NVIDIA DIGITS による ディープラーニング画像分類

森野慎也
シニアソリューションアーキテクト、GPU-Computing
エヌビディア

本ラボの目的

- ディープラーニングのイントロダクション
- ネットワークのトレーニング
- トレーニングの結果を理解する。

- コンピュータビジョン・画像分類に関するハンズオン
- CaffeとDIGITSを使用する

本ラボが意図しないこと

- 機械学習に対する基本的なイントロダクション
- ニューラルネットワークに関する正確な定式化
- Caffe、DIGITSや他のツールに関する、すべての機能の紹介
- 最先端のディープラーニング企業になるためのノウハウ

前提知識

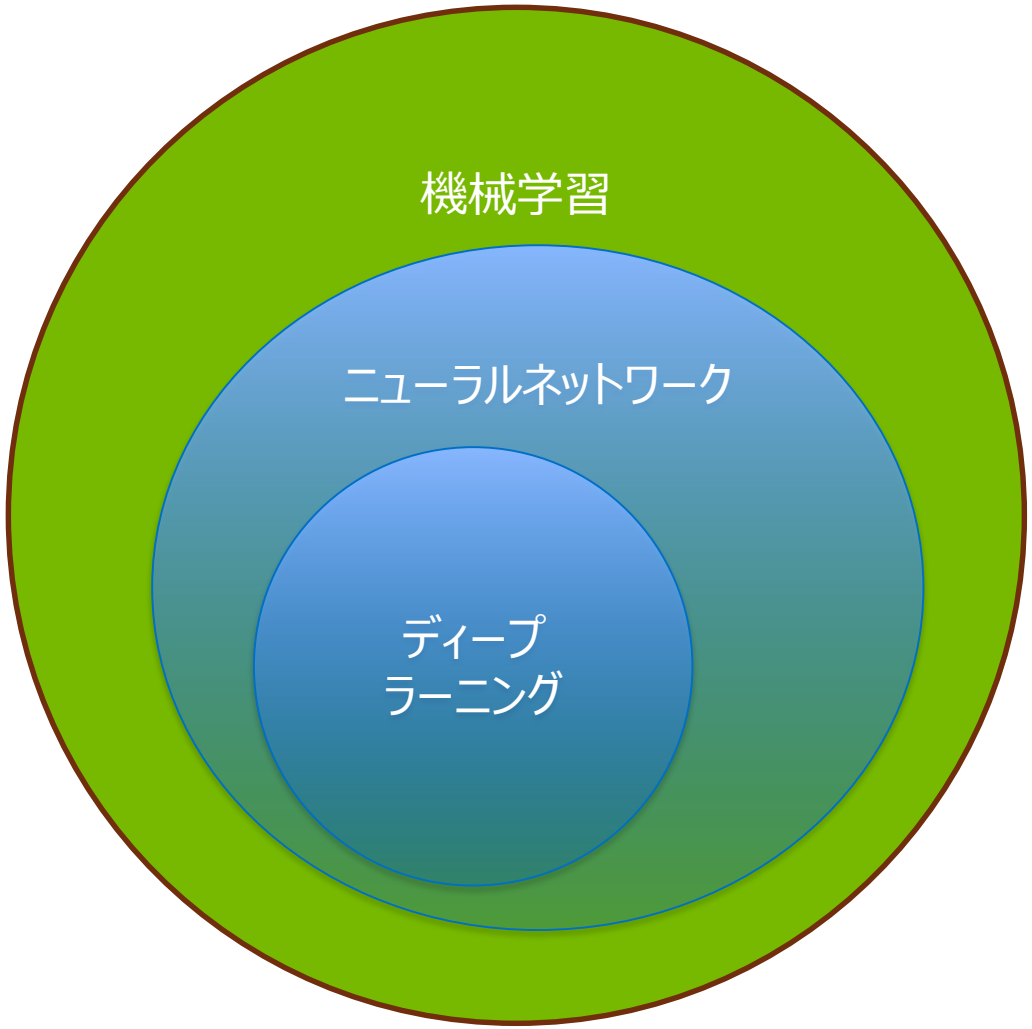
- ディープラーニングに関するバックグラウンドは不要です。

- ウエブブラウザを使用
  - URLにアクセス
  - ファイルのダウンロード
  - ファイルマネージャを用いてのファイルの配置

本ラボの受講後には...

- ディープラーニングのワークフローを理解できています。
- コンボリューショナルニューラルネットワークをセットアップ、トレーニングすることができます。
- 本当のエキスパートになるための、最初のステップを実践することができます。
  自分自身のネットワークの設定ができ、より深く理解するために、何をすればよいかがわかります。

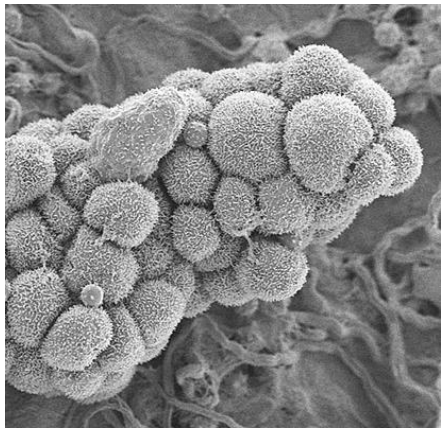
ディープラーニングとは？


機械学習
ニューラルネットワーク
ディープ
ラーニング

ディープラーニングの広い応用

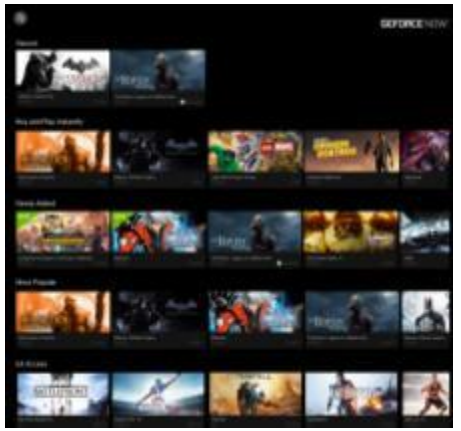
インターネット & クラウド

画像分類
言語認識
翻訳
言語処理
感情分析
リコメンデーション



薬学 & 生物学

ガン細胞検出
糖尿病のグレード決定
創薬

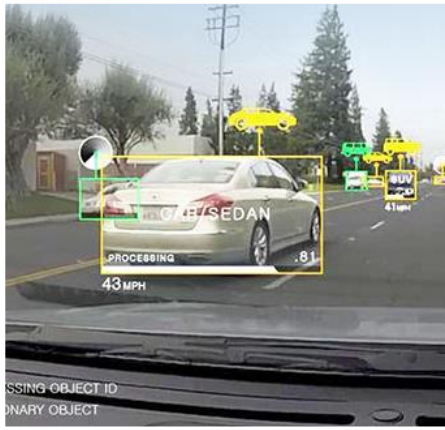


メディア&エンターテイメント

動画のキャプション付け
動画検索
リアルタイム翻訳

セキュリティ & 防衛

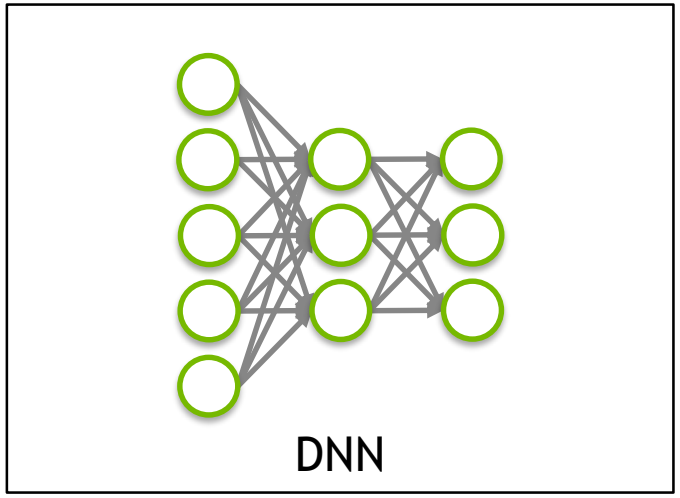
顔検出
ビデオ監視
衛星画像

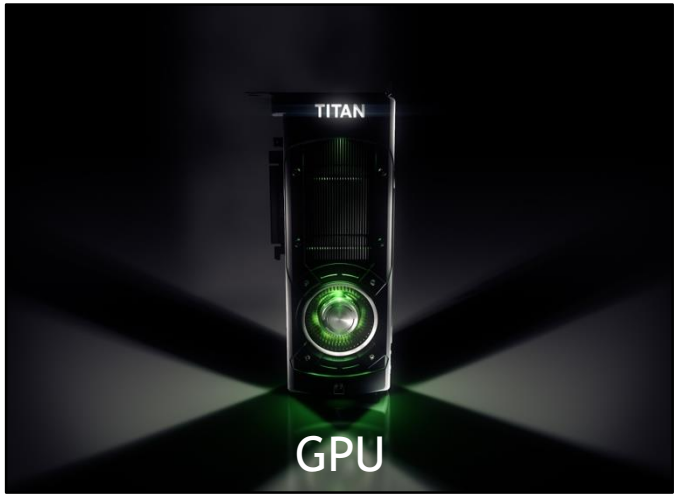


自動走行車

歩行者検出
レーントラッキング
道路標識の認識

機械学習におけるビッグバン



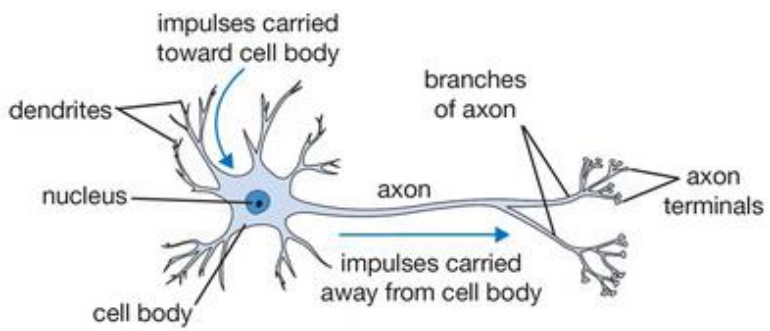


“GoogleのAIエンジンは コンピュータハードウエアの世界における変化を示している。GPUを搭載したマシンに依存しているのだ… これまでに、より広範なテクノロジが用いられてきたが、それ以上に強く、GPUに依存しているのだ。”

