第7章

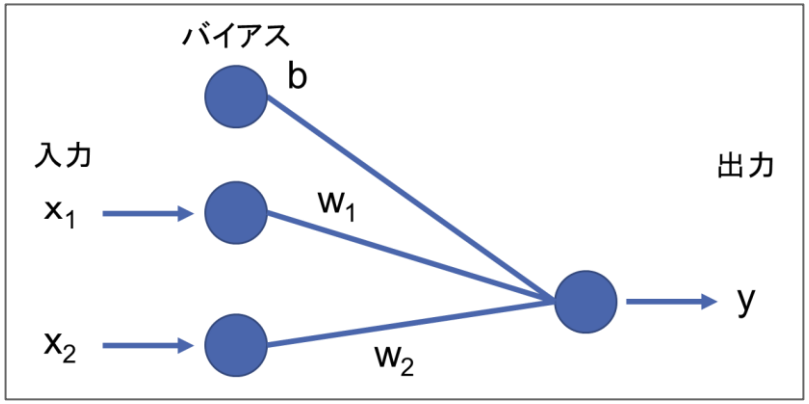
ニューラルネットワークによる分析・分類

単純パーセプトロン

入力と出力から成る最も基本的なニューロンモデル

入力Xに対して重みとバイアスを用いた計算により得られた値と教師データYの誤差を計算し、誤差が小さくなるように重みとバイアスを更新して学習

単純パーセプトロンをKerasで構築し、分類を試みる



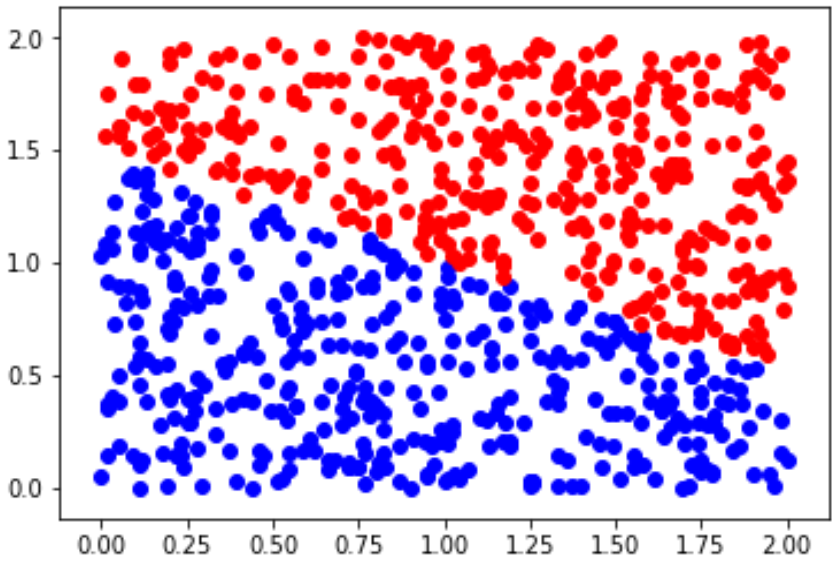
$$y = w_1 x_1 + w_2 x_2 + b$$

準備するデータセット

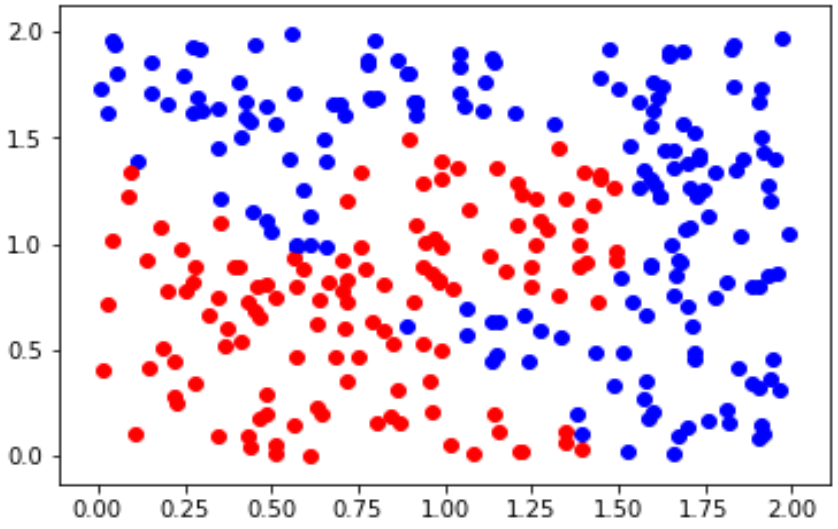
- 学習（訓練）用データセット
  - トレーニングデータセット（Training data set）
  - 入力と正解データのセット
- 学習評価用データセット
  - バリデーションデータセット（Validation data set）
  - 入力と正解データのセット
  - 学習（重みの更新）には使わないデータを評価として使う
  - 汎化性能のチェック
- 推測（予測）用データセット
  - テストデータセット（Test data set、Prediction data set）
  - 入力のみ

演習

2種類のデータに対して分類を行う



線形分離可能な単純なデータ



線形分離不可能な入り組んだデータ

演習前に

データをColabにインポートしておく
「データの読み込みと描画」の4行目でデータフォルダを設定

```
datadir = "/content/drive/My Drive/data/"
```

ソースのデフォルトはGoogle Driveに直接dataフォルダを置いた例となっているので、自分の環境に合わせて書き換える
最後に / を付け忘れないように

Kerasの基本

model: ニューラルネットワークの定義

model.add: レイヤーの追加

model.compile: 定義したネットワークの構築

model.fit: ネットワークの学習

多くの場合、history = model.fit とし、学習結果をhistroyに保存しておいて結果のプロットを行う

公式ドキュメント：https://keras.io/ja/

活性化関数、損失誤差については後述

ここではsigmoidとmean_squared_errorを使用

KerasとTensorflowの関係

本講座では、Kerasの書式によってTensorflowを使用する

もともとKerasは独立したライブラリであったが、近年Tensorflow内に取り込まれた。ただし、その後も独立したライブラリとして存在している

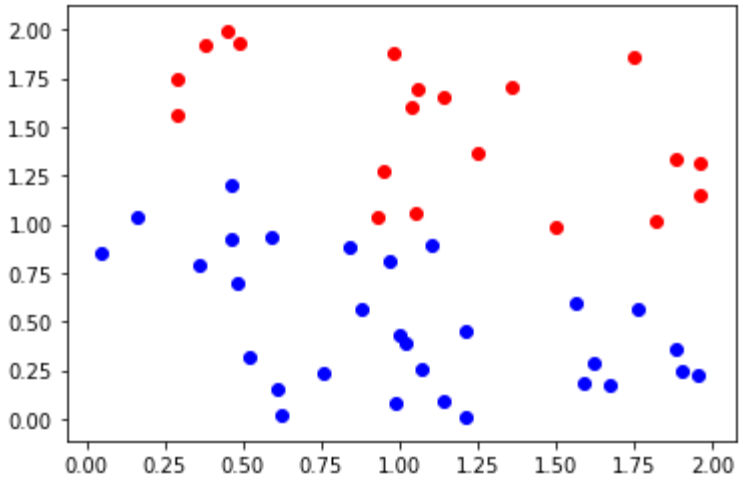
Kerasはバックエンドとして、Tensorflowの他にTheano, CNTKといったライブラリも使用できるが、ここではTensorflowのみを使用

純粋にTensorflowだけでニューラルネットワークを組む場合には、セッションという概念が必要になり、以下の記述が用いられるが、本講座ではこの書式は使用しない

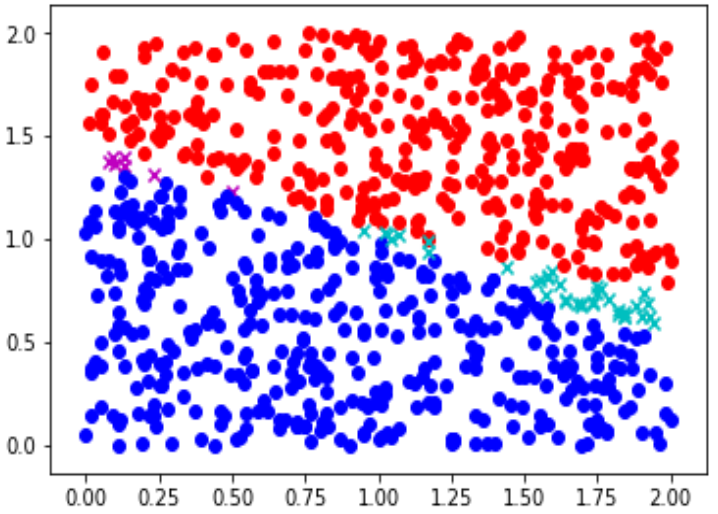
```
with tf.Session() as sess:
    print(sess.run())
```

線形分離可能なデータの結果

訓練データで学習し、評価用データで検証する



訓練データ



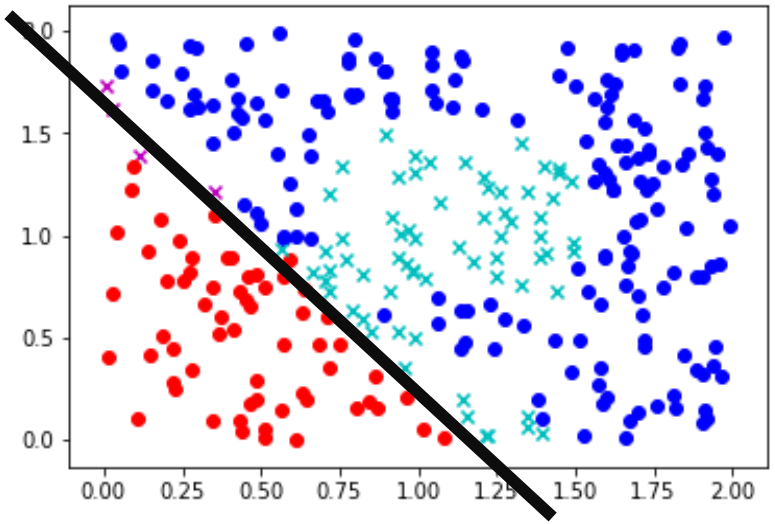
評価用データ

線形分類しかできない

単純パーセプトロンによる入力、重み、バイアス、出力の式は

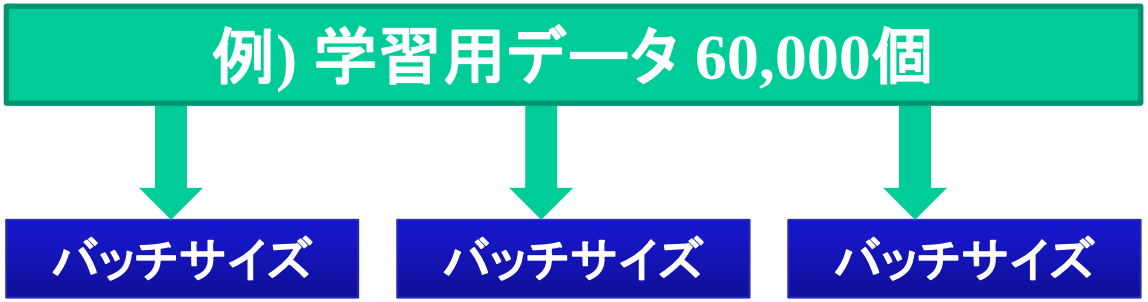
$$y = w_i x_i + b \rightarrow y = ax + b$$

すなわち、単なる1次式、線形回帰、2値予測になる
複雑なデータに対しても、次のように無理矢理1次式で分類してしまう
そこで、中間層と非線形関数の導入（活性化関数）でこれを解決する



他パラメータ

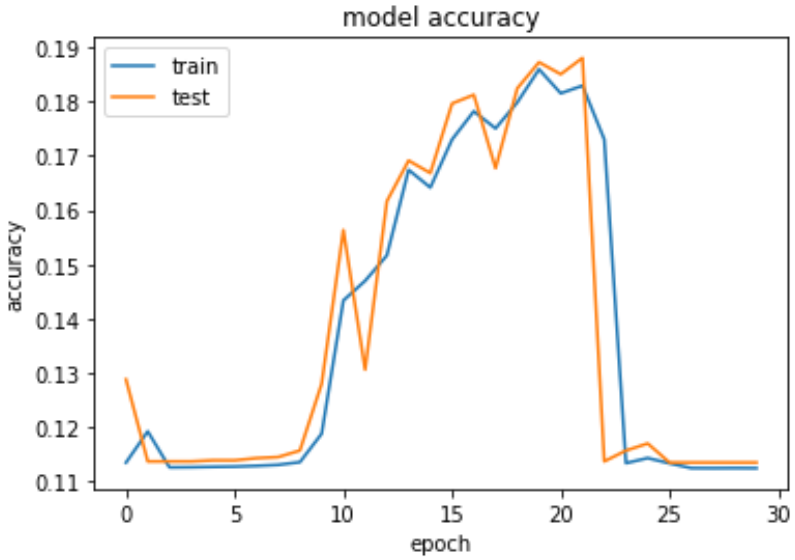
さらにバッチ数、エポック数というパラメータもある
学習用データをある程度のひとまとまりで学習するが、その数のこと
当然学習データ数より多くはできない。大抵2の乗数で設計
ある程度はバッチサイズがないと学習効果が出ない
全部で何回学習するかがエポック数。最初は少なめで傾向をつかむ



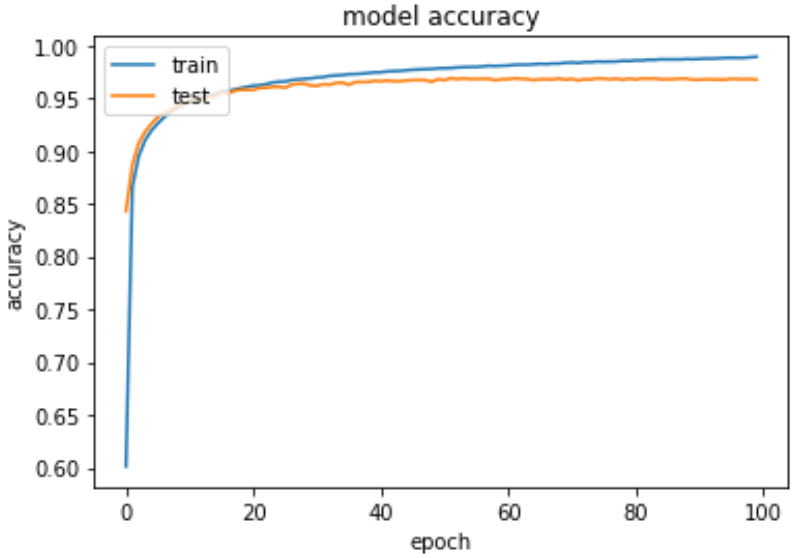
結果をどう見るか

loss（損失）が極力小さく、accuracy（正確さ）が極力大きいのが望ましい
結果のグラフで現れるのはaccuracyなので、右肩上がりが望ましい

悪い例



良い例



ロジスティック回帰

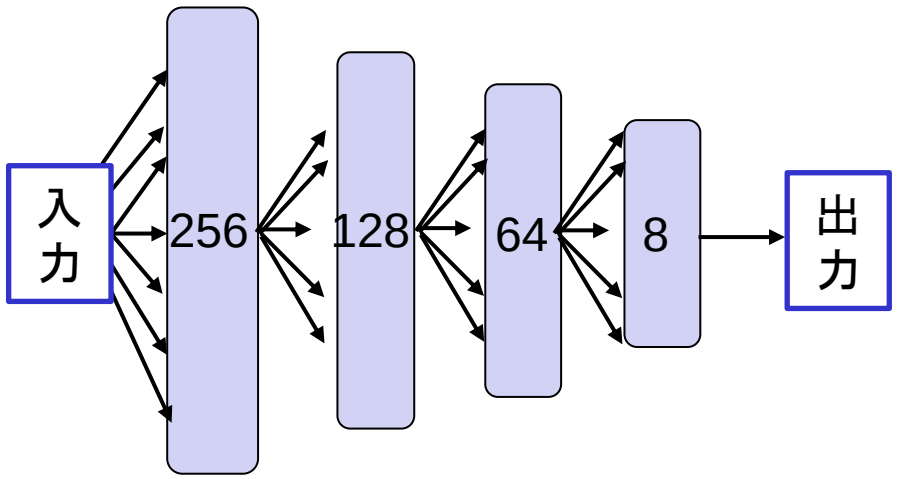
単純な線形分離のコードに対し、シグモイド関数を用いた活性化関数を追加すると、ロジスティック回帰

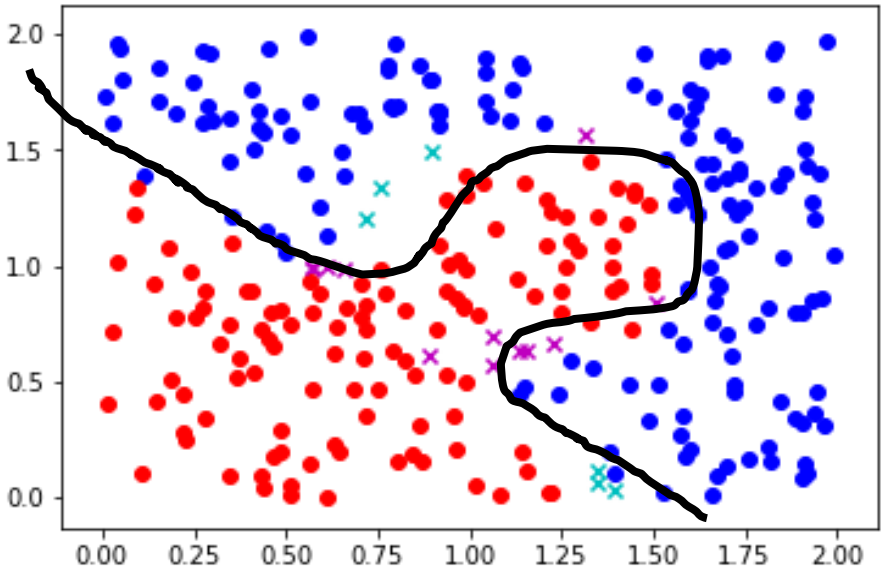
ただし、入力層と出力層のみのモデルでは線形分類しかできない

そのため、多層構造に変更する

DNN(Deep Neural Network)の演習

入出力含め6層のDNNを作成し、分類する
model.add で層を追加可能





ネットワークの設計

Kerasは krs.Sequential()に追記する形で組み立てていく

```
model = krs.Sequential()
model.add(krs.layers.Dense(256, input_shape=(train_data.shape[1],)))
model.add(krs.layers.Activation("tanh"))
```

最初の入力

活性化関数

この書式でも可

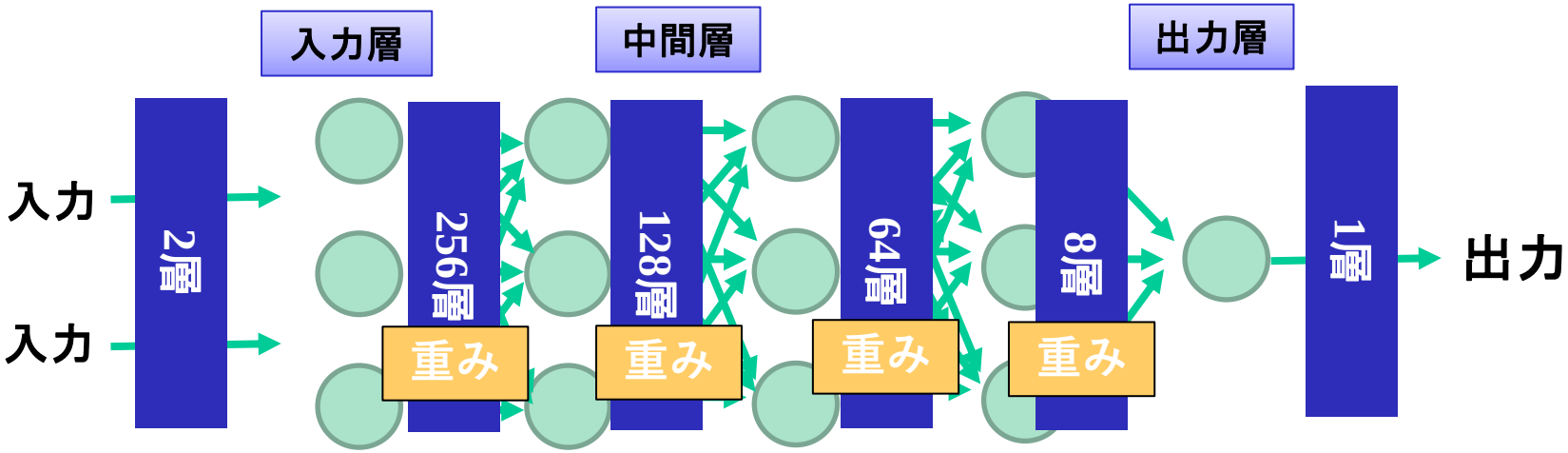
```
model.add(krs.layers.Dense(64, activation= "tanh"))
```

ニューラルネットワークの層の数

活性化関数

使用するネットワーク

すべての層が全結合された、最も基本的なDeep Neural Network
入力は画素数そのもの、出力はラベル数。途中の層は適当に決めている
もちろん増えれば増えるほど精度は向上するが、計算が遅くなる



多次元データ分類

Kaggleの天気データを用いて、データの整形と分類を実施する

元データ
https://www.kaggle.com/selfishgene/historical-hourly-weather-data

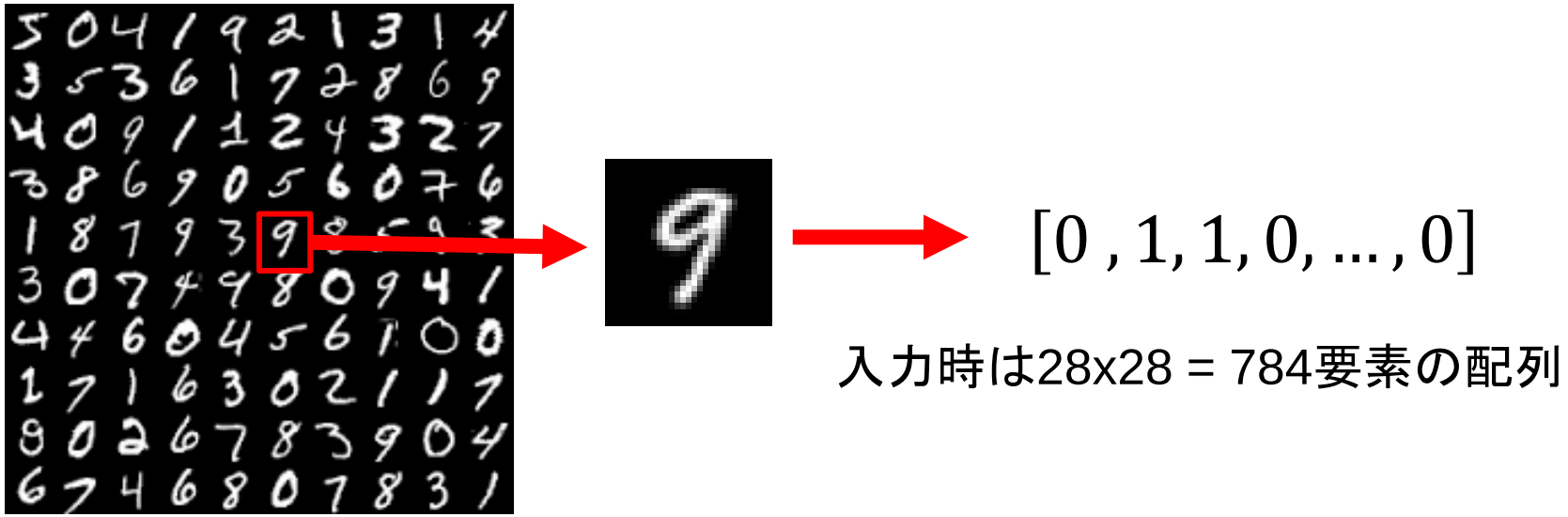
データ整形

NNに入力できるように天候データのうち、Denverの2017年データを整理
Humidity（湿度）, Pressure（気圧）,Temperature（気温）,
Wind_direction（風向）, Wind_speed（風速）, Weather（天気）を抽出
天気はweather_description.csvに書かれていて34種類ある

