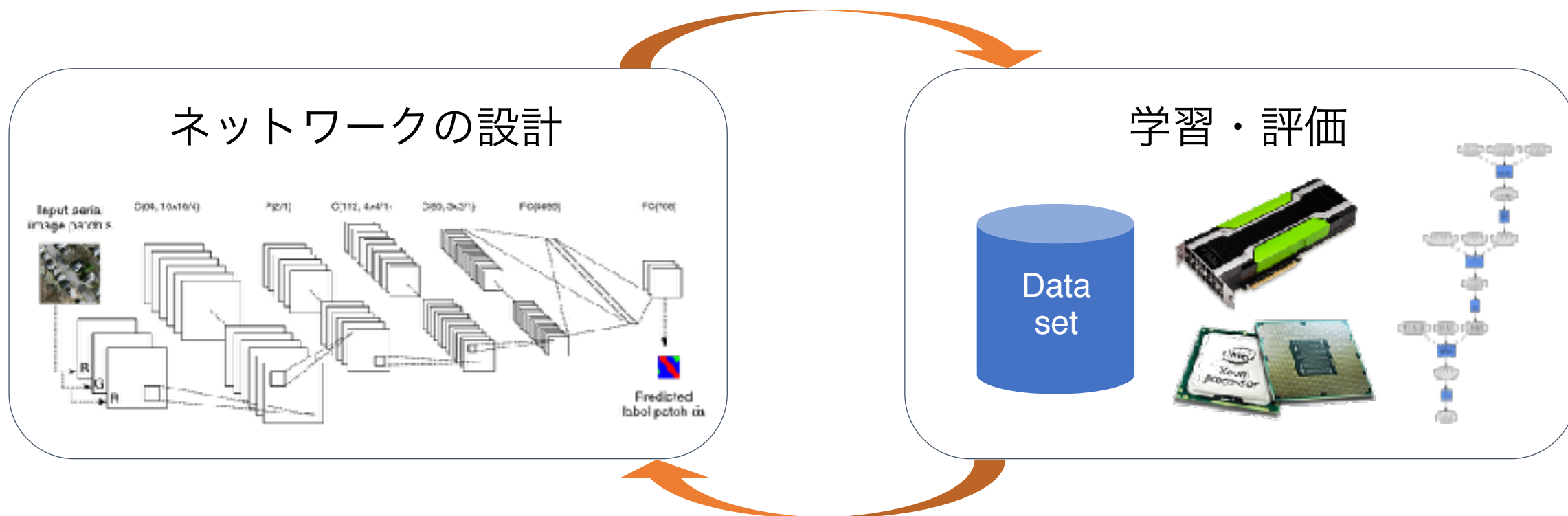


# 深層学習フレームワークChainer

- ニューラルネットワークの設計・学習・評価等、深層学習を用いた研究開発に必要な一連の機能を提供



# Chainer の特徴と機能

## Powerful

☑ CUDA

CUDAを用いたGPU計算のサポート

☑ cuDNN

cuDNNによる高速な学習／推論

☑ NCCL

NCCLを用いた高速なマルチGPU学習をサポート

## Flexible

☑ Convolutional Network

N次元の入力に対応したConvolution, Deconvolution, Pooling, BN, 等

☑ Recurrent Network

LSTM, Bi-directional LSTM, GRU, Bi-directional GRU, 等のRNNコンポーネント

☑ Many Other Components

ニューラルネットワークで使われる多くのレイヤ定義、各種ロス関数

☑ Various Optimizers

SGD, MomentumSGD, AdaGrad, RMSProp, Adam, 等の最適化手法が選択可能

## Intuitive

☑ Define-by-Run

複雑なネットワークの記述が容易

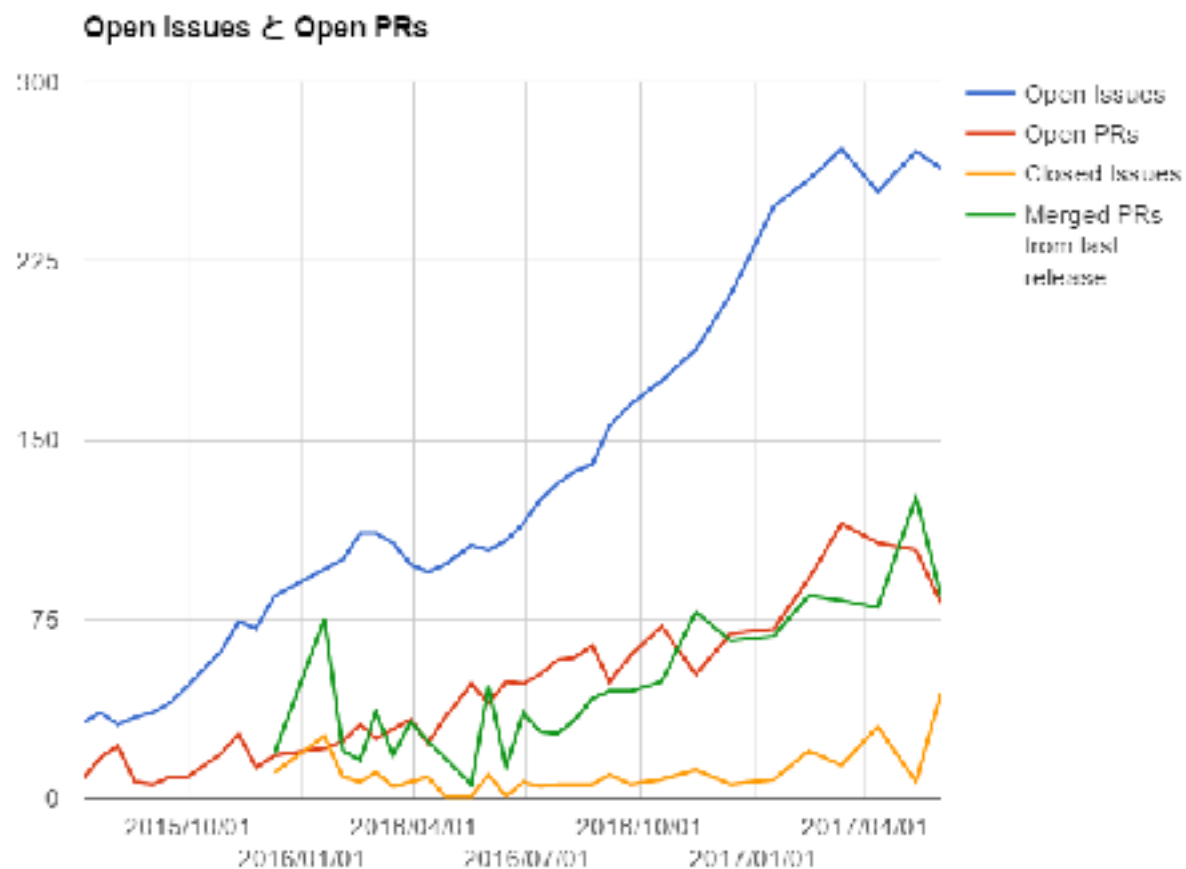
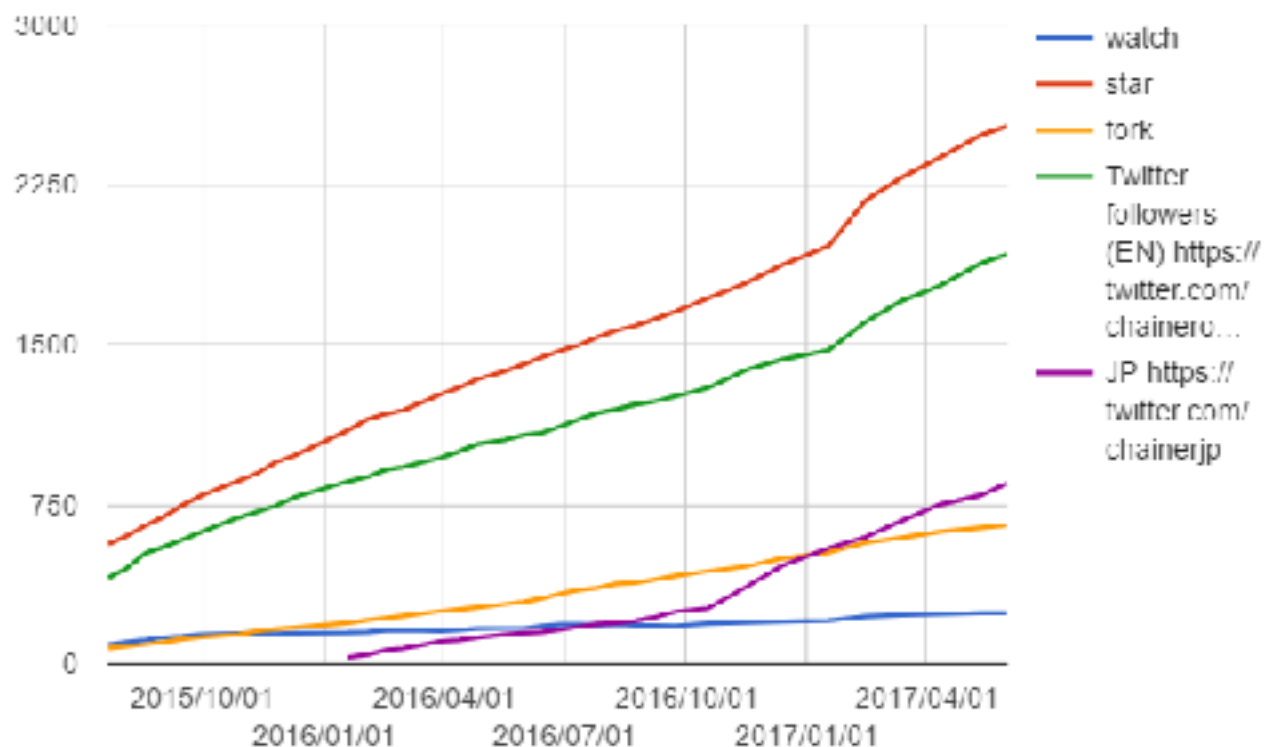
☑ High debuggability

