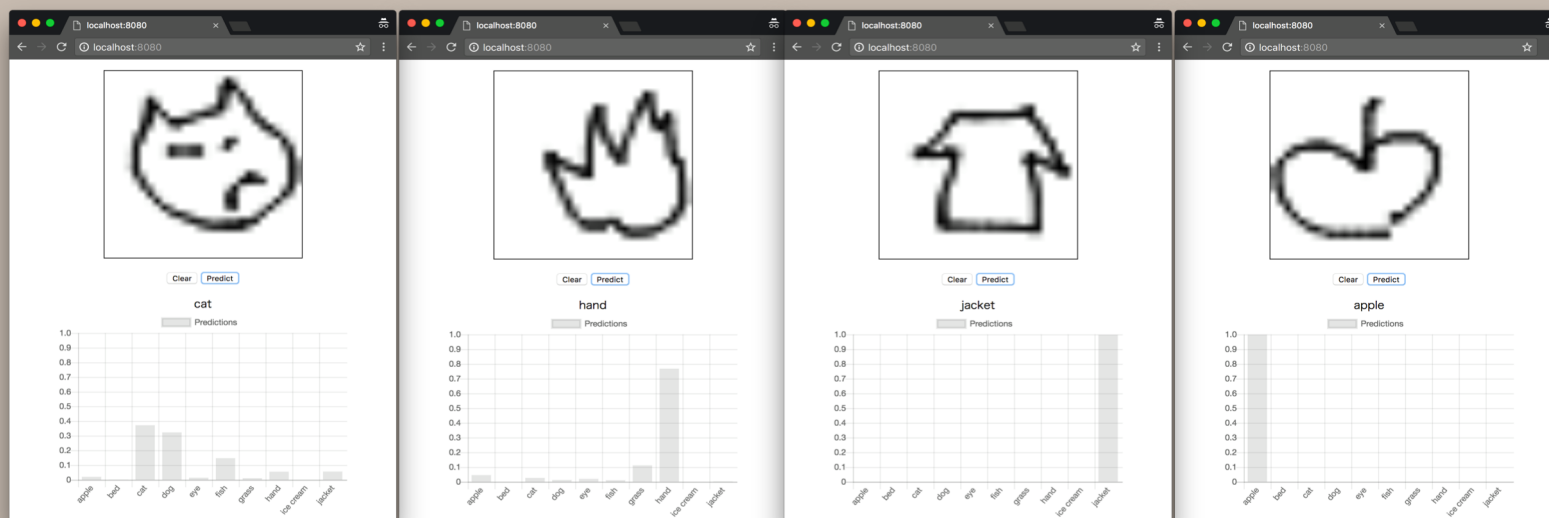


<https://goo.gl/yATZq6>

機械 / 深層学習でのデータとモデル

Amazon SageMaker と TensorFlow.jsを組み合わせる



丸山不二夫

機械学習・深層学習での サービス/アプリ開発の 現状と最近の動向

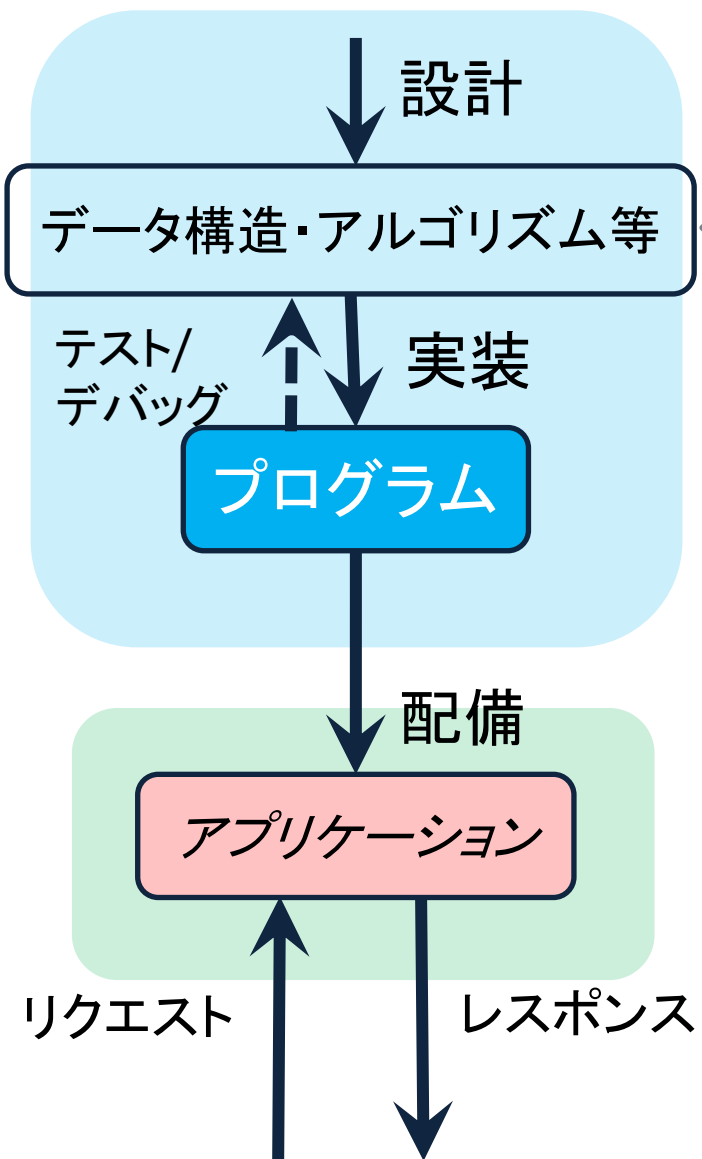
まず、基本的なことから確認しよう

機械学習・深層学習での サービス/アプリ開発

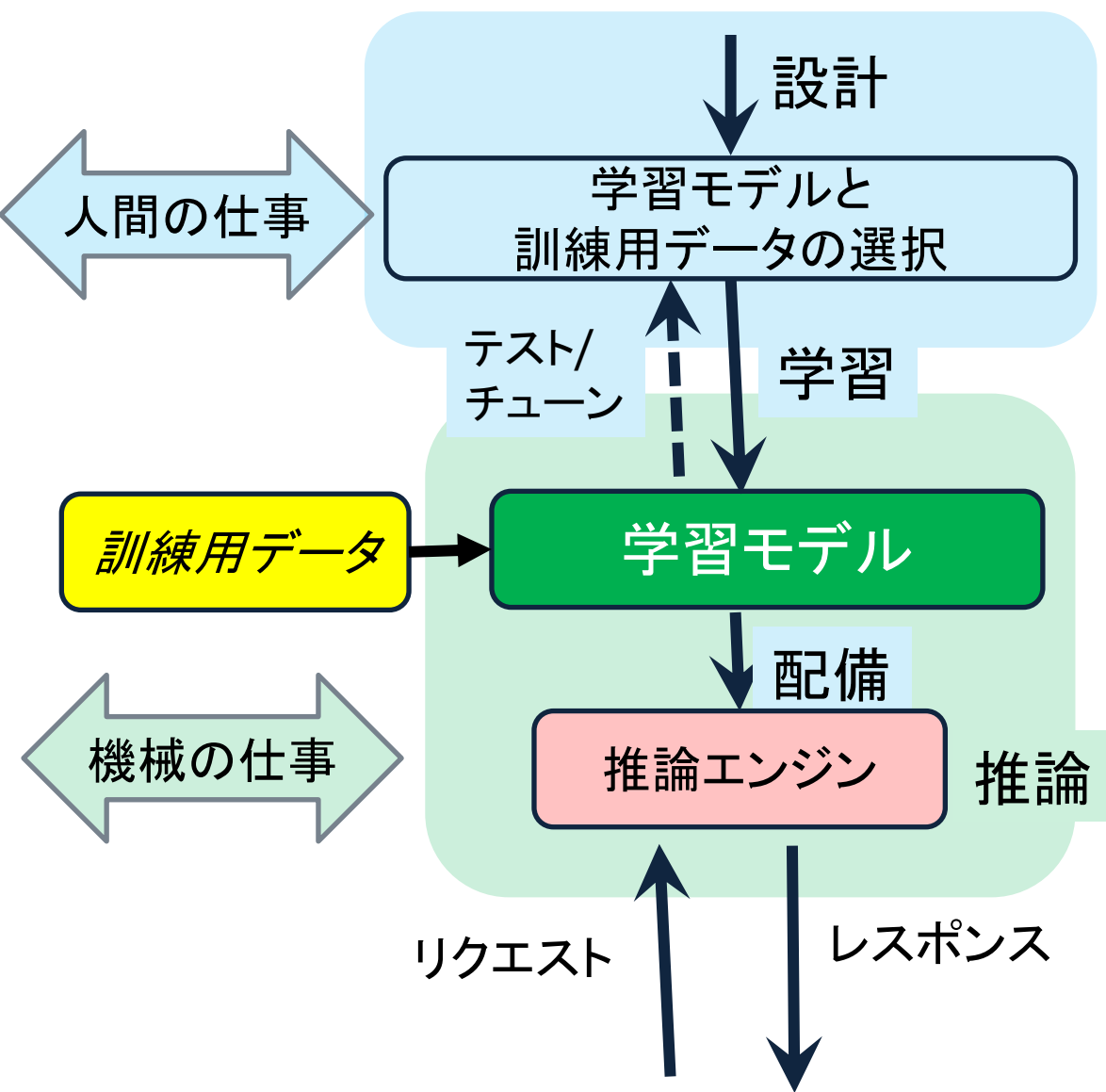
プログラミングによるサービス開発との比較

- 機械学習モデルは、基本的に統計学的モデルだが、ディープ・ラーニングのモデルは、ニューラル・ネットワーク・モデルを用いる。そうした違いはあるものの、「賢い」サービスを構成する開発のプロセスは、同じものである。
- このプロセスは、通常のプログラミング言語でサービスをプログラムする開発プロセスと比べると、人間が行う作業が圧倒的に少ない。人間が行っていたサービスのロジックをコードすることに相当する部分は、同じ働きをする推論エンジンの構築を、モデルの学習を通じて機械が代替する。(その分、CPU/GPUがフル稼働するのだが。)
- **いいモデル**が与えられて利用できるのなら、人間が行うことは、基本的には、モデルを選んで、**いいデータ**を与えることが中心になる。

プログラミングによる サービス開発

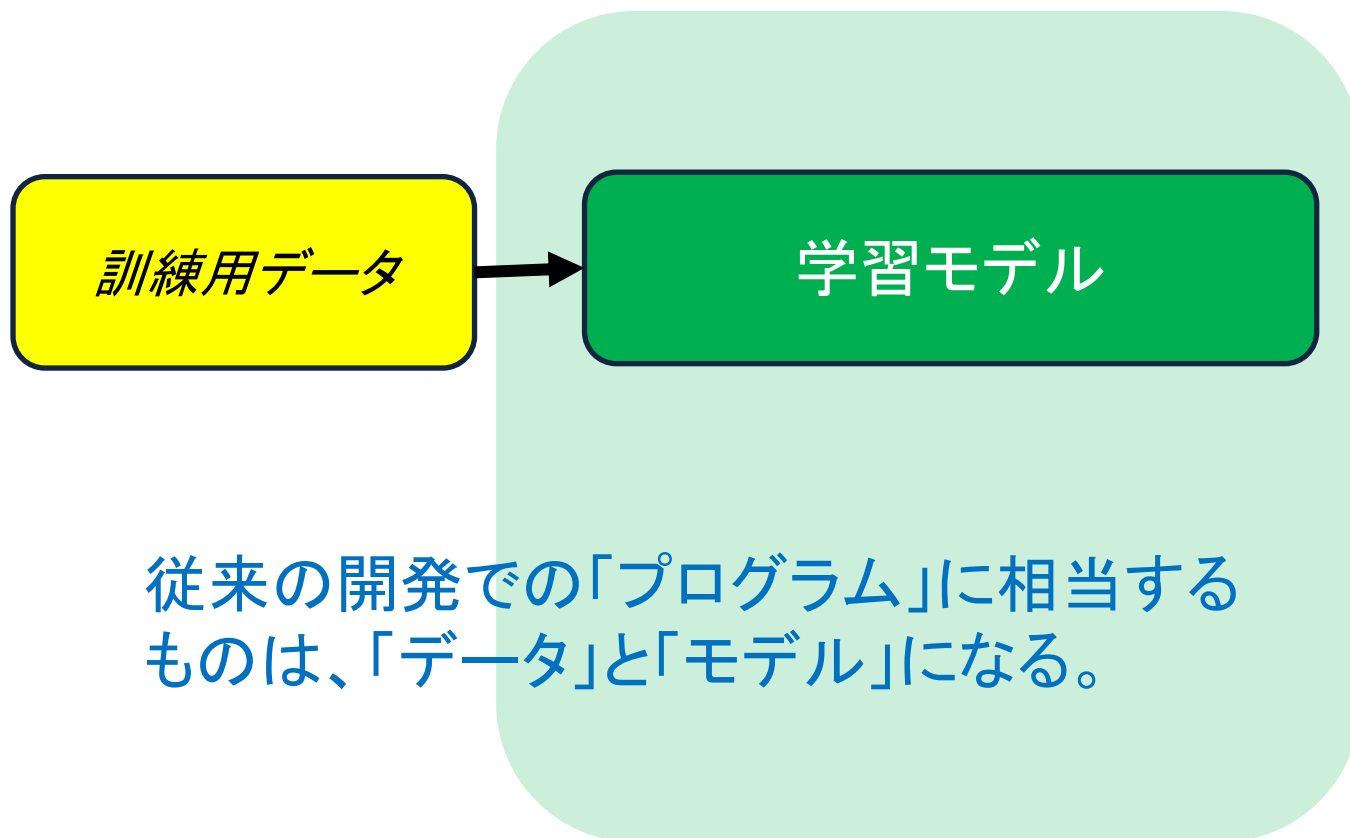


機械学習・深層学習での サービス開発



機械学習・深層学習でのサービス開発

データとモデルがポイントになる

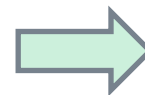


留意すべきこと

データ作りも「学習」も簡単ではない

- ❑ 従来の開発での「プログラム」に相当するものは、「データ」と「モデル」になる。ML/DLの開発では、データとモデルがポイントになる。
- ❑ 今回取り上げる開発のスタイルでは、データは、データそのものに加えて、全てのデータはその特徴を正しく表す一つの情報を持っている。「ラベル」ということが多いが、データにラベルづけをする作業は、時間とコストのかかる仕事である。
- ❑ モデルに、一度データを流し込めば、マシンが認識能力を獲得するわけではない。マシンは何度も認識に失敗する。データをモデルに読ませて認識をさせ、その結果をラベルと比較し間違っていたら、モデルのパラメータを、少しずつだが全部修正する。その後、またデータを読ませて、パラメータを修正する。こうした作業を、正しい認識ができるまで繰り返す。この「学習」の作業は、時間がかかる。

モデルの学習



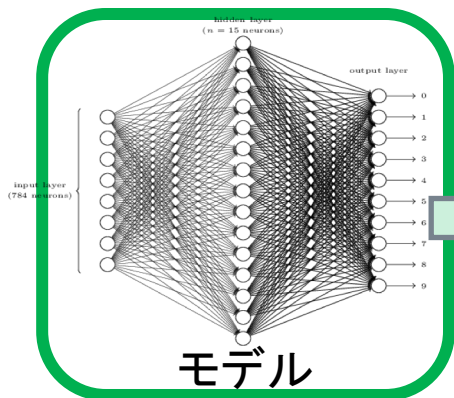
繰り返しデータを読んで
モデルのパラメータを
修正する

訓練用データ

訓練用データ

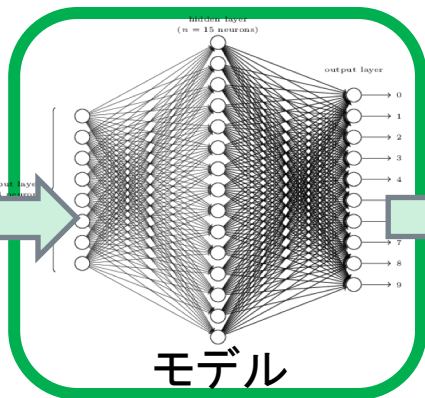
訓練用データ

訓練用データ



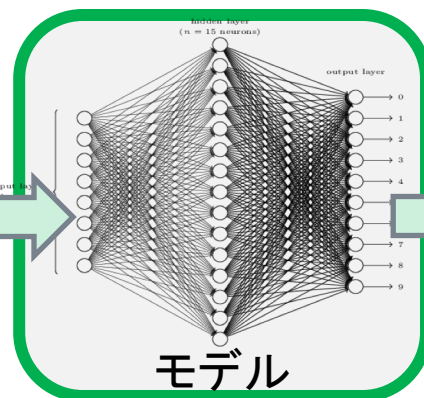
モデル

fish



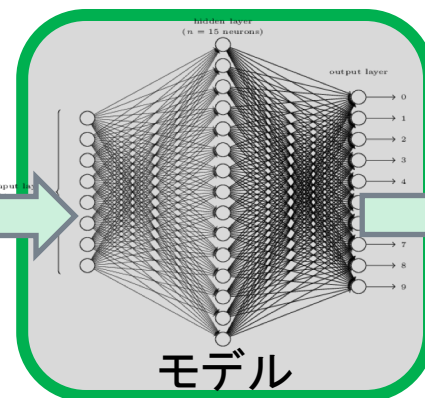
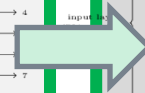
モデル

apple



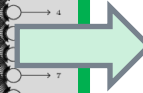
モデル

hand



モデル

fish

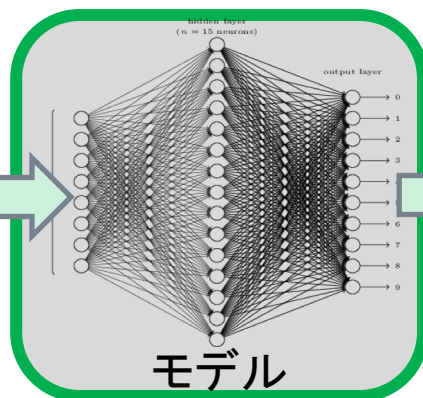


訓練用データ

訓練用データ

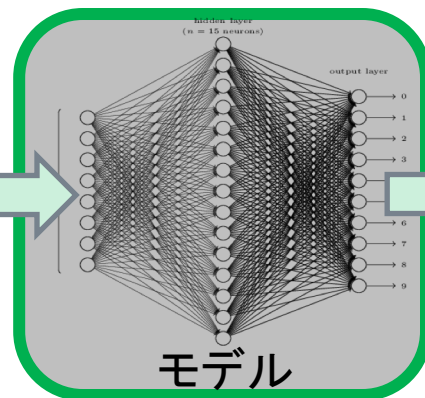
訓練用データ

訓練用データ



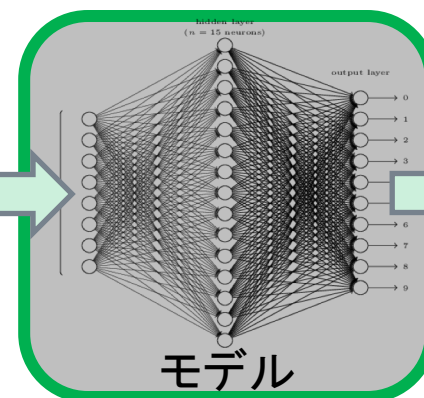
モデル

dog



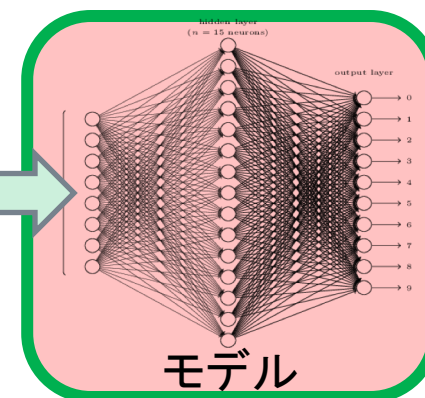
モデル

cat



モデル

cat



モデル

cat



①モデルに学習用の
データを入力し
予測を計算します

モデル

最適化

③誤差が最小化するように
モデルのパラメータを
機械的に最適化します

予測結果

apple	: 0.00
bed	: 0.00
cat	: 0.18
dog	: 0.81
eye	: 0.00
fish	: 0.00
grass	: 0.00
hand	: 0.00
ice cream	: 0.00
jacket	: 0.01