これを4種類に分けておく　晴れ：0、曇り：1、雨：2、雪：3
天気例）
broken clouds,曇り
drizzle,霧雨
dust,降塵
few clouds,少し曇り
moderate rain,中くらいの雨
overcast clouds,陰気な雲
proximity shower rain,すぐ近くでにわか雨

MNISTの分類

Mixed National Institute of Standards and Technology database
NY大学のYann LeCun教授らによる手書き文字DB
http://yann.lecun.com/exdb/mnist/
手書き文字を28x28で揃えたフォーマット
様々なエンジンでチュートリアルに用いられている
Kerasでは mnist.load_data() で自動的にダウンロードされ格納される



入力時は28x28 = 784要素の配列

MNISTデータの取り寄せ

基本的にKerasのチュートリアルやGithubにサンプルソースがある
MNISTデータの取り寄せはこの1行でOK

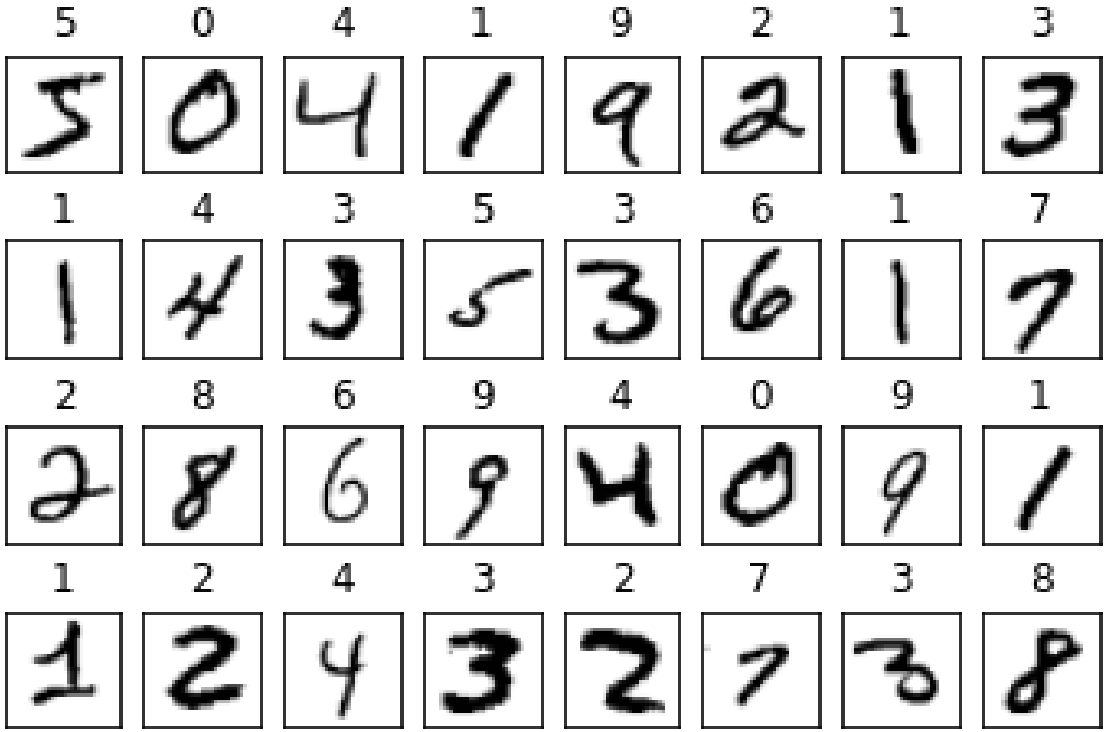
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

x_train ： 学習用（訓練・教師用）データ
y_train ： 学習用（訓練・教師用）ラベル
x_test ： 検証用データ
y_test ： 検証用ラベル
学習用データは60,000個、検証用データは10,000個

MNISTの中味

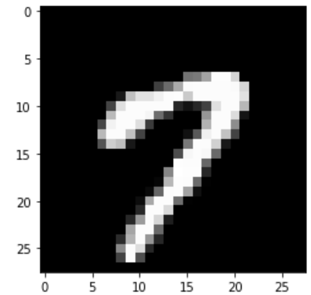
x_trainは多次元リスト

x_train[0]が数字の5, x_train[1]が数字の0 と、それぞれ28x28の画素データ



実際の中味は……

0-255の数値データ




```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 115 121 162 253 253 213   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  63 107 170 251 252 252 252 252 250 214   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  25 192 226 226 241 252 253 202 252 252 252 252 252 225   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  68 223 252 252 252 252 252  39  19  39  65 224 252 252 183   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 186 252 252 252 245 108  53   0   0   0 150 252 252 220  20   0   0   0   0   0   0]
 [  0   0   0   0   0   0  70 242 252 252 222  59   0   0   0   0   0 178 252 252 141   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 185 252 252 194  67   0   0   0   0  17  90 240 252 194  67   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  83 205 190  24   0   0   0   0   0 121 252 252 209  24   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  77 247 252 248 106   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 253 252 252 102   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 134 255 253 253  39   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   6 183 253 252 107   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  10 102 252 253 163  16   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  13 168 252 252 110   2   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  41 252 252 217   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  40 155 252 214  31   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0 165 252 252 106   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  43 179 252 150  39   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 137 252 221  39   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  67 252  79   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

ニューラルネットワークに入力するには

画像を2次元構造のまま高度に扱う手法（畳み込み）があるが、これは次回取り上げる

今回のような単純なDeep Neural Network(DNN)に入力するには、2次元構造ではなく、1次元の必要がある

そのため、[28, 28]のデータを[784]の巨大なリストにしてしまう

```
x_train_in = x_train.reshape(mnist_num, 28*28)
```

配列の形を変えるのがNumpyのreshape

正解ラベルの変換

正解ラベルもOne-hot形式に変換する必要がある

One-hot：正解が2 → **[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**

これはKerasの関数で1発変換

```
y_train_in = krs.utils.to_categorical(y_train, num_classes)
```

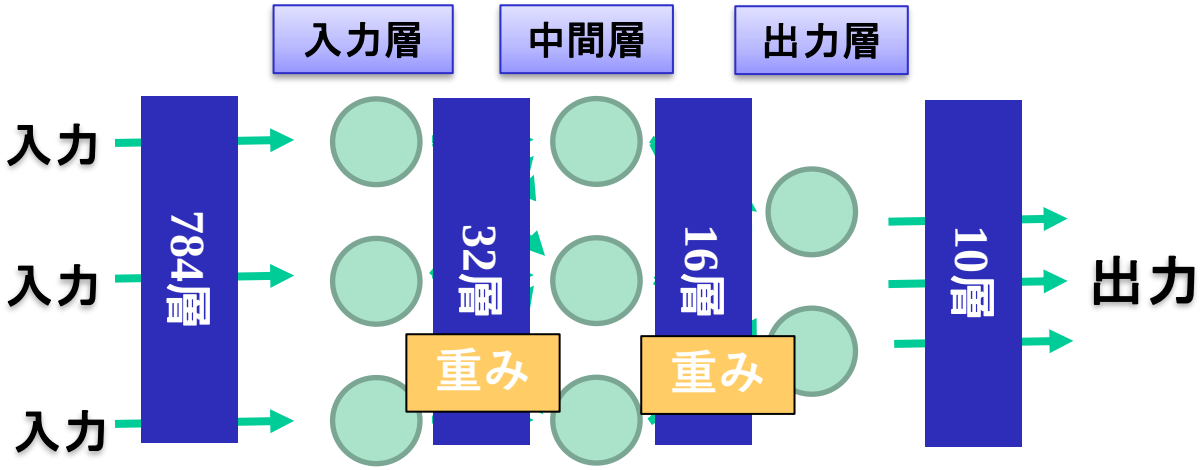
なぜこの形式が必要か

ニューラルネットワークでの分類問題は、あらかじめ設定されたラベルに対する**確率**が出力される

例）**[0, 0, 0.82, 0.04, 0, 0.1, 0.02, 0, 0.01, 0.01]** → 一番確率が高いのが2

使用するネットワーク

すべての層が全結合された、最も基本的なDeep Neural Network
入力は画素数そのもの、出力はラベル数。途中の層は適当に決めている
もちろん増えれば増えるほど精度は向上するが、計算が遅くなる



結果を良くするには

バッチサイズを増やす
ニューロンをもっと増やす
層自体を増やす

これをコピペしていけば層が増える

```
model.add(krs.layers.Dense(16, activation='relu'))
```

重みデータの保存と読み込み

指定回数の学習を終えると、各層の間のパラメータ（重み）ができあがる
これを次のコードで保存

```
model.save("mnist_predict_model.h5", include_optimizer=True)
```

保存しておくことで、良い結果のパラメータを次のコードで読み込める

```
model = krs.models.load_model("mnist_predict_model.h5", compile
 = True)
```

.h5はHDF5フォーマットのファイル
時系列データ保存用の標準フォーマットで、機械学習エンジンではほぼデファクトスタンダートのファイル形式

学習結果の検証

検証は、model.predictで行う　※検証用データをすべて引数で渡す

```
test_result = model.predict(x_test_in)
```

print(test_result[z])の結果は以下になるので

```
[0.08148291 0.1121015  0.10275728 0.10229278 0.09919951
0.09352663 0.09991397 0.10545233 0.10157305 0.10170012]
```

Numpyのargmaxで最も大きい値のインデックスを調べると

```
result =np.argmax(test_result,1)
print(result[z])
1
```

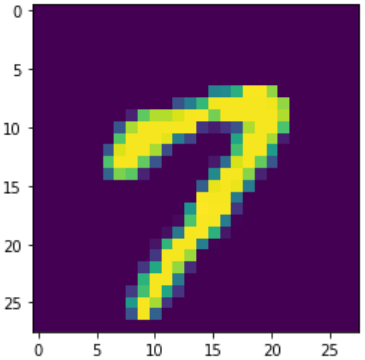
手書き文字で分類させてみる

この学習結果に、自分で書いた手書き文字を読ませてみる（用意済み）
手順は以下の通り

1. フォルダ内にpngファイルを格納し、Pythonで一括読み込み
2. OpenCVで画像ファイルを読み込む
3. Numpy配列に変換し、768個の1行配列にさらに直す

```
hand_image = images_tmp.reshape(filecount, 28*28)
predictions = model.predict(hand_image)
print(np.argmax(predictions,1))
```

根本的にデータを直す



実はもっと効率的なやり方：データの正規化
何もしなければカラー画像として計算されている
もともとモノクロ画像、さらに計算上0-1の範囲の方がよい
値が大きくなると損失関数が発散したり振動したりしやすくなる
元データはintなのでfloatに変換し、255で割る

