Deep learning Semi

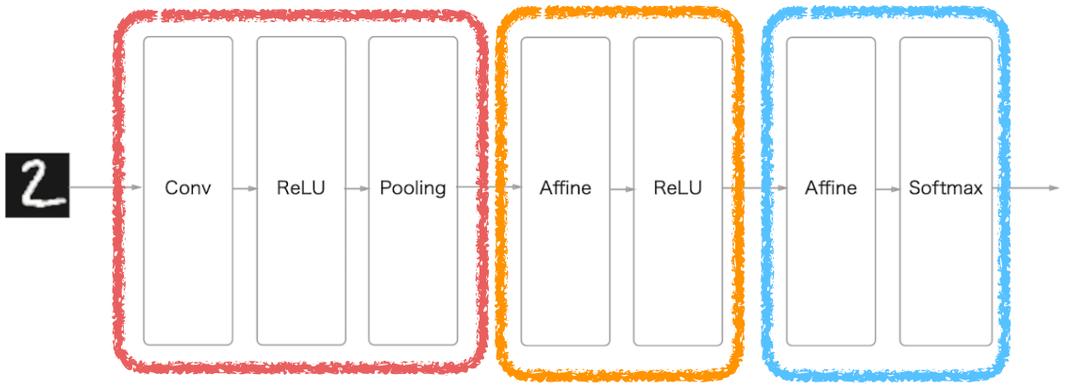
Oct. 22/2020

Tsuri Kimu
Ochanomizu University

ディープなネットワークへ

✤ 手書き数字認識：よりディープなCNN

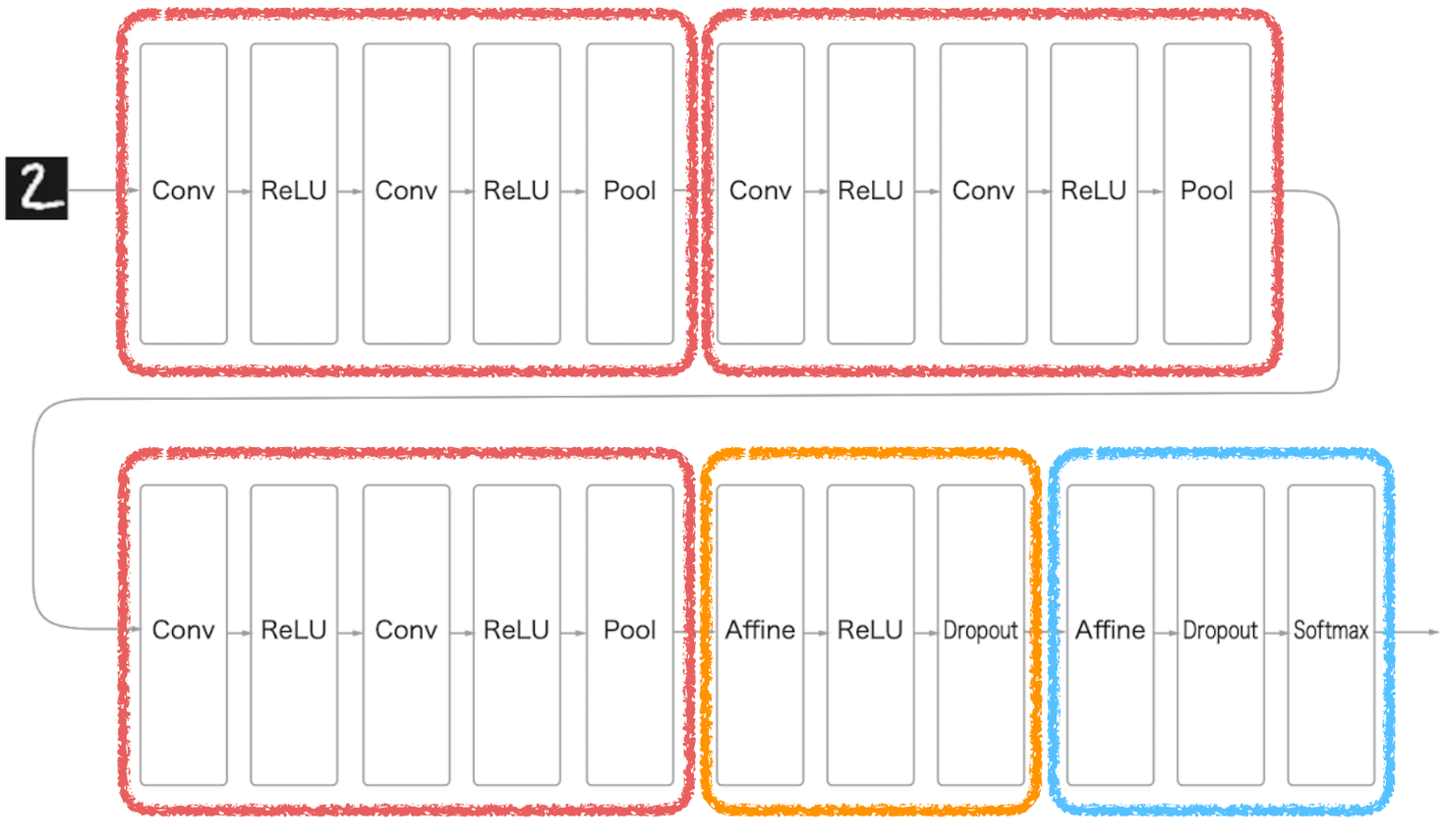
前回までのCNN



認識精度 **98~99%**

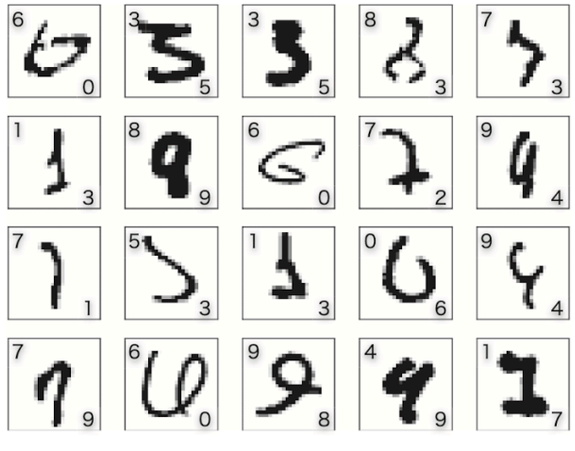
- 3 × 3 の小さなフィルターによる畳み込み層
- 活性化関数は ReLU
- 全結合層の後に Dropout レイヤを使用
- Adam による最適化
- 重みの初期値として「He の初期値」を使用

よりディープなCNN



認識精度 **99%**

誤認識した例：



人が”認識ミス”するレベル

さらに認識精度を上げるには？

✤ MNISTデータセットに対するランキングサイト
https://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html

Classification datasets results　About　Datasets　Contact

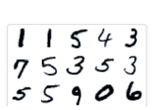
What is the class of this image ?

Discover the current state of the art in objects classification.

