# CUDA ConvnetとDeep Learning with MultiGPU

株式会社クロスコンパス

http://www.xcompass.com

2014.9.5

# ConvNet

- ConvNet
  - 畳み込みニューラルネットワークのC++/CUDAによる実装
  - 任意の非巡回グラフのモデル設計が可能
  - Fermi-generation GPU (GTX 4xx, GTX 5xx, Tesla相当)が必要
  - 自然画像分類ベンチマークCIFAR10
    10種類画像の分類を約20分でエラー13%まで減らす
  - http://code.google.com/p/cuda-convnet/

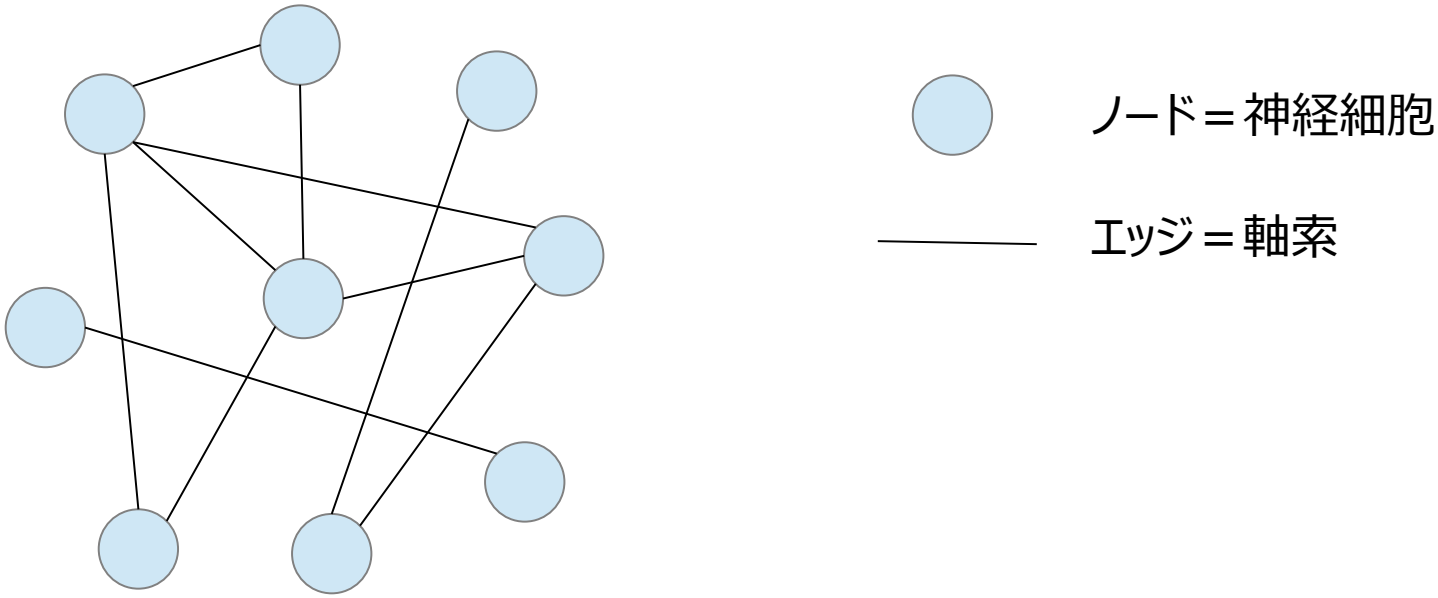
- **ConvNet2 (2014/07/18～)**
  - **Kepler-generation GPUs (Geforce Titan, K20, K40)の学習時間が短く**
  - **マルチGPUの学習が可能**
  - **https://code.google.com/p/cuda-convnet2/**

# ConvNetで用いられるニューラルネットワーク

| 名前 | 結合 | 重み共有 | 出力 |
|---|---|---|---|
| data | - | | データ、画像サイズなど |
| logprob, labvec | - | | 分類ラベル |
| fc | 全 | - | ノード活性化関数による |
| conv | 畳み込み | ○ | ノード活性化関数による |
| local | 畳み込み | × | ノード活性化関数による |
| rnorm | 畳み込み | - | 入力の正規化 |
| pool | 畳み込み | - | 入力の最大値、平均など |
| dropout | - | - | ドロップアウト |
| probs | 全 | - | Softmax |
| logprob | - | - | Cross Entropy |

