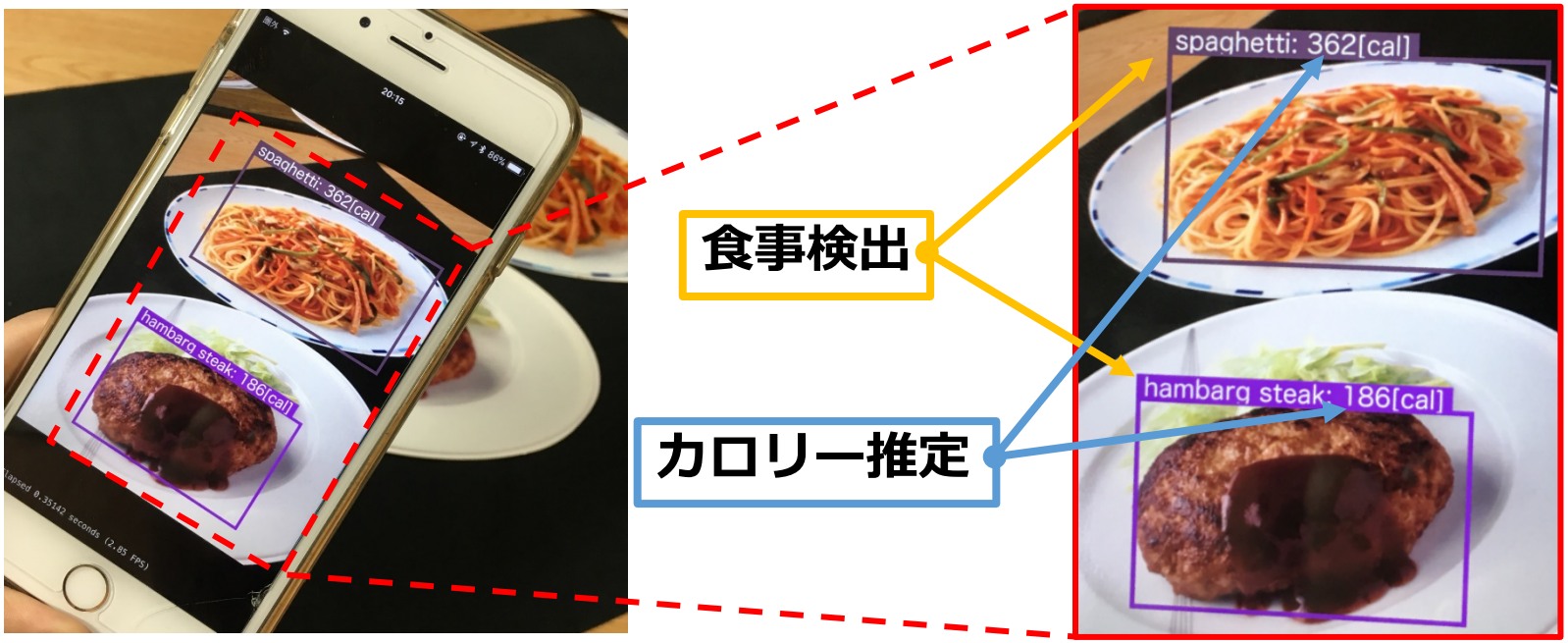
- 認識タスクならそれほどパフォーマンスに差はない
  - むしろCPUの方が若干早い
  - CNNの処理時間 < GPUのオーバーヘッド