```
x_train_nr = x_train   他データに移す
x_test_nr = x_test
x_train_nr = x_train_nr.astype('float32')   型変換
x_train_nr /= 255   255で割る
```

これでとても良い結果になる。**データの前処理はとても大事！**

第８章

Inception-V3 ,CIFAR-10

内容

- Inception-V3による画像判定
- 無料の画像データセットCIFAR-10の分類を実施
- 前回実施した全結合ネットワークDNNでの結果と、畳み込みによるCNNでの結果を比較し、CNNが画像に向いていることを確認
- 前回と同じく、Keras/Tensorflowを使用

CIFAR-10 データ元：
https://www.cs.toronto.edu/~kriz/cifar.html

Inception-V3

- Google開発の画像判別器
- ImageNetから学習: http://www.image-net.org/
- Tensorflowに動作用のサンプルが付属（ただしver 1.x）
- CNNという手法によって学習

実行してみよう

- 最初に学習済みデータを/tmp/imagenet 以下にダウンロード
- 144行目のファイル名を任意のファイルに変更

Persian cat (score = 0.64804)
tabby, tabby cat (score = 0.00998)
paper towel (score = 0.00778)
Egyptian cat (score = 0.00749)
plastic bag (score = 0.00482)

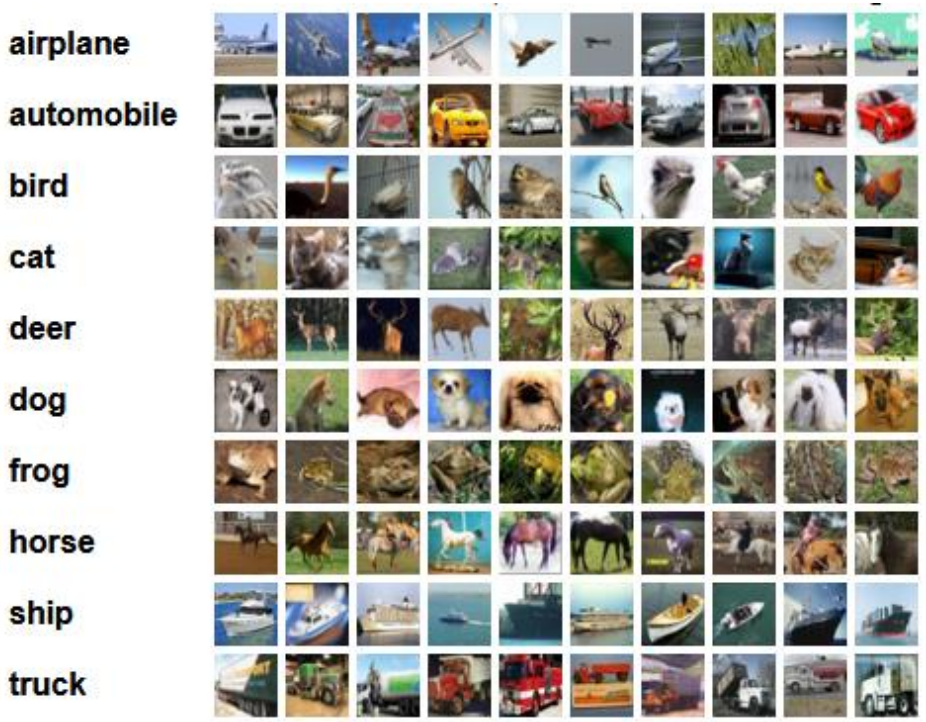
CIFAR-10をやってみる

1. CIFAR-10データの読み込みと内容チェック
2. CIFAR-10データの中味を見てみる
3. DNNで学習と推測
4. CNNで学習と推測
5. 何か他のデータで推測
6. CIFAR-100の紹介

CIFAR-10の仕様

Canadian Institute For Advanced Research

- 10種類の分類、各画像は32x32
- 5万枚の学習用データ
- 1万枚の検証用データ
- MNISTと違って**カラー画像**



CIFAR-10データの取得

MNIST同様に、1行でOK

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

※前回のMNIST

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

x_train ： 学習用（訓練・教師用）データ
y_train ： 学習用（訓練・教師用）ラベル
x_test ： 検証用データ
y_test ： 検証用ラベル
学習用データは50,000個、検証用データは10,000個

参考：CIFAR-100

- CIFAR-10の発展系
- 100種類の画像データベース
- 20種類のカテゴリ
- 各カテゴリに5種類のラベル

- 例）ラベル14: 人間
- カテゴリ内：baby, boy, girl, man, women

第9章

データの加工と可視化

データの加工で扱う内容(1)

データに関する知識
- ベクトル、行列、テンソル
- Machine readableなデータ
- 量的データ
- 質的データ
- 比例尺度、間隔尺度、順序尺度、名義尺度
- 離散データ、連続データ

数値データの前処理
- 欠損値、異常値、外れ値
- それぞれの対処

文字データの前処理
- 重複、誤記、表記ゆれ
- それぞれの対処（名寄せ）

データの加工で扱う内容(2)

適切なグラフの使い方

- 棒、折れ線、円、帯、ヒストグラム、積み上げ棒、散布図
- データの読み込み
- グラフの作成

レコードデータの加工

- データ構造の加工
  - 抽出、集約、結合、分割、生成、展開
- データ内容の加工
  - 数値型、カテゴリ型、日時型、文字型

データの加工で扱う内容(3)

マルチメディアデータの加工

- 画像データ
  - 画像読み込みと表示
  - 画像データの構造
  - 輝度値の加工
  - 輝度値標準化・正規化
  - 輝度値の可視化（輝度値ヒストグラム）

見出データの加工で扱う内容(4)

実データによる演習

- 可視化によるデータの理解
- 教師無し学習のためのデータ加工
- 教師あり学習のためのデータ加工

スカラー、ベクトル、行列、テンソル

※数学的には厳密な定義があるが、あくまでプログラミングにおけるデータ構造として考える

スカラー：1つの数値

ベクトル：1次元配列として表現されるもの

行列：2次元配列として表現されるもの

テンソル：スカラー、ベクトル、行列の一般的な概念

- 配列の次元数はテンソルの階数と対応する
- 階数0：スカラー、階数1：ベクトル、階数2：行列
- 3次元配列として表されるものは階数3のテンソル

Machine readableなデータ

総務省のオープンデータの定義

（地方公共団体のオープンデータの推進、https://www.soumu.go.jp/menu_seisaku/ictseisaku/ictriyou/opendata/）

- 営利目的、非営利目的を問わず二次利用可能なルールが適用されたもの
- 機械判読に適したもの
- 無償で利用できるもの

goo国語辞典の記述

- machine readable（機械可読）：データやコンテンツなどがデジタル化されており、機械やコンピューターで直接読み取って利用できる形式であること。具体的には、一般的なアプリケーションソフトで利用可能なファイル形式や、タグなどのマークアップによって構造化されたデータのこと、また画像・音声・動画がデジタル化されていることなどを指す。機械可読。

Machine readableとHuman readable

Machine readable

- レコードデータ
  - csvやエクセルの表
- デジタル画像
  - 行列として扱うことが可能
- その他、フォーマットが厳密に定義されているデジタルデータ

Human readable

- 紙に書かれた数値やテキストなど
- アナログデータ
- pdf
- テキストが書かれた画像
- 構造化されていない数値やテキスト群
  - 奥村晴彦, "「ネ申Excel」問題",情報教育シンポジウム2013論文集, p.93-98
- デジタルデータでもMachine readableでないものもある

機械学習の入力となるデータ

機械学習では、同一の形のテンソルとなるデータを入力とする

- ベクトルであれば、使うデータは全て同じ要素数のベクトル
- 行列であれば、全て同じ行数と列数の行列

例えば↓のような形のデータは使えない（要素数の異なるベクトル）

データ1：2,3,1,5,4

データ2：3,5,4

データ3：2,5,3,2

量的データと質的データ

量的データ

- 個数や長さ、重さのような数量を表すもの
- 5個、10cm、2.5kg、15000円、100km/sなど

質的データ

- 性別や生物の種など、数量でない分類項目を表すもの
- 男/女、犬/猫/ネズミ、A/B/O/AB、大/中/小、など

4つの尺度

質的データ

名義尺度

- 単に分類するためにつけたラベル

順序尺度

- 順序には意味があるが、その間隔には意味がない数値を割り当てたもの

量的データ

間隔尺度

- メモリが等間隔であるもの、または等間隔と仮定されているもの

比例尺度

- 原点（0）の決め方が決まっており、間隔と比率両方に意味があるもの

情報量としては、名義＜順序＜間隔＜比例

- 大は小を兼ねることができる
- 順序尺度で意味があるものは間隔尺度以上でも意味が有る

名義尺度

プログラミングでは名義尺度のラベルとして整数を使うことが多い

- 2種類ならブーリアン型を使うこともある
- 0：猫、1：犬、2：ネズミ、3：コアラ、・・・など

区別するためだけに用いるため、等しいか等しくないかだけに意味がある

意味が有る統計量

- 度数、最頻値

意味が無い統計量

- 平均、分散、標準偏差など、ラベルの値を使った計算全般

順序尺度

こちらもプログラミングではラベルとして数値を使うことが多い

- 1：1位、2：2位、3：3位、・・・など

大小の比較は可能だが、間隔や比率などの「どのくらい」には意味が無い

- 1位と0.5ゲーム差の2位と、1位と10ゲーム差の3位でも、データとしてわかることは1位、2位、3位という順序だけ（ゲーム差は比例尺度）

意味が有る統計量

- 度数、最頻値、中央値

意味が無い統計量

- 平均、分散、標準偏差など、ラベルの値を使った計算全般

便宜的に順位などの値を
比例尺度のように扱うこともある

間隔尺度

知能指数や摂氏温度、アンケート調査など

プログラミングでは整数型、実数型の数値データとして扱われる

意味が有る統計量

- 順序尺度で意味が有るもの、値の和と差、平均、分散など

意味が無い統計量

- 比率
- 数値データなので比率を計算できるように見えるが、その値に意味は無い
- 摂氏30℃は摂氏10℃より20℃暑いが、3倍暑いとは言えない
- アンケート調査
  - 回答と値の割り当てによって比率は一定ではない、差は一定

便宜的に値を
比例尺度のように
扱うこともある

比例尺度

身長、体重、金額、絶対温度など

和差積商の計算がすべて意味のある値となる

全ての統計量が意味のあるものとして計算可能

数値データの前処理

欠損値

- 値が無い状態
- コンピュータプログラムが実行時エラーを起こす

異常値

- 主に計測ミスや記録ミスによって起こるありえない値
  - 正の値しかとらないのに負の値（データ範囲の異常）
  - 部屋の温度の記録で3000℃（現実的でない値）

外れ値

- 明らかな異常とは判定できないが、他の値から大きく外れた値
- 実用上は、外れ値と異常値を区別できない場合もあるので注意が必要

欠損値の対応

値の除去

- リストワイズ除去
- ペアワイズ除去

何らかの値で埋める

- 決められた値（平均値、中央値など）
  - 欠損値を意味する何らかの値を使うこともある（0など）
- 前の行の値など、近傍の値
- k-近傍法での予測値

値の補間

- 線形補間
- スプライン補間
- その他の曲線による補間

値の除去

A	B	C
1	0.3	3
3	0.1	4
4		8
5	0.4	3

欠損値のあるデータ

A	B	C
1	0.3	3
3	0.1	4
5	0.4	3

リストワイズ除去
欠損値のある行そのものを除去

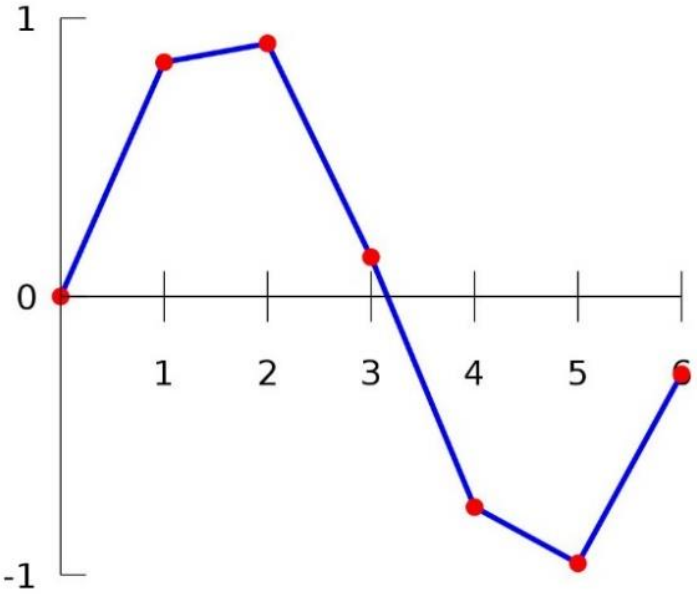
A	C
1	3
3	4
4	8
5	3

ペアワイズ除去
目的に必要な列だけ抜き出した後、リストワイズ除去

値の補間

線形補間

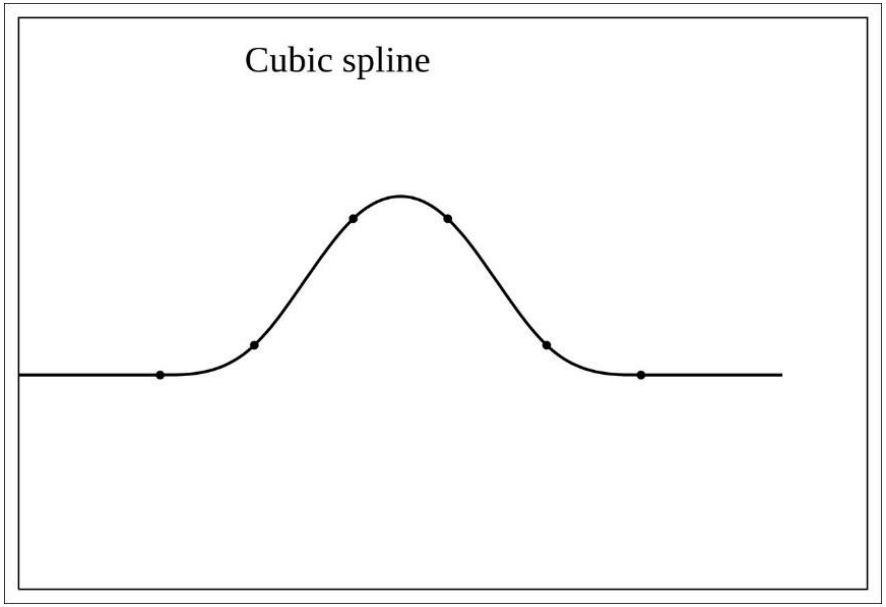
- 線形（1次）多項式を使った補間



https://ja.wikipedia.org/wiki/線形補間

スプライン補間

- スプライン曲線を使った補間



https://ja.wikipedia.org/wiki/線形補間
Stamcose - 投稿者自身による作品, CC 表示-継承 3.0,
https://commons.wikimedia.org/w/index.php?curid=16560057による

異常値の対応

値の修正が可能かどうかを検討する

- 数値の読み間違えなど

除去が可能かどうかを検討する

- 除去が可能であれば、対応は欠損値の対応に準ずる

異常値の対応には、対象と計測方法の知識が必要

外れ値の対応

外れ値があると、正規化や予測モデルの構築に悪影響を与える

しかし、外れ値を除去することは、極端な値の状況を考慮しないことになる

慎重に除去が可能かどうかを検討する必要がある

外れ値の見極め

- データを可視化して、目視で見つけた外れ値を恣意的に除去する
- 正規分布を前提にして、平均値と標準偏差を使って見つける
  - 平均値±定数×標準偏差
  - 定数は3以上を使うことが多い
  - 正規分布の場合、平均±3×標準偏差の範囲に約99.73%の値が収まる

文字データの前処理

重複、誤記、表記ゆれデータはまとめる

表記ゆれ

- 異字体（斎藤、斉藤、齋藤、齊藤など）
- コンピュータとコンピューターなど

その他

- アルファベットや数字の全角文字と半角文字
- 空白文字（スペースとタブ）、区切り文字（, と ; など）

データの不整合性に対する対処をデータクレンジングと呼ぶ

人手で行う必要がある作業も多い

pythonにおけるデータの読み込み

pandasによるcsvファイルの読み込み

- csv形式のレコードデータ
- 「09_1_データの読み込みと可視化.ipynb」を参照

読み込んだデータは、データフレーム型となる

- 形としてはいわゆる2次元配列
- 列ごとに異なるデータ型が可能
- numpyのarray形式は全て同じ型

グラフの種類

「09_1_データの読み込みと可視化.ipynb」を参照

棒グラフ： pyplot.bar

折れ線グラフ：pyplot.plot

ヒストグラム：pyplot.hist

散布図：pyplot.scatter

棒グラフ

pyplot.bar

主な引数

left (必須)	各棒の X 軸の数値
height (必須)	各棒の高さ
width	棒の太さ (デフォルト値: 0.8)
bottom	各棒の下側の余白。(積み上げ棒グラフを作るときに使用)
color	棒の色。
edgecolor	棒の枠線の色
linewidth	棒の枠線の太さ。
tick_label	X 軸のラベル

折れ線グラフ

pyplot.plot

主な引数

xdata (必須)	X 軸方向の数値
ydata (必須)	Y 軸方向の数値
linewidth	線の太さ
linestyle	線のスタイル。 solid（デフォルト）, dashed, dashdot, dotted
color	線の色
marker	マーカーの種類。 参考 https://matplotlib.org/api/markers_api.html#module-matplotlib.markers
markersize	マーカーの大きさ
markeredgewidth	マーカーの枠線の太さ
markeredgecolor	マーカーの枠線の色
markerfacecolor	マーカーの塗りつぶしの色

散布図

pyplot.scatter

主な引数

x, y	グラフに出力するデータ
s	サイズ (デフォルト: 20)
c	色、または、連続した色の値
marker	マーカーの形 (デフォルト: ‘o’= 円)
cmap	カラーマップ。c が float 型の場合のみ利用可能です。

ヒストグラム

pyplot.hist

主な引数

x (必須)	度数分布ではない生データの配列。
bins	階級数。(デフォルト: 10)
range	ビンの最小値と最大値を指定。(デフォルト: (x.min(), x.max()))
align	各棒の中心を X 軸目盛上のどの横位置で出力するか。 left, mid（デフォルト）, right
orientation	棒の方向。 horizontal, vertical(デフォルト)
rwidth	各棒の幅を数値または、配列で指定。
color	ヒストグラムの色。
label	凡例

pandasによる欠損値処理

「09_2_レコードデータの加工.ipynb」を参照

欠損値の発見

- isnull()

欠損値の除去

- dropna()

欠損値の置換

- fillna()
  - 直近の前の値
  - 直近の後の値
  - 特定の値

Pandasによる異常値、外れ値処理

「09_2_レコードデータの加工.ipynb」を参照

異常値、外れ値の発見

- ソートしてグラフ描画し、目視で確認
- 平均値と標準偏差による切り分け

異常値、外れ値の除去

- 欠損値処理に準ずる

レコードデータの加工

データ構造の加工

- 行や列に対して、抽出、集約、結合、分割を行うことで、入力データとして加工する
  - 機械学習の入力データは、テンソルの形
- 取得できたデータをそのまま入力データにできないこともある
  - 余分なデータが含まれる
  - 複数ファイルに収められている

データ内容の加工

- 型変換
- 比例尺度から名義尺度・順序尺度へ（数値データからカテゴリデータへ）
- カテゴリデータの集約（細かいカテゴリから大まかなカテゴリへ）

マルチメディアデータの加工

画像データ

- 画像データは、色チャンネル数を含めた3次元配列（3階テンソル）として扱われる
  - グレースケール：1チャンネル
  - カラー：3チャンネル
  - 透過画像：4チャンネル（カラー＋アルファ）

音声データ

- 音声データは、波形データ（信号の強さの時系列データ）として扱われる

画像データの加工

「09_3_画像データの加工.ipynb」を参照

OpenCVによる画像データ加工

- 画像データはnumpyのarray形式のデータとして扱われる
- numpyによる数値操作が可能
- OpenCV独自の処理も可能

Jupyter notebook上でmatplotlibによる画像表示をする場合は、色チャンネルの順番が逆になるので注意が必要

輝度値の加工

画素値は0～255の整数

OpenCVで画像を読み込んだ時点では、uint8（符号なし8ビット整数）型

輝度値を操作して実数にする場合は、型変換しないと桁落ちする

numpyの値操作

- 配列全体を対象にした演算
- 要素の直接演算
- スライス記法による対象範囲の指定
- 平均や標準偏差の計算

輝度値の正規化、標準化

画像の正規化（値を0～1の範囲にする）

- 画素値は0～255の範囲であるため、単純にそれぞれの画素値を255で割る

画像の標準化

- 画像全体の画素値の平均と標準偏差を計算し、それぞれの画素値から平均値を引いて標準偏差で割る

輝度値の分布の可視化

輝度値ヒストグラム

- 0～255の画素値の数（画素値の度数分布）をヒストグラムにする
- 正規化や標準化した画像の場合でも、最小値～最大値の範囲でbin（階級）数を256としたヒストグラムを作ると分布は変わらない
- 隣接画素の情報は持っていないが、画素値の分布は画像特徴量の一つ
- カラー画像の場合は、3つのチャンネル全体でヒストグラム化する場合と、3つのヒストグラムを作成する場合がある。
- matplotlibのhistでヒストグラムを作成する場合、1つのベクトル（1次元配列）が1つのデータ群であるため、2次元配列の場合は平坦化（1次元配列化）する必要がある。

実データによる演習

Kaggleのデータを使った演習

可視化によるデータの理解

- 必要に応じてMachine readableに加工し、グラフ化してデータの傾向を観察する

教師無し学習のためのデータ加工

- Machine readableなデータに対して、欠損値処理、異常値処理を行い、機械学習の入力となるテンソルの形に加工する

教師あり学習のためのデータ加工**（今回はこちらを演習）**

- テンソルの形のデータと教師データを作成し、学習用データと評価用データに分割する

Kaggleのデータを使った例（1）

Craft Beers Dataset
2K+ craft canned beers from the US and 500+ breweries in the United States.

https://www.kaggle.com/nickhould/craft-cans

「09_4_データ加工の例_beers.ipynb」を参照

styleの分類がこのデータから可能かを調べるための予備分析のようなことを行う

データの内容と型

Unnamed: 0	int64	上から順のID
abv	float64	アルコール度数 (Alcohol By Volume)
ibu	float64	苦みの強さ (International Bitterness Unit
id	int64	ID（nameに対応）
name	object	銘柄
style	object	ビールの種類（ラガー、エールなど）
brewery_id	int64	醸造所ID
ounces	float64	一缶の量（オンス）

Craft Beers Datasetの加工

- データ種類のチェック
  - 数値データとして意味を持つものは3つ
  - abv, ibu, ounces
- 欠損値のチェック
  - 欠損値があるので欠損値処理を行う
- 外れ値のチェック
  - 今回は外れ値無しとみなす
- 分類目的のstyleの様子を観察
  - styleの数、style毎の銘柄数
  - ごく少数の銘柄しかないstyleもある
  - 今回は仮に10以上の銘柄があるstyleを対象とする
- 相関を見るために散布図を描く
- 主成分分析で次元圧縮し、主成分で散布図を描く
  - 特徴量としての主成分得点

- 教師なし学習＝入力と同じデータを教師とする学習
  - 教師あり学習データの入力のみ

Kaggleのデータを使った演習（2）

Solar Radiation Prediction. Task from NASA Hackathon.

- https://www.kaggle.com/dronio/SolarEnergy

「09_5_データ加工の例_solar.ipynb」を参照

データの内容と型

UNIXTime	int64	Unix時間（1970年1月1日午前0時0分0秒からの経過時間）
Data	object	日付（Dateのスペルミス？）
Time	object	時刻
Radiation	float64	日射量
Temperature	int64	気温（華氏）
Pressure	float64	気圧（Hg）
Humidity	int64	湿度（%）
WindDirection(Degrees)	float64	風向（角度）
Speed	float64	風速（miles / h）
TimeSunRise	object	日の出時刻
TimeSunSet	object	日没時刻

Radiationの値を
他のデータから予測する

Solar Radiation Predictionの加工

- 欠損値の確認
- 折れ線グラフでのデータの確認
- 散布図でのデータの確認
- データ加工（派生データ作成、昼間の時間）
  - 日付データへの変換
  - 時間データへの変換
  - 数値型への変換
- 相関行列のヒートマップ
  - radiationの予測に影響しそうなデータを目視で確認

教師あり学習のためのデータ作成

「09_6_教師あり学習データ作成.ipynb」を参照

- 不使用のデータ列を削除
- 学習用データと評価用データに分割

「09_7_教師あり学習演習.ipynb」を参照

- DNNで教師あり学習を行い、回帰モデルを作成する
  - Radiationを予測する
  - どのようなデータを入力とするか、標準化または正規化の有無、DNNの構造、バッチサイズ、その他さまざまな要素を考慮して精度を上げてみてください。

第10章

リアルタイム検出
物体検出手法, VGG-16, SSD

リアルタイム検出

実機のWebカメラを使用して、リアルタイムであるターゲットを認識させる

必要なもの：

- 対象画像（学習用、評価用）
- 背景画像（学習用、評価用）
- データセットを記述したcsvファイル

かなりの部分はこちらで用意してありますが（料理番組方式）、本当はこれらを全部自分で用意します。ぜひ1度自分でチャレンジしてください

リアルタイム検出の手順

次の順番で行う

1. 対象物を撮影する
2. 背景画像を撮影する
3. 対象物画像を水増しして300枚用意する
4. 背景画像を分割して625枚用意する
5. 対象物画像と背景画像をcsvファイルに記述する
6. 用意したデータで学習を行う
7. 学習されたデータを用いて、リアルタイムで判定させる

対象物の撮影

対象物をカメラアプリで撮影し、128x128のサイズで切り出す
正方形で切り出すため、なるべく縦横比が極端でないものを選ぶ
顔は意外と認識されにくい（頑張って下さい）
手のひらは割と簡単。スマホケース、ペットボトルの模様、社員証など
ファイル名はtarget.jpgとする。（詳しい手順は、この後の対象画像作成手順を参照）

今度は、同じ位置で背景のみで撮影し、ファイル名をother.jpgとする。サイズ等は変更しない

撮影

対象物が入った写真を撮影 →　対象物だけ切り抜いてtarget.jpg
次に対象物がない写真を撮影　→　other.jpg

認識させたい対象物

自分が映り込まないようにして撮影

指定サイズでの切り出し(1)

正方形での切り出しが可能なXnViewを使用する

https://forest.watch.impress.co.jp/library/software/xnview/ などからダウンロード（XnViewで検索すると窓の杜が最初にヒット）

もちろん、オフィシャルサイトからDLしてもよい。
https://www.xnview.com/

XnViewを起動し、カメラアプリで撮影した対象物のファイルを読み込む
もし画像が鏡映しになっていたら、画像 → 反転 → 左右反転

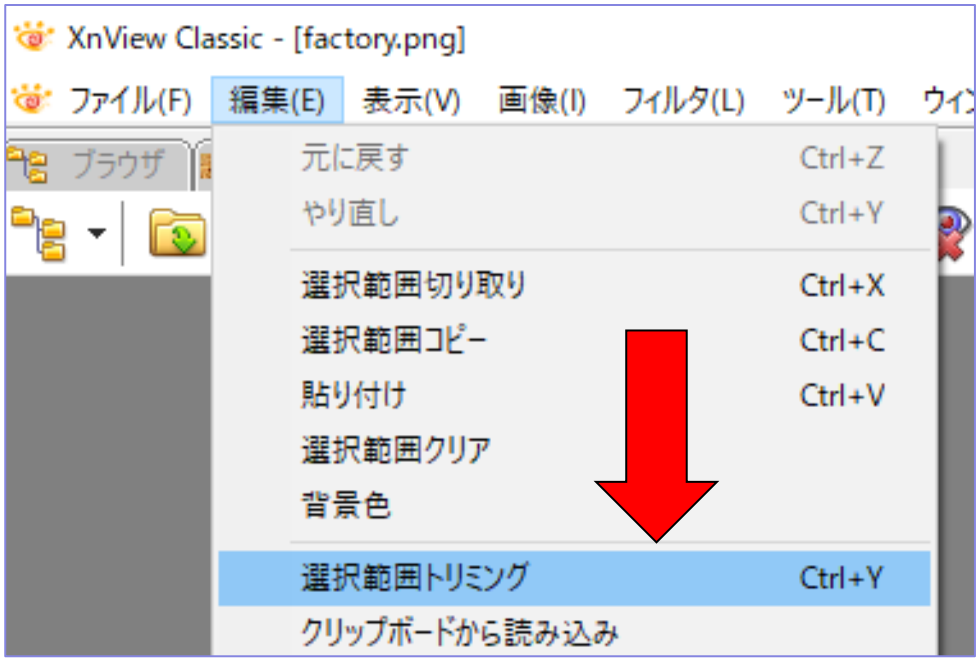
指定サイズでの切り出し(2)

編集 → 選択範囲縦横率設定 → 1:1 (1.00) を選択
これで正方形で範囲選択が可能

指定サイズでの切り出し(3)

任意の範囲を選択後、編集 → 選択範囲トリミング を選択し、切り抜く

切り抜いた後、画像 → リサイズ、でサイズ変更画面が開くので、縦横128ピクセルを指定してサイズ変更し、target.jpg という名前で保存



CSVファイル

こちらで準備済み

- target_or_other_train.csv　：学習（訓練用）ファイルパスとラベル
- target_or_other_test.csv　：検証用ファイルパスとラベル

それぞれのファイル内に、各画像のパスとラベルがカンマ区切りで書かれている

エディタやExcelなどで1度中味をチェックすること

水増しと学習

Colab: 08-Augmentation_Learning.ipynb を開く

VM内にtarget.jpgとother.jpg, csvファイルをコピーし、実行する

先程のcsvファイルと、フォルダtrain_target, test_targetとの対応、実際のファイルの存在をチェックしておく

学習後、重みファイル(detection_weight.h5)をダウンロードし、後は実機で

ネットワークはある程度しか書いていないので、認識できるように、拡張して下さい

重みファイルはネットワークごとに名前を変えておいて、比較するとよい

実機での認識

実機では、Anaconda promptから。前回作成したpython 3.7環境を使用

conda activate py37 など

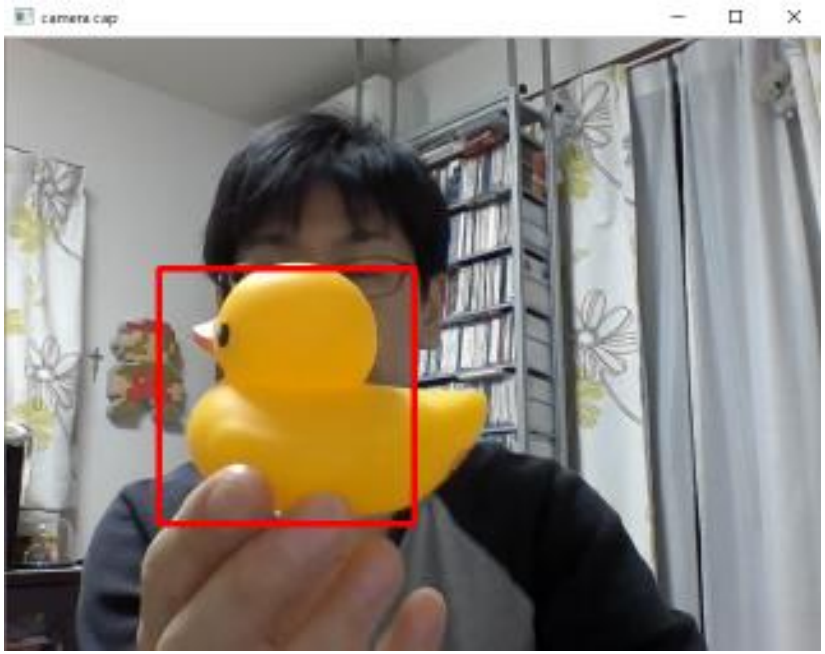
さらに必要なパッケージをまずインストールする