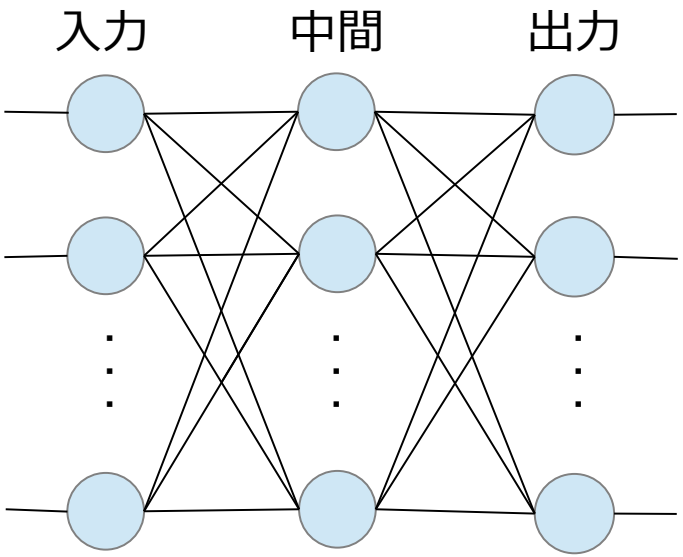
# ニューラルネットワーク

- Neural Network (NN)
- 神経回路網を模倣することで脳機能、主に人工知能を実現する努力
- ノードとエッジで構成

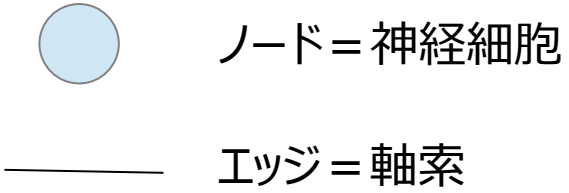


# フィードフォワード　ニューラルネットワーク

- Feed-forward Neural Network (FNN)
- ループ結合を持たないニューラルネットワーク

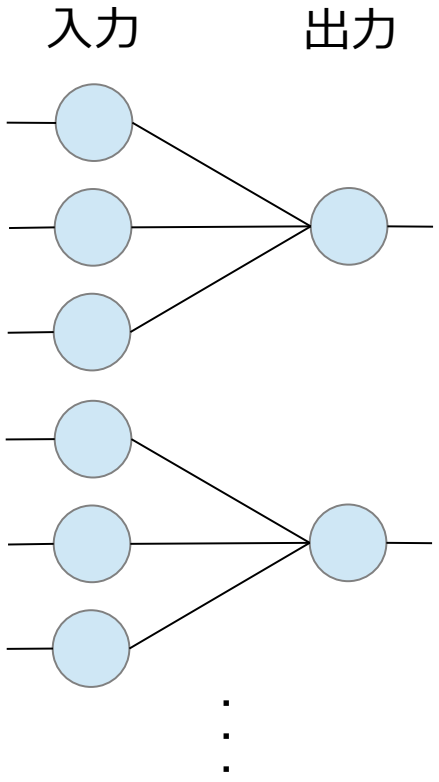



フィードフォワードニューラルネットワーク

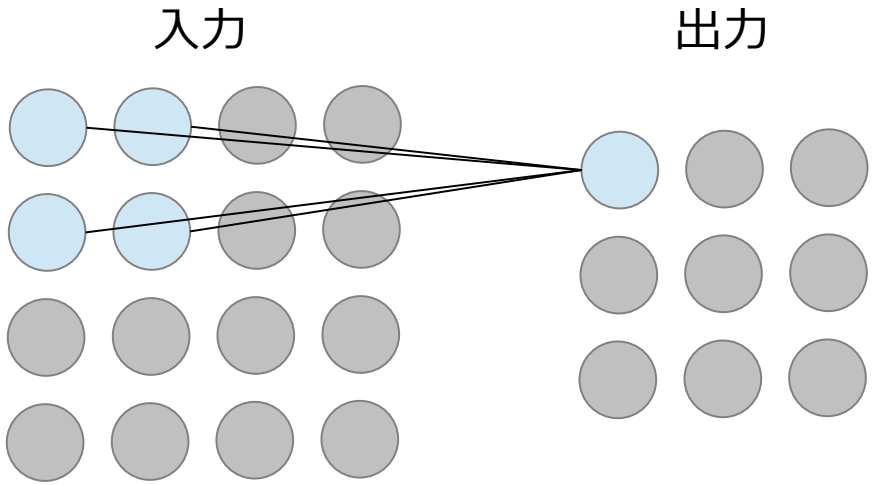


# 局所結合ニューラルネットワーク

- 事前知識を用いたニューラルネットワーク設計の一つ