- 高解像度の画像を処理するなど，高負荷の処理をする場合は，GPU利用が効果的
  - スタイル変換の例

実装例：食事検出＋カロリー推定

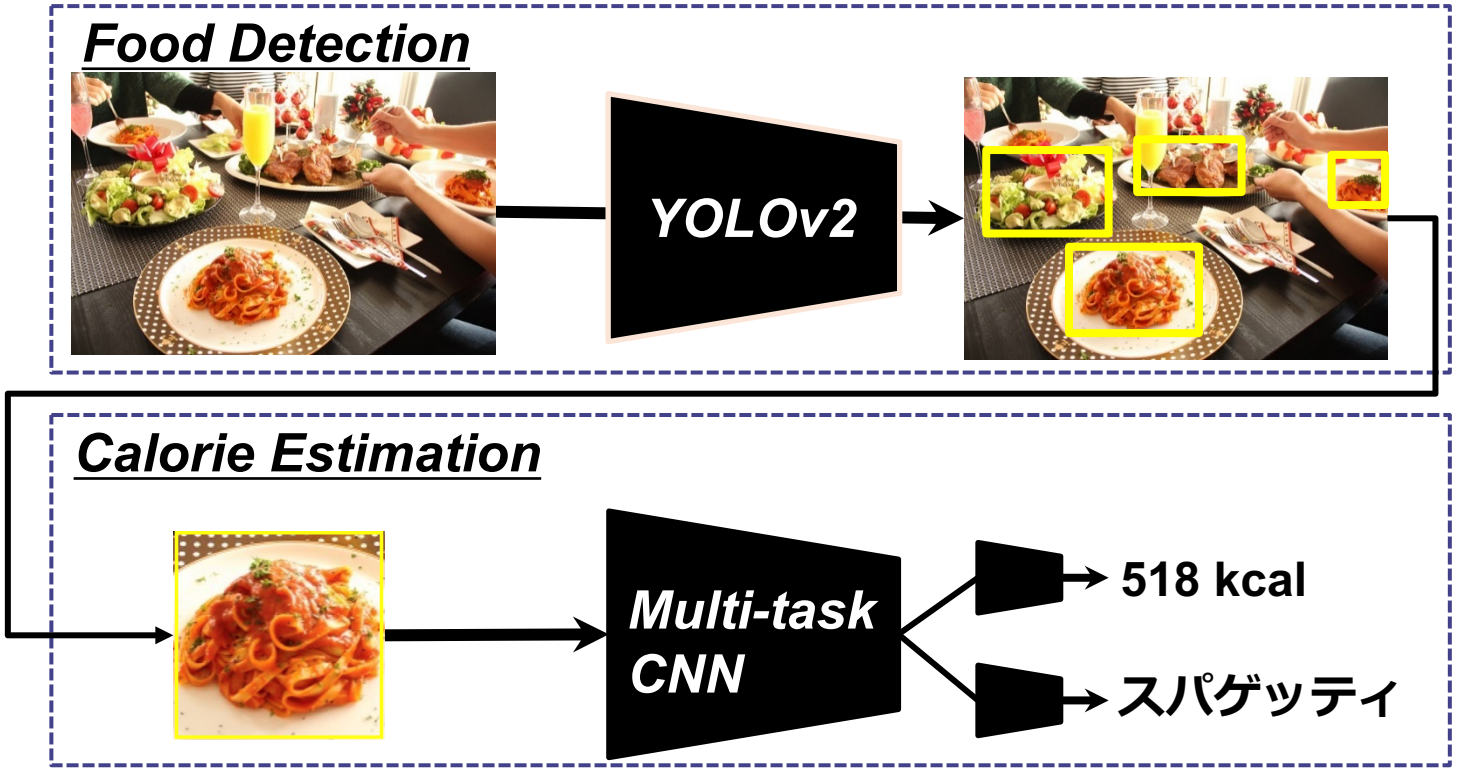
- 物体検出とカロリー推定を組み合わせたアプリ
  - 物体検出：YOLOv2[Redmon et al, CVPR, 2017]
  - カロリー推定：Multi-task CNN[Ege et al, MVA, 2017]
    - **カロリー推定の部分は午後のセッション**で！