Pythonライブラリであるためエラー箇所の特が容易：デバッグしやすい

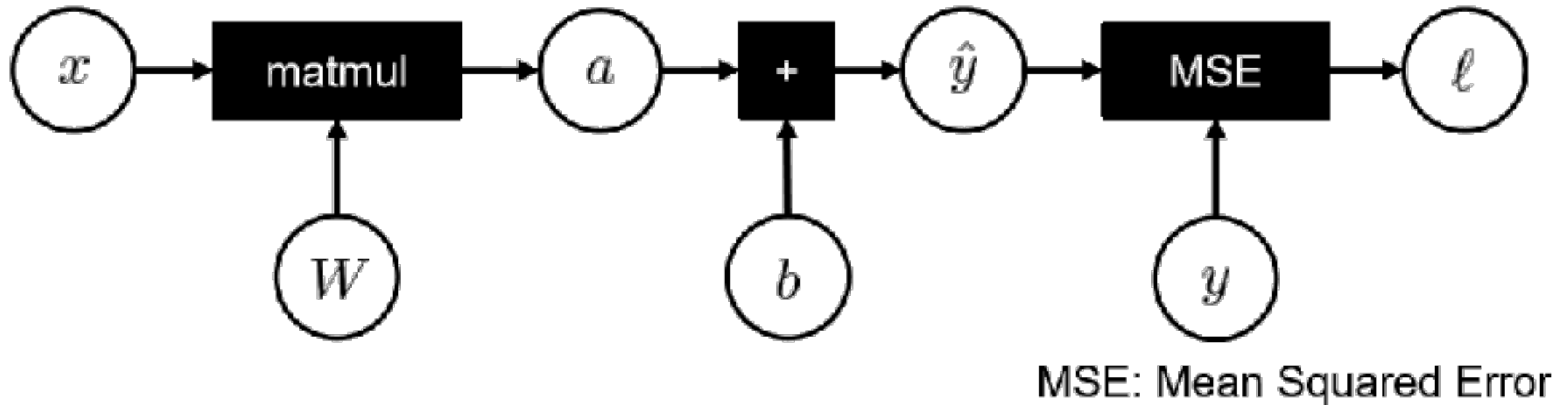
☑ Simple APIs

様々なNNの学習で共通する部分を抽象化、一連の学習フローを簡易に記述可

# ChainerのPopularity Growth



# ニューラルネットワーク = 計算グラフ



$$l = \text{MSE}(\text{matmul}(x, W) + b, y)$$

- 入力ベクトルに対して線形・非線形な関数を多数適用する**計算グラフ**だと捉えるとよい

# 計算グラフをどうやって扱うか

## 静的

計算グラフの定義と、  
定義に従って計算を  
実際に行うコードが  
別に存在

## 動的

実際に計算を行うコード  
自体が計算グラフ  
の定義として扱われ  
る

# Chainerでは？ → 動的

Chainerは“Define-**by**-Run”を採用した初めての深層学習フレームワーク\*

\* autogradがChainerに先駆けて存在したが、深層学習フレームワークとしてはChainerが初めてと思われる

- Define-**and**-Run（静的グラフ）

まず計算グラフを構築し、構築した計算グラフにデータを流すという、2ステップから成る（Caffe, theano, TensorFlowなど）

- Define-**by**-Run（動的グラフ）

通常の実行演算をする感覚で順伝播処理をすると同時に、逆伝播用の計算グラフが構築される（**Chainer**, DyNet, PyTorchなど）

# Define-and-Run と Define-by-Run

## Define-and-Run

# 構築

```
x = Variable('x')
y = Variable('y')
z = x + 2 * y
```

# 評価

```
for xi, yi in data:
    eval(z, (xi, yi))
```

## Define-by-Run

# 構築と評価が同時

```
for xi, yi in
data:
    x = Variable(xi)
    y = Variable(yi)
    z = x + 2 * y
```

データを見ながら  
違う処理をしてもよい

# Define-by-Runがもたらす柔軟性と直感性

- 「実際に行われたForward計算」がそのままネットワークの定義となること  
がもたらす高い自由度
  - データ次第でネットワーク構造を変化させることが容易
  - ネットワークを自体をプログラムで定義できる  
＝ネットワークの構造をデータではなく、プログラムとして扱える
- **Chainerの場合、その「Forward計算」をPythonで書ける**
  - Pythonの構文を用いてネットワーク構造が自由に書ける
  - ネットワークの計算の途中にprint文を挟むといった任意の処理を挿入するのがDefine-by-Runであれば容易（ネットワークをコンパイルしてしまう場合はこういったデバッグが難しい。）
  - 同じネットワークのコードを少ない変更でいろいろな目的に使い回しやすい（条件分岐を部分的に追加するだけで済む、など）
  - Pythonのデバッガを使ってネットワークの中間値などをチェックしたり、ネットワークの設計自体をチェックするなどが容易



# Chainer v2.0.1

大幅なメモリ消費量削減、ユーザフィードバックを反映しAPIを整理

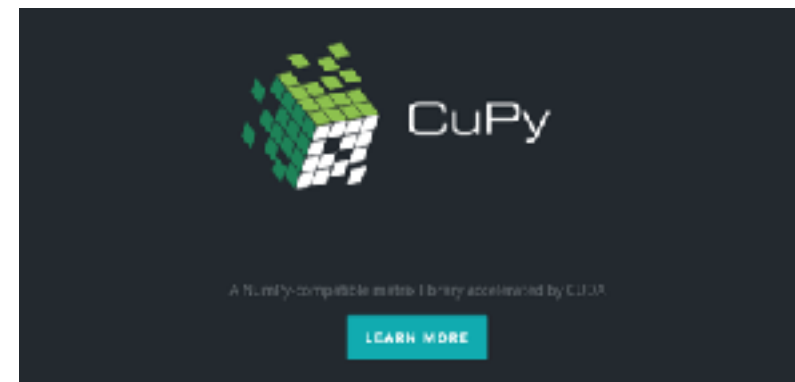


- Aggressive Buffer Release for 学習時メモリ使用量削減の効果→  
<https://chainer.org>



- GPUを使った配列演算をNumPyとほぼ同一のインターフェースでできるCuPyを独立したライブラリとしてリリース  
<https://cupy.chainer.org>

architecture	v1.22.0	v2-abr	ratio
nin	1233 MB	910 MB	74.6%
googlenetbn	3485 MB	2959 MB	84.9%
resnet50	5835 MB	3868 MB	66.3%



# CuPyの進化

---

ChainerにおけるGPU計算を全て担当するライブラリが独立

---

NumPy互換APIで低コストにCPUコードをGPUへ移行

---

特異値分解などの線形代数アルゴリズムをGPU実行

---

KMeans, Gaussian Mixture ModelなどのExampleの充実



<https://github.com/cupy/cupy>

```
import numpy as np
x = np.random.rand(10)
W = np.random.rand(10, 5)
y = np.dot(x, W)
```

GPU

```
import cupy as cp
x = cp.random.rand(10)
W = cp.random.rand(10, 5)
y = cp.dot(x, W)
```