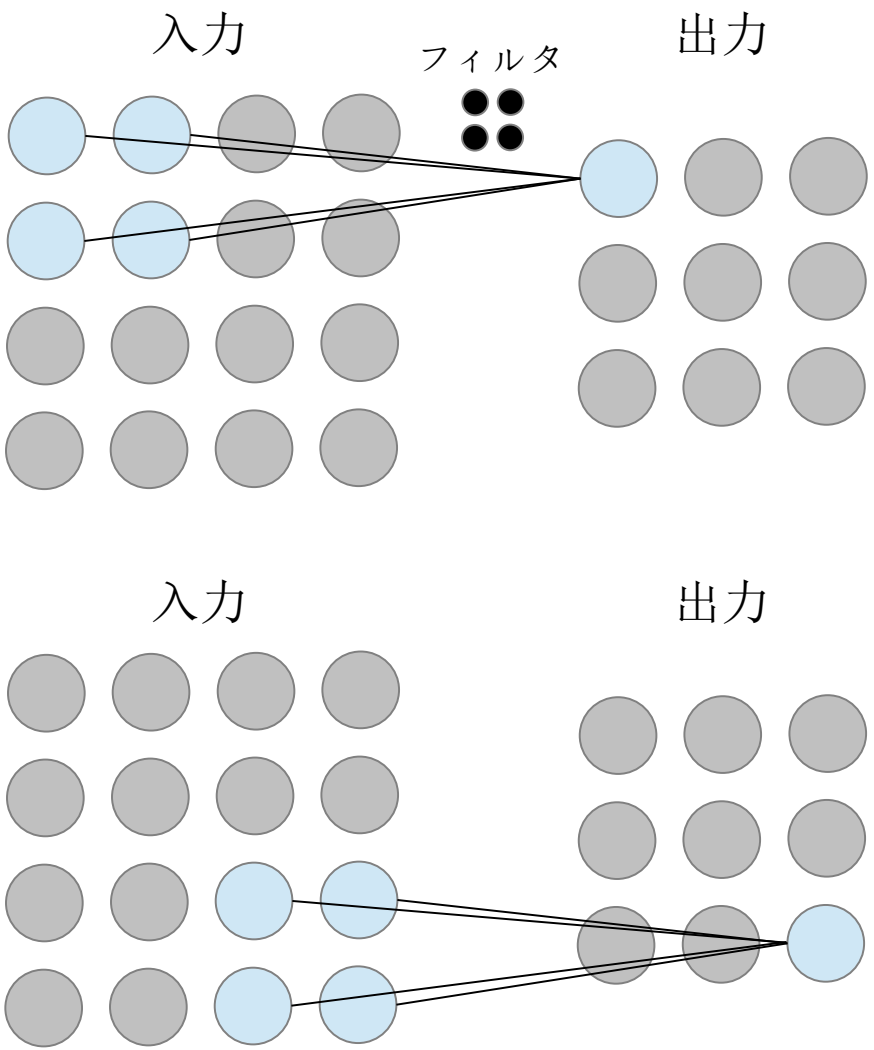



1次元局所結合




2次元局所結合

# 畳み込みニューラルネットワーク

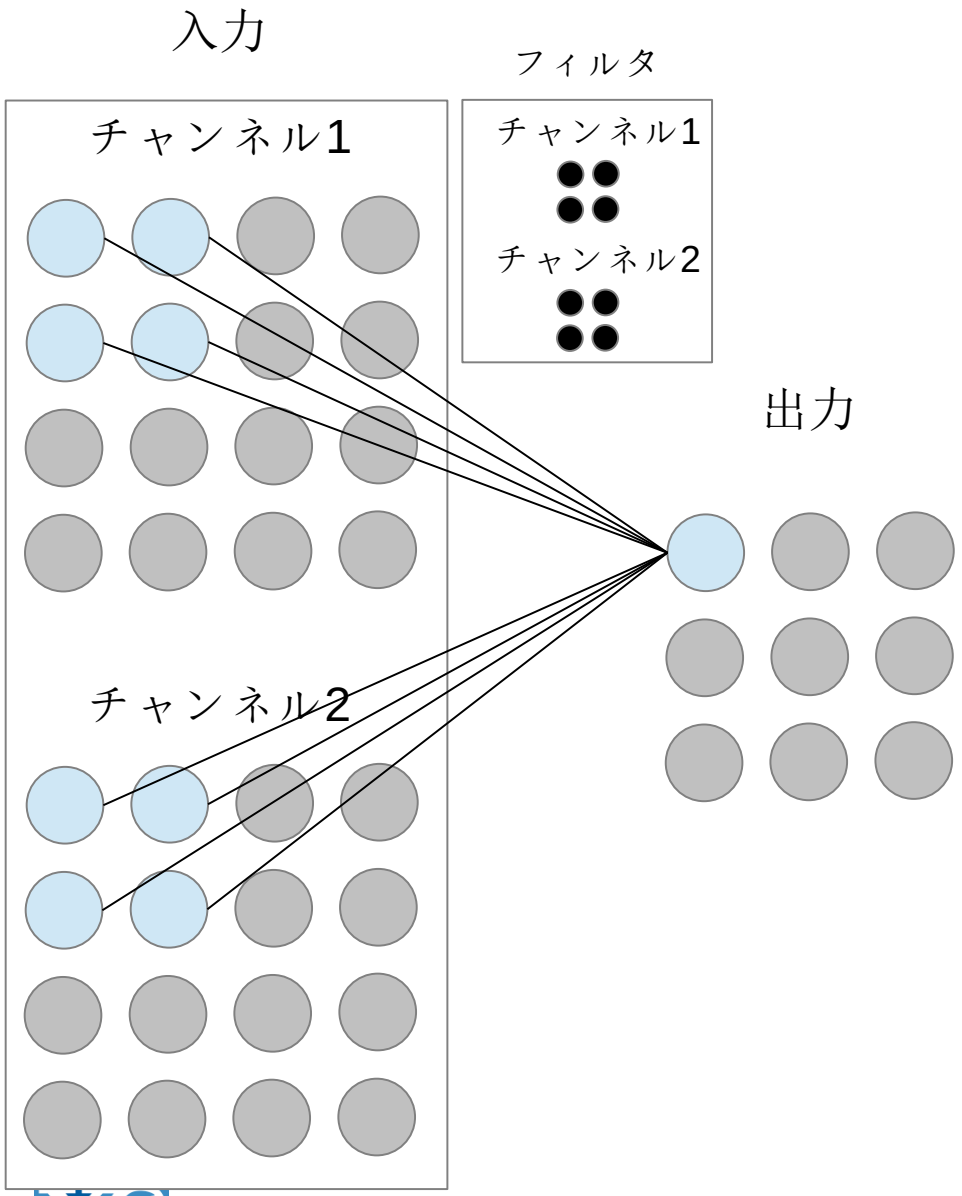



- Convolutional NN (CNN)
- 指定範囲入力ノードをグループ化
- 同グループは同出力ノードに結合
- 結合パラメータのグループを

フィルタと呼ぶ

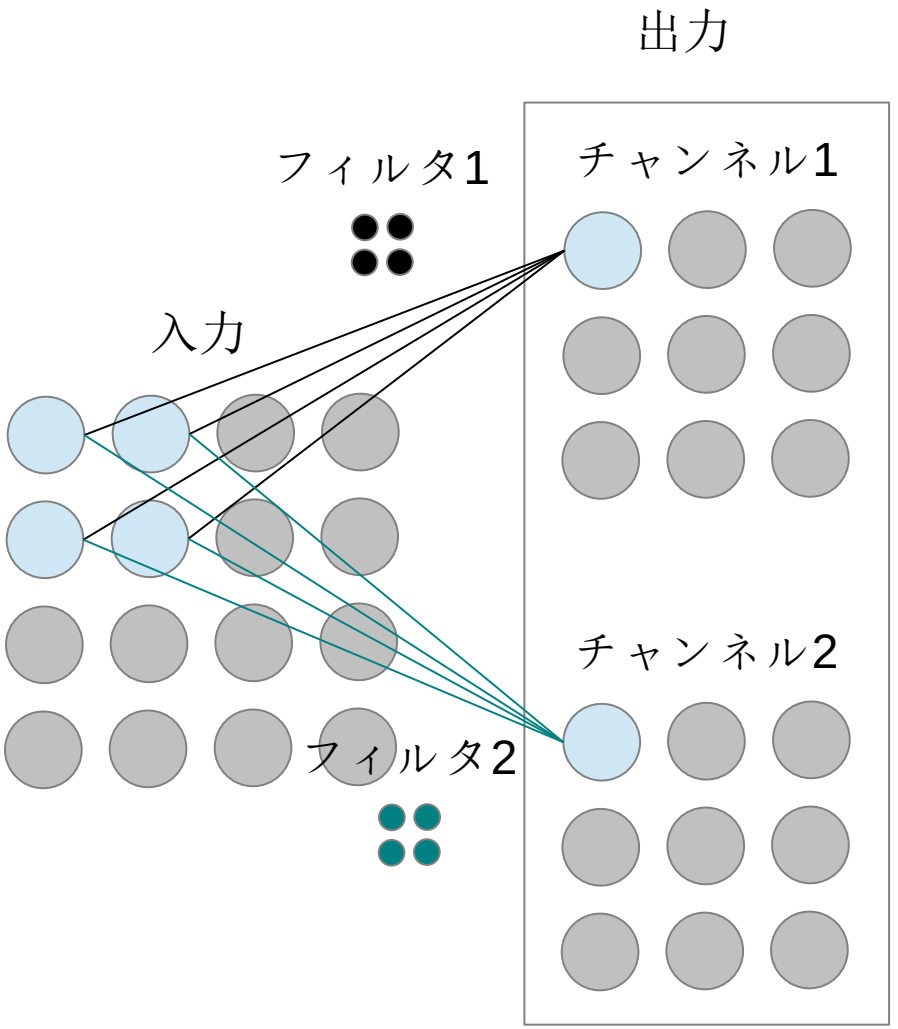
- 各グループの結合パラメータを共有することでパラメータを削減可能

# 畳み込みニューラルネットワーク




- デジタル画像を構成するピクセルは3原色で表現されている
- チャンネルとは、その原色の一つだけで構成される動作サイズ画像
- 入力に多数チャンネルで構成された場合、同数チャンネルのフィルタによって出力ノードに結合