(13)	13:00-13:30	CNNによる複数料理写真からの同時カロリー量推定	○會下拓実・柳井啓司（電通大）

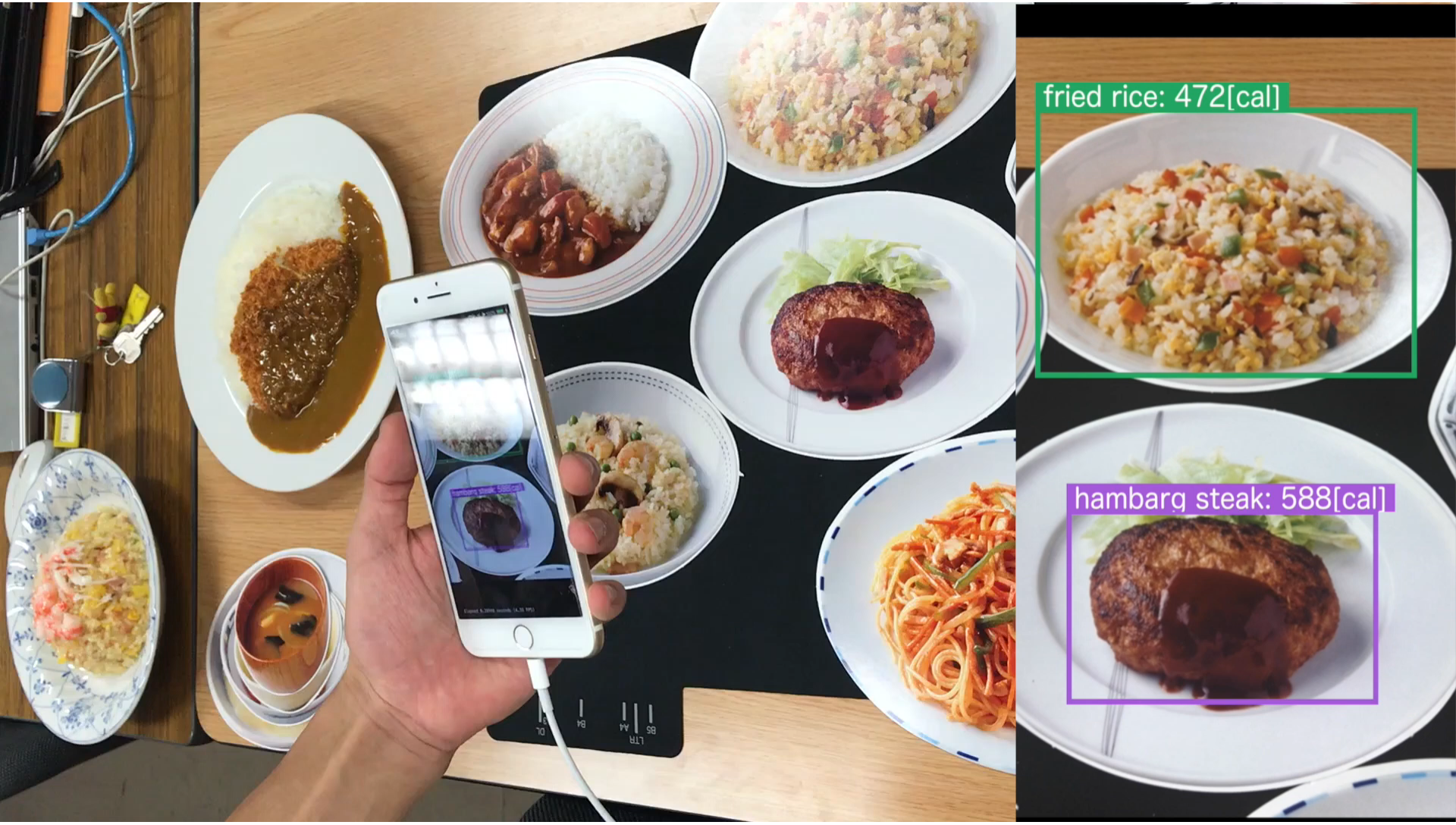


食事検出＋カロリー推定(処理フロー)

1. YOLOv2で食事検出
2. 検出された各食事領域のバウンディングボックス情報から画像をクロップ
3. クロップした各食事画像をカロリー値を推定するCNNの入力とする



食事検出＋カロリー推定(デモ動画)



MADiMa 2017

Tanno, R., Ege, T. and Yanai, K.: DeepCalorieCam: An iOS App for Dish Detection and Calorie Estimation, Proc. of International Workshop on Multimedia Assisted Dietary Management (MADIMA) (2017)

良いとこ悪いとこ

- **良いところ**
  - ネットワーク構造などは既存のフレームワークベースなので，一々ネイティブ側でレイヤの実装などは考えなくて良いので，学習ハードルが低い
  - GPU利用なのでネットワークを色々同時に使える
  - GPUノートブックは不要
    - デモをしようと思ったら，最新のiPhoneさえあればOK
  - 研究成果は直ぐにiOSに実装してデモ！

- **悪いところ**
  - CoreML側がどう処理してるか細かい部分まで把握できない
  - マルチスケール入力が不可能(FCNの場合でも入力画像サイズは固定)
    - CoreMLの現在の仕様？

まとめと今後の課題

- 昨今話題のCore ML(GPU)を弊研究室のCNNライブラリ(CPU)と比較してパフォーマンスを計測した
  - 認識などの処理が軽いタスクや比較的，低い解像度(224x224など)の場合はCPUの方が若干早い結果に
    - GPUのオーバーヘッドが大きい
  - しかし，高負荷の処理や高い解像度(640x480など)の場合は，GPUを使った実行の方が圧倒的に早い結果に

- Core MLを使った所感…
  - 各レイヤの実装やCNN高速化はAppleにお任せなので，モバイルへのdeployが非常に便利に
    - ただ，Core ML側がサポートしてないレイヤは使えないので，超最新のネットワークなどは未対応

- 今後の課題
  - バッテリ消費量のGPU vs CPUについても今後分析を行う

結論

PCで動作するものは
iPhoneでも十分に動きます...