# Chainerの追加パッケージ

分散深層学習・深層強化学習・コンピュータビジョン



分散学習

**MN**

ChainerMN: 分散深層学習用追加パッケージ

高いスケーラビリティ（128GPUで100倍の高速化）



強化学習

**RL**

ChainerRL: 深層強化学習ライブラリ

DQN, DDPG, A3C, ACER, NSQ, PCL, etc. OpenAI Gym サポート



画像認識

**CV**

ChainerCV: 画像認識アルゴリズム・データセットラッパーを提供

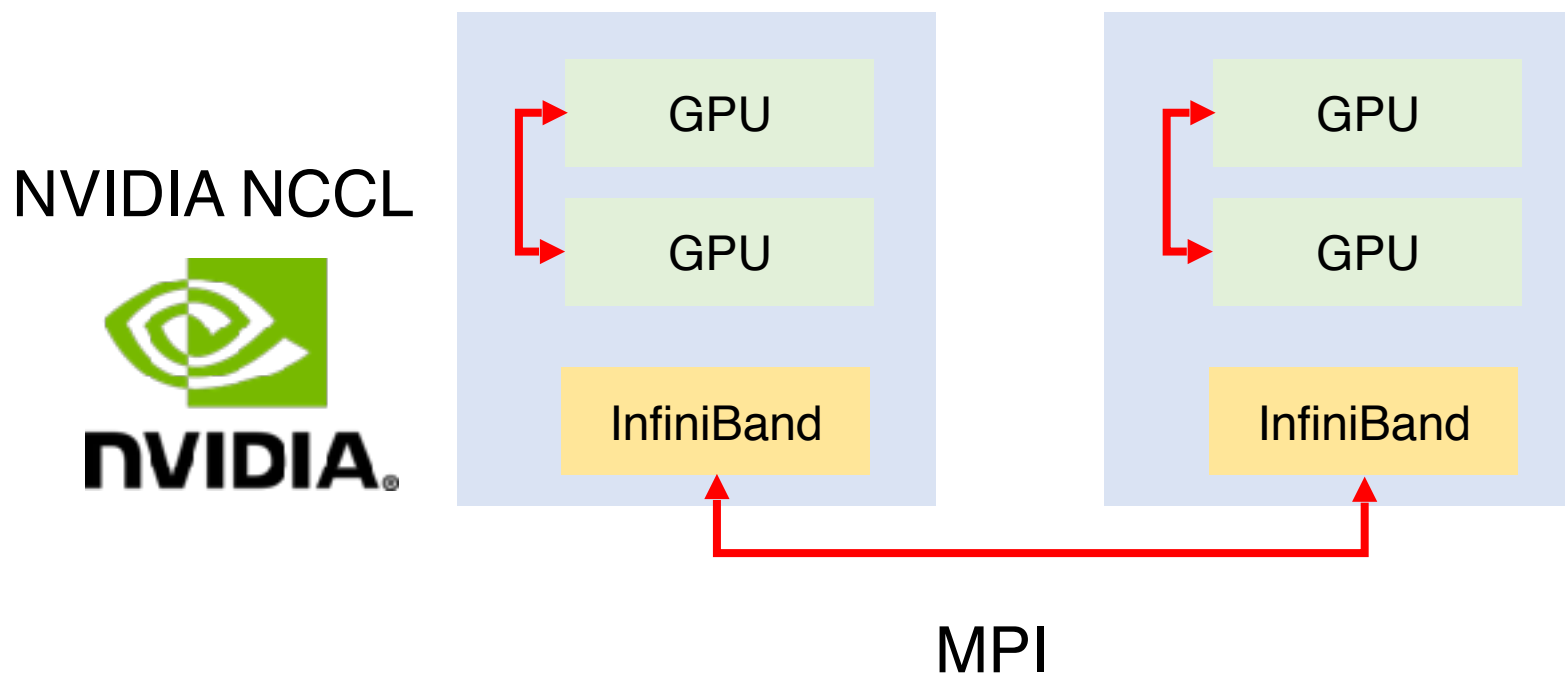
Faster R-CNN, Single Shot Multibox Detector (SSD), SegNet, etc.

