

## ハンズオンラボ DIGITS による物体検出入門

山崎和博

ディープラーニング ソリューションアーキテクト エヌビディア

DEEP

LEARNING INSTITUTE

## AGENDA

ディープラーニングとは

Qwiklabs/DIGITSの使い方

DIGITSによる物体検出入門ハンズオン

# ディープラーニングとは

## 様々な分野でディープラーニングを応用





**人工ニューロン** 神経回路網をモデル化



F(x)=max(0,x)



人工ニューラルネットワークは、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。



入力層



#### ■ ロバスト性

- 特徴量の設計を行う必要がない
- 特徴は、学習用データのバラつきの影響を押さえながら、自動的に学習・獲得される

■ 一般性

同じニューラルネットワークのアプローチを、多くの異なるアプリケーションやデータに適用できる

■ スケーラブル

- より多くのデータで大規模並列化を行う事でパフォーマンスが向上する



# 畳込みニューラルネットワーク(CNN)

• 画像認識・画像分類で使われる、高い認識精度を誇るアルゴリズム。畳込み層で画像の特徴を学習



DEEP

## **畳込み層(CONVOLUTIONAL LAYER)**



DEEP LEARNING

Ø. DVIDIA

# ディープラーニング フレームワーク

## ディープラーニング フレームワーク GPUで高速化されたディープラーニング フレームワークが多数存在



https://developer.nvidia.com/deep-learning-frameworks

# ディープラーニング フレームワーク

	Caffe	Torch7	Theano	TensorFlow	Chainer	
インターフェース	C++/Python/Matlab	Lua/C	Python	C/C++/Python	Python	
cuDNN	5	5	5	5	5	
ライセンス	BSD-2	BSD	BSD	Apache 2.0	МІТ	
マルチGPU (1ノード)	0	0		0	0	
モデルの柔軟性	$\bigtriangleup$	Ø	Ø	0	Ø	
CNN	0	0	0	0	0	
RNN		0	0	0	0	
RBM	×	0	$\bigcirc$	0	0	
備考	高速 Caffe Model Zoo	多数のアルゴリズムを サポート	自動微分	自動微分 TensorBoard	Define by Run CuPy	



## 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
  - 高速で十分に検証された実装
  - ・ 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



<u>caffe.berkeleyvision.org</u> <u>http://github.com/BVLC/caffe</u>



CAFFEの機能 データのプリプロセスと管理

LevelDB・LMDB データベース

インメモリ (C++・Python のみ)

HDF5

画像ファイル

生画像から LevelDB/LMDB を 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)

平均画像の生成

### データ変換

画像のトリミング、リサイズ、 スケーリング、位置反転

平均値を引く



CAFFEの機能 ニューラルネットワークの定義

### Protobuf モデルフォーマット

- ▶ ネットワーク構造や学習パラメータの定義に使われる
- ▶ 様々なフォーマットに対応
  - Caffeのオートジェネレータで生 成可能
  - ▶構文チェック機能も存在
- ▶ コーディング不要

```
name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"
convolution param {
      num output: 20
      kernel size: 5
      stride: 1
      weight filler {
             type: "xavier"
```



CAFFEの機能 ニューラルネットワークの定義

### Loss<mark>関数</mark>:

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 DIGITS GPUで高速化されたディープラーニング トレーニング システム

モデルの作成 モデルのテスト 学習データの作成 学習過程の可視化 New Image Classification Model ship\_type3 g aerial Abort Job Delete Job Delete Job Select Dataset Data Transformation aertal ship\_type ship\_noshi maist0 maist\_35 Crop Size Datase solver, pototat Network (train/val train\_val prototat Network (deploy) deploy prototat Raw caffe output soffe output Job Information Initialized at 01:38:38 AM (1 second)
 Running at 01:38:40 AM ship type Subtract Mean File Done May 25, 03:23:53 · Initialized at Thu Jun 04, 03:27:52 AM Image Size 255x255 Image Type COLOR Job Director Running at Thu Jun 04, 03:27:53 AM (6 minutes, 25 seconds) Done at Thu Jun 04, 03:34:20 AM Image Type Color Image Dimensi 250x250 Resize Mode squash Encoding eate DB (train Solver Option Initialized at 01:38:38 AM (1 see Running at 01:38:40 AM Training spochs Custom Network Create DR (train) DB Entries Ratch size Solver typ Mane: 22 203 Man: Aviation: 39 1822 ..... 1 000 -1 000 -1 000 -1 000 -1 000 -1 000 which GPUIsI you would like to use 6

