ICLR2018読み会@PFN

ICLR2018におけるモデル軽量化

May 26, 2018

Tomohiro Kato
AI System Dept.
DeNA Co., Ltd.

本資料について

- 本資料は ICLR2018読み会@PFN (2018/05/26) の発表資料です
  - https://connpass.com/event/88077/
- Deep Learningのモデル軽量化手法について、
  ICLR2018に投稿された論文を紹介しています。

Agenda

- Deep Learningのモデル軽量化
- ICLR2018におけるモデル軽量化
  - Quantization
  - Pruning
  - Efficient Architecture
  - Architecture Search
  - Platform
- まとめ

Agenda

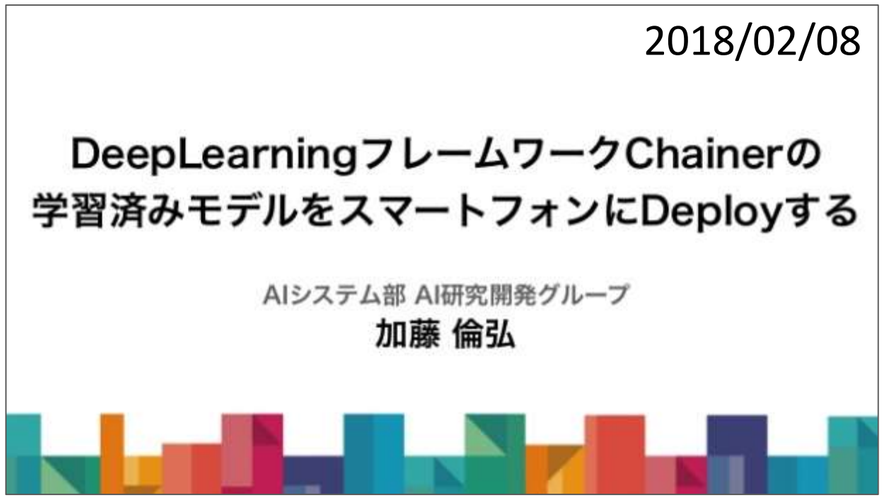
- Deep Learningのモデル軽量化
- ICLR2018におけるモデル軽量化
  - Quantization
  - Pruning
  - Efficient Architecture
  - Architecture Search
  - Platform
- まとめ

当日の発表はYouTubeにアップロードされています
https://youtu.be/5wW6OFK_Y94?t=8m11s

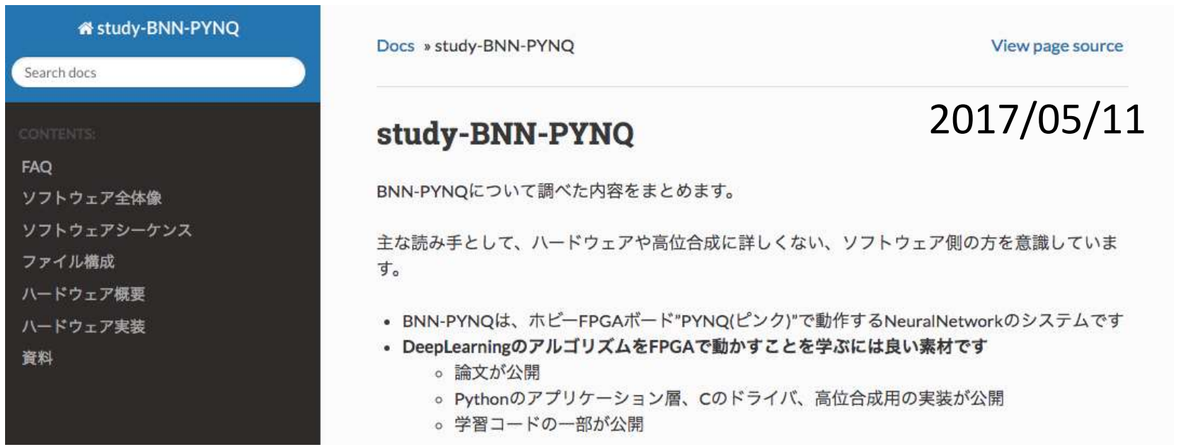
自己紹介

- 加藤 倫弘 (かとう ともひろ)
- ㈱ DeNA システム本部 AIシステム部
- Computer Vision関連のプロジェクトを担当
- 興味は、低レイヤでのDeep Learning

Twitter @_tkato_



Chainer x NNVM/TVMの話
https://www.slideshare.net/tkatojp/deeplearningchainerdeploy-87479889



ホビーFPGAでバイナリNNを動かした解説
https://tkat0.github.io/study-BNN-PYNQ/

Deep Learningのモデル軽量化の目的

- モバイルやエッジデバイスで動かしたい
- CPUや省メモリでも動かしたい
- 学習を高速化したい
- GPUが使えない、使いたくない
- 消費電力を下げたい
- 理由は色々
  - プライバシー、レイテンシ、クラウド費用、実験効率化 ...

モデル軽量化の課題

- 精度を落とさずに性能（推論速度やメモリ使用量）を向上できるか
- 学習スキームやハイパーパラメータ探索を単純化・高速化できるか
- ToyタスクやTinyなモデルではうまくいくが...
- 簡単に実装できるか（ソフトウェア、ハードウェア）

Quantization （量子化）

- モデルのストレージ容量やメモリ容量の節約、専用命令で高速化
- 通常FP32のWeightやActivationをFP16, INT8, 4bit, 2bit, 1bit等で近似する

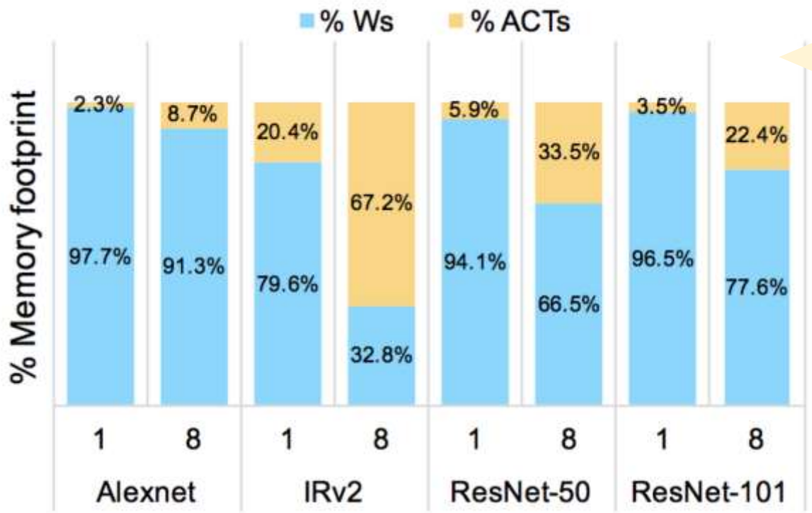


Figure 1: Memory footprint of activations (ACTs) and weights (W) during inference for mini-batch sizes 1 and 8.

特にバッチ数が小さい推論ほど、
Weightがメモリの大半を占めるので減らしたい

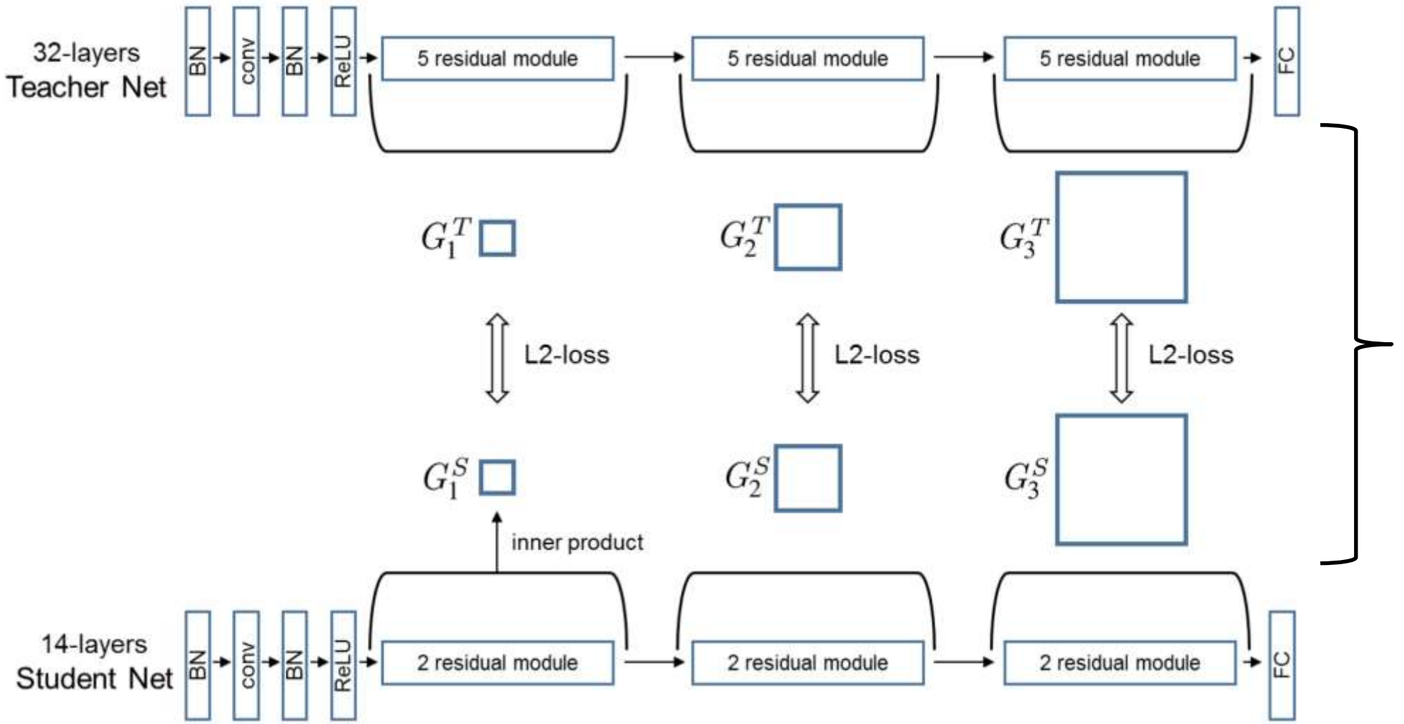
神資料

https://www.slideshare.net/kentotajiri/ss-77136469

Asit Mishra. 2017. Apprentice: Using Knowledge Distillation Techniques
To Improve Low-Precision Network Accuracy. arXiv:1711.05852. Retrieved from https://arxiv.org/abs/1711.05852