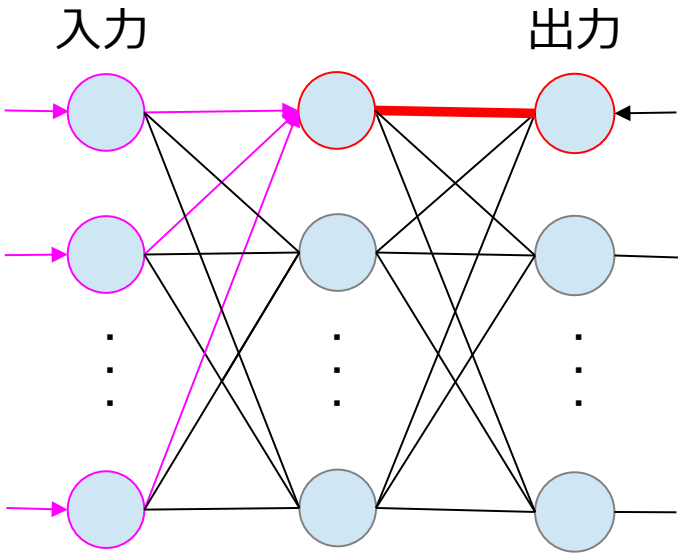
# 畳み込みニューラルネットワーク



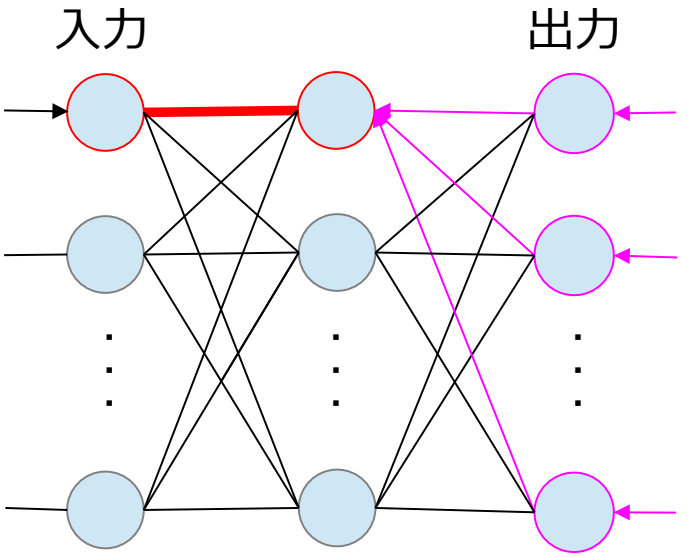

- フィルタが多数ある場合、同数分の出力チャンネルが得られる

# 誤差逆伝搬(バックプロパゲーション)

- FNNの学習法の一つ
- 出力の誤差を小さくなるように誤差と入力で重みを調整
- 学習対象の重みまで、入力を伝搬、出力誤差を逆伝搬



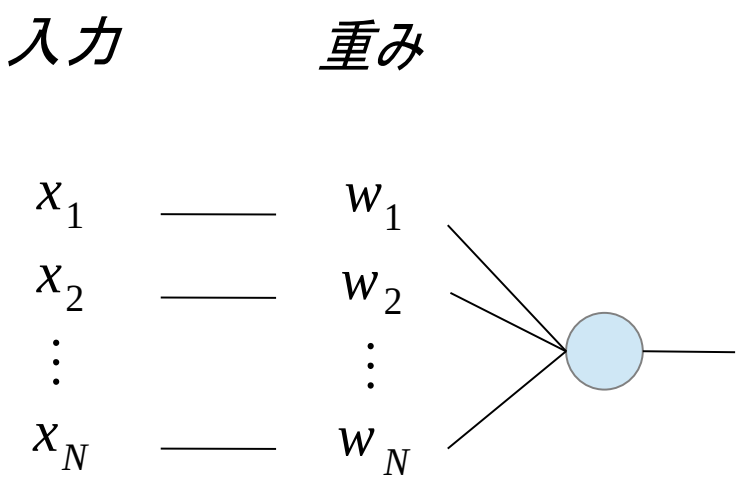

学習対象重み(赤エッジ)まで
入力を伝搬(ピンクエッジ)




学習対象重み(赤エッジ)まで
出力誤差を伝搬(ピンクエッジ)

# ノードの出力

- データが入力された際、エッジの持つ重みと掛け算を行い
- その結果を結合先ノードで足し算を行う
- ノードの持つ活性化関数を通して出力する(Sigmoid, tanh, ReLU)



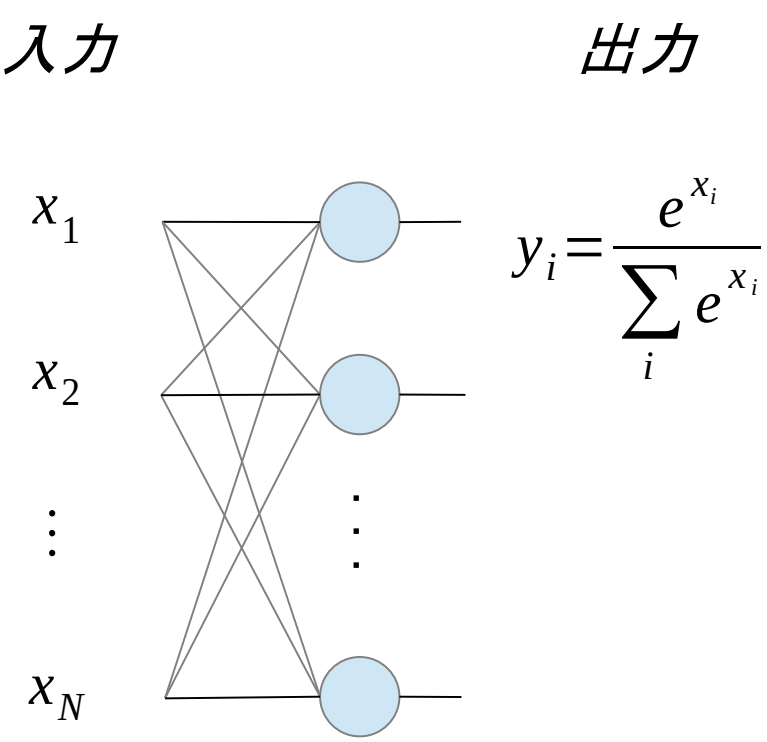

$$z = \sum_i x_i w_i$$

$$y = f(z)$$

$f$ : 活性化関数

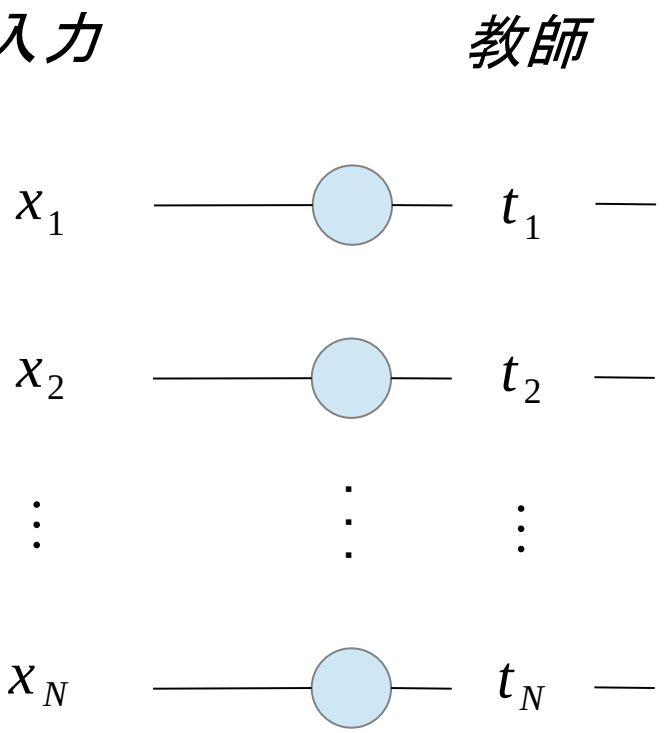
# Softmax

- データ分類するニューラルネットワークの出力に用いられる
- 出力の正規化の一つ
- 出力の合計が1.0になり、分類の確からしさとして扱う



# Cross Entropy

- データ分類するニューラルネットワークの目的関数に用いられる
- 出力誤差の一つ
- 入力と教師が一致する場合誤差が0になる

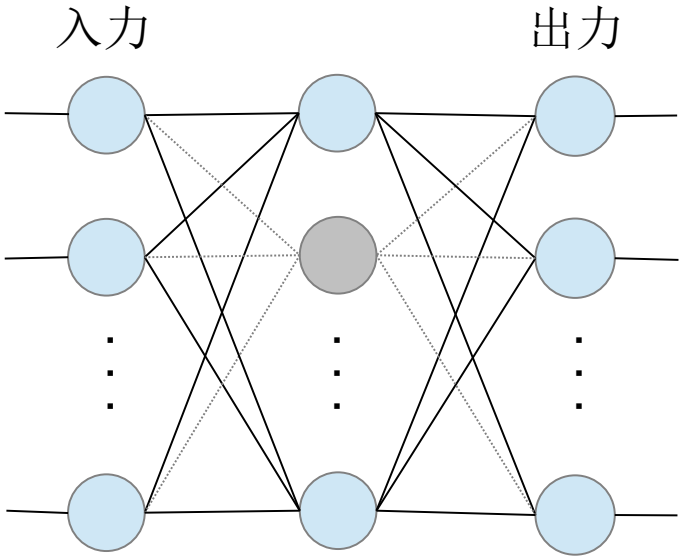



出力

$$y_i = t_i \log x_i + (1-t_i)\log(1-x_i)$$
$$= \begin{cases} \log x_i & , t_i = 1 \\ \log(1-x_i) & , t_i = 0 \end{cases}$$