誤差

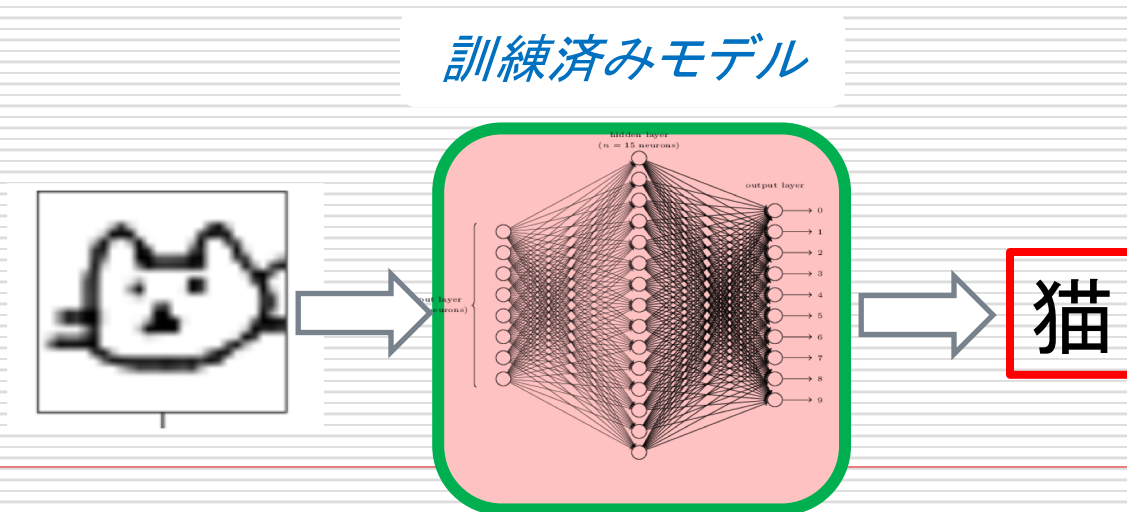
②予測結果と本当の答えの
誤差を数値化します

本当の答え
(教師データ)

apple	: 0.00
bed	: 0.00
cat	: 1.00
dog	: 0.00
eye	: 0.00
fish	: 0.00
grass	: 0.00
hand	: 0.00
ice cream	: 0.00
jacket	: 0.00

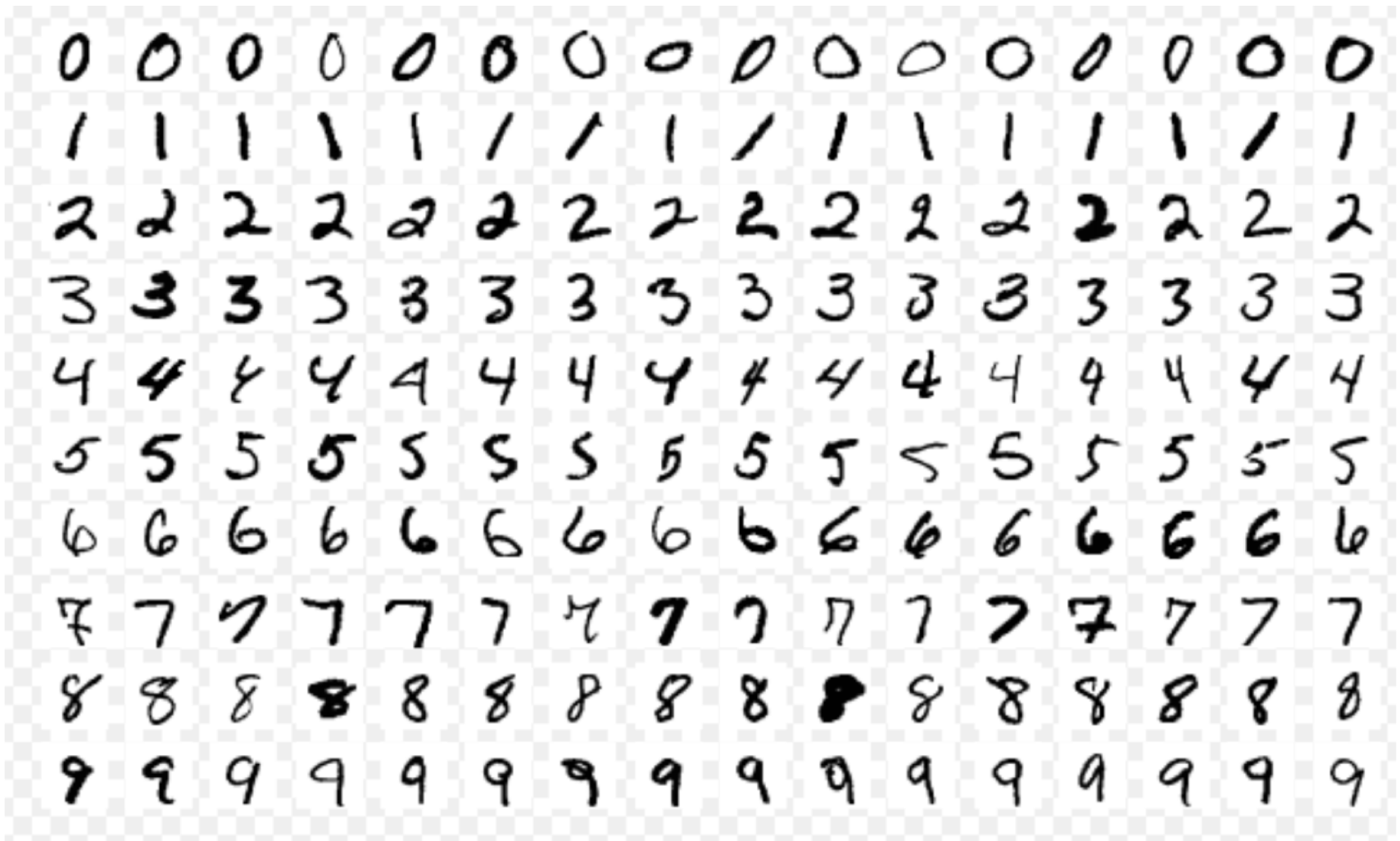
いったん、モデルの学習が済めば、 「推論」は、比較的、簡単にできる

- ただし、いったん、モデルの学習が済めば、「推論」は、比較的、簡単にできる。
- このモデルの「学習」と、学習済みモデルを利用した「推論」の処理コストの非対称性の認識は、アプリを作る上で重要である。



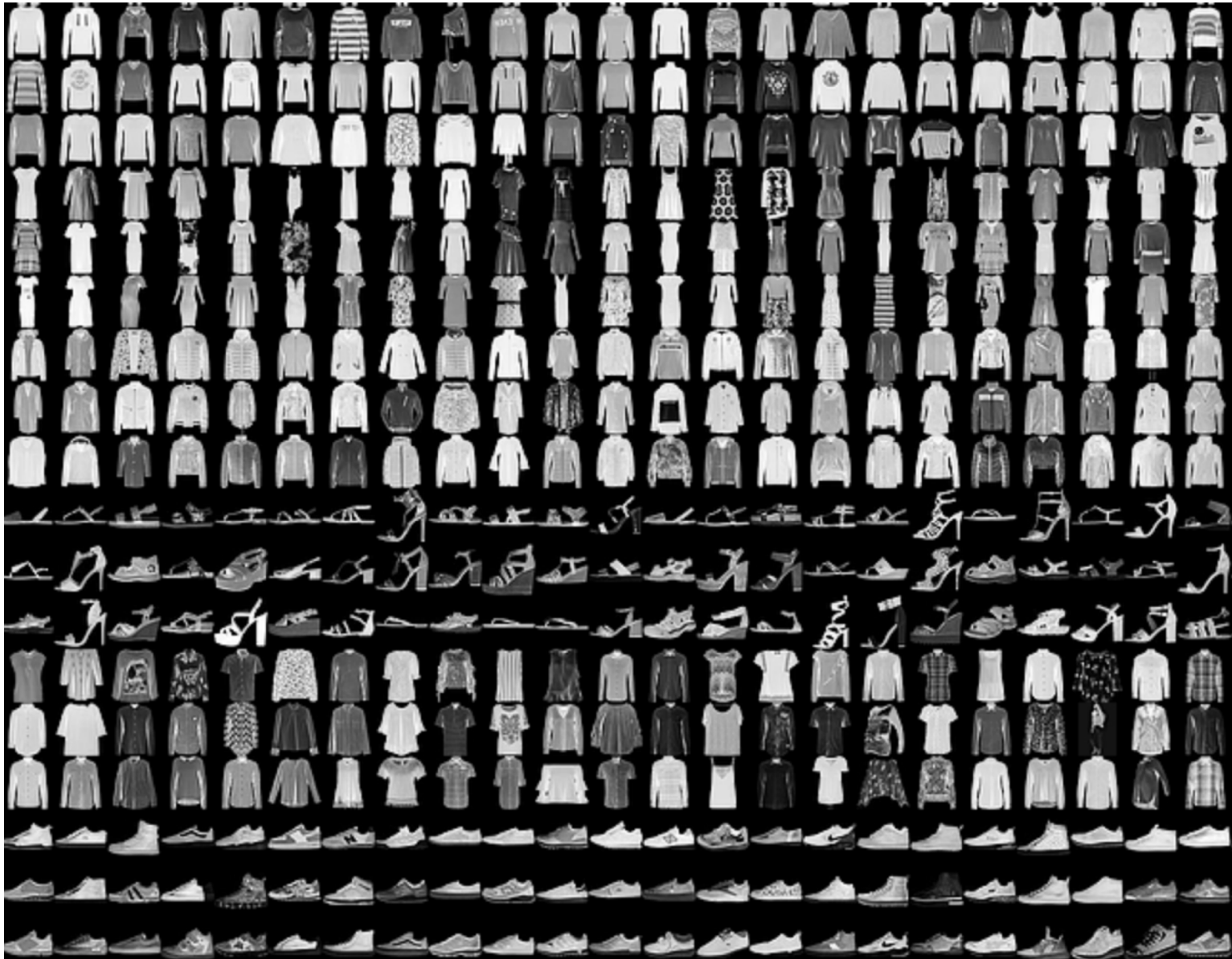
機械学習・深層学習で 利用される訓練用データの例

MNIST Dataset



<http://yann.lecun.com/exdb/mnist/>

Fashion MNIST Dataset



<https://github.com/zalandoresearch/fashion-mnist>

Quick Draw! the Data



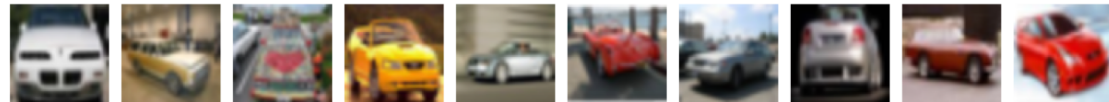
<https://quickdraw.withgoogle.com/data>

CIFAR10 Dataset

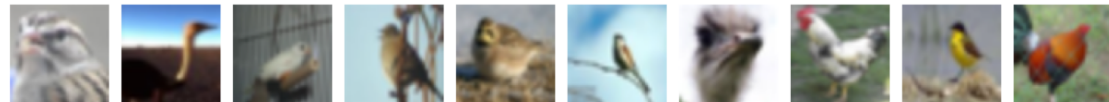
airplane



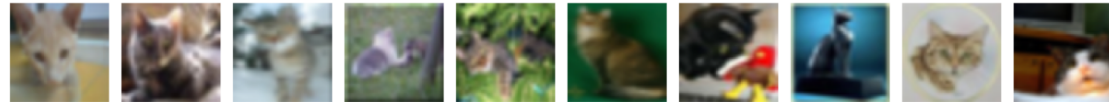
automobile



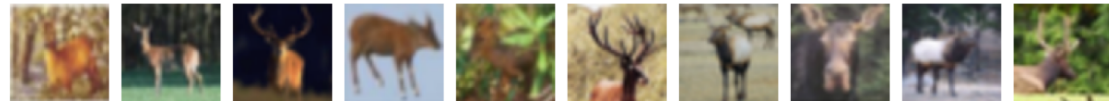
bird



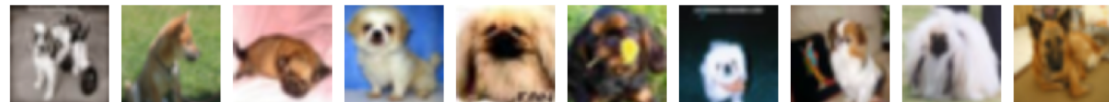
cat



deer



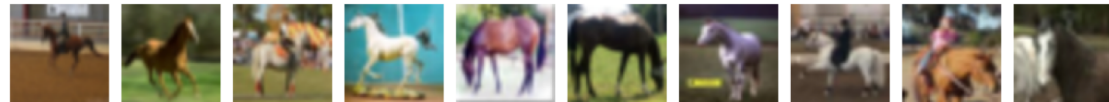
dog



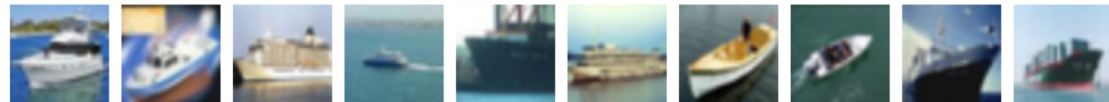
frog



horse



ship



truck



<https://www.cs.toronto.edu/~kriz/cifar.html>

IMAGENET

Numbers in brackets: (the number of synsets in the subtree).

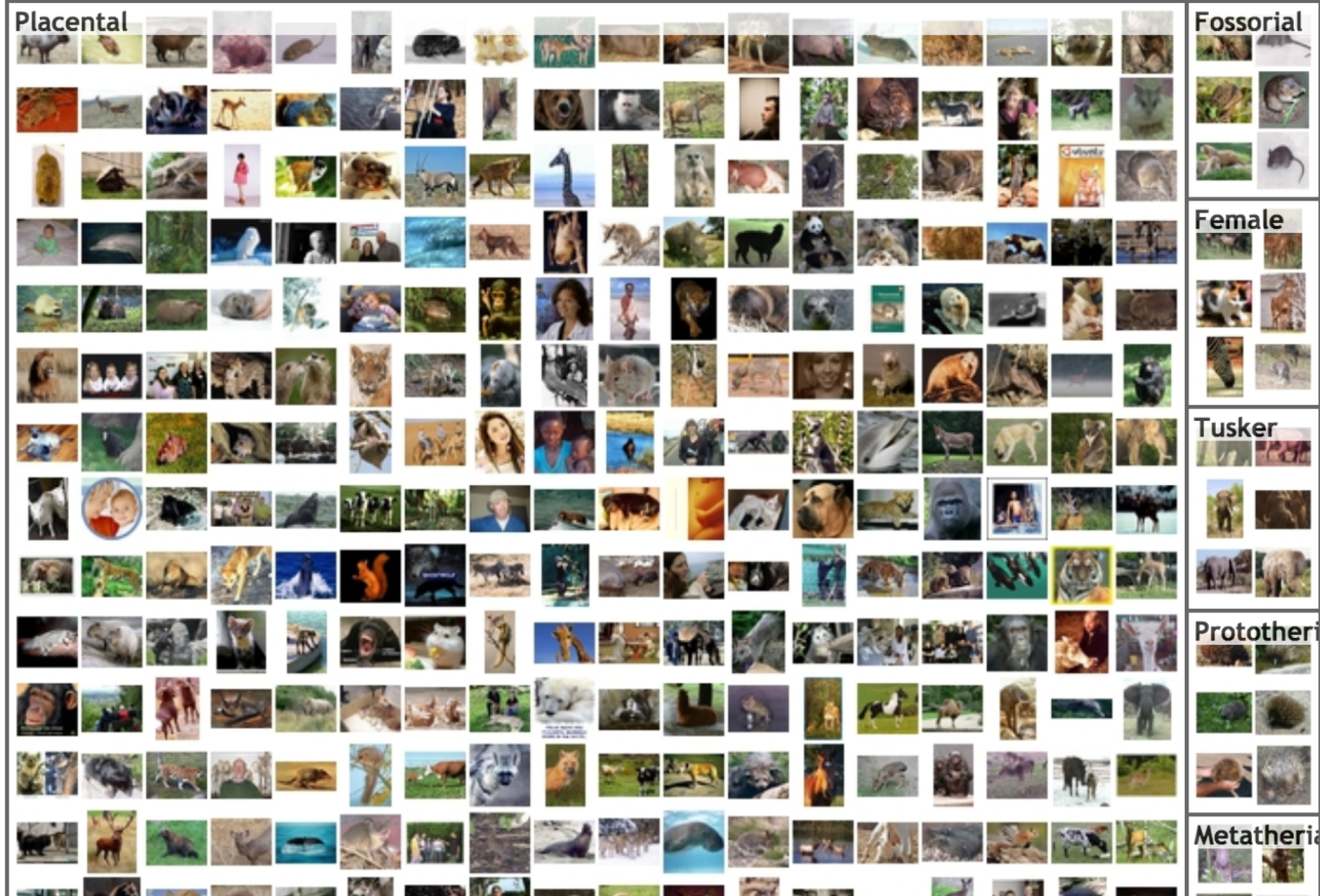
- ImageNet 2011 Fall Release (32326)
 - plant, flora, plant life (4486)
 - geological formation, formation (1)
 - natural object (1112)
 - sport, athletics (176)
 - artifact, artefact (10504)
 - fungus (308)
 - person, individual, someone, somebody (1484)
 - animal, animate being, beast, brute, creature, fauna (21841)
 - invertebrate (766)
 - homeotherm, homoiotherm, homeothermic (1)
 - work animal (4)
 - darer (0)
 - survivor (0)
 - range animal (0)
 - creepy-crawly (0)
 - domestic animal, domesticated (1)
 - molter, moult (0)
 - varmint, varment (0)
 - mutant (0)
 - critter (0)
 - game (47)
 - young, offspring (45)
 - poikilotherm, ectotherm (0)
 - herbivore (0)
 - peeper (0)

