skymind

畳み込みニューラル
ネットワークの基礎と応用

Tableau データサイエンス勉強会第4回 ー画像認識とAIの巻ー

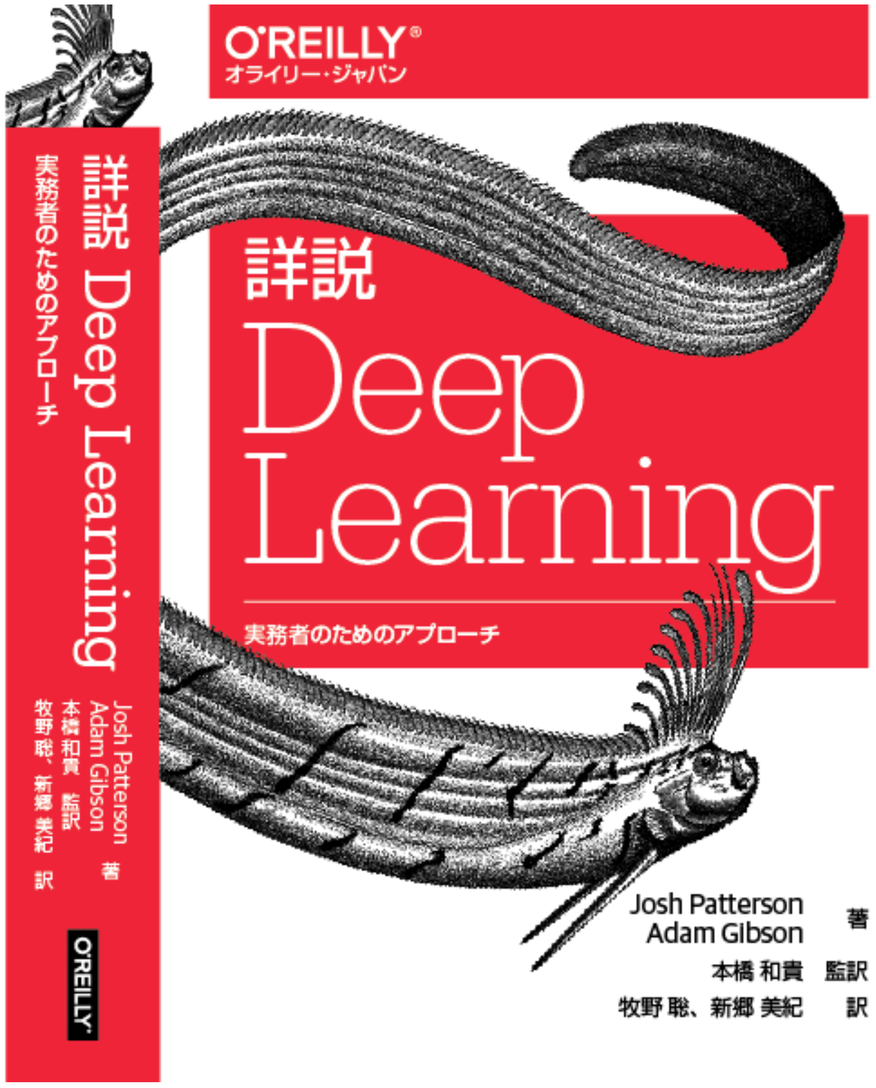
https://techplay.jp/event/750555

2019-10-21 (Mon) @ TECH PLAY SHIBUYA

自己紹介

- **本橋 和貴** @kmotohas
  - スカイマインド株式会社
    - Deep Learning Engineer (前職ではDL+ROS)
  - 素粒子物理学実験（LHC-ATLAS実験）出身
    - 博士（理学）
  - 好きな本：詳説 Deep Learning — 実務者のためのアプローチ

2019年8月9日発売



- 原著 “Deep Learning — A Practitioner’s Approach” は 2017年8月発売
- JVM言語用ディープラーニング開発フレームワーク Deeplearning4j (DL4J) を用いた解説書
  - 著者は DL4J の開発者 Adam Gibson、Skymind Inc を創業
  - ソフトウェア/アプリケーション/システム・エンジニアなどがメインターゲット
  - ディープラーニングの基礎からHadoop/Sparkといったビッグデータ分析基盤との連携まで解説