```
pip install opencv-python
pip install pillow
pip install tensorflow
```

その後、`python detection_roi.py` と実行

結果

緑の枠内にターゲットを持っていく
無事に認識されると赤枠になる
どうにもダメなら、ネットワークやエポック数など変えてみる



もし改良するなら

ターゲットの画像サイズをもう少し大きくしてみる (192x192など?)
画像サイズを大きくしたら、検出用ソースも手を入れる
ネットワークの規模を大きくする
そもそも水増し画像をもっと増やしてみる（CSVファイルにも手を入れる）

物体検出

物体検出手法の概要

画像のどの位置に何があるか、を判断するのが物体検出

検出：物体と思われる領域を見つける処理

識別：領域内の物体について識別する処理

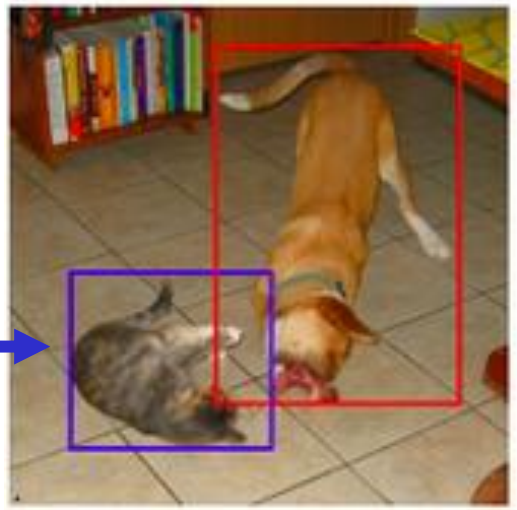
この2つから成り立っている

単なるCNNでは画像全体から判断していたが、物体検出ではいかにして領域を見つけるかが鍵になる

自動運転には欠かせない

画像中からこの枠で囲うのが検出

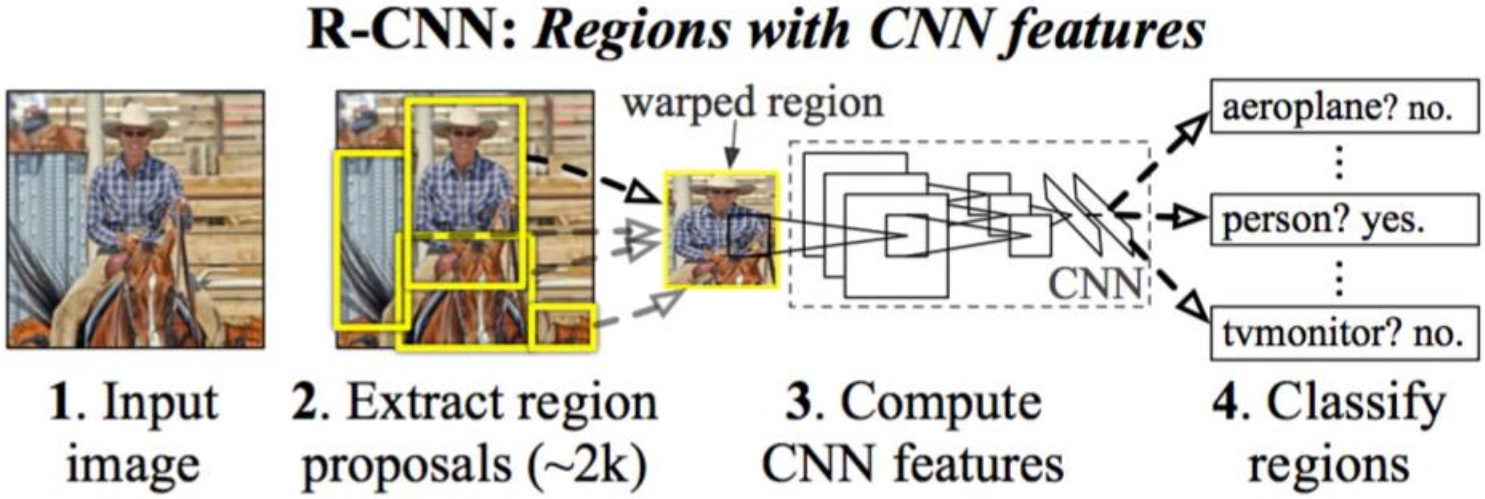
枠の中が猫と出力するのが識別



物体検出手法の概要(R-CNN)

入力画像に対して、物体がある候補領域(region proposal)を2000個まで抽出し、各領域に対して画像をリサイズしてCNNから特徴マップを生成

Region proposalは色や濃淡勾配などから切り分けている



Ross Girshick, Jeff Donahue, et al.,"Rich feature hierarchies for accurate object detection and semantic segmentation", (2013), https://arxiv.org/pdf/1311.2524

SSD: Single Shot Multibox Detector

現在最もよく使用されている物体判別器

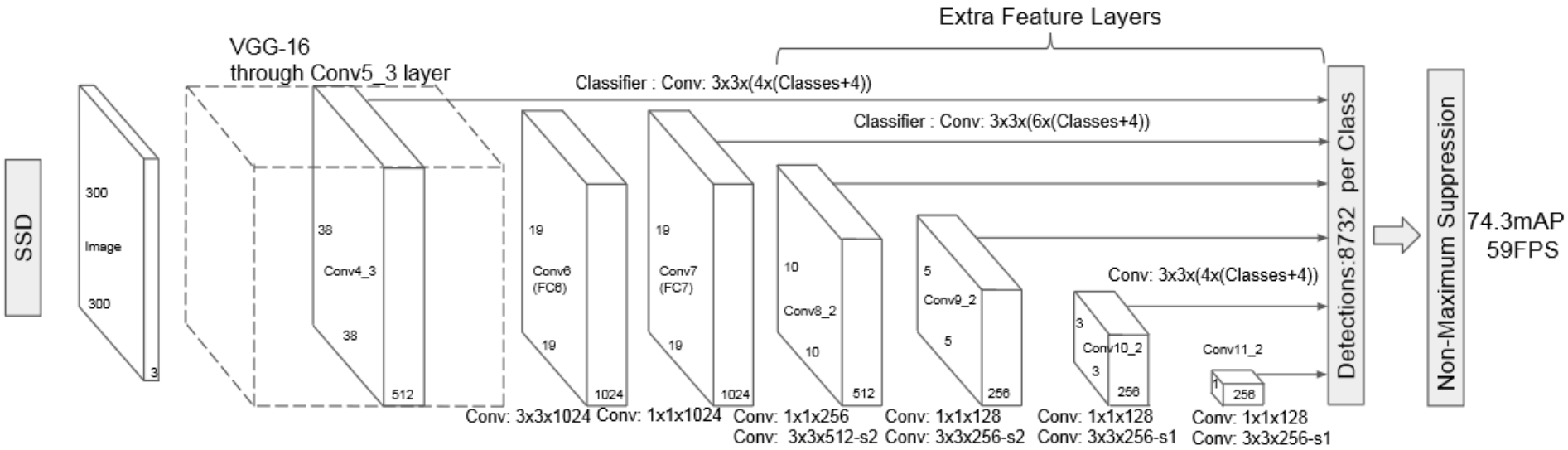
CNNを使用して、画像内のどの位置にどのような物体が存在するかを判別

物体判別器には他にYOLOやFaster R-CNNなどのモデルもあるが、元論文ではSSDの方がより優れた結果を挙げている

Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al., **SSD: Single Shot MultiBox Detector**, https://arxiv.org/abs/1512.02325, (2015)

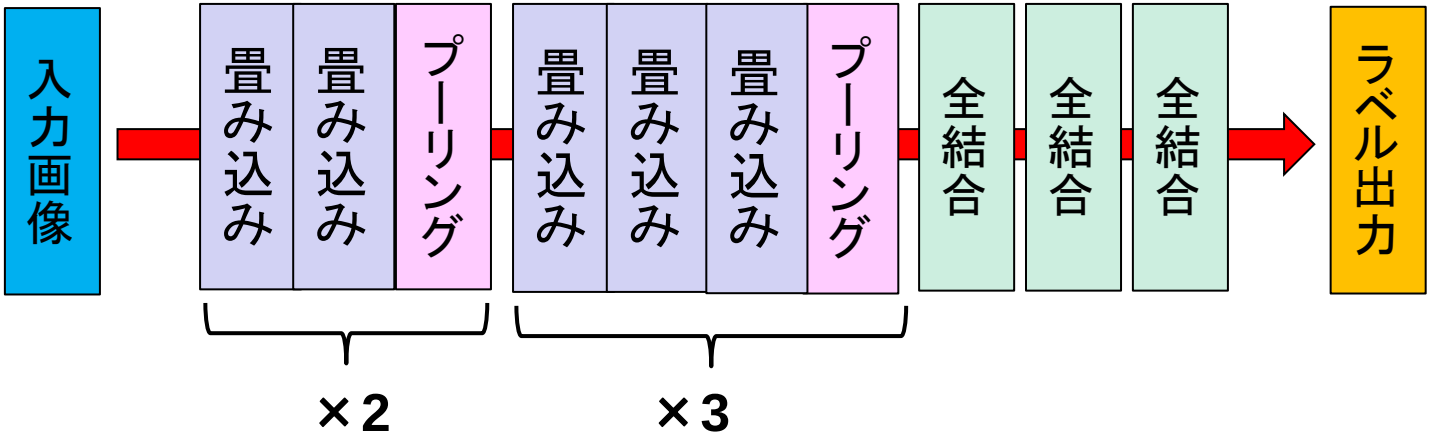
SSDのネットワーク

VGG-16をベースとして、特徴マップを拡張する
ではVGG-16をまず……



VGG-16

ImageNetで学習された16層のモデル。ILSVRC2014で2位を獲得した
224×224のカラー入力画像に対し、以下のネットワークで学習
※プーリング層以外をすべて足すと16層



Karen Simonyan, Andrew Zisserman, “**Very Deep Convolutional Networks for Large-Scale Image Recognition**”, https://arxiv.org/abs/1409.1556, (2014)

VGG-16の特徴

入力画像は224x224、これを112x112, 56x56, … と7x7まで小さくする

最終出力は1000

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
```

VGG-16演習

Colab: 08-VGG16.ipynb を実行

実機：vgg16.py, vgg16_camera.py

実機では、Anaconda promptから。前回作成したpython 3.7環境を使用

conda activate py37 など

実行前に必要なパッケージをインストールする

```
pip install keras
```

※独立パッケージをKerasを使用

VGG-16演習: 実機

vgg16.py: Colabでの演習を実機に移したもの
画像表示を行わない分高速です（意外とColabは画像表示が遅い）

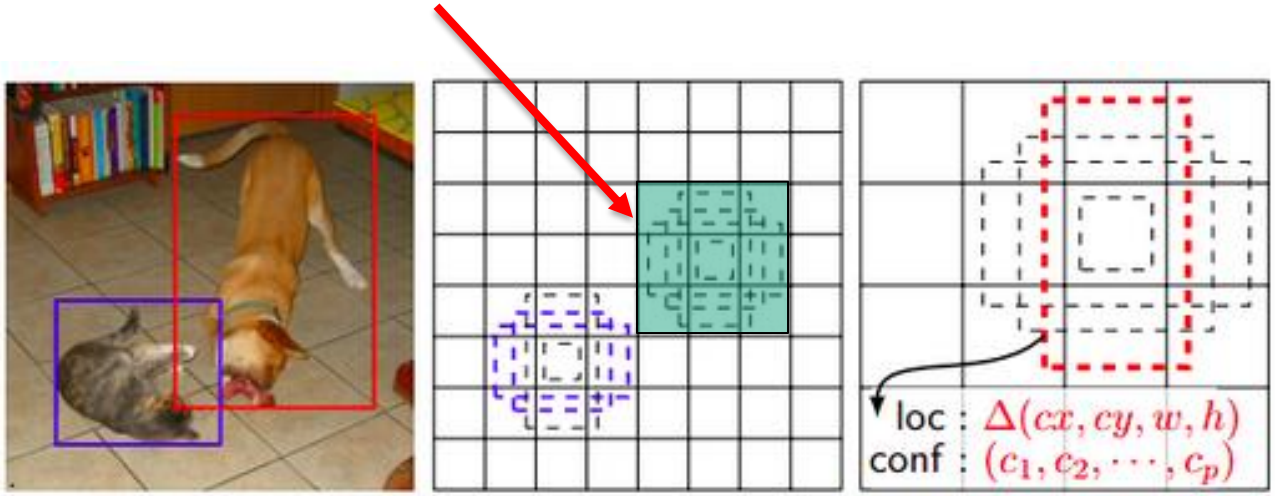
vgg16_camera.py: WebカメラからVGG-16で判別
起動するとWebカメラの映像が映るので、ESCキーを押すと、キャプチャがphoto.jpgで保存され、その内容がVGG-16で判別される
色々な物を持ったり、（最小限の接触で）近くの人に協力してもらって実行して下さい

改めてSSD

図(a)の実線の青い矩形(教師データ)と、特徴マップである図(b)の点線の青い矩形(推定値)の位置・サイズが一致し、また物体クラスが猫となるようにネットワークの重みを更新していく

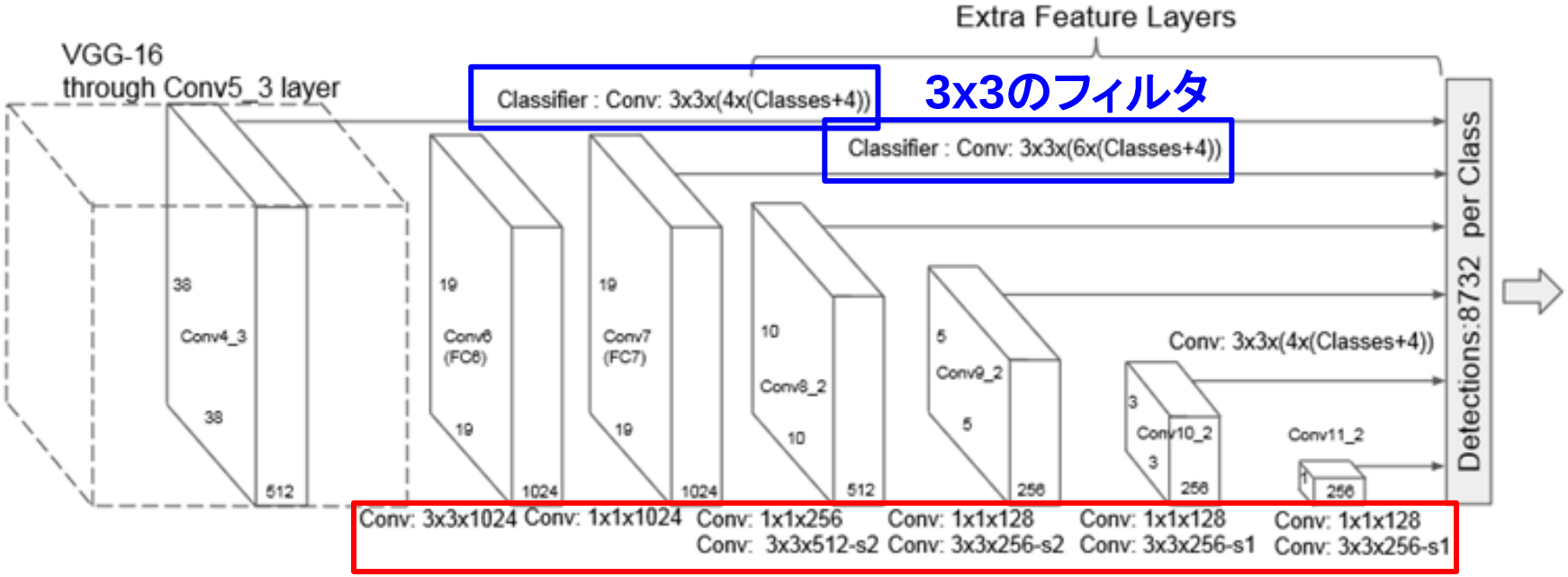
同じように、図(a)の実線の赤い矩形(教師データ)と、図(c)の点線の赤い矩形(推定値)の位置・サイズを一致させ、犬と推定できるように学習を行う

3x3のフィルタをかけるのが特徴



(a) Image with GT boxes (b) 8 × 8 feature map (c) 4 × 4 feature map

ネットワーク構造



VGG-16で38x38x512の特徴マップを作成。

その後、19x19x1024, 10×10×512, 5×5×256, 3×3×256, 1×1×256, と特徴マップを作成して入力としている

最終的に、1つの入力画像に対して分類クラスあたり8732個の物体領域候補が出力される

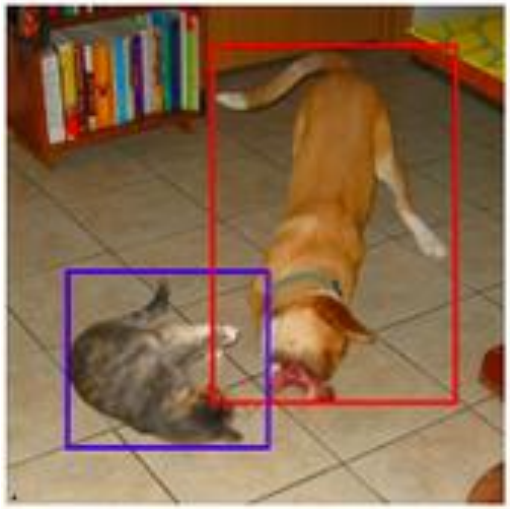
デフォルトボックスの対応

物体のアスペクト比は当然物体ごとに異なる

各畳み込み層で4種類のアスペクト比、4種類の位置をパラメータを持ち、すなわち4×4=16通り、さらに最初のVGG-16からの出力なら画素は38x38なので、38x38x16 = 23104の位置特徴量を持つ

（各レイヤごとにパラメータは若干異なる）

これによって、猫と犬のアスペクト比の違いを学習する



SSD演習(1)

Google ColabでSSDを試行
あらかじめ構築されたモデルを使用するので学習は必要ない
モデルは2つ用意されており、

- **FasterRCNN+InceptionResNet V2**: high accuracy,
- **ssd+mobilenet V2**: small and fast.

の記述通り、前者がより正確、後者はモデルがコンパクトで速い。両者での結果を比較してみるとよい（実際かなり異なる）

SSD演習(2)

ネットから適当に探す、あるいは自分で撮影、手持ちの画像をSSDにかけてください

その結果をTeamsにアップロードし、比較してみましょう

どのような画像がうまくいくか、あるいはうまくいかないか

使用できるネットワーク

今回、VGG-16やInceptionResNet, MobileNetなどを使用したが、KerasやTensorflowではその他にも様々なネットワークが使用できる

詳細はGitHubを参照

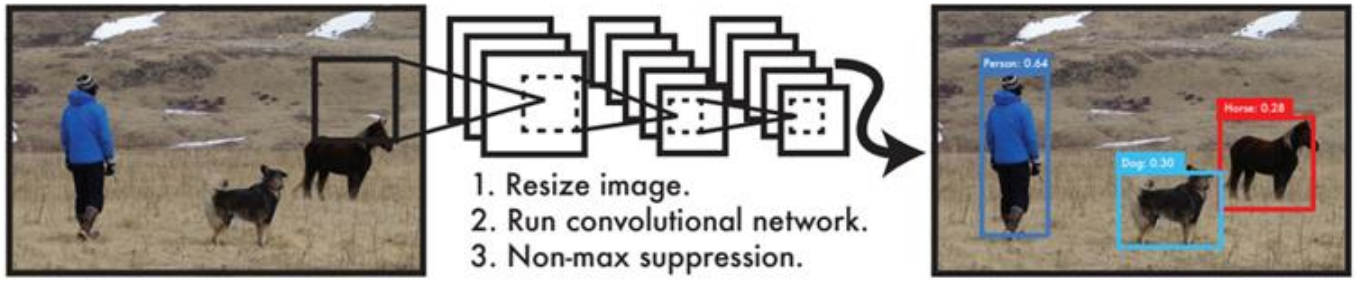
https://github.com/keras-team/keras-applications

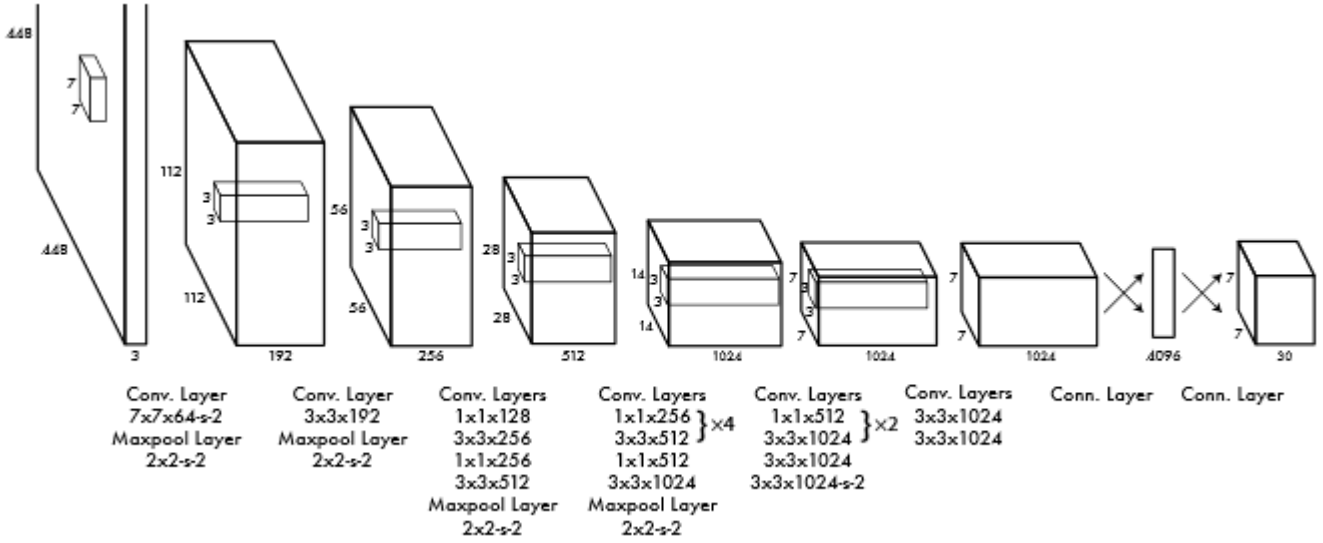
他に有名なネットワークとしてYOLOv3など

https://pjreddie.com/darknet/yolo/

YOLOの手法(1)

画像を正方形(448x448など)にリサイズし、畳み込みへの入力とする



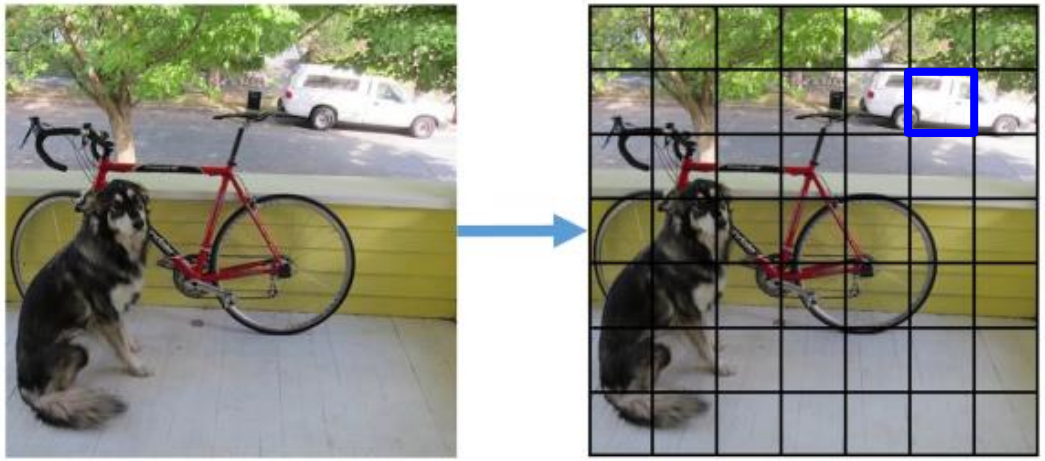


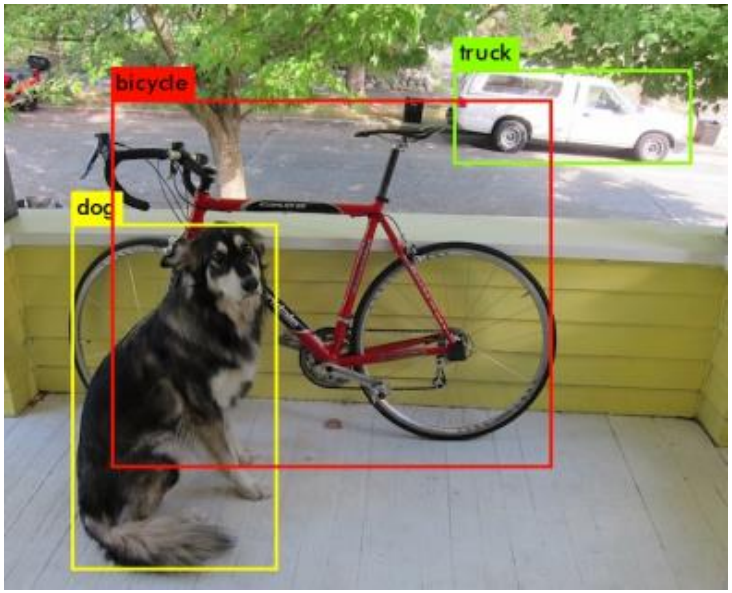
24層の畳み込み,
2層の全結合

Paper: https://pjreddie.com/media/files/papers/yolo_1.pdf

YOLOの手法(2)