MNIST
CIFAR-10
CIFAR-100
STL-10
SVHN
ILSVRC2012 task 1

MNIST

who is the best in MNIST ?



MNIST 50 results collected

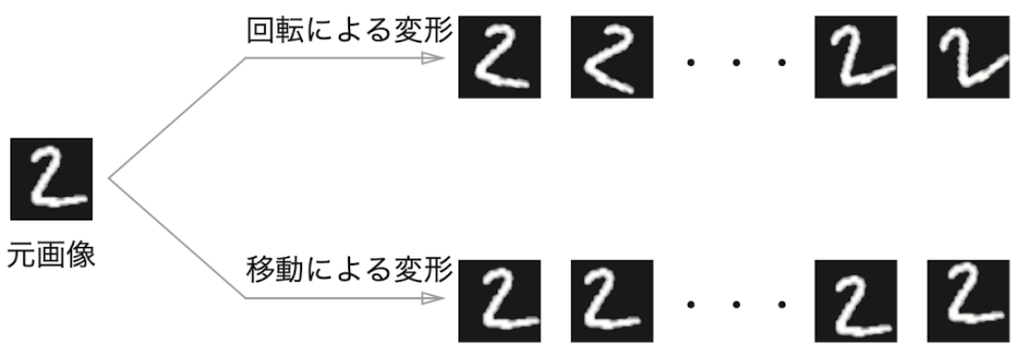
Units: error %

Result	Method	Venue	Details
0.21%	Regularization of Neural Networks using DropConnect	ICML 2013	
0.23%	Multi-column Deep Neural Networks for Image Classification	CVPR 2012	
0.23%	APAC: Augmented PAttern Classification with Neural Networks	arXiv 2015	
0.24%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
0.29%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
0.31%	Recurrent Convolutional Neural Network for Object Recognition	CVPR 2015	

CNNベースとした手法が上位で、
Deepにしなくても精度が良い

✤ 人工的にトレーニングデータを拡張する

- **Data Augmentation** (データの拡張)