WIRED

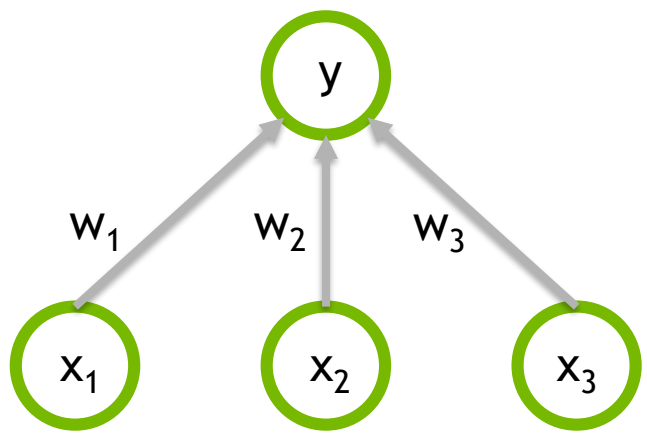
人工的なニューロン

生物学的ニューロン



From Stanford cs231n lecture notes

人工のニューロン



$$y=F(w_1x_1+w_2x_2+w_3x_3)$$

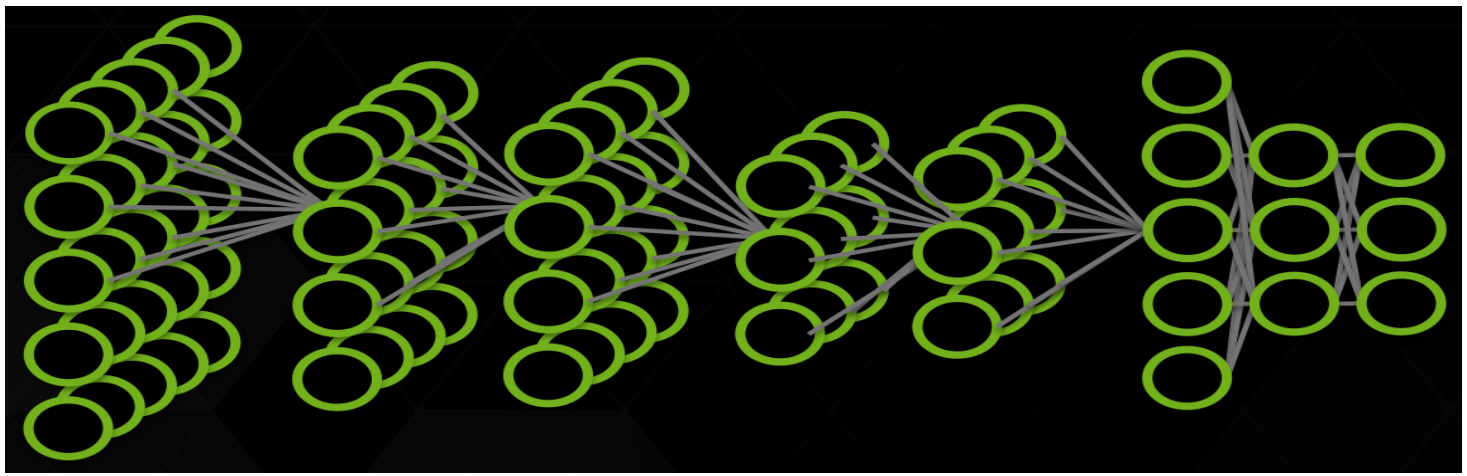
NVIDIA DEEP LEARNING INSTITUTE

人工的な神経ネットワーク

トレーニングできる単純な数学的なユニットの集合は、
複雑な機能を学ぶことができる

隠れ層

入力層



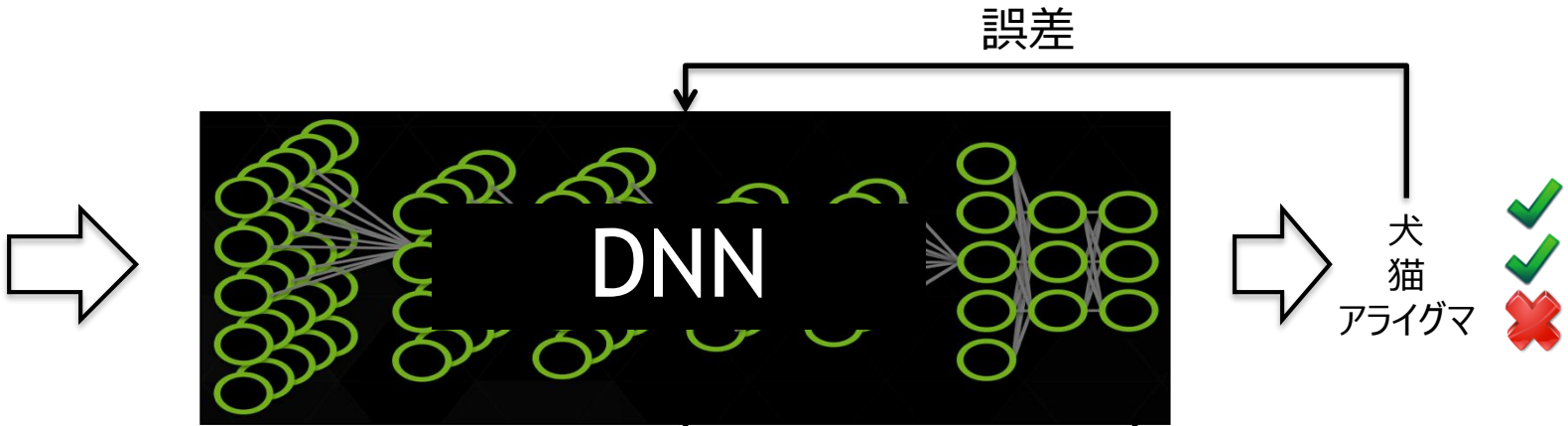
出力層

人口の神経ネットワークは、十分なトレーニングデータが与えられれば、
生の入力データから出力を決定する、非常に複雑な関数を近似することができる。

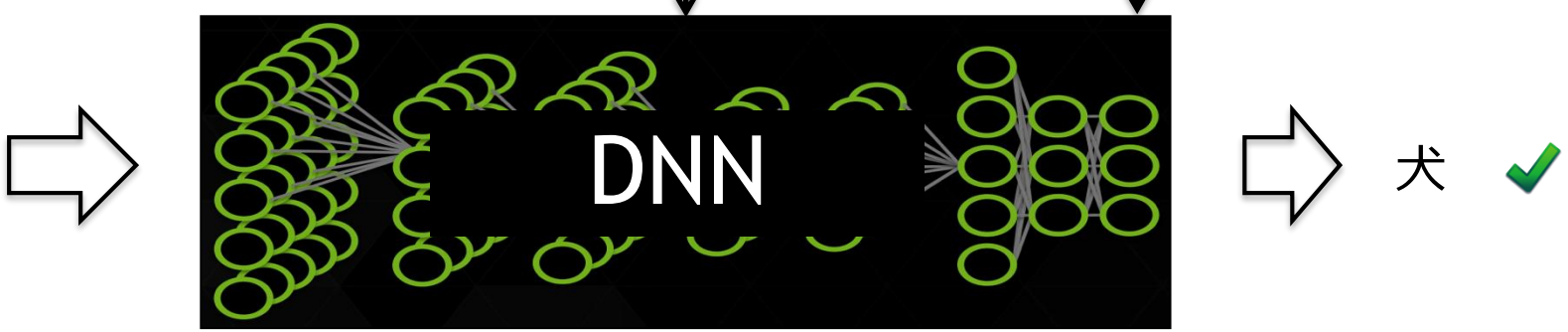
NVIDIA DEEP LEARNING INSTITUTE

ディープラーニングのアプローチ

トレーニング:

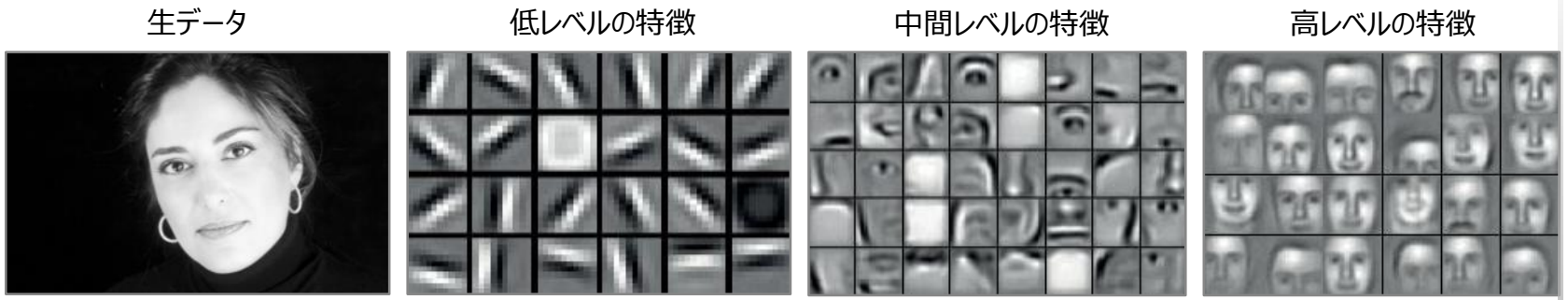


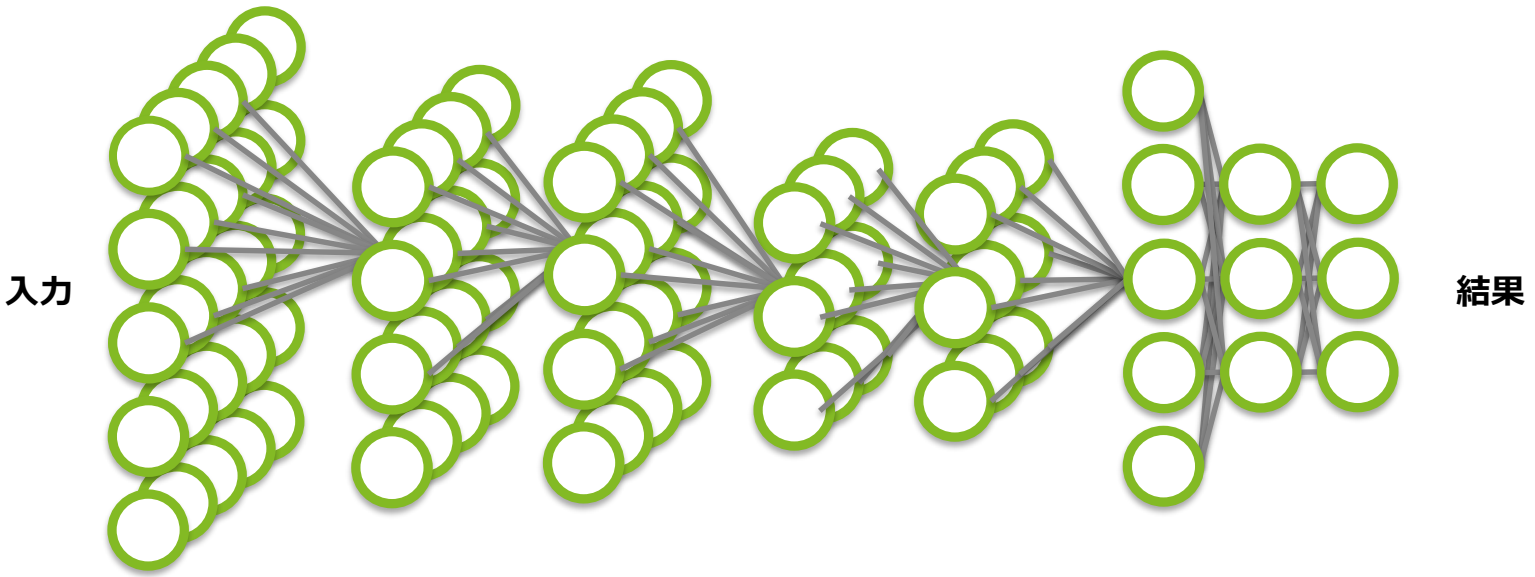
デプロイ:



NVIDIA DEEP LEARNING INSTITUTE

ディープニューラルネットワーク (DNN)





アプリケーションの構成要素:

タスクの目的objective
例: 顔の同定
トレーニングデータ
1千万-1億 のイメージ
ネットワークアーキテクチャ
~10から-数百のレイヤー
10億のパラメータ
学習アルゴリズム
~30 Exaflops
1-30 GPU日