画像全体をSxSに分割（論文ではS=7）
分割された1つのセルをB個に分ける（同じくB=2）
それぞれのセル内にてクラス分類を行い、信頼度の高いものだけを採用
それぞれの信頼度の高い部分を太く囲う
1つのセル内に複数の物体があると弱い





第１１章

生成モデル

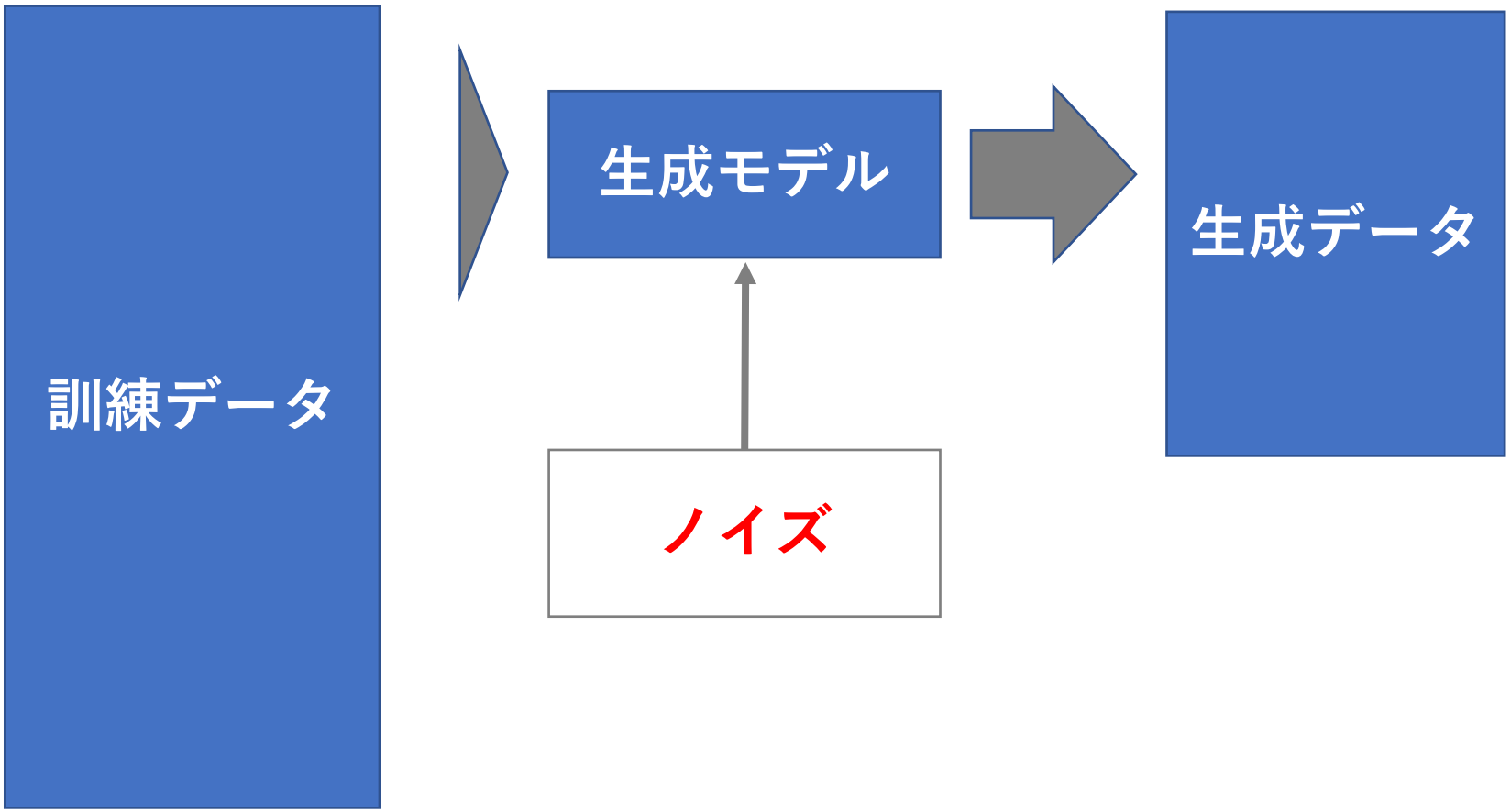
PGGANを参照

- Progressive Growing of GANs for Improved Quality, Stability, and Variation
- https://www.youtube.com/watch?v=G06dEcZ-QTg

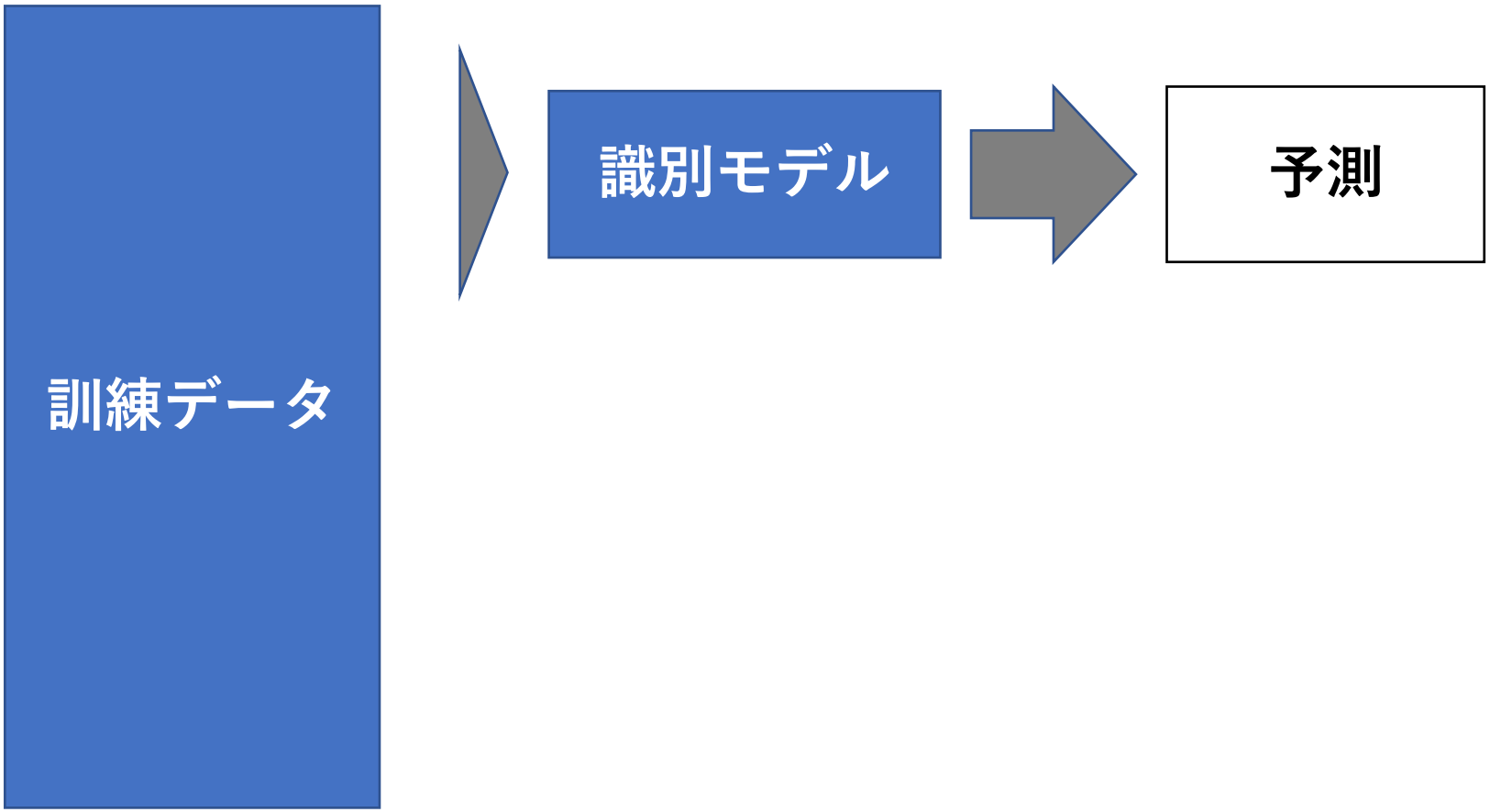
PGGAN

- Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen; Progressive Growing of GANs for Improved Quality, Stability, and Variation
- 著者はNVIDIAの人たち
- YouTubeの冒頭の顔の写真はCelebAというデータセットで訓練した結果を元に生成されたもの

生成モデルとは



一方、識別モデルは…



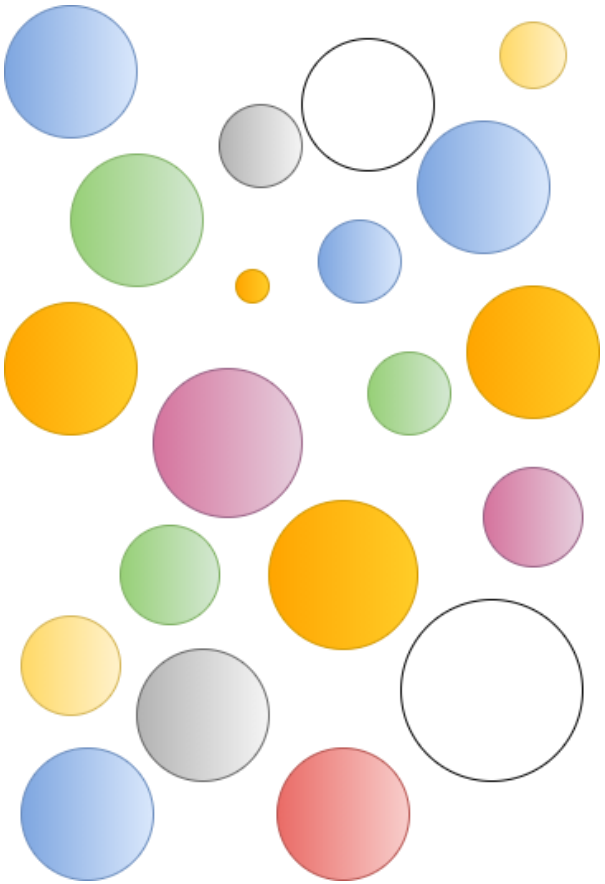
識別モデリングと生成モデリング

- **識別モデリング**
  - 観測 $\boldsymbol{x}$ が与えられたときのラベル $\boldsymbol{y}$ の得られる確率を推定

- **生成モデリング**
  - 観測 $\boldsymbol{x}$ の分布（確率分布）を推定する

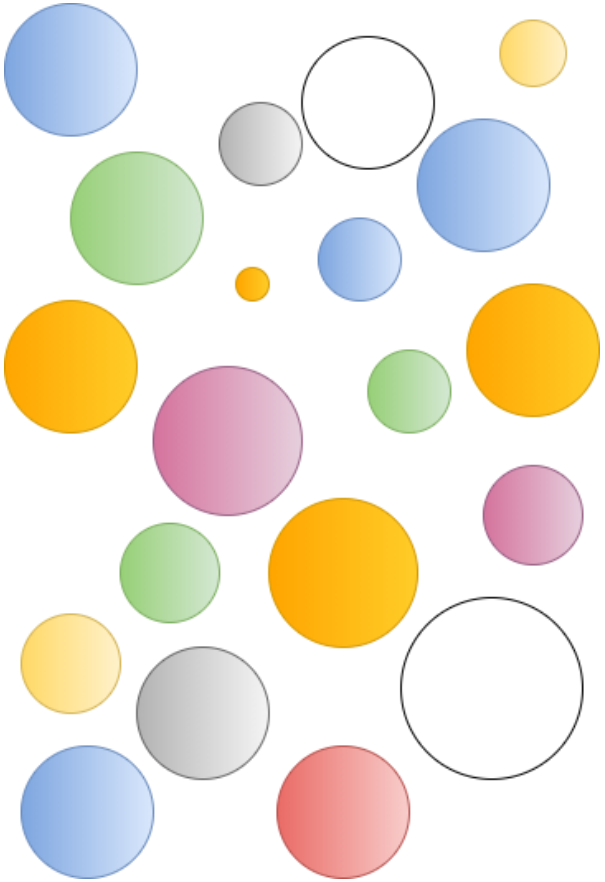
生成モデリングの潜在空間と生成データの関係

- 右の画像の一つ一つの丸は nxn 配列として表せる
- しかし、nxnの空間内で丸を表すものはごくわずか
- nxnの空間内を適当に取ってきても、それが丸を表すことは期待できない

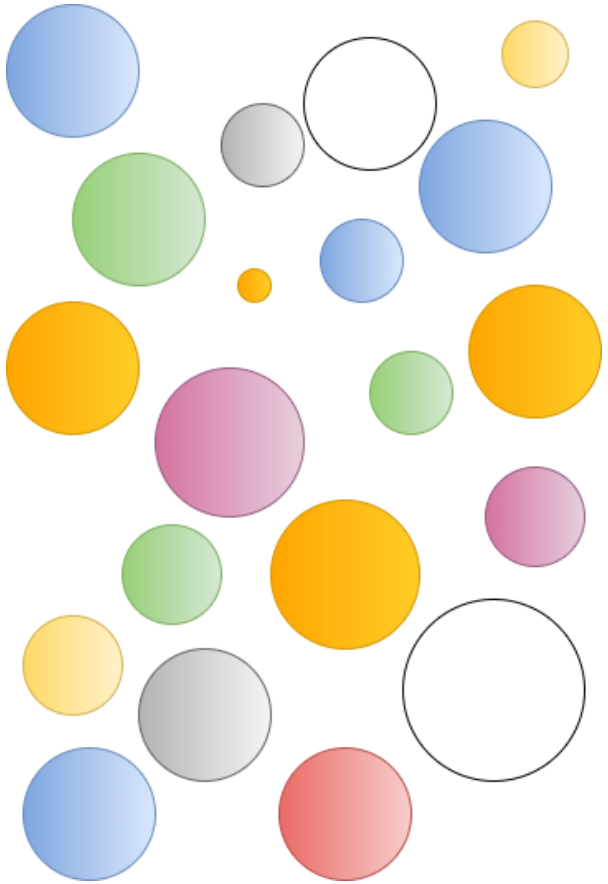


生成モデリングの潜在空間と生成データの関係

- しかし、ひとつひとつの丸は2つのパラメタで表せることは、我々人間にはすぐにわかる
- （半径、色）
- 半径と色を変形すると、ここにない画像も生成できる



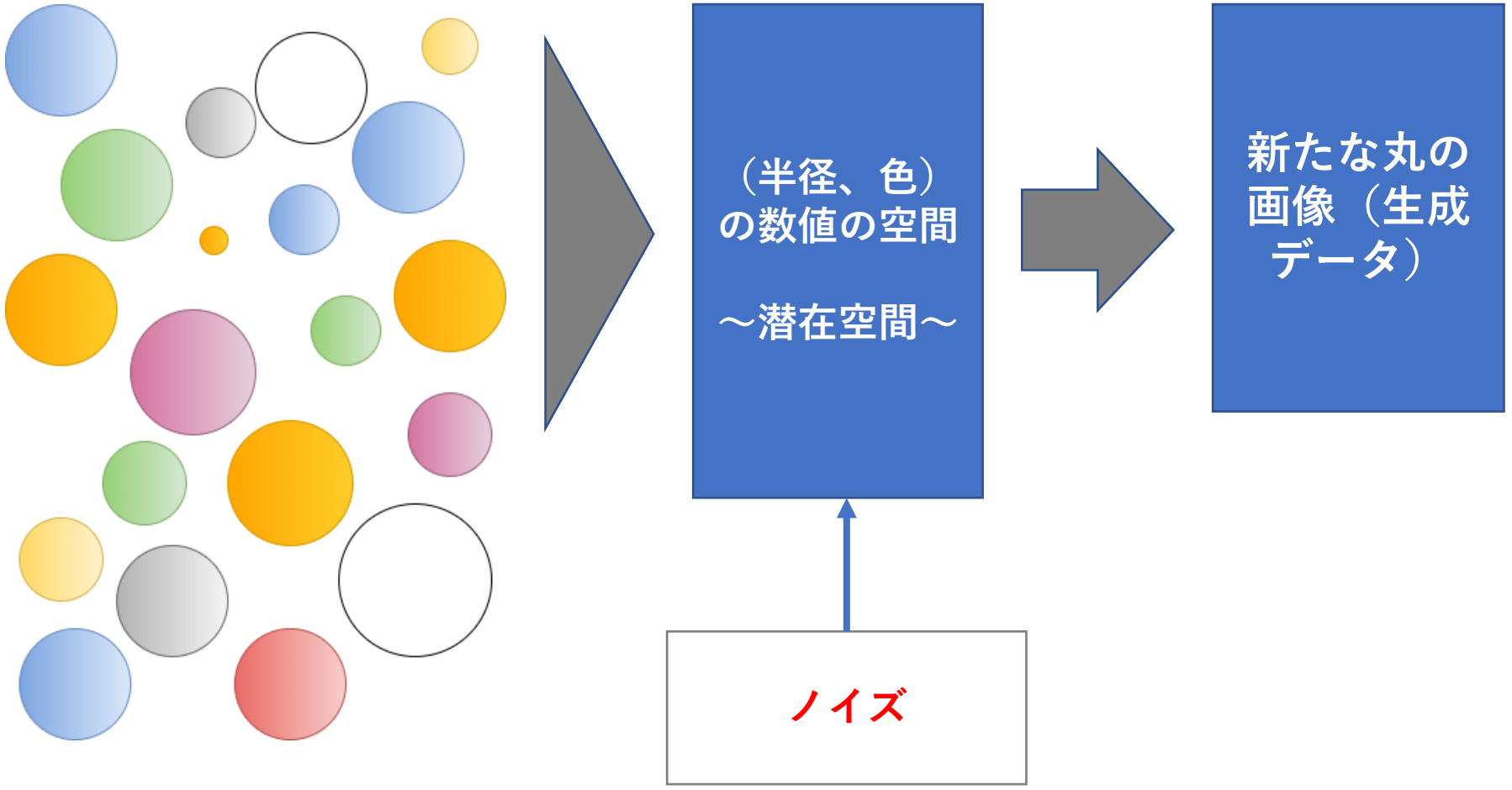
生成モデリングの潜在空間と生成データの関係



（半径、色）
の数値の空間

～潜在空間～

生成モデリングの潜在空間と生成データの関係



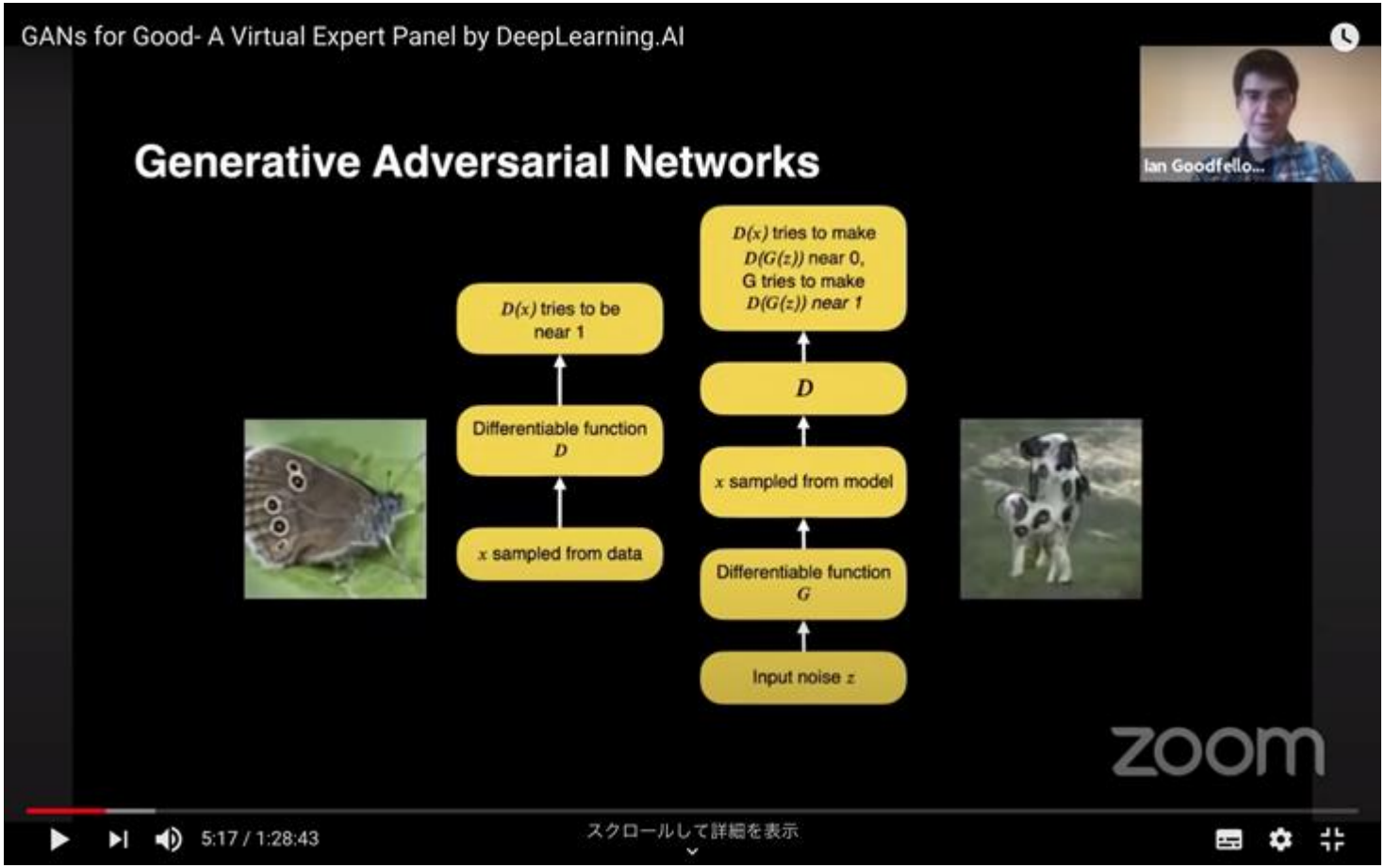
デモ（PGGAN)

- PGGAN自体の詳しい説明はせず、潜在空間と生成画像の関係について見てみる
- 訓練には膨大な計算機資源と時間が必要
- デモには訓練済みのモデルを使用（TensorFlow Hub）