# ドロップアウト

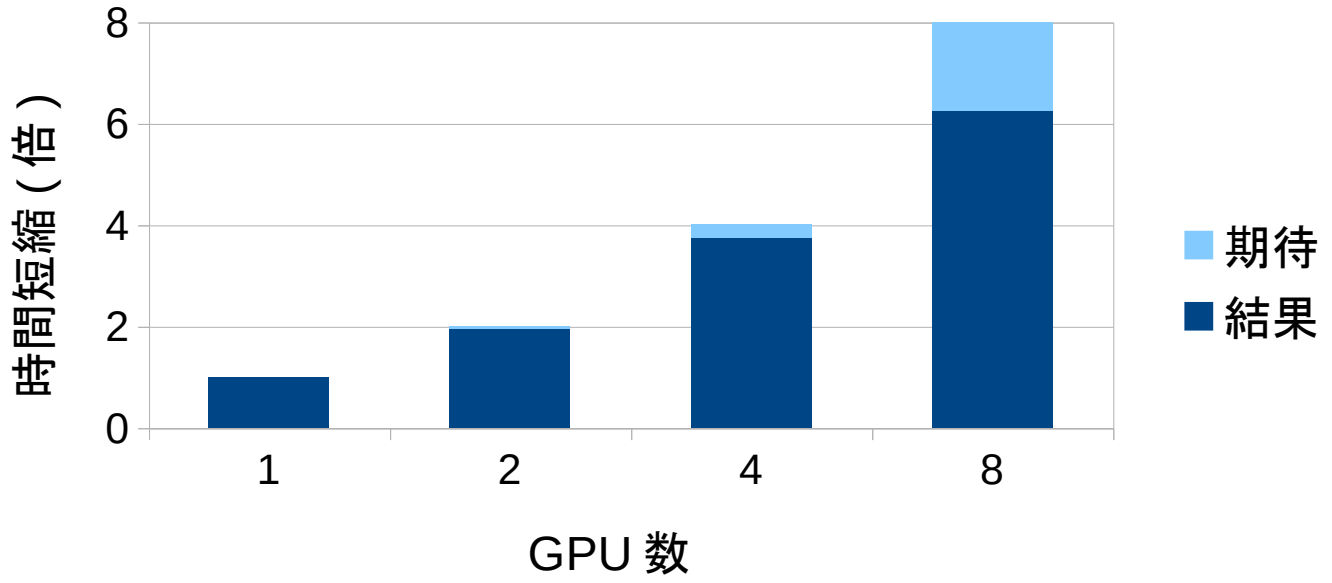
- Improving neural networks by preventing co-adaptation of feature detectors, G. E. Hinton et al., 2012
- 小数データで多層ニューラルネットワークを学習する際に生じる過学習の回避法の一つ
- ランダムにノードを無効化




ドロップアウト (グレー)

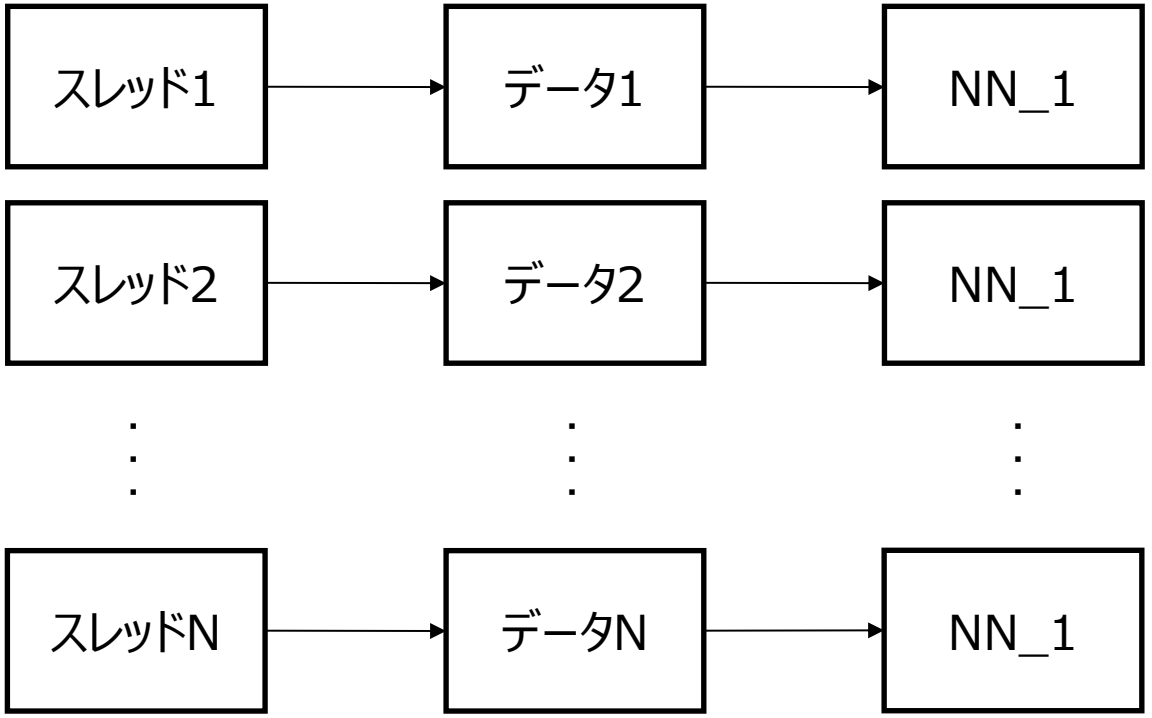
# マルチGPUによるCNN学習

- One weird trick for parallelizing convolutional neural networks, A. Krizhevsky, 2014
- 8 NVIDIA K20 GPUs
- 2 Intel 12 core CPUs