http://developer.nvidia.com/digits

### NVIDIA DIGITS GPUで高速化されたディープラーニングトレーニング システム ▶ 手元のPCからWebブラウザでアクセス可能なディープラーニングトレーニング システム



- 画像分類と物体検出の為のDNNのデザ インと可視化の機能を提供
- NVIDIAが最適化したフレームワークで高 速に学習が可能
- ハイパーパラメータのチューニングを強力に
   サポート
- 学習のジョブを簡単にスケジューリング、リ アルタイムにaccuracyとlossを監視
- 様々な学習データフォーマット、フレーム ワークに対応

## ハンズオンを開始しましょう

### ログインからラボの開始まで

- https://nvlabs.qwiklab.com にアクセス
   ログイン (or 新規ユーザ作成)
   クラス "DLI20170522-Japan"を選択
   エヌビディアDIGITSによる新体検出入門
- "エヌビディアDIGITSによる物体検出入門"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!



## DIGITSによる物体検出入門 ラボの開始

#### <u>https://nvlabs.qwiklab.com/</u>にアクセス [DLI20170522-Japan]->[エヌビディアDIGITSによる物体検出]を選択

セッジョン中のクラス: DLI2017-Japan			()	<b>10.2</b> 合計時間	•0	<b>7</b> 修了したラボ	<b>き 3</b> 受講済みクラス
クラスの詳細	^	≪INVIDIA. エヌビディアDIGITSによる物体検出入門 ③					
<ul> <li>NOTIONAL エヌビディアDIGITSIこよる物体検出入門</li> <li>NOTIONAL DIGITSIこよるディープラーニング入門</li> </ul>		エヌビディアDIGITSで		"ラの顔を検出を	時間	:	302レシット 90分
		行ってみま	ってみましょう		アク	セス時間:	115 分
					開始	時間:	8分
					レベ	lb:	Beginner

## DIGITSによる物体検出入門 ラボの開始



残り時間: **D1:55:00** 



### DIGITSの使い方: ワークフロー





### DIGITSの使い方:ホーム画面への戻り方



DIGITSの使い方: データセット/モデルの表示



DIGITSの使い方: データセット/モデルの作成





### **DIGITSによる物体検出入門** DIGITSの使い方: ネットワーク構造の変更方法

### - DIGITSは、Caffeを使って学習を行う事が出来る





DIGITSの使い方:ネットワーク構造の変更方法

### - Caffeのモデル定義ファイル(prototxt)を書き換える



DEEP LEARNING

Jupyter notebook上での処理実行









## ディープラーニングによる物体検出 クジラの顔を検出する

▶ データセット

Right Whale Recognition <u>https://www.kaggle.com/c/noaa-</u> <u>right-whale-recognition</u>







1. スライディング ウィンドウ

#### 画像全域をスキャンして検出

2. 候補の生成と分類

検出対象となる領域の候補を生成し、検出

3. 全畳み込みネットワーク (FCN)

畳み込みネットワークのみを用いて検出

4. DetectNet

回帰を併用し、物体を囲む矩形領域を推定





### ディープニューラルネットワークに「鯨」か「非鯨」の判別を学習させる




#### 画像を分割、それぞれのパッチ(領域)に対して検出を試みる









- 1. DIGITSを起動
- 2. 「鯨」「非鯨」を判別するニューラルネットワークを作成

[New Dataset / Images]->[Classification]からデータセットを作成 [New Model / Images]->[Classification]からモデルを作成し、学習

3. 学習後のモデルを用いて推論処理を行う



### DIGITSを開く

Imagesから

"Classification"を選択

初回だけログインを求められ るので何か入力(小文字で お願いします)





- 1. DIGITSを開く
- 2. [New Dataset / Images] -> [Classification]を選択する
- 3. 以下を入力 (2 ~ 3分かかります)