GANs for Good- A Virtual Expert Panel by DeepLearning.AI

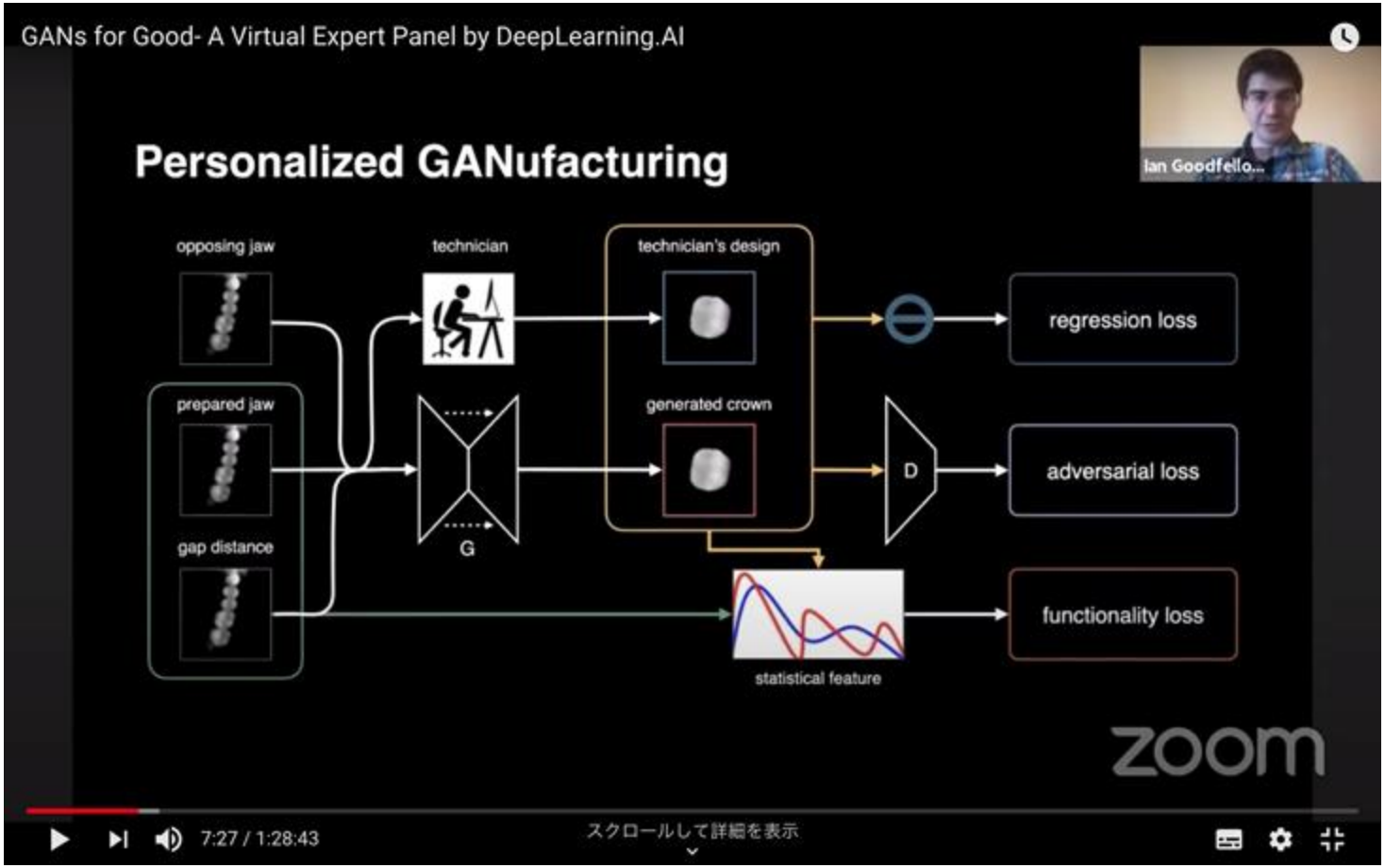
- 先月（9/30）Zoom上で開催（OpenAI主催）
- https://www.youtube.com/watch?v=9d4jmPmTWmc
- GANの産みの親Ian Goodfellowが基調講演
- 以下でGoodfellowのこの講演の概要を参照

GANs for Good- A Virtual Expert Panel by DeepLearning.AI

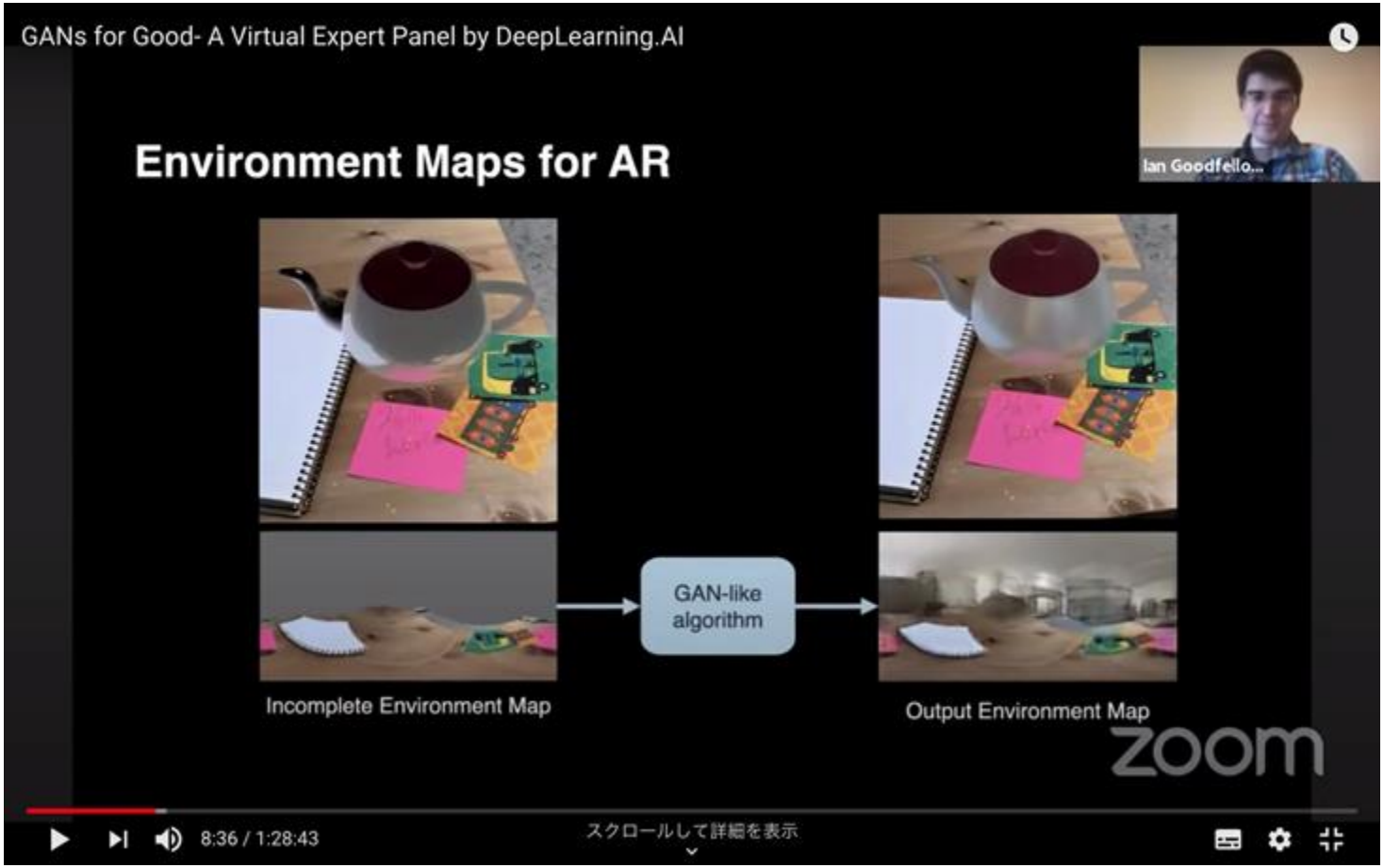


GANs for Good- A Virtual Expert Panel by DeepLearning.AI

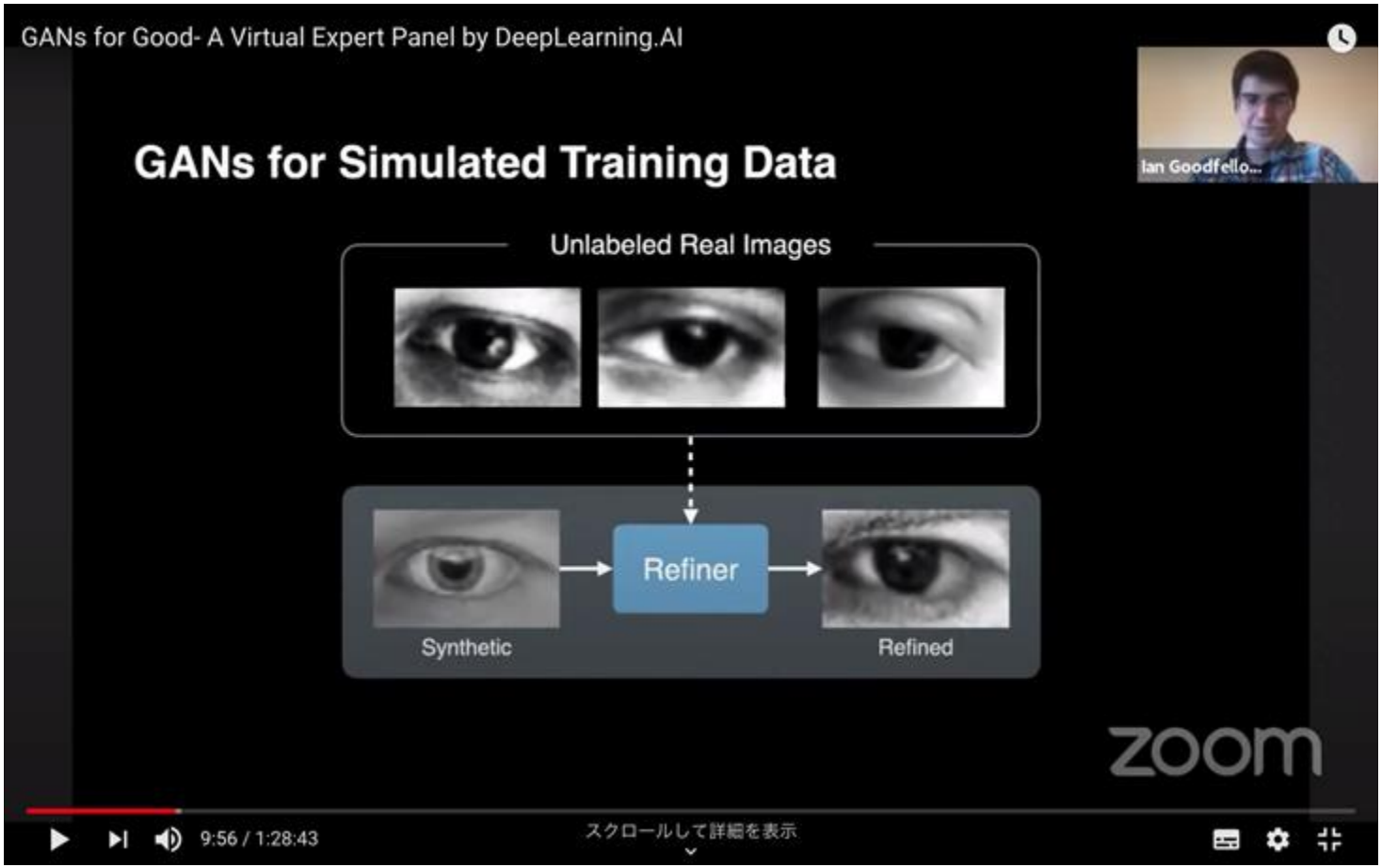
GANs for Good- A Virtual Expert Panel by DeepLearning.AI



GANs for Good- A Virtual Expert Panel by DeepLearning.AI

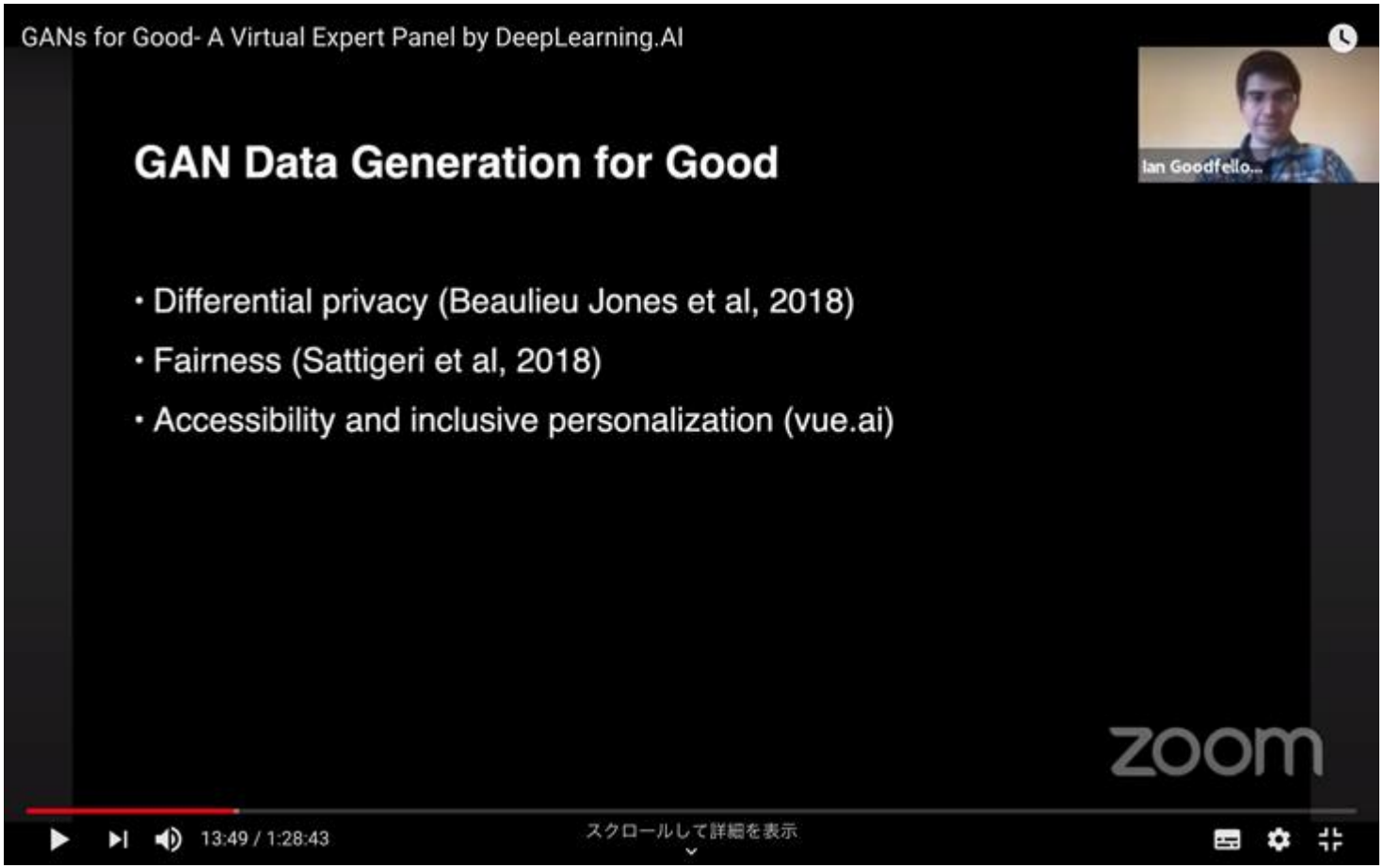


GANs for Good- A Virtual Expert Panel by DeepLearning.AI

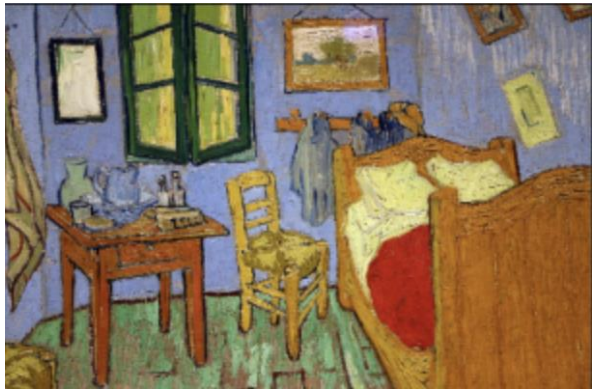


GANs for Good- A Virtual Expert Panel by DeepLearning.AI

GANs for Good- A Virtual Expert Panel by DeepLearning.AI



Neural Style Transfer



Neural Style Transfer

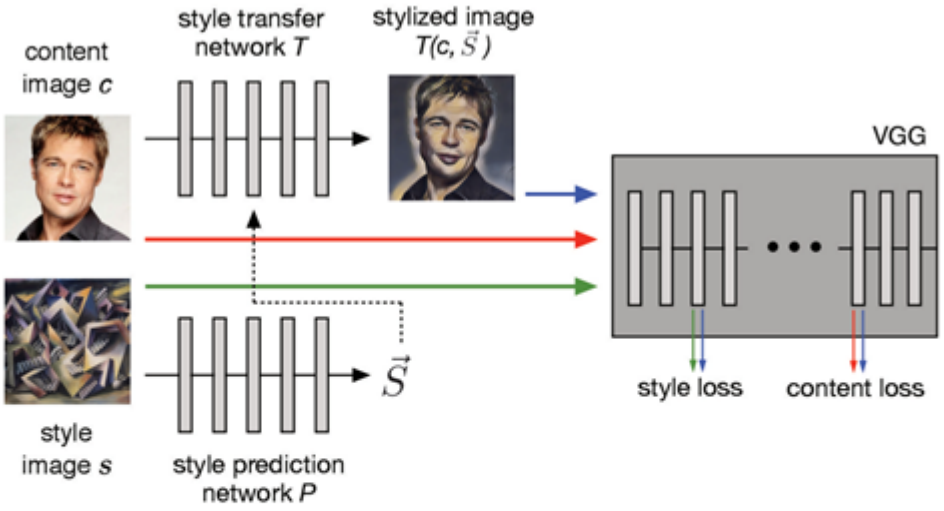


Figure 2: Diagram of model architecture. The style prediction network $P(\cdot)$ predicts an embedding vector $\vec{S}$ from an input style image, which supplies a set of normalization constants for the style transfer network. The style transfer network transforms the photograph into a stylized representation. The content and style losses [9] are derived from the distance in representational space of the VGG image classification network [23]. The style transfer network largely follows [3] and the style prediction network largely follows the Inception-v3 architecture [24].

Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, Jonathon Shlens; Exploring the structure of a real-time, arbitrary neural artistic stylization network, https://arxiv.org/abs/1705.06830

Neural Style Transferのデモ

- Colab上でデモを行います
- 画像は自由に選択できるので、色々と試行
- Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, Jonathon Shlens; Exploring the structure of a real-time, arbitrary neural artistic stylization network
- 初見のスタイルにも対応できる高速変換可能なネットワーク

第１２章

再帰型ネットワーク

Simple RNN, LSTM, GRU, ...

再帰型ニューラルネットワーク

まずは用語の表面的なことについて。

- Recurrent Neural Network (RNN)の例
  - Simple RNN
  - Long Short-Time Network (LSTM、1997年提案)
  - Gated Recurrent Unit (GRU、2014年提案)
- 文脈によってはRNNがSimple RNNだけを指すこともある
- Recurrentは回帰型とも

再帰型ニューラルネットワーク

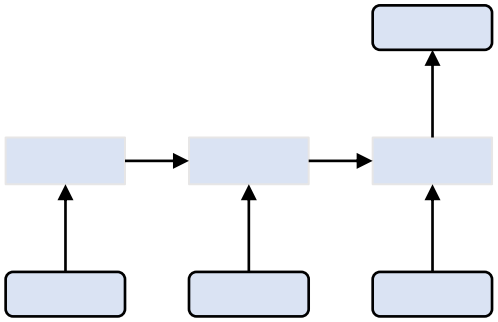
- 適用対象となるデータは系列データ
  - 自然言語など
- 用途は、主に系列データの分類タスク。生成も
  - 文章の感情分析（印象がよいかわるいかなど）
  - 文章生成
  - （翻訳）

デモ

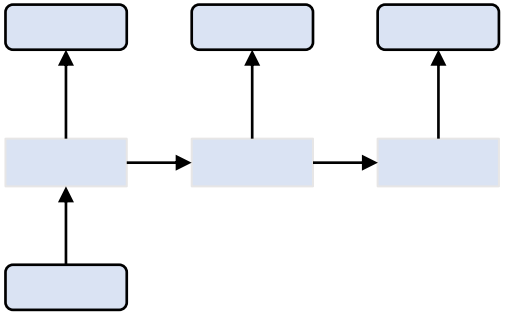
- RNNを利用して、レビュー文の感情分析を行う
- まずは、何ができるのかを見てもらいます

系列データ

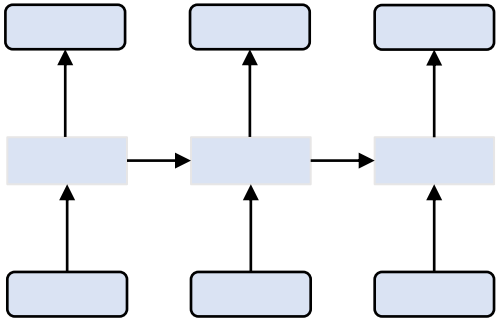
系列データのモデル4分類



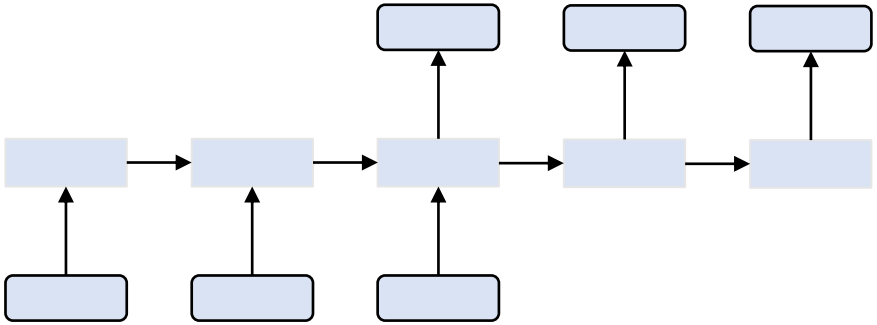
多対一



一対多



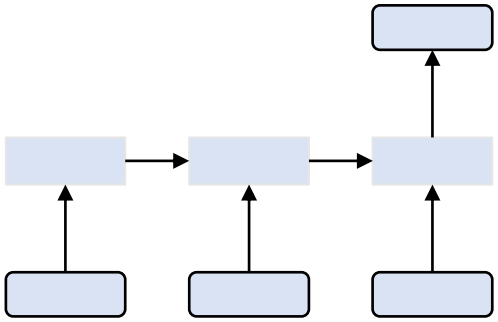
多対多（同期）



多対多（遅延）

cf. Andrej Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks,* http://karpathy.github.io/2015/05/21/rnn-effectiveness/

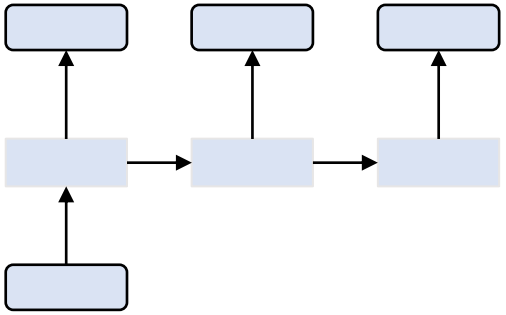
系列データのモデル4分類



多対一

例

- テキストの感情分析
- 時系列データの異常判断
- その他



一対多

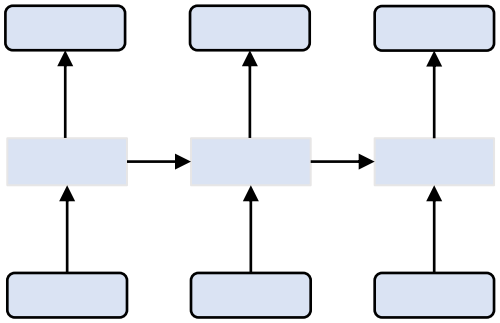
例

- 画像キャプショニング
- その他

系列データのモデル4分類

例

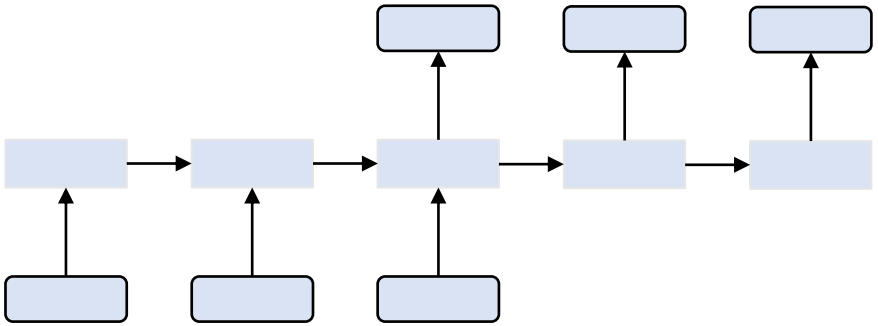
- 動画分類（フレームごと）
- その他



多対多（同期）

例

- 自然言語の翻訳
- その他



多対多（遅延）

系列データの表し方

一般形

- $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$

例

- 何らかの測定値（T = 6 の例）
  - $(1.5, 2.3, 0.2, 5.2, 4.9, 3.1)$
- 自然言語（T = 4 の例）
  - “I have a pen.”