Treemap Visualization

Images of the Synset

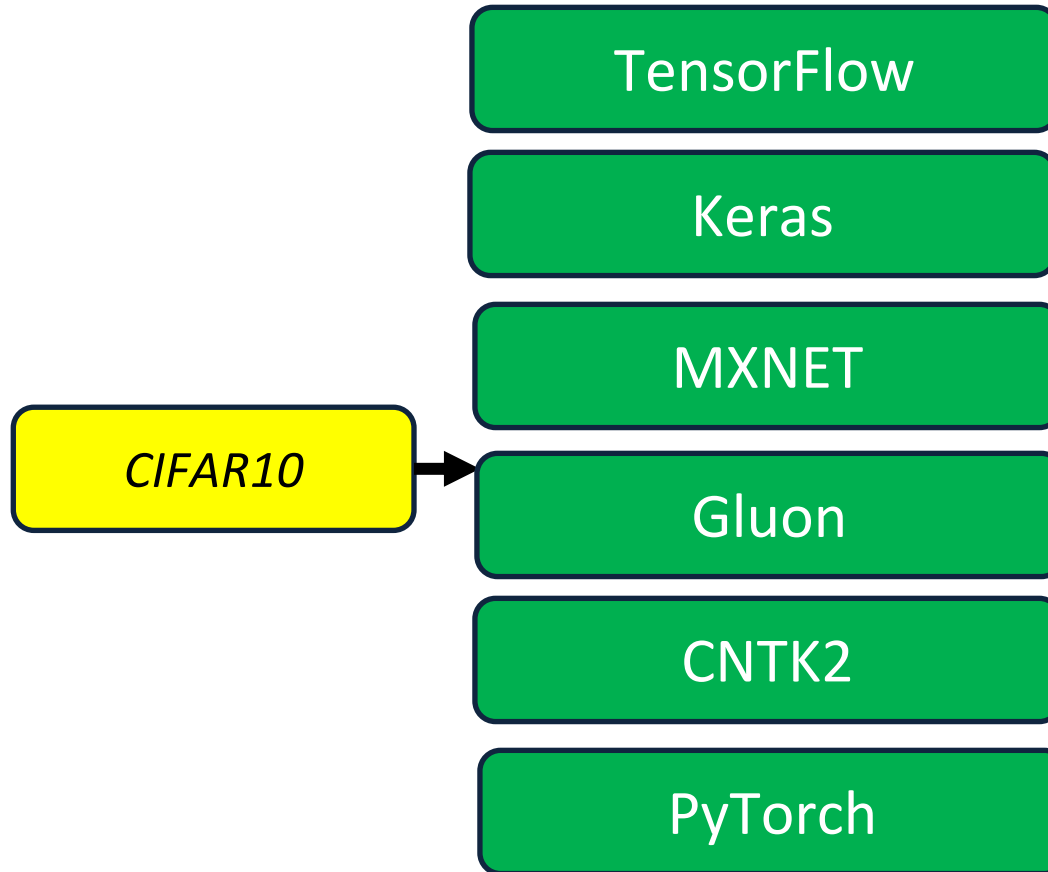
Downloads

ImageNet 2011 Fall Release > > > Vertebrate, craniate > Mammal, mammalian



<http://image-net.org/explore>

同じCIFAR10データを
複数のFrameworkのモデルで利用できる



MNIST



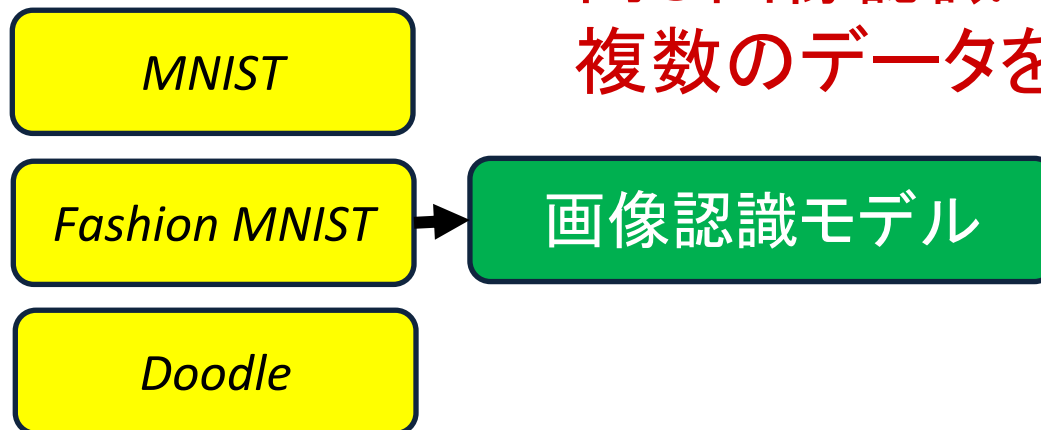
Fashion MNIST



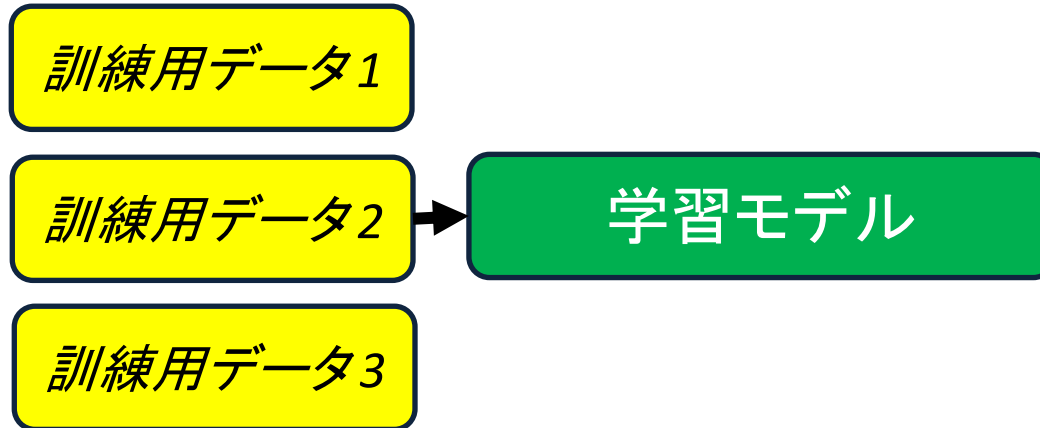
Doodle



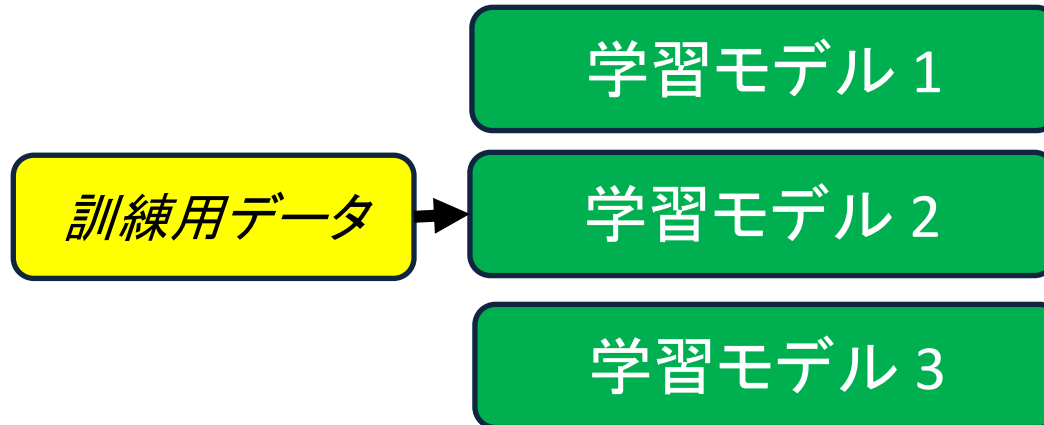
同じ画像認識モデルで
複数のデータを処理できる



同一モデル・複数データ



同一データ・複数モデル



データとモデルの選択は独立で、様々な組み合わせが可能である

現状の問題

機械学習・深層学習での
Frameworkのセグメント化

Framework

Google

TensorFlow
Keras
TensorFlow.js

ベンダーごとに、
開発用に多くの
Frameworkが
提供されている

Facebook

Torch
PyTorch
Caffe2

Amazon

MXNET

Microsoft

CNTK
CNTK2

Core ML
Chainer

Framework

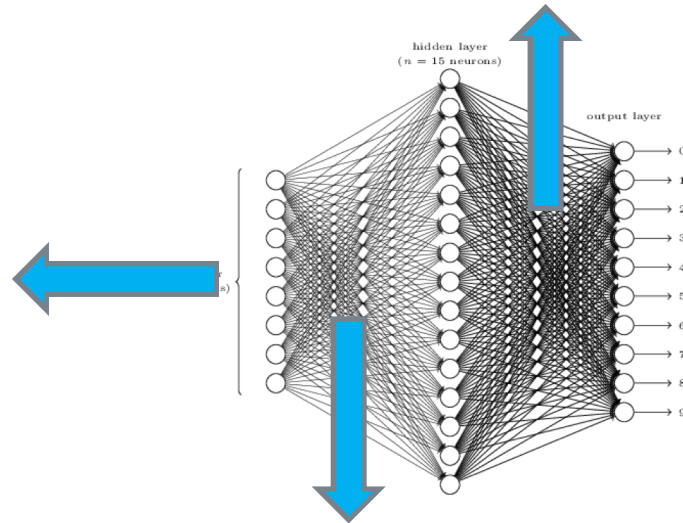
Google

TensorFlow
Keras
TensorFlow.js

同じネットワークが
フレームワーク毎に
別々の言語で、別々
に表現・実装される

Facebook

Torch
PyTorch
Caffe2



Amazon

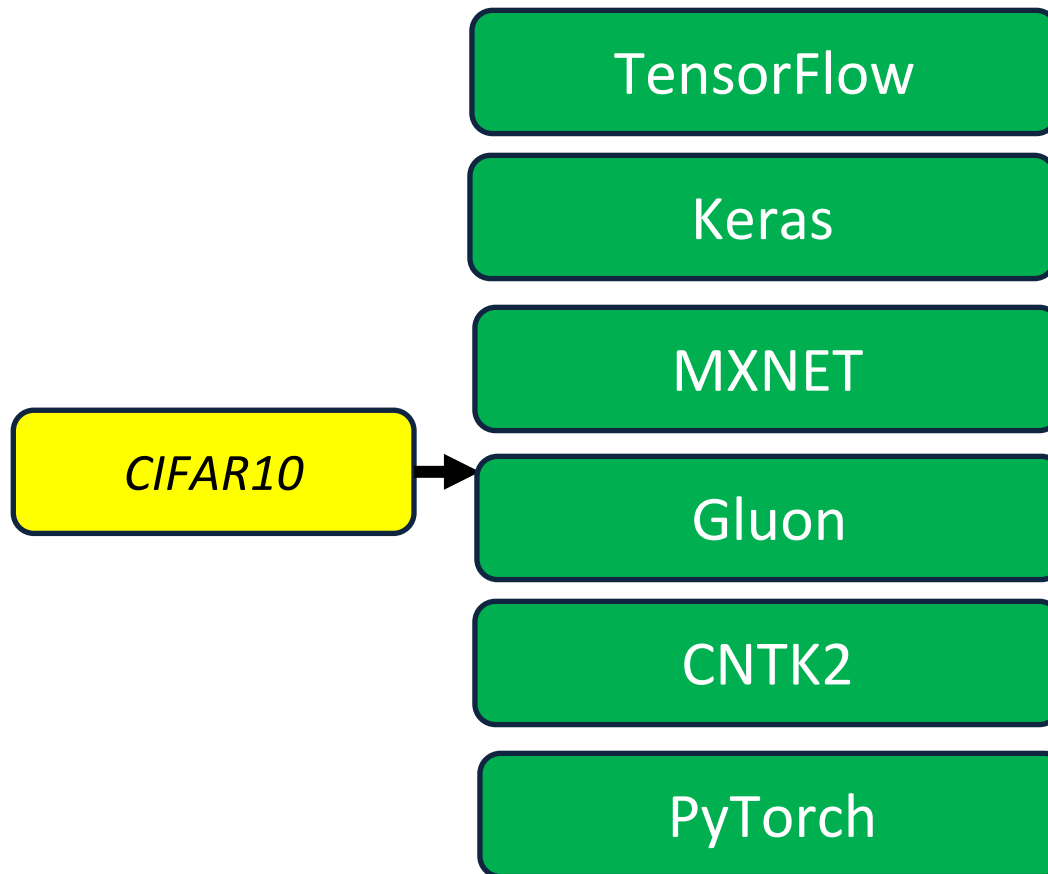
MXNET

Microsoft

CNTK
CNTK2

Core ML
Chainer

たとえば、同じCIFAR10データが
複数のFrameworkのモデルで利用される



CIFAR10 TensorFlow (一部)

```
202 # conv1
203 with tf.variable_scope('conv1') as scope:
204     kernel = _variable_with_weight_decay('weights',
205                                         shape=[5, 5, 3, 64],
206                                         stddev=5e-2,
207                                         wd=None)
208     conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding='SAME')
209     biases = _variable_on_cpu('biases', [64], tf.constant_initializer(0.0))
210     pre_activation = tf.nn.bias_add(conv, biases)
211     conv1 = tf.nn.relu(pre_activation, name=scope.name)
212     _activation_summary(conv1)
213
214 # pool1
215 pool1 = tf.nn.max_pool(conv1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
216                          padding='SAME', name='pool1')
217
218 # norm1
219 norm1 = tf.nn.lrn(pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
220                   name='norm1')
```

<https://goo.gl/9nft6Q>

CIFAR10 keras

```
34 model = Sequential()
35 model.add(Conv2D(32, (3, 3), padding='same',
36                 input_shape=x_train.shape[1:]))
37 model.add(Activation('relu'))
38 model.add(Conv2D(32, (3, 3)))
39 model.add(Activation('relu'))
40 model.add(MaxPooling2D(pool_size=(2, 2)))
41 model.add(Dropout(0.25))
42
43 model.add(Conv2D(64, (3, 3), padding='same'))
44 model.add(Activation('relu'))
45 model.add(Conv2D(64, (3, 3)))
46 model.add(Activation('relu'))
47 model.add(MaxPooling2D(pool_size=(2, 2)))
48 model.add(Dropout(0.25))
49
50 model.add(Flatten())
51 model.add(Dense(512))
52 model.add(Activation('relu'))
53 model.add(Dropout(0.5))
54 model.add(Dense(num_classes))
55 model.add(Activation('softmax'))
```

<https://goo.gl/PQmKqB>

CIFAR10 MXNET(Resnet)

```
115 data = mx.sym.BatchNorm(data=data, fix_gamma=True, eps=2e-5, momentum=bn_mom, name='bn_data')
116 (nchannel, height, width) = image_shape
117 if height <= 32:          # such as cifar10
118     body = mx.sym.Convolution(data=data, num_filter=filter_list[0], kernel=(3, 3), stride=(1,1), pad=(1, 1),
119                               no_bias=True, name="conv0", workspace=workspace)
120 else:                    # often expected to be 224 such as imagenet
121     body = mx.sym.Convolution(data=data, num_filter=filter_list[0], kernel=(7, 7), stride=(2,2), pad=(3, 3),
122                               no_bias=True, name="conv0", workspace=workspace)
123     body = mx.sym.BatchNorm(data=body, fix_gamma=False, eps=2e-5, momentum=bn_mom, name='bn0')
124     body = mx.sym.Activation(data=body, act_type='relu', name='relu0')
125     body = mx.sym.Pooling(data=body, kernel=(3, 3), stride=(2,2), pad=(1,1), pool_type='max')
126
127 for i in range(num_stages):
128     body = residual_unit(body, filter_list[i+1], (1 if i==0 else 2, 1 if i==0 else 2), False,
129                          name='stage%d_unit%d' % (i + 1, 1), bottle_neck=bottle_neck, workspace=workspace,
130                          memonger=memonger)
131     for j in range(units[i]-1):
132         body = residual_unit(body, filter_list[i+1], (1,1), True, name='stage%d_unit%d' % (i + 1, j + 2),
133                               bottle_neck=bottle_neck, workspace=workspace, memonger=memonger)
134     bn1 = mx.sym.BatchNorm(data=body, fix_gamma=False, eps=2e-5, momentum=bn_mom, name='bn1')
135     relu1 = mx.sym.Activation(data=bn1, act_type='relu', name='relu1')
136     # Although kernel is not used here when global_pool=True, we should put one
137     pool1 = mx.sym.Pooling(data=relu1, global_pool=True, kernel=(7, 7), pool_type='avg', name='pool1')
138     flat = mx.sym.Flatten(data=pool1)
139     fc1 = mx.sym.FullyConnected(data=flat, num_hidden=num_classes, name='fc1')
140     if dtype == 'float16':
141         fc1 = mx.sym.Cast(data=fc1, dtype=np.float32)
142     return mx.sym.SoftmaxOutput(data=fc1, name='softmax')
```

<https://goo.gl/UBJwvF>

CIFAR10 PyTorch (ResNet18)

```
52  # Model
53  print('==> Building model..')
54  # net = VGG('VGG19')
55  net = ResNet18()
56  # net = PreActResNet18()
57  # net = GoogLeNet()
58  # net = DenseNet121()
59  # net = ResNeXt29_2x64d()
60  # net = MobileNet()
61  # net = MobileNetV2()
62  # net = DPN92()
63  # net = ShuffleNetG2()
64  # net = SENet18()
65  net = net.to(device)
```

<https://goo.gl/HjDRZC>

CIFAR10 Gluon (ResNet64_v2)

```
34 class ResNet164_v2(nn.HybridBlock):
35     def __init__(self, num_classes, verbose=False):
36         super(ResNet164_v2, self).__init__()
37         self.verbose = verbose
38         with self.name_scope():
39             net = self.net = nn.HybridSequential()
40             # block 1
41             net.add(nn.Conv2D(64, 3, 1, 1, use_bias=False))
42             # block 2
43             for _ in range(27):
44                 net.add(Residual_v2_bottleneck(64))
45             # block 3
46             net.add(Residual_v2_bottleneck(128, same_shape=False))
47             for _ in range(26):
48                 net.add(Residual_v2_bottleneck(128))
49             # block 4
50             net.add(Residual_v2_bottleneck(256, same_shape=False))
51             for _ in range(26):
52                 net.add(Residual_v2_bottleneck(256))
53             # block 5
54             net.add(nn.BatchNorm())
55             net.add(nn.Activation('relu'))
56             net.add(nn.AvgPool2D(8))
57             net.add(nn.Dense(num_classes))
58
```

<https://goo.gl/9D5t8o>

データ名 Framework名（ネットワーク・モデル名）

CIFAR10 MXNET(Resnet)

データ名 Framework名（ネットワーク・モデル名）

CIFAR10 PyTorch (ResNet18)

データ名 Framework名（ネットワーク・モデル名）

CIFAR10 Gluon (ResNet64_v2)

Gluon

<https://github.com/gluon-api/gluon-api>

Oct 12, 2017



Framework

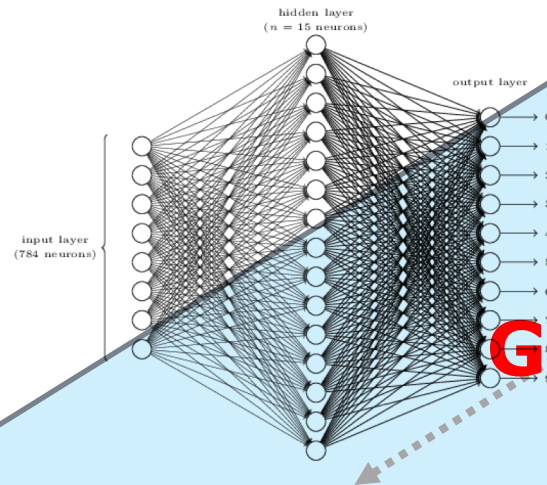
同じネットワークが
同じ表現を持たせ
ようという試み

Google

TensorFlow
Keras
TensorFlow.js

Facebook

Torch
PyTorch
Caffe2



Amazon

MXNET

Glueon

Microsoft

CNTK
CNTK2

Core ML
Chainer

MicrosoftとAWS、ディープラーニングの「Gluon」をオープンソース公開

「米Microsoftと米Amazon.com傘下のAWS(Amazon Web Services)は10月12日(現地時間)、ディープラーニングのライブラリおよびインタフェースの「Gluon」(グルーオン:クオーク同士を結合する働きがあるとされる基本粒子という意味)を[GitHub](https://github.com)で公開したと発表した。ライセンスはApache License 2.0。」

「Gluonは、様々なニューラルネットワークのコンポーネントの集まりを使って機械学習モデルを作成するための分かりやすいAPIを提供する。コンポーネントを他のデータ構造と同様に扱えるため、機械学習に慣れていない開発者でも、比較的簡単にモデルを構築できるという。」

The Gluon API Specification

- The Gluon API specification is an effort to improve speed, flexibility, and accessibility of deep learning technology for all developers, regardless of their deep learning framework of choice. The Gluon API offers a flexible interface that simplifies the process of prototyping, building, and training deep learning models without sacrificing training speed.
-

It offers four distinct advantages:

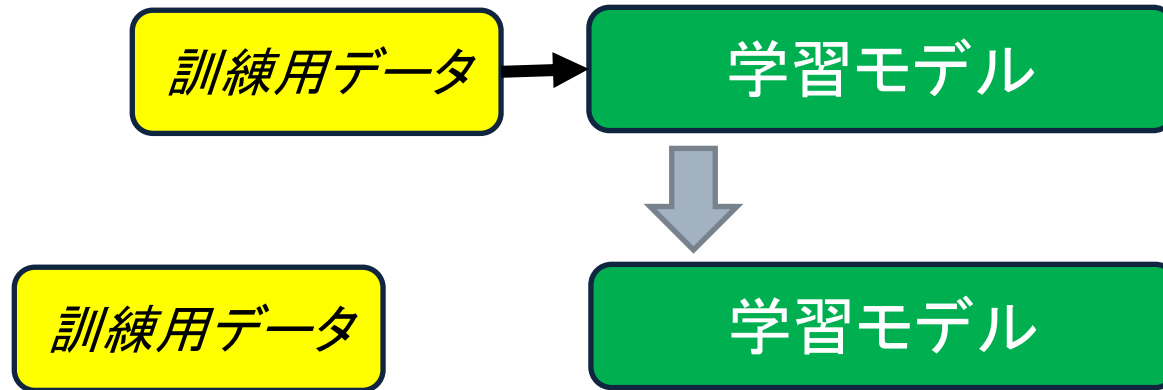
- ❑ **Simple, Easy-to-Understand Code:** Gluon offers a full set of plug-and-play neural network building blocks, including predefined layers, optimizers, and initializers.
 - ❑ **Flexible, Imperative Structure:** Gluon does not require the neural network model to be rigidly defined, but rather brings the training algorithm and model closer together to provide flexibility in the development process.
 - ❑ **Dynamic Graphs:** Gluon enables developers to define neural network models that are dynamic, meaning they can be built on the fly, with any structure, and using any of Python's native control flow.
 - ❑ **High Performance:** Gluon provides all of the above benefits without impacting the training speed that the underlying engine provides.
-

モデルのライブラリー化とModel Zoo

モデルの規格の統一化

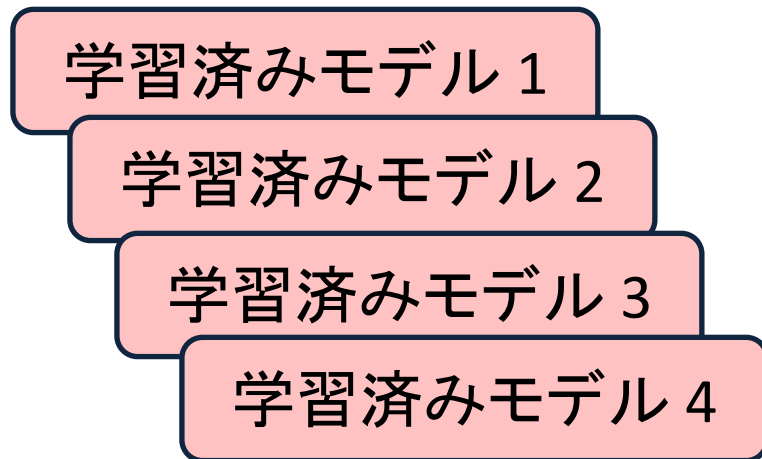
訓練済みモデルの変換とその利用

データとモデルの分離

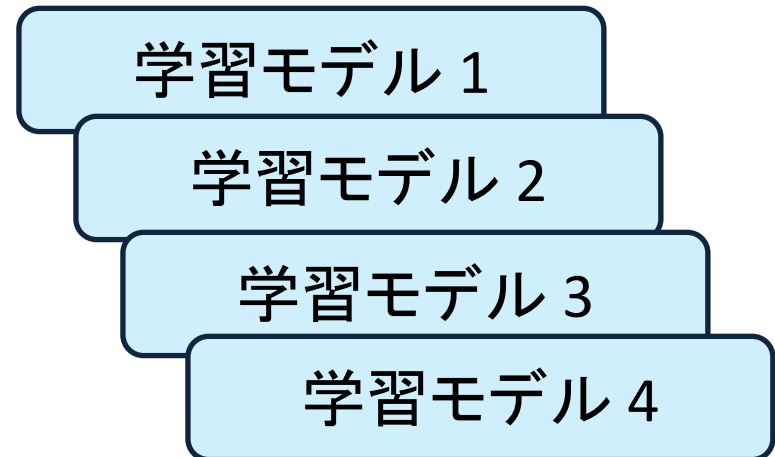


Model Zoo (Framework別)

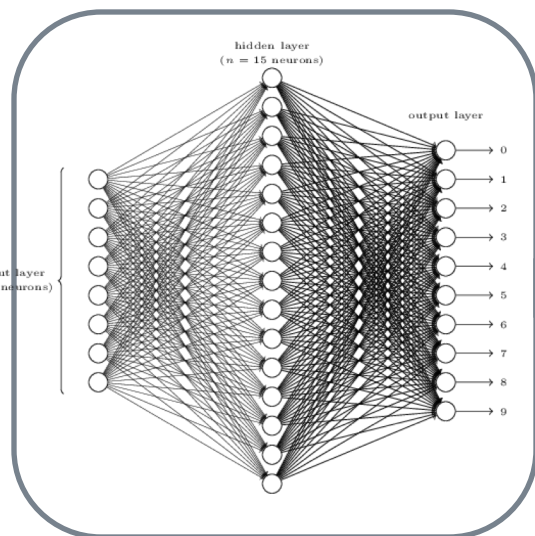
学習済みモデルの利用



モデルのライブラリー化

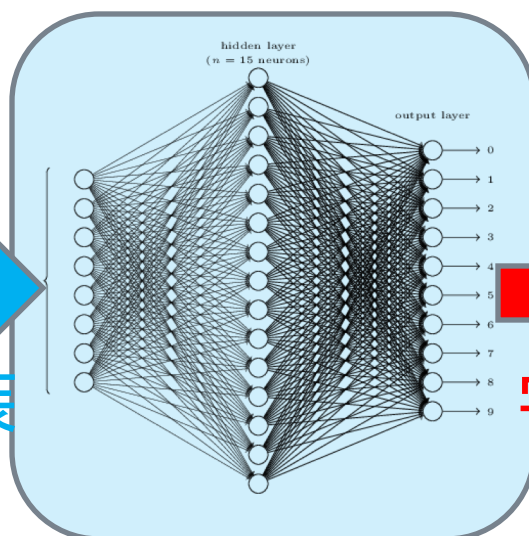


ネットワーク・モデル



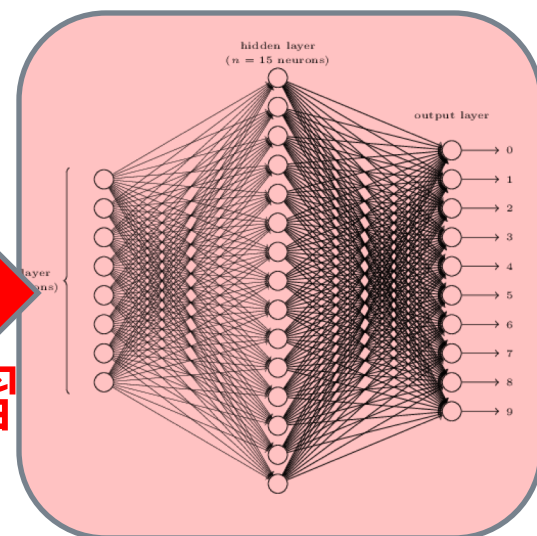
実装

Frameworkごとの 実装モデル(未学習)



学習

Frameworkごとの 学習済みモデル



ネットワーク・モデルは、モデルの「設計図」。同じタイプのモデルなら、全てのFrameworkで同じものになる。

ネットワーク・モデルは、それぞれのFrameworkのスタイル、言語に従ってコンピュータ上に「実装」される。

続いてデータを読み込んで「学習」される。

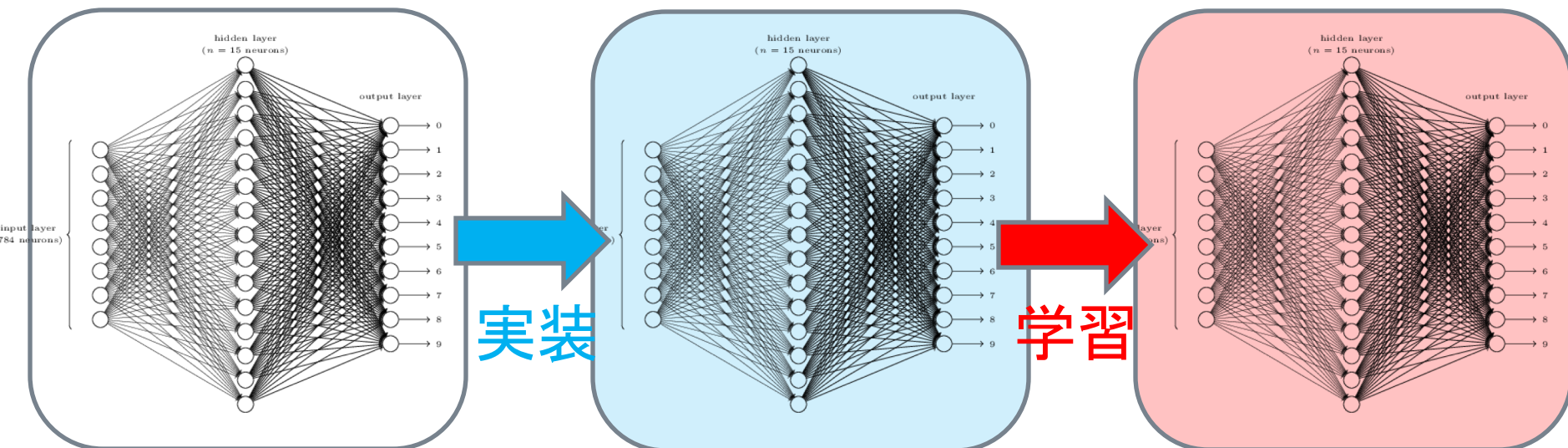
「学習」というのは、モデル上であらかじめ用意されたたくさんのパラメーターに「推論」に適した値が入れられて行くこと。

一定の繰り返しの後、学習は終わり、「学習済みモデル」が出来上がる。

ネットワーク・モデル

Frameworkごとの 実装モデル(未学習)

Frameworkごとの 学習済みモデル



```
# net = VGG('VGG19')
net = ResNet18()
# net = PreActResNet18()
# net = GoogLeNet()
# net = DenseNet121()
# net = ResNeXt29_2x64d()
# net = MobileNet()
# net = MobileNetV2()
# net = DPN92()
# net = ShuffleNetG2()
# net = SENet18()
```

A

TensorFlow

Keras

MXNET

Gluon

CNTK2

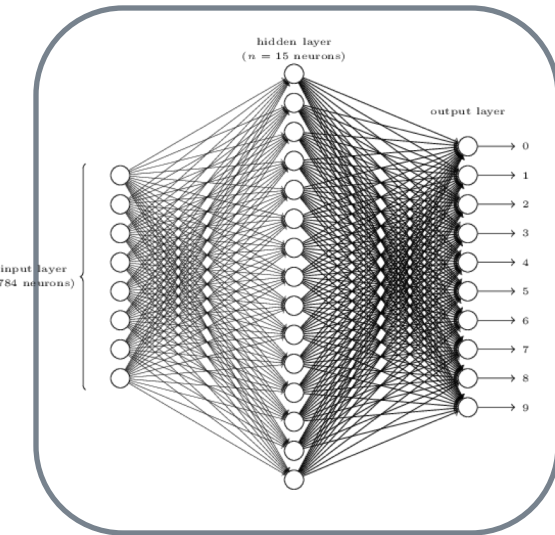
PyTorch

B

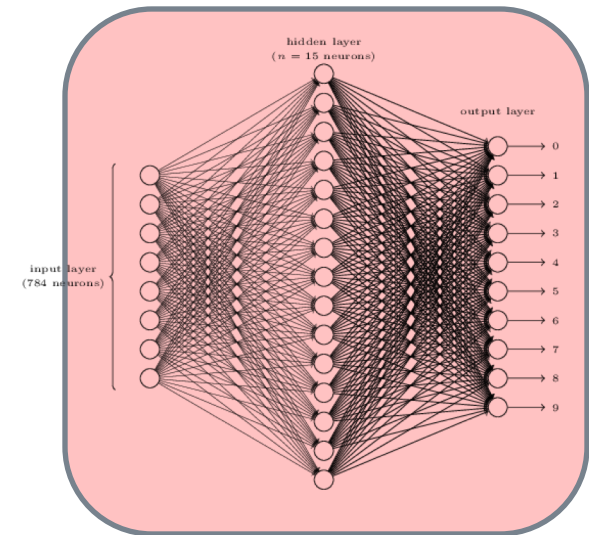
AxB

の組み合わせの
「学習済みモデル」
が、ありうる。

ネットワーク・モデル

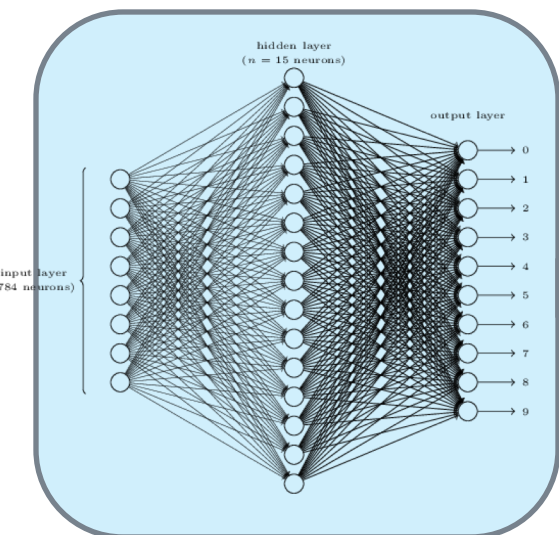


Frameworkごとの 学習済みモデル



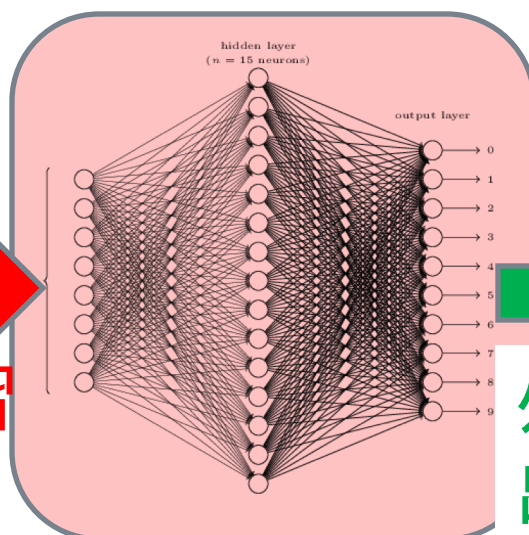
両者の関係を考えてみる。
Frameworkのスタイル、言語
による実装の違いを除けば、
両者には、基本的には、対応
関係がある。

Frameworkごとの 実装モデル(未学習)



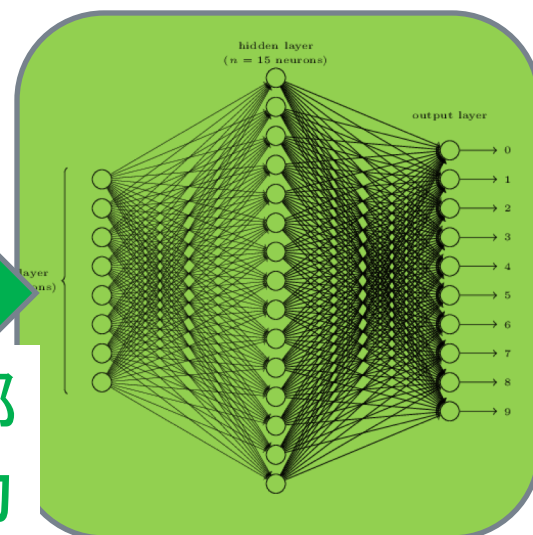
学習

Frameworkごとの 学習済みモデル



外部
出力

Saved Model



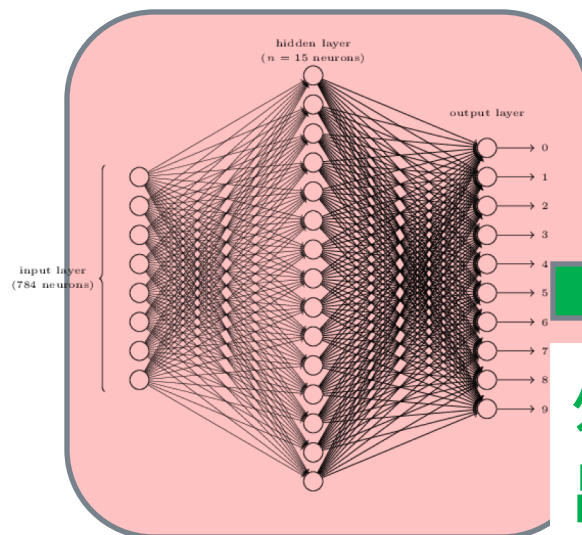
Framework上の学習済みモデルを、Frameworkの外部に出力したモデルを Saved Model という。通常、学習済みモデルというのは、この Saved Model のことである。

Frameworkごとの
学習済みモデル

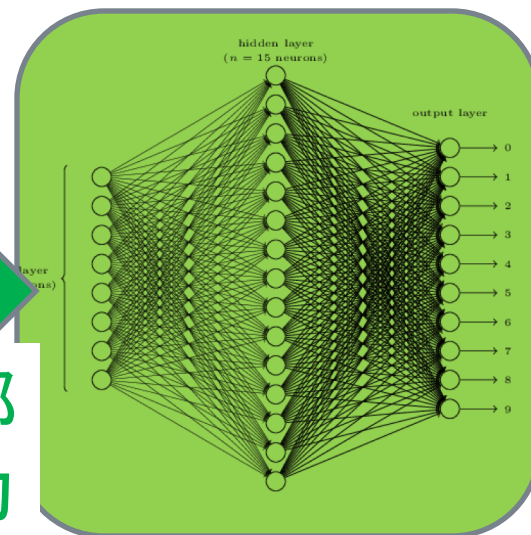
規格が統一された
Saved Model

Saved Model
の規格統一

ONNX
NNEF



外部
出力

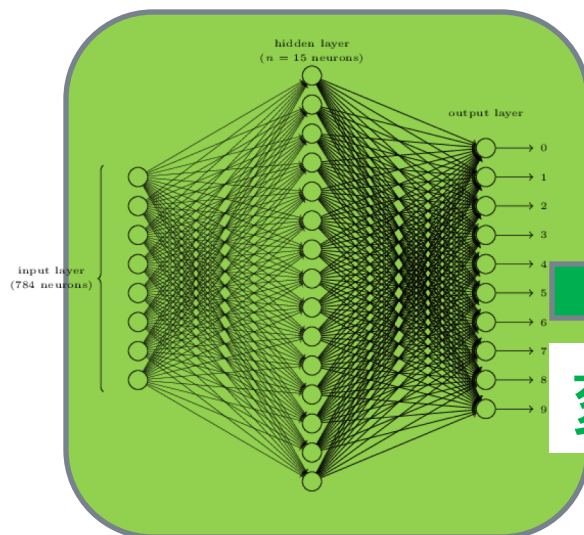


Saved Model1

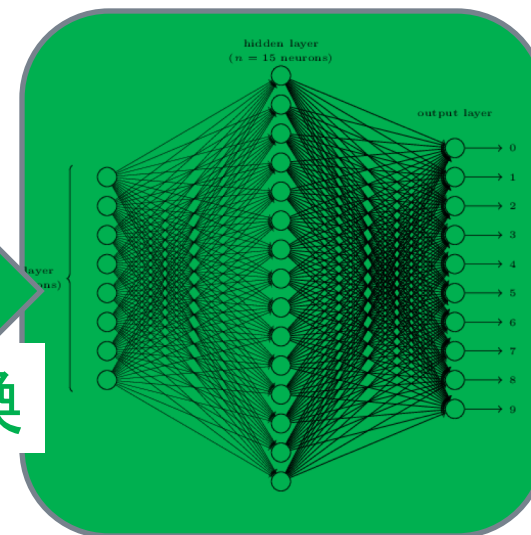
Saved Model2

Saved Model
の相互変換

Converter



変換



Converterの使用例

```
: !tensorflowjs_converter \  
  --input_format=tf_saved_model \  
  --output_node_names='probabilities,classes' \  
  --saved_model_tags=serve \  
  ./export/Servo/* \  
  ./webapp/model
```