Agenda

機械学習の導入

Kerasを用いたコーディングサンプル

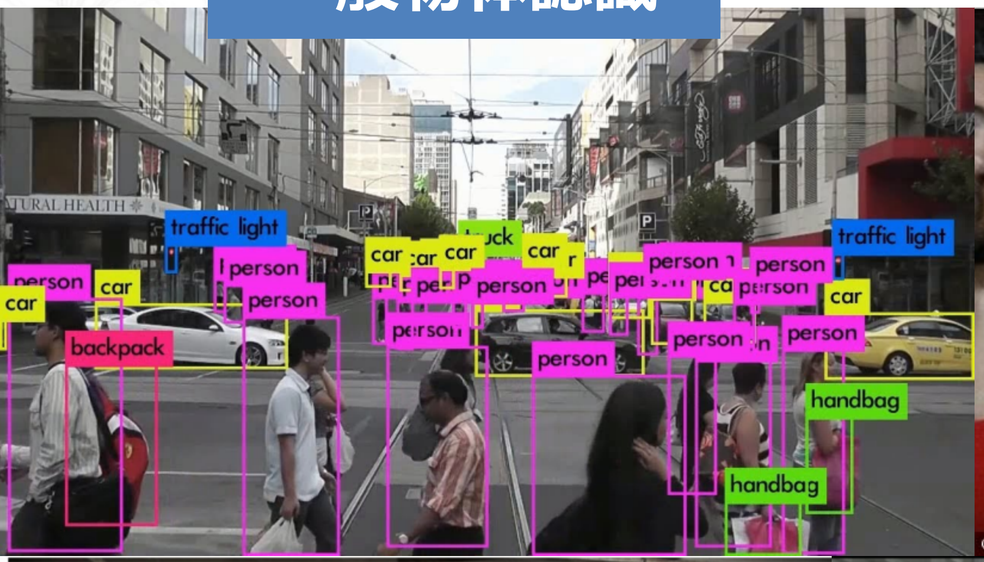
畳み込みニューラルネットワークの基礎

CNNのフィルタの理解

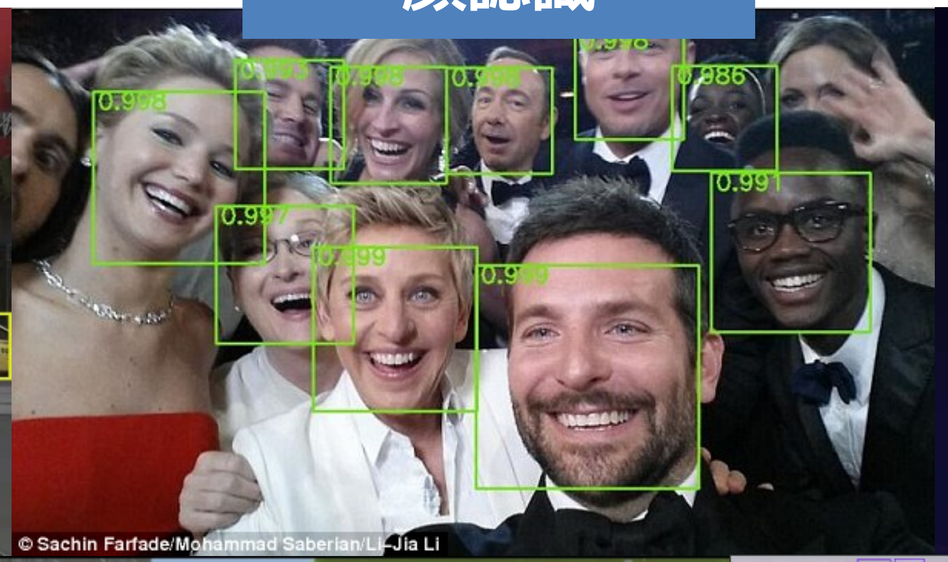
CNNアーキテクチャの紹介

skymind | 画像認識のアプリケーション

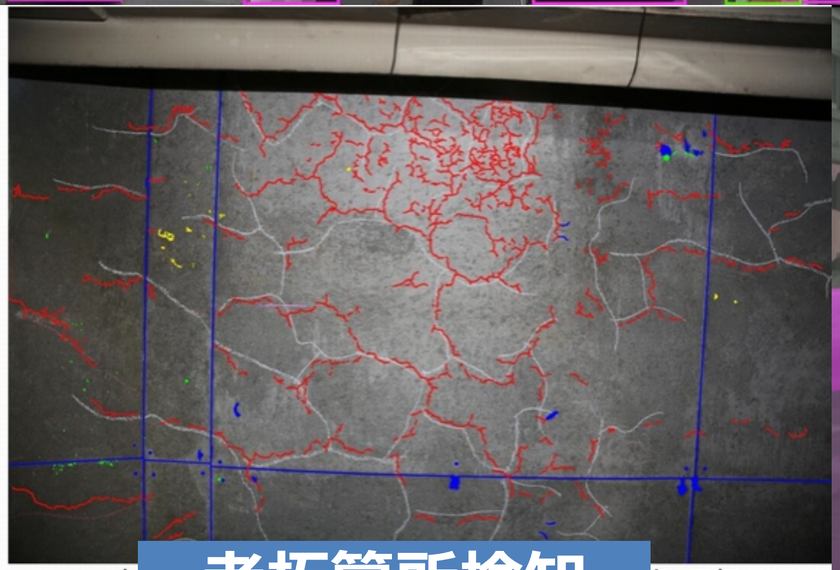
一般物体認識



顔認識

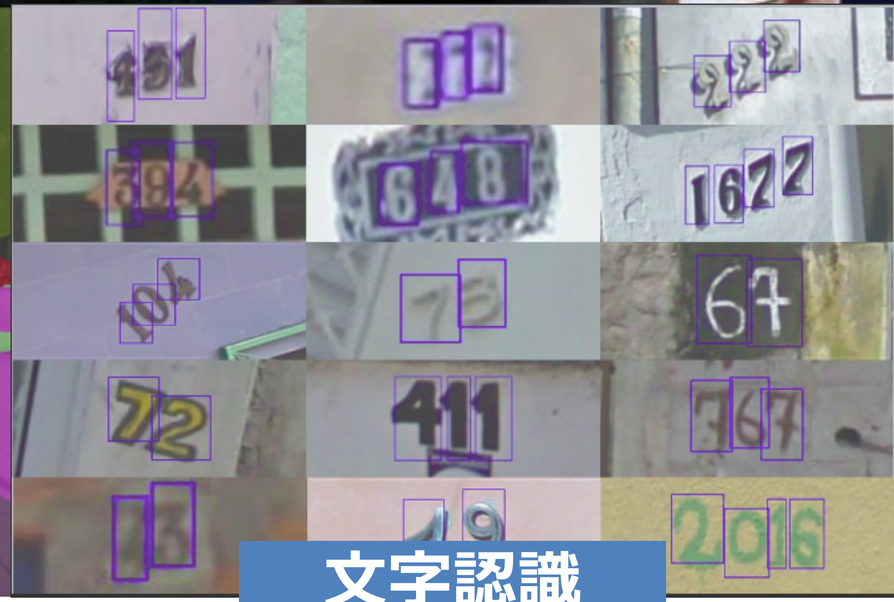


年齢推定



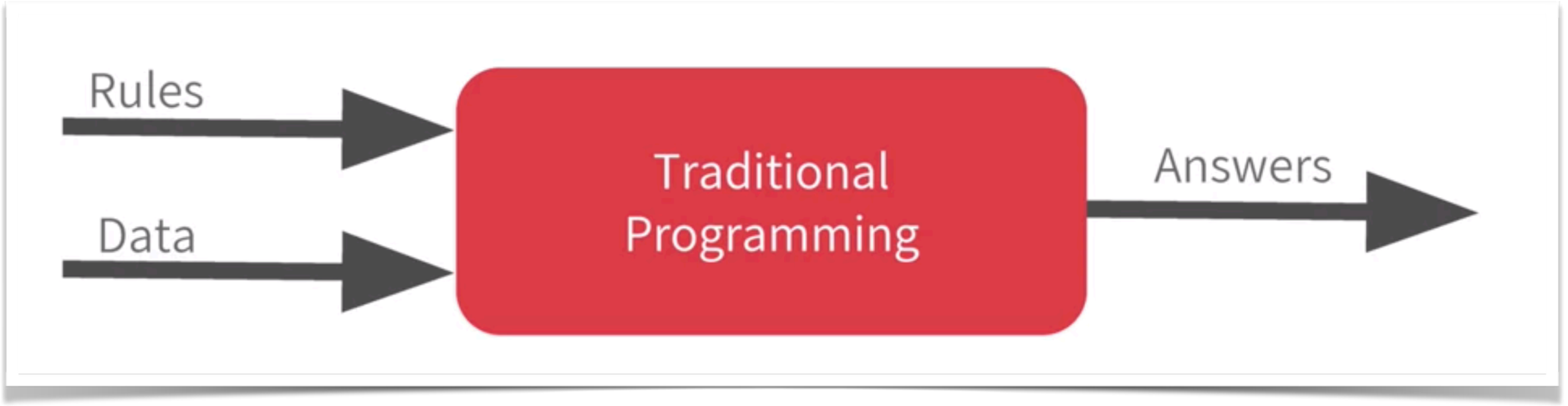
老朽箇所検知

セグメンテーション



文字認識

ニューラルネットワークの基礎



https://www.coursera.org/learn/introduction-tensorflow

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

https://www.coursera.org/learn/introduction-tensorflow

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

https://www.coursera.org/learn/introduction-tensorflow

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

https://www.coursera.org/learn/introduction-tensorflow

Activity Recognition

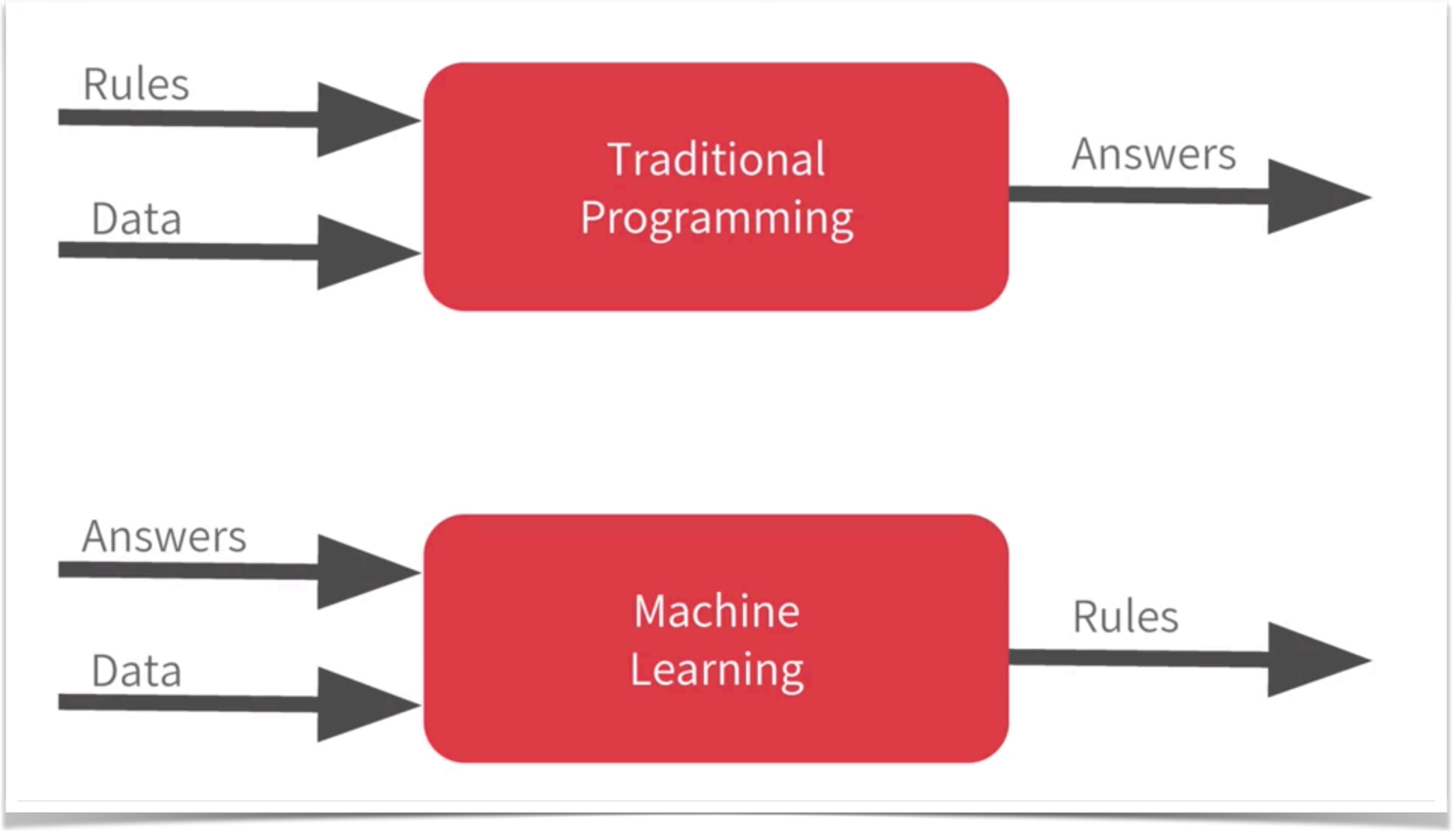
```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// Oh crap
```

https://www.coursera.org/learn/introduction-tensorflow



https://www.coursera.org/learn/introduction-tensorflow

Activity Recognition

```
01010010101001010101
10010101010010111101
01001010100101010010
01010010101001010101
```

Label = WALKING

```
10101001010010101010
01010100100100100001
00100111110101011111
10101001001111010111
```

Label = RUNNING

```
10010100111110101010
11010101110101011101
10101011110101010111
11111100011110101010
```

Label = BIKING

```
11111111110100111010
00111111010111101010
01011101010101011101
10101010101001111101
```

Label = GOLFING
(Sort of)

https://www.coursera.org/learn/introduction-tensorflow

機械学習の"Hello World”

x = -2, -1, 0, 1, 2, 3, 4

y = -3, -1, 1, 3, 5, 7, 9

$$y = f(x)$$

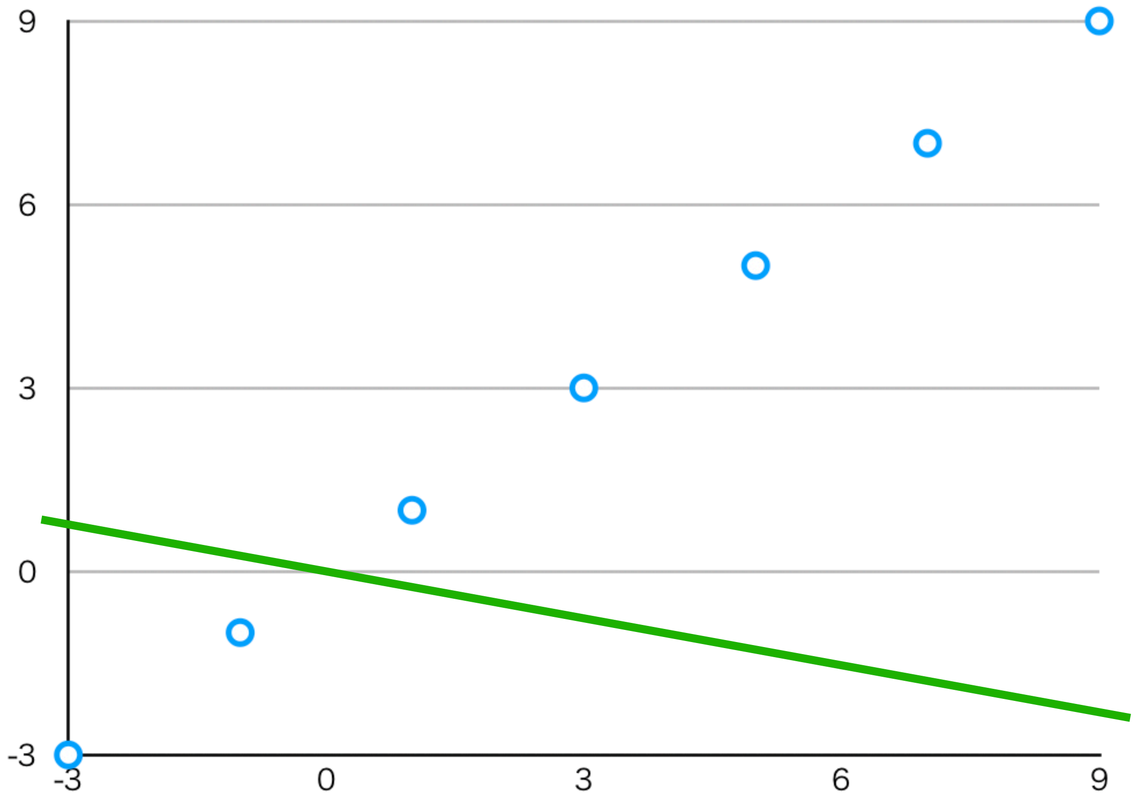
機械学習の"Hello World”

$$x = -2, -1, 0, 1, 2, 3, 4$$
$$y = -3, -1, 1, 3, 5, 7, 9$$

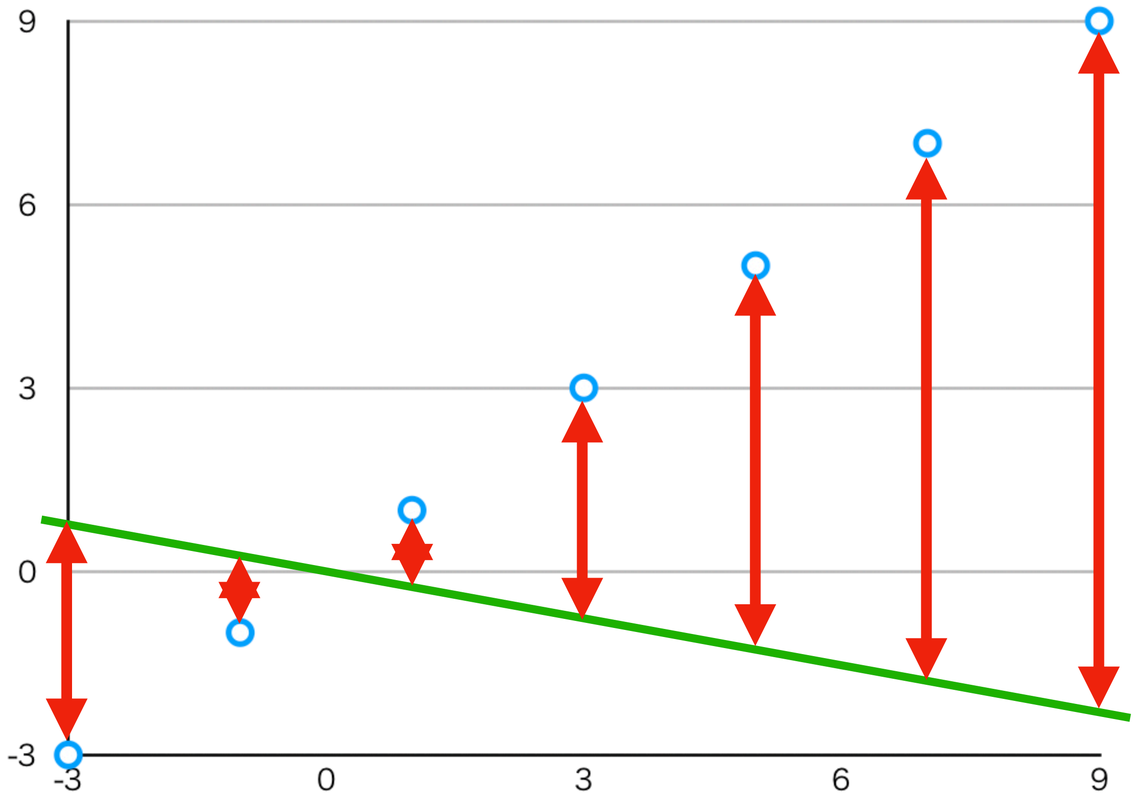
$$y = f(x) = 2x + 1$$

機械学習のアプローチ

- 適当にモデルを初期化 (y_=ax+b)

