DS-4 Chainer2MPSGraph:高速深層学習モバイルアプリ作成のためのモデルコンバータ

泉 裕貴，堀田 大地，丹野 良介，柳井 啓司(電気通信大学)

1.デモの概要

- モバイルGPU利用のためのコンバータChainer2MPSGraphの提案
- 提案コンバータによるデモアプリ3種 (認識1つ、画像変換2つ)
  - VGG16[1]
  - マルチスタイル変換 (MultiStyleFastStyleTransfer[2])
    (昨年のMIRUで発表済み)
  - マルチ食事画像変換 (conditional CycleGAN+ Twitter Food Images)
    (PS2-24で発表)

2. Chainer2MPSGraph

- Chainerの学習済みモデルとネットワークファイルから
  パラメータファイルとSwiftのネットワークファイルを自動生成
  - 自動生成による実装の容易化
  - GPU利用による演算の高速化
  - グラフ構造を用いることによる更なる高速化

CoreMLより高速実装

方法

- Chainerのモデルで1度順伝搬を行う
- ネットワークの各ノードがChainで繋がる
- モデルの出力からネットワークを遡る
- 遡りながらパラメータ等の情報を得る
- それらの情報からファイルを生成

MPSNNGraph

MPSNNGprahは、iOSでGPUを使うためのMetal Performance Shadersのうちのグラフ構造を利用するためのAPI

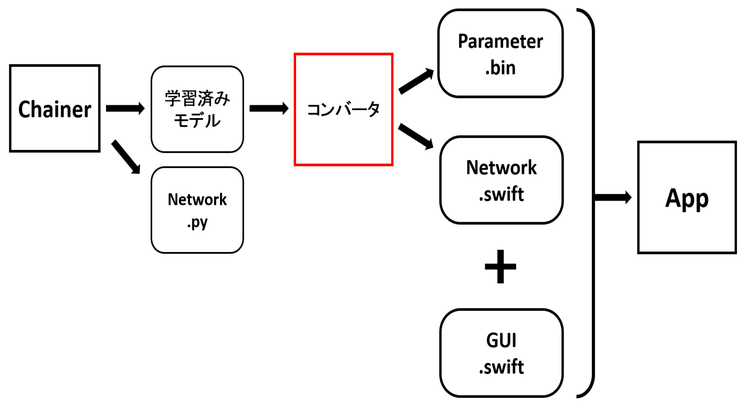
ネットワークコード生成例

```
let conv1 = MPSCNNConvolutionNode(source: conv01.resultImage,
                                  weights: DataSource("conv1", 9, 9, 3, 32, 1, useBias: true))
let relu1 = MPSCNNNeuronReLUNode(source: conv1.resultImage)
let bn1 = MPSCNNBatchNormalizationNode(source: relu1.resultImage,
                                       dataSource: DataSource2("bn1", 32))
let conv2 = MPSCNNConvolutionNode(source: bn1.resultImage,
                                  weights: DataSource("conv2", 4, 4, 32, 64, 2, useBias: true))
let relu2 = MPSCNNNeuronReLUNode(source: conv2.resultImage)
let bn2 = MPSCNNBatchNormalizationNode(source: relu2.resultImage,
                                       dataSource: DataSource2("bn2", 64))
let conv3 = MPSCNNConvolutionNode(source: bn2.resultImage,
                                  weights: DataSource("conv3", 4, 4, 64, 128, 2, useBias: true))
let relu3 = MPSCNNNeuronReLUNode(source: conv3.resultImage)
let bn3 = MPSCNNBatchNormalizationNode(source: relu3.resultImage,
                                       dataSource: DataSource2("bn3", 128))
let concat1 = MPSNNConcatenationNode(sources: [bn3.resultImage, styleImage1,
                                               styleImage2, styleImage3, styleImage4])
let conv4 = MPSCNNConvolutionNode(source: concat1.resultImage,
                                  weights: DataSource("conv4", 1, 1, 141, 128, 1, useBias: true))
let relu4 = MPSCNNNeuronReLUNode(source: conv4.resultImage)
let bn4 = MPSCNNBatchNormalizationNode(source: relu4.resultImage,
                                       dataSource: DataSource2("bn4", 128))
```

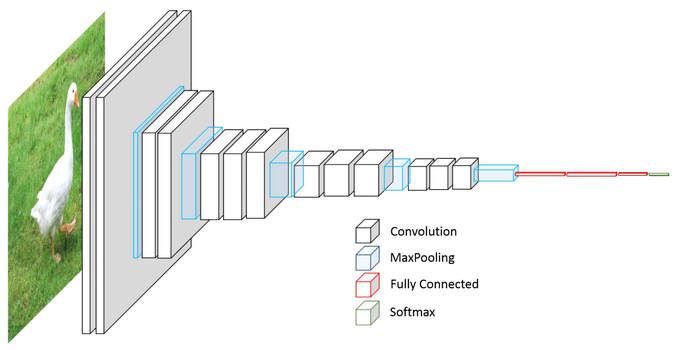
3. モバイル実装ワークフロー

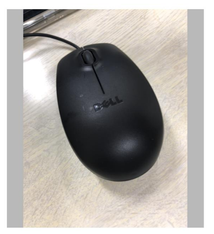
- Chainerでモデルを学習
- 学習済みモデルとネットワークファイルをコンバータに入力
- コンバータがファイルを生成
  - パラメータファイル
  - Swiftのネットワークファイル
- GUIコードの作成
- 生成したファイルと
  GUIコードを組み合わせる

アプリケーション完成!