Core MLを利用して研究成果を
モバイルに直ぐdeploy！

カバーするレイヤ例

Layers

List of Torch7 layers that can be converted into their CoreML equivalent:

1. Sequential
2. ConcatTable
3. SpatialConvolution
4. ELU
5. ReLU
6. SpatialBatchNormalization
7. Identity
8. CAddTable
9. SpatialFullConvolution
10. SpatialSoftMax
11. SpatialMaxPooling
12. SpatialAveragePooling
13. View
14. Linear
15. Tanh
16. MulConstant
17. SpatialZeroPadding

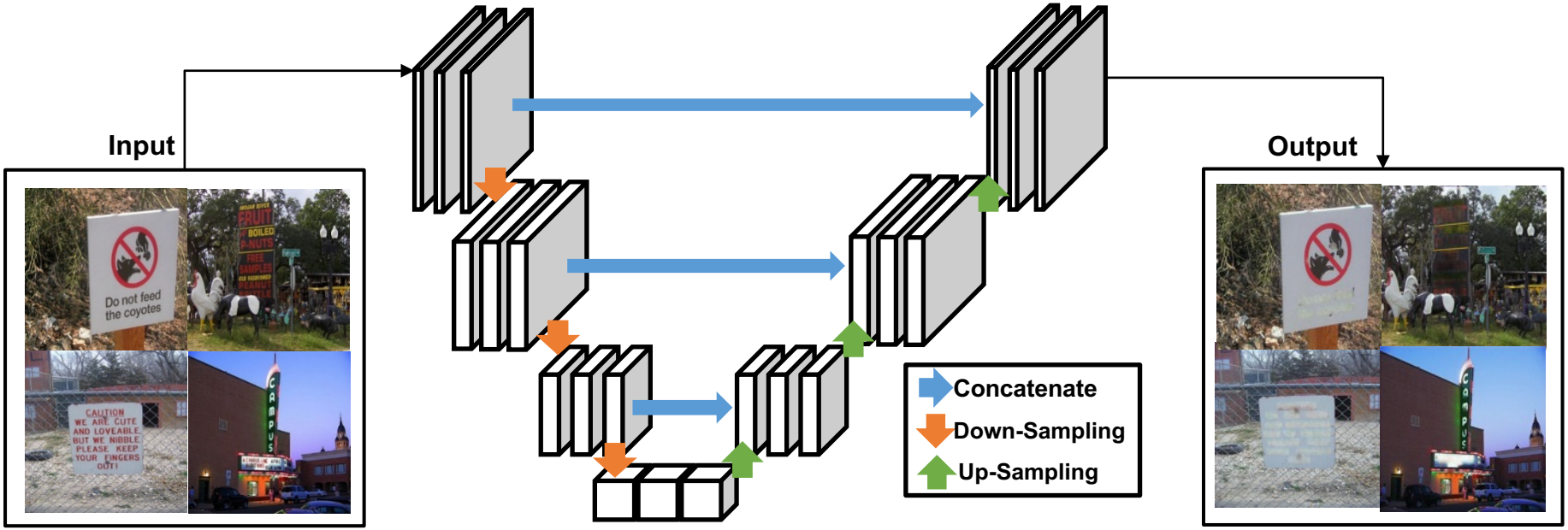
予備動画２種類

文字隠蔽ネットワーク

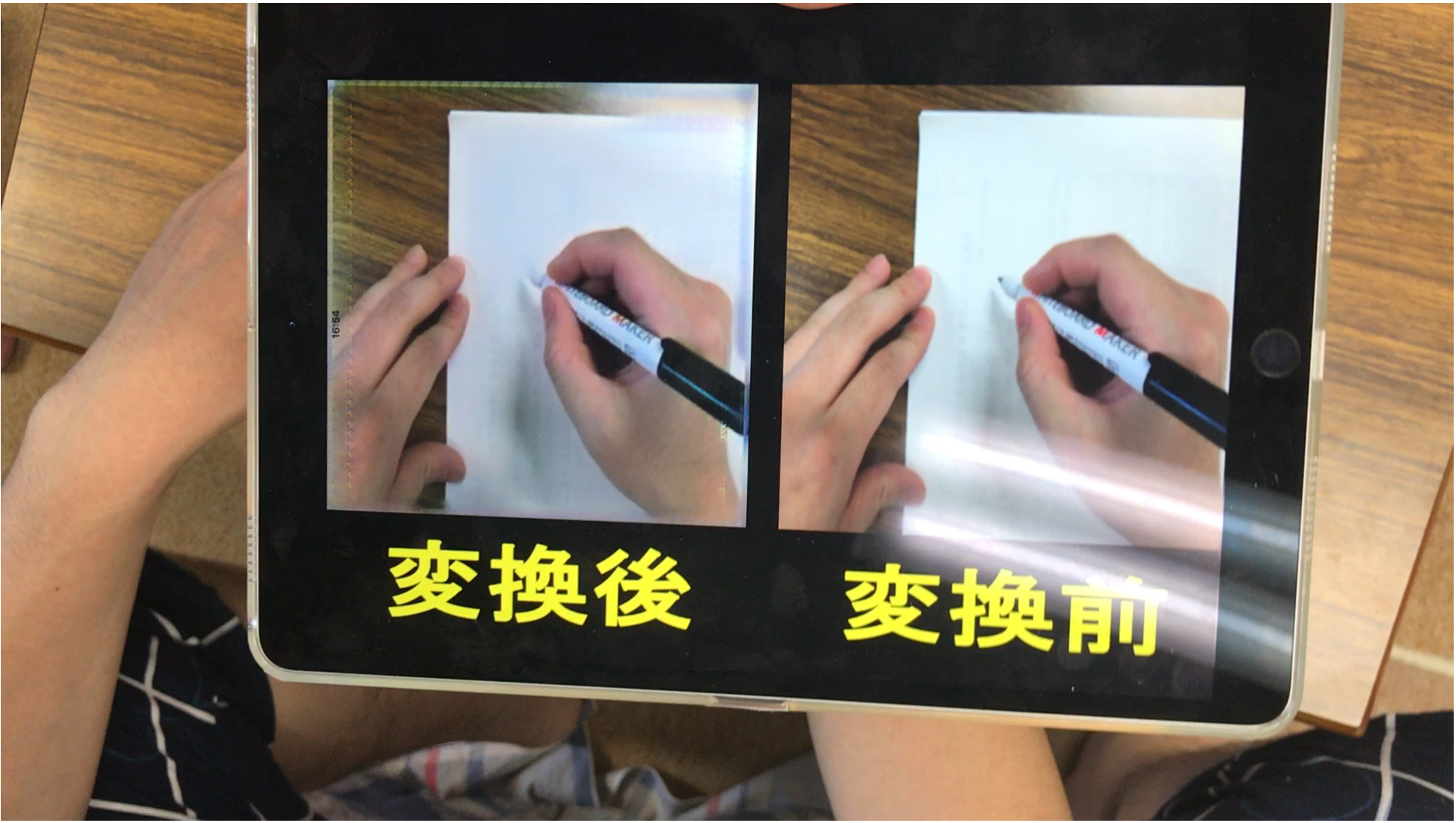
- 情景画像中の文字を隠蔽するネットワーク
  - Input: 文字有 -> Output: 文字無