New Image Classificatio	on Dataset	DB backend	
		LMDB	
Image Type	Use Image Folder Use Text Files	Image Encoding 😡	
Color		PNC (lossloss)	
Image size (Width x Height)       256     x     256	/home/ubuntu/data/whale/data/train	Grou 2. Dataset Name	
Resize Transformation	2	"Whate_taces"	
See example 1. Training in "/home/ubur	nages ntu/data/whale/data/train" Separate validation images folder Separate test images folder	Dataset Name whale_faces Create 3."Create"ボタンを クリック	

1. スライディング ウィンドウ 学習データを見てみる

データセットの作成完了後、 "Explorer the db"ボタンを 押すと右の画面











- 1. DIGITSのホーム画面に移動
- 2. [New Model / Images] -> [Classification]を選択

New Image Classification	on Model	
Select Dataset 😧	Solver Options Training epochs 🕑	Data Transformations Subtract Mean <b>O</b> Image
1. "whale_faces"を選択	Snapshot interval (in epochs) 🕑	Crop Size 😧
Done 07:03:34 AM Image Size 256x256	Validation interval (in epochs) <b>O</b>	
Image Type COLOR DB backend	Random seed 😧	2. Training epochsを"5" に変更
Create DB (train)	[none]	

1. スライディング ウィンドウ モデルの作成、学習





# 1. スライディング ウィンドウ

### 1画像を用いた認識のテスト



v

#### Trained Models



Epoch #5

#### Test a single image

Image Path 😧

/home/ubuntu/data/whale/data/train/face/w\_2606.jp

#### Upload image

Browse.

Show visualizations and statistics 😧

Classify One

#### Download Model Make Pretrained Model

#### Test a list of images

Upload Image List

#### rowse...

Accepts a list of filenames or urls (you can use your val.txt file)

Image folder (optional)

Relative paths in the text file will be prepended with this value before reading

Number of images use from the file

All

Leave blank to use al

Classify Many Number of images to show per category

#### 1. Image Path を入力 /home/ubuntu/data/whale/data/train/face/w\_2606.jpg

### 2. [Classify One]ボタンを押す



#### 学習が終わったら、ジョブIDを控える

DIGITS Image Classification Model

### whale\_faces\_baseline @

Owner: smorino





#### Jupyter notebookに戻り、先ほど学習したモデルを用いて推論(インファレンス)処理を行う

In [3]:	%matplotlib inline	
	import numpy as np import matplotlib.pyplot as p 先ほど作成した、モデルのジョブIDに変更 import caffe import time JOB_NUM = '20170416-081849-e436' ## Remember to set this to be the job number for your model	
	MODEL_FILE = '/home/ubuntu/digits/digits/jobs/' + JOB_NUM + '/deploy.prototxt' PRETRAINED = '/home/ubuntu/digits/digits/jobs/' + JOB_NUM + '/snapshot_iter_270.caffemodel'	# Do not change # Do not change
	<i># Choose a random image to test against</i> RANDOM_IMAGE = str(np.random.randint(10)) IMAGE_FILE = 'data/samples/w_' + RANDOM_IMAGE + '.jpg'	
	<pre># Tell Caffe to use the GPU caffe.set_mode_gpu() # Initialize the Caffe model using the model trained in DIGITS net = caffe.Classifier(MODEL_FILE, PRETRAINED,</pre>	



### 認識結果





高精度な、クジラの顔画像分類器が作成された。

認識精度約98%

処理時間約10秒





#### 顔に対応する領域候補を生成 ニューラルネットワーク(CNN)で検出を行う





注. ハンズオンは行いません







対象となる候補領域数の削減による高速化 候補生成アルゴリズム次第で、物体の位置特定精度を向上可能



より複雑な多段処理パイプライン

候補を生成するための追加モデルの構築や学習

誤検出率は低くない

生成される候補の数によって推論時間が変わる



## 3. 全畳み込みネットワーク (FCN) <sub>考え方</sub>



スライディング ウインドウ 一つの領域で、一個の判定結果



FCN 一つの領域から、ヒートマップとして 判定結果を出力

## 3. 全畳み込みネットワーク (FCN) ハンズオン

- 検出方法1で作成した「鯨」と「非鯨」を分類するモデルを再利用
   [Clone Job]でモデルを複製
- 2. 全畳み込みネットワークに書き換える

ネットワーク定義を書きかえる (fc6 - fc8まで)