Trained Model, IR, Converter

Google

TensorFlowHub

IR 中間表現
Intermediate
Representation

各Frameworkの Model Zoo

Facebook

Amazon

ONNX

外部出力された
モデルのレベルで
共通のフォーマット
を利用する

Microsoft

NNEF

Model Zoo

MXNet Model Zoo (訓練済みのモデル有り)

□ CNN

- [ImageNet](#)
- [CIFAR10](#)
- [PASCAL VOC](#)
- [UCF101](#)
- [Places2](#)
- [Multimedia Commons](#)

□ RNN

- [Penn Treebank \(PTB\)](#)
- [Shakespeare](#)
- [IMDB reviews](#)
- [Facebook bAbI](#)



...

https://mxnet.apache.org/model_zoo/index.html

Café Model Zoo

- ❑ Berkeley-trained models
- ❑ Network in Network model
- ❑ Models from the BMVC-2014 paper "Return of the Devil in the Details: Delving Deep into Convolutional Nets"
- ❑ Models used by the VGG team in ILSVRC-2014
- ❑ Places-CNN model from MIT.
- ❑ GoogLeNet GPU implementation from Princeton.
- ❑ Fully Convolutional Networks for Semantic Segmentation (FCNs)
- ❑ CaffeNet fine-tuned for Oxford flowers dataset
- ❑ CNN Models for Salient Object Subitizing.
- ❑ Deep Learning of Binary Hash Codes for Fast Image Retrieval

Caffe2 Model Zoo

- ❑ [bvlc_alexnet](#)
- ❑ [bvlc_googlenet](#)
- ❑ [densenet121](#)
- ❑ [detectron](#)
- ❑ [inception_v1](#)
- ❑ [inception_v2](#)
- ❑ [resnet50](#)
- ❑ [squeezenet](#)
- ❑ [vgg19](#)
- ❑ [zfnet](#)

Gluon Model Zoo (訓練済みのモデル有り)

- ☐ [AlexNet](#)
- ☐ [DenseNet](#)
- ☐ [Inception V3](#)
- ☐ [ResNet V1](#)
- ☐ [ResNet V2](#)
- ☐ [SqueezeNet](#)
- ☐ [VGG](#)
- ☐ [MobileNet](#)
- ☐ [MobileNetV2](#)

TensorFlow Hub

<https://www.tensorflow.org/hub/>

Image Modules

- ❑ For image classification, two kinds of modules are available:
 - ❑ Modules to do [image classification](#) with the particular classes that the module has been trained for.
 - ❑ Modules to extract [image feature vectors](#), (a.k.a. "bottleneck values") for use in custom image classifiers. (This is elaborated in the [image retraining tutorial](#).)
 - ❑ Click on a module to view its documentation, or reference the URL from the TensorFlow Hub library like so:
-
- ❑ `m = hub.Module("https://tfhub.dev/...")`

Image Module

- ☐ Inception V1
 - ☐ Inception V2
 - ☐ Inception V3
 - ☐ Inception-ResNet V2
 - ☐ MobileNet V1
 - ☐ MobileNet V1 instrumented for quantization with TF-Lite ("/quantops")
 - ☐ MobileNet V2
 - ☐ NASNet-A large
 - ☐ NASNet-A mobile.
 - ☐ PNASNet-5 large
 - ☐ ResNet V1 / V2
-

Text Modules

- ❑ Universal Sentence Encoder

Encoder of greater-than-word length text trained on a variety of data. [universal-sentence-encoder](#)

- ❑ ELMo

Deep Contextualized Word Representations trained on the 1 Billion Word Benchmark. [elmo](#)

- ❑ NNLM embedding trained on Google News

- ❑ Word2vec trained on Wikipedia

Other Modules

□ Generative Adversarial Networks

[progan-128](#)

□ DEep Local Features (DELF)

[delf](#)

```
import tensorflow as tf
import tensorflow_hub as hub

with tf.Graph().as_default():
    embed = hub.Module("https://tfhub.dev/google/nnlm-en-
dim128-with-normalization/1")
    embeddings = embed(["A long sentence.", "single-word",
"http://example.com"])

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

    print(sess.run(embeddings))
```

ONNX

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

Sep 7, 2017



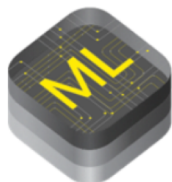
ONNX

ONNX

□ Frameworks



□ Converters



□ Runtimes



MicrosoftとFacebook、AI開発者向けオープンソースプロジェクト「ONNX」

「米Microsoftと米Facebookは9月7日（現地時間）、AI（人工知能）開発者がフレームワーク間のスイッチを簡単にできるようにするオープンソースプロジェクト「Open Neural Network Exchange（ONNX）」を公開したと発表した。[GitHub](#)で公開されてる。」

「ONNXは、あるAIフレームワークで構築した学習モデルを異なる機械学習システムに簡単に切り替えることを目的としており、Facebookの「[Caffe2](#)」と「PyTorch」、Microsoftの「Cognitive Toolkit（CNTK）」をサポートする。FacebookはONNXのために、Caffe2とPyTorchの双方をアップデートした。PyTorchのニューラルネットワークモデルをCaffe2にデプロイできる。」

Apache MXNet で ONNX をサポート

「AWS は [ONNX-MXNet](#) の利用開始を発表しました。これは Open Neural Network Exchange ([ONNX](#)) ディープラーニングモデルを Apache MXNet にインポートするためのオープンソース Python パッケージです。MXNet は充実した機能を備えたスケーラブルなディープラーニングフレームワークで、Python、Scala、R といった人気の言語に対し API を提供します。MXNet で ONNX 形式をサポートすることで、開発者は PyTorch、Microsoft Cognitive Toolkit、Caffe2 など、他のフレームワークを使用してモデルを構築したりトレーニングすることができます。また、高度に最適化されたスケーラブルなエンジンの MXNet を使用した推論に対し、こうしたモデルを MXNet にインポートすることもできます。」

<https://goo.gl/2bEWNv>

Open Neural Network Exchange (ONNX)

- [Open Neural Network Exchange \(ONNX\)](#) is the first step toward an open ecosystem that empowers AI developers to choose the right tools as their project evolves. ONNX provides an open source format for AI models. It defines an extensible computation graph model, as well as definitions of built-in operators and standard data types. Initially we focus on the capabilities needed for inferencing (evaluation).
- Caffe2, PyTorch, Microsoft Cognitive Toolkit, Apache MXNet and other tools are developing ONNX support. Enabling interoperability between different frameworks and streamlining the path from research to production will increase the speed of innovation in the AI community. We are an early stage and we invite the community to submit feedback and help us further evolve ONNX.

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

Overview

Deep learning with neural networks is accomplished through computation over dataflow graphs. Some frameworks (such as CNTK, Caffe2, Theano, and TensorFlow) make use of static graphs, while others (such as PyTorch and Chainer) use dynamic graphs. However, they all provide interfaces that make it simple for developers to construct computation graphs and runtimes that process the graphs in an optimized way. The graph serves as an Intermediate Representation (IR) that captures the specific intent of the developer's source code, and is conducive for optimization and translation to run on specific devices (CPU, GPU, FPGA, etc.).

<https://github.com/onnx/onnx/blob/master/docs/Overview.md>

Why a common IR?

Today, each framework has its own proprietary representation of the graph, though they all provide similar capabilities – meaning each framework is a siloed stack of API, graph, and runtime. Furthermore, frameworks are typically optimized for some characteristic, such as fast training, supporting complicated network architectures, inference on mobile devices, etc. It's up to the developer to select a framework that is optimized for one of these characteristics. Additionally, these optimizations may be better suited for particular stages of development. This leads to significant delays between research and production due to the necessity of conversion.

<https://github.com/onnx/onnx/blob/master/docs/Overview.md>

Why a common IR?

With the goal of democratizing AI, we envision empowering developers to select the framework that works best for their project, at any stage of development or deployment. The Open Neural Network Exchange (ONNX) format is a common IR to help establish this powerful ecosystem.

By providing a common representation of the computation graph, ONNX helps developers choose the right framework for their task, allows authors to focus on innovative enhancements, and enables hardware vendors to streamline optimizations for their platforms.

ONNX is designed to be an open format. We welcome contributions from the community and encourage everyone to adopt ONNX in their ecosystem.

<https://github.com/onnx/onnx/blob/master/docs/Overview.md>

Model Metadata

Name	Type	Format	Description
ir_version	int64		The version of the IR format specification
opset_import			Custom operator sets imported by the model
domain	string	Valid DNS name	A namespace for the model, following the style of package names, that is, reverse DNS domain name.
model_version	int64		The version of the model
producer_name	string		Name of framework/tool that generated this model
producer_version	string		Version of the framework/tool that generated this model

<https://goo.gl/Nc8tUe>

Graphs

- ❑ Each computation dataflow graph is structured as a list of nodes that form a graph. Each node is a call to an operator. Nodes have zero or more inputs, one or more outputs, and zero or more attribute-value pairs. Additionally, a valid ONNX graph must fulfill the following requirements:
 - The graph MUST be free of cycles.
 - The graph MUST be in SSA (static single assignment) form, meaning outputs of all nodes are unique.
 - The nodes list MUST be in topologically sorted order, meaning if input of node N2 is the output of node N1, N2 must appear after N1 in the nodes list.

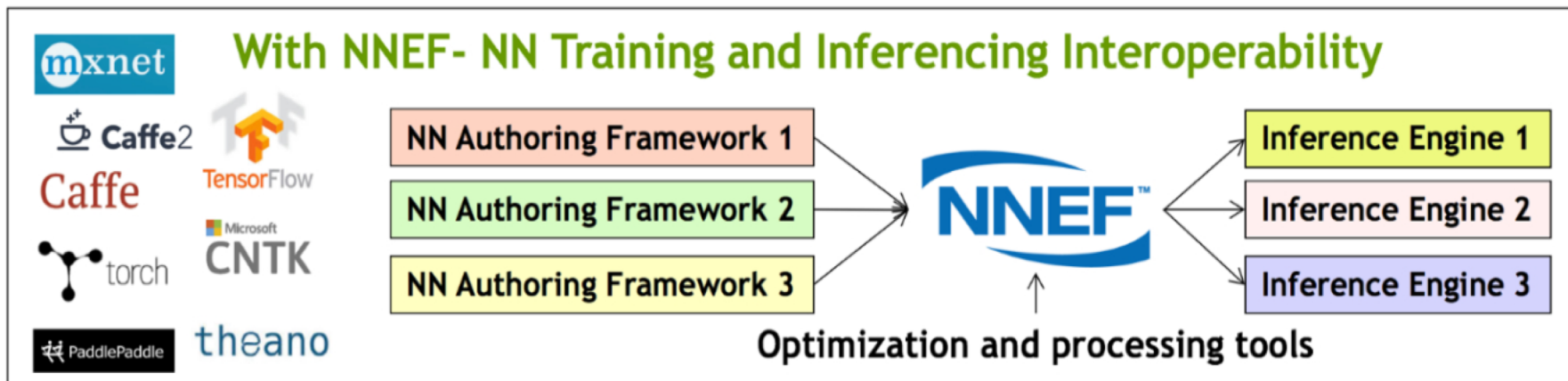
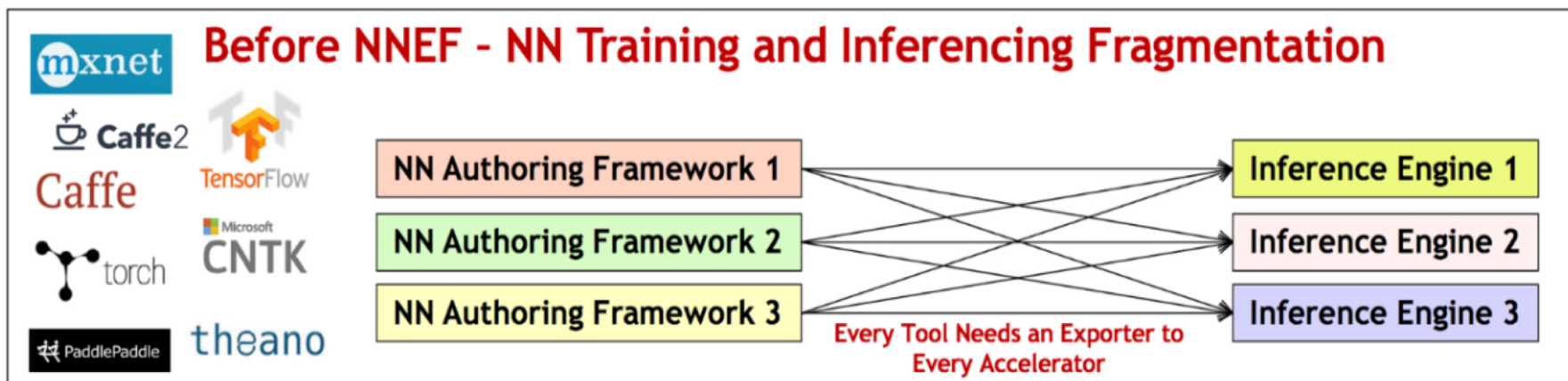
NNEF

<https://www.khronos.org/nnef>

Dec 20, 2017



NNEF - Solving Neural Net Fragmentation



Neural Network Exchange Format (NNEF)

- NNEF reduces machine learning deployment fragmentation by enabling a rich mix of neural network training tools and inference engines to be used by applications across a diverse range of devices and platforms. The release of NNEF 1.0 as a provisional specification enables feedback from the industry to be incorporated before the specification is finalized - comments and feedback are welcome on the NNEF GitHub repository.

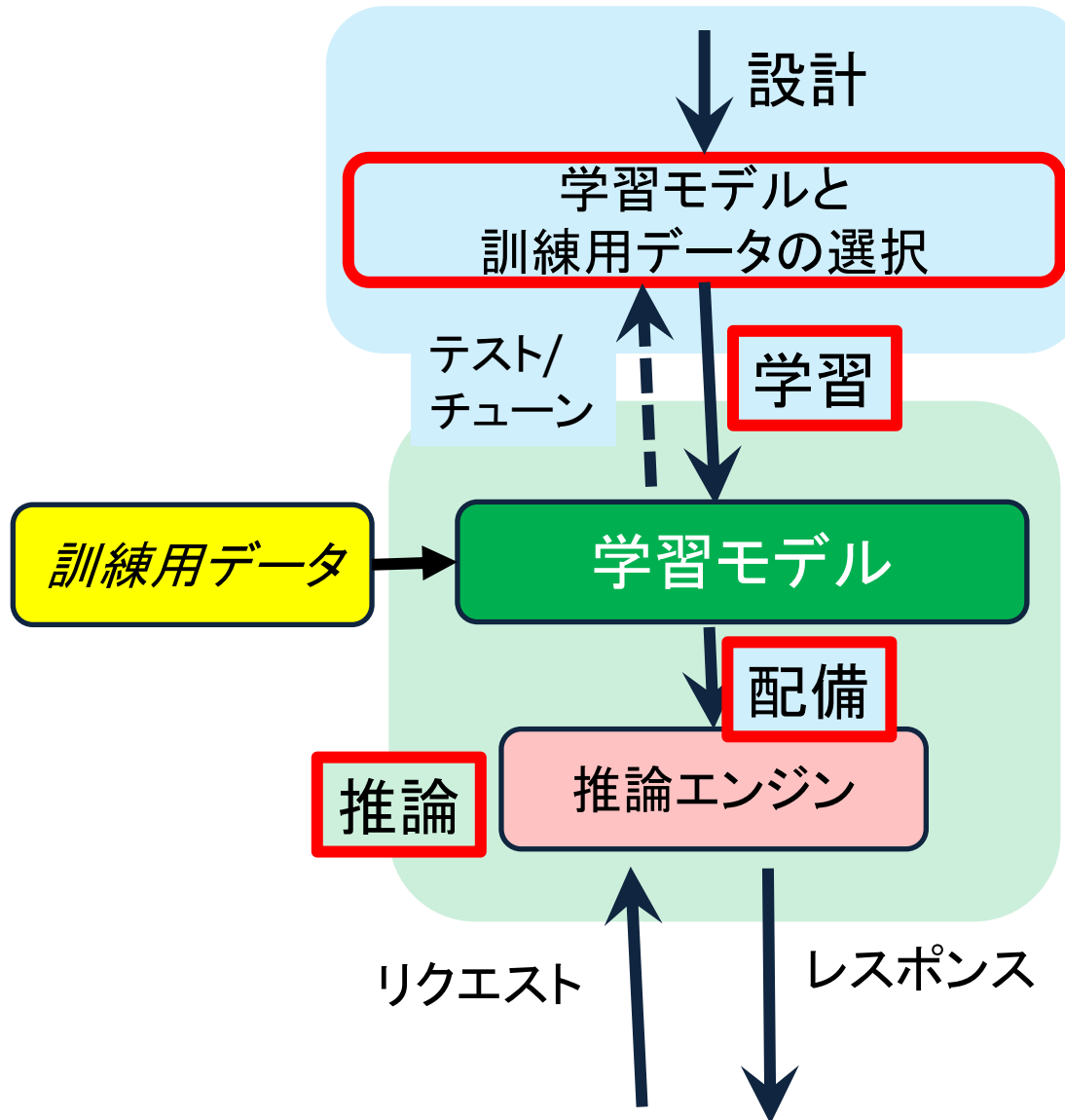
Goal of NNEF

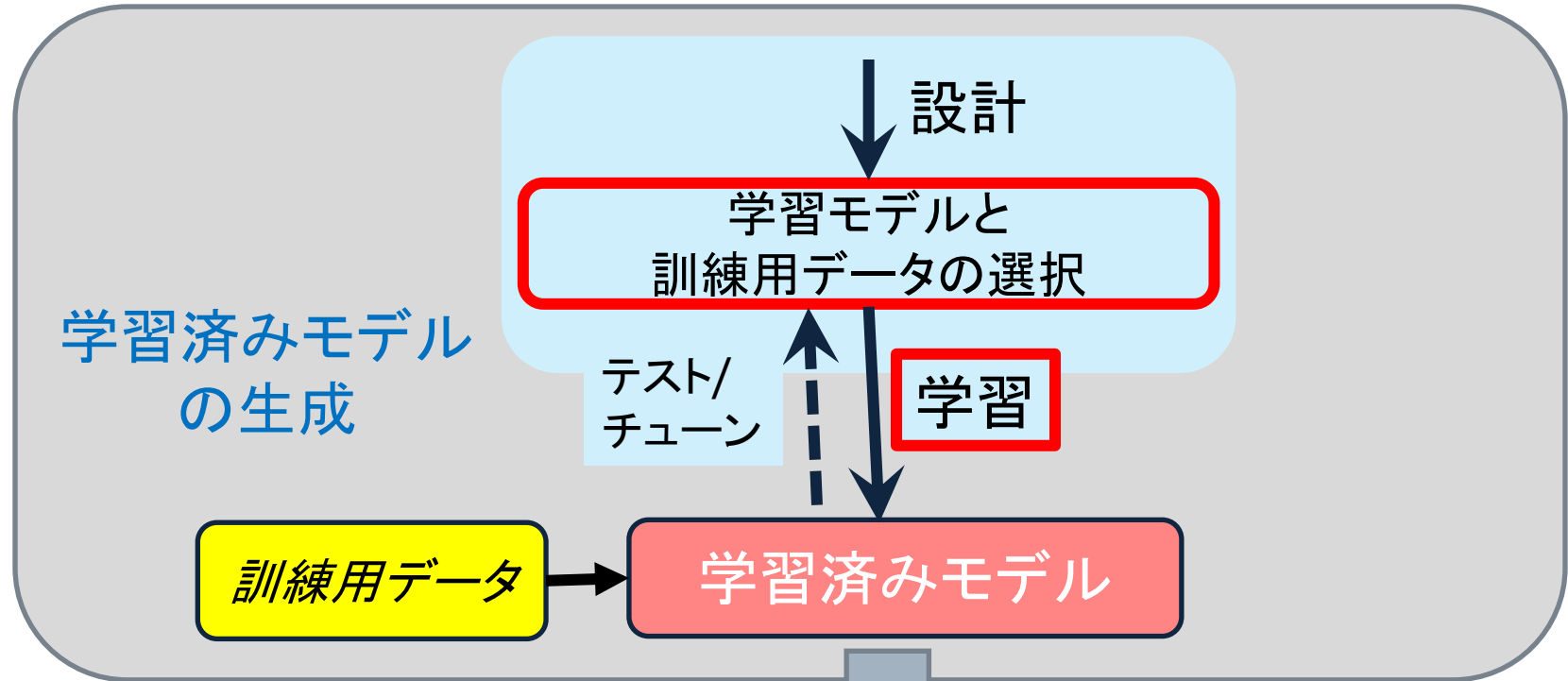
- The goal of NNEF is to enable data scientists and engineers to easily transfer trained networks from their chosen training framework into a wide variety of inference engines. A stable, flexible and extensible standard that equipment manufacturers can rely on is critical for the widespread deployment of neural networks onto edge devices, and so NNEF encapsulates a complete description of the structure, operations and parameters of a trained neural network, independent of the training tools used to produce it and the inference engine used to execute it.
-