・回転変形
・移動変形
・一部を切り出す（crop処理）
・左右をひっくり返す（flip処理）
・拡大縮小

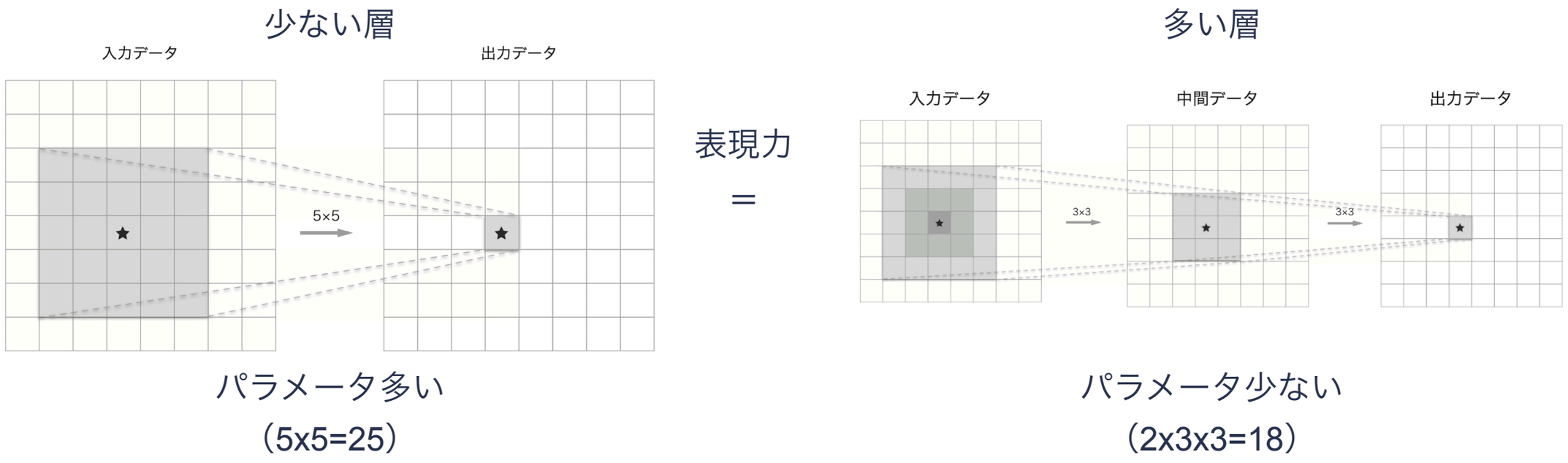
層を深くするモチベーション

✤ 重要性

- 大規模画像認識コンペティションの上位を占める手法がディープだから

✤ 利点

- ネットワークのパラメータ数を少なくできる

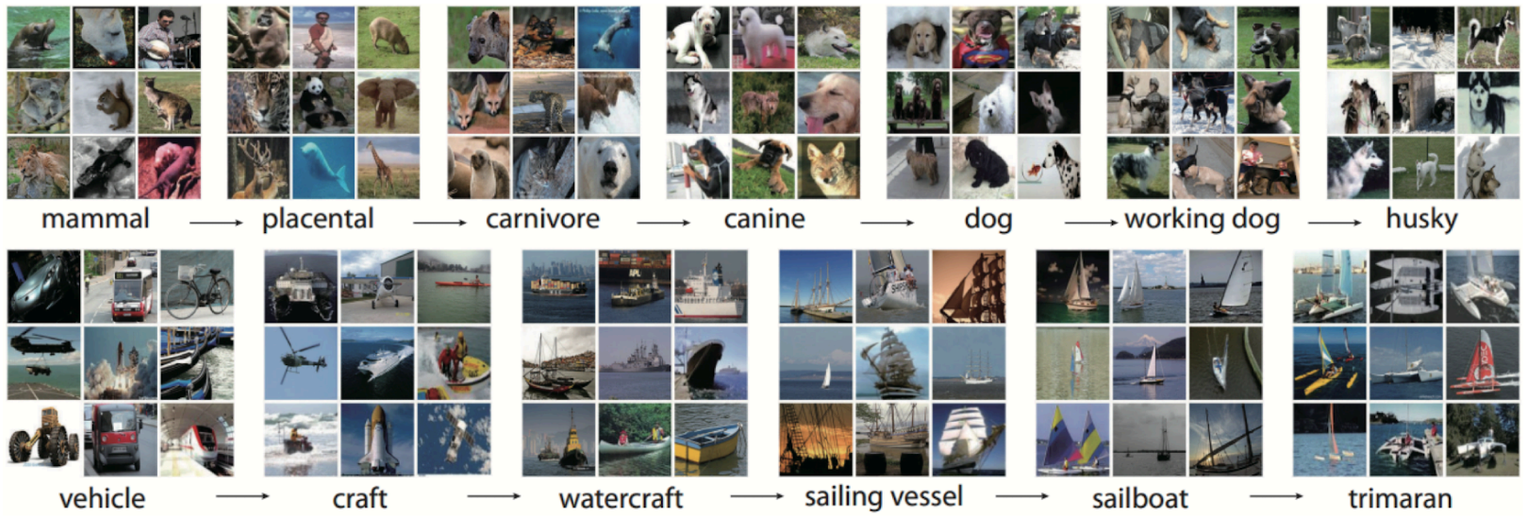