ChainerMN

Chainer + **Multi-Node**

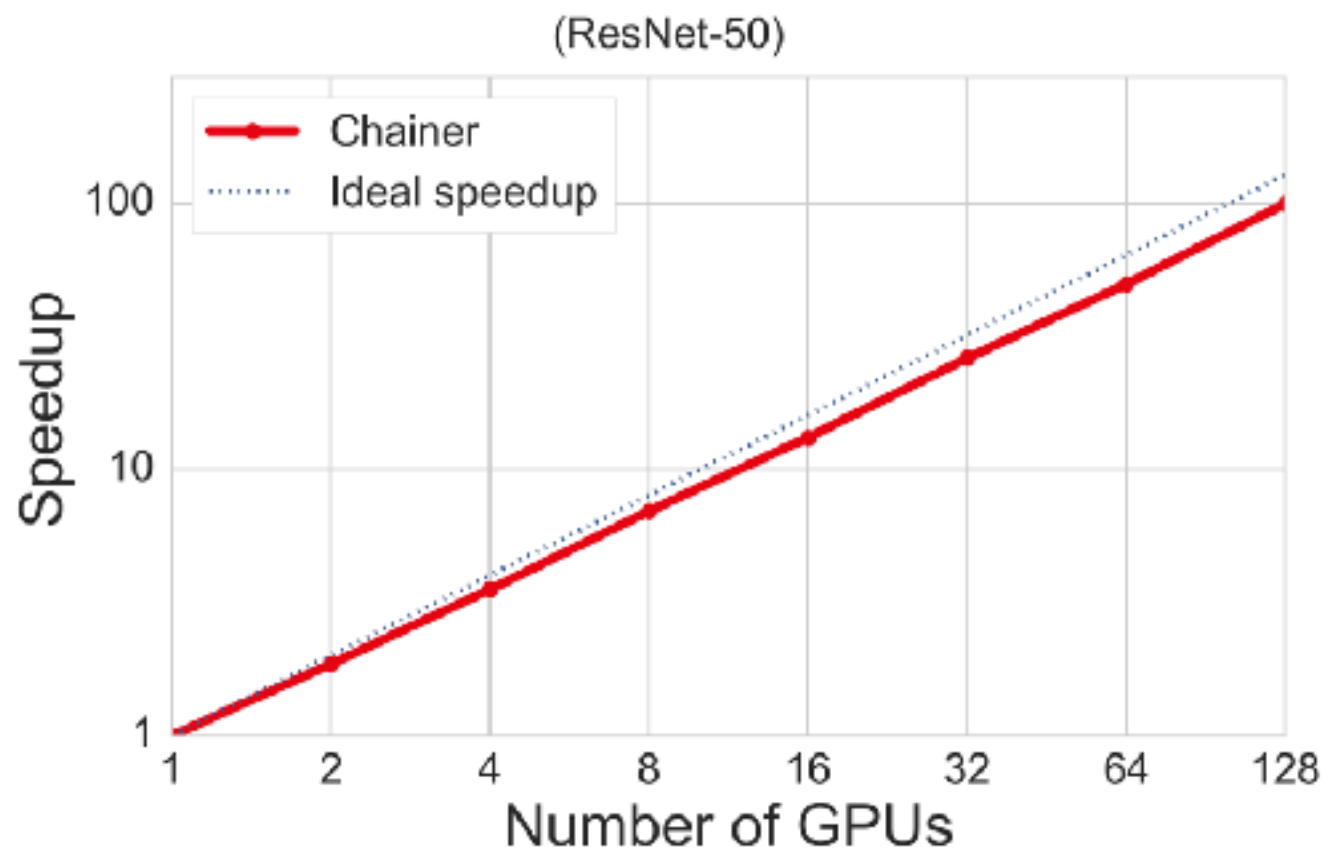
# ChainerMN: Multi-node

Chainerの使いやすさはそのままに、複数GPU、複数ノード安協で高速に学習することができる



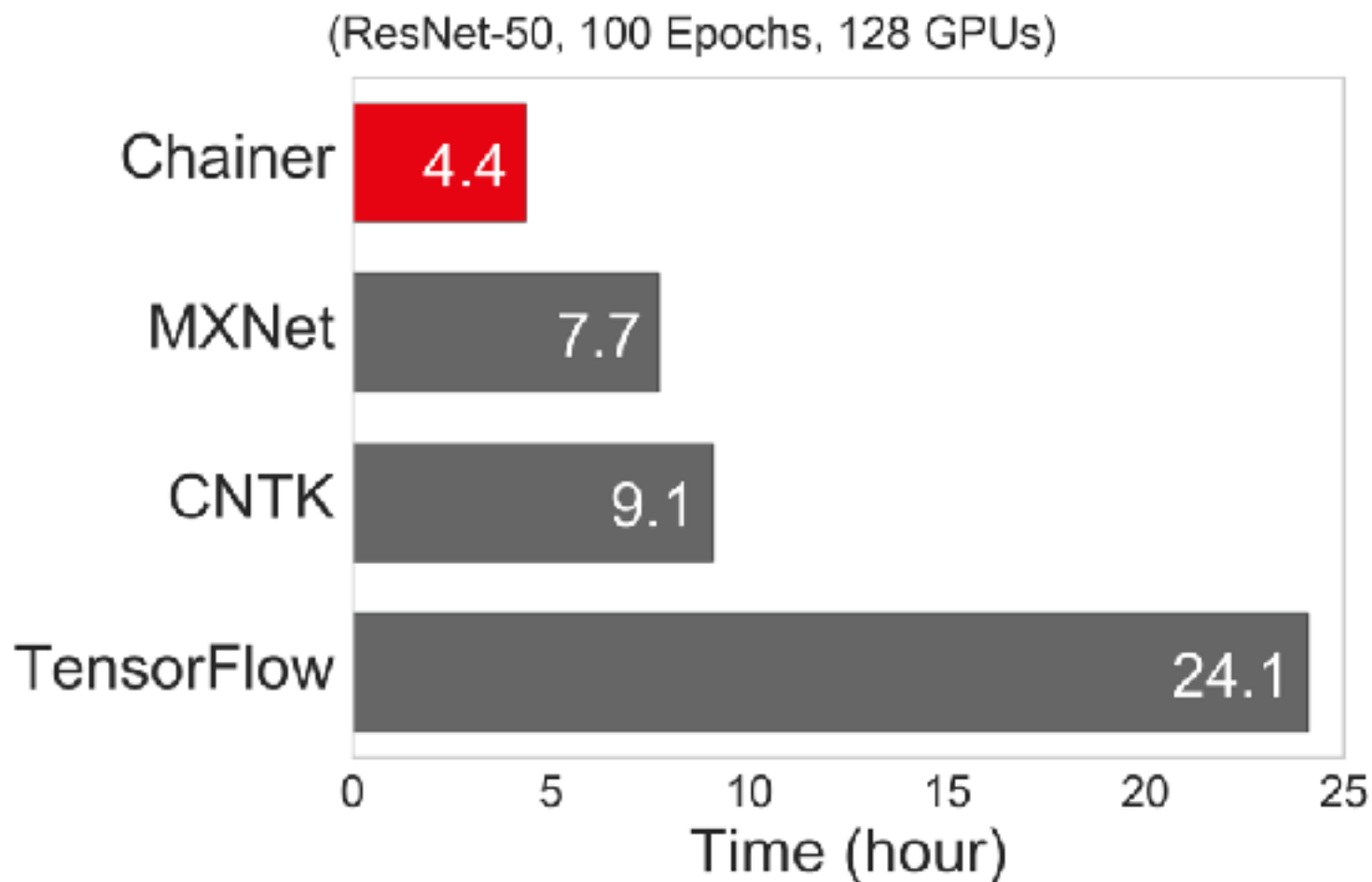
# ChainerMNによる分散深層学習

128GPUsを使っておよそ100倍の高速化に成功



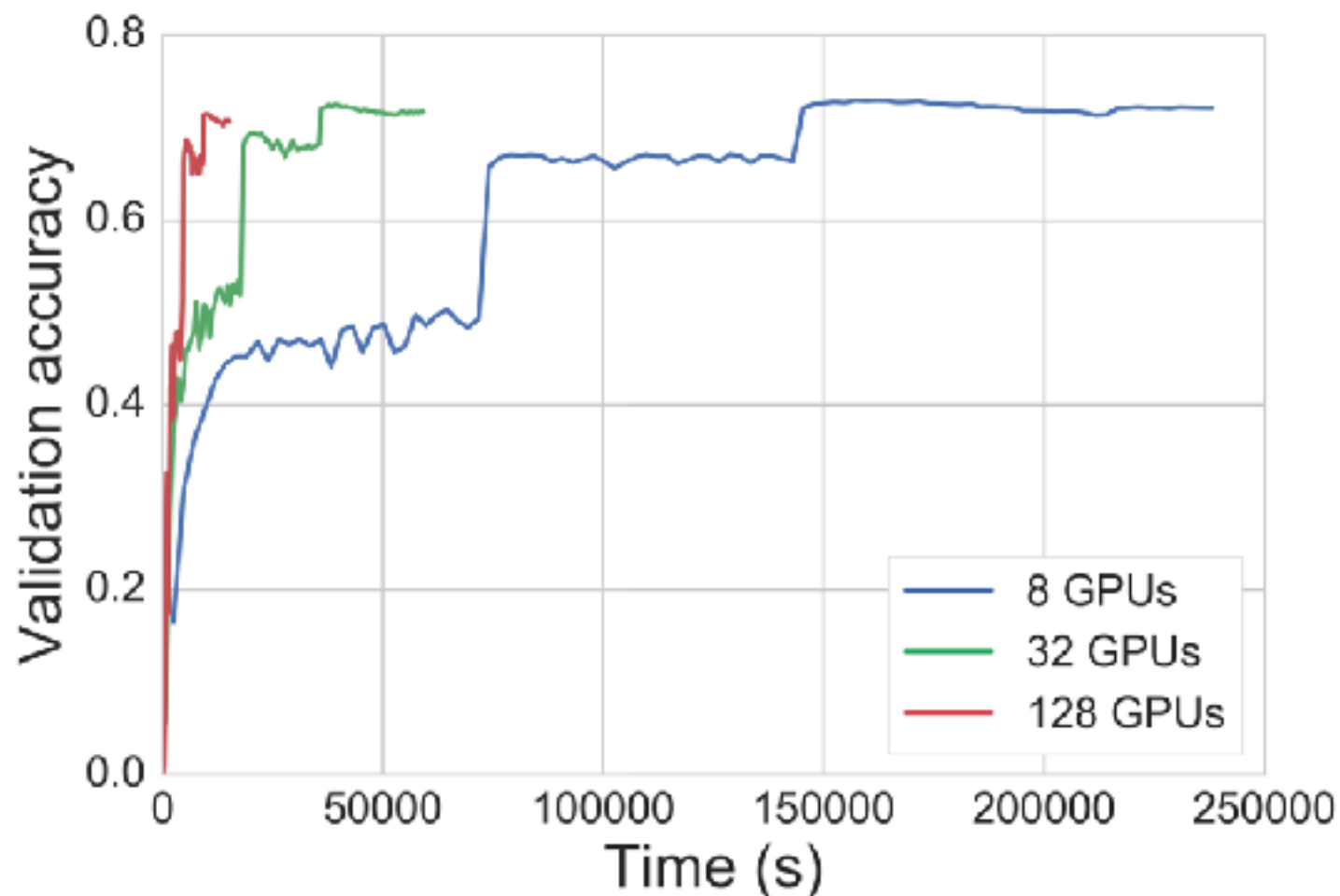
# 他フレームワークと比較しても高速

ImageNetデータセットにてResNet-50モデルを100エポック学習するのに要した時間による比較



# ChainerMNによる分散深層学習

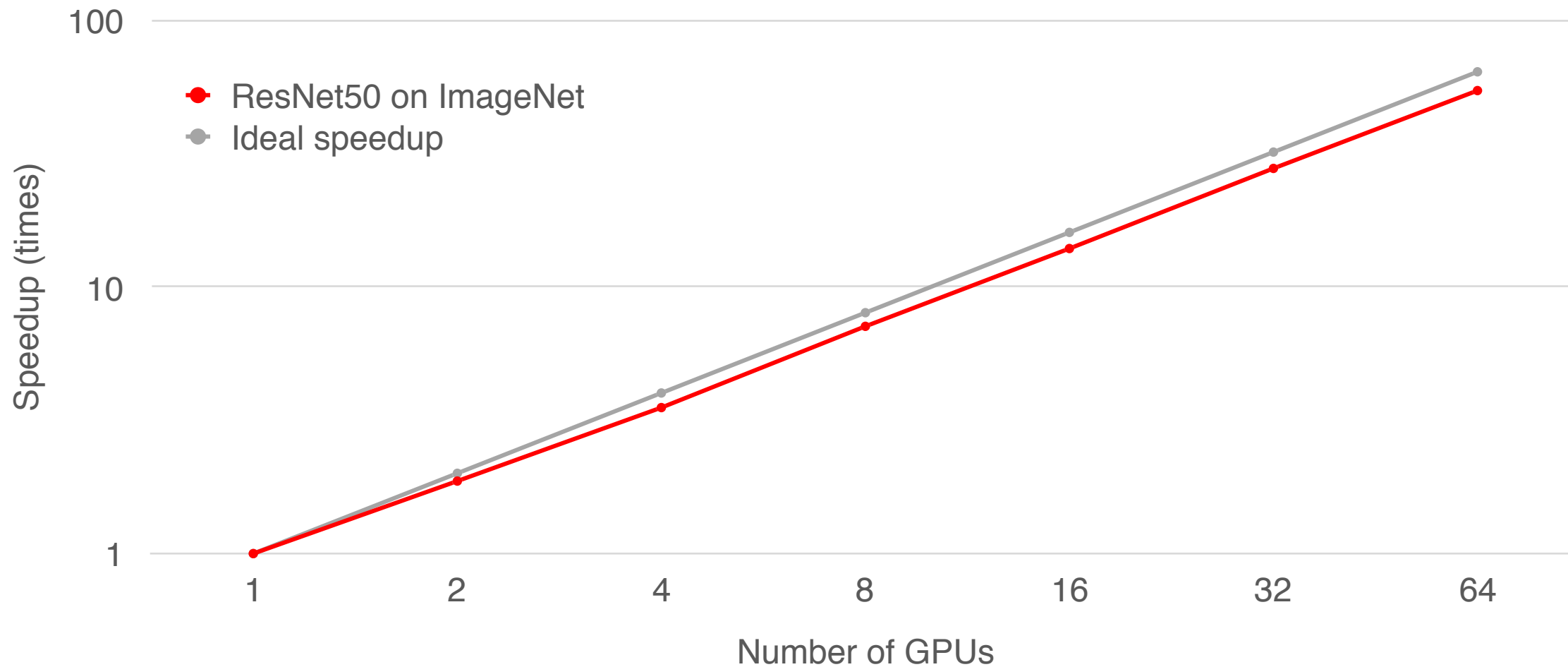
使用する分散ノード数（GPU数）を増やしていったとしても、同等の精度まで到達することを確認





# 高いスケールアウト性能をAzure上で実現

ChainerMN on Azure (K80, batchsize=32, InfiniBand)



# ChainerMNによる分散深層学習

既存のコードを多少書き換えるだけで利用できる（！）

```
optimizer =  
chainer.optimizers.MomentumSGD()
```



```
optimizer = chainermn.DistributedOptimizer(  
    chainer.optimizers.MomentumSGD())
```

# ChainerMN クラスタセットアップのための ARM (Azure Resource Manager) Template を公開予定

The screenshot displays the Azure portal interface for a Virtual Machine Scale Set named "Chainer00 - Scaling". The left-hand navigation pane includes a search bar and several menu items: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS, Instances, and Scaling (which is currently selected). The main content area features a "Save" button and a "Discard" button. Under the "SKU" section, the "Size" is set to "Standard NC24 (24 cores, 224 GB memory)" via a dropdown menu. The "Number of instances" is configured using a slider, with the value "32" displayed in a box on the right. The "Autoscaling" section shows the "Autoscaling" toggle switch is currently turned "Off".