[Nakamura et al, ICDAR, 2017]

- U-Netを利用
  - 画像サイズが同じものを深い層から段階的に統合
  - 局所的特徴を保持したまま，全体的位置情報の復元が可能

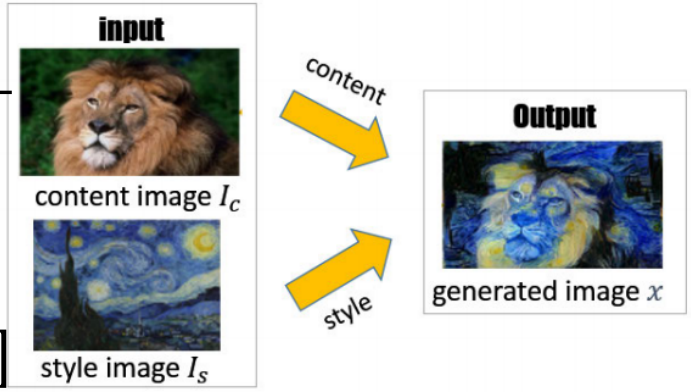


文字隠蔽ネットワーク(デモ動画)



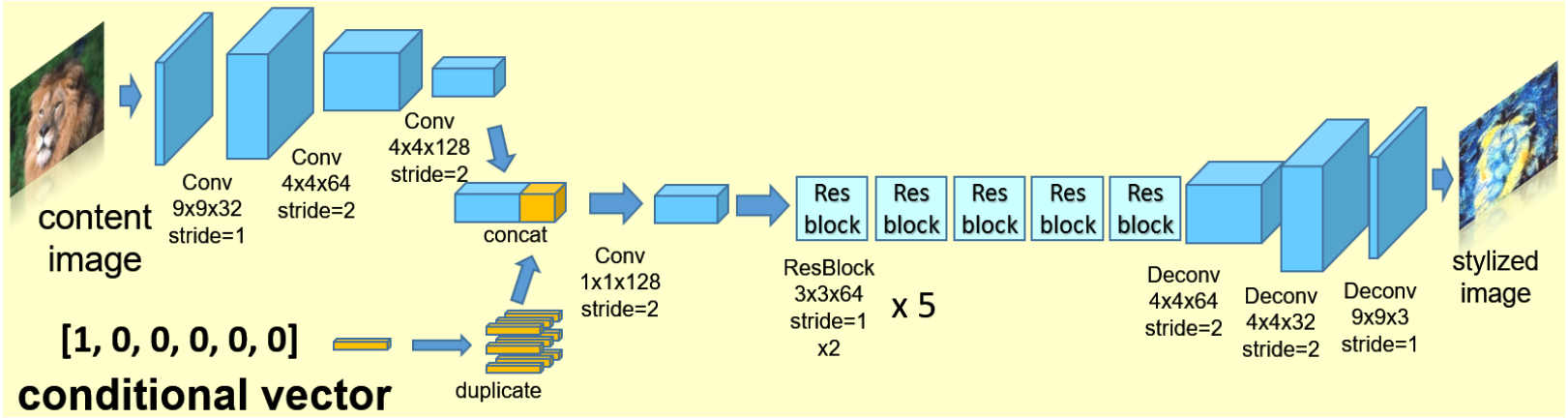
マルチスタイル変換

- Neural Style Transfer[Gatys et al, 2015]
  - 画像の形状を保持したまま画風を変換
  - 順伝搬及び誤差逆伝搬を複数回繰り返す
  - GPUで画像の生成に数十秒程度