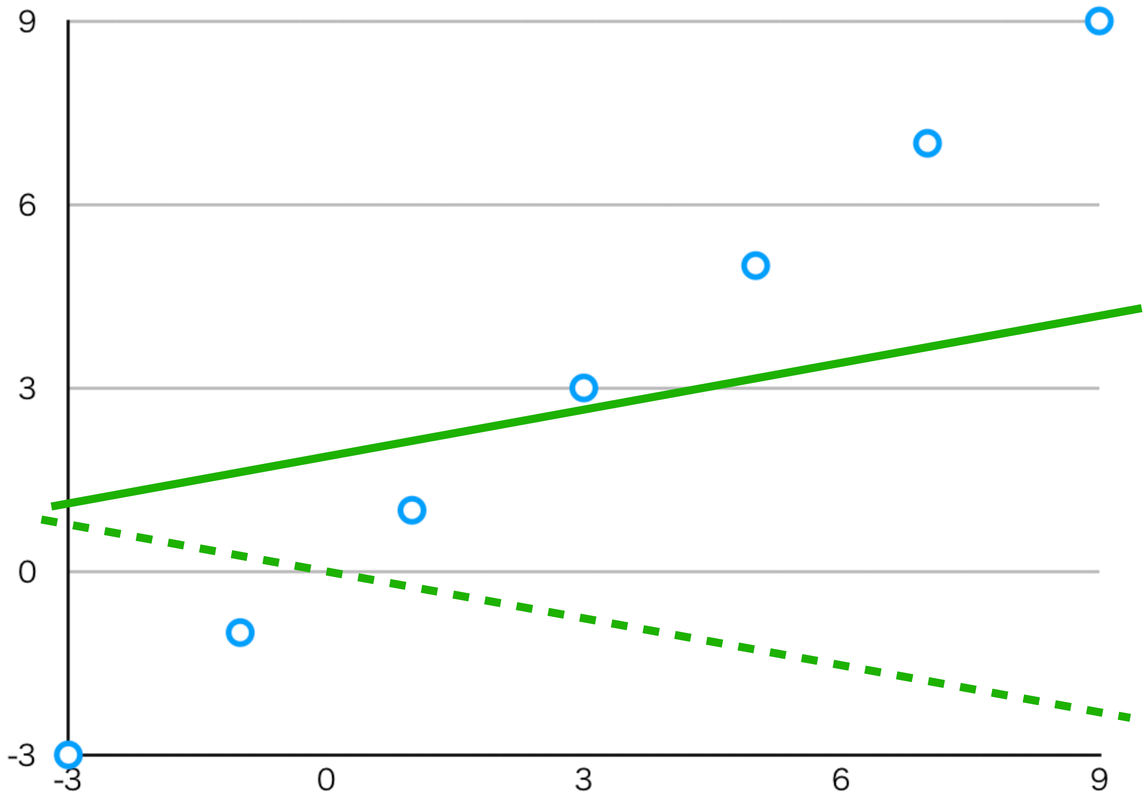


機械学習のアプローチ



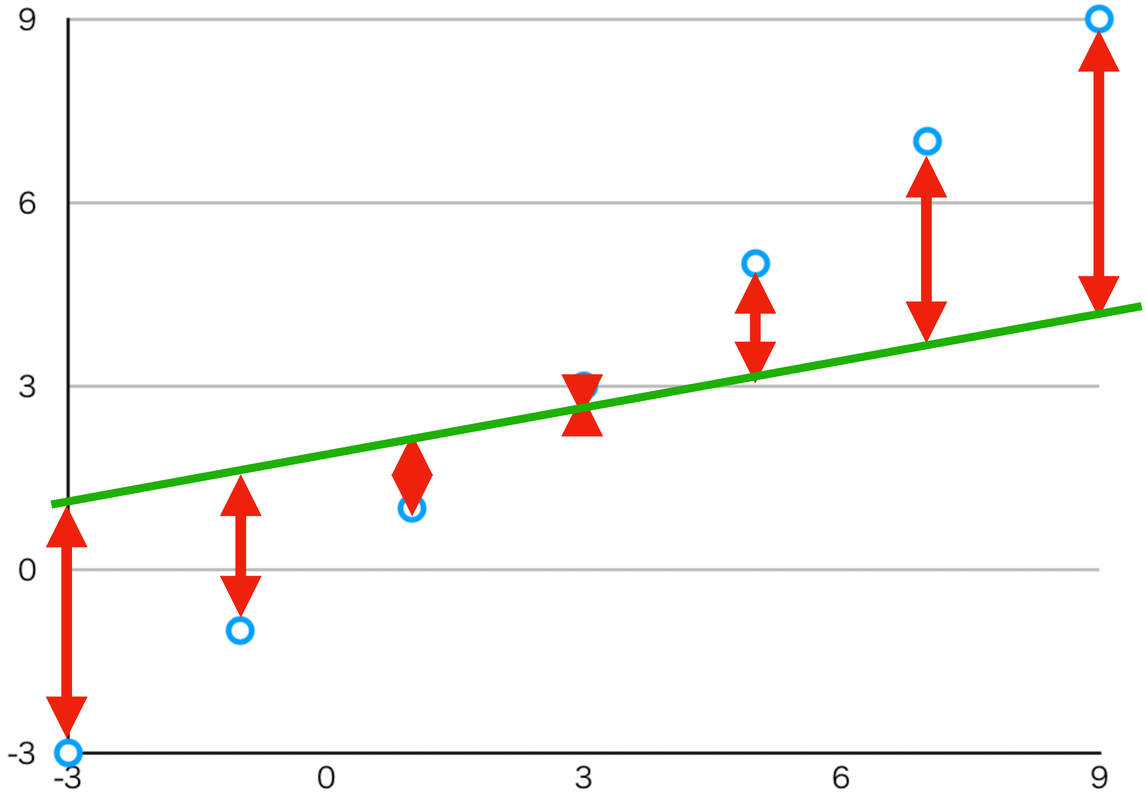
- 適当にモデルを初期化 ($y\_=ax+b$)
- **誤差（損失）を計算 ($L=1/N\ \Sigma(y-y\_)^2$)**

機械学習のアプローチ



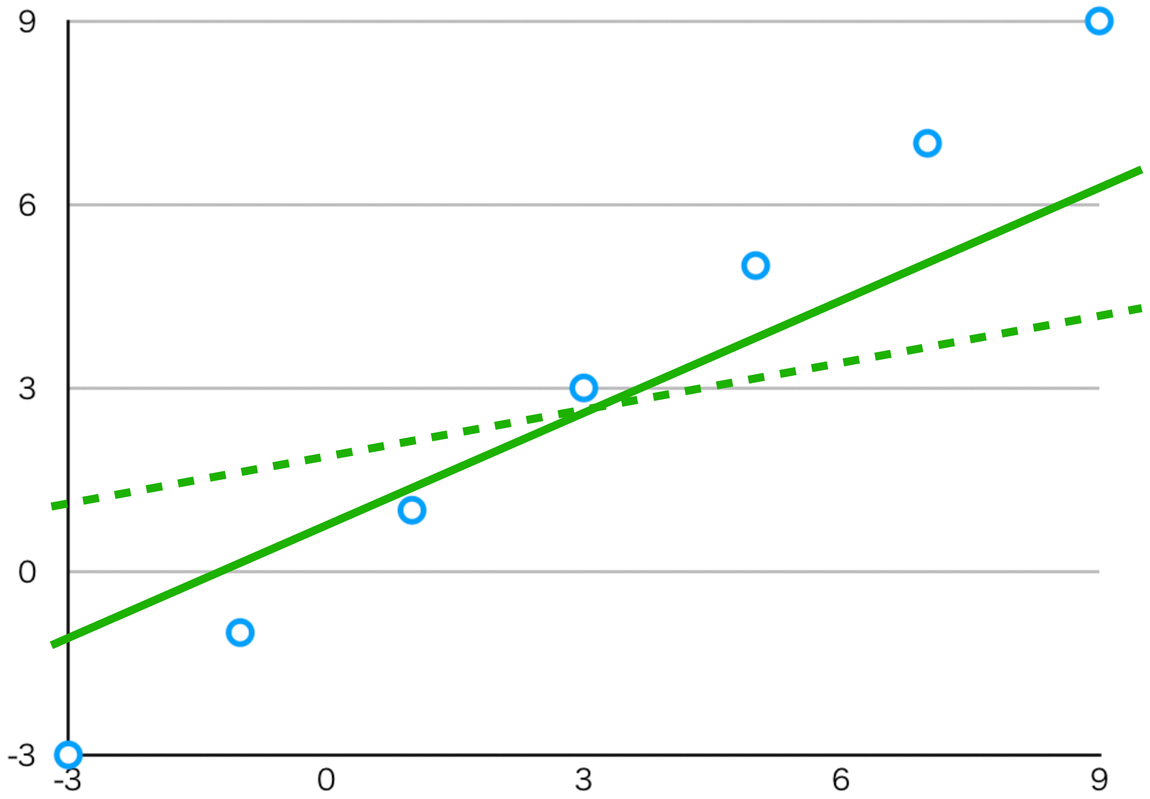
- 適当にモデルを初期化 ($y\_=ax+b$)
- 誤差（損失）を計算 ($L=1/N\ \Sigma(y-y\_)^2$)
- **誤差が小さくなるようにパラメータ (a, b) を少し更新 ($a \leftarrow a - \eta\ \partial L/\partial a$)**

機械学習のアプローチ



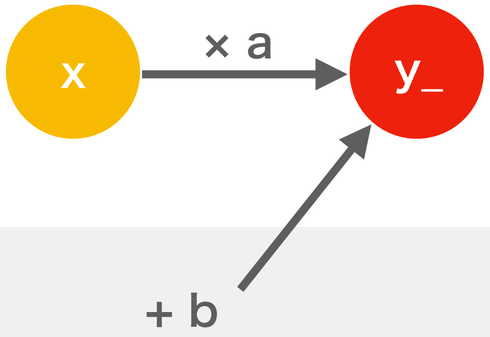
- 適当にモデルを初期化 ($y\_=ax+b$)
- 誤差（損失）を計算 ($L=1/N\ \Sigma(y-y\_)^2$)
- 誤差が小さくなるようにパラメータ (a, b) を少し更新 ($a \leftarrow a - \eta\ \partial L/\partial a$)
- **誤差（損失）を計算 ($L=1/N\ \Sigma(y-y\_)^2$)**

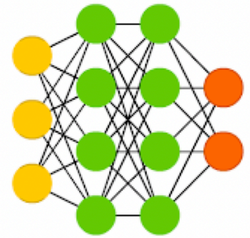
機械学習のアプローチ



- 適当にモデルを初期化 ($y\_=ax+b$)
- 誤差（損失）を計算 ($L=1/N\ \Sigma(y-y\_)^2$)
- 誤差が小さくなるようにパラメータ (a, b) を少し更新 ($a \leftarrow a - \eta\ \partial L/\partial a$)
- 誤差（損失）を計算 ($L=1/N\ \Sigma(y-y\_)^2$)
- **誤差が小さくなるようにパラメータ (a, b) を少し更新 ($a \leftarrow a - \eta\ \partial L/\partial a$)**
- …

```
import tensorflow as tf
import numpy as np

model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

>>>print(model.predict([10.0]))
[[18.977331]]
```