機械学習・深層学習での アプリケーション開発

訓練済みモデルの利用
TensorFlow.jsを使ってWebアプリ化

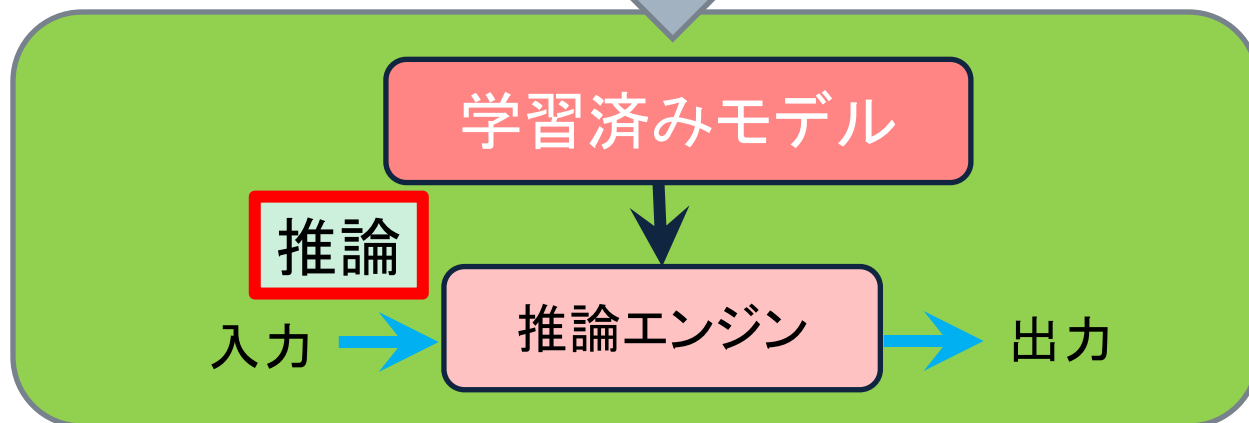
機械学習・深層学習での サービス開発



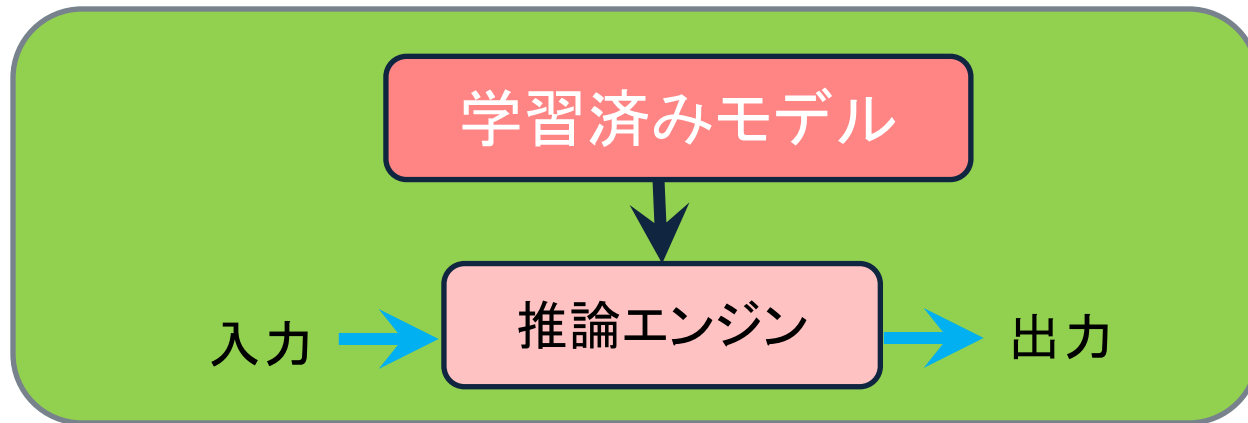


機械学習・深層学習での
アプリケーション開発

学習済みモデルの利用



学習済みモデルを利用した 機械学習・深層学習アプリケーション



TensorFlow.js を使うと、アプリを
簡単に、Webアプリ化できる

学習済みモデルをロードする

```
19 // load the pre-trained model
20 const model = await loadFrozenModel(modelUrl, weightsUrl);
21 console.log('Loaded model:', model);
```

学習済みモデルを実行して推論する

```
43 // perform recognition
44 const output = model.execute({[INPUT_NODE_NAME]: input}, OUTPUT_NODE_NAME);
45 const probabilities = output.dataSync();
46
```

本日のハンズオン資料

doodle/examples/tensorflow_js_simple/src/index.js

<https://goo.gl/rfX9fk>

Web application with Doodle image recognition via TensorFlow.js

学習済みモデルをロードする

```
52     async created() {
53         console.log('Loading model from ', this.modelUrl, this.weightsUrl)
54         try {
55             this._model = await loadFrozenModel(this.modelUrl, this.weightsUrl)
56         }
```

学習済みモデルを実行して推論する

```
67     recognize(imageData) {
68         tf.tidy(() => {
69             const img = tf.fromPixels(imageData, 1).toFloat()
70             const v255 = tf.scalar(255.)
71             const grayscaled = tf.div(tf.sub(v255, img), v255)
72             const results = this._model.execute({
73                 'image_1': grayscaled.expandDims(0)
74             })
75             this.probabilities = Array.from(results['probabilities'].dataSync())
76             this.classes = Array.from(results['classes'].dataSync())
77         })
78     }
```

[doodle/examples/tensorflow_js/src/cmsp/doodle.vue](https://github.com/doodleapp/examples/blob/master/tensorflow_js/src/cmsp/doodle.vue)

WebCam ImageNet Recognition

<https://tfjs-mobilenet-webcam.netlify.com/>

学習済みモデル(ImageNet)をロードする

```
18 window.onload = async () => {  
19  
20   const model = await loadFrozenModel(  
21     MODEL_PATH_PREFIX + MODEL_FILENAME,  
22     MODEL_PATH_PREFIX + WEIGHTS_FILENAME);  
23 }
```

学習済みモデルを実行して推論する

```
53 guessButton.addEventListener('click', () => {  
54  
55   const predictions = tf.tidy(() => {  
56     const input = webcam.capture();  
57     console.log('Invoking model.execute() with input:', input);  
58     const output = model.execute({[INPUT_NODE_NAME]: input}, OUTPUT_NODE_NAME);  
59     console.log('output:', output);  
60     return output.dataSync();  
61   });
```

<https://github.com/maru-labo/tfjs-mobilenet-webcam>

NNVM Compiler: Open Compiler for AI Frameworks

<http://www.tvmlang.org/2017/10/06/nnvm-compiler-announcement.html>

Oct 6, 2017

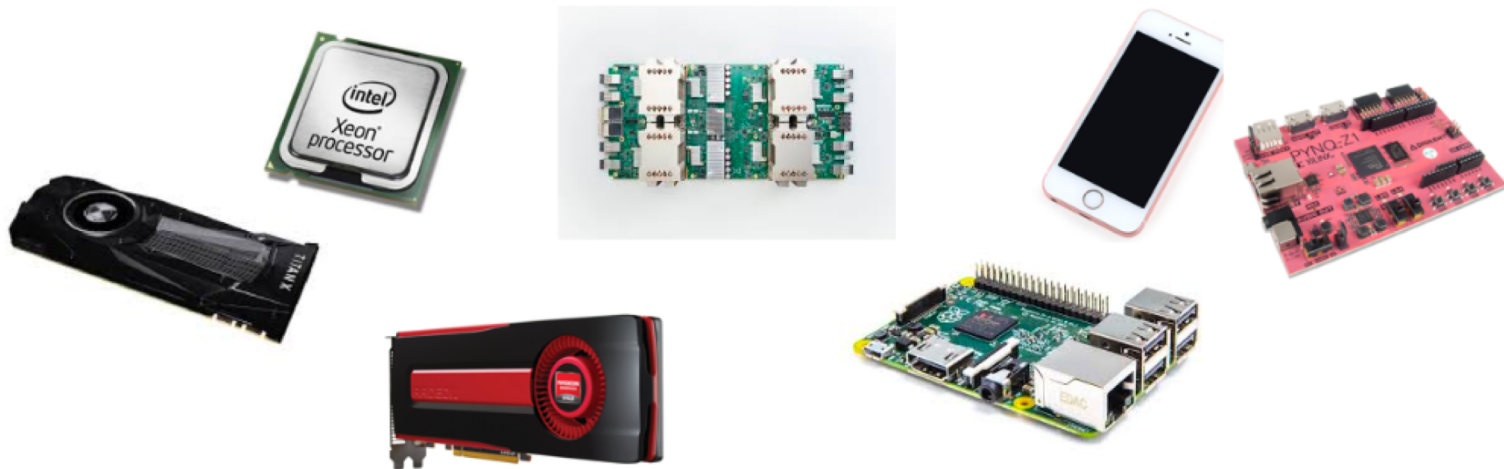
TVM: An End to End IR Stack for Deploying Deep Learning Workloads on Hardware Platforms

Deep learning has become ubiquitous and indispensable. Part of this revolution has been fueled by scalable deep learning systems, such as TensorFlow, MXNet, Caffe and PyTorch. Most existing systems are optimized for a narrow range of server-class GPUs, and require significant effort be deployed on other platforms such as mobile phones, IoT devices and specialized accelerators (FPGAs, ASICs). As the number of deep learning frameworks and hardware backends increase, we propose a unified intermediate representation (IR) stack that will close the gap between the productivity-focused deep learning frameworks, and the performance- or efficiency-oriented hardware backends.

<http://www.tvm-lang.org/2017/08/17/tvm-release-announcement.html>

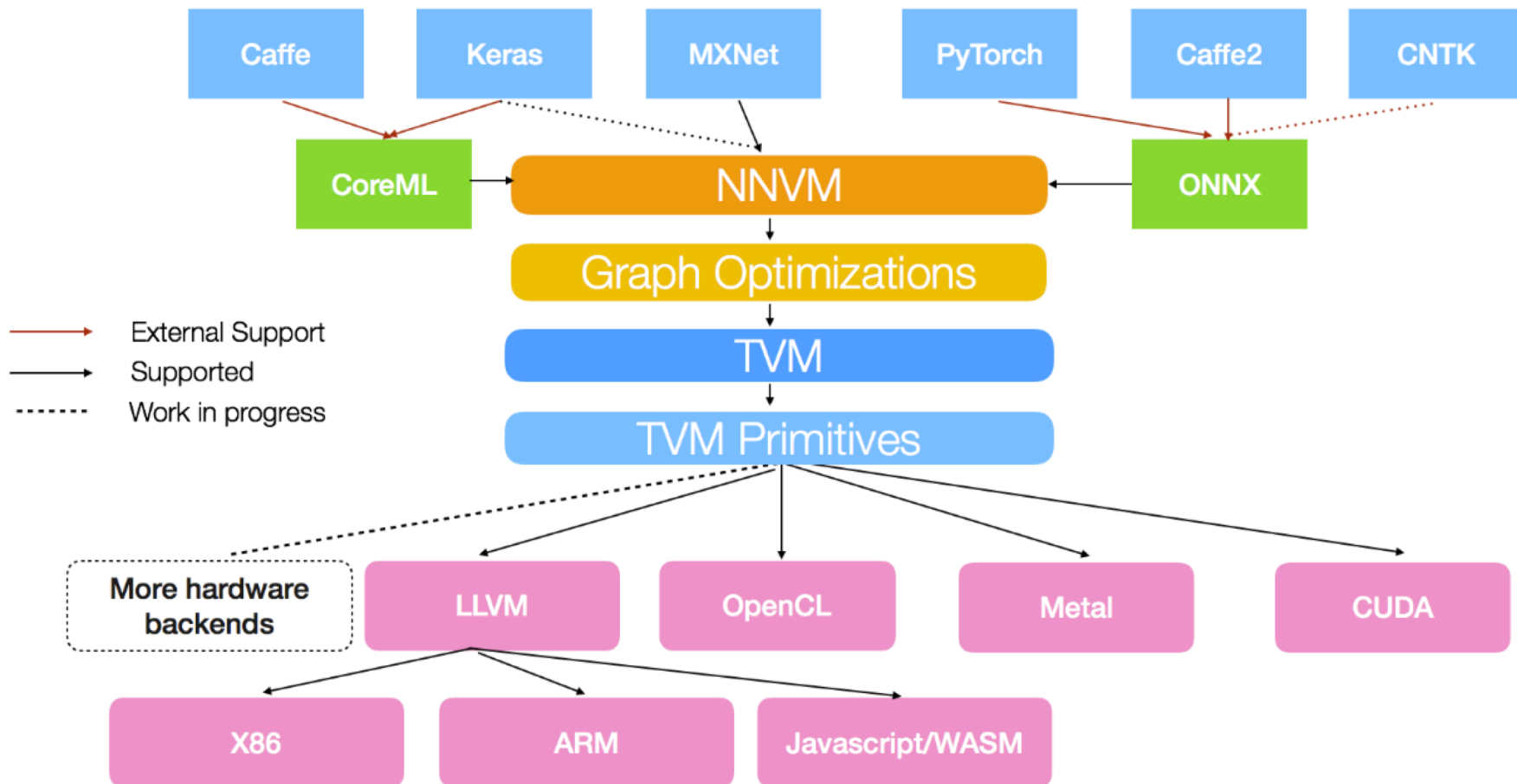
Deep Learning Frameworks

The gap between frameworks and hardware backends



NNVM Compiler: Open Compiler for AI Frameworks

Deep learning has become ubiquitous and indispensable. We are seeing a rising need for deploying deep learning workloads on many kinds of platforms such as mobile phones, GPU, IoT devices and specialized accelerators. Last month, we announced TVM stack to close the gap between deep learning frameworks, and the performance- or efficiency-oriented hardware backends. TVM stack makes it easy to build an end to end compilation for a deep learning framework. However, we think it would even be better to have a unified solution that works for all frameworks.