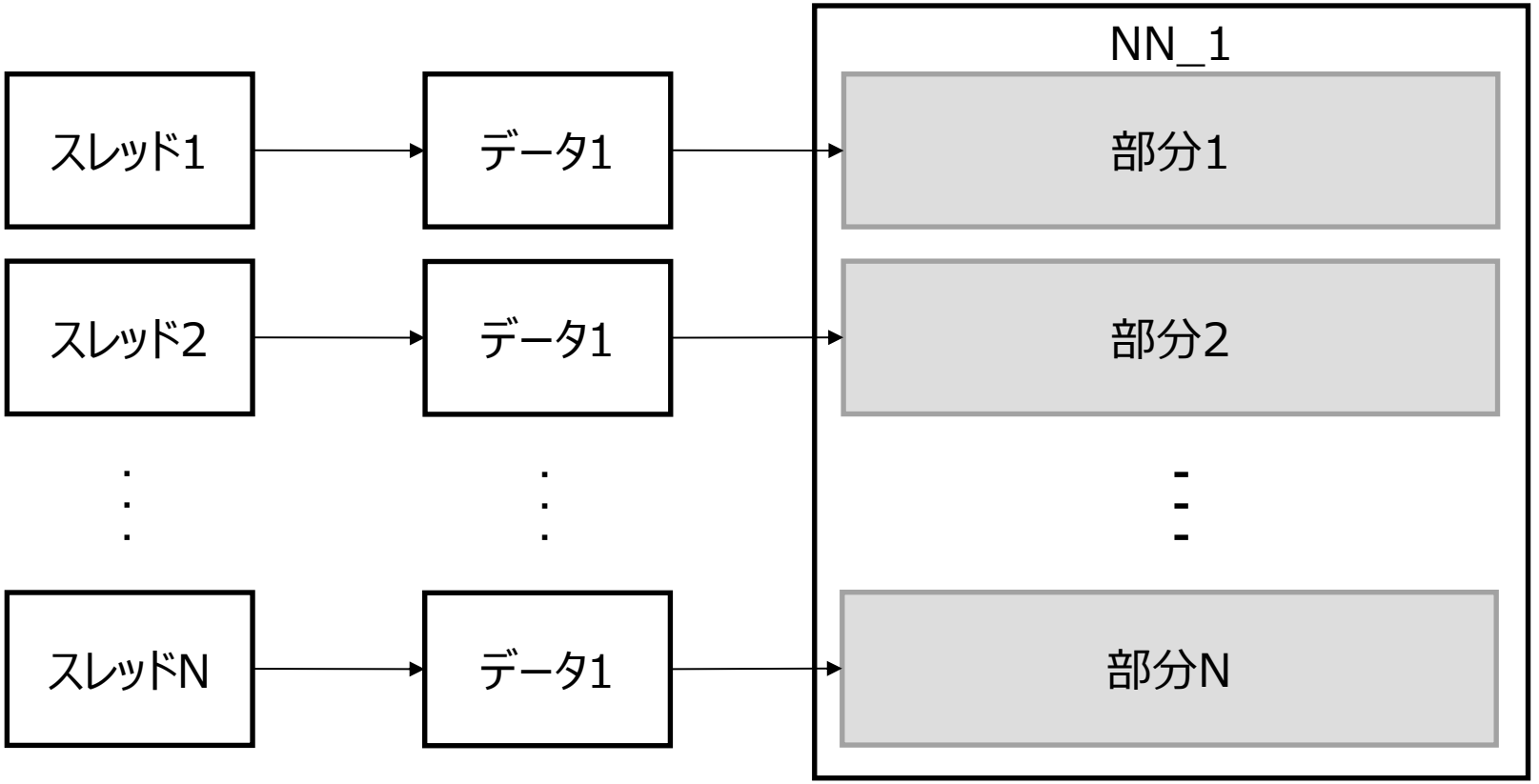
# Data Parallelism

- 各スレッドが別データで同ネットワークを学習
- 重みの計算コストが多ければ効果的 (CNN)



# Model Parallelism

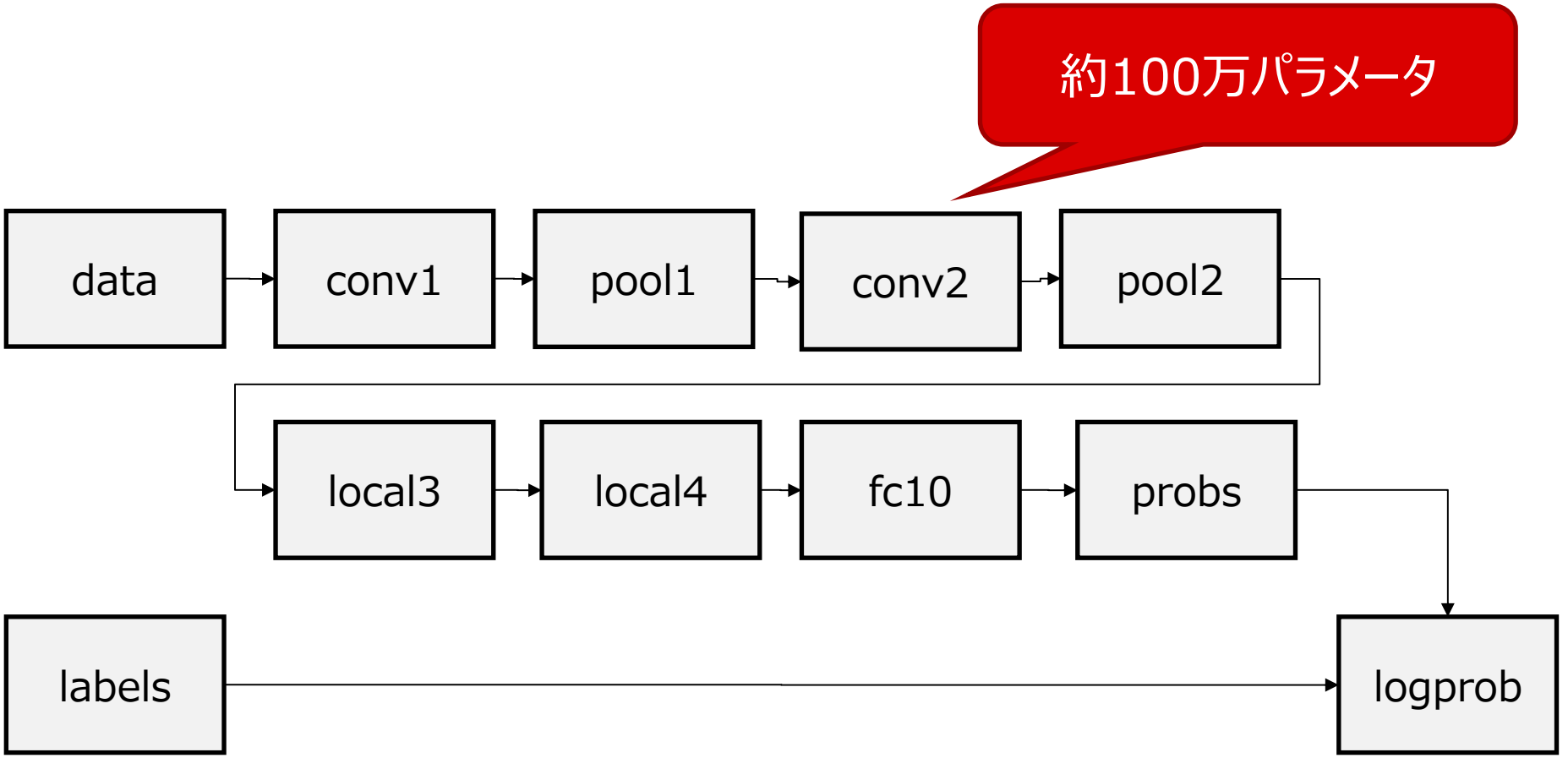
- 各スレッドが同データでネットワークの違う部分を学習
- ネットワークが可能な限り、スレッドが互いにコミュニケート
- ノード活性化の計算コストが多ければ効果的 (全結合FNN)



# 実験

- ConvNet / ConvNet2
- データ
  - CIFAR10
  - ImageNet
- GPUs
  - GeForce GTX 560 Ti
  - 2 Tesla K20c