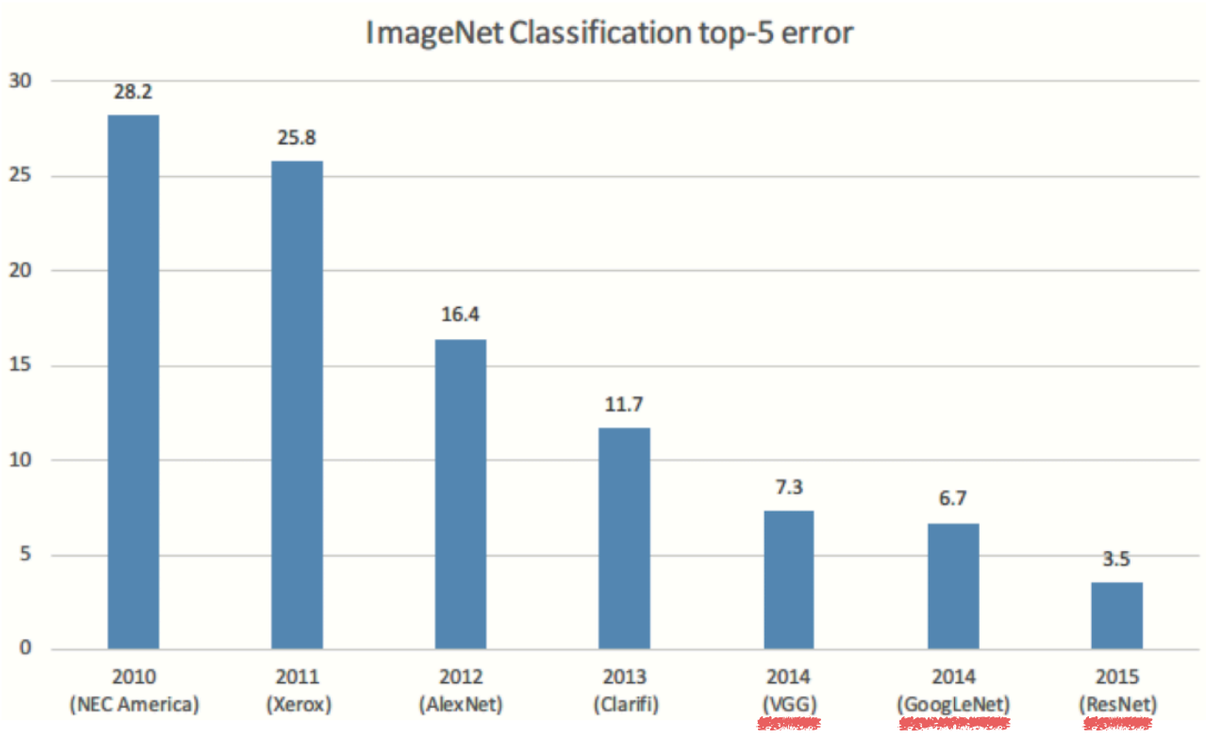
- 学習の効率化
  - 解くべき問題を階層別に分けることができるため、シンプルな問題として学習ができる　→ 効率よく学習することができる

ILSVRC

✤ 大規模画像認識コンペティション（ImageNet Large Scale Visual Recognition Challenge)