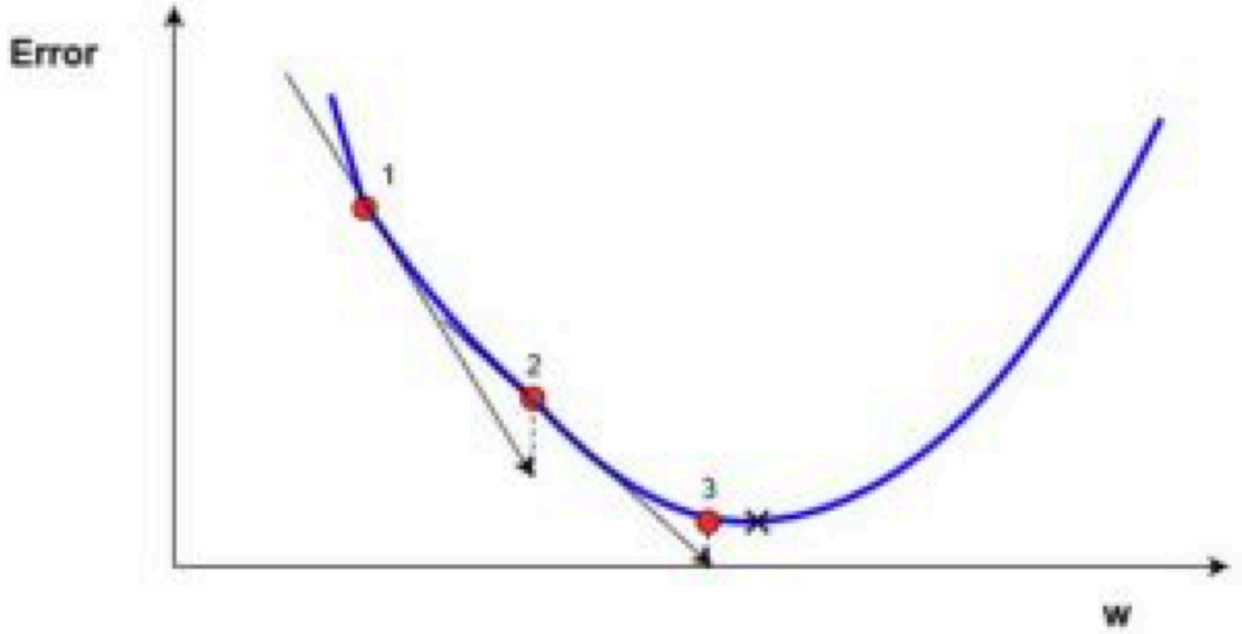


```
1  import tensorflow as tf
2  import numpy as np
3
4  model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])
5  model.compile(optimizer='sgd', loss='mean_squared_error')
6
7  xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
8  ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
9
10 model.fit(xs, ys, epochs=500)
11
12 >>>print(model.predict([10.0]))
13 [[18.977331]]
```

```
import tensorflow as tf
import numpy as np

model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

>>>print(model.predict([10.0]))
[[18.977331]]
```

skymind

確率的勾配降下法 SGD, stochastic gradient descent.
デファクトスタンダード
（改良版：momentum, Nesterovの加速法, RMSProp, Adam, ...）



$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

勾配降下法

https://www.yamakei-online.com/journal/detail.php?id=3185

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

「目隠しで足元の勾配情報のみを使って山の頂上を目指すようなもの」

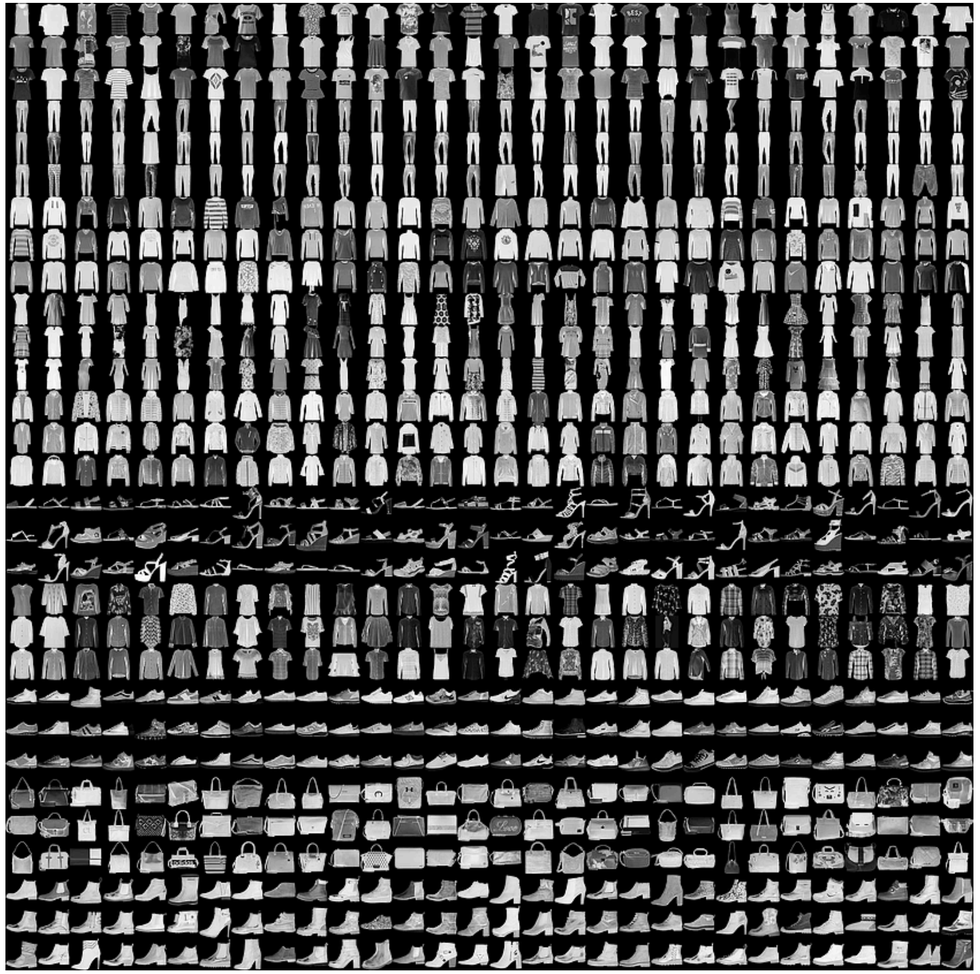
https://twitter.com/momiji_fullmoon/status/1110316960611368960

学習率 η は歩幅のイメージ（η小＝すり足、η大＝巨人の一歩）

Fashion MNIST Dataset

- 7万画像
- 10カテゴリ
- 28×28 pixels
- 実験用データセット

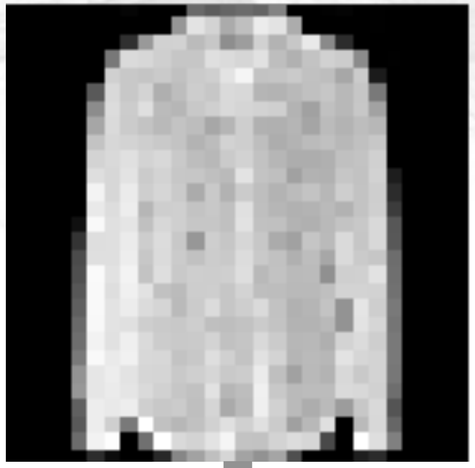
ラベル	記述
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot



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

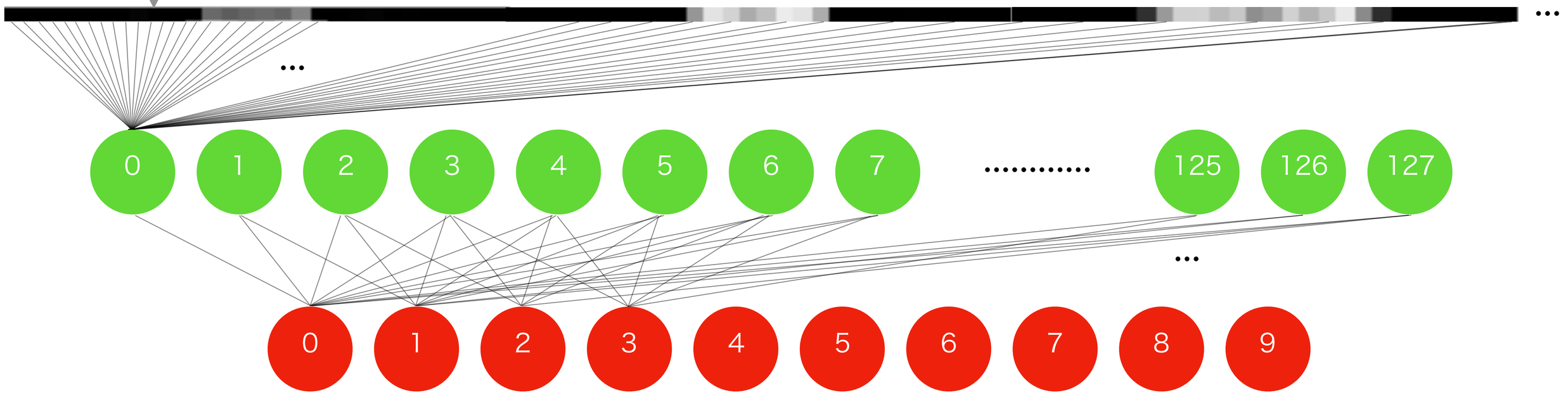
```
import tensorflow as tf
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
train_images = train_images / 255.
test_images = test_images / 255.
```

```
import tensorflow as tf
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
train_images = train_images / 255.
test_images = test_images / 255.

model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```



```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Flatten: (28, 28) => (784)



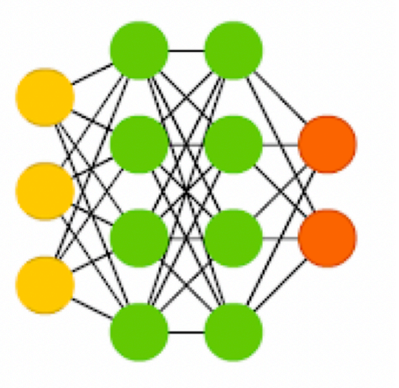
```
import tensorflow as tf
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
train_images = train_images / 255.
test_images = test_images / 255.

model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer=tf.train.AdamOptimizer(),
    loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=128, verbose=1,
    validation_data=(test_images, test_labels))
```

Dense Layer の欠点

- 入力のベクトルの**全要素**の相関をみている

  > 住宅価格予測みたいな話ならまだいい

  > もう作ってる特徴量と特徴量の組み合わせ



  - 例）東京墨田区 & 床面積 30m^2 & 1K & 風呂トイレ別 & 新築 => 家賃月10万円

- **「画像の特徴量」を抽出**してからDense Layerに渡せば効率的

畳込みニューラルネットワーク (Convolutional Neural Network; CNN)

畳み込みニューラルネットワークの基礎

畳み込み (Convolution)

144	60	19
188	82	32
156	55	27

-1	0	-2
0.5	4.5	-1.5
1.5	2	-3

CURRENT_PIXEL_VALUE = 82

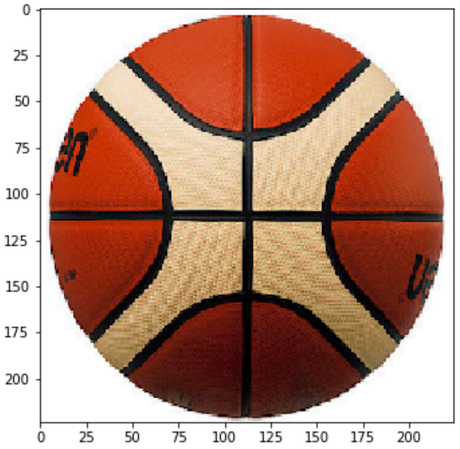
NEW_PIXEL_VALUE =

(-1 * 144) + (0 * 60) + (-2 * 19)

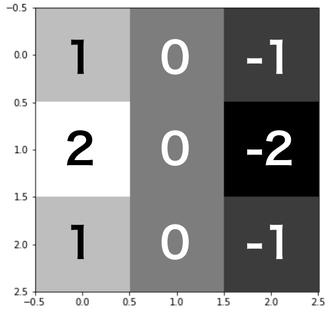
+ (0.5 * 188) + (4.5 * 82) + (-1.5 * 32)

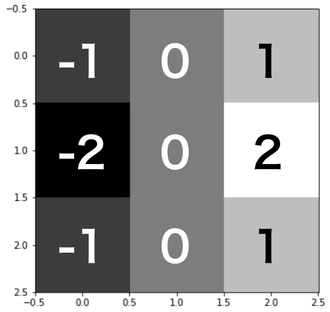
+ (1.5 * 156) + (2 * 55) + (-3 * 27)

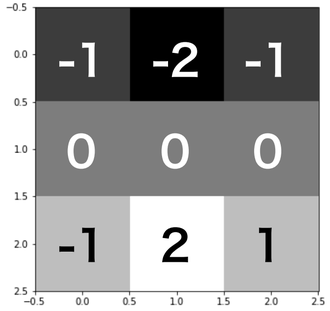
$$u_{ijm} = \sum_{k=0}^{K-1} \sum_{p=0}^{W-1} \sum_{q=0}^{H-1} z_{i+p,j+q,k}^{(l-1)} h_{pqkm} + b_{ijm}$$