NVIDIA DEEP LEARNING INSTITUTE

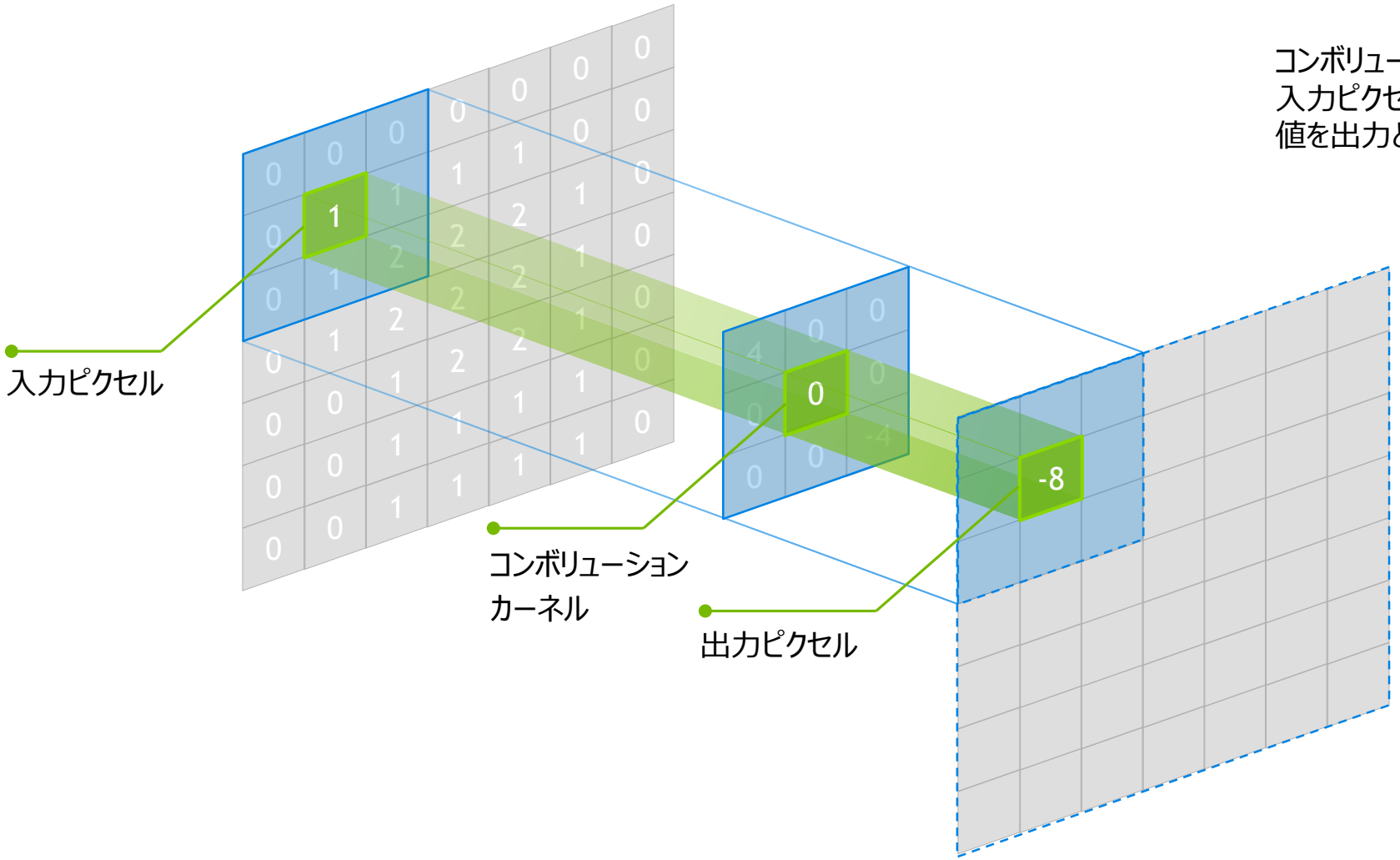
ディープラーニングの利点

- 堅牢性
  - 事前に特徴量のデザインをする必要がない。
    - 現在のタスクに最適な特徴量が、自動的に学習される。
  - 元来データに存在する多様性に対しても、学習により、堅牢となる。
- 一般化
  - 同じニューラルネットワークによるアプローチは、多数の異なるアプリケーション、および、データタイプに対して適用される。
- スケーラブル
  - より多くのデータを用いた学習により、性能が改善する。
    演算手法は、超並列化することができる。

NVIDIA DEEP LEARNING INSTITUTE

コンボリューション

コンボリューションカーネルの係数と、
入力ピクセルを掛け、足し合わせた
値を出力とする。



NVIDIA DEEP LEARNING INSTITUTE

CAFFE・DIGITS

本日使用するツール

CAFFE とは？

オープンソースのディープラーニングフレームワーク

- Berkeley Vision and learning Center (BVLC) において開発
- 多くのコントリビュータにより構成されるオープンソースコミュニティ
- C++/CUDA による実装。高速、よく検証されたコード
  シームレスな GPU によるアクセラレーション
- コマンドライン、Python, MATLAB インターフェース
- リファレンスモデルや、サンプルもある。

caffe.berkeleyvision.org
http://github.com/BVLC/caffe

CAFFE の機能

データのプリプロセスと管理

データフォーマット

LevelDB・LMDB データベース

インメモリ (C++・Python のみ)

HDF5

画像ファイル

プリプロセスツール

生画像からの LevelDB/LMDB の作成

トレーニング用と検証用のデータセット作成(シャッフル付き)

平均イメージの生成

データ変換

イメージのトリミング・リサイズ、スケーリング、位置反転

平均値を引く

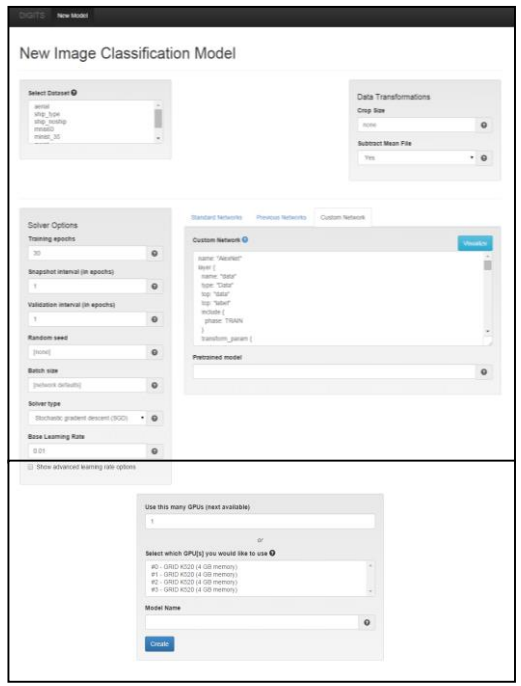
NVIDIA DEEP LEARNING INSTITUTE

NVIDIA DIGITS

インタラクティブなディープラーニングトレーニングのための GPU システム

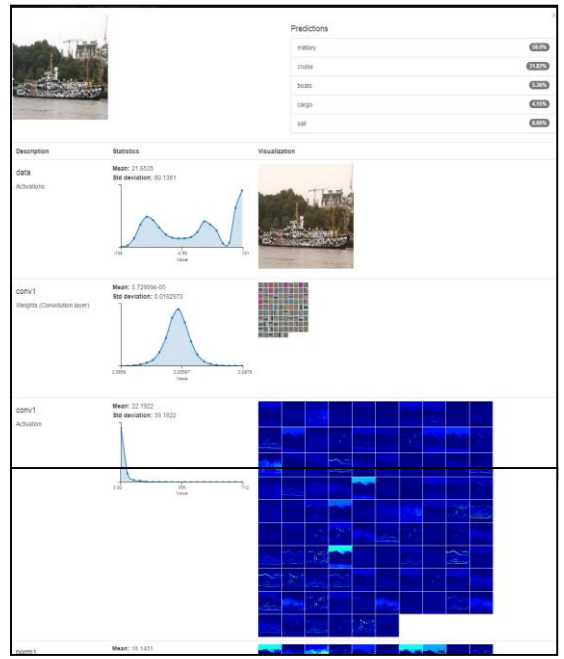
データの準備

DNN の設定



トレーニングの進行モニタ

可視化



チュートリアル:
手書き文字認識

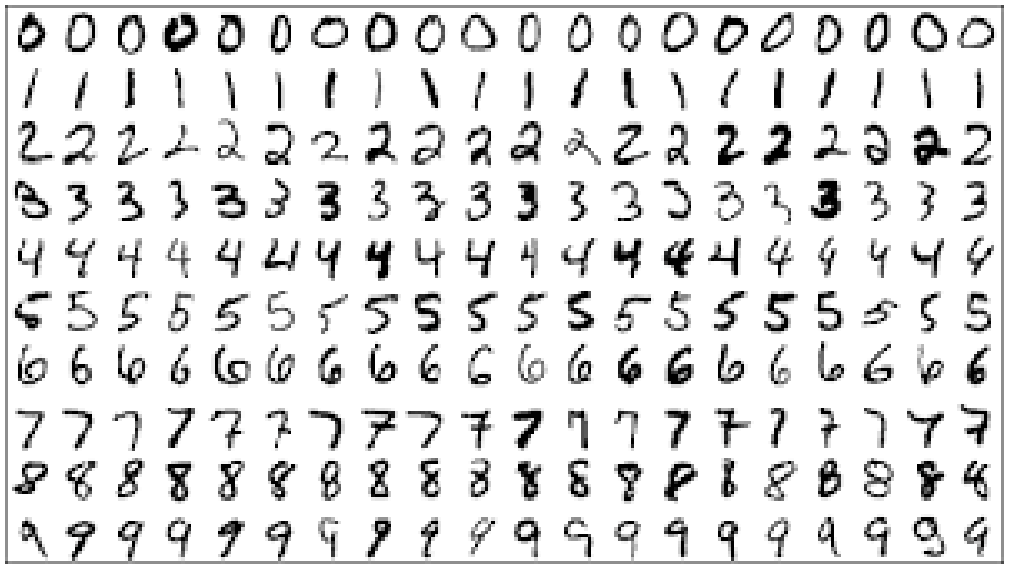
チュートリアルのゴール

- ディープラーニングのワークフローを学ぶ
- 手書き文字認識
  - コンボリューショナルネットワークのトレーニング (Caffe と DIGITS)
- 複数の異なる手法を試し、結果を改善する
  - トレーニングと最適化の繰り返しは、ディープラーニングにおける本質的な手続き

手書き文字認識

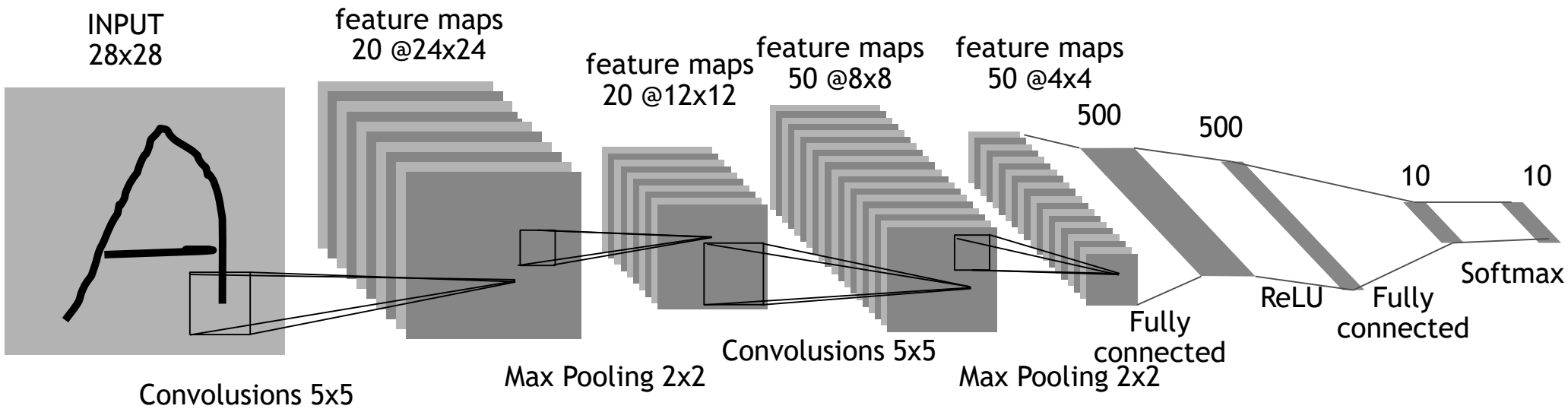
機械学習におけるHELLO WORLD?