- ImageNet : 100万を超える画像のデータセット

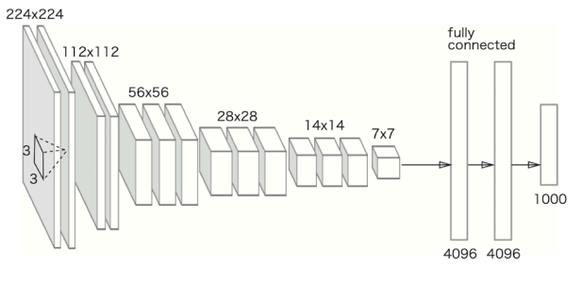




上位の手法

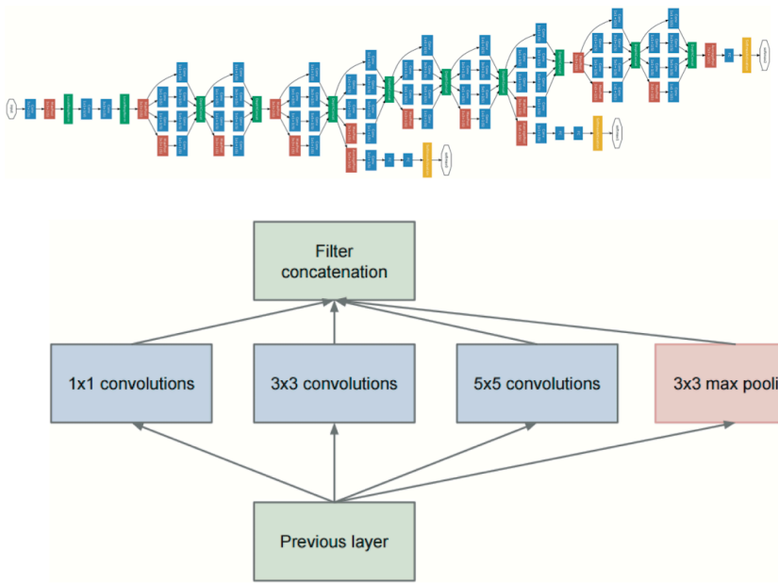
✤ VGG

- 基本的なCNNの形
- 全16層



✤ GoogLeNet

- 横にも幅が
  インセプション構造

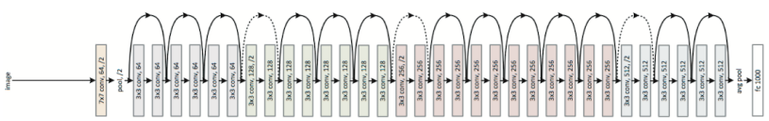


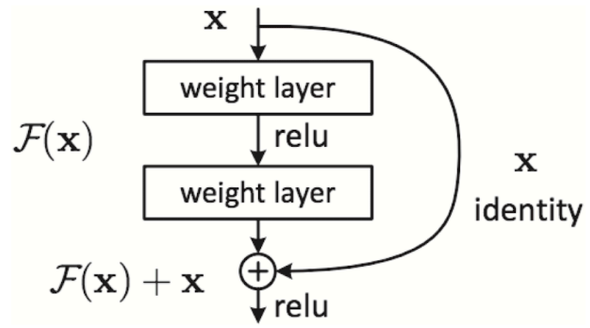
サイズの違うフィルターを複数使用
その結果を結合

1x1のチャンネル方向のサイズを減らす
→高速化

✤ ResNet

- Microsoftチーム
- スキップ構成



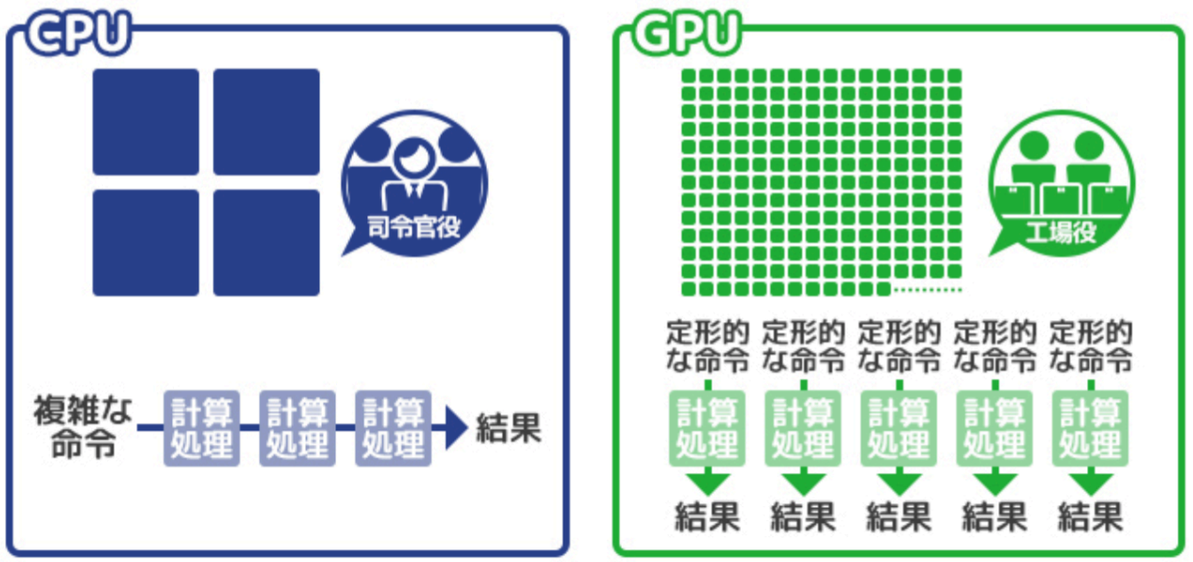


2層をおきにスキップ

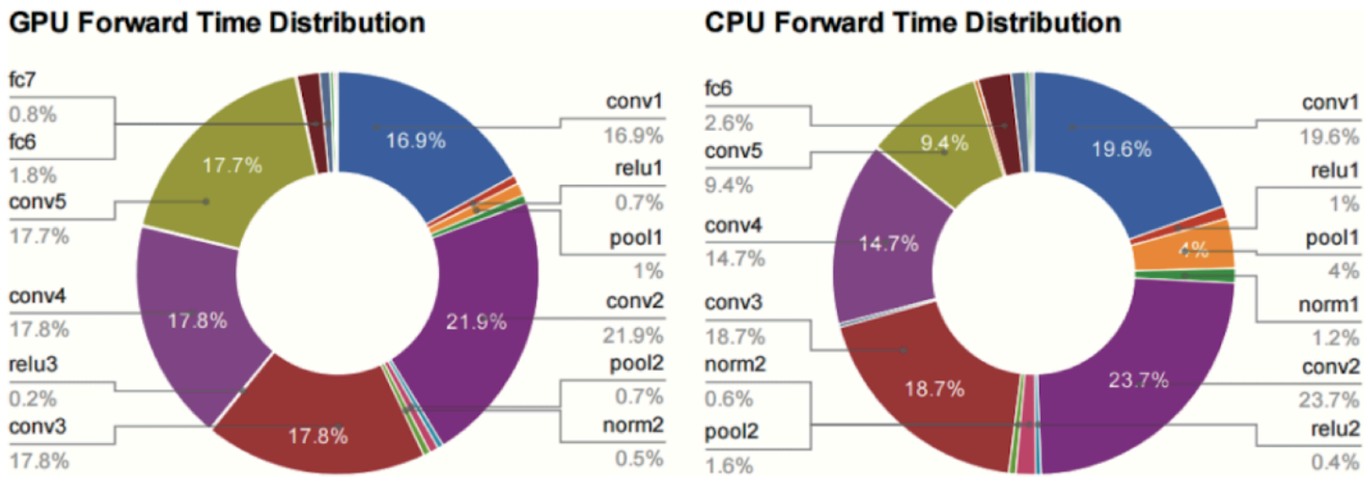
150層以上にしても認識精度が
向上し続ける

ディープラーニングの高速化

- CPU (Central Processing Unit)
  - 連続的な計算処理
  - コア数：数個
- GPU (Graphics Processing Unit)
  - 並列的な計算処理
  - コア数：数千個