Distillation （蒸留）

- Smallなモデルは Largeなモデルより精度が低くなりやすい
- Smallなモデル(Student)の学習の補助に、Largeなモデル(Teacher)を使う
  - Student単体での学習より精度が高くなる



この例では、
Teacherに似たFeature mapを
出力させるように学習

http://openaccess.thecvf.com/content_cvpr_2017/papers/Yim_A_Gift_From_CVPR_2017_paper.pdf

Pruning （枝刈り）

- Weightの数を削減して、ストレージ容量やメモリ容量の節約、高速化
  - 値が0に近いWeightは削除できる

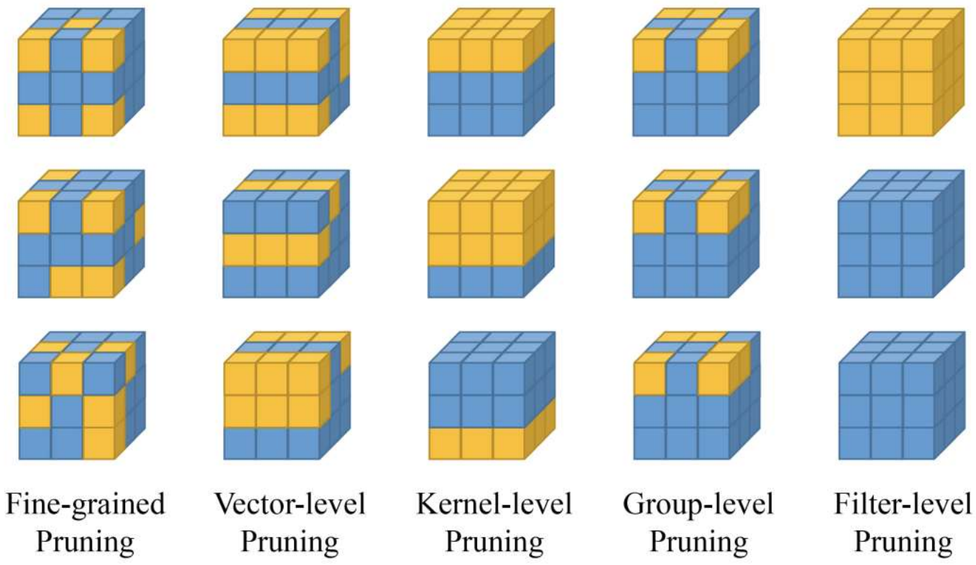


Figure 1. Different pruning methods for a convolutional layer which has 3 convolutional filters of size $3 \times 3 \times 3$.

行列のサイズを小さくできるケースでは
高速化が簡単（疎行列に比べて）

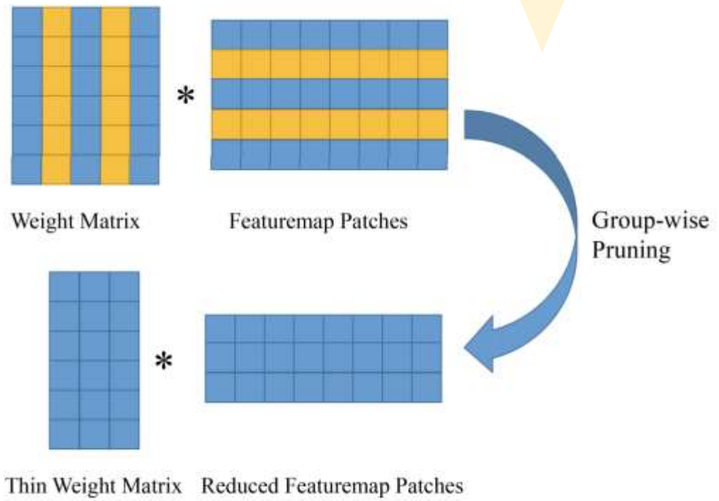


Figure 2. Group-level Pruning.

Jian Cheng. 2018. Recent Advances in Efficient Computation of Deep Convolutional Neural Networks. arXiv:1802.00939. Retrieved from https://arxiv.org/abs/1802.00939

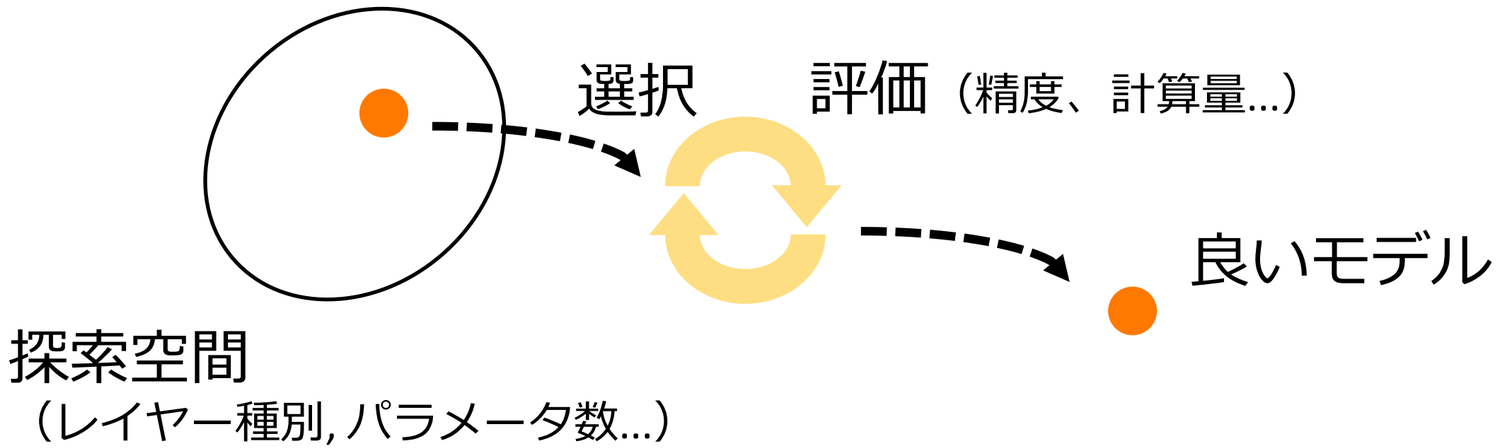
Efficient Architecture

- 軽量に設計したConvolution層など、アーキテクチャレベルでの最適化
  - Fire module (SqueezeNet)
  - Depthwise Separable Convolution (MobileNet)
- 神資料

https://www.slideshare.net/ren4yu/ss-84282514

Architecture Search

- Hand Designedなモデル設計で軽量化するのは難しい
- モデルのアーキテクチャ設計を自動化したい
  - Neural Architecture Search (NAS) *
- 強化学習(RL)ベース、進化的計算(GA)ベース、Progressive Search ...



* Barret Zoph. 2016. Neural Architecture Search with Reinforcement Learning. arXiv:1611.01578. Retrieved from https://arxiv.org/abs/1611.01578

Platform

- MobileNetをPruningしてさらにQuantizationした。完璧。
  - しかしGPUでおそくて とてもつらい
- DeepLearningフレームワークやハードウェア側は対応していますか？
  - 例えば、フレームワークによってdepthwiseはまだ遅かったり
    - slim.separable_conv2d is too slow · Issue #12132 · tensorflow/tensorflow
      - https://github.com/tensorflow/tensorflow/issues/12132
- 最近ではGraph最適化とかカーネル最適化が流行っていますね
  - Learning to Optimize Tensor Programs (AutoTVM) *

* Tianqi Chen. 2018. Learning to Optimize Tensor Programs. arXiv:1805.08166.
Retrieved from https://arxiv.org/abs/1805.08166

ICLR2018におけるモデル軽量化

- 各カテゴリで私が興味をもった論文を紹介していきます
- こんなものがあるよ、を広く知ってもらうことが目的です
  - 数式はほとんどでてきません
- 時間の関係でやむを得ず紹介できない論文も多数あります

※ 以降の論文紹介スライドでの図表は、断りがない場合は紹介論文からの引用です

QUANTIZATION

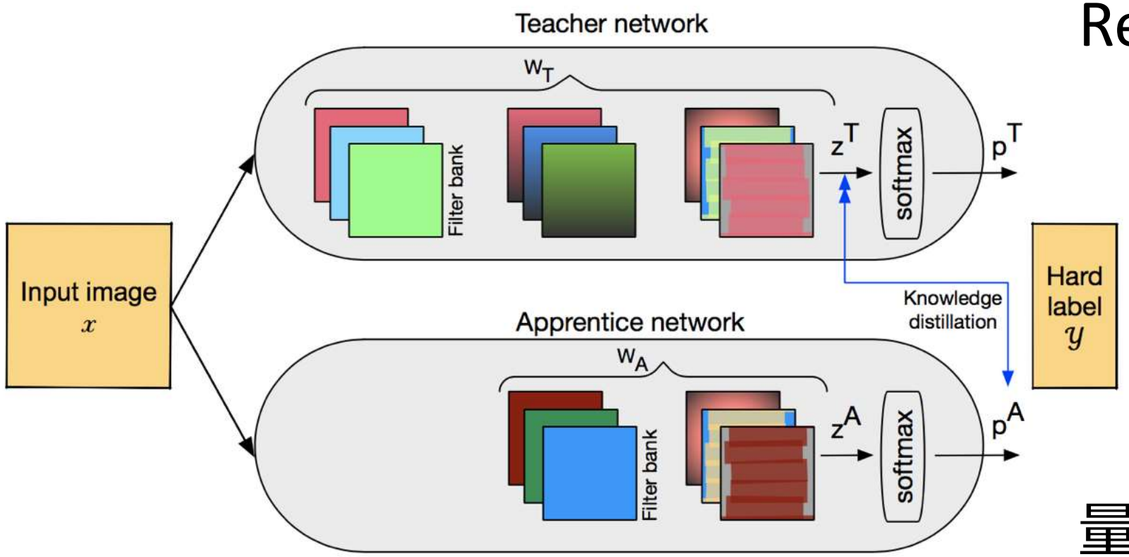
Quantization@ICLR2018: Summary

- Distillationを組み合わせて高精度化
- 学習時の量子化による高速化・省メモリ化
- Wide-ResNetをバイナリ化
- 微分可能な学習時の量子化 (soft quantization)