- MNIST データセット
  (Yann LeCun's Web site)
  http://yann.lecun.com/exdb/mnist/
- サイズ：28x28、グレースケール、
  ピクセル値 0 - 255
  トレーニング用データ：6万
  テスト用データ：1万
- 入力ベクタサイズ：784 (= 28 x 28)
- 出力は、0-9までの整数



NVIDIA DEEP LEARNING INSTITUTE

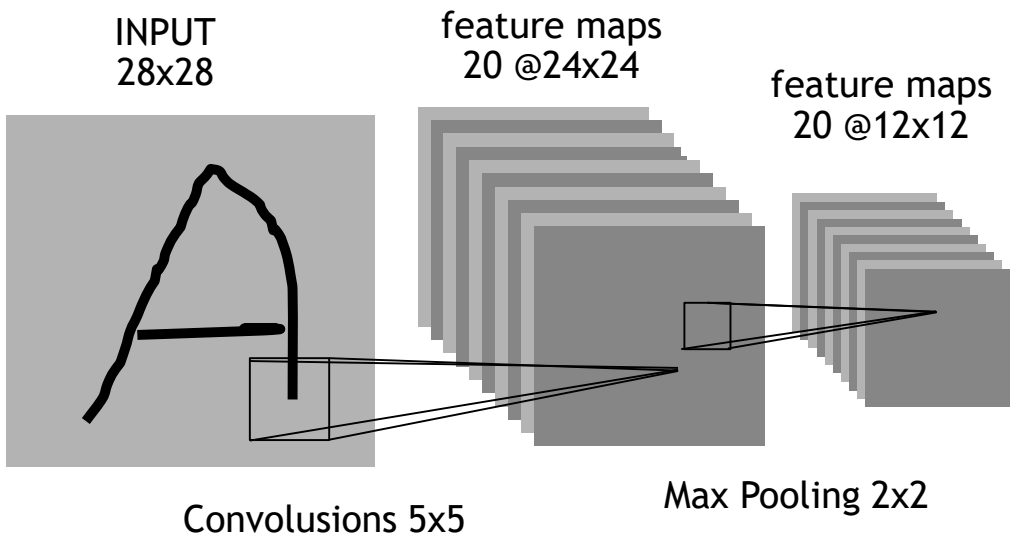
本日使うディープニューラルネットワーク



NVIDIA DEEP LEARNING INSTITUTE

コンボリューション / サブサンプリング

入力層



- コンボリューションレイヤー

  コンボリューションの算出
  複数の特徴マップを生成する

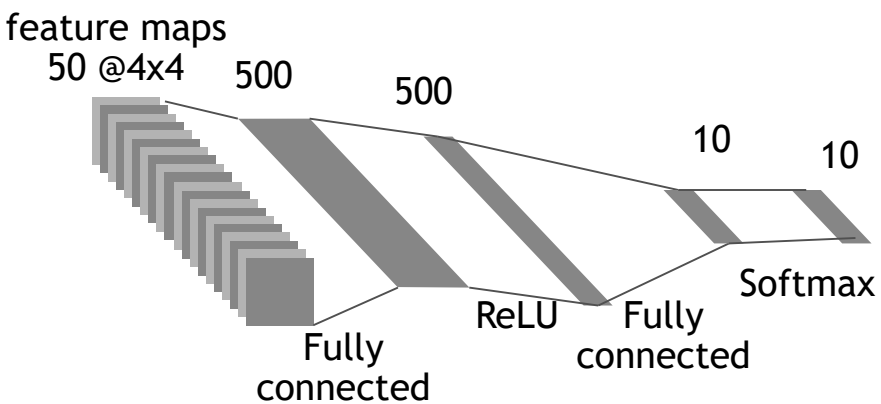
- 最大値プーリング

  プーリング
  2x2 の領域から、最大値を取得

- サブサンプリングで、大きさを縦横半分に

NVIDIA DEEP LEARNING INSTITUTE

全結合レイヤ、活性化関数



- 50 @ 4x4 = 800ピクセルを、ベクトルにする。
- Fully connectedレイヤ (行列)
- ReLU：活性化関数
- $max(0, x)$
- Softmax：活性化関数
- $\frac{exp(a_i)}{\sum_j^n exp(a_j)}$

NVIDIA DEEP LEARNING INSTITUTE

ニューラルネットワークモデルの定義

Deep Learning model definition

Protobuf モデルフォーマット

ネットワークモデル、および、トレーニングパラメータの定義

Caffeコードの自動生成、自動形式チェック

高い可読性、強く型付けされている

Googleにより開発された

```
name: “conv1”
type: “Convolution”
bottom: “data”
top: “conv1”
convolution_param {
      num_output: 20
      kernel_size: 5
      stride: 1
      weight_filler {
            type: “xavier”
      }
}
```

NVIDIA DEEP LEARNING INSTITUTE

CAFFE の機能

ニューラルネットワークの定義

Loss関数:

分類

- Softmax
- Hinge loss

線形回帰

- Euclidean loss

多値分類

- Sigmoid cross entropy loss

などなど…

使用可能なレイヤー種別:

- Convolution
- Pooling
- Normalization

利用可能な関数:

- ReLU
- Sigmoid
- Tanh

などなど…

NVIDIA DEEP LEARNING INSTITUTE

ラボのセットアップ

ログインからラボの選択まで

https://nvlabs.qwiklab.com にアクセス

ログイン (もしくは、新規ユーザ作成)

クラス “**DLI20170522-Japan**” を選択

“DIGITSによるディープラーニング入門”を選択。“選択”ボタンを押す。

“ラボを開始”ボタンを押す

わからないことがあったら、会場のアシスタントに聞いてください！

ラボ開始

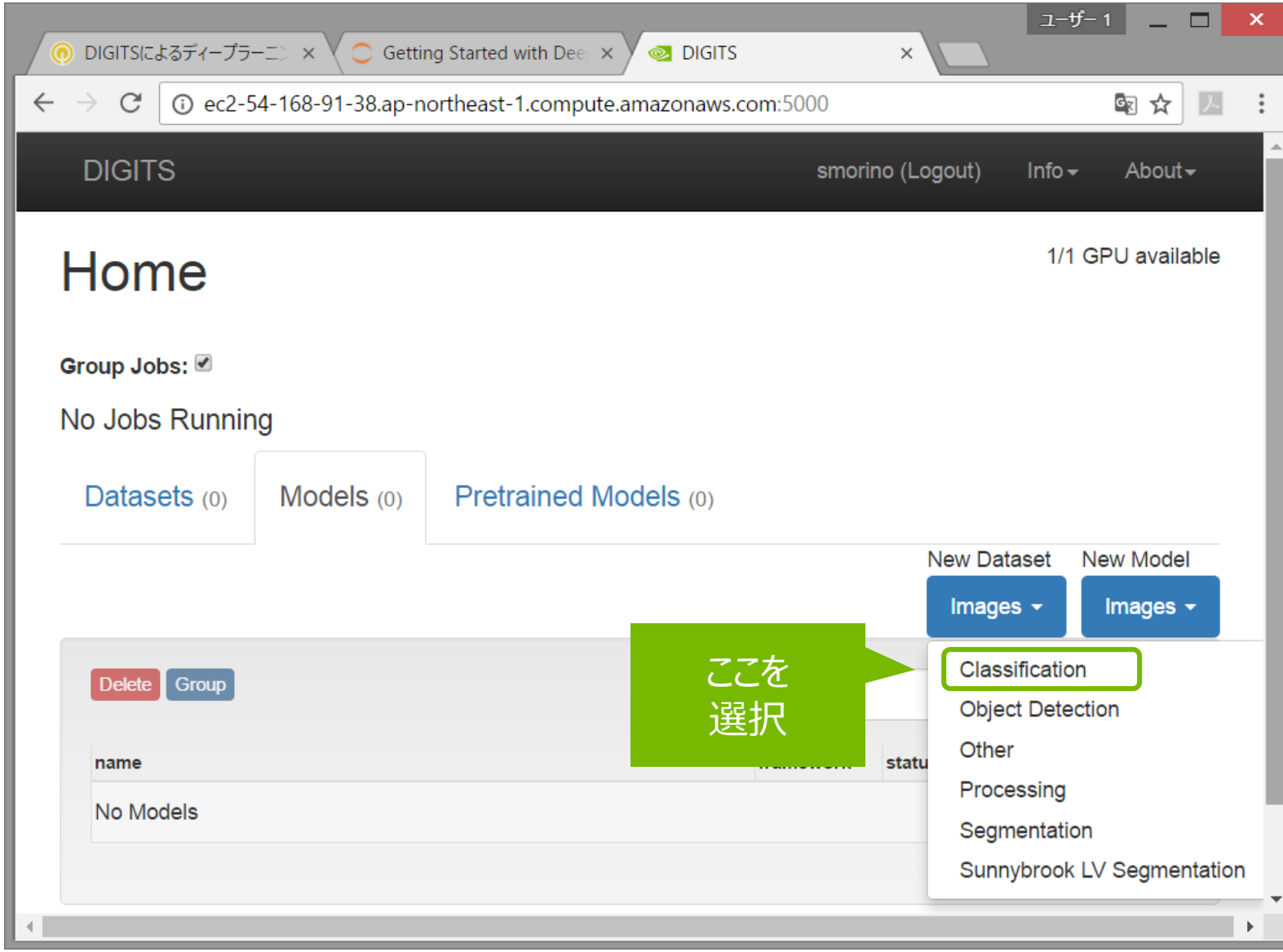
保護された通信 | https://nvlabs.qwiklab.com/focuses/1885
QWIKLABS
セッション中 3
予定されているクラス 0
受講済み 6
クラスの資料
アカウント
サインアウト
ラボの評価:
DIGITSによるディープラーニング入門
終了
残り時間:
01:53:30
接続情報
ラボの接続方法
ラボに接続するにはインストラクションをご参照ください
注意: このラボで使用しているAWSリソースには、受講中のハンズオンラボ、またはQwiklab™に関連しないデータは一切送信しないでください
カスタム接続の詳細
Click here to launch your lab.
ラボの詳細
ここをクリック