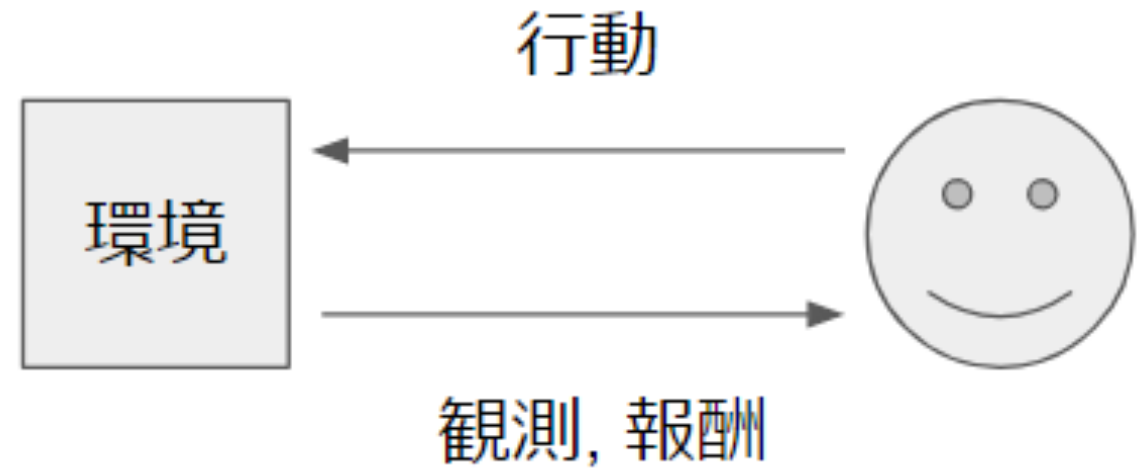
ChainerRL

Chainer + **R**einforcement **L**earning

# ChainerRL: 深層強化学習ライブラリ

## 強化学習？

エージェントが環境とのインタラクションを通じて報酬を最大化する行動を学習する



## ChainerRL

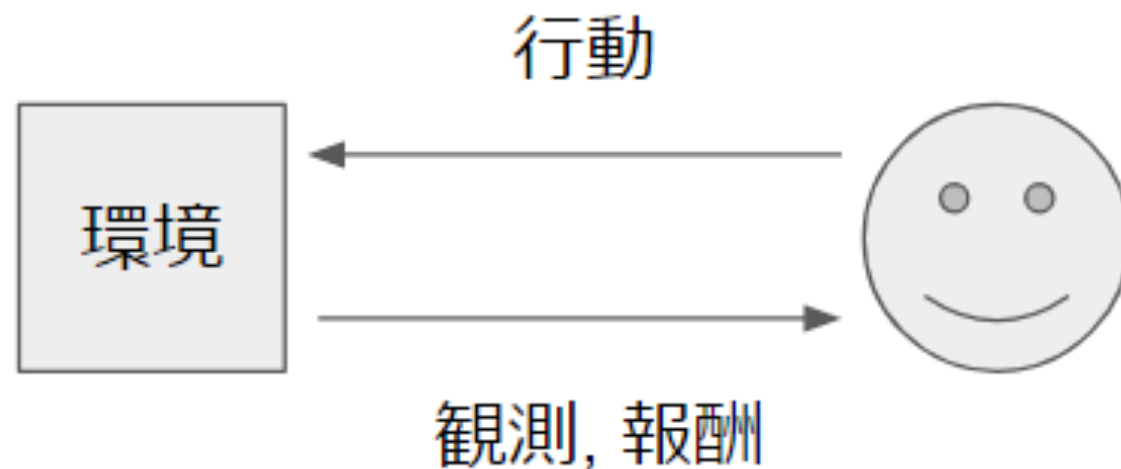
build passing coverage 71% docs latest pypi v0.1.0

ChainerRL is a deep reinforcement learning library that implements various state-of-the-art deep reinforcement algorithms in Python using [Chainer](#), a flexible deep learning framework.

# ChainerRLによる強化学習の流れ

## 1. 環境を構築

```
1 env = YourEnv()  
2 # reset は環境をリセットして現在の観測を返す  
3 obs = env.reset()  
4 action = 0  
5 # step は環境にアクションを送り, 4つの値(次の観測, 報酬, エピソード終端かどうか, 追加情報)を返す  
6 obs, r, done, info = env.step(action)
```



# ChainerRLによる強化学習の流れ

## 2. モデルを定義 (contd.)

Q-Function: 観測→各行動の価値 (将来の報酬の和の期待値)

```
1 class CustomDiscreteQFunction(chainer.Chain):
2     def __init__(self):
3         super().__init__(l1=L.Linear(100, 50)
4                               l2=L.Linear(50, 4))
5     def __call__(self, x, test=False):
6         h = F.relu(self.l1(x))
7         h = self.l2(h)
8         return chainerrl.action_value.DiscreteActionValue(h)
```

ActionValue: Discrete, Quadratic



# ChainerRLによる強化学習の流れ

## 2. モデルを定義

Policy: 観測→行動の確率分布

```
10 class CustomGaussianPolicy(chainer.Chain):
11     def __init__(self):
12         super().__init__(l1=L.Linear(100, 50)
13                               mean=L.Linear(50, 4),
14                               var=L.Linear(50, 4))
15     def __call__(self, x, test=False):
16         h = F.relu(self.l1(x))
17         mean = self.mean(h)
18         var = self.var(h)
19         return chainerrl.distribution.GaussianDistribution(mean, var)
```

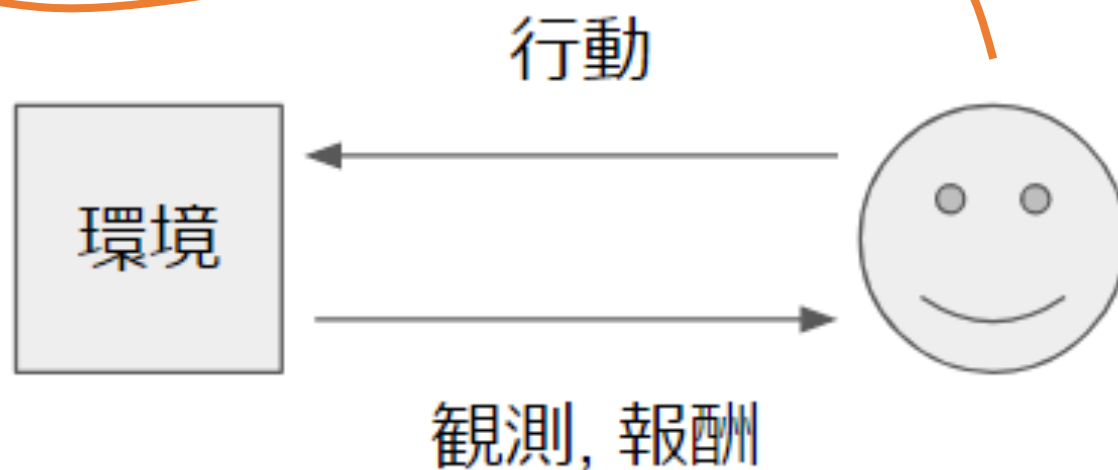
 Distribution: Softmax, Mellowmax, Gaussian,...