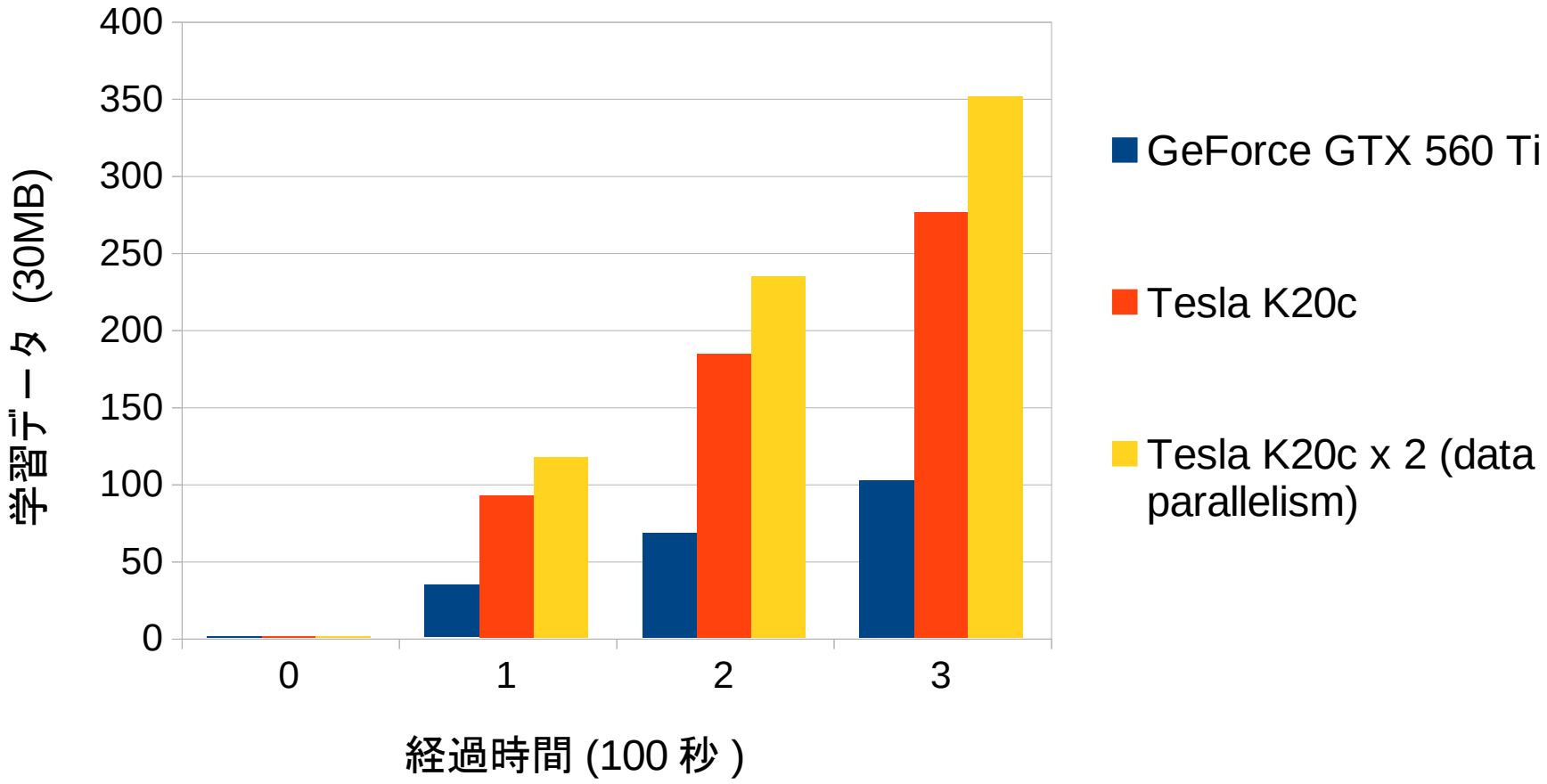
# ネットワーク構造 (CIFAR10)



# パラメータ数 (CIFAR10)

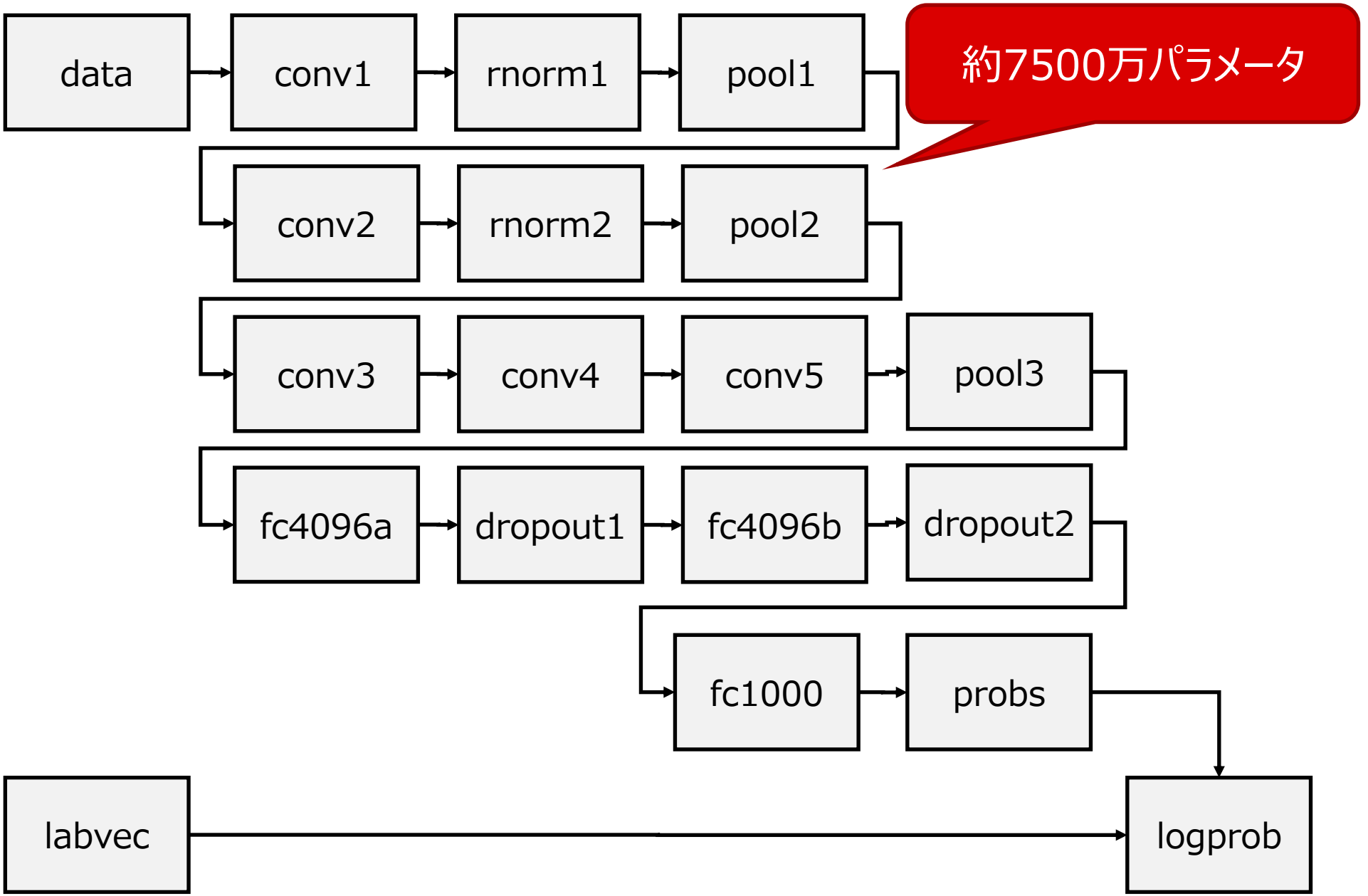
| | output | weight | input channel | input width | filter size | filter width | stride |
|---|---|---|---|---|---|---|---|
| data | 1728 | | 3 | 24(32) | | | |
| labels | 1 | | | | | | |
| conv1 | 36864 | 4800 | 3 | 24 | 64 | 5 | 1 |
| pool1 | 9216 | | 64 | 24 | | 3 | 2 |
| conv2 | 9216 | 36864 | 64 | 12 | 64 | 3 | 1 |
| pool2 | 2304 | | 64 | 12 | | 3 | 2 |
| local3 | 1152 | 663552 | 64 | 6 | 32 | 3 | 1 |
| local4 | 1152 | 331776 | 32 | 6 | 32 | 3 | 1 |
| fc10 | 10 | 11520 | 32 | 6 | | | |
| probs | 10 | | | | | | |
| logprob | | | | | | | |
| sum | 61653 | 1048512 | | | | | |