Poster

Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy

- Distillationで量子化したモデルの精度向上
- ternary(2bit), 4bitのweightのモデルではImageNetでSoTA



full-precision
ResNet-50,101 （学習済み）

Hard Target (教師データのLoss)と
Soft Target (TeacherのLogitsとのLoss)

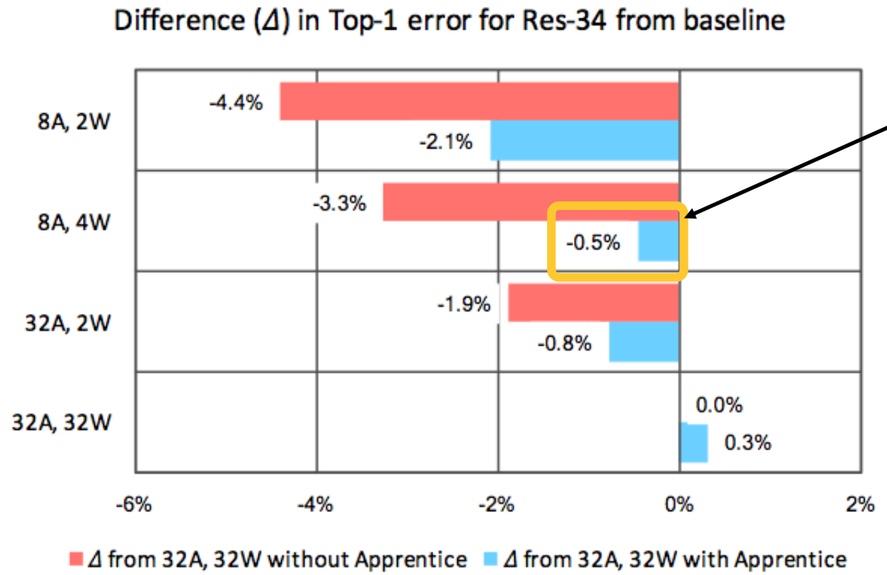
量子化したモデル（full-precisionで学習済み）

Asit Mishra. 2017. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy. arXiv:1711.05852. Retrieved from https://arxiv.org/abs/1711.05852

Poster

Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy

- Distillationによって、精度劣化を抑えられる
- 既存研究での精度より1.5～3 %良い精度



(a) Apprentice versus baseline accuracy for ResNet-34.

Figure 4: Distillationの有無による差（ImageNet-1k）

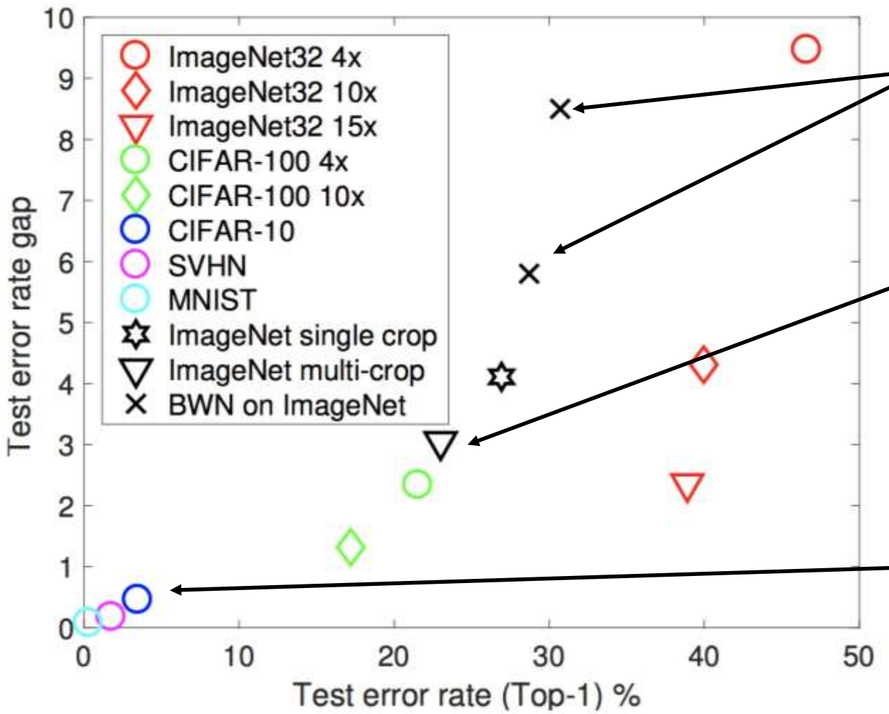
8bit activation, 4bit weightでも、
full-precision時から-0.5%の劣化

Studentより精度が良いTeacherを使うことが重要
モデルサイズは重要でない

Poster

Training wide residual networks for deployment using a single bit for each weight

- **1bit weight の wide-resnet**で、full-precision (FP) との精度差を低減



FPとの精度差

比較対象 BWN [Rastegari et al. 2016]

ImageNetでfull-precisionとの差 **3%**

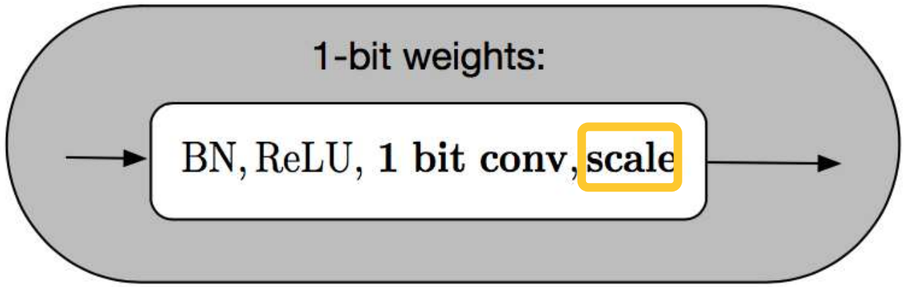
CIFAR10ならfull-precisionとほぼ同じ精度

Test error rate (Top1) %

Mark D. McDonnell. 2018. Training wide residual networks for deployment using a single bit for each weight. arXiv:1802.08530. Retrieved from https://arxiv.org/abs/1802.08530

Training wide residual networks for deployment using a single bit for each weight

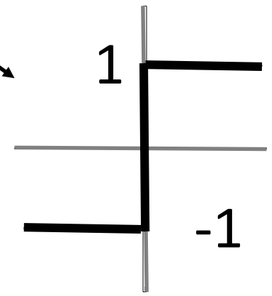
- 基本的にはBinary Weight Network(BWN)と同じ
- レイヤ毎のscale値を改善
  - full-precisionと同じ出力分布を得るため
  - Weightの初期化時に一度計算すれば済む



HeNormalでW_iを初期化し、
その際の標準偏差をscaleに使う

$$\hat{\mathbf{W}}_i = \sqrt{\frac{2}{F_i^2 C_{i-1}}} \operatorname{sign}(\mathbf{W}_i), \quad i = 1, \ldots, L,$$

FxF: カーネルサイズ
C: 入力チャネル数



Adaptive Quantization of Neural Networks

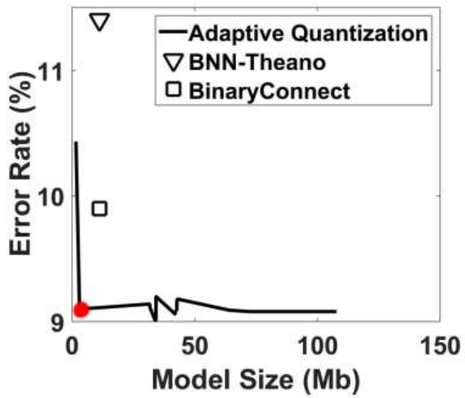
Poster

- Weight毎に異なるbit精度 （実装たいへんそう...）
- Weightのbit数をLossとして、精度の制約の元で最適化
- 学習しながら徐々に各Weightが最適なbit数に収束（0bitまで！）

量子化後のモデルのWeightのbit数の総和

$$\min_{W} N_Q(W) = \sum_{i=1}^{n} N_q(\omega_i) \qquad \mathcal{L}(W) \leq \bar{\ell}$$

量子化前の train loss を上限とする
（精度は下げたくないので）



(b) CIFAR-10

量子化後のbit数分布

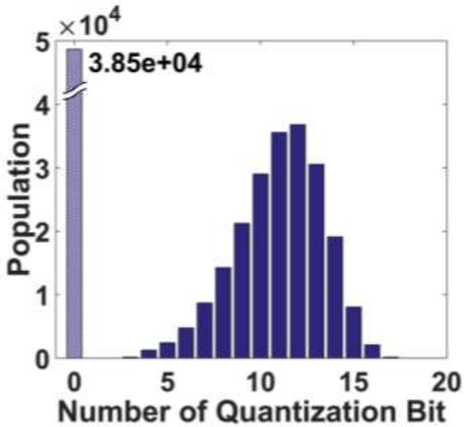


Figure 1(b)

精度が必要なWeightは16bit近い
0bit(=pruningできる)も多い

Soroosh Khoram. 2018. Adaptive Quantization of Neural Networks. ICLR2018.
Retrieved from https://openreview.net/forum?id=SyOK1Sg0W

Mixed Precision Training

Poster

- FP16で学習し、メモリ転送コスト削減
  - Volta世代のTensorCoreならFP32に対して2-6倍高速化も
- 学習時のハイパーパラメータはFP32の時と同じでOK