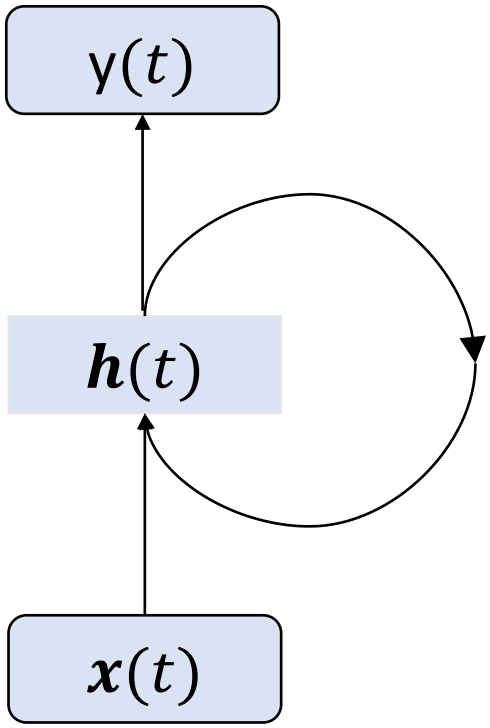
→ (“I”, “have”, “a”, “pen”)

→ (2243, 1098, 1, 5031)

→ $(e_{2243}, e_{1098}, e_1, e_{5031})$　（e_i は第i成分のみが1で他は0のベクトル）

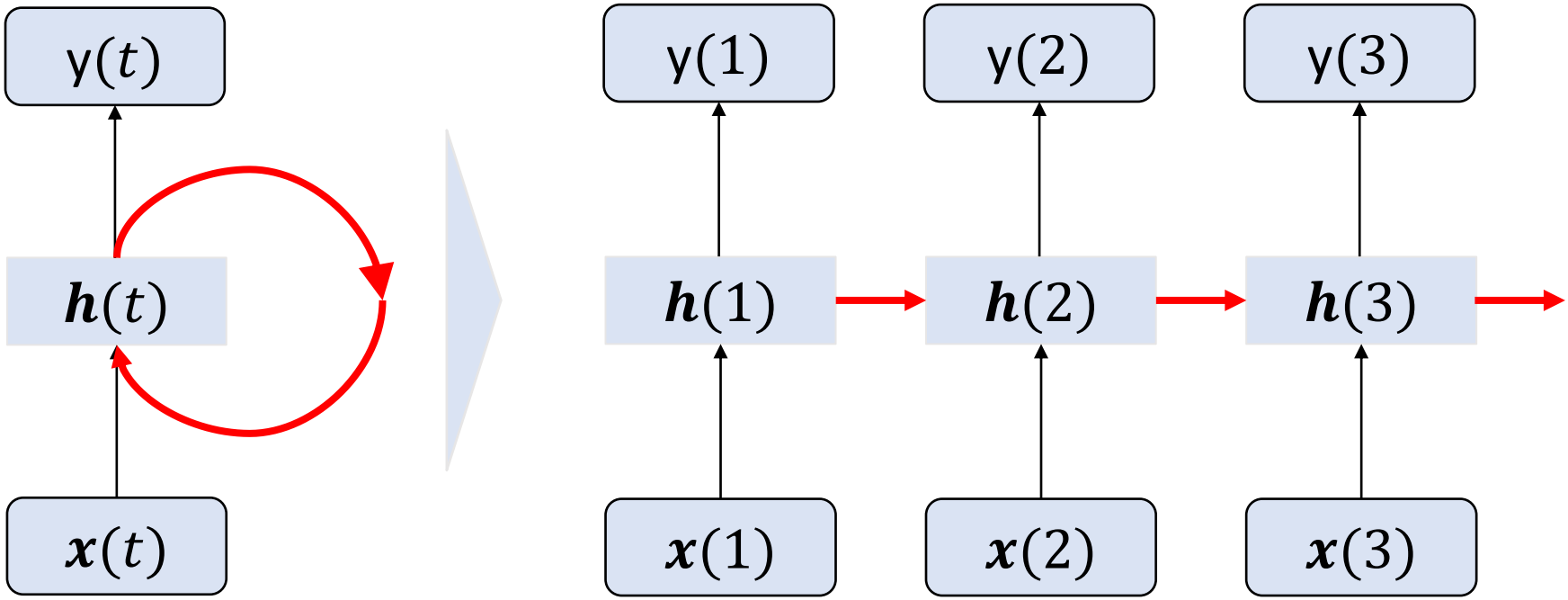
単純RNN

単純RNNの構造



- 各時刻のデータが順に入力される（パラメータtを時刻と呼ぶことにする）
- 多対多、多対一モデルのモデル
- ループを持つ
- ループ部分のデータは一時刻遅れて入力される

再帰モデルの展開表現



単純RNNのトレーニング

BPTT(BackPropergation Throught Time)

- RNNを展開したネットワークにおいて、誤差逆伝播法により最適化する
- t 方向に深いネットワークの場合（つまり T が大きい場合）、誤差逆伝播の際、t について一定区間ごとに中間層の結びつきを切断したものとして、重みの更新を行う（Truncated BPTT）。

単純RNNの性質、問題点

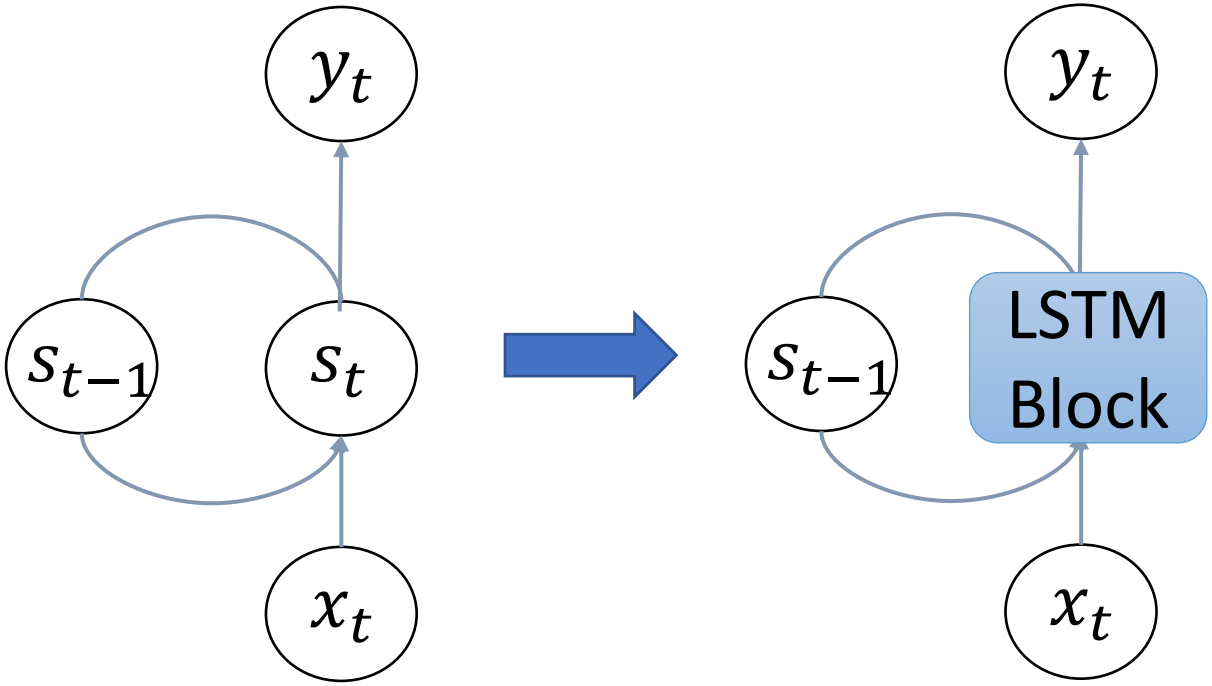
- 時間方向にdeepなネットワークであり、勾配爆発、勾配消失問題が生じる

- 勾配爆発
  - 勾配クリッピングにより対処
- 勾配消失
  - せいぜい20ステップ程度の時刻しか学習できない
  - モデルを改良する必要あり → LSTMへ発展

LSTM

LSTM

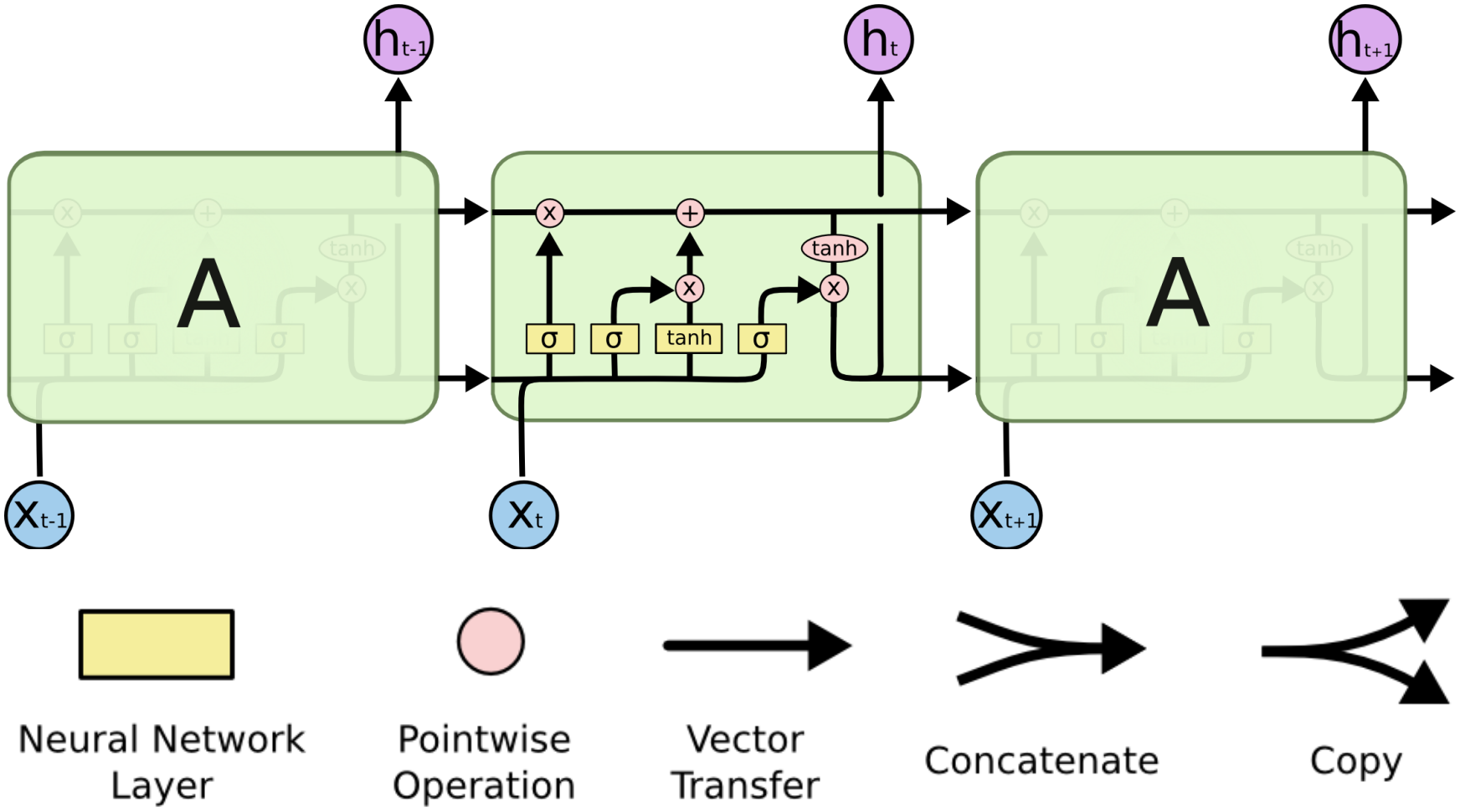
- RNNの拡張型、短期記憶(short-term memory)を長期(Long)に渡って活用するニューラルネットワーク
- RNNの中間層のユニットをLSTM blockと呼ばれるメモリと3つのゲートを持つブロックに置き換えたもの



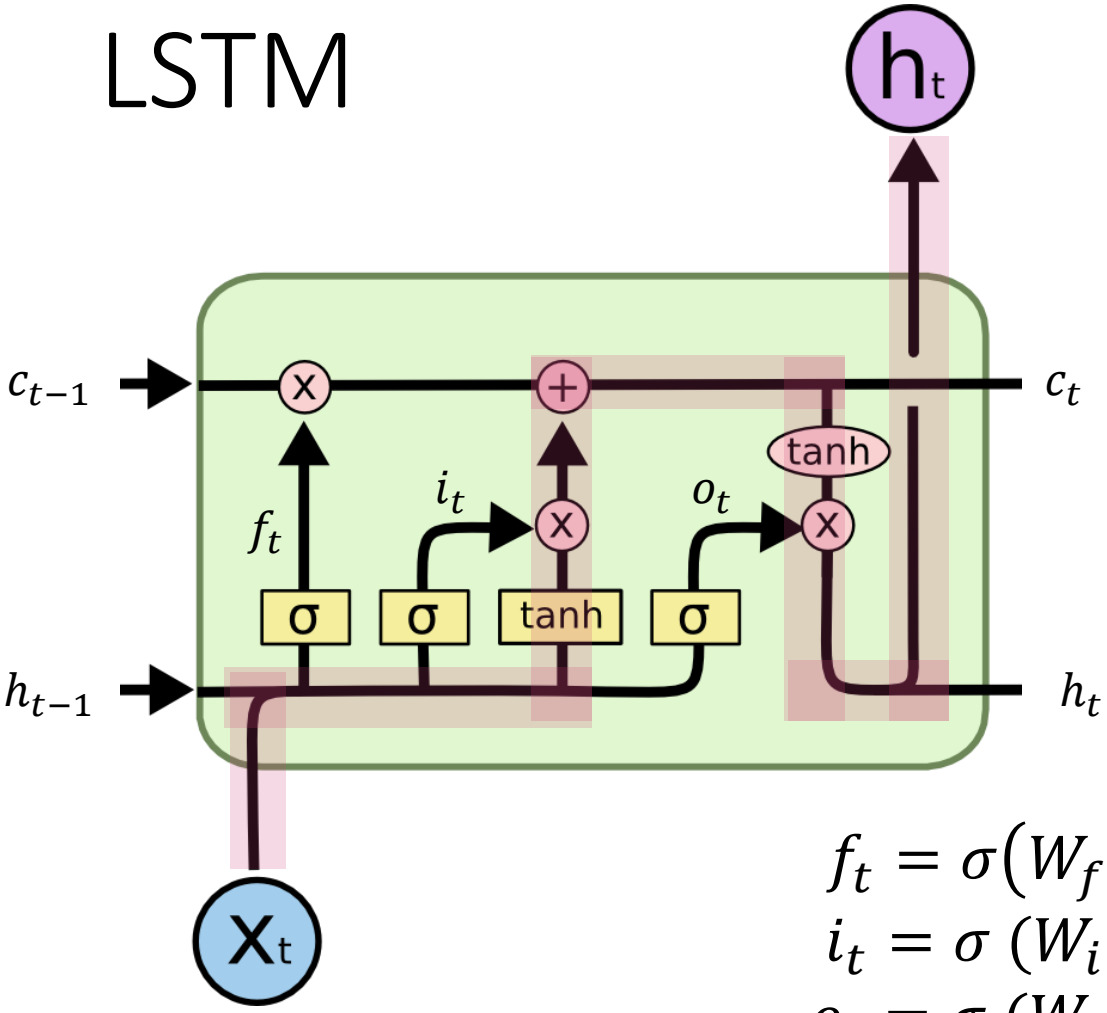
LSTM

- LSTM = Long Short-Term Memory network（長・短期記憶ネットワーク）
- RNN というのはループ構造を持つネットワークの総称であり、LSTM も RNN の一種であるが、RNN と言ったとき前節の単純RNNを指すこともあるので注意
- 単純RNNの中間層をLSTM層と呼ばれる層で置き換え
- ゲート構造、メモリセルの導入
- 勾配消失問題に対応
- Hochreiter and Schmidhuber, 1997
- Gers et al., 2000

LSTM


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM



f_t: 忘却ゲートを制御

i_t: 入力ゲートを制御

o_t: 出力ゲートを制御

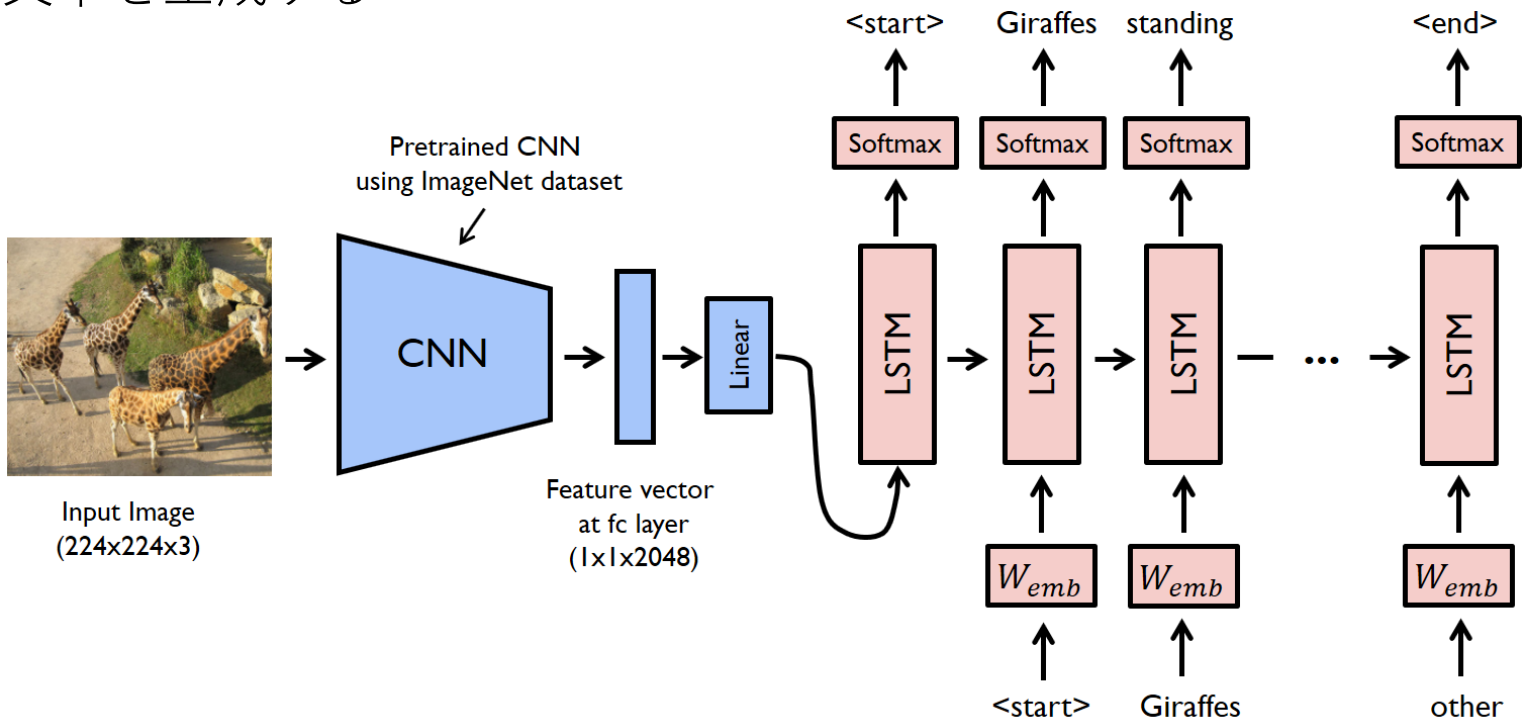
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_\sigma \cdot [h_{t-1}, x_t] + b_o\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(W_c \cdot [x_t, h_{t-1}] + b_c\right)$$
$$h_t = o_t \circ \tanh\left(c_t\right)$$

LSTMの有効利用例

- 制約なし（枠なし）手書き文字認識
  - Graves et al., 2009
- 音声認識
  - Graves et al., 2013; Graves and Jaitly, 2014
- 手書き文字生成
  - Graves, 2013
- 機械翻訳
  - Sutskever et al., 2014
- 画像キャプショニング
  - Kiros et al., 2014; Vinyals et al., 2014; Xu et al., 2015
- 文章の構文解析
  - Vinyals et al., 2014