# 学習の速さ (CIFAR10)




**2GPUの使い方次第で同時に処理できるデータが増加**

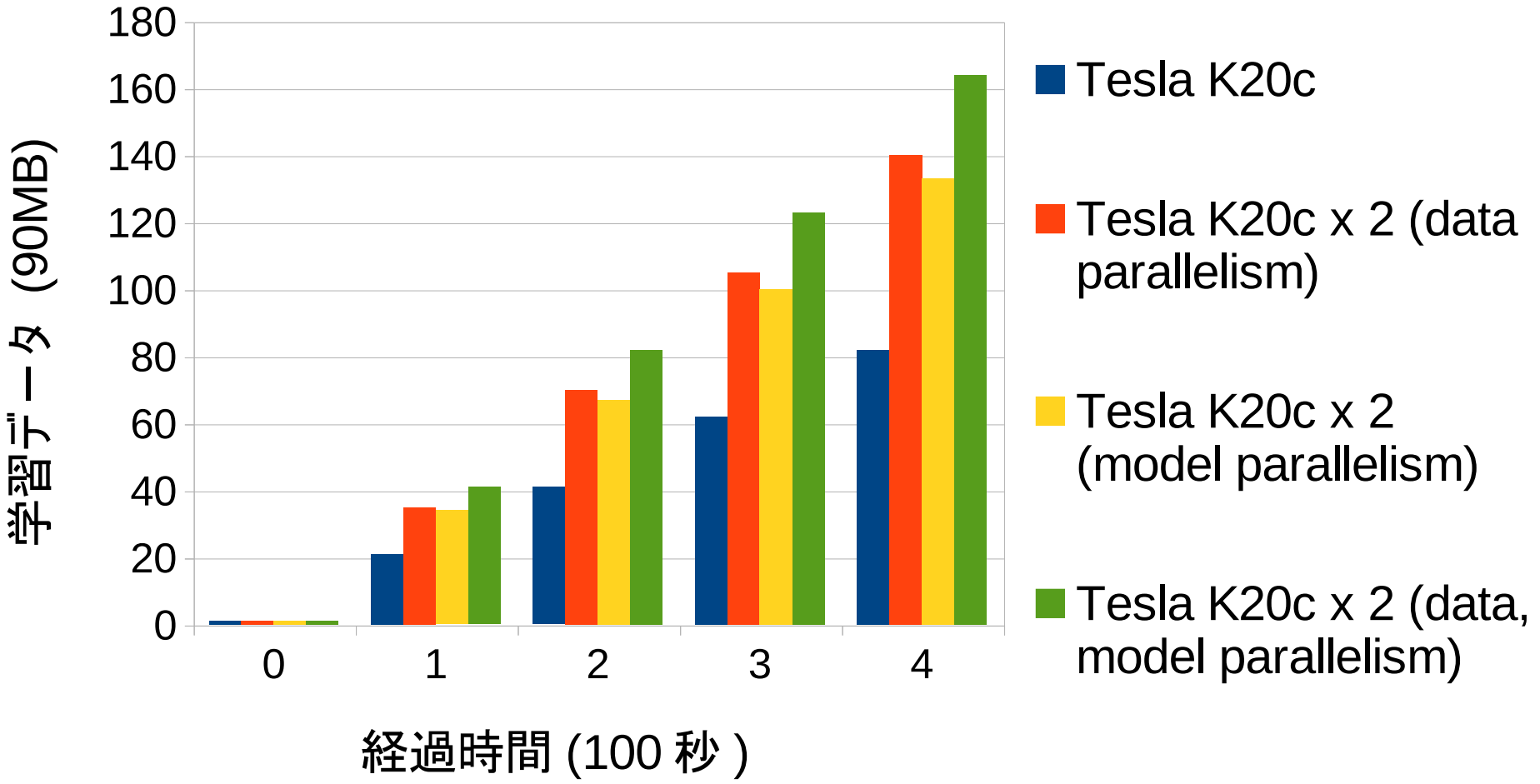
# ネットワーク構造 (ImageNet)



# パラメータ数 (ImageNet)

| | output | weight | input channel | input width | filter size | filter width | stride |
|---|---|---|---|---|---|---|---|
| data | 150528 | | 3 | 224(256) | | | |
| labvec | 1 | | | | | | |
| conv1 | 254016 | 23232 | 3 | 224 | 64 | 11 | 4 |
| rnorm1 | 254016 | | 64 | 63 | | 5 | |
| pool1 | 61504 | | 64 | 63 | | 3 | 2 |
| conv2 | 184512 | 307200 | 64 | 31 | 192 | 5 | 1 |
| rnorm2 | 184512 | | 192 | 31 | | 5 | |
| pool2 | 43200 | | 192 | 31 | | 3 | 2 |
| conv3 | 86400 | 663552 | 192 | 15 | 384 | 3 | 1 |
| conv4 | 57600 | 884736 | 384 | 15 | 256 | 3 | 1 |
| conv5 | 57600 | 589824 | 256 | 15 | 256 | 3 | 1 |
| pool3 | 12544 | | 256 | 15 | | 3 | 2 |
| fc4096a | 4096 | 51380224 | 256 | 7 | | | |
| dropout1 | 4096 | | | | | | |
| fc4096b | 4096 | 16777216 | | | | | |
| dropout2 | 4096 | | | | | | |
| fc1000 | 1000 | 4096000 | | | | | |
| probs | 1000 | | | | | | |
| logprob | | | | | | | |
| sum | 1364817 | 74721984 | | | | | |

# 学習の速さ (ImageNet)




**2GPUの使い方次第で同時に処理できるデータが増加**

# Reference


- Improving neural networks by preventing co-adaptation of feature detectors, G. E. Hinton et al., 2012
- One weird trick for parallelizing convolutional neural networks, A. Krizhevsky, 2014