計算精度のため、UpdateはFP32

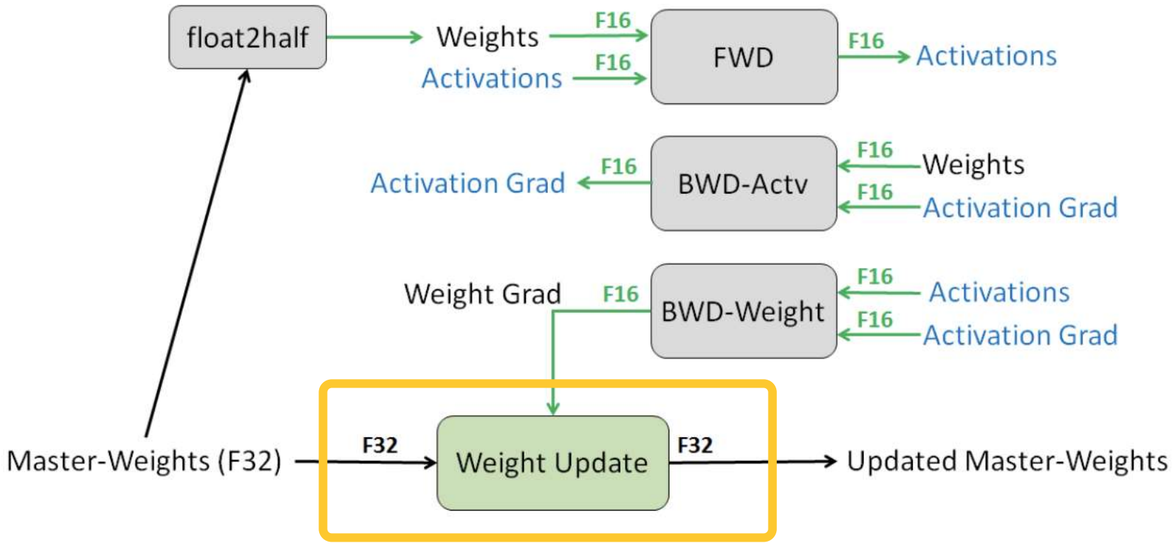
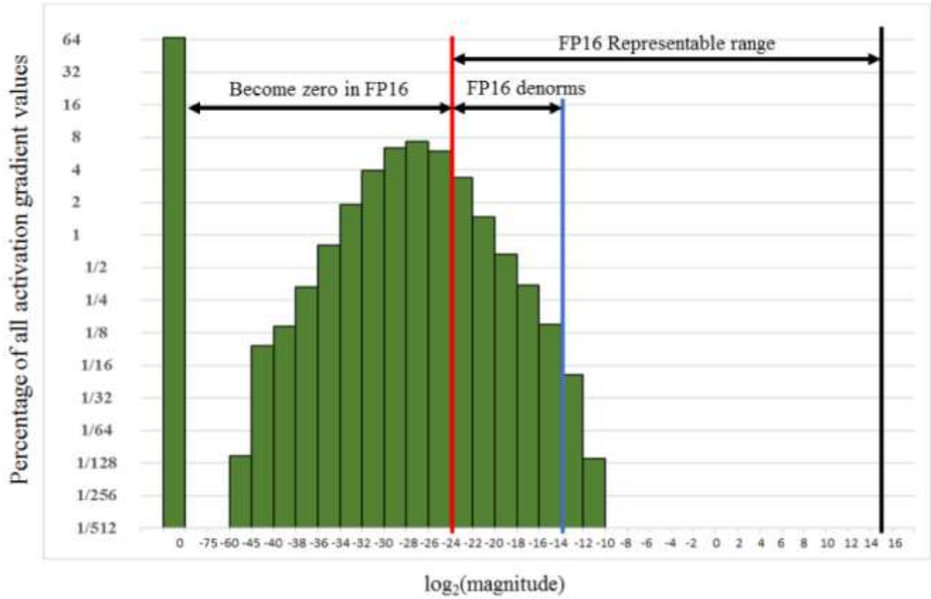


Figure 1: Mixed precision training iteration for a layer.

GradientがFP16で0にならないように
LossのScalingを行う



Paulius Micikevicius. 2017. Mixed Precision Training. arXiv:1710.03740.
Retrieved from https://arxiv.org/abs/1710.03740

Mixed Precision Training of Convolutional Neural Networks using Integer Operations

Poster

- Integer16/32でCPUで学習。
  - CPUでの学習がFP32に比べて1.8倍高速化
  - Integer型でDynamic Fixed Pointを表現し、行列演算
- 学習時のハイパーパラメータはFP32の時と同じで、同精度

Table 1: Training configuration and ImageNet-1K classification accuracy

Models	Batch-size / Epochs	Baseline		Mixed precision DFP16	
		Top-1	Top-5	Top-1	Top-5
ResNet-50	1024 / 90	75.70%	92.78%	75.77%	92.84%
GoogLeNet-v1	1024/ 80	69.26%	89.31%	69.34%	89.31%
VGG-16	256 / 60	68.23%	88.47%	68.12%	88.18%
AlexNet	1024 / 88	57.43%	80.65%	56.94%	80.06%

Dipankar Das. 2018. Mixed Precision Training of Convolutional Neural Networks using Integer Operations. arXiv:1802.00930. Retrieved from https://arxiv.org/abs/1802.00930

Training and Inference with Integers in Deep Neural Networks

Oral

- 専用チップでの学習/推論のため、backpropも8bit以下にする
- 低bitの学習に欠かせないBatchNormalizationも
  浮動小数点演算が不要になるように改良

backpropで必要な**Gradient**と**Error**のbit数も量子化している

Table 1: Test or validation error rates (%) in previous works and WAGE on multiple datasets. Opt denotes gradient descent optimizer and withM means SGD with momentum, BN represents for batch normalization and 32 bits refers to float32, ImageNet top-k format: top1/top5.

Method	k_W	k_A	k_G	k_E	Opt	BN	MNIST	SVHN	CIFAR10	ImageNet
BC	1	32	32	32	Adam	✓	1.29	2.30	9.90	-
BNN	1	1	32	32	Adam	✓	0.96	2.53	10.15	-
BWN[1]	1	32	32	32	withM	✓	-	-	-	43.2/20.6
XNOR	1	1	32	32	Adam	✓	-	-	-	55.8/30.8
TWN	2	32	32	32	withM	✓	0.65	-	7.44	34.7/13.8
TTQ	2	32	32	32	Adam	✓	-	-	6.44	42.5/20.3
DoReFa[2]	8	8	32	8	Adam	✓	-	2.30	-	47.0/ -
TernGrad[3]	32	32	2	32	Adam	✓	-	-	14.36	42.4/19.5
WAGE	2	8	8	8	SGD	✗	**0.40**	**1.92**	6.78	51.6/27.8

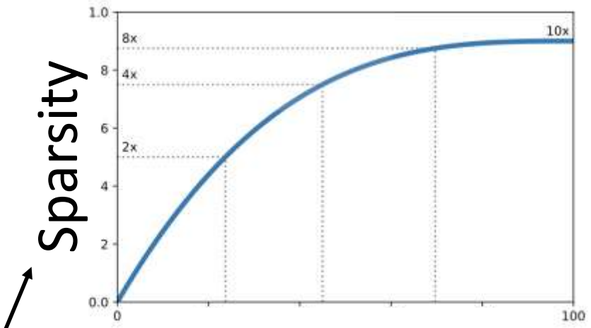
Shuang Wu. 2018. Training and Inference with Integers in Deep Neural Networks. arXiv:1802.04680. Retrieved from https://arxiv.org/abs/1802.04680

PRUNING

To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression

- 1-stageのpruning
  - 2-stage: pretrain → **pruning→ fine-tuning**
  - 1-stage: pretrain → **pruning & fine-tuning**
- 絶対値が小さいWeightから徐々にpruning ({1,0} のマスク行列を使う)

Figure 1 Gradual Pruning



Sparsity function

冗長なWeightが多い初期は一気に、
Weightが減ってきたら慎重にpruning

$$s_t = s_f + (s_i - s_f)\left(1 - \frac{t - t_0}{n\Delta t}\right)^3 \quad \text{for} \quad t \in \{t_0,\ t_0 + \Delta t,\ \ldots,\ t_0 + n\Delta t\}$$

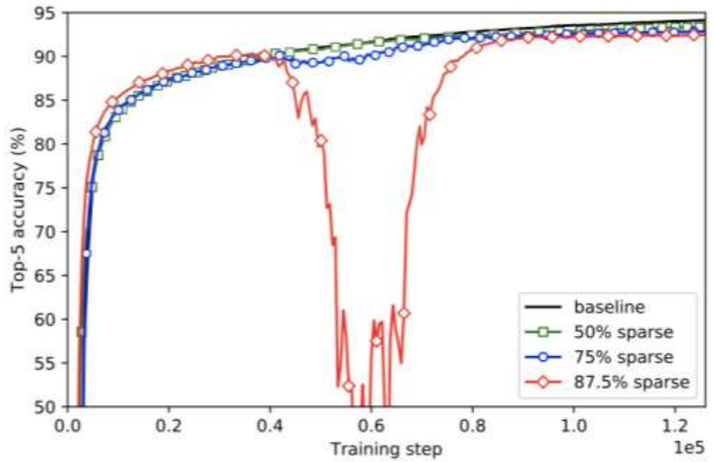
例: t=100 stepかけて徐々にsparsity=0.8までpruning

Michael Zhu. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv:1710.01878. Retrieved from https://arxiv.org/abs/1710.01878

To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression

- pruningにより精度低下しても、trainingで回復
  - Learning Rateも重要。 pruningと精度回復のバランスが重要。
- パラメータ数が同じモデルでも、大きいモデルで学習→pruningが精度良い
  - あわせて読みたい: The Lottery Ticket Hypothesis *