# ChainerRLによる強化学習の流れ

## 3. エージェントを定義

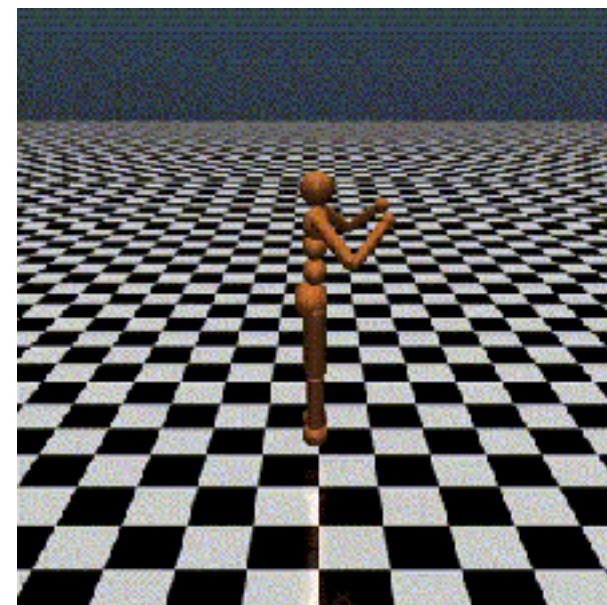
```
1 q_func = CustomDiscreteQFunction()  
2 optimizer = chainer.Adam()  
3 optimizer.setup(q_func)  
4 agent = chainerrl.agents.DQN(q_func, optimizer, ...) # 残りの引数は省略
```



# ChainerRLによる強化学習の流れ

## 4. インタラクションさせる

```
1  # Training
2  obs = env.reset()
3  r = 0
4  done = False
5  for _ in range(10000):
6      while not done:
7          action = agent.act_and_train(obs, r)
8          obs, r, done, info = env.step(action)
9          agent.stop_episode_and_train(obs, r, done)
10     obs = env.reset()
11     r = 0
12     done = False
13 agent.save('final_agent')
```



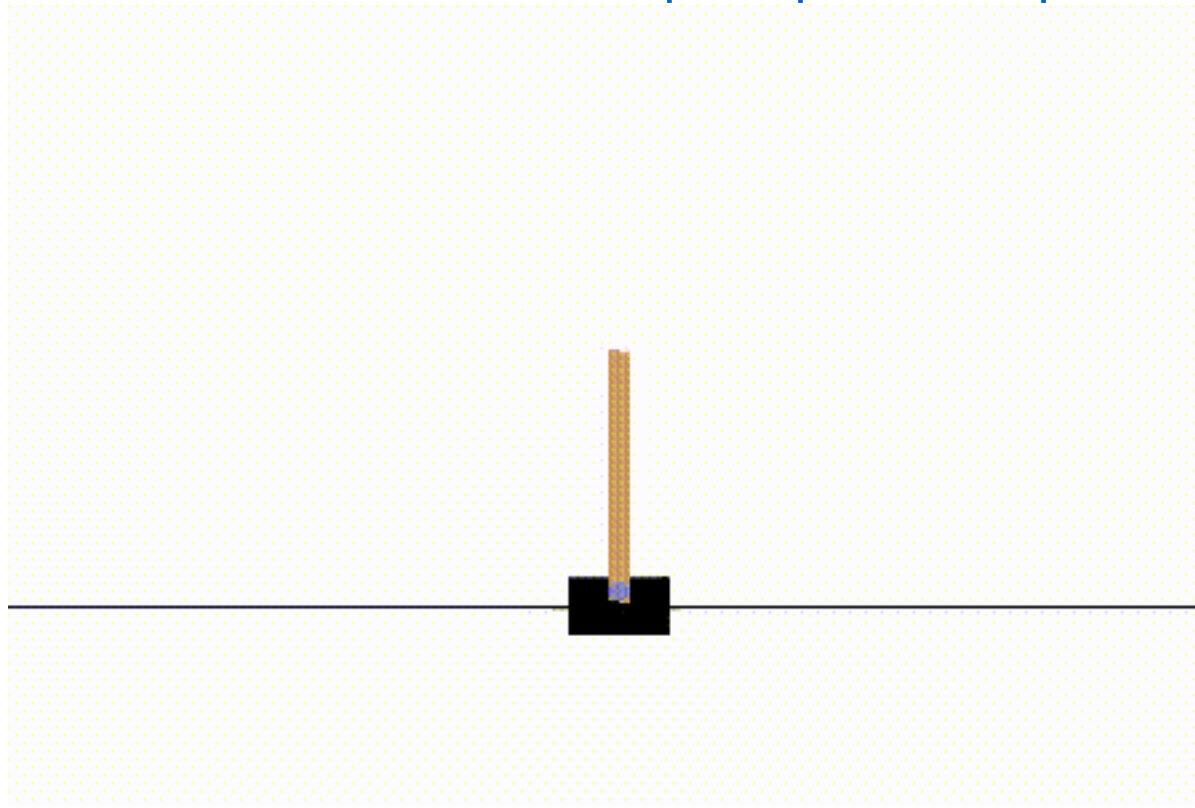
# 実装済みのアルゴリズム

- Deep Q-Network ([Mnih et al., 2015](#))
- Double DQN ([Hasselt et al., 2016](#))
- Normalized Advantage Function ([Gu et al., 2016](#))
- (Persistent) Advantage Learning ([Bellemare et al., 2016](#))
- Deep Deterministic Policy Gradient ([Lillicrap et al., 2016](#))
- SVG(0) ([Heese et al., 2015](#))
- Asynchronous Advantage Actor-Critic ([Mnih et al., 2016](#))
- Asynchronous N-step Q-learning ([Mnih et al., 2016](#))
- Actor-Critic with Experience Replay ([Wang et al., 2017](#)) <- NEW!
- Path Consistency Learning ([Nachum et al., 2017](#)) <- NEW!
- etc.

# ChainerRL Quickstart Guide

- Jupyter notebookでQ-functionを定義し、Double DQNでCart Pole Balancingを学習

<https://github.com/pfnet/chainerrl/blob/master/examples/quickstart/quickstart.ipynb>



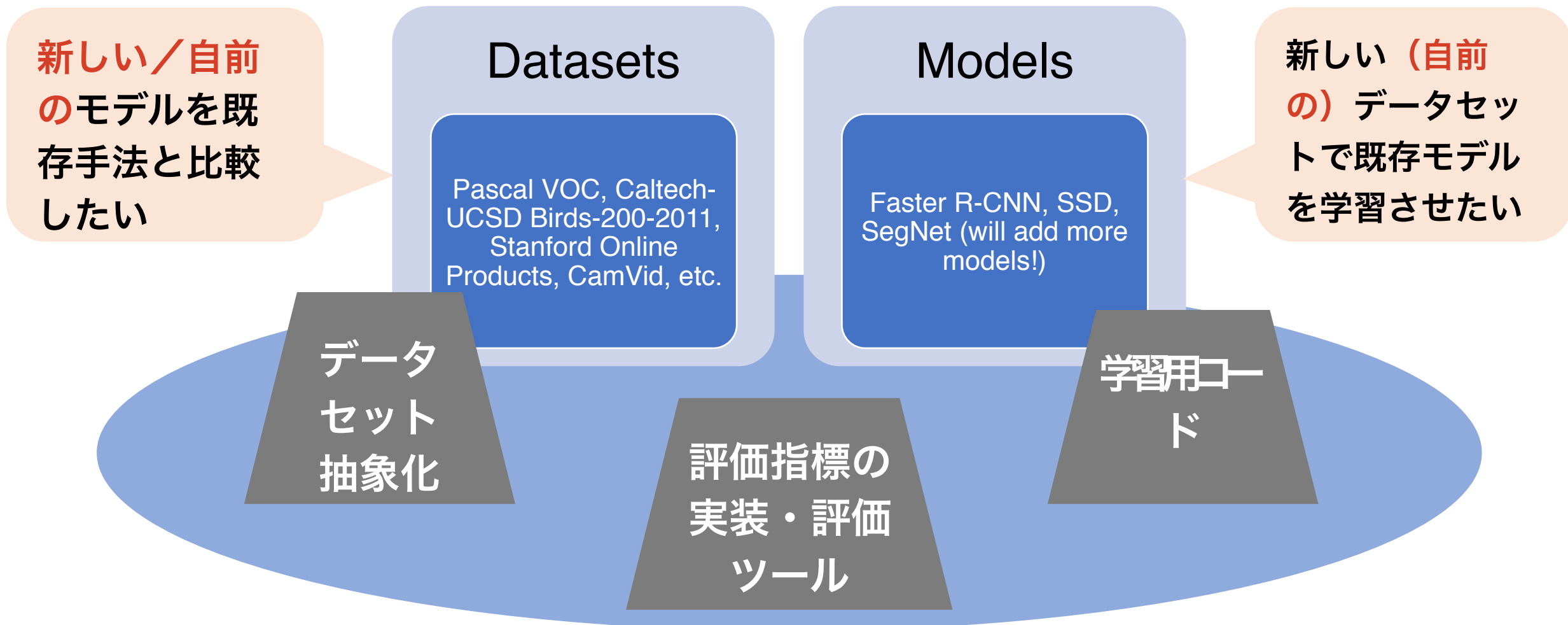
ChainerCV

**Chainer + Computer Vision**

# ChainerCV

<https://github.com/pfnet/chainercv>

深層学習を応用した画像認識の研究・開発をもっと簡単に



# ChainerCV

深層学習を応用した画像認識の研究・開発をもっと簡単に

## 最新のアルゴリズムを自前のデータで

- セグメンテーションのアルゴリズム（SegNet, ...）や、実装が複雑な物体検出のアルゴリズム（Faster R-CNN, SSD, ...）のモデル定義と、学習用コード、推論用コードの提供