- 各処理にかかる時間比率 (AlexNet)



Convolutional layer をいかに高速に効率よく行うか？

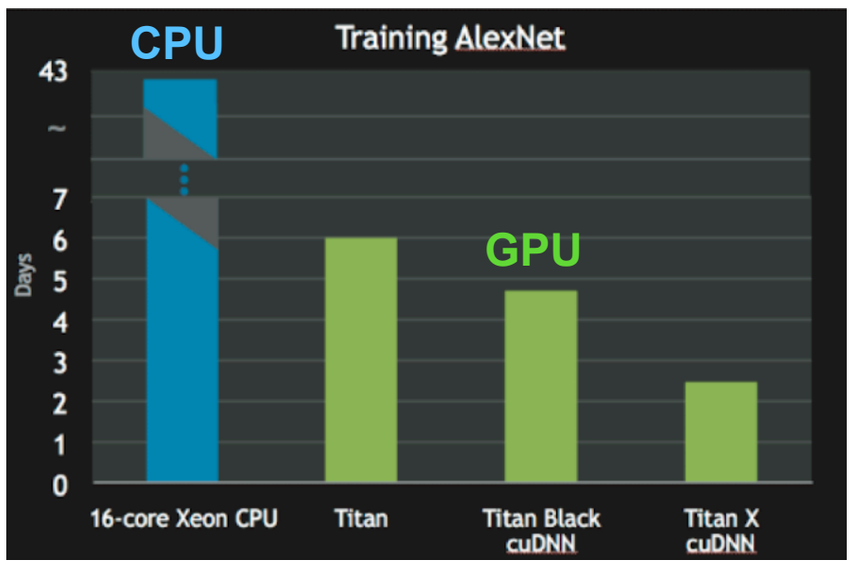
GPUによる高速化

✤ GPU (Graphics Processing Unit)
- 元々グラフィック処理を行うために作られた
- 並列処理が可能

✤ 畳み込み層で行う処理（最も時間がかかる層）

＝　積和演算
- 大量の積和演算をすることになる

✤ 学習に要する時間 (AlexNet)



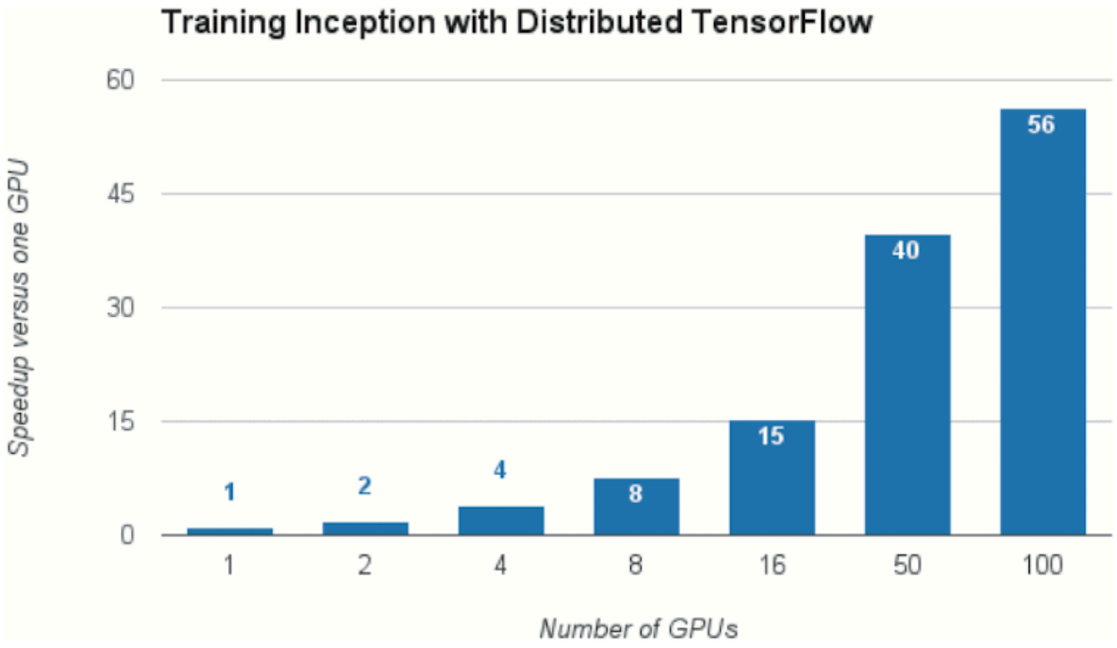
数日レベルまで短縮

分散学習

✤ 複数のマシンで分散して計算を行う

- TensorFlow, CNTK のフレームワーク

GPU1個の時を1として
高速化率



使用するGPUの数

数時間レベルまで短縮

✤ 分散学習の課題

- マシン間の通信、データの同期

  → いいフレームワークを使う（TensorFlowなど）

演算ビットの削減

✤ ボトルネック
- メモリ容量
  - 大量の重みパラメータ、中間データをメモリに収める必要がある
- バス帯域
  - バスを流れるデータが増加すると、帯域を超えたところで制限される
  - **ビット数をできるだけ小さくする**