Weightを87.5%除去しても
ImageNet Top-5で最終的に2%の精度劣化



同じパラメータ数でも約10%精度が異なる

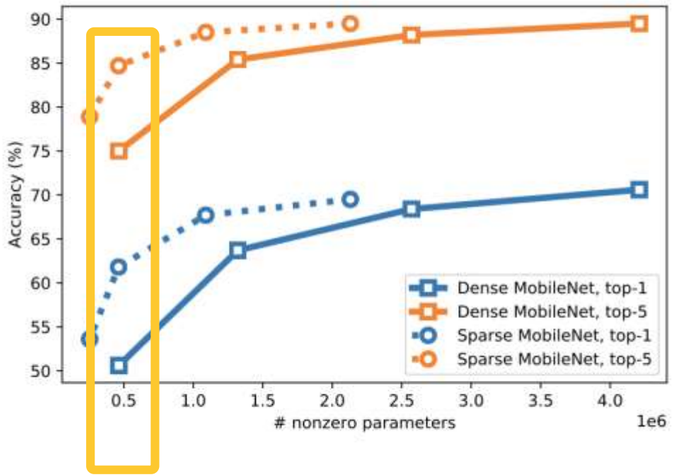


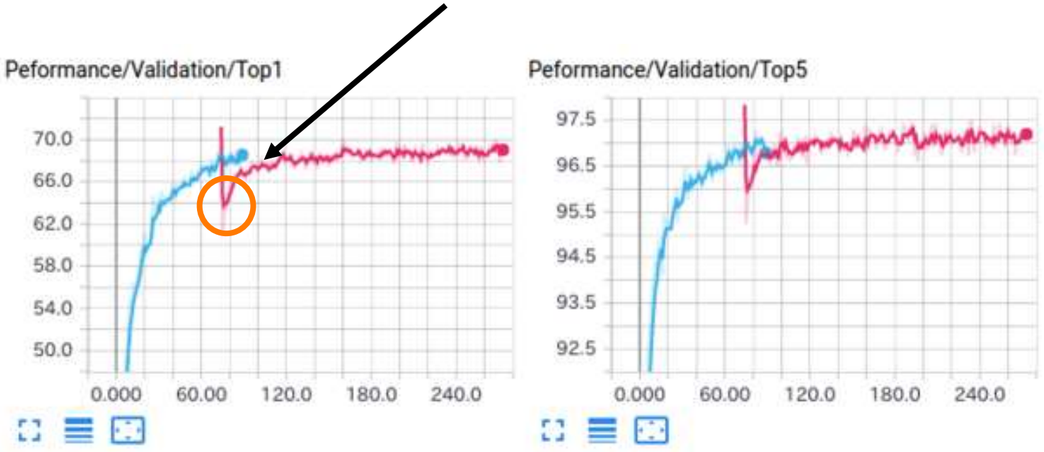
Figure 3: MobileNet sparse vs dense results

* Jonathan Frankle. 2018. The Lottery Ticket Hypothesis: Finding Small, Trainable Neural Networks. arXiv:1803.03635. Retrieved from https://arxiv.org/abs/1803.03635

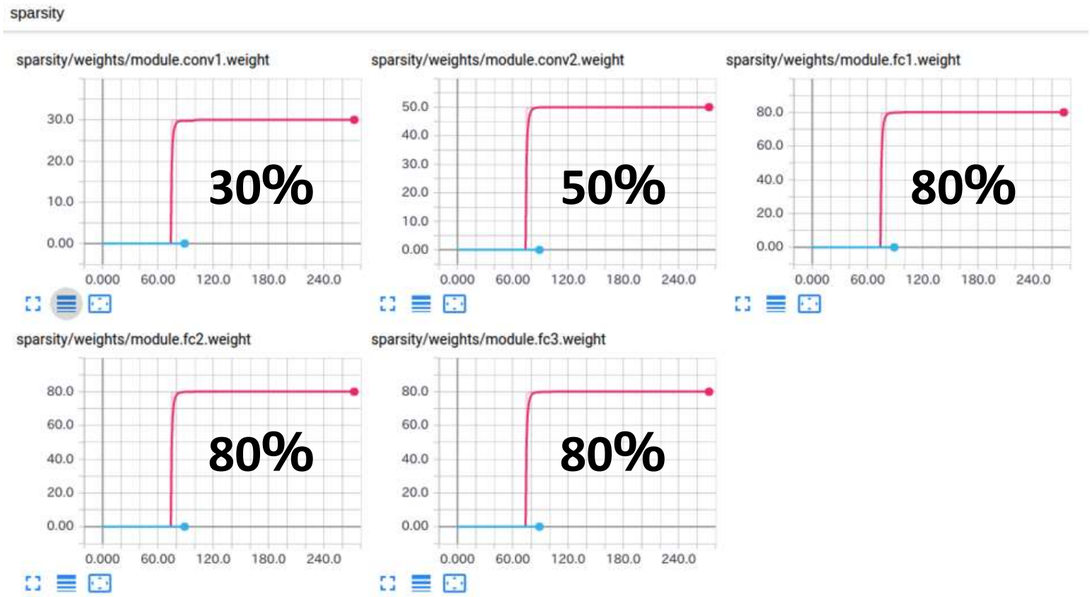
参考: Neural Network DistillerでGradual Pruningを試してみた

- DistillerでDeepLearningのモデルを軽量化: Gradual Pruning編 - tkato's blog
  - http://tkat0.hateblo.jp/entry/2018/05/22/082911
- Pytorch向けの軽量化ライブラリDistiller。お手軽でよいです。

pretrain（青）から、pruning開始後（赤）に
一度精度が落ちるが、その後の学習で回復



レイヤー毎に異なるSparsityを指定して、
100 epochかけてpruning&re-train



EFFICIENT ARCHITECTURE

Efficient Architecture@ICLR2018: Summary

- Early Termination （早期終了）
  - 入力の難易度や利用できる計算リソースに応じて、モデルの計算量を変える

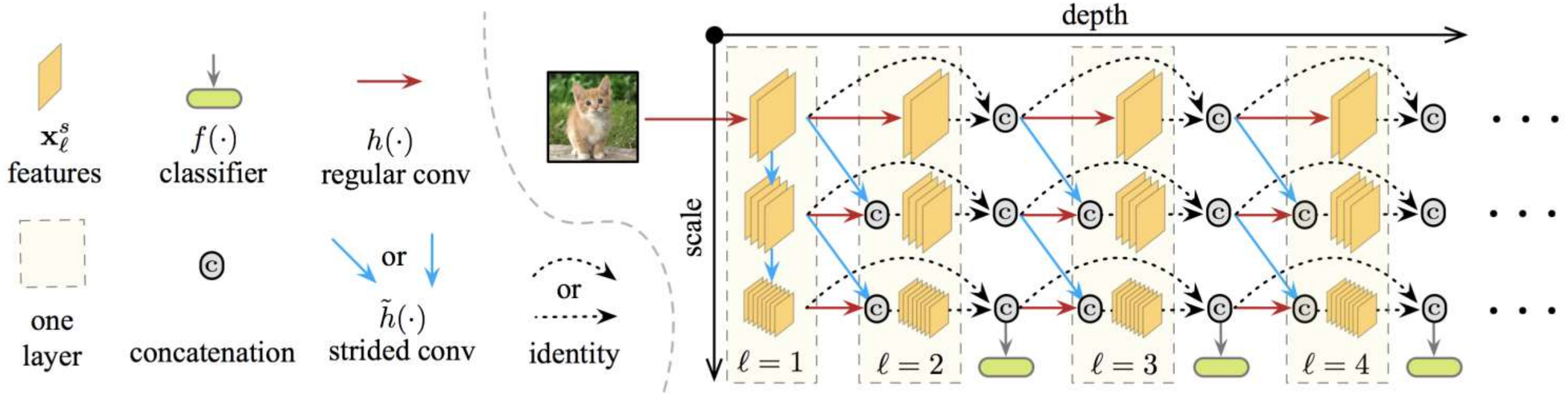
両方 “horse”

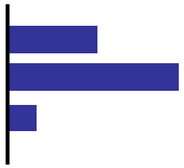
Gao Huang. 2017. Multi-Scale Dense Networks for Resource Efficient Image Classification. arXiv:1703.09844. Retrieved from https://arxiv.org/abs/1703.09844

Oral

Multi-Scale Dense Networks for Resource Efficient Image Classification

- 1モデル学習 → リソースの異なるデバイスにデプロイしたい
  - デバイスのリソースが許す限り、深く実行 (Anytime prediction)
  - 信頼度が閾値を超えるまで、深く実行 (Budgeted batch classification)





← 信頼度 = Softmax出力の最大値

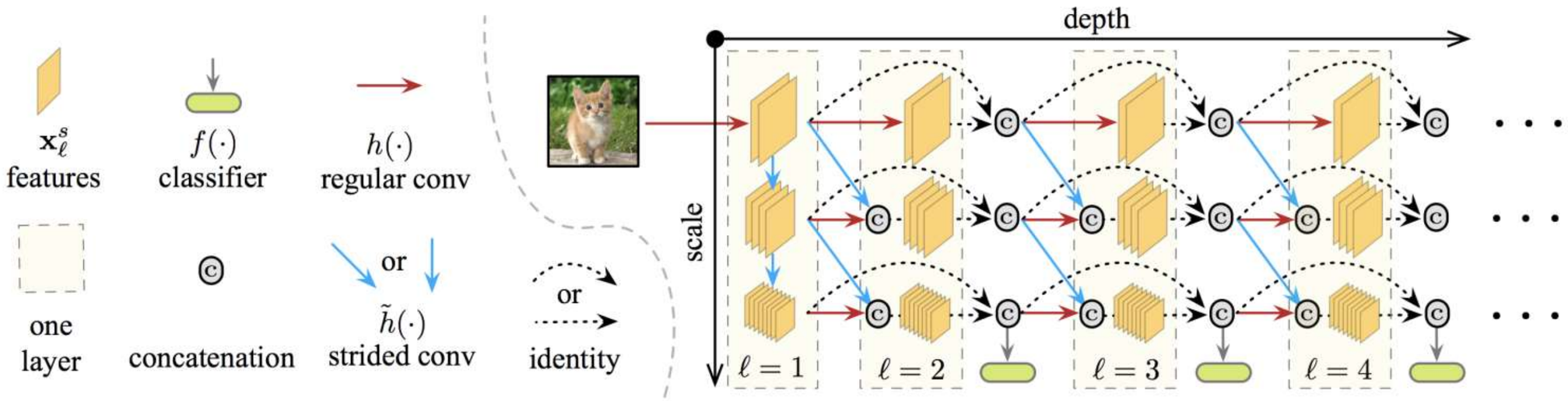
どこでも出力できるが、深いほど精度は良い

Gao Huang. 2017. Multi-Scale Dense Networks for Resource Efficient Image Classification. arXiv:1703.09844. Retrieved from https://arxiv.org/abs/1703.09844

Oral

Multi-Scale Dense Networks for Resource Efficient Image Classification

- 早期終了で精度を上げるためのモデルアーキテクチャ設計
- Multi-scale feature maps
  - earlier layerでも画像全体に対する特徴量を得るため
- Dense Connectivity
  - earlier layerにclassifierをつけることによる、deeper layerの精度劣化を防ぐため



Oral

Multi-Scale Dense Networks for Resource Efficient Image Classification

- 少ない計算量でも精度が高い

Figure 3: Anytime prediction on ImageNet.