Amazon SageMaker

SageMaker:Platform

Google

TensorFlow
Keras
TensorFlow.js

Facebook

Torch
PyTorch
Caffe2

SageMaker

基本的には、主要な
プラットフォーム全てに
対応した開発環境

Amazon

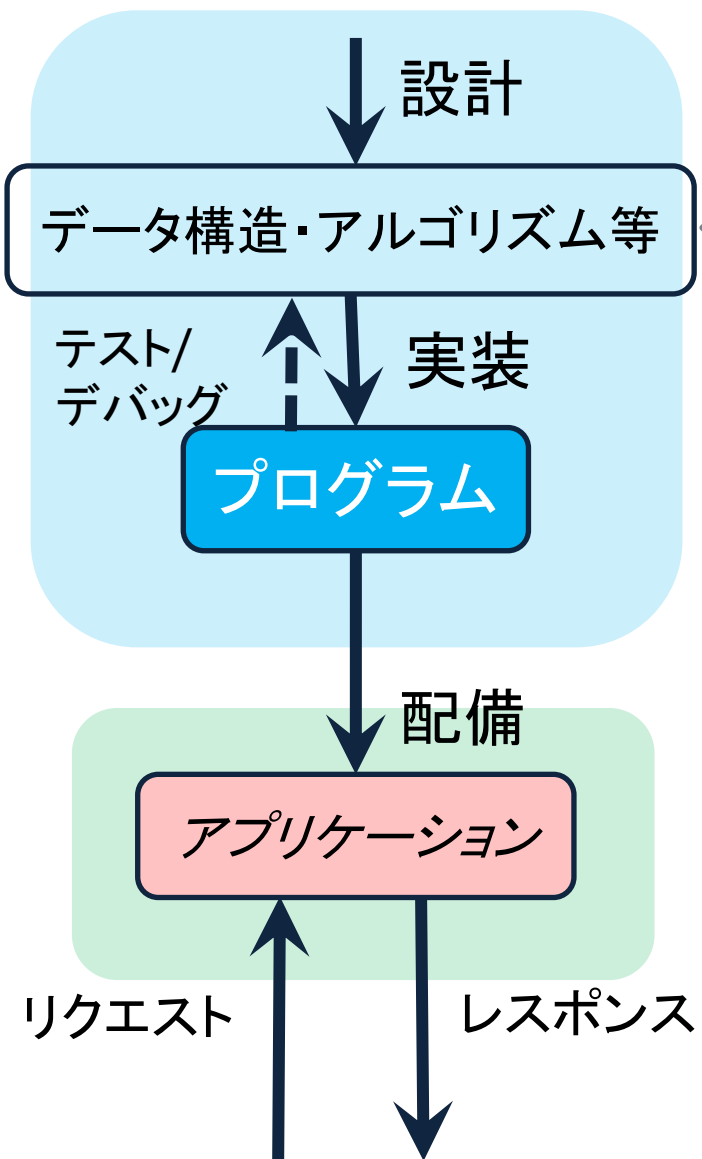
MXNET

Microsoft

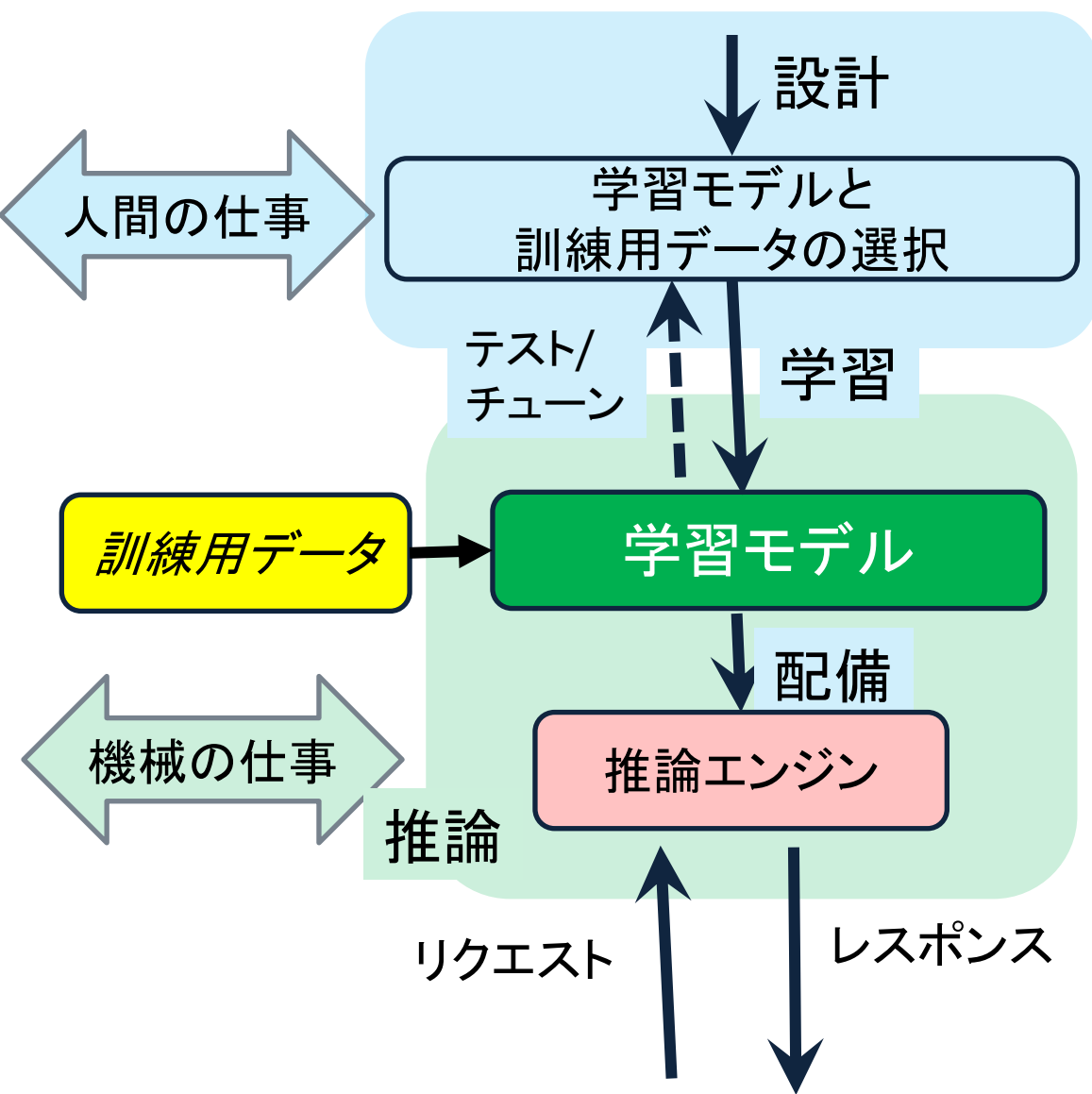
CNTK
CNTK2

Amazon SageMakerでの開発の流れ

プログラミングによる サービス開発



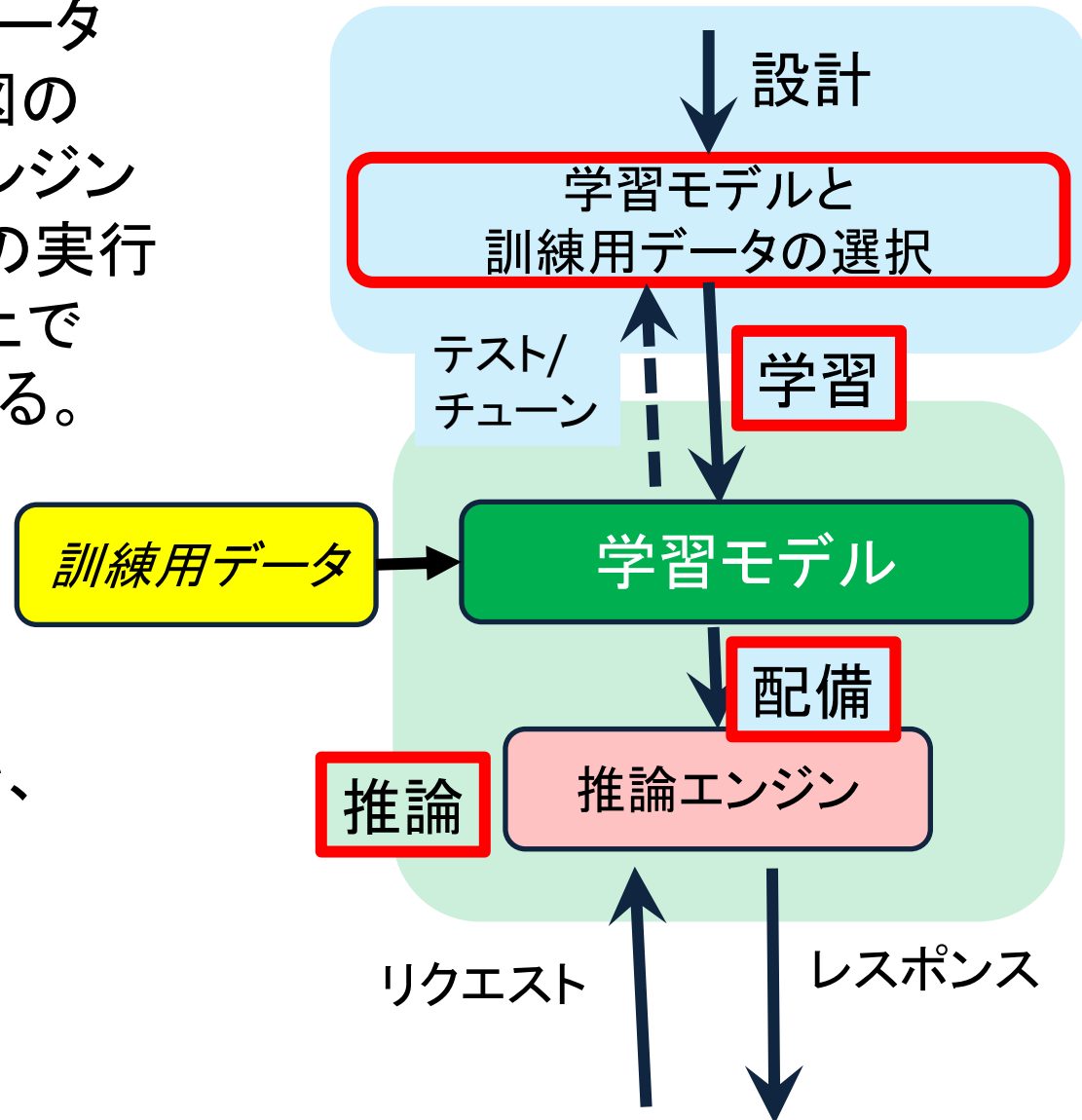
機械学習・深層学習での サービス開発



機械学習・深層学習での サービス開発

Amazon SageMakerは、学習モデルと訓練用データの選択を行うと、右の図のモデルの学習・推論エンジンの配備・推論サービスの実行までを、一つのツール上で「一気通貫」で作り上げる。

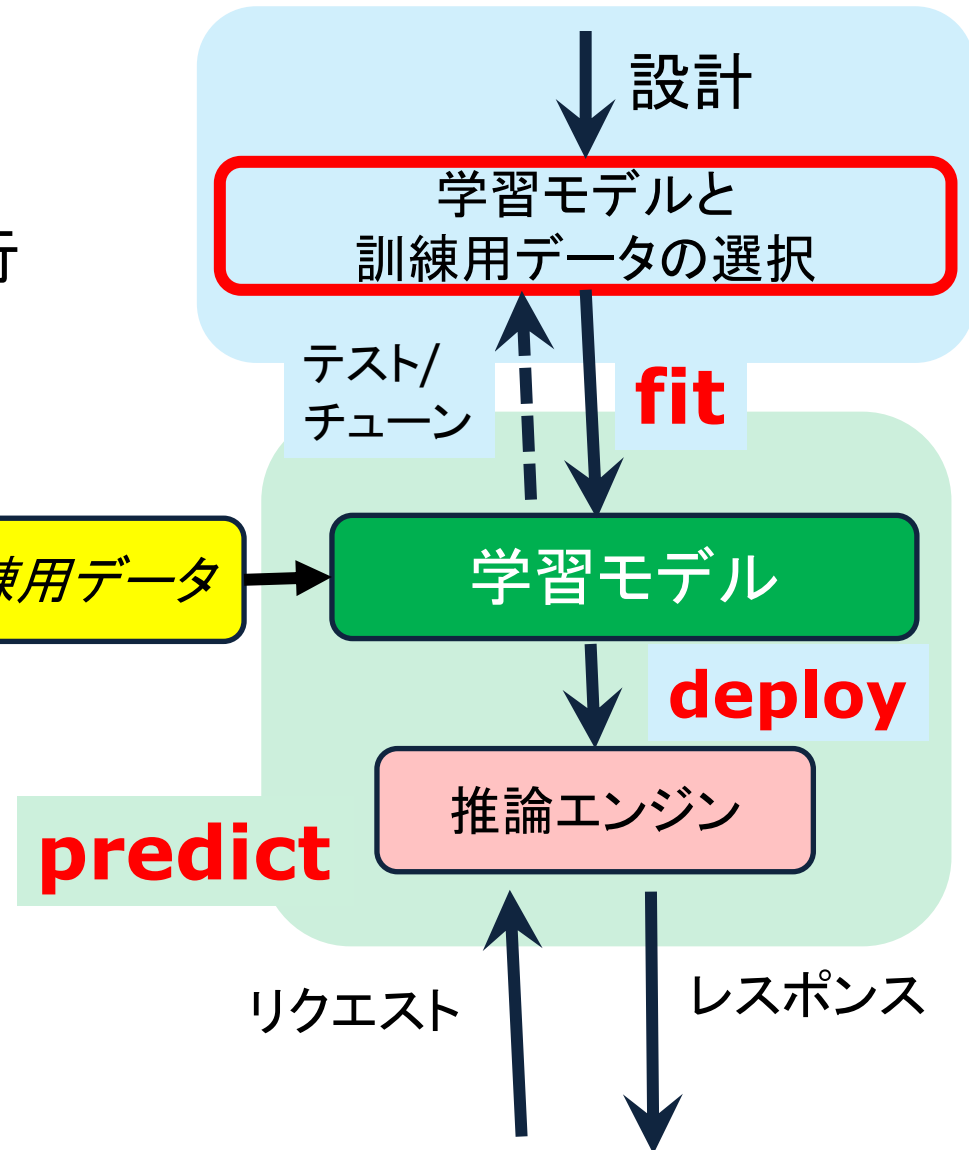
典型的な処理の流れを、
見ておこう。



機械学習・深層学習での サービス開発

Amazon SageMakerは、
学習モデルと訓練用データ
の選択を行うと、右の図の
モデルの学習・推論エンジ
ンの配備・推論サービスの実行
までを、一つのツール上で
「一気通貫」で作り上げる。

典型的な処理の流れを、
見ておこう。



モデルの生成

```
m = MXNet("mnist.py",  
  role=role,  
  train_instance_count=1,  
  train_instance_type="ml.c4.xlarge",  
  hyperparameters={'batch_size': 100,  
    'epochs': 20,  
    'learning_rate': 0.1,  
    'momentum': 0.9,  
    'log_interval': 100})
```

fit

```
m.fit(inputs)
```

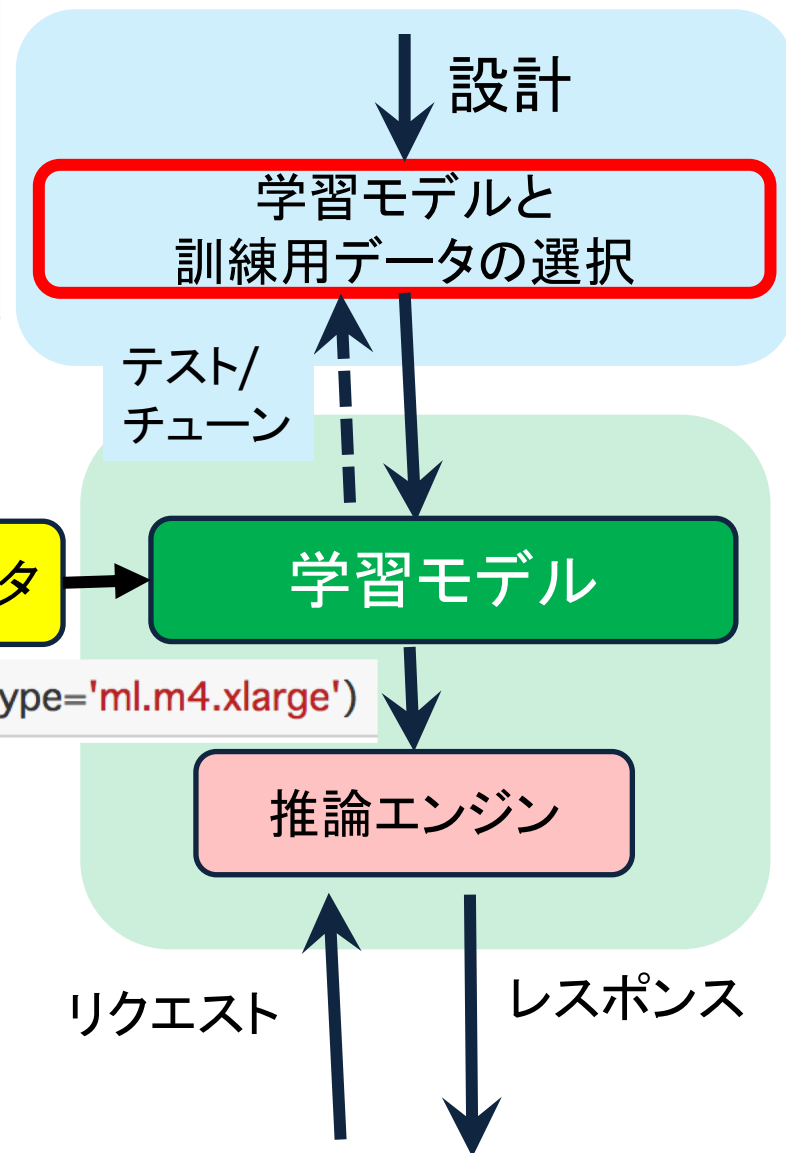
deploy

```
predictor = m.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

predict

```
response = predictor.predict(data)  
print int(response)
```

機械学習・深層学習での サービス開発



Amazon SageMakerでのサンプル

モデルmの生成

```
m = MXNet("mnist.py",
    role=role,
    train_instance_count=1,
    train_instance_type="ml.c4.xlarge",
    hyperparameters={'batch_size': 100,
                      'epochs': 20,
                      'learning_rate': 0.1,
                      'momentum': 0.9,
                      'log_interval': 100})
```

サンプル 1

**MXNetの
MNISTの
Gluonモデル**

<https://goo.gl/Sjsnc5>

モデルmの訓練: **fit**

```
m.fit(inputs)
```

推論エンジン predictorとして配備: **deploy**

```
predictor = m.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

推論の実行: **predict**

```
response = predictor.predict(data)
print int(response)
```


モデルmの生成

```
m = MXNet("cifar10.py",
          role=role,
          train_instance_count=2,
          train_instance_type="ml.p2.xlarge",
          hyperparameters={'batch_size': 128,
                           'epochs': 50,
                           'learning_rate': 0.1,
                           'momentum': 0.9})
```

サンプル 2

**MXNetの
cifar10の
ResNetモデル**

<https://goo.gl/DQhQCP>

モデルmの訓練: **fit**

```
m.fit(inputs)
```

推論エンジン predictorとして配備: **deploy**

```
predictor = m.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

推論の実行: **predict**

```
for i, img in enumerate(image_data):
    response = predictor.predict(img)
    print('image {}: class: {}'.format(i, int(response)))
```

モデルmnist_estimatorの生成

```
mnist_estimator = TensorFlow(entry_point='mnist.py',  
                             role=role,  
                             training_steps=1000,  
                             evaluation_steps=100,  
                             train_instance_count=2,  
                             train_instance_type='ml.c4.xlarge')
```

サンプル 3

Tensorflowの MNIST 分散モデル

<https://goo.gl/otk9Cd>

モデルmnist_estimatorの訓練: **fit**

```
mnist_estimator.fit(inputs)
```

推論エンジン mnist_predictorとして配備: **deploy**

```
mnist_predictor = mnist_estimator.deploy(initial_instance_count=1,  
                                         instance_type='ml.m4.xlarge')
```

推論の実行: **predict**

```
predict_response = mnist_predictor.predict(tensor_proto)
```

モデルestimatorの生成

サンプル 4

```
estimator = TensorFlow(entry_point='resnet_cifar_10.py',
                        source_dir=source_dir,
                        role=role,
                        hyperparameters={'min_eval_frequency': 10},
                        training_steps=1000, evaluation_steps=100,
                        train_instance_count=2, train_instance_type='ml.c4.xlarge',
                        base_job_name='tensorboard-example')
```

**Tensorflowの
cifar10の
ResNetモデル**

<https://goo.gl/yRzhTE>

モデルestimatorの訓練: **fit**

```
estimator.fit(inputs, run_tensorboard_locally=True)
```

推論エンジン predictorとして配備: **deploy**

```
predictor = estimator.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

推論の実行: **predict**

```
response = predictor.predict(data)
print int(response)
```

SageMakerで、すぐに利用可能なモデル

SageMaker Built-in Algorithm

詳細は、リンクを参照されたい

- ❑ Linear Learner <https://goo.gl/4G9sV8>
- ❑ Factorization Machines <https://goo.gl/jJ9c3T>
- ❑ XGBoost Algorithm <https://goo.gl/GDnLgd>
- ❑ Image Classification Algorithm <https://goo.gl/nc3k2k>
- ❑ SageMaker Sequence2Sequence <https://goo.gl/McXLj2>
- ❑ K-Means Algorithm <https://goo.gl/V2HWiy>
- ❑ Principal Component Analysis <https://goo.gl/cXFGPF>
- ❑ Latent Dirichlet Allocation (LDA) <https://goo.gl/X6n54E>
- ❑ Neural Topic Model (NTM) <https://goo.gl/Fgbdhg>
- ❑ DeepAR Forecasting <https://goo.gl/gtsyQ6>
- ❑ BlazingText <https://goo.gl/Yktcm1>

<https://goo.gl/9xZdL4>

```
containers = {'us-west-2': '174872318107.dkr.ecr.us-west-2.amazonaws.com/linear-learner:latest',  
             'us-east-1': '382416733822.dkr.ecr.us-east-1.amazonaws.com/linear-learner:latest',  
             'us-east-2': '404615174143.dkr.ecr.us-east-2.amazonaws.com/linear-learner:latest',  
             'eu-west-1': '438346466558.dkr.ecr.eu-west-1.amazonaws.com/linear-learner:latest'}
```

```
import boto3  
import sagemaker  
  
sess = sagemaker.Session()  
  
linear = sagemaker.estimator.Estimator(containers[boto3.Session().region_name],  
                                       role,  
                                       train_instance_count=1,  
                                       train_instance_type='ml.c4.xlarge',  
                                       output_path=output_location,  
                                       sagemaker_session=sess)  
linear.set_hyperparameters(feature_dim=784,  
                           predictor_type='binary_classifier',  
                           mini_batch_size=200)  
  
linear.fit({'train': s3_train_data})
```

```
linear_predictor = linear.deploy(initial_instance_count=1,  
                                 instance_type='ml.m4.xlarge')
```

```
result = linear_predictor.predict(train_set[0][30:31])  
print(result)
```

**An Introduction to
Linear Learner
with MNIST**

<https://goo.gl/URsZqV>

```
containers = {'us-west-2': '174872318107.dkr.ecr.us-west-2.amazonaws.com/factorization-machines:latest',  
             'us-east-1': '382416733822.dkr.ecr.us-east-1.amazonaws.com/factorization-machines:latest',  
             'us-east-2': '404615174143.dkr.ecr.us-east-2.amazonaws.com/factorization-machines:latest',  
             'eu-west-1': '438346466558.dkr.ecr.eu-west-1.amazonaws.com/factorization-machines:latest'}
```

```
import boto3  
import sagemaker  
  
sess = sagemaker.Session()  
  
fm = sagemaker.estimator.Estimator(containers[boto3.Session().region_name],  
                                   role,  
                                   train_instance_count=1,  
                                   train_instance_type='ml.c4.xlarge',  
                                   output_path=output_location,  
                                   sagemaker_session=sess)  
fm.set_hyperparameters(feature_dim=784,  
                        predictor_type='binary_classifier',  
                        mini_batch_size=200,  
                        num_factors=10)  
  
fm.fit({'train': s3_train_data})
```

**An Introduction to
Factorization Machines
with MNIST**

```
fm_predictor = fm.deploy(initial_instance_count=1,  
                          instance_type='ml.m4.xlarge')
```

```
result = fm_predictor.predict(train_set[0][30:31])  
print(result)
```

<https://goo.gl/u2UDvq>

```
# select the algorithm container based on this notebook's current location
```

```
containers = {  
    'us-west-2': '266724342769.dkr.ecr.us-west-2.amazonaws.com/lda:latest',  
    'us-east-1': '766337827248.dkr.ecr.us-east-1.amazonaws.com/lda:latest',  
    'us-east-2': '999911452149.dkr.ecr.us-east-2.amazonaws.com/lda:latest',  
    'eu-west-1': '999678624901.dkr.ecr.eu-west-1.amazonaws.com/lda:latest'  
}  
region_name = boto3.Session().region_name  
container = containers[region_name]  
  
print('Using SageMaker LDA container: {} ({}).format(container, region_name))
```

```
session = sagemaker.Session()  
  
# specify general training job information  
lda = sagemaker.estimator.Estimator(  
    container,  
    role,  
    output_path='s3://{}/{}'/output'.format(bucket, prefix),  
    train_instance_count=1,  
    train_instance_type='ml.c4.2xlarge',  
    sagemaker_session=session,  
)
```

**An Introduction to
SageMaker LDA**

<https://goo.gl/rGgbJ9>


```
# set algorithm-specific hyperparameters
lda.set_hyperparameters(
    num_topics=num_topics,
    feature_dim=vocabulary_size,
    mini_batch_size=num_documents_training,
    alpha0=1.0,
)

# run the training job on input data stored in S3
lda.fit({'train': s3_train_data})
```

```
lda_inference = lda.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge', # LDA inference may work better at scale on ml.c4 instances
)
```

```
results = lda_inference.predict(documents_test[:12])
print(results)
```

**An Introduction to
SageMaker LDA**

<https://goo.gl/rGgbJ9>

Tensorflow Estimator

https://www.tensorflow.org/programmers_guide/estimators

“Tensorflow, Machine Learning and Learning to learn” Sherry Moore

<https://goo.gl/Gr2jvX>

Canned Estimators

Models in a box

Estimator

tf.keras

Build models

tf.layers

Python Frontend

C++ Frontend

...