Imagesから

“Classification”を選択

Login

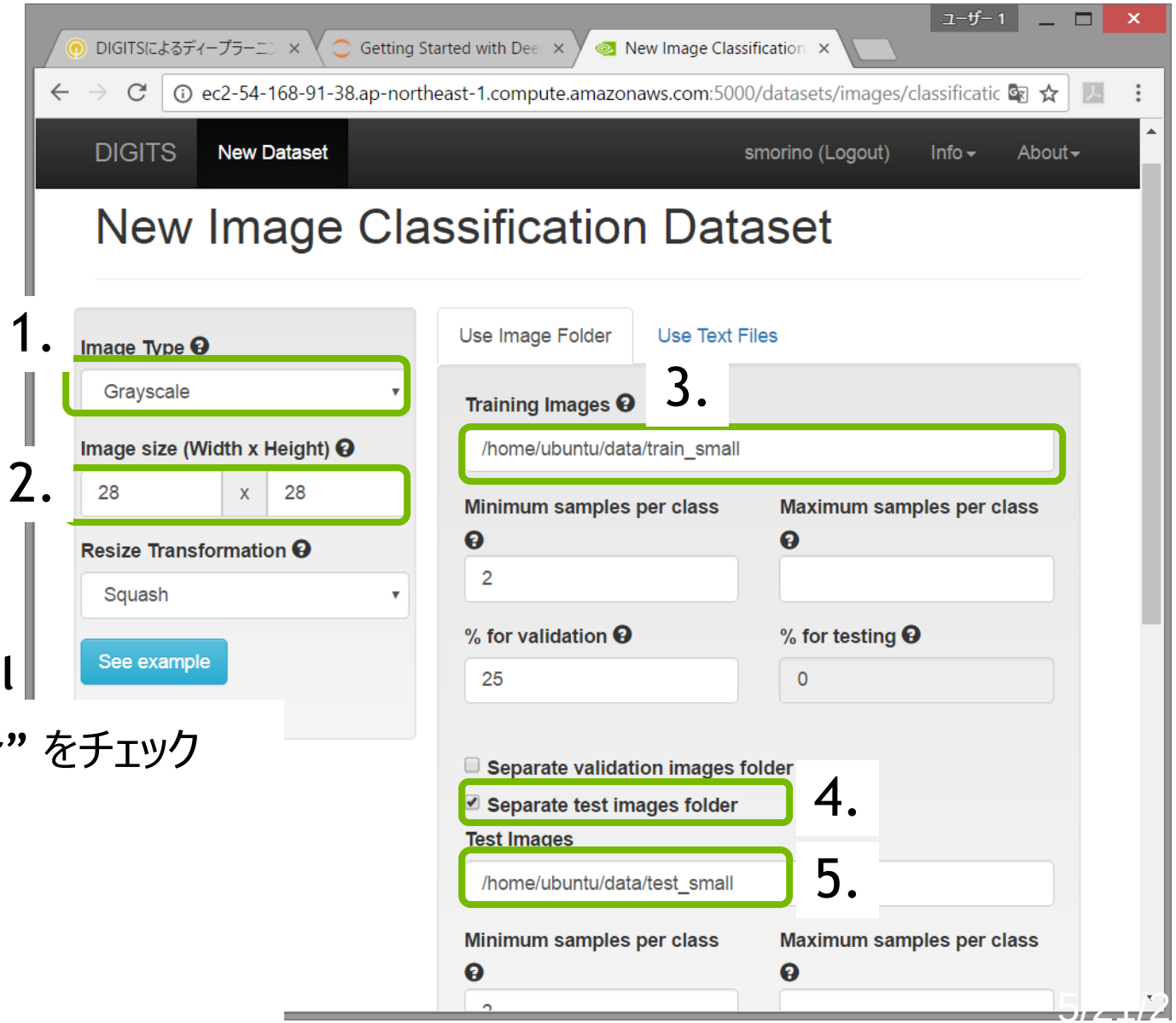
小文字を使ってください。



データセットの作成

その 1

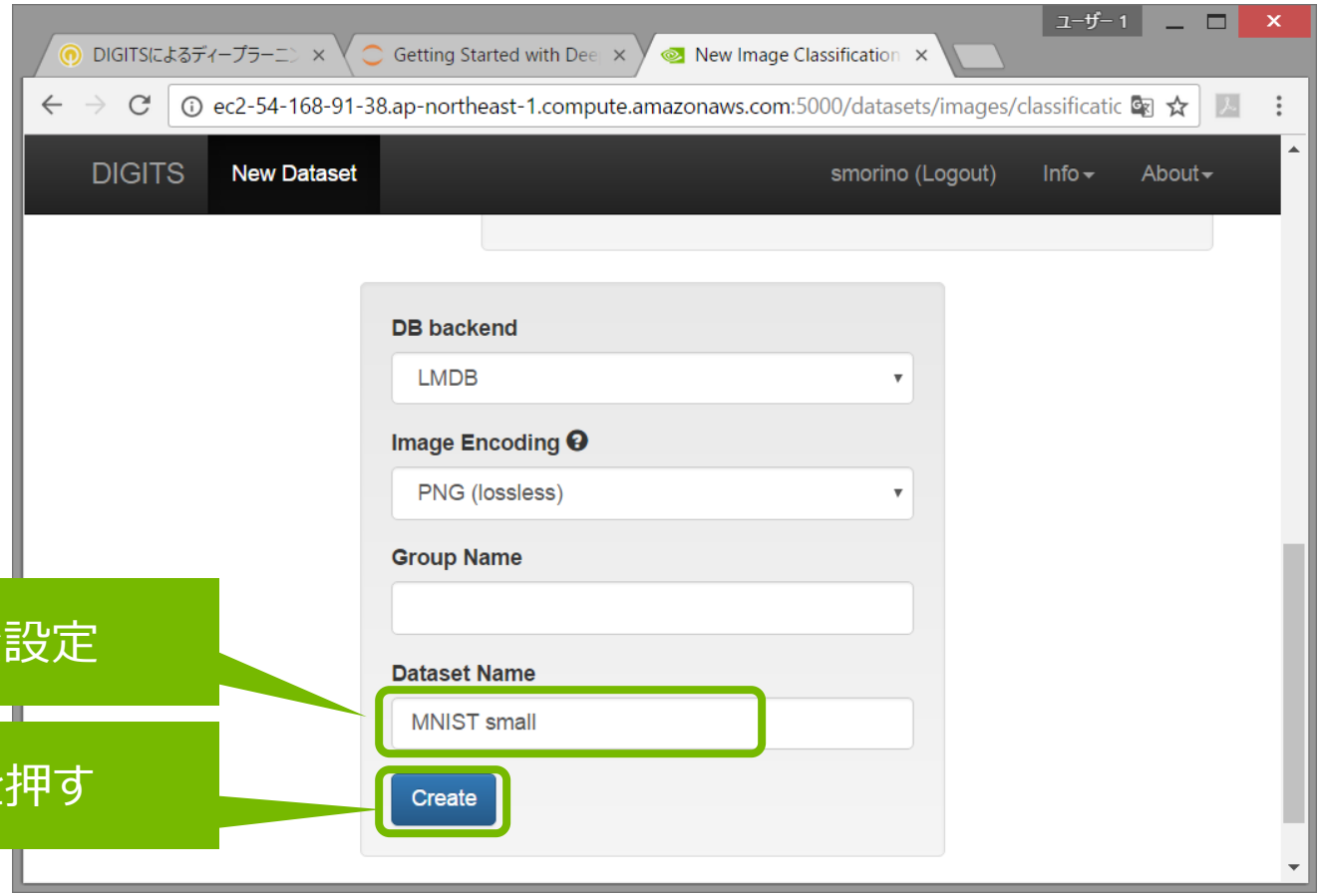
1. Image Type : Grayscale
2. Image Size : 28 x 28
3. Training Images:
/home/ubuntu/data/train_small
4. **“Separate test images folder”** をチェック
5. Test Images :
/home/ubuntu/data/test_small