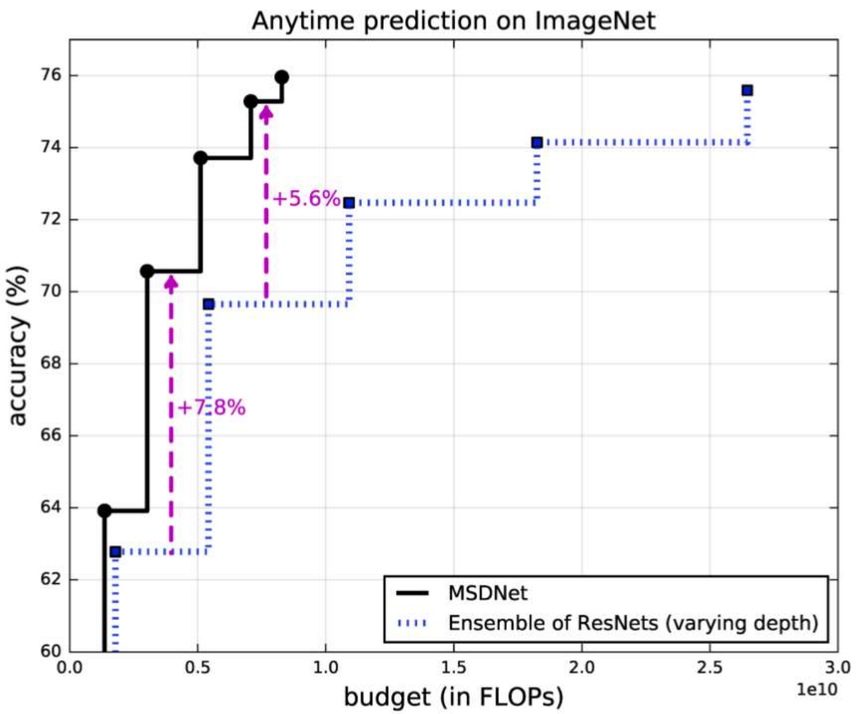
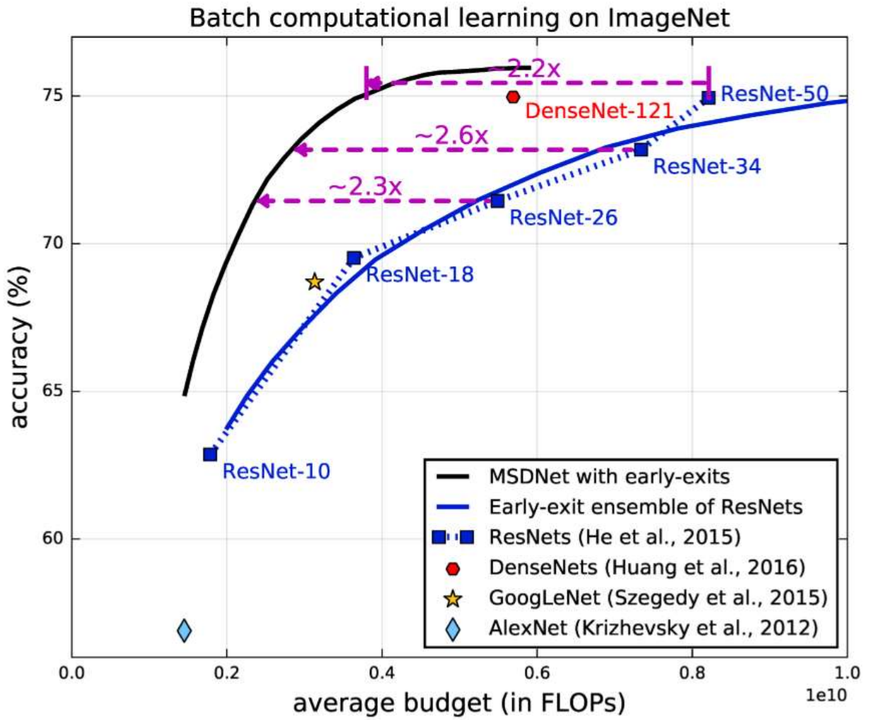


Figure 4: Prediction under batch computational budget on ImageNet.



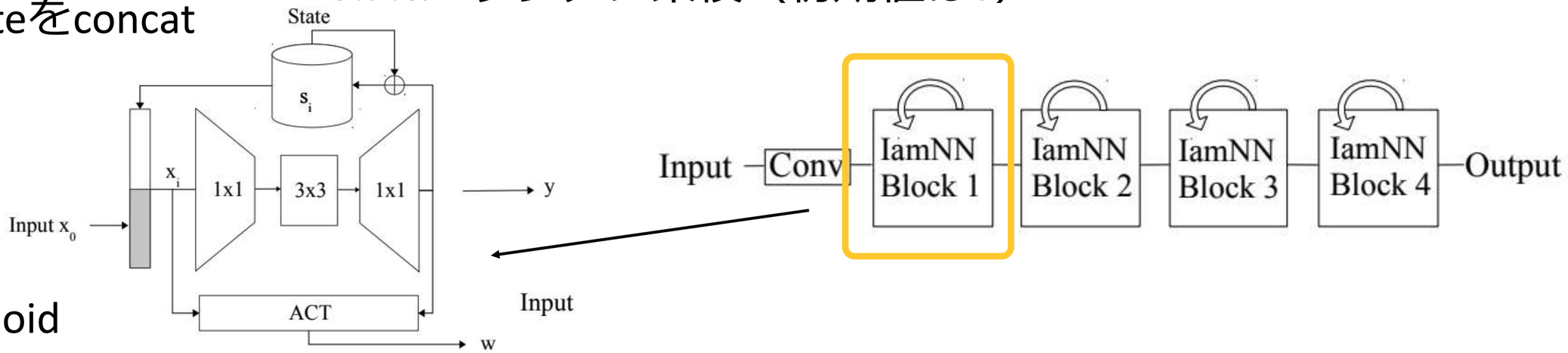
https://github.com/gaohuang/MSDNet

IamNN: Iterative and Adaptive Mobile Neural Network for efficient image classification *1

- ResNetを軽量化（Weight Sharing, Early Terminationの一種の組み合わせ）
- 1Block内でConvolutionをRNN的に繰り返し、特徴量をrefineする
- ACT(Adaptive Compute Time) Block*2 の累積スコアが1になったら次のBlockへ

1つ前のstateをconcat

stateバッファに累積（初期値は0）



fc - fc - sigmoid

1 Block内のBatchNormのWeightは、繰り返し毎に独立

*1 Sam Leroux. 2018. IamNN: Iterative and Adaptive Mobile Neural Network for Efficient Image Classification. arXiv:1804.10123. Retrieved from https://arxiv.org/abs/1804.10123

*2 Alex Graves. 2016. Adaptive Computation Time for Recurrent Neural Networks. arXiv:1603.08983. Retrieved from https://arxiv.org/abs/1603.08983

IamNN: Iterative and Adaptive Mobile Neural Network for efficient image classification

- ImageNetでは、ResNet152に対してTop5で 93.3% → 89.2% と下がるが90%のWeight削減と平均計算量を65%にまで削減

Dataset	Network	Params	FLOPS	Top1/ Top5 (%)
CIFAR10	ResNet101	42 M	2.5G	93.8
	IamNN	4.5 M	1.1G (.7G - 2G)	94.6
CIFAR100	ResNet101	43 M	2.5G	79.3
	IamNN	4.6 M	1.6G (.7G - 2G)	77.8
ImageNet	ResNet152	60 M	11.5 G	77.0 / 93.3
	ResNet18	12 M	1.8 G	69.5 / 89.2
Single crop	IamNN 1 iter	4.8 M	0.9 G	60.8 / 83.2
Single network	**IamNN**	**5 M**	**4 B** (2.5G - 9G)	**69.5 / 89.0**
	ShaResNet34	14 M	11 G	71.0 / 91.5
	Googlenet	7 M	1.6 G	65.8 / 87.1
	MobileNet1	4.2 M	570 M	70.6 / 89.5
	ShuffleNet2x	5.6 M	524 M	70.9 / 89.8
	SqueezeNet	1.3 M	830 M	57.5 / 80.3

Table1

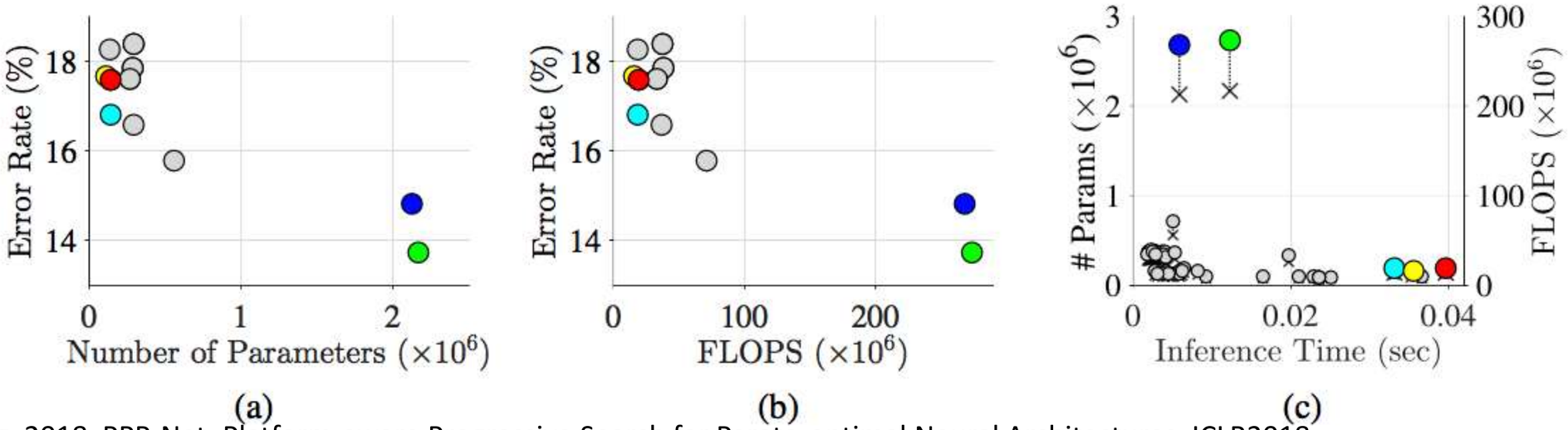
あれ、MobileNetの方が良いのでは...
→ IamNNもseparable化すれば
さらに軽量化できるはず
(と言っていた)