4. VGG16[1]





1: mouse,computermouse (98.97%)
2: CDplayer (0.25%)
3: lenscap,lenscover (0.14%)
4: modem (0.11%)
5: joystick (0.10%)

Elapsed: 110.21 msec

1: cup (72.31%)
2: coffeemug (17.58%)
3: espresso (3.66%)
4: teapot (1.77%)
5: measuringcup (0.90%)

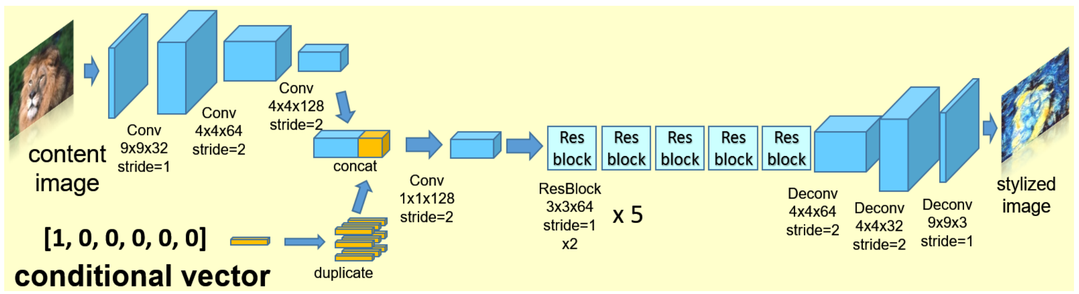
Elapsed: 108.05 msec

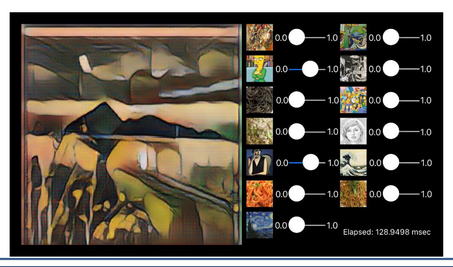
実装方法	平均認識時間	最速認識時間	最遅認識時間
MPS	155.21[ms]	146.72[ms]	170.00[ms]
CoreML	144.87[ms]	134.47[ms]	151.00[ms]
Chainer2MPSGraph	108.99[ms]	107.67[ms]	110.68[ms]

5. Multi Style Transfer[2]

FastStyleTransfer[4]
- 非常に高速に画風変換が可能
- 単一の画風変換のみ学習

→ 1つのモデルで複数スタイルの画風変換を可能にした





実装方法	認識時間
Chainer2C[3] (CPU)	138.55[ms]
Chainer2MPSGraph (GPU)	96.26[ms]
CoreML(FastStyleTransfer)[参考値]	101[ms]

6. conditional CycleGAN (PS2-24で発表中!!)

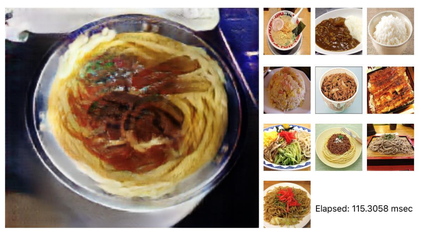
識別器(D)の損失関数

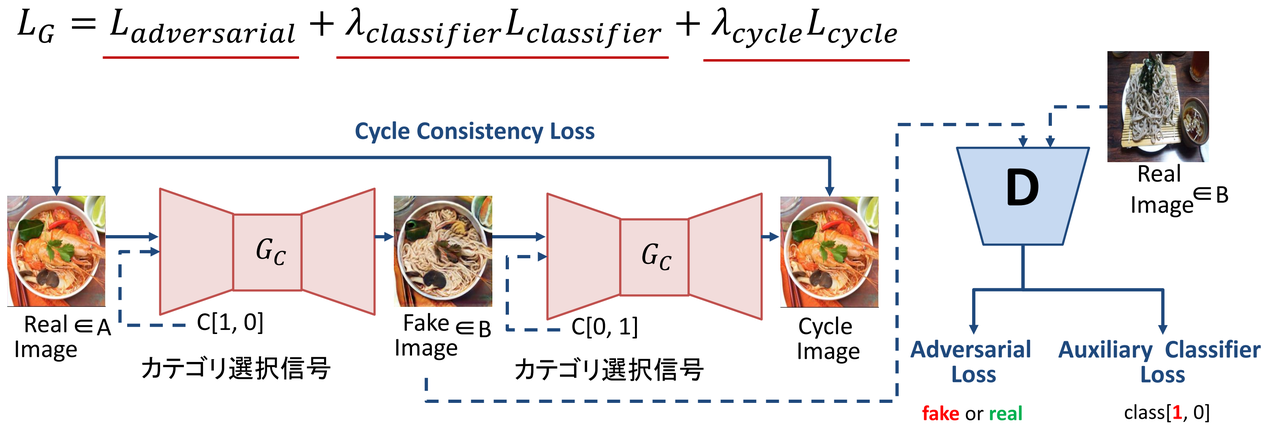
$$L_D = L_{adversarial} + \lambda_{classifier} L_{classifier}$$

変換器(G)の損失関数

損失重み: $\lambda_{classifier}$, λ_{cycle}

$$L_G = L_{adversarial} + \lambda_{classifier} L_{classifier} + \lambda_{cycle} L_{cycle}$$





参考文献

[1] K. Simonyan and A. Zisserman ： Very deep convolutional networks for largescale image recognition, ICLR, 2015.
[2] K. Yanai and R. Tanno ： Conditional Fast Style Transfer Network, ICMR, 2017.
[3] R. Tanno and K. Yanai. ： Caffe2C: A Framework for Easy Implementation of CNN-based Mobile Applications, MUSICAL, 2016
[4] J. Johnson, A. Alahi, L. Fei-Fei ： Perceptual Losses for Real-Time Style Transfer and Super-Resolution, ECCV, 2016.