データセットの作成

その2

1. “MNIST small”を設定
2. “Create” ボタンを押す



モデルの作成
その 1

1. 左上の”DIGITS”をクリックして、

   HOME画面に戻る

2. New Model

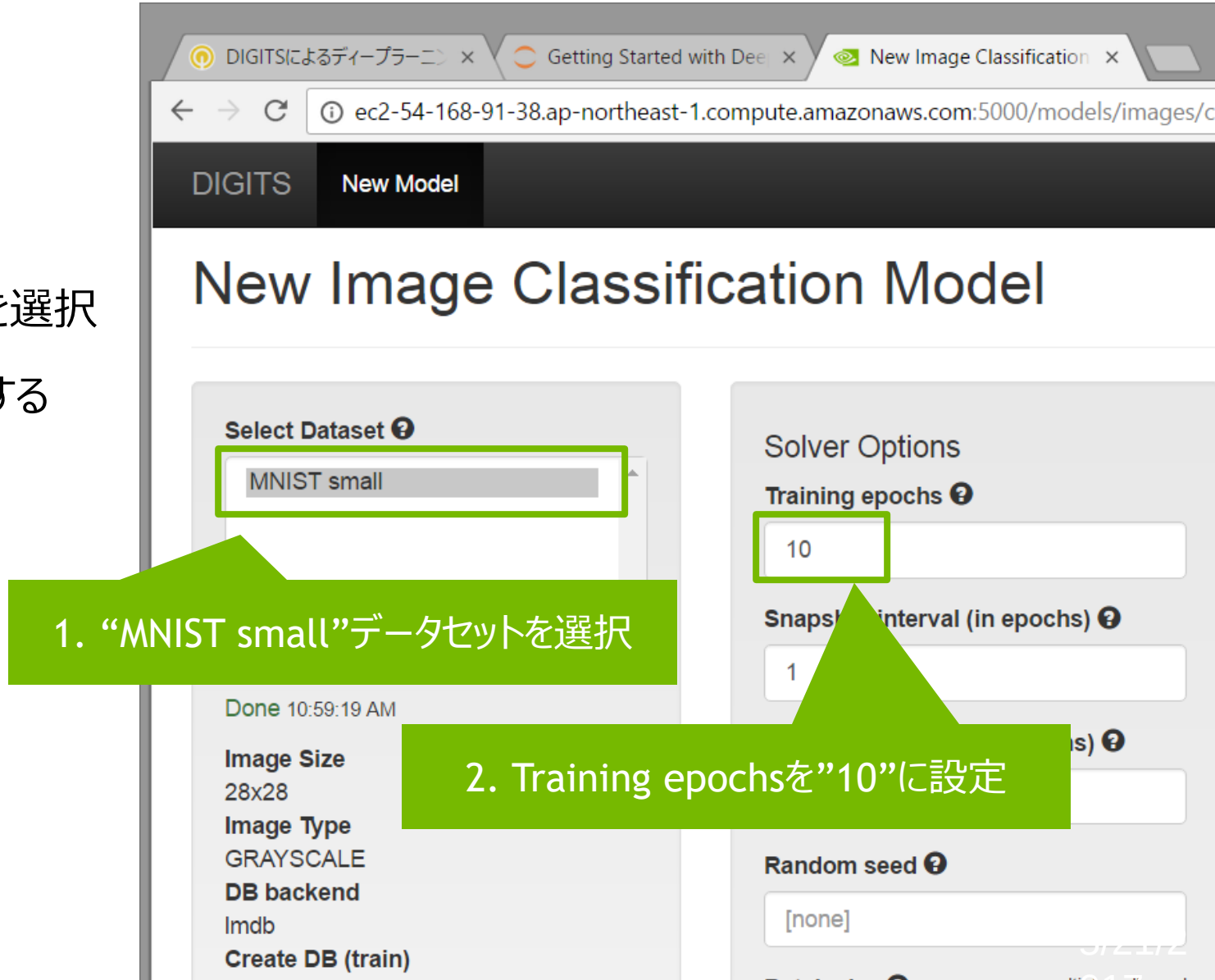
   Imagesから、

   Classificationを選択

モデルの作成

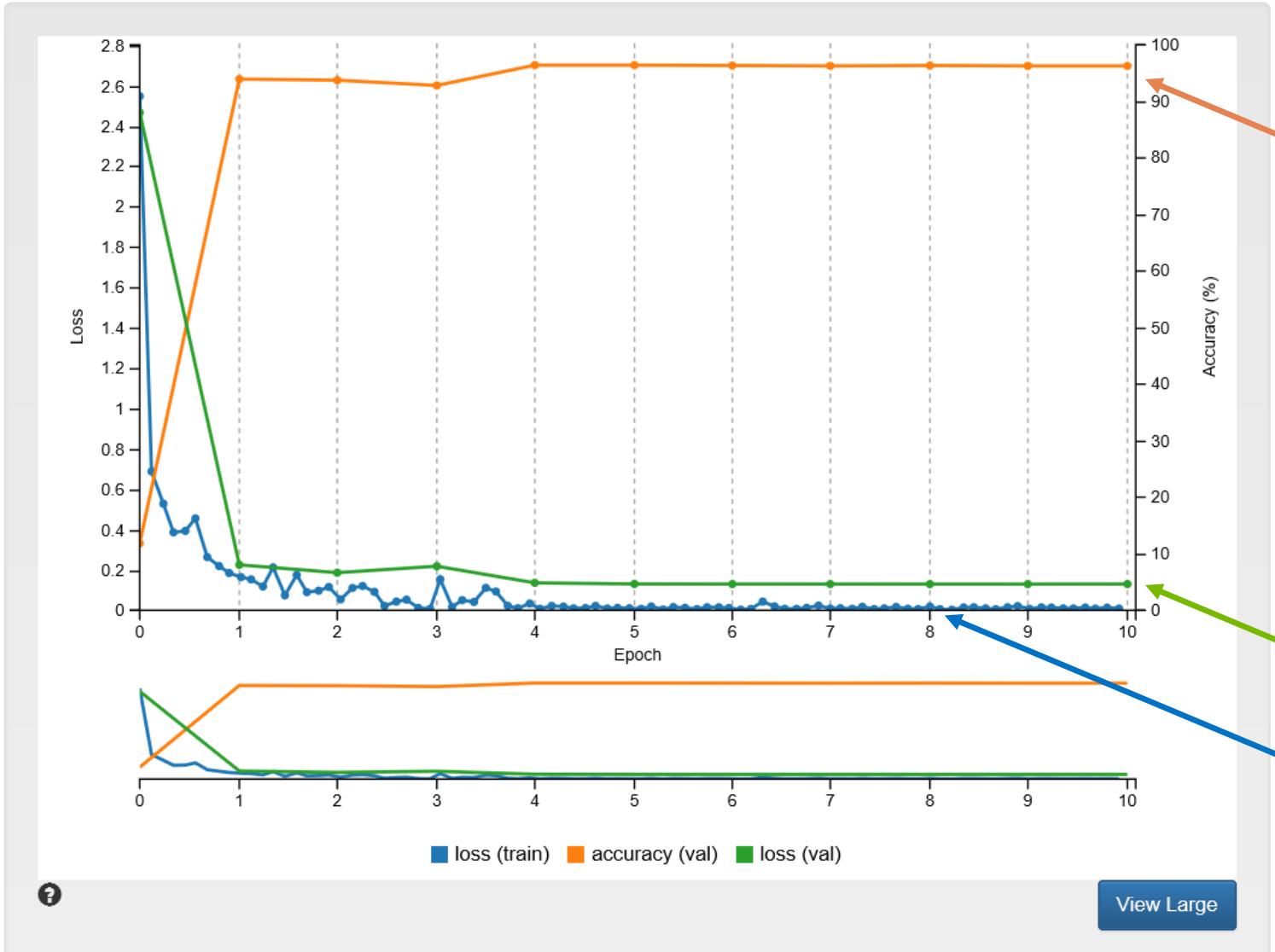
その 2

1. “**MNIST small**” データセットを選択
2. “**Training Epochs**” を10にする



モデルの作成
その 2

1. フレームワークは、“**Caffe**”を選択
2. モデルは、 “**LeNet**”を選択
3. モデル名は、“**MNIST small**”を入力



Accuracy
検証データセット
から求められる。

Loss 関数
(検証用)

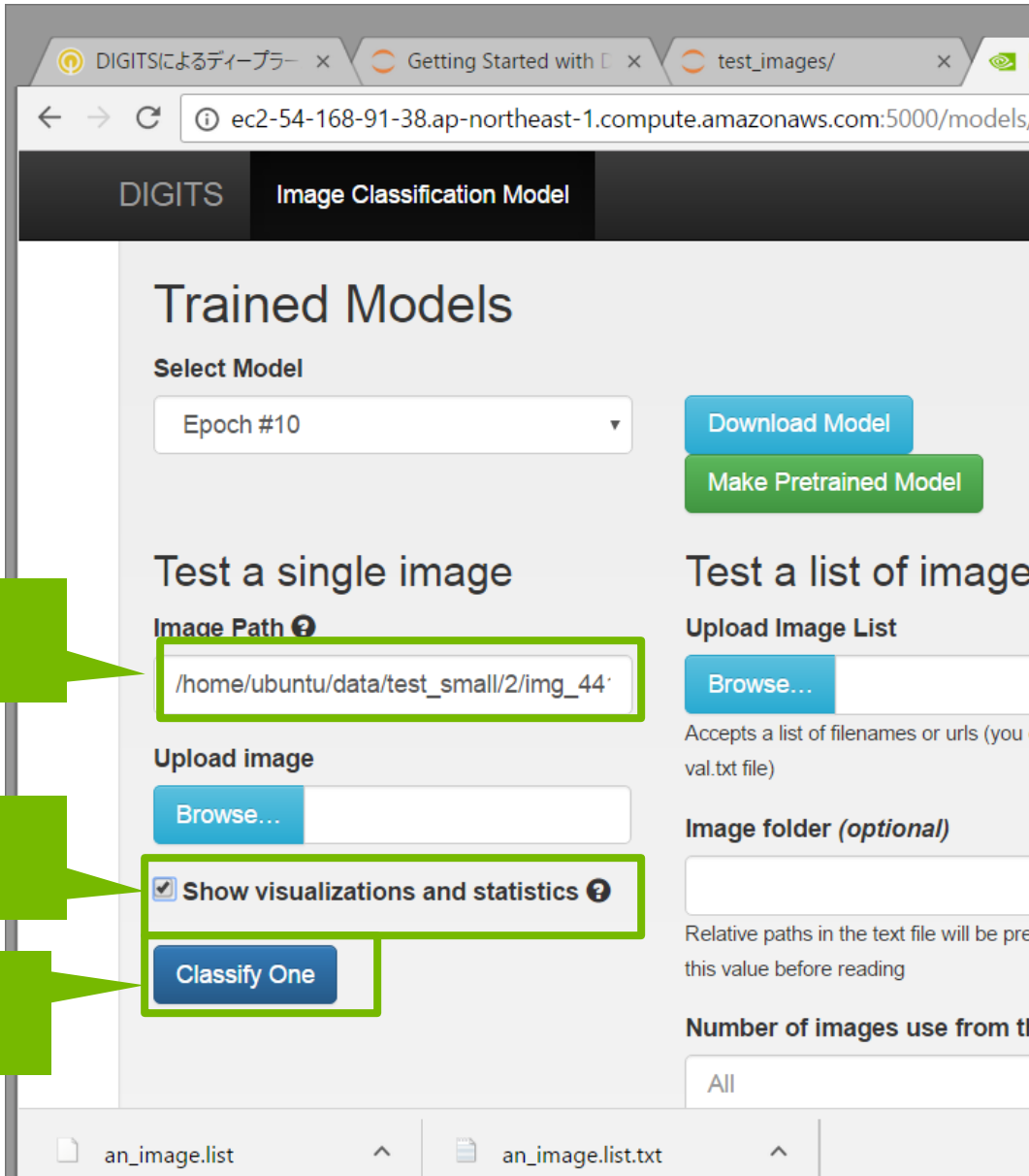
Loss 関数
(トレーニング)

一つのファイルでテスト

1. Image Pathに、以下を入力
/home/ubuntu/data/test_small/2/img_4415.png

2. “Show visualization and statistics”をチェック

3. “Classify One”ボタンを押す



現実世界のイメージを試す

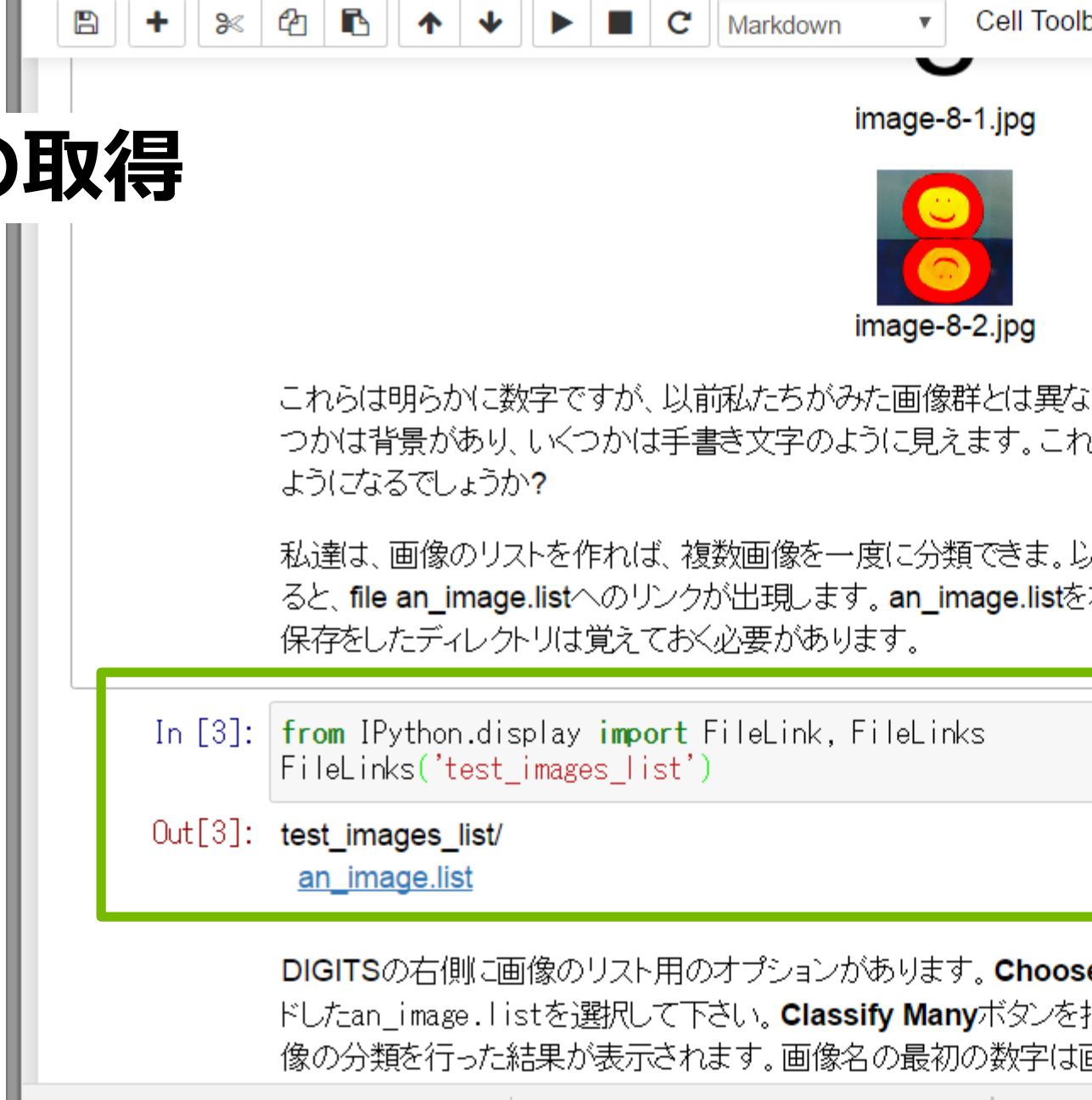
イメージリストファイルの取得

jupyterのページに戻る。

In [3] を実行(Ctrl + Enter)

出力される、an_image.listを右クリック

「テキストファイル」として保存



イメージリストファイル

テストデータへのパスの列

テキストエディタで開いた時の見え方

```
/home/ubuntu/notebook/test_images/image-1-1.jpg
/home/ubuntu/notebook/test_images/image-2-1.jpg
/home/ubuntu/notebook/test_images/image-3-1.jpg
/home/ubuntu/notebook/test_images/image-4-1.jpg
/home/ubuntu/notebook/test_images/image-7-1.jpg
/home/ubuntu/notebook/test_images/image-8-1.jpg
/home/ubuntu/notebook/test_images/image-8-2.jpg
```