ARCHITECTURE SEARCH

PPP-Net: Platform-aware Progressive Search for Pareto-optimal Neural Architectures *1

- デバイスで精度が良い＆速いモデルを探索
  - Progressive Neural Architecture Search *2 を拡張
- モデル選択の指標は、精度に加えFLOPs、パラメータ数、推論時間
- 探索空間をMobile CNNの部品に限定、精度の予測をして学習を効率化

生成されたアーキテクチャ。既存軽量手法(CondenseNet)と同じアーキテクチャも (赤)

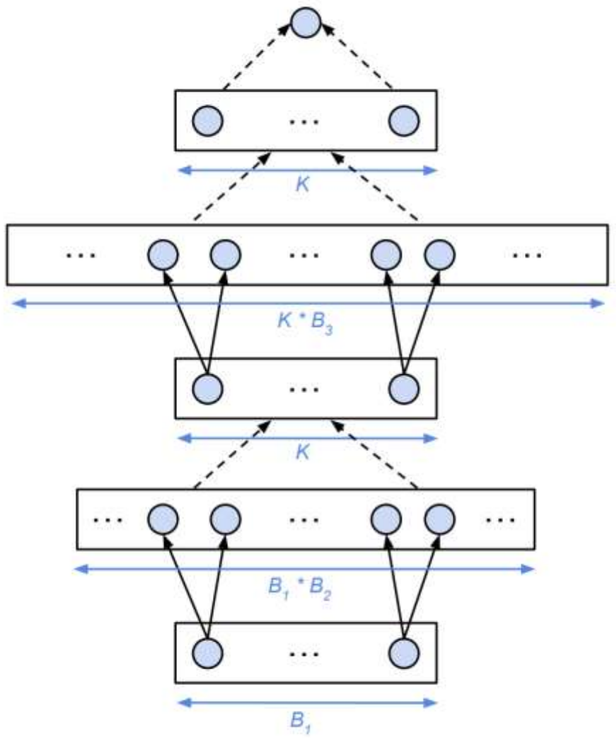


*1 Jin-Dong Dong. 2018. PPP-Net: Platform-aware Progressive Search for Pareto-optimal Neural Architectures. ICLR2018. Retrieved from https://openreview.net/forum?id=B1NT3TAIM

*2 Chenxi Liu. 2017. Progressive Neural Architecture Search. arXiv:1712.00559. Retrieved from https://arxiv.org/abs/1712.00559

PPP-Net: Platform-aware Progressive Search for Pareto-optimal Neural Architectures

- 「1層 増やす → 評価して絞りこむ」 を繰り返す



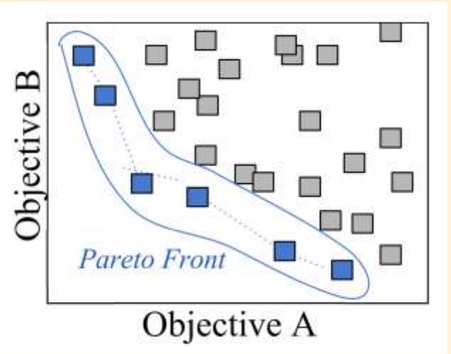
S_3

S'_3

S_2 評価して、K個に絞り込む

S'_2 2層目の全候補(Norm)の組み合わせ

S_1 1層目の全候補 (Conv)



精度、FLOPs、パラメータ数、速度がパレート最適なものを選択
精度は、精度推定用のRNNで得る
（全候補のfine-tuning不要となり速い）

Mobile CNNで使われるレイヤー

1x1 Conv	Batch Norm-Relu
3x3 Conv	Batch Norm
1x1 Group Conv	No op
3x3 Group Conv	
1x1 Learned Group Conv	
3x3 Depth-wise Conv	

Chenxi Liu. 2017. Progressive Neural Architecture Search. arXiv:1712.00559. Retrieved from https://arxiv.org/abs/1712.00559

N2N learning: Network to Network Compression via Policy Gradient Reinforcement Learning

- 元のモデルをどうやって小さくするか、を強化学習で解く
  - 2つのPolicy：レイヤーの削除とレイヤ内のパラメータ削減
- 報酬は、圧縮後のモデルの精度と圧縮率（パラメータ数）
  - 精度重視（軽いが精度低いモデルは報酬が少ない設計）

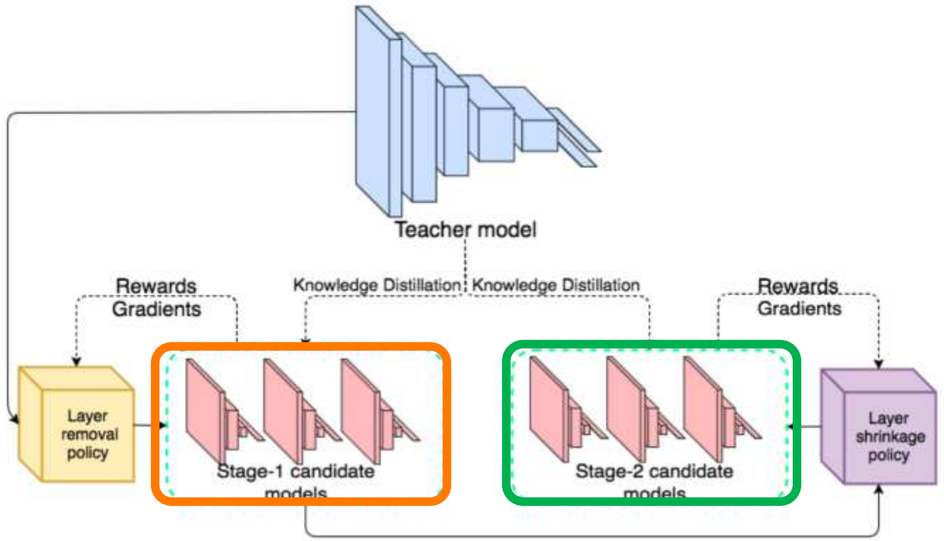
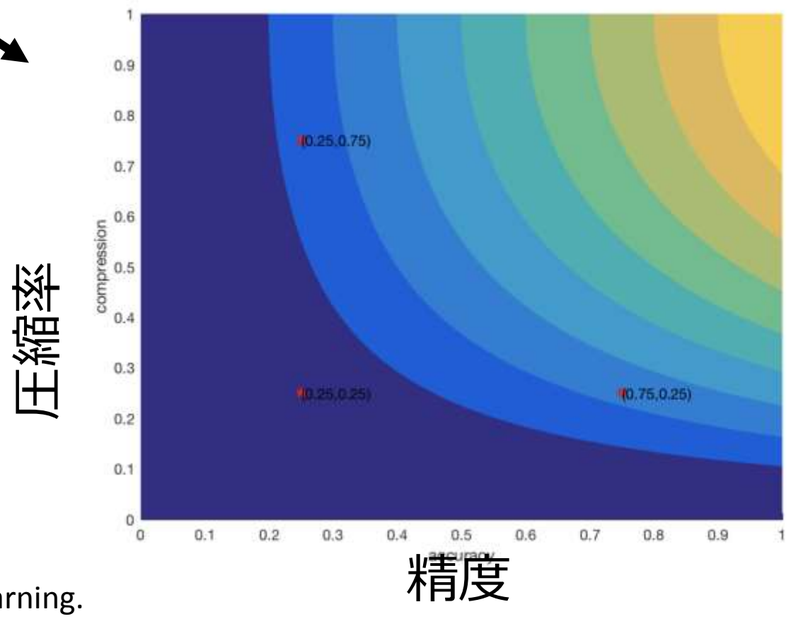


Figure 1: Layer Removal Policy removes layers of Teacher network architecture (stage-1 candidates) then Layer Shrinkage Policy reduces parameters (stage-2 candidates).

報酬関数



Anubhav Ashok. 2017. N2N Learning: Network to Network Compression via Policy Gradient Reinforcement Learning. arXiv:1709.06030. Retrieved from https://arxiv.org/abs/1709.06030

N2N learning: Network to Network Compression via Policy Gradient Reinforcement Learning