✤ ディープラーニングとビット数
- ビット数が大きくなると数値計算誤差が小さくなる
- ディープラーニングではそこまでbit数を必要としない
- 16bitの半精度浮動小数点でも問題なく学習できることがわかっている

✤ Pythonでの実装
- 一般には64bit
- NumPyの16bitの型が用意されている

他にも、Bit数削減として、
重みや中間データを1bitで表現する
「Binarized neural networks」という手法があるらしい

まとめ

✤ ネットワークを深く(deep)することで、性能の向上が期待できる

- ILSVRCのコンペティションでDeep learningを用いた手法が上位を独占
- 有名なネットワークの VGG, GoogLeNet, ResNet を紹介

✤ Deep learningの高速化

- GPCを活用
- 分散学習をさせる
- bit数の削減

Backup

Adam

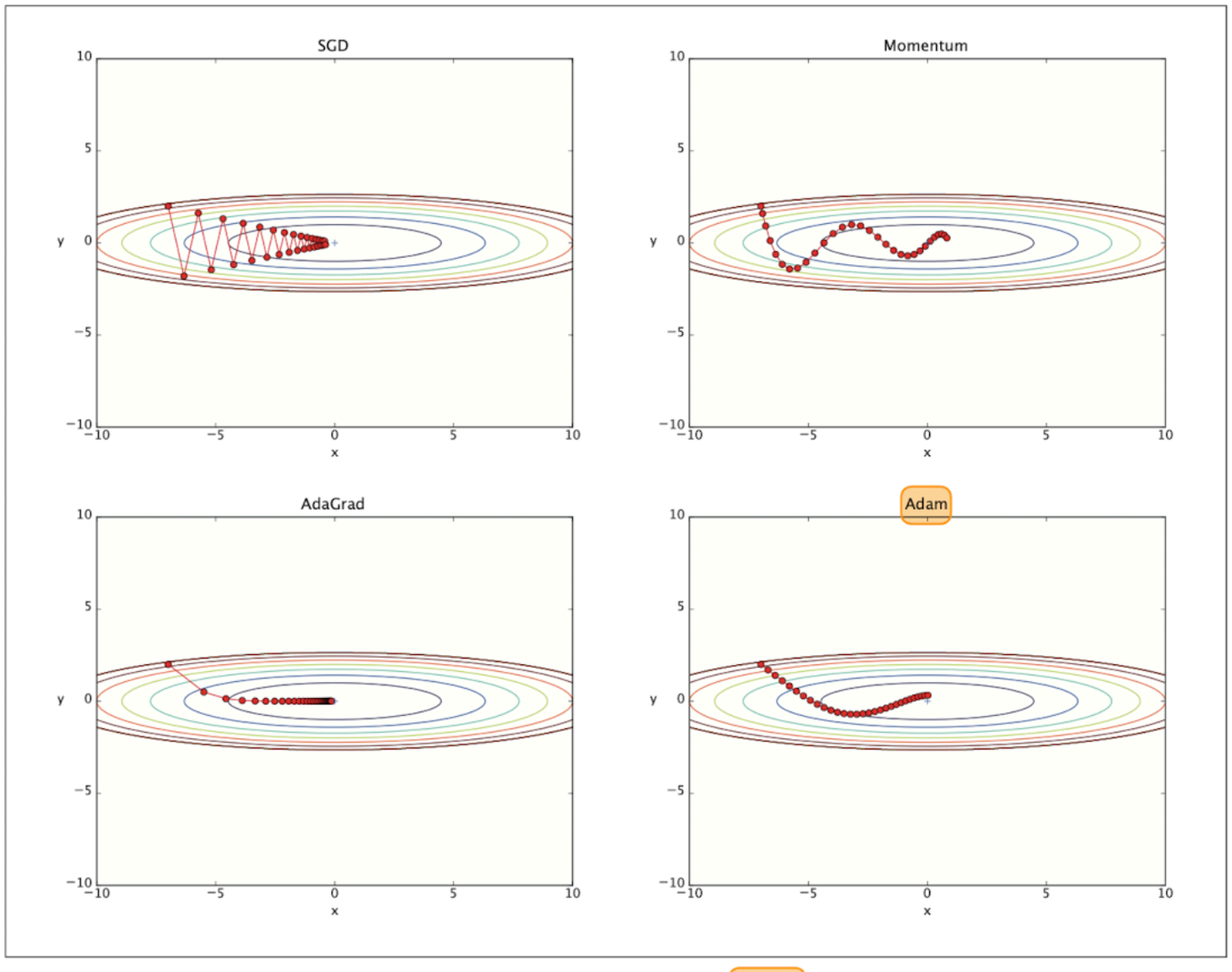
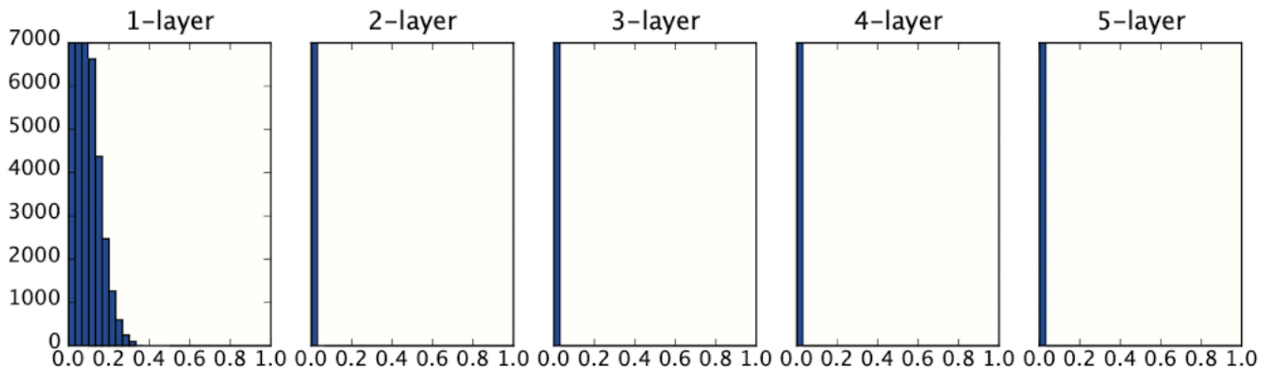