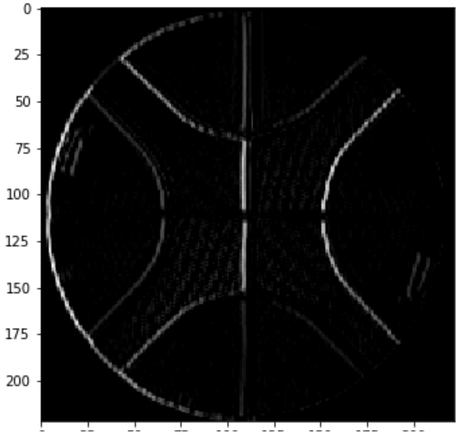


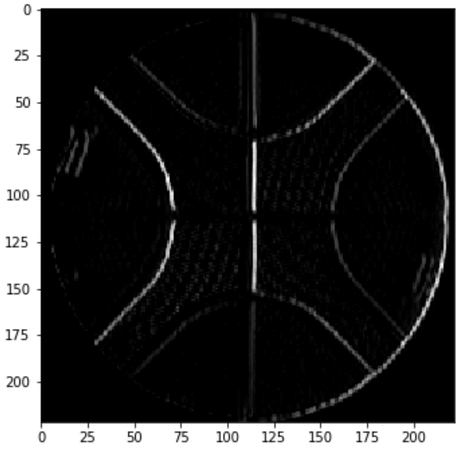
https://www.bbkong.net/fs/alleyoop/molten_BGL7

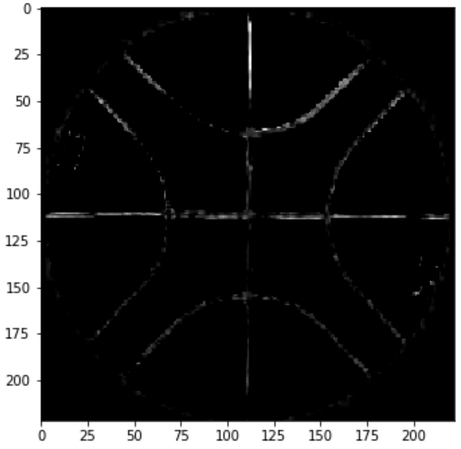












stride=(1, 1), padding='valid'

stride=(1, 1), padding='valid'

stride=(1, 1), padding='valid'

stride=(1, 1), padding='valid'

stride=(1, 1), padding='valid'

stride=(1, 1), padding='valid'

stride=(1, 1), padding='valid'

stride=(2, 2), padding='valid'

stride=(2, 2), padding='valid'

stride=(2, 2), padding='valid'

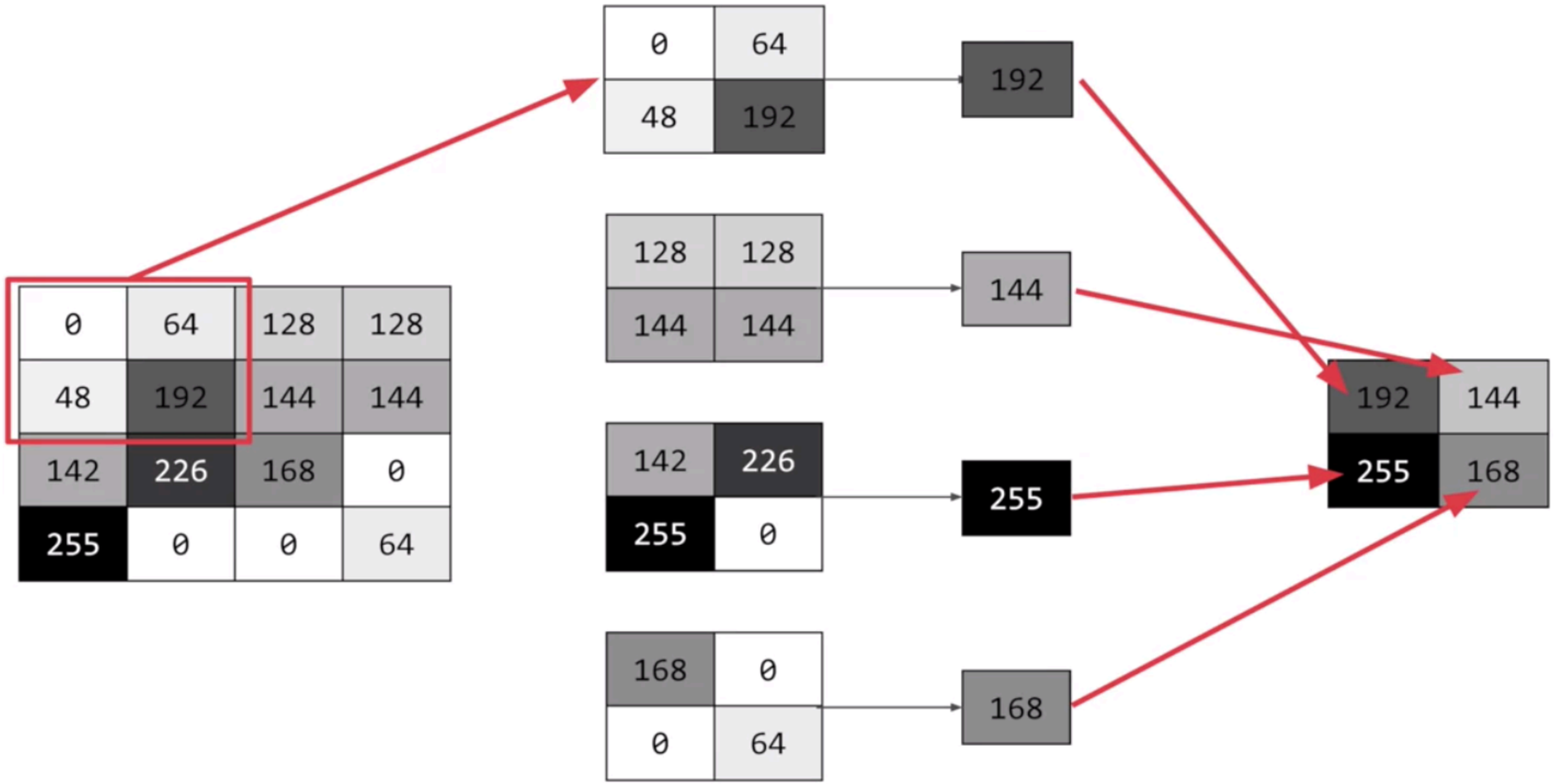
stride=(1, 1), padding='same'

stride=(1, 1), padding='same'

stride=(1, 1), padding='same'

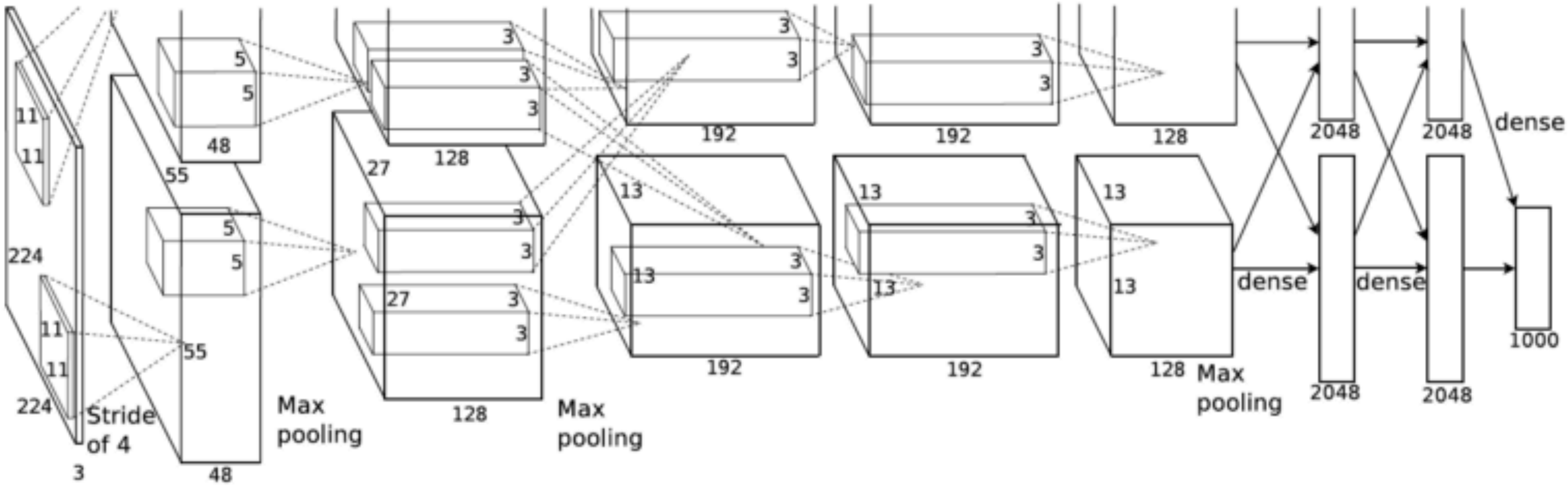
stride=(1, 1), padding='same'