Layer removal policy network

a = 各レイヤーを{残す, 削除}

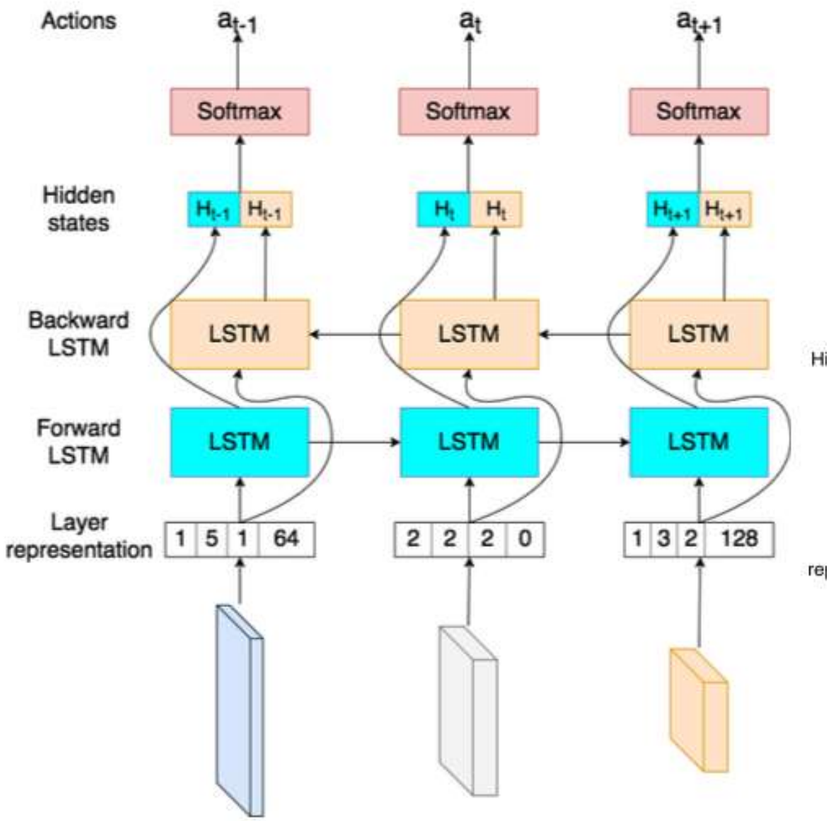
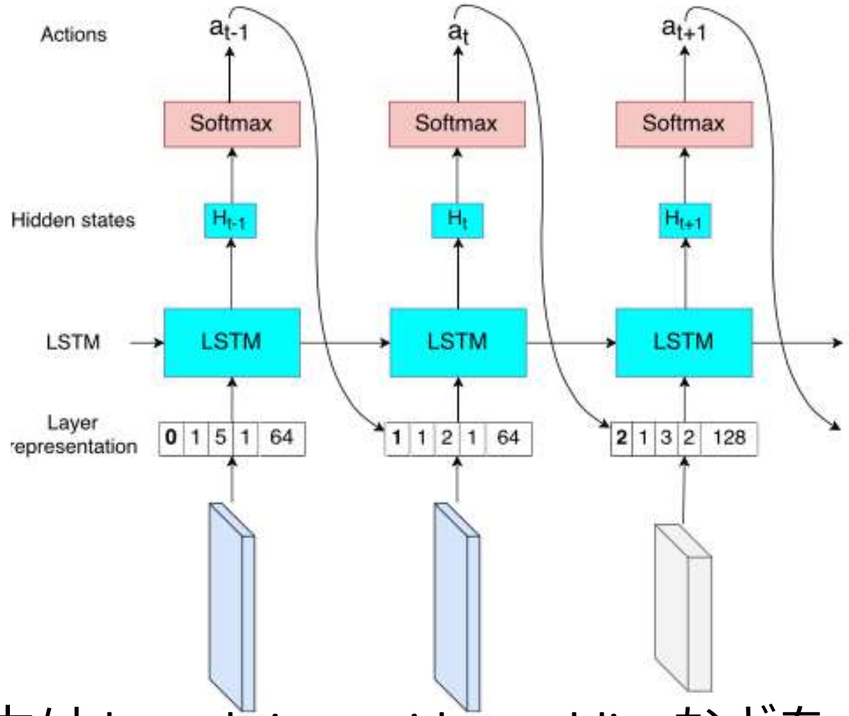


Figure 2

Layer shrinkage policy network

channel数などを元のa倍にする
a = {0.1, 0.2, ..., 1.0} (離散値)



入力は kernel size, stride, paddingなどをembedする

$$x_t = (l, k, s, p, n, s_{\text{start}}, s_{\text{end}}),$$

N2N learning: Network to Network Compression via Policy Gradient Reinforcement Learning

- 既存のPruningやDistillationより、精度も計算量も良いモデルを得た

Table 2: Pruning (Baseline)

Model	Acc.	#Params	Compr.	Δ Acc.
Teacher (MNIST/VGG-13)	99.54%	9.4M	—	—
Pruning	99.12%	162K	58x	-0.42%
Ours	**99.55%**	**73K**	**127x**	**+0.01%**
Teacher (CIFAR-10/VGG-19)	91.97%	20.2M	—	—
Pruning	91.06%	2.3M	8.7x	-0.91%
Ours	**92.05%**	**1.7M**	**11.8x**	**+0.08%**

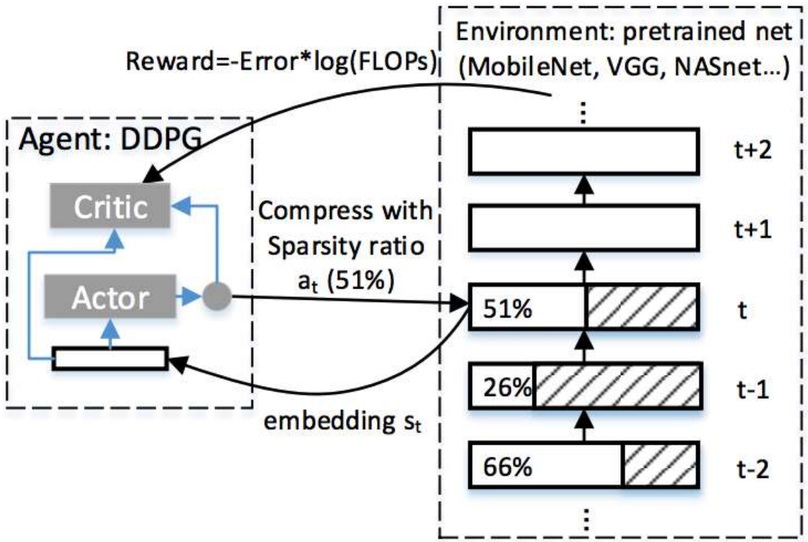
Table 3: Knowledge distillation with hand designed models (Baseline)

Model	Acc.	#Params	Compr.	Δ Acc.
Teacher (SVHN/ResNet-18)	95.24%	11.17M	—	—
SqueezeNet1.1	89.34%	727K	15x	-5.90%
Ours	**95.38%**	**564K**	**19.8x**	**+0.18%**
Teacher (CIFAR-10/ResNet-18)	92.01%	11.17M	—	—
FitNet-4	91.33%	1.2M	9.3x	-0.63%
VGG-small	83.93%	1.06M	10.5x	-8.08%
Ours	**91.81%**	**1.00M**	**11.0x**	**-0.20%**

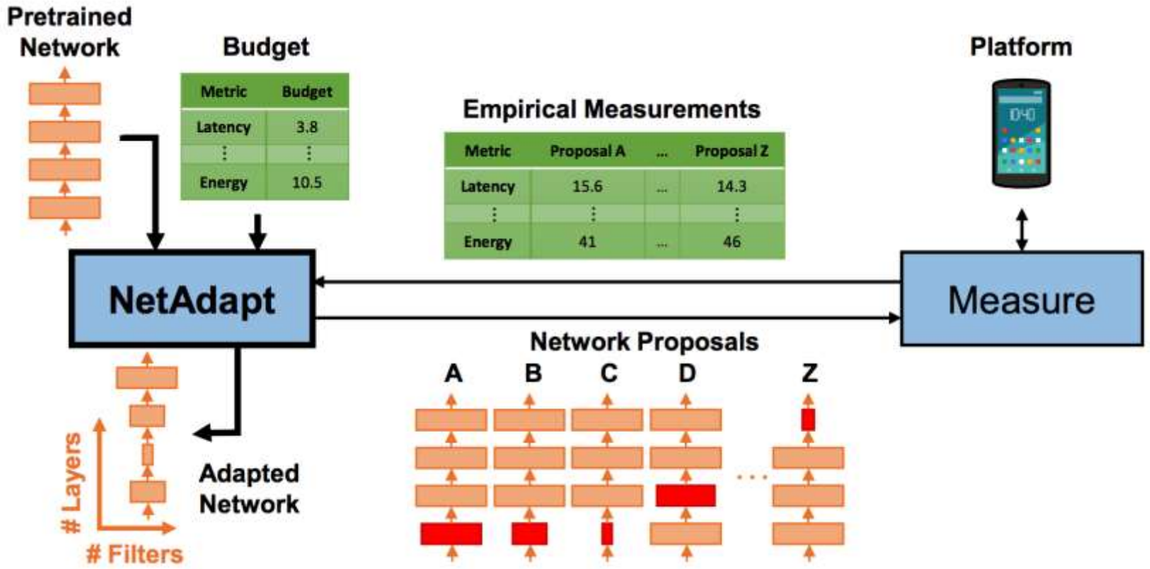
- 時間は結構かかる（報酬計算のため、各モデル候補で数epoch学習が必要）
  - MNIST / VGG-13 → 4 h
  - CIFAR10 / ResNet-18 → 20 h
  - ImageNet32x32 / ResNet-34 → 272h

参考：Architecture Searchによるモデル軽量化

- **ADC: Automated Deep Compression and Acceleration with Reinforcement Learning** *1
  圧縮したモデルのfine-tuning不要で
  報酬計算できるので学習が速い



- **NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications** *2
  実デバイスで計測しながら最適なモデル選択



*1 Yihui He. 2018. ADC: Automated Deep Compression and Acceleration with Reinforcement Learning. arXiv:1802.03494.
Retrieved from https://arxiv.org/abs/1802.03494
*2 Tien-Ju Yang. 2018. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications. arXiv:1804.03230.
Retrieved from https://arxiv.org/abs/1804.03230

PLATFORM

Platform

- 理論的に軽量化できた → 実現手段、実行環境が重要

Workshop

- DLVM: A modern compiler infrastructure for deep learning systems *1
  - NNVMやTensorFlow XLAのような、NNコンパイラ（既に開発終了...）

Poster

- Efficient Sparse-Winograd Convolutional Neural Networks *2
  - PruningしたCNNでもWinogradで高速化

Poster

- Espresso: Efficient Forward Propagation for Binary Deep Neural Networks *3
  - BNNをCPU/GPU(CUDA)で高速化するライブラリ

*1 Richard Wei. 2017. DLVM: A modern compiler infrastructure for deep learning systems. arXiv:1711.03016. Retrieved from https://arxiv.org/abs/1711.03016
*2 Xingyu Liu. 2018. Efficient Sparse-Winograd Convolutional Neural Networks. arXiv:1802.06367. Retrieved from https://arxiv.org/abs/1802.06367
*3 Fabrizio Pedersoli. 2017. Espresso: Efficient Forward Propagation for BCNNs. arXiv:1705.07175. Retrieved from https://arxiv.org/abs/1705.07175

まとめと所感

- ICLR2018におけるモデル軽量化
  - DistillationやArchitecture Searchによる軽量化
  - 特定のデバイスに最適化する汎用的な学習スキーム
  - Largeなモデルや難しいタスクでの手法評価も増えてる（ResNet / ImageNet）
  - 学習高速化など、リサーチャーが恩恵に預かれる技術も
- 所感
  - 実現したい目的に応じて、軽量化の引き出しを増やしておくことが重要
  - リソースに応じたAdaptiveな推論などは、
    システムの一部にDeep Learningを組み込む上で重要