- Fast Style Transfer[ECCV 2016][1]
  - 特定の画風変換を順伝搬で行うCNNを学習
  - 非常に高速に画像を変換可能
  - 1つのモデルで単一の画風変換のみ学習
  - 消費メモリの増大，学習に時間要

・ [1]の拡張
・ 単一モデルで複数のスタイルを任意重みで合成



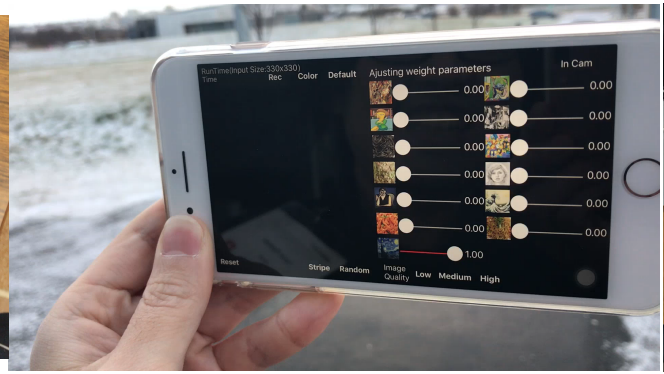
マルチスタイル変換(デモ動画)

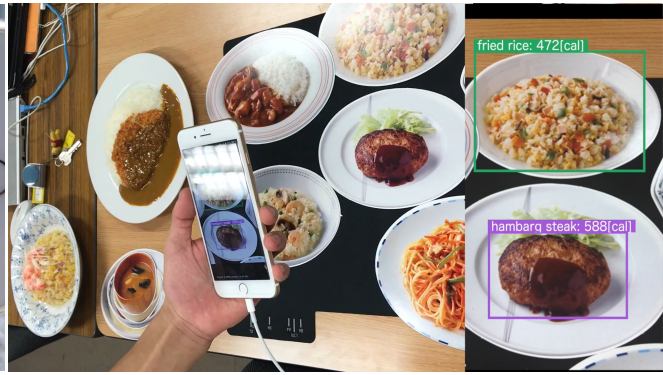
Tanno, R. and Yanai, K.: DeepStyleCam: A Real-time Style Transfer App on iOS, Proc. of International MultiMedia Modeling Conference (MMM) (2017)

予備，補足

まとめ

- 深層学習ネットワークをモバイルに実装
  - 技術的知見や応用例を検討
  - CoreMLとオリジナル順伝搬ライブラリとの比較
  - 実行速度ベンチマーク
  - 実装例3種類の紹介