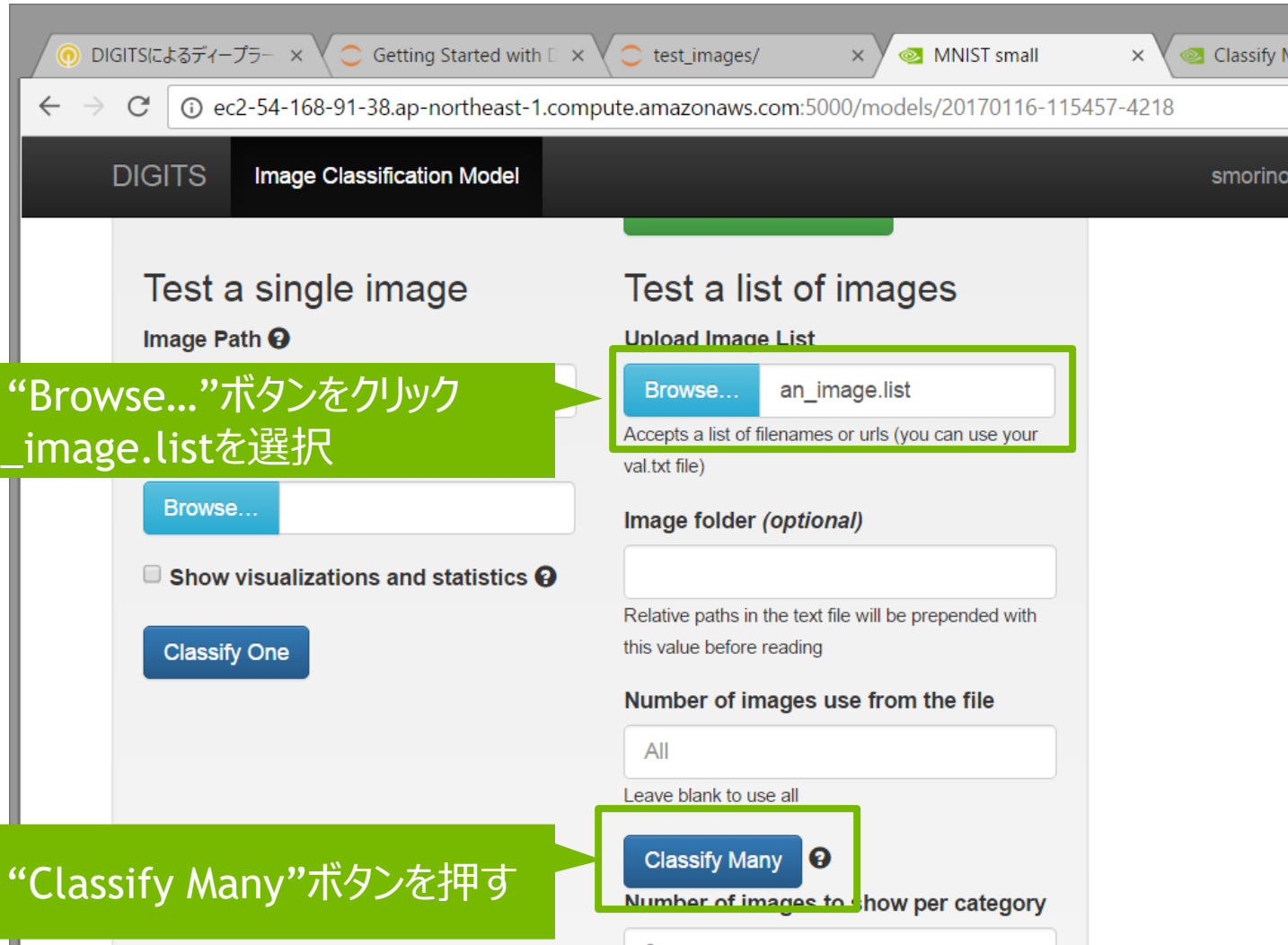
イメージリストでテストする

1. Upload Image List

   “Browse…”ボタンを押す

   an_image.listを選択

2. “Classify Many”ボタンを押す



最初の結果

Small dataset (30 epoch)

- Accuracy ：96 %
- トレーニングは、1分で終わる。

	SMALL DATASET
	1 : 99.90 %
	2 : 69.03 %
	8 : 71.37 %
	8 : 85.07 %
	0 : 99.00 %
	8 : 99.69 %
	8 : 54.75 %

NVIDIA DEEP LEARNING INSTITUTE

FULL DATASET

6倍大きなデータセット

- データセット
  - Training Images： /home/ubuntu/data/train_full
  - Test Image： /home/ubuntu/data/test_full
  - Dataset Name : MNIST full
- モデル
  - “MNIST small”をクローン(Cloneボタンを押す).
  - モデル名に、“MNIST full”を設定し、”Create”ボタンを押す.

NVIDIA DEEP LEARNING INSTITUTE

2つ目の結果

Full dataset (30 epoch)

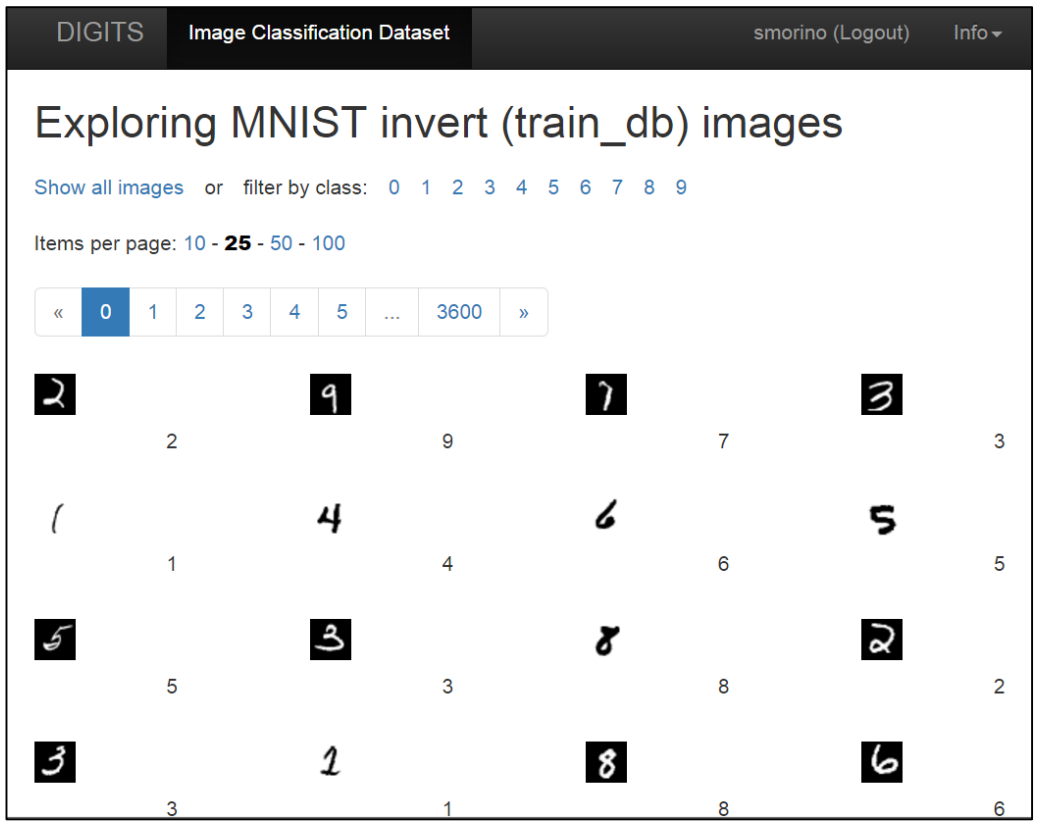
- 99 % of accuracy achieved.
- 現実世界の画像の認識には、改善なし。

	SMALL DATASET	FULL DATASET
	1 : 99.90 %	0 : 93.11 %
	2 : 69.03 %	2 : 87.23 %
	8 : 71.37 %	8 : 71.60 %
	8 : 85.07 %	8 : 79.72 %
	0 : 99.00 %	0 : 95.82 %
	8 : 99.69 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %

NVIDIA DEEP LEARNING INSTITUTE

DATA AUGMENTATION

反転したイメージの追加



- Pixel(Inverted) = 255 - Pixel(original)
- 黒い背景に白い文字
  → 白い背景に黒い文字
- Training Images :
  /home/ubuntu/data/train_invert
- Test Image :
  /home/ubuntu/data/test_invert
- Dataset Name : MNIST invert

NVIDIA DEEP LEARNING INSTITUTE

DATA AUGMENTATION

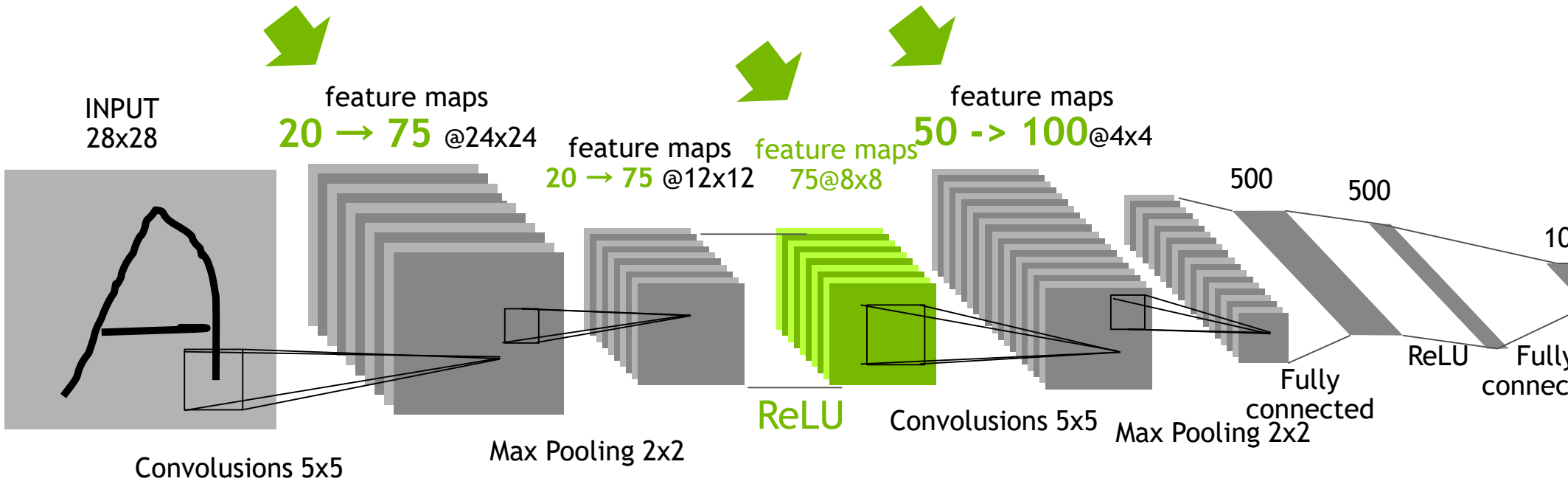
反転したイメージの追加(30 epoch)

	SMALL DATASET	FULL DATASET	+INVERTED
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %

NVIDIA DEEP LEARNING INSTITUTE

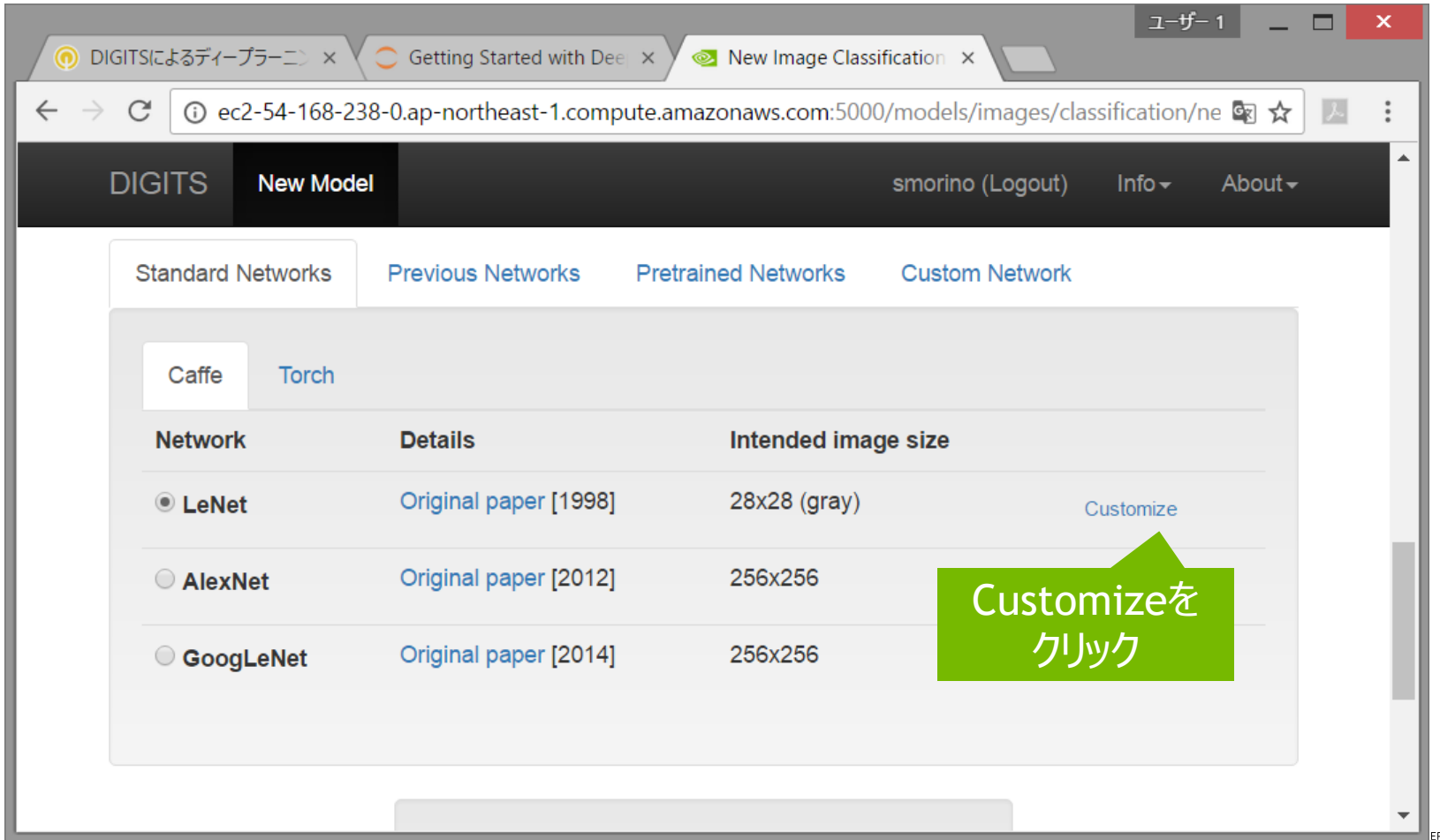
ネットワークの修正

ReLUレイヤの追加と、コンボリューションフィルタの修正

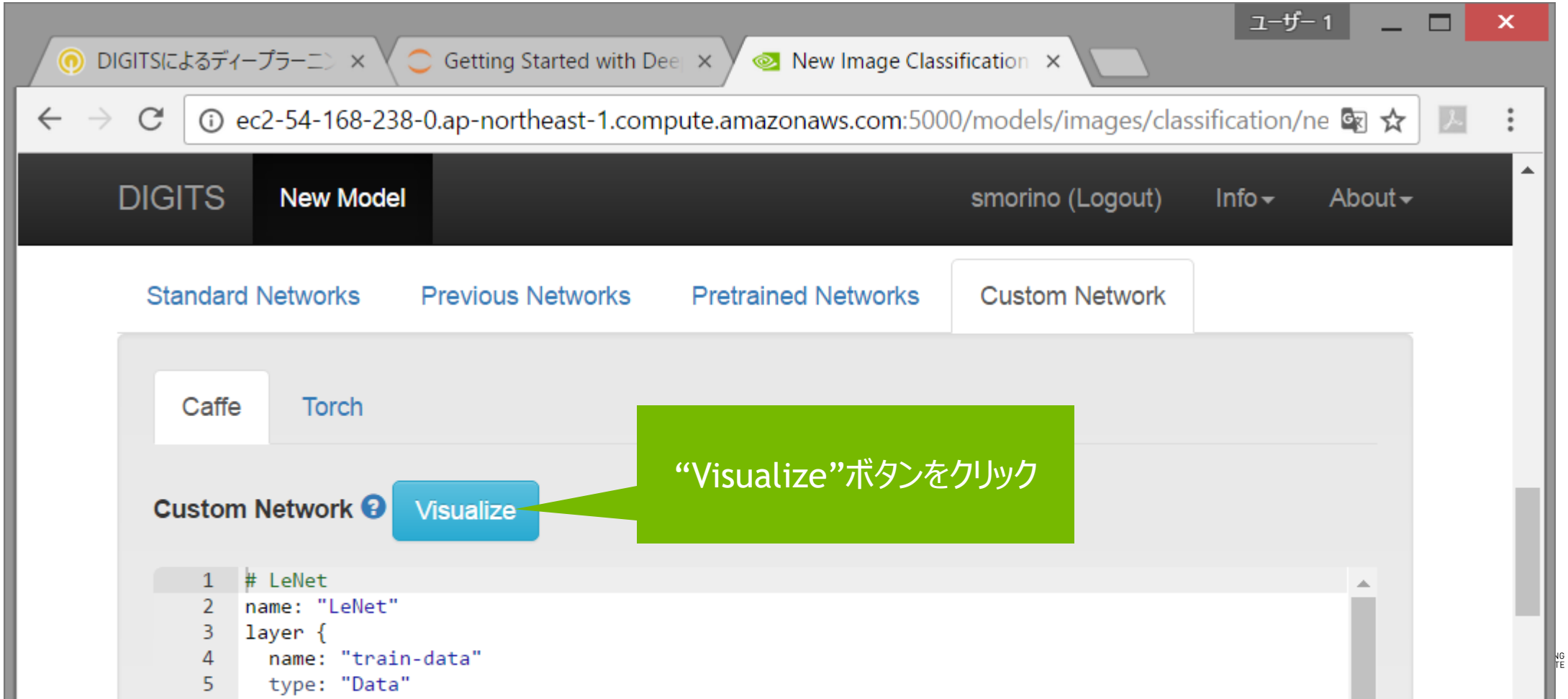


NVIDIA DEEP LEARNING INSTITUTE

LENETのネットワークをカスタマイズ



ネットワークの可視化


ユーザー 1
DIGITSによるディープラーニ
Getting Started with Dee
New Image Classification
ec2-54-168-238-0.ap-northeast-1.compute.amazonaws.com:5000/models/images/classification/ne
DIGITS
New Model
smorino (Logout)
Info
About
Standard Networks
Previous Networks
Pretrained Networks
Custom Network
Caffe
Torch
Custom Network
Visualize
“Visualize”ボタンをクリック
1 # LeNet
2 name: "LeNet"
3 layer {
4 name: "train-data"
5 type: "Data"

ネットワークの修正

ReLUレイヤの追加と、コンボリューションフィルタの修正

```
layer {
    name: "pool1“
    type: "Pooling“
    …
}

/* ReLUレイヤの追加 → */
layer {
    name: "reluP1"
    type: "ReLU"
    bottom: "pool1"
    top: "pool1"
}
/* ← ここまで */

layer {
    name: “conv2“
```

```
layer {
  name: "conv1"
  type: "Convolution"
    ...
    convolution_param {
    num_output: 75 /* feature map数 */
    ...
layer {
    name: "conv2"
    type: "Convolution"
    ...
    convolution_param {
    num_output: 100 /* feature map数 */
    ...
```

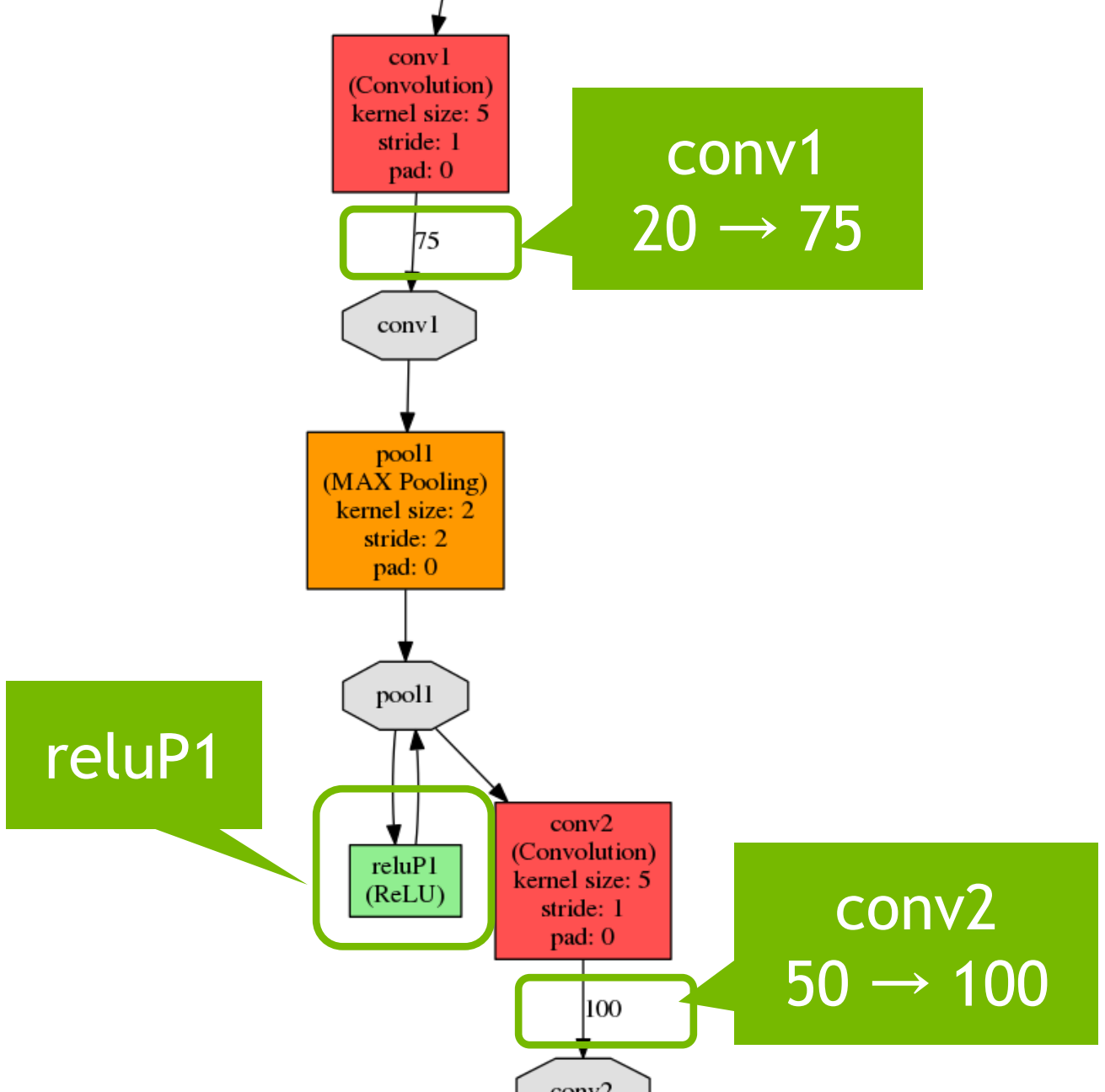
NVIDIA DEEP LEARNING INSTITUTE

ネットワークの修正

ReLUレイヤの追加

編集後、Visualizeボタンを押して、

ネットワークを確認。



ネットワークの修正

フィルタとReLUレイヤの追加 （ 30 epoch ）

	SMALL DATASET	FULL DATASET	+INVERTED	ADDING LAYER
	1 : 99.90 %	0 : 93.11 %	1 : 90.84 %	1 : 59.18 %
	2 : 69.03 %	2 : 87.23 %	2 : 89.44 %	2 : 93.39 %
	8 : 71.37 %	8 : 71.60 %	3 : 100.0 %	3 : 100.0 %
	8 : 85.07 %	8 : 79.72 %	4 : 100.0 %	4 : 100.0 %
	0 : 99.00 %	0 : 95.82 %	7 : 82.84 %	2 : 62.52 %
	8 : 99.69 %	8 : 100.0 %	8 : 100.0 %	8 : 100.0 %
	8 : 54.75 %	2 : 70.57 %	2 : 96.27 %	8 : 70.83 %

NVIDIA DEEP LEARNING INSTITUTE

www.nvidia.com/dli