TensorFlow Distributed Execution Engine

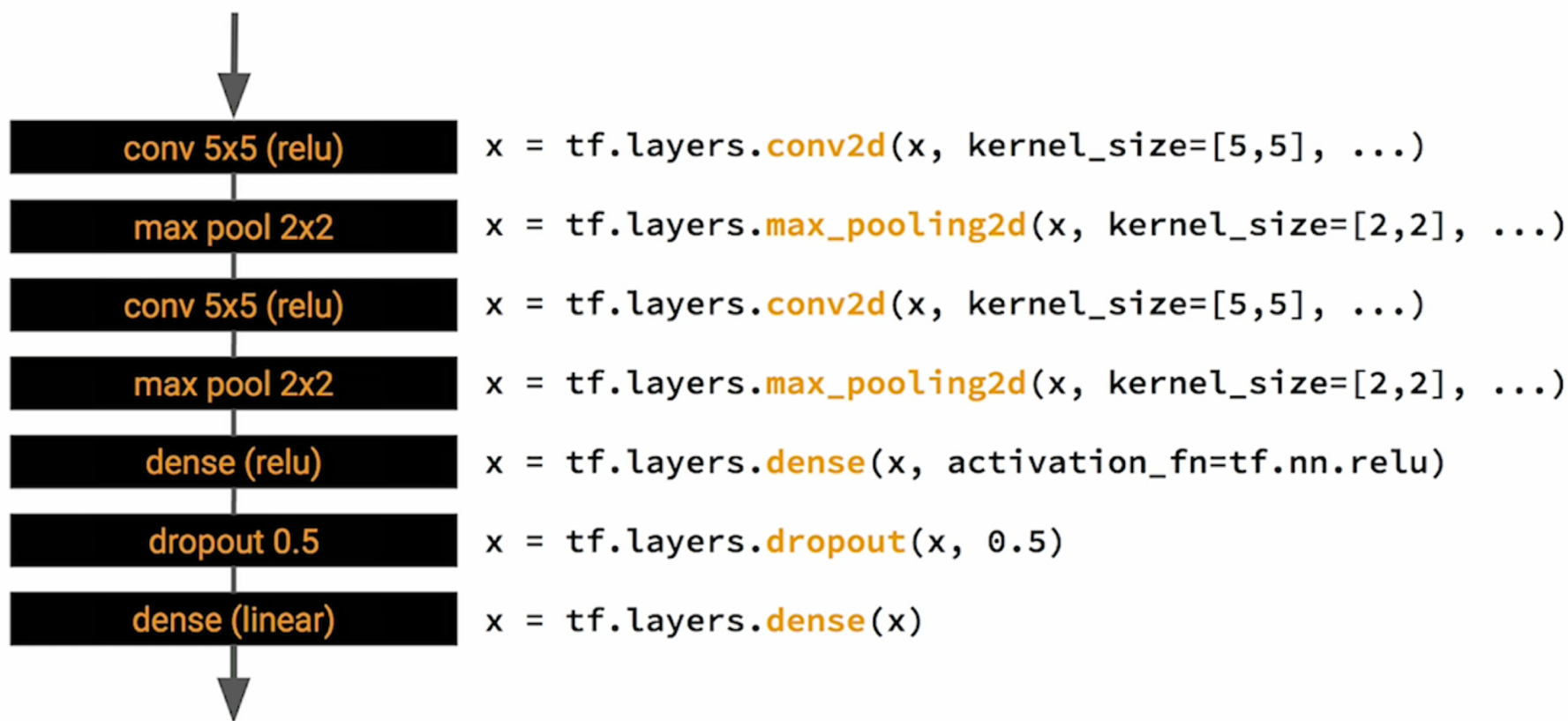
CPU

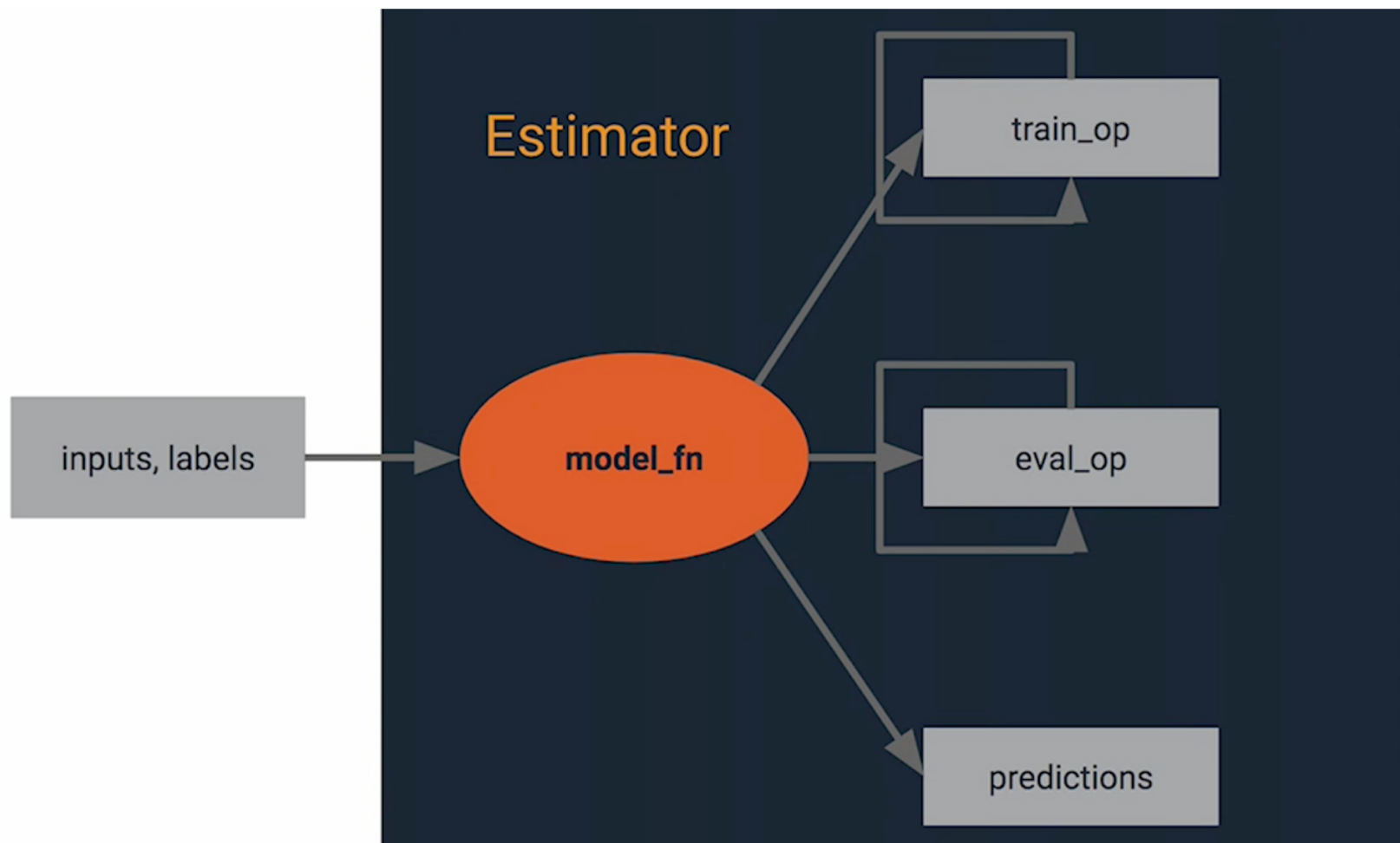
GPU

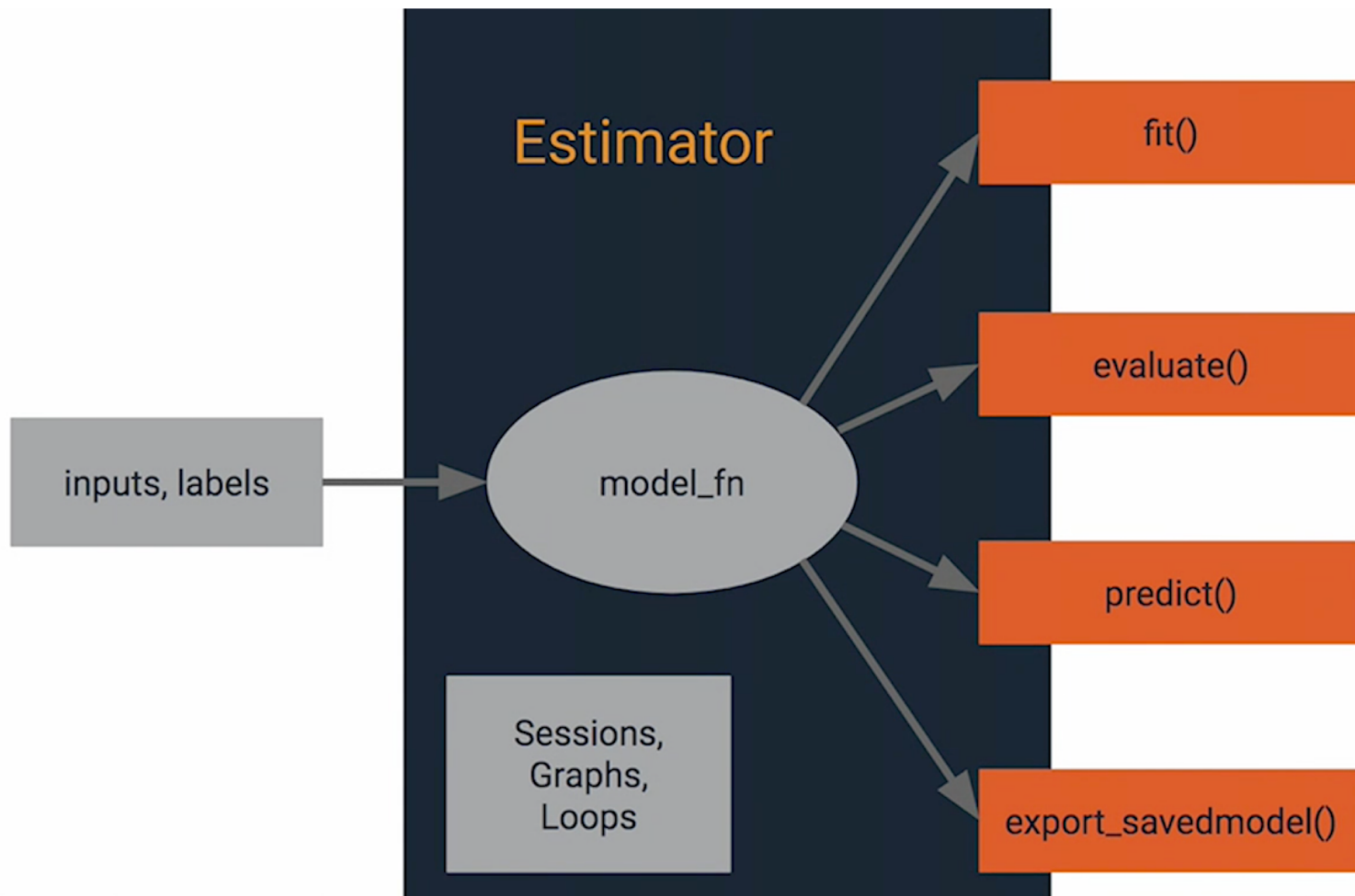
Android

iOS

...



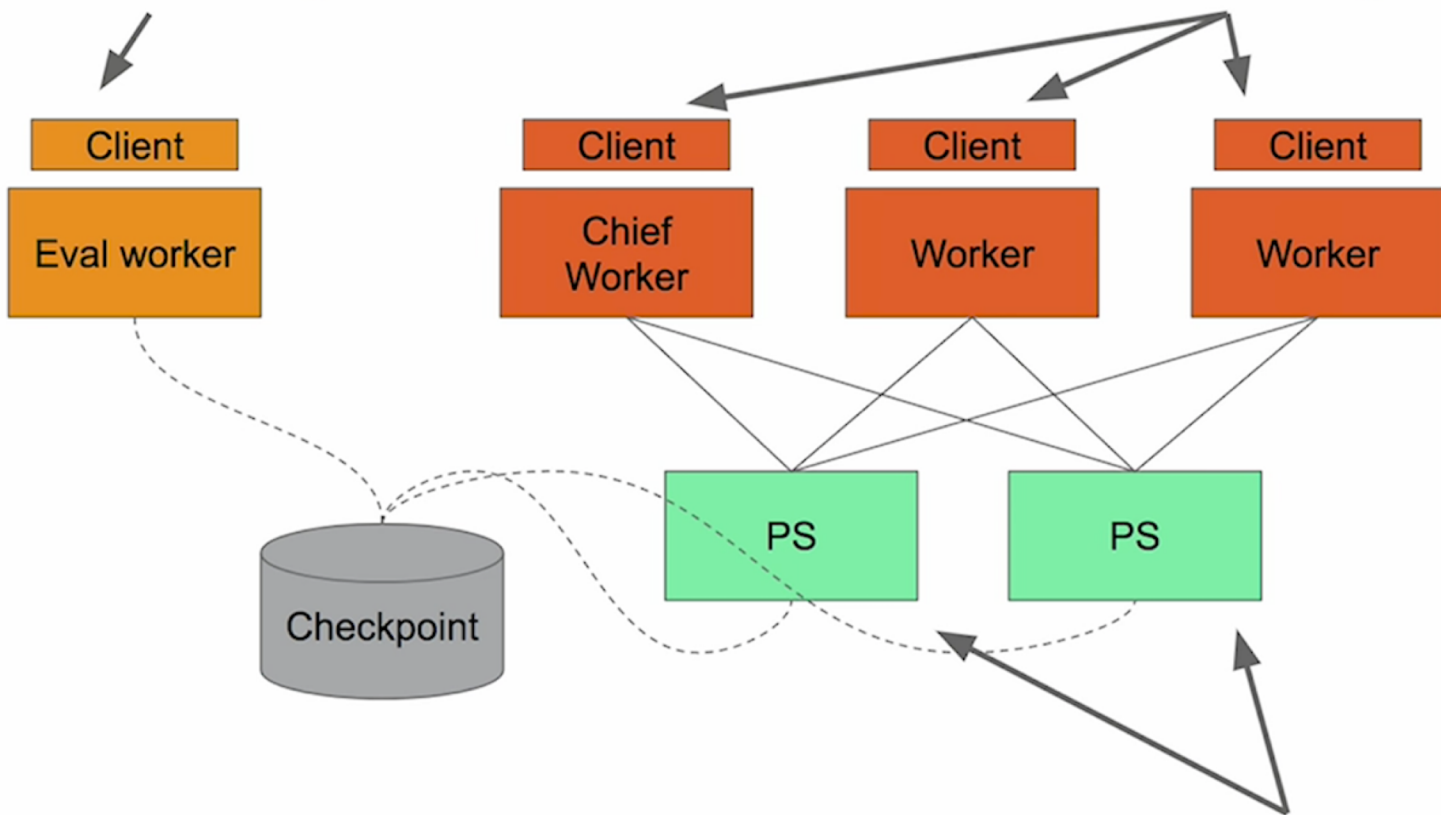




```
area = real_valued_column("square_foot"),  
rooms = real_valued_column("num_rooms"),  
zip_code = sparse_column_with_integerized_feature("zip_code", 100000)  
  
regressor = LinearRegressor(feature_columns=[area, rooms, zip_code])  
  
classifier.fit(train_input_fn)  
  
classifier.evaluate(eval_input_fn)
```

estimator.evaluate()

estimator.train()



start TensorFlow server


```
TF_CONFIG = {  
    "cluster": {  
        "chief": ["chief.example.com:2222"],  
        "worker": ["worker0.example.com:2222",  
                   "worker1.example.com:2222",  
                   "worker2.example.com:2222"],  
        "ps": ["ps0.example.com:2222",  
               "ps1.example.com:2222"]  
    },  
    "task": {"type": "worker", "index": 1},  
    "model_dir": "/path/to/model",  
}
```

```
# Create an Estimator
```

```
est = tf.estimator.Estimator(...)
```

```
# Start training, eval, or PS
```

```
tf.estimator.train_and_evaluate(
```

```
    est,
```

```
    training_input_fn=..., # Specify training inputs
```

```
    [EvalSpec(eval_input_fn, ...)]) # Specify eval configuration
```

```
# This code can run on all nodes
```

```
# Configured by environment TF_CONFIG
```

A white, rectangular, vertical device with a camera mounted on top. The device features the AWS logo, an Intel logo, and a circular sensor. It is positioned in the center of the image against a blurred background.

いつか、誰でも、人工知能を
簡単に作れる時代が来る。

Amazon SageMaker
クラウド・ハンズオン

セミナー予告

5/22 マルレク2018 第一回 「ポスト・ディープラーニングの 人工知能技術を展望する」

主催： 丸山事務所

<https://marulec01.peatix.com/>

参加費： 1,000円

6/16 「ディープラーニング入門6時間集中講座 ～未来の「常識」を基礎から学ぼう」

主催： MaruLabo+角川アスキー総研

参加費： 6,000円