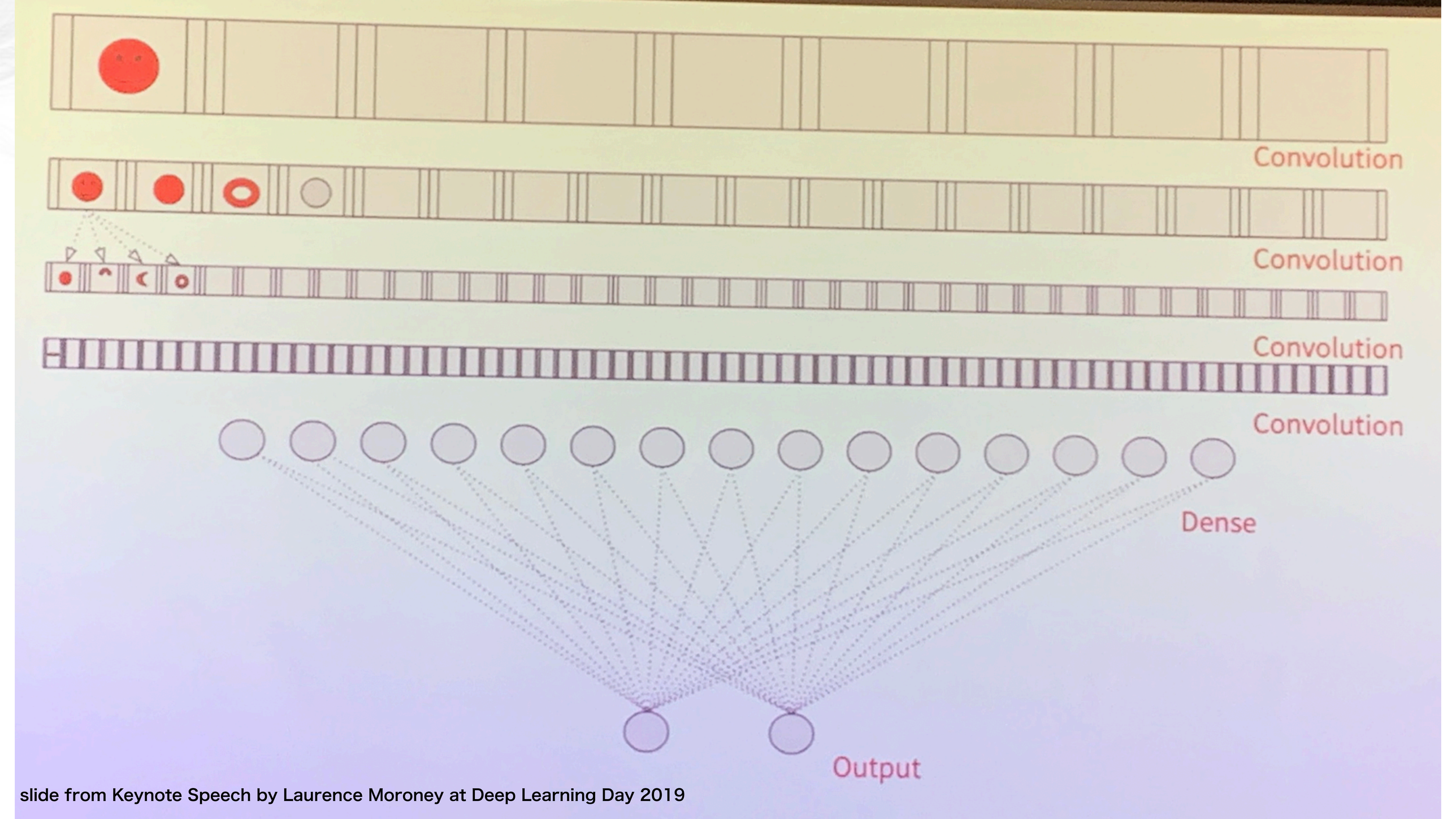
Max Pooling



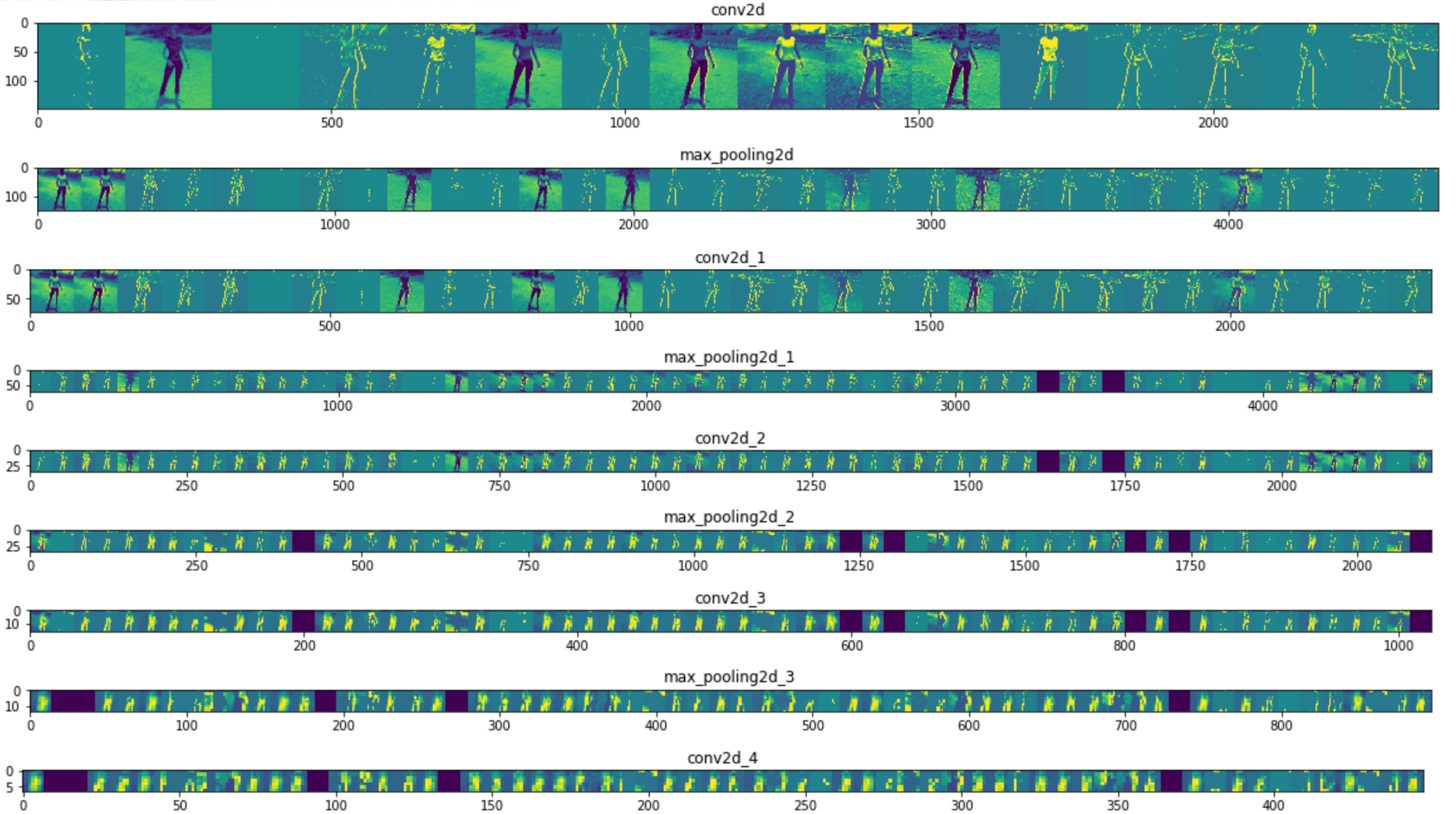
https://www.coursera.org/learn/introduction-tensorflow

AlexNet


11
11
224
224
3
Stride
of 4
5
5
48
55
48
5
5
11
11
55
Max
pooling
3
3
3
128
27
128
3
3
27
3
3
128
Max
pooling
3
3
192
3
3
192
13
13
3
3
192
13
13
3
3
192
13
13
128
Max
pooling
dense
2048
2048
dense
dense
2048
2048
dense
1000



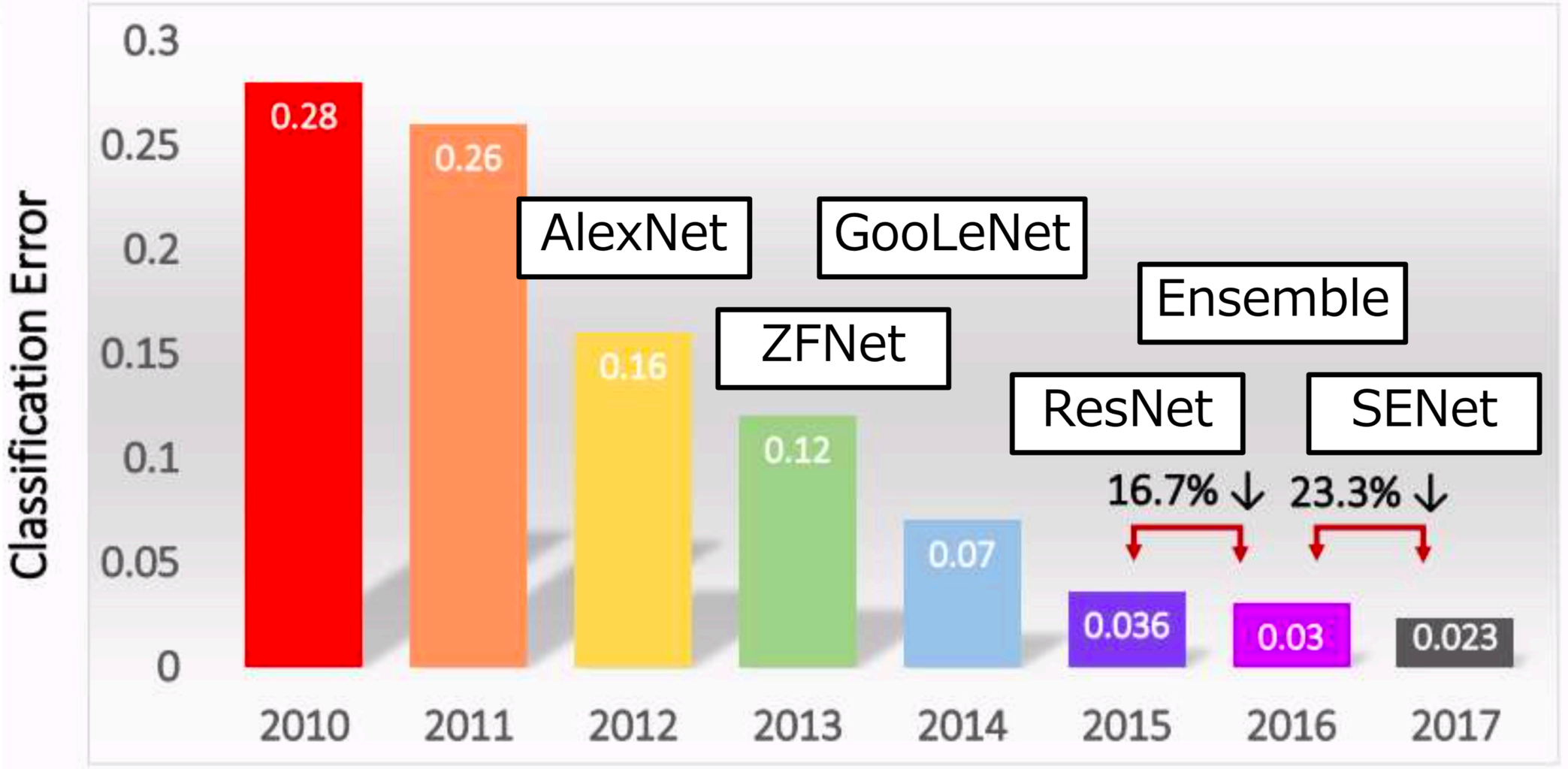
slide from Keynote Speech by Laurence Moroney at Deep Learning Day 2019



https://medium.com/@lmoroney_40129/codelabs-from-googlemlsummit-f9d53cac8d24

応用的なCNNアーキテクチャの紹介

ImageNet コンペの優勝モデル

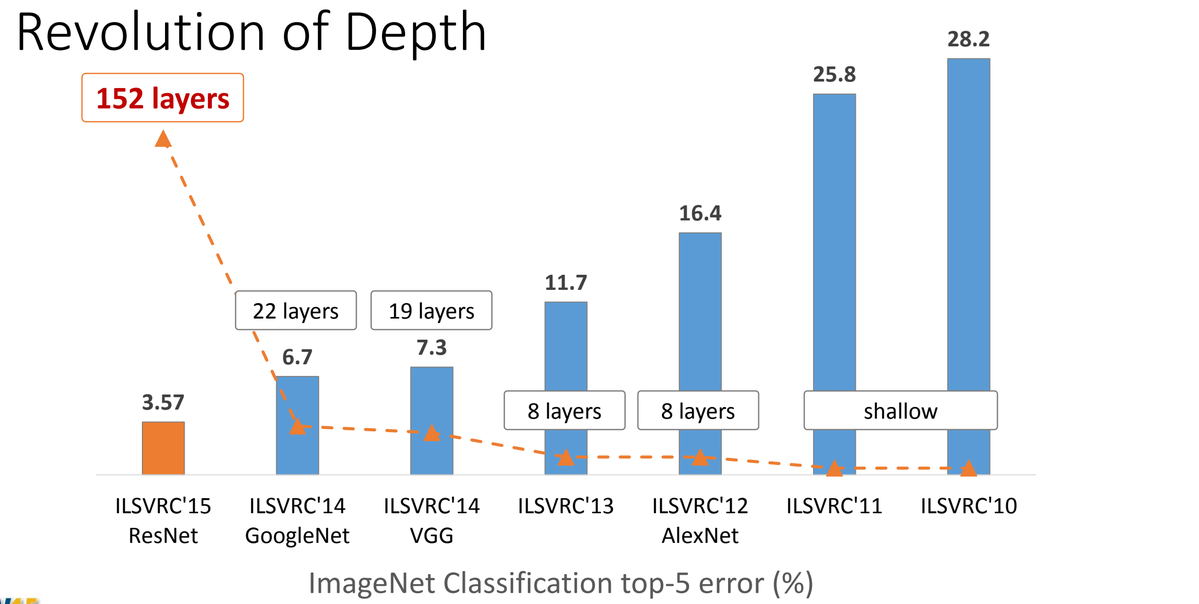


https://www.slideshare.net/ren4yu/ss-84282514

ResNet

- 2015年のImageNetコンペ (ILSVRC) 優勝モデル
- Residualモジュール（ショートカット機構）の導入

Microsoft Research



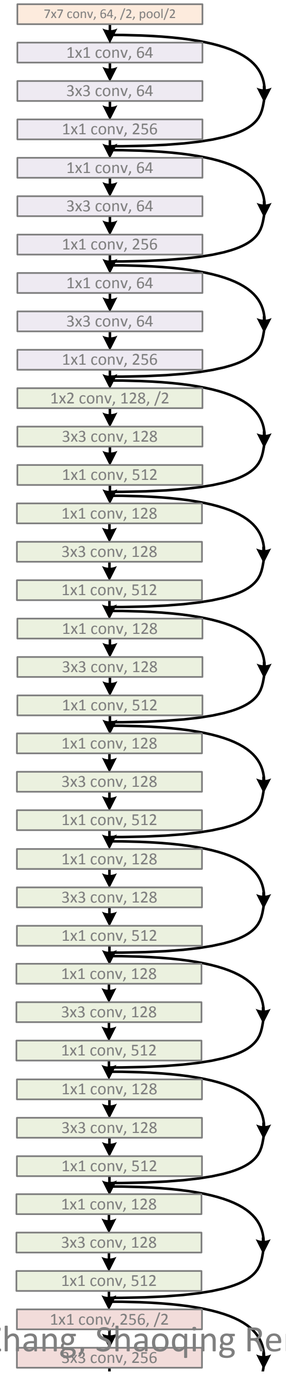
ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

http://image-net.org/challenges/talks/ilsvrc2015_deep_residual_learning_kaiminghe.pdf