図6-8　最適化手法の比較：SGD、Momentum、AdaGrad、Adam

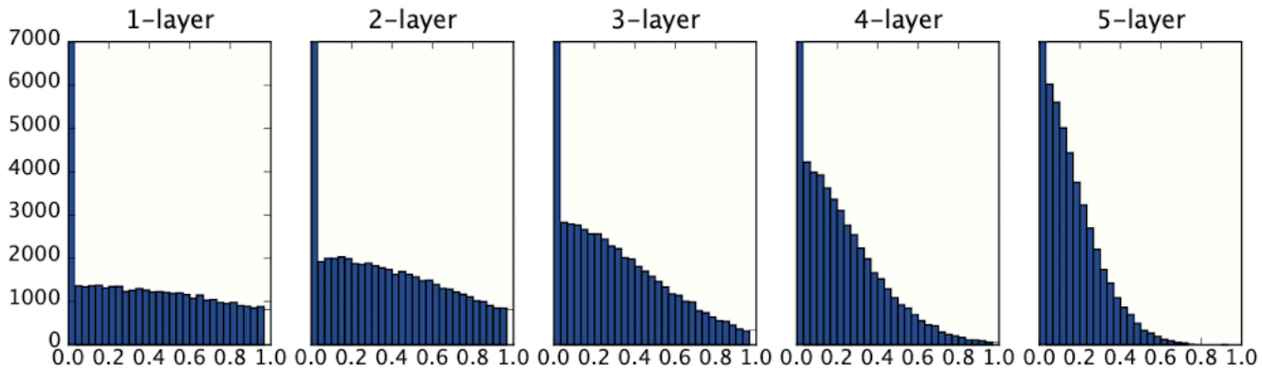
MomentumとAdaGradの手法を融合した感じの手法。

Heの初期値

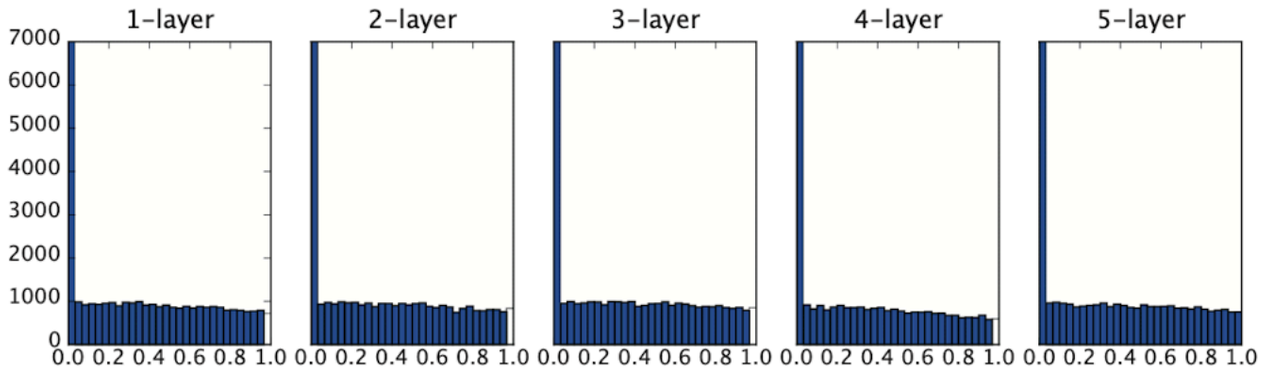


標準偏差が0.01のガウス分布を重みの初期値とした場合

学習の進みが早い



「Xavierの初期値」の場合



「Heの初期値」の場合