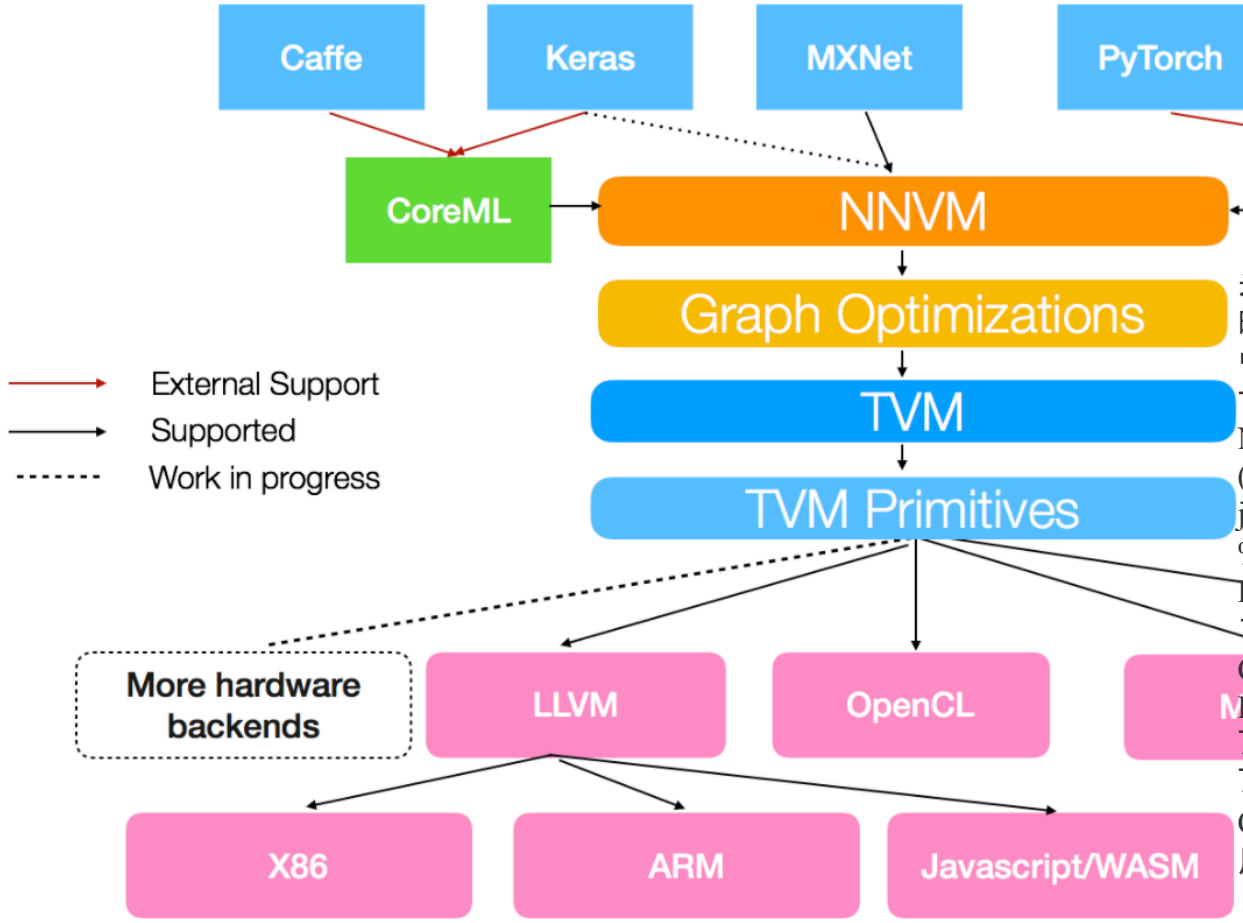




- 今後の課題
  - 深層学習＋AR(拡張現実)を融合したカロリー推定アプリの実現

The gap between framework and Hardware backends

NNVM Compiler: Open Compiler for AI Frameworks



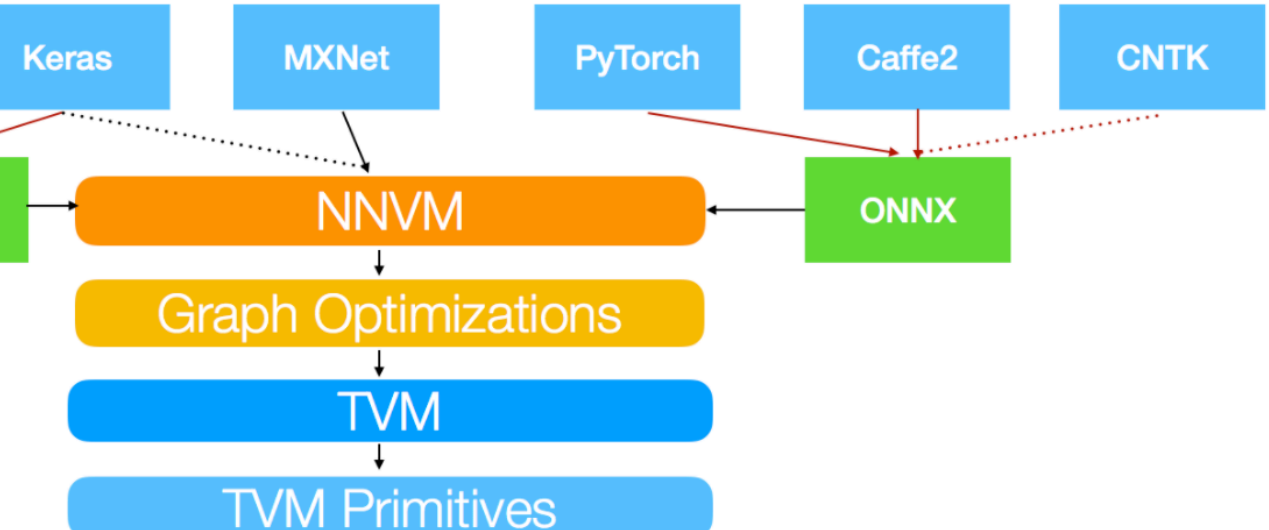
米Microsoftと米Facebookは9月7日（現地時間）、AI（人工知能）開発者がフレームワーク間のスイッチを簡単にできるようにするオープンソースプロジェクト「Open Neural Network Exchange（ONNX）」(https://www.msn.com/ja-jp/news/techandscience/microsoft%E3%81%A8facebook%E3%80%81ai%E9%96%8B%E7%99%BA%E8%80%85%E5%90%91%E3%81%91%E3%82%AA%E3%83%BC%E3%83%97%E3%83%B3%E3%82%BD%E3%83%BC%E3%82%B9%E3%83%97%E3%83%AD%E3%82%B8%E3%82%A7%E3%82%AF%E3%83%88%E3%80%8Connx%E3%80%8D/ar-AArtKDy から引用)