再度、学習を行う

3. 学習後のモデルを用いて推論処理を行う



## 3. 全畳み込みネットワーク (FCN) モデル作成

- 1. DIGITSのホーム画面に戻る
- 2. [Models]を選択
- 3. "whale\_faces\_baseline"をク リック (先ほど作成したモデルに移動)

Datasets (3)	Models (3)	Pretrained Mode	<b>Is</b> (0)			
				Ne	ew Dataset	New Model
					Images <del>-</del>	Images ·
Delete Group			O Filter			¢
Delete Group 3. "Wh	ale_faces	_baseline"をク	O Eilter リック	รเลเนร	elapsed	\$ - submitted ▲
Delete Group 3. "Wh name ▼ Ungrouped	ale_faces	_baseline"をク	O Eilter リック	รเลเนร	elapsed	¢ - submitted ∧
Delete Group 3. "Wh name Ungrouped whale_faces_bas	ale_faces	_baseline"をク	O Eilter リック caffe	Done	elapsed 4m	submitted ∧
Delete Group 3. "Wh name Ungrouped whale_faces_bas whale_detectnet	ale_faces	_baseline"をク	O Eilter リック Caffe Caffe	Done Done	elapsed 4m 3h	<ul> <li>✿ -</li> <li>submitted ▲</li> <li>7:36 AM</li> <li>Jul 23, 16</li> </ul>

## 3. 全畳み込みネットワーク (FCN) ハンズオン

#### 1. [Clone Job]でモデルを複製

DIGITS Image Classification Model

smorino (Logout) Info - About-

### whale\_faces\_baseline @

Owner: smorino



Job Directory /home/ubuntu/digits/digits/jobs/2017041 6-063529-0d17

Dataset

whale faces

Job Status Done

Initialized at 06:35:29 AM (1 second)

## 3. 全畳み込みネットワーク (FCN) ネットワーク定義の書き換え

- 1. [Standard Networks]->[(AlexNet横の)customize]をクリック
- 2. テキストボックス内部のネットワーク定義を書き換える

			DIGITS New Model
Standard Networks	evious Networks Pretrained Networks Cus	stom Network	
Caffe Torch			
Network	Details	customizeをクリック	
C LeNet	Original paper [1998]	28x28 (gray)	「「「C6CTC8を宜込み層に変更/TC/を削除)
e AlexNet	Original paper [2012]	256x256 Customize	Custom Network ? Visualize
○ GoogLeNet	Original paper [2014]	256x256	248 pool: MAA 249 kernel_size: 3 250 stride: 2
			251 } 252 } 253 laver {
			254name: "fc6"255type: "InnerProduct"
			256 bottom: "pool5" 257 top: "fc6" 258 page 4
			259 lr_mult: 1 260 decay mult: 1
			261 } 262 param {
			263 1r_mult: 2 264 decay mult: 0
			265 }
			206 Inner_product_param { 267 num output: 4096

## 3. 全畳み込みネットワーク (FCN) 学習の開始

### [Model Name]を"whale\_faces\_fcn"として、[Create]ボタンをクリック

Group Name 🕑	Whale_faces_fcn & Owner: mana		Clone Job Abort Job Delete Job
Model Name @ whale_faces_fcn Create  Createをクリック	Job Directory /home/ubuntu/digits/digits/jobs/20170308-101059- 4ce1 Disk Size 0 B Network (train/val) train_val.prototxt Network (deploy) deploy.prototxt Network (deploy) deploy.prototxt Network (original) original.prototxt Solver solver.prototxt Raw caffe output caffe_output.	Dataset whale_faces Done 09:37:58 AM Image Size 256x256 Image Type COLOR DB backend Imdb Create DB (train) 6814 images Create DB (val) 2272 images	Job Status Running • Initialized at 10:10:59 AM (1 second) • Running at 10:11:01 AM Train Caffe Model Running • 82% Estimated time remaining: 34 sec onds • Initialized at 10:11:00 AM (1 second) • Running at 10:11:01 AM
	0.7 0.6 0.5 0.5 0.4 0.3	-90 -90 -80 -70 -60 (g) -50 -50 -40 -40	Hardware GRID K520 (#0) Memory 2.28 GB / 4 GB (57.1%) GPU Utilization 9% Temperature 64 °C Process #3476