Revolution of Depth

ResNet, 152 layers

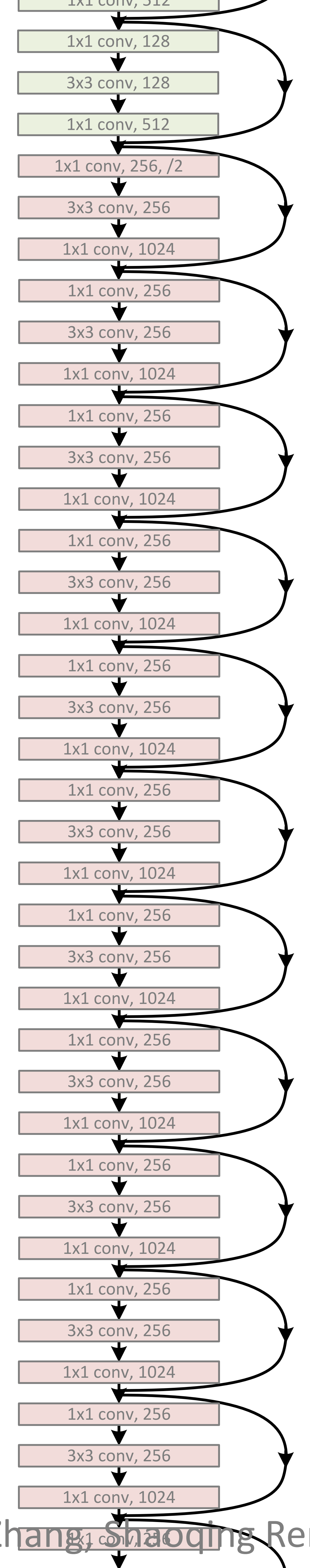


(there was an animation here)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Revolution of Depth

ResNet, 152 layers



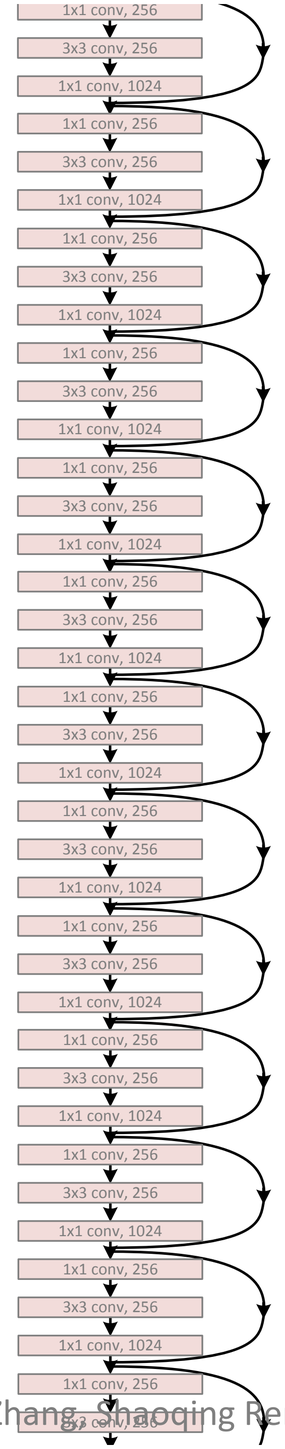
(there was an animation here)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Microsoft Research

Revolution of Depth

ResNet, 152 layers



(there was an animation here)

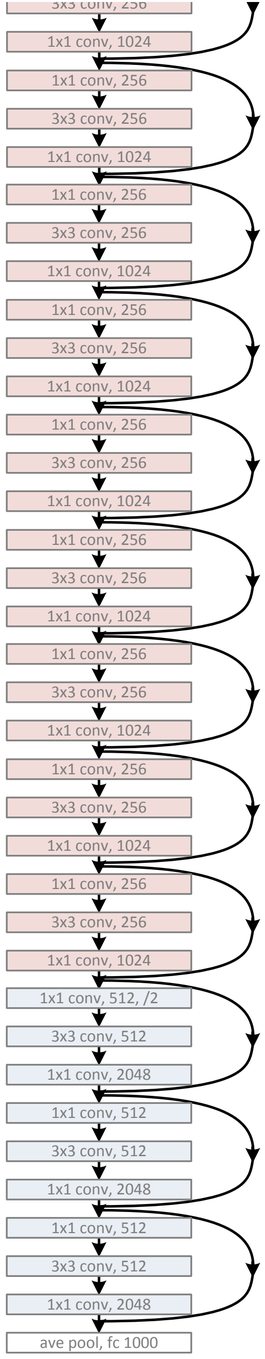
ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Microsoft Research

Revolution of Depth

ResNet, 152 layers



(there was an animation here)

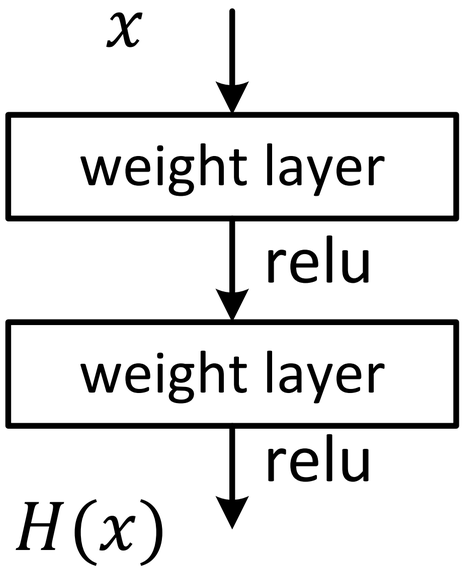
ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

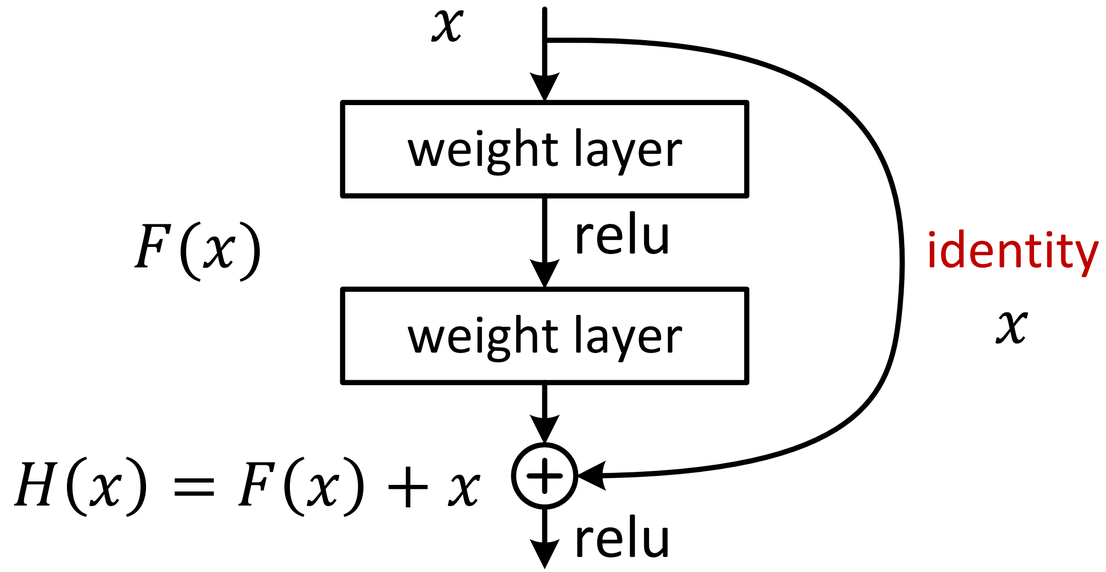
ResNet

- Plaint net

any two stacked layers

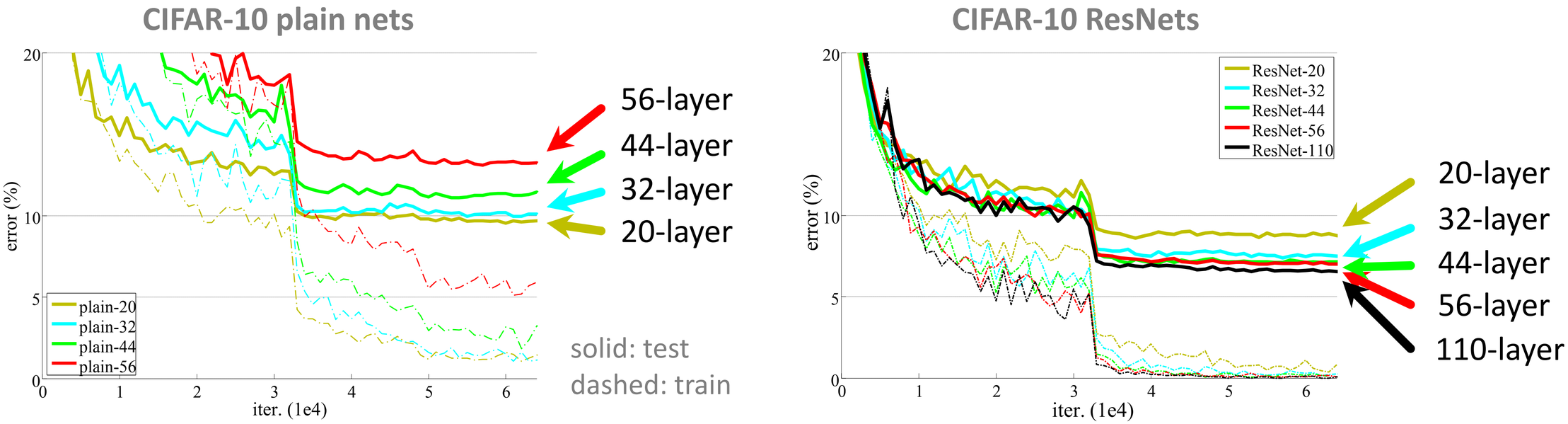


- Residual net



http://image-net.org/challenges/talks/ilsvrc2015_deep_residual_learning_kaiminghe.pdf

CIFAR-10 experiments



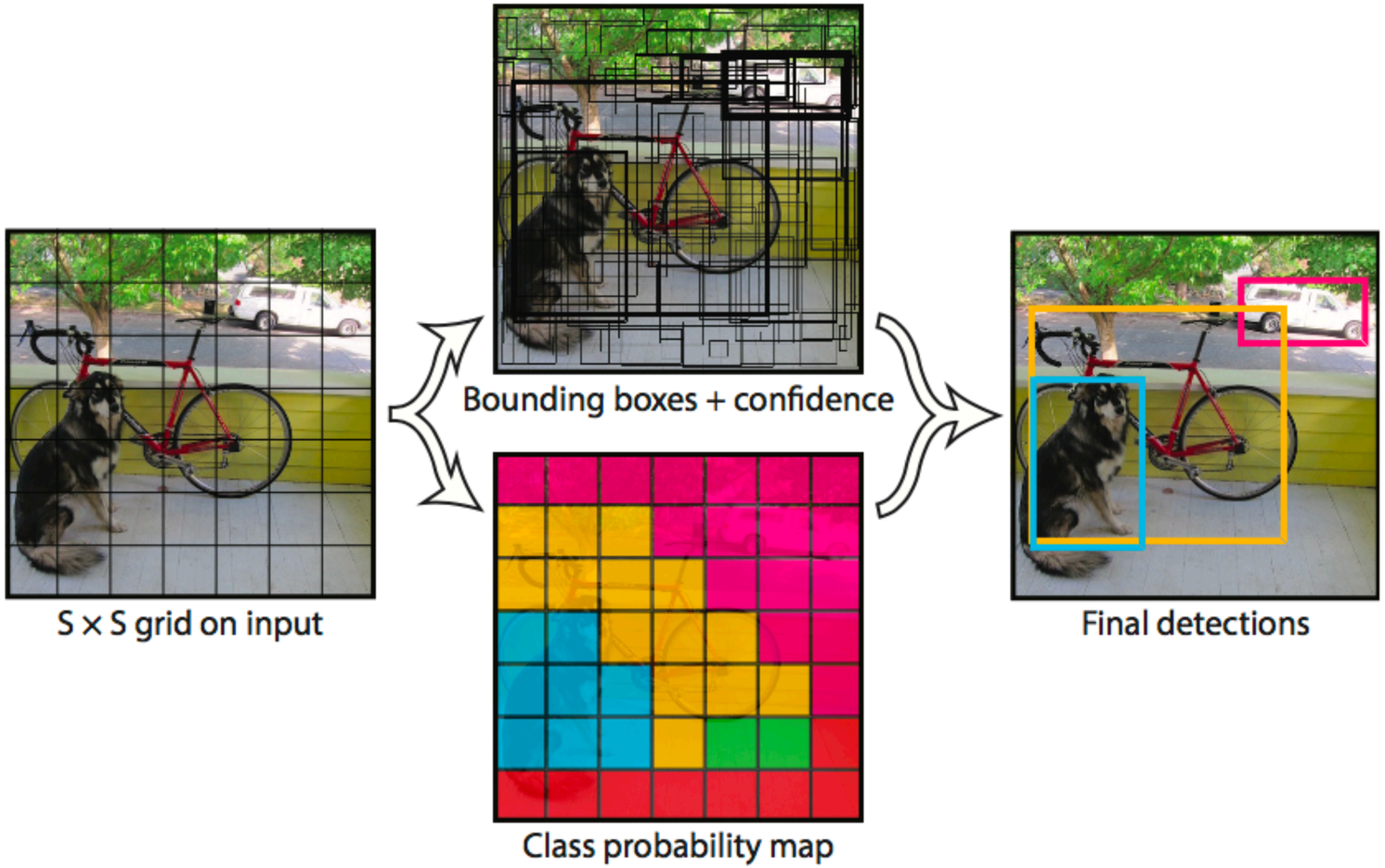
- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Further Reading