[NNVM Compiler: Open Compiler for AI Frameworks, 2017/10/6](http://www.tvmlang.org/2017/10/06/nnvm-compiler-announcement.html)から引用)

ONNX: Open Neural Network Exchange

- あるAIフレームワークで構築した学習モデルを異なる機械学習システムに簡単に切り替えることを目的

- 異なるフレームワーク間の相互運用性を可能にして研究から製造へのパスを流れさせることは AI コミュニティにおける革新スピードを早める



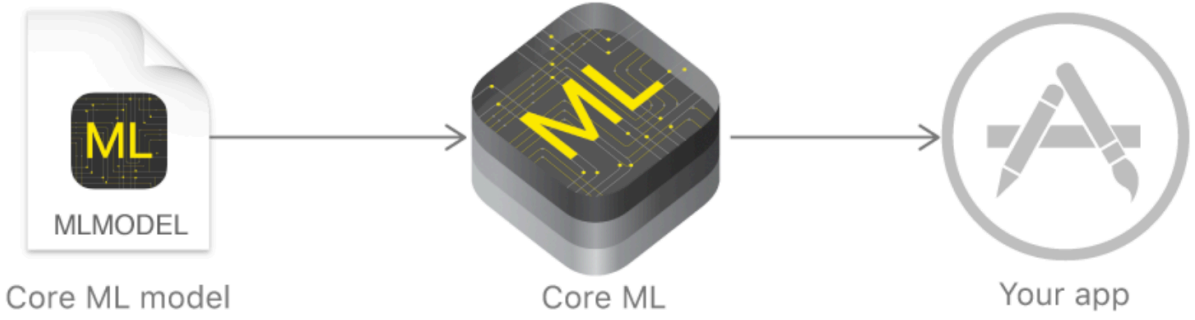
AR+深層学習の裏付け

- http://araruaru.hatenadiary.com/entry/2017/09/22/124536
- マーカーARとマーカーレスAR
  - SLAM(Simultaneous Localization and Mapping)技術がiphoneに搭載
  - 高精度なマーカーレスARの実現
- 実装例
  - 音の空間記録(https://wired.jp/2017/09/28/artist-uses-an-iphone/)

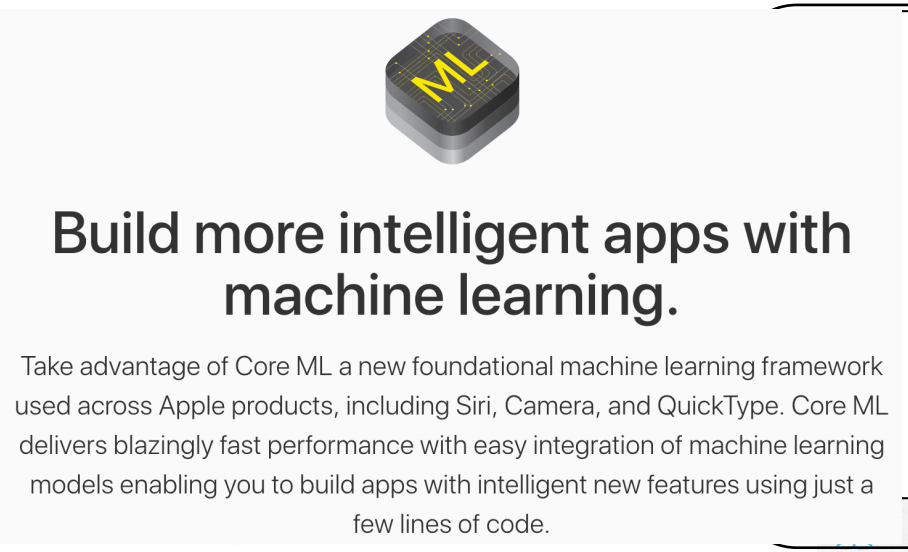
Core MLはご存知ですか？

Build more intelligent apps with machine learning.

Take advantage of Core ML a new foundational machine learning framework used across Apple products, including Siri, Camera, and QuickType. Core ML delivers blazingly fast performance with easy integration of machine learning models enabling you to build apps with intelligent new features using just a few lines of code.



Core MLはご存知ですか？



Core ML