## 再現性の確認を重視した安心の実装

- 学習用コードは、きちんと文献値の再現が行えることがチェックできなければマージしない

<> Code

🔔 Issues 12

🔗 Pull requests 8

📁 Projects 0

📖 Wiki

<https://github.com/pfnet/chainercv>

Set of tools for Computer Vision using Chainer

chainer

computer-vision

neural-network

deep-learning

python

numpy

cupy

gpu

cuda

Manage topics

Edit

# ChainerCV

深層学習を応用した画像認識の研究・開発をもっと簡単に

## 訓練済みの重みをすぐに利用可能

- 公開データセットを用い、文献値の再現が確認されたChainerによるモデルの学習コードを用いて得られた訓練済みパラメータも同時に配布

## オリジナルのモデルを既存のデータセットで

- Chainerで実装された新しいモデルの検証を行うために、既存の公開データセットをすぐに使い始めることができます。

<> Code

🔔 Issues 12

🔗 Pull requests 8

📁 Projects 0

📖 Wiki

<https://github.com/pfnet/chainercv>

Set of tools for Computer Vision using Chainer

chainer

computer-vision

neural-network

deep-learning

python

numpy

cupy

gpu

cuda

Manage topics

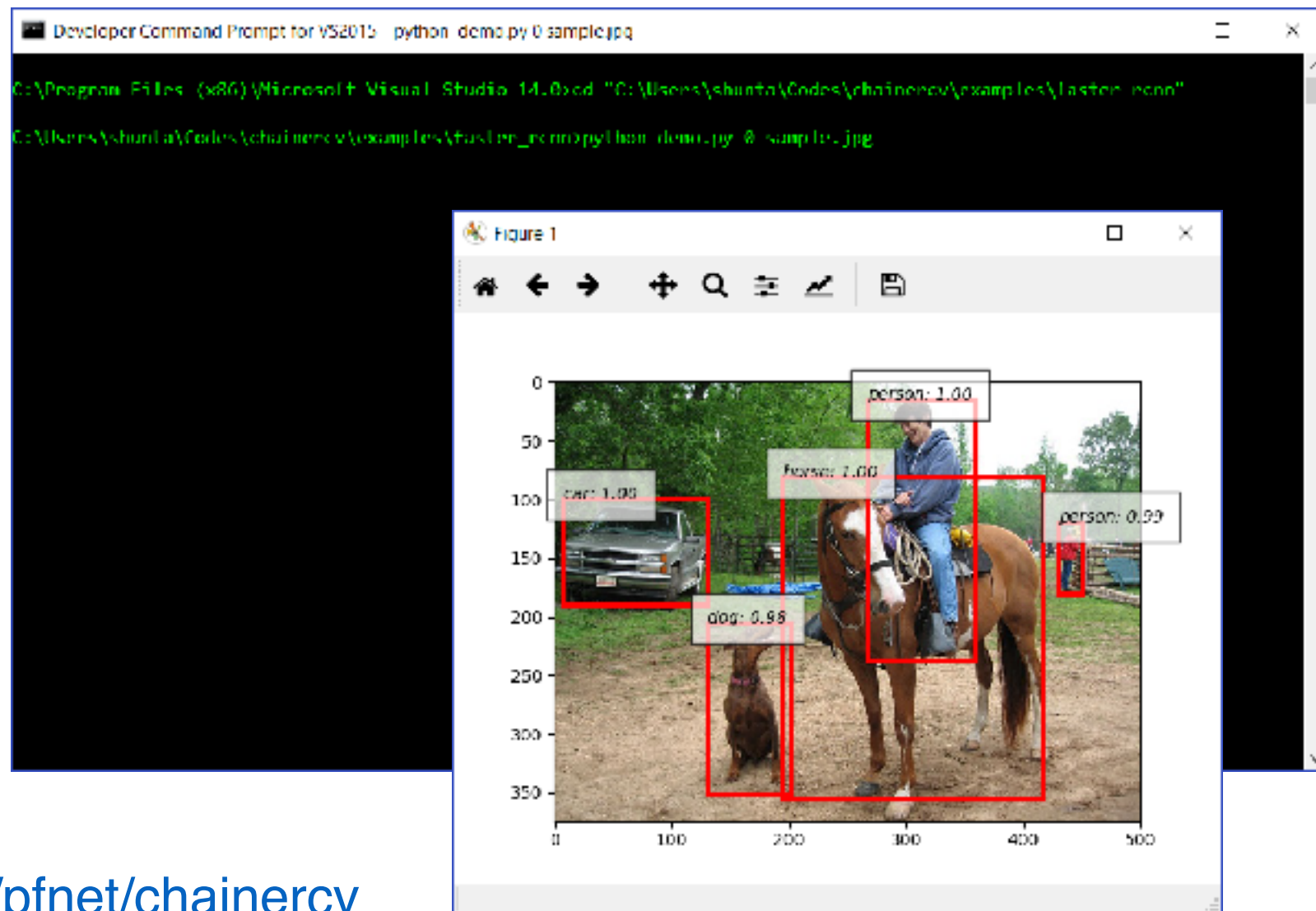
Edit



# ChainerCV

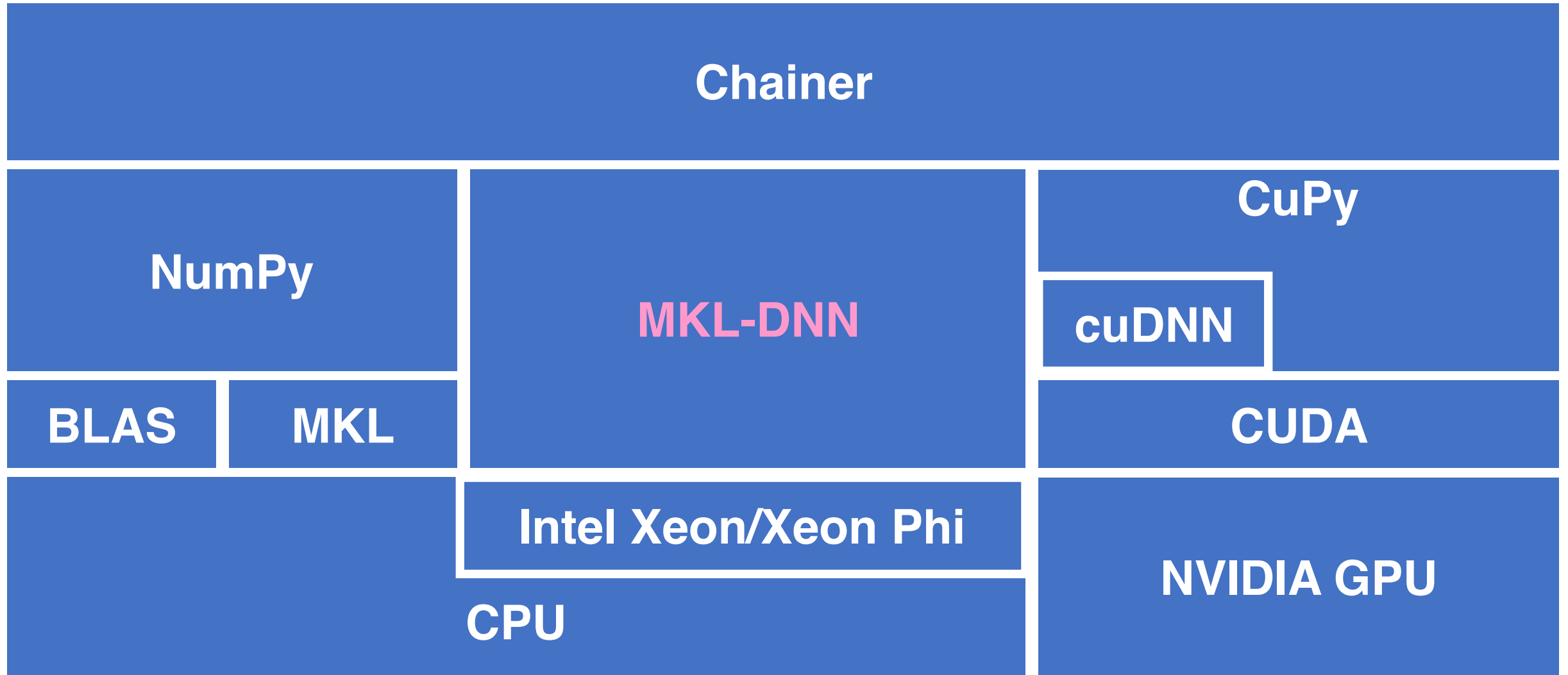
Chainerインストール後、**pip install chainervcv** するだけ

- Githubリポジトリ（pfnet/  
chainervcv）にある  
examplesから、  
faster\_rcnnの推論コード  
を実行（画像の準備が別  
途必要）
- 学習済み重みがない場合  
は、自動的にダウンロード  
が始まるので、事前に  
準備することが非常に少  
ない <https://github.com/pfnet/chainervcv>