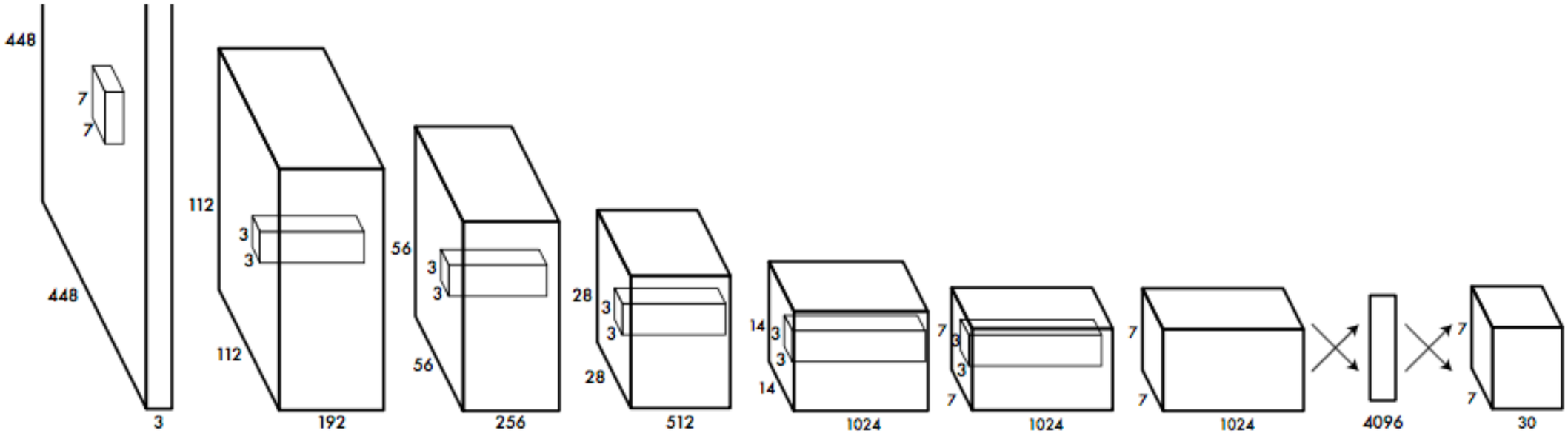
- 一年半前 (2017-12) の資料ですが素晴らしいスライドです
  - 畳み込みニューラルネットワークの研究動向 by DeNA 内田さん
  - https://www.slideshare.net/ren4yu/ss-84282514

YOLO (You Only Look Once)



https://www.youtube.com/watch?v=MPU2HistivI

YOLOのアーキテクチャ



Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256
3x3x512
} ×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

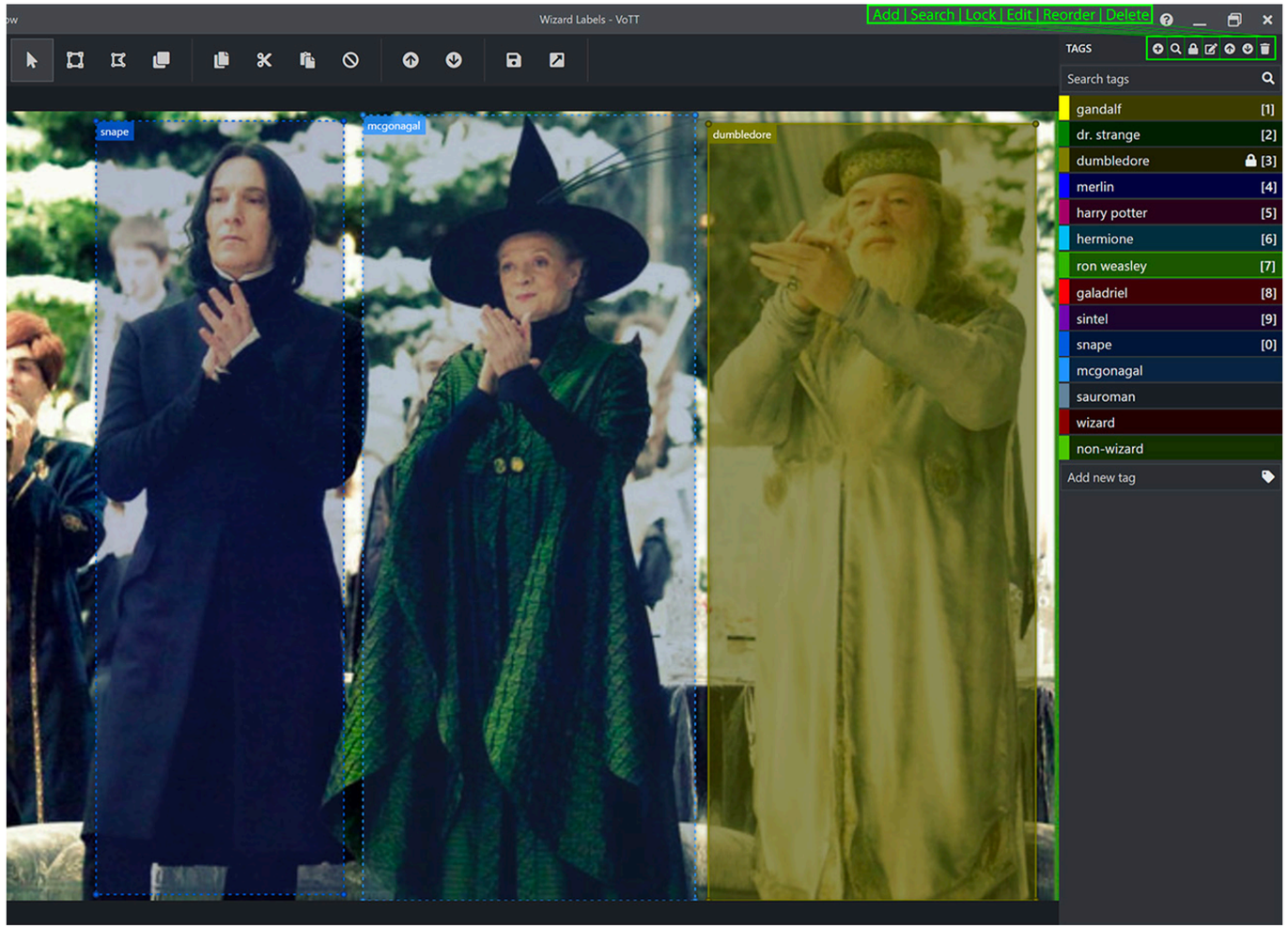
Conv. Layers
1x1x512
3x3x1024
} ×2
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

Conn. Layer

Conn. Layer

Labeling / Annotation



https://github.com/Microsoft/VoTT

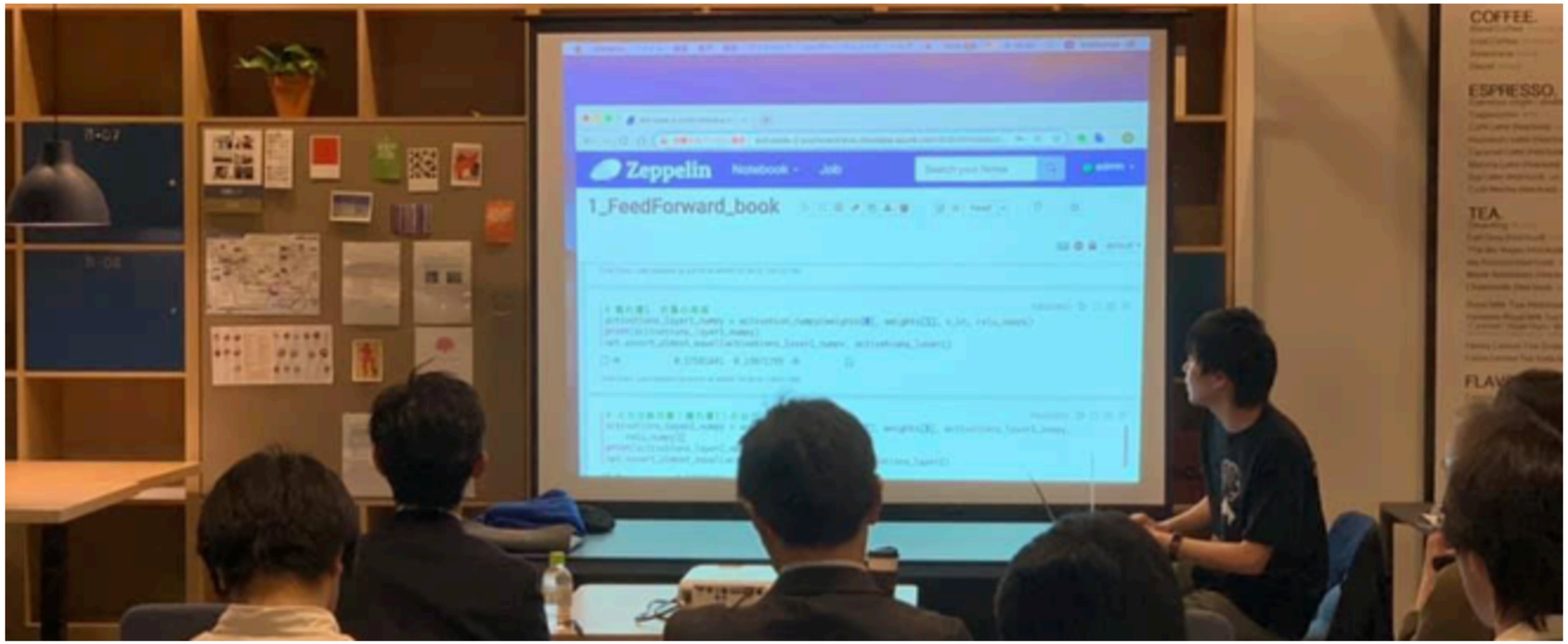
まとめ

- 機械学習ではデータからルールを確率的に学ぶ
- 画像に対して全結合の重みを最適化するのは非効率
- CNNで画像の特徴量抽出方法まで自動で行うアプローチ
- ResNetなど、さらに効率的に行うモデルが提案されている
- 何がうまくいくかは正直データによるので、経験的にうまくいきそうなアーキテクチャの組み合わせだけ用意してあとは勝手に選んで欲しい
  → Auto ML

10月
23

実践者向けディープラーニング勉強会　第七回

主催：Miki_Shingo



- https://dl4-practitioners.connpass.com/event/149810/
- 毎月開催、来月11/18はGoogleの人がAutoMLの話をしてくれる予定
- 無料