## 3. 全畳み込みネットワーク (FCN) <sub>推論</sub>

#### 1. 学習が終わったら、ジョブIDを控える

Image Classification Model

DIGITS

whale_faces_fcn &		2. スクリプト中のジョブIDを更新して、実行
		In [ ]: %matplotlib inline
学習済みのモデル Job Directory /home/ubuntu/digits/digits/jobs/20170416-083932- 709c Disk Size 0 B Network (train/yal)	のジョブID whale faces Done 08:13:56 AM Image Size	import numpy as np import matplot lib pyplot as plt import caffe モデルのジョブIDに変更 import copy from scipy.misc import minesize import time
train_val.prototxt Network (deploy) deploy.prototxt	256x256 Image Type COLOR	MODEL_FILE = '/home/ubuntu/digits/digits/jobs/' + JOB_NUM + '/deploy.prototxt' PRETRAINED = '/home/ubuntu/digits/digits/jobs/' + JOB_NUM + '/snapshot_iter_270.caffemodel'
Network (original) original.prototxt Solver solver.prototxt Raw caffe output	DB backend Imdb Create DB (train) 6814 images Create DB (val)	<i># Choose a random image to test against</i> RANDOM_IMAGE = str(np.random.randint(10)) IMAGE_FILE = 'data/samples/w_' + RANDOM_IMAGE + '.jpg'

### 3. 全畳み込みネットワーク (FCN) 検出結果

EARNING INSTITUTE LIVIDIA.

EEP

## 3. 全畳み込みネットワーク (FCN) まとめ



スライディングウインドウ : 約10 秒

全畳み込みネットワーク:約1.5秒

(画像の大きさに応じ、1.5~6秒程度)

#### 性質

多くの場合、FCNの方が高精度、より多くのクジラを発見。

砕ける波や海面に反射する日光により混乱することもあるが、 適切なデータ拡大を使用することにより、誤検出を軽減。



### **4. DETECTNET** 考え方: 学習時

- 1. 画像をグリッドに分割
- 2. グリッド単位でバウンディングボックス・信頼度を予測できるよう、ネットワークに学習させる



Pascal VOC 2012 images

### **4. DETECTNET** 考え方: 学習時

#### 学習用データの考え方





### **4. DETECTNET** 考え方: ネットワーク、文献



DetectNet: Deep Neural Network for Object Detection in DIGITS <a href="https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/">https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/</a>



### **4. DETECTNET** 考え方: 推論

- 1. 画像をグリッドに分割
- 2. グリッドごとに、バウンディングボックス・占有度(Coverage)を推論
- 3. バウンディングボックスをクラスタリング
- 4. 精度(mAP)を算出



### **4. DETECTNET** ハンズオン

- 1. DIGITSでDetectNetを使った鯨の位置検出を行うモデルを作成する
  - [New Datasets / Images]->[Object Detection]から、
     "whales\_detectnet"という名前でデータセットを作成する
  - 事前に準備されている"whale\_detectnet"を選択し、 [Clone Job]でモデルを複製
  - 設定を行い、再学習
- 2. DIGITSのテスト機能を使い、学習済みのモデルのテストを行う



### 4. DETECTNET データセット作成

[Datasets]->[Object Detection]を選択

2. 以下を入力

1. Training image folder "/home/ubuntu/data/whale/data\_336x224/train/images"

2. Training label folder "/home/ubuntu/data/whale/data 336x224/train/labels"

3. Validation image folder "/home/ubuntu/data/whale/data\_336x224/val/images"

4. Validation label folder "/home/ubuntu/data/whale/data\_336x224/val/labels"

5. Pad Image 値が入っている場合、削除し、空にする

#### **Object Detection Dataset Options**

Images can be stored in any of the supported file formats ('.png','.jpg','.jpeg','.bmp','.ppm'). Training image folder 🚱