Intel Chainer

# Intel Chainer with MKL-DNN Backend



# Intel Chainer with MKL-DNN Backend

## MKL-DNN


- Intelアーキテクチャ用に最適化されたDNNプリミティブライブラリ（立ち位置としてはNVIDIAにおけるcuDNN）
- サポートするCPU:
  - ✓ Intel Atom(R) processor with Intel(R) SSE4.1 support
  - ✓ 4th, 5th, 6th and 7th generation Intel(R) Core processor
  - ✓ Intel(R) Xeon(R) processor E5 v3 family (code named Haswell)
  - ✓ Intel(R) Xeon(R) processor E5 v4 family (code named Broadwell)
  - ✓ Intel(R) Xeon(R) Platinum processor family (code name Skylake)
  - ✓ Intel(R) Xeon Phi(TM) product family x200 (code named Knights Landing)
  - ✓ Future Intel(R) Xeon Phi(TM) processor (code named Knights Mill)
- 上記のCPU上でのニューラルネットワークの計算を大幅に高速化する

# Intel Chainer with MKL-DNN Backend


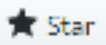

## convnet-benchmarks\*による速度比較








	Intel Chainer	Chainer with NumPy (MKL-Build)
Alexnet Forward	429.16 ms	5041.91 ms
Alexnet Backward	841.73 ms	5569.49 ms
Alexnet Total	1270.89 ms	10611.40 ms


およそ8.35倍の高速化！

 **mitmul / convnet-benchmarks**  
forked from soumith/convnet\_benchmarks

\*ChainerV2用対応したもの

 Unwatch ▾ 1  Star 1  Fork 441

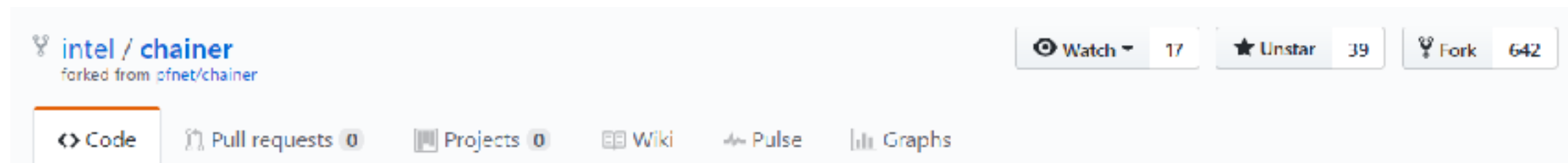
 Code  Pull requests 0  Projects 0  Wiki  Pulse  Graphs  Settings

Easy benchmarking of all publicly accessible implementations of convnets  Edit

# Intel Chainer with MKL-DNN Backend

ChainerV2のフォークとしてIntelにより活発に開発が続けられている

<https://github.com/intel/chainer>



The screenshot shows the GitHub repository page for 'intel / chainer', which is a fork of 'ofnet/chainer'. The repository has 17 watchers, 39 unstars, and 642 forks. The main navigation bar includes links for Code, Pull requests (0), Projects (0), Wiki, Pulse, and Graphs.

A flexible framework of neural networks for deep learning <http://chainer.org>

README.md

pyPI v2.0.0b1 license Other build passing coverage 87% docs stable

## Chainer: a neural network framework

## Intel® Software Optimization

This is a fast implementation of integration Chainer with Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN). It accelerates Chainer on CPU, esp. Intel® Xeon® and Intel® Xeon Phi™ processors.

Applications using Chainer

# Object Detection





# Semantic Segmentation



self, etc.	dynamic	ground	road	sidewalk
parking	rail track	building	wall	fence
guard rail	bridge	tunnel	pole	polegroup
traffic light	traffic sign	vegetation	terrain	sky
person	rider	car	truck	bus
caravan	trailer	train	motorcycle	bicycle



# Ponanza Chainer

- 世界コンピュータ将棋選手権に出場、2位
- Ponanza（世界コンピュータ将棋選手権2連覇（2015, 2016））ベース
- Ponanzaが探索を行う指し手のオーダリングにDeep Learningを応用した技術を使用し、対Ponanzaで8割以上の勝率

Team PFN

Team Ponanza



Issei Yamamoto



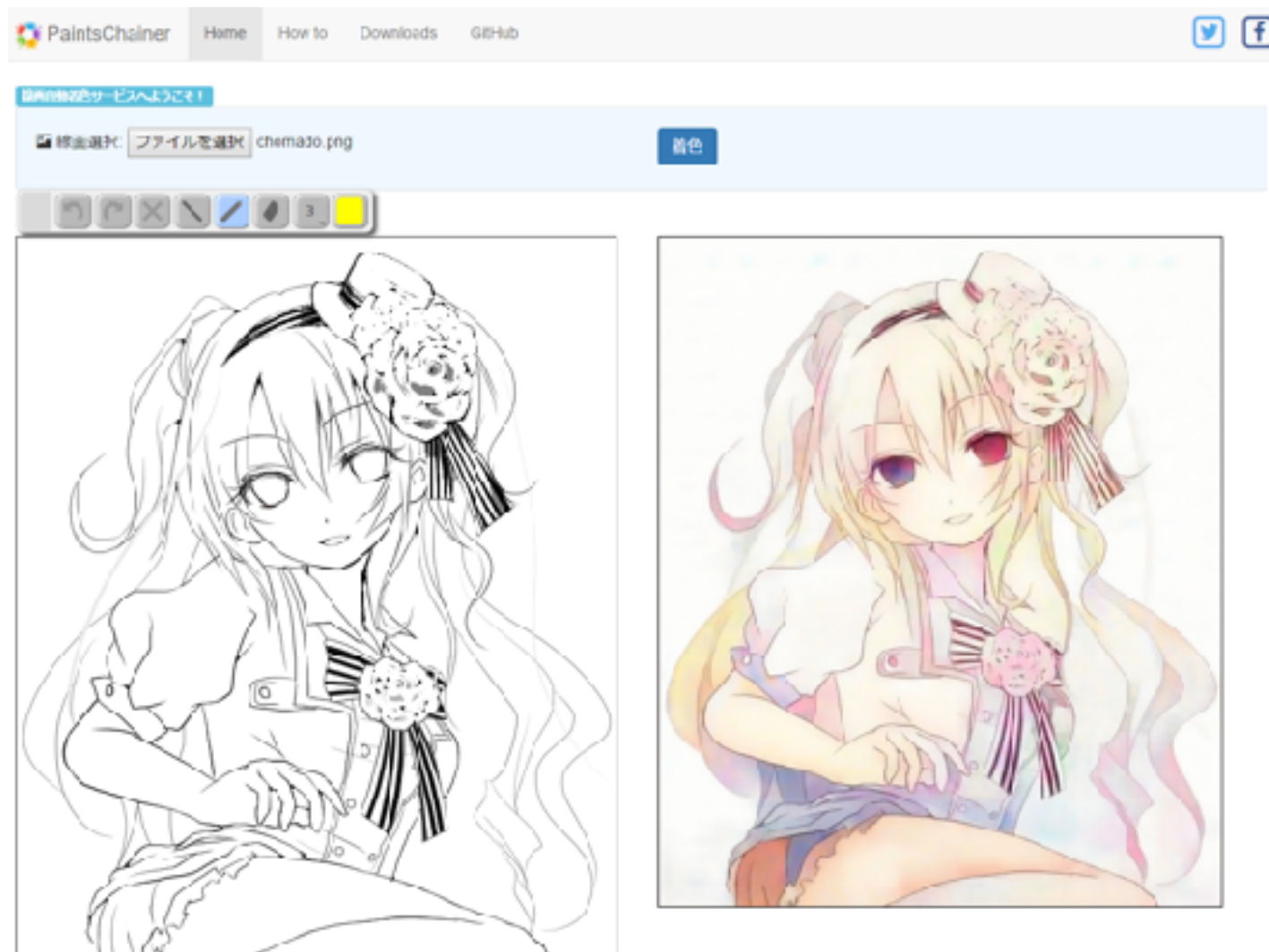
Akira Shimoyama



# Paints Chainer

<https://paintschainer.preferred.tech>

- 線画の自動着色を行うWebサービス
- 線画とその着色済み画像のペアを大量に用意
- 線画+ヒントを入力にして、着色後画像を出力する畳み込みネットワークを訓練
- ただそれだけでは塗りにバリエーションが出ない...
- Adversarial lossを加えることでより自然で多様な塗りを実現



線画：ちょまどさん

Companies supporting Chainer

# Chainerの開発に加わっている企業



Microsoft



Contributing to Chainer

# Chainerはオープンソースです。

- <https://github.com/pfnet/chainer> からPRを送ることができます。
- さらに加速している深層学習研究の発展スピードに合わせて、Chainerも開発方針等の更新を行います。
  - リリースサイクルの変更
  - 新しい機能を優先的に取り入れる方針に
  - とにかく最先端の研究成果を素早く取り入れられることを重視

Date	ver 2	ver 3	ver 4
1 month later	v2.0.2	v3.0.0b1	
2 months later	v2.0.3	v3.0.0rc1	
3 months later		v3.0.0	v4.0.0a1