/home/ubuntu/data/whale/data\_336x224/train/images

Label files are expected to have the .txt extension. For example if an image file is named foo.png the corresponding label file should be foo.txt. Training label folder 🚱

/home/ubuntu/data/whale/data 336x224/train/labels

Validation image folder 🚱

/home/ubuntu/data/whale/data 336x224/val/images

Validation label folder

/home/ubuntu/data/whale/data 336x224/val/labels

Pad image (Width x Height)

Resize image (Width x Height) @

width

Х height



1. "whales\_detectnet"という名前で、データセットを作成する





### **4. DETECTNET** 作成済みのモデルを使う

- 1. DIGITSのHome画面に戻る
- 2. "Models"を選択
- 3. "whale\_detectnet"をクリック





### **4. DETECTNET** 作成済みのモデルを使う

● エヌビディアDIGITSによる物 (× ○ Object detection)	× value_detectnet ×	그-ザ-1 _ 🗆 🗙					
← → C ① ec2-54-89-174-135.compute-1.am	nazonaws.com:5000/models/20160722-180900-cc67	🗟 🌣 🔟 :					
DIGITS Generic Image Model		smorino (Logout) Info → About →					
whale_detectne	. "Clone Job"をクリック	Clone Job Delete Job					
Job Directory	Dataset	Job Status Done					
/home/ubuntu/digits/digits/jobs/2016072 2-180900-cc67	whale full	• Initialized at Jul 22 2016, 06:09:00					
Disk SizeDone Jul 22 2016, 06:04:52 PMPM (1 second)275 MB• DB backend: Imdb• Running at Jul 22 2016, 06:09:01Network (train/val)• DB backend: ImdbPM (3 hours, 16 minutes)train_val.prototxt• Create train_db DB• Done at Jul 22 2016, 09:25:26 PM							
					Network (deploy)	• Entry Count: 2698	(Total - 3 hours, 16 minutes)
					deploy.prototxt	<ul> <li>Feature shape (3, 224, 336)</li> </ul>	Train Caffe Model Done -
original.prototxt	<ul> <li>Label shape (1, 2, 16)</li> </ul>						
Solver	Create val. db DB						



### 4. DETECTNET 学習: 3 epochのみ

New Object Detection Model

- 1. "whale\_detectnet"を選択後、 [Clone Job]でモデルを複製
- 2. 右の設定を行う

時間の都合で3 epochだけ、 学習します(8分ほどかかります)

#### Select Dataset O Solver Options Data Transform whales detectnet Training epochs 0 Subtract Mean O whale full 3 Image 1.whales\_detectnetを選択 Snapshot interval 10 Crop Size 😧 whales detectnet 2.Training epochsを3に変更 Done 10:35:58 AM Validation interval (in epochs) @ DB backend: Imdb · Create train db DB 1 Entry Count: 2698 Feature shape (3, 384, 1248) Random seed O • Label shape (1, 2, 16) · Create val db DB [none] Entry Count: 890 Feature shape (3, 384, 1248) Batch size 0 multiples allowed • Label shape (1, 2, 16) 10

3.Batch sizeを10に変更

DEEP LEARNING



- 1. モデル名を"whales\_detectnet\_2"とする
- 2. "Create"ボタンをクリックして、学習を開始する

	Group Name 😧
	Model Name 😧
1. "whales_detectnet_2"を入力	whale_detectnet_2
2."Create"ボタンをクリック	Create



# 4. DETECTNET

### 学習: 3 epoch ~ 100 epoch



100 epoch

3 epochまで



### **4. DETECTNET** 推論: 100 epoch版

- 1. DIGITSのHome画面に戻る
- 2. "Models"を選択
- 3. "whale\_detectnet"をクリック

whale\_detectnetは、100 epoch 完了時の学習結果

3. "whale\_detectnet"をクリック


## **4. DETECTNET** 推論: 100 epoch版



## **4. DETECTNET** まとめ



ほとんどのクジラの顔を 正確に検出

誤検出率が非常に低い

336x224ピクセル画像の 平均処理時間 "22ミリ秒"



## THANK YOU!