LSTMの応用：複数モデルの組み合わせ

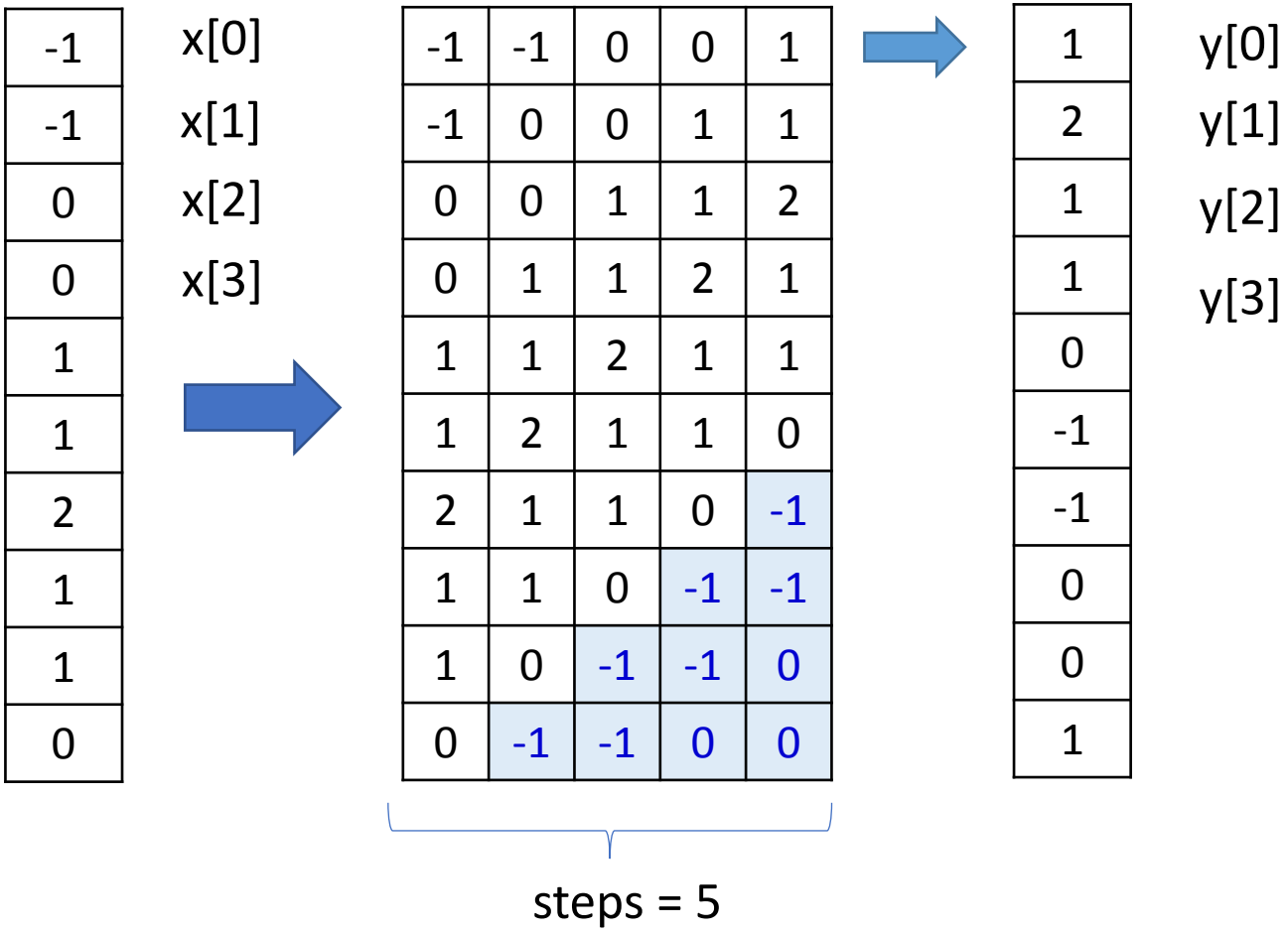
- 入力画像の説明文を生成するニューラルネットワーク
- CNNによって画像から抽出された特徴ベクトルをLSTMに入力し、文章を生成する



図はPytorch Githubより

説明変数と目的変数

- 元の時系列データを、説明変数と目的変数に変換



デモおよび演習

- RNNによるテキスト生成
- LSTMまたはGRU利用

第13章

ニューラルネットワークによる自然言語処理 –前半–

前回の補足：TensorFlow Hub

TensorFlow Hub

https://www.tensorflow.org/hub?hl=ja

TensorFlow Hub は、すぐに微調整してどこにでもデプロイ可能なトレーニング済み機械学習モデルのリポジトリです。BERT や Faster R-CNN などのトレーニング済みモデルを、わずか数行のコードで再利用できます。(Webサイトより)

実際のモデル一覧

https://tfhub.dev/

Hubはモデル一覧だけなので、チュートリアルがわかりやすい

https://www.tensorflow.org/tutorials?hl=ja

前半・後半でやること

前半：
Deep Learning以前の自然言語処理
自然言語をコンピュータ処理するためのいろいろな「お約束」
クラウド上でできること（自分でやるより遙かに賢い）
簡易コーパスの作成

後半：
Deep Learningを利用した自然言語処理
word2vecの利用・作成
BERT

自然言語処理(Natural Language Processing)

自然言語を対象とした情報処理

- 形態素解析
- 構文解析
- 意味解析
- 文脈解析
- 自動要約
- 自動翻訳
- 固有表現抽出

従来は主に辞書や文章（コーパス）を収集し、人手で意味づけを行い、それを元に統計的手法で行うことが多かった

NLPのこれまでと現在

形態素解析エンジンChaSenの開発で品詞分解（分かち書き）が容易に

ベクトル空間、SVMなどによって分類

Syntacticではなく、Heuristicな手法が試みられていた

近年はword2vecによるWikipediaコーパスやGoogle BERTによって飛躍的に精度が向上。BERTの登場後、XLNet, RoBERTa, ALBERTなどBERTを上回るものも発表された

Google Cloud Natural Language API

Google Cloud Natural Language API
https://cloud.google.com/natural-language

事前トレーニング済みモデルにより、感情分析、エンティティ分析、エンティティ感情分析、コンテンツ分類、構文分析などの自然言語理解の機能を使用可能

さらにカスタマイズする場合は、AutoML Natural Languageを使用する

実際にやってみると

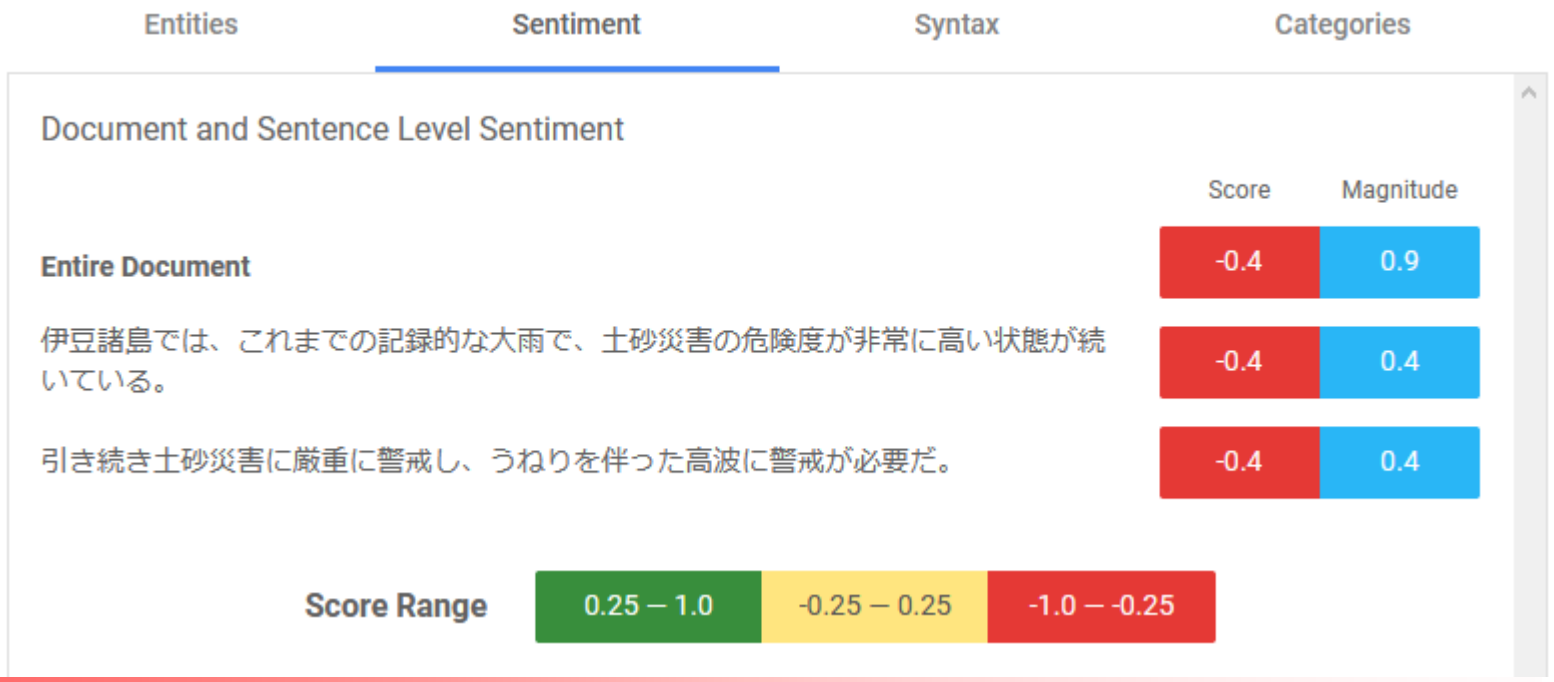
伊豆諸島では、これまでの記録的な大雨で、土砂災害の危険度が非常に高い状態が続いている。引き続き土砂災害に厳重に警戒し、うねりを伴った高波に警戒が必要だ。

エンティティ（実在性）分析

感情分析(Sentiment)

Score: -1から1まで。-1がネガティブ、1がポジティブ

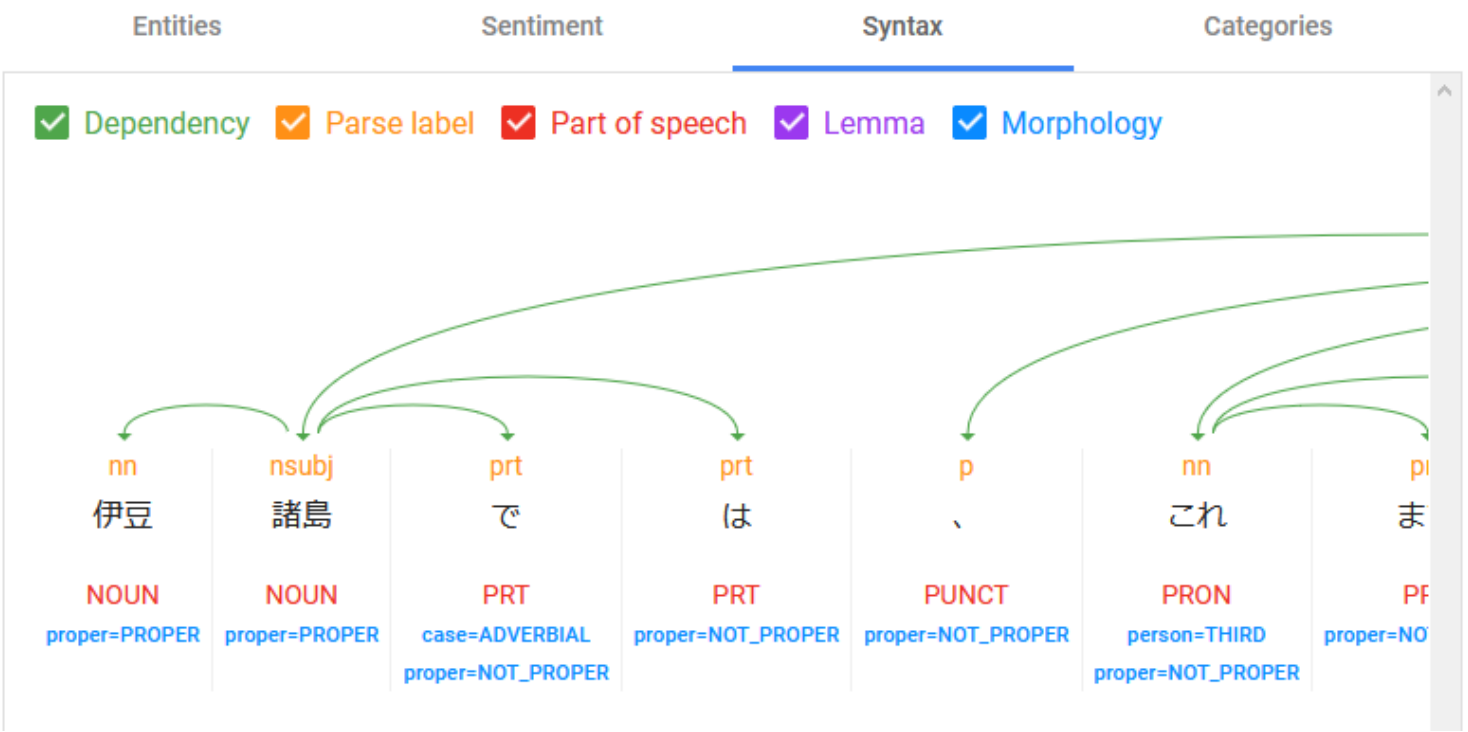
Magnitude：指定したテキストの感情の強度（ポジティブ、ネガティブの両方）が正の値で示される。Scoreとは異なり、Magnitude は正規化されていないため、テキスト内で感情が表現されるたびにMagnitudeが増加される。（長い文章ほど高くなる傾向）



構文解析

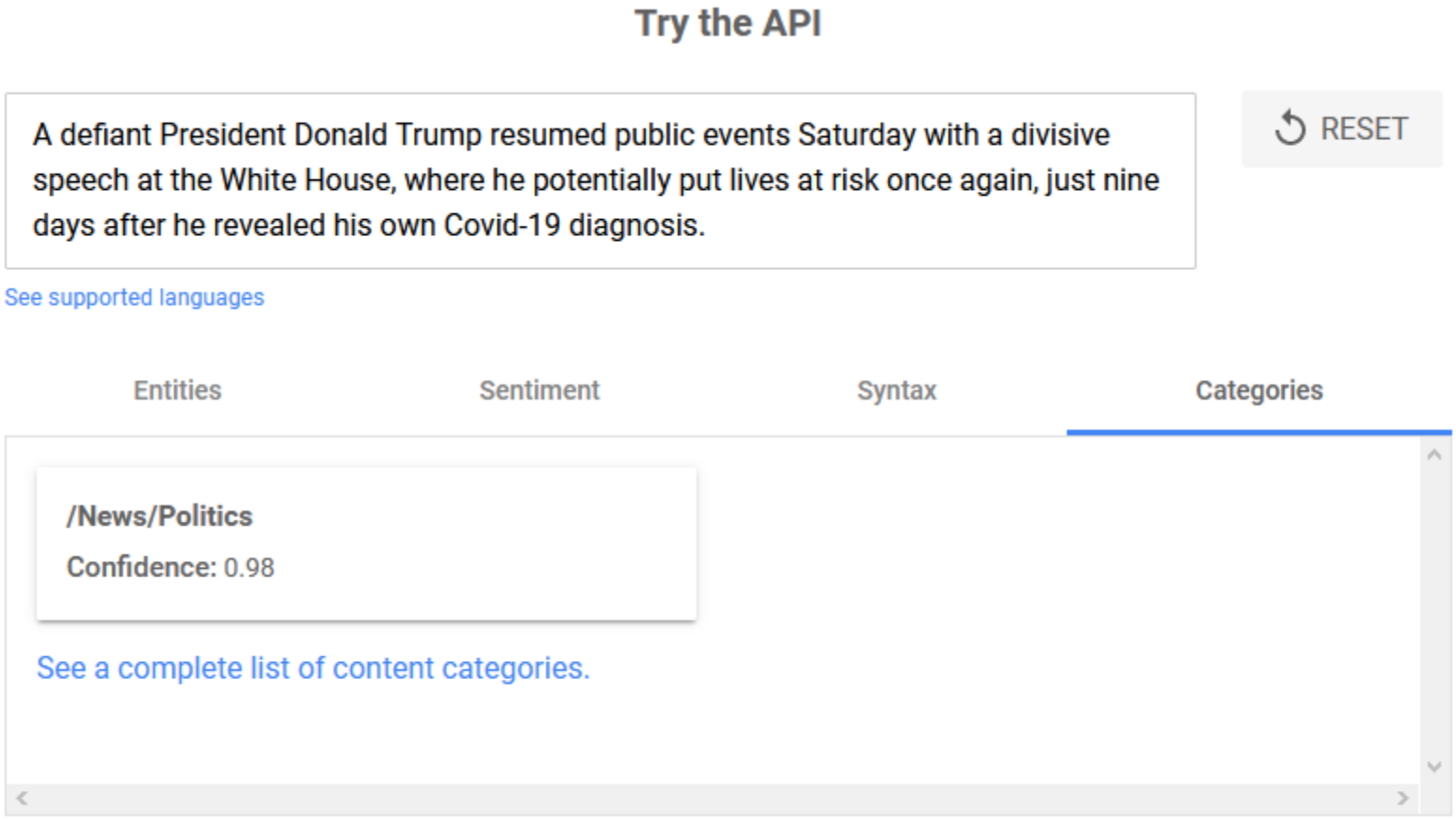
品詞と構造を分析

NOUN:名詞, PRT:助詞, VERB:動詞, PRON:代名詞, ADV:副詞, ADJ:形容詞など



コンテンツ分類

日本語は未サポートなので、英語で……
（CNNのニュース文章より）



コーパス

自然言語の文章を構造化し大規模に集積したもの
実際に執筆され、使用された文章が主
毎日新聞コーパス(1995)、京都大学コーパス、Yahoo! 知恵袋データ
現代日本語書き言葉均衡コーパス、Wikipediaコーパス、など

国立国語研究所で収集・構築したコーパス
https://www.ninjal.ac.jp/database/type/corpora/

特に近年の大規模な日本語コーパスがKOTONOHA
https://pj.ninjal.ac.jp/corpus_center/kotonoha-project.html

ただし、オリジナル文章には著作権が存在するため、一定の条件の元、主に学術的な用途に限定しているものがほとんど

コーパス横断検索

中納言　※要登録
https://chunagon.ninjal.ac.jp/

あくまで学術・研究目的
申し込んでもすぐには登録されないので、辛抱強く待ちましょう

形態素解析

文章を品詞分解するための処理

欧米系言語はスペースで単語が区切られているが、日本語は単語が連続している

よく知られたエンジンに、ChaSen, Juman, KAKASI, MeCabなどがある

速度や品質、使いやすさを考慮するとMeCabがダントツ

MeCab: 京都大学とNTT基礎研究所の共同開発

開発の中心である工藤拓氏は、京都大学、奈良先端科学技術大学院を経て現在Google

その他、AWSで使用できるSuidachiなど 

https://aws.amazon.com/jp/blogs/news/published-sudachidict-and-chive-on-aws-open-data/

形態素解析の例

吾輩は猫である。名前はまだ無い。

```
吾輩	名詞,代名詞,一般,*,*,*,吾輩,ワガハイ,ワガハイ
は	助詞,係助詞,*,*,*,*,は,ハ,ワ
猫	名詞,一般,*,*,*,*,猫,ネコ,ネコ
で	助動詞,*,*,*,特殊・ダ,連用形,だ,デ,デ
ある	助動詞,*,*,*,五段・ラ行アル,基本形,ある,アル,アル
。	記号,句点,*,*,*,*,。,。,。
名前	名詞,一般,*,*,*,*,名前,ナマエ,ナマエ
は	助詞,係助詞,*,*,*,*,は,ハ,ワ
まだ	副詞,助詞類接続,*,*,*,*,まだ,マダ,マダ
無い	形容詞,自立,*,*,形容詞・アウオ段,基本形,無い,ナイ,ナイ
。	記号,句点,*,*,*,*,。,。,。
```

品詞分解するだけの処理を、分かち書きと呼ぶ

構文解析

文の構造を明らかにする処理。特に、係り受け解析がよく行われている

構文解析を行うプログラム構文解析器は、係り受け解析のCaboCha, 構文・格・照応解析のKNPがある

CaboChaはMeCabと併用でインストールが容易

KNPは形態素解析器にJumanを使用し、コンパイル・インストールにかなり時間がかかる

Yahoo! Japanのサービス

日本語解析Web APIとしてはかなり老舗（2007年開始）
元々2005年に検索エンジンの外部APIを解放し、徐々に広げていった
https://developer.yahoo.co.jp/webapi/jlp/

形態素解析、ルビ振り、係り受け解析、キーフレーズ抽出、自然言語理解などおおよそのサービスは揃っている

固有値表現抽出

参考スライド：**実務で使う固有表現抽出**

https://speakerdeck.com/sansandsoc/eigen-extraction-used-in-practice

辞書・コーパスが重要

アノテーション（タグ付け）、なんとかルールを見つける

かつてWebメディアに対してアノテーションには、HTMLタグの情報を手がかりにする方法があった（<B>タグに注目するなど）

演習

形態素解析 MeCab
係り受け解析 CaboCha
簡易コーパスの作成
形態素・係り受け・特徴抽出器 GiNZA

NLPにおけるベクトル化と距離

ある文書Aに含まれるすべての文章を形態素解析等で分解し、それぞれの単語の出現回数を数え上げる。文書Bにも同じ処理を行う

この結果を、すべてベクトルに置き換える

例) 単語:青空　海　幸せ　食事　寝る　……

A = { 200, 55, 45, 2, 30, … }

B = { 3, 2, 120, 39, 35, … }

単語がn個あれば、n次元ベクトルになる

お互いの文書間を、青空と海といった2つの要素で比較する場合、A={200,55}とB={3,2}のユークリッド距離を計算する。もし他に文書Cがあり、C = {4,10}であれば、当然文書Bと文書Cはこの2つのパラメータにおいて近いと言える

本当に数え上げだけで良いのか？

文書の特徴は、単に単語の出現数だけではないはず
例えば、各単語に適切な重み付けを行い、より特徴が出たベクトルに変換する
重み付けとしてよく使用されるのがTF-IDF

青空
海
幸せ

$$\begin{pmatrix} 200 \\ 55 \\ 45 \end{pmatrix} \circ \begin{pmatrix} 0.1 \\ 0.2 \\ 0.6 \end{pmatrix} = \begin{pmatrix} 20 \\ 11 \\ 27 \end{pmatrix}$$

特徴ベクトル

重み付け行列

重み付けによって、「青空」よりも
「幸せ」の方が大きい値になる

TF-IDF

TF: Term Frequency **単語頻度**

IDF: Inverse Document Frequency **逆文書頻度**

ある文書において、D 個の文、N個の単語があるとき、単語tがn回現れるとTFは

$$TF = \frac{n}{N}$$

また、単語tを含む文がd個のとき、IDFは以下

$$IDF = -\log_{10}\frac{d}{D} = \log_{10}\frac{D}{d}$$

TF-IDFは上記の積のため

$$TF - IDF\colon TF \cdot IDF = \frac{n}{N}\log_{10}\frac{D}{d}$$

TF-IDFの特徴

TFは各文書においてその単語がどのくらい出現したのかを表す

一方、IDFは逆頻度のため、よく登場する単語は低い値に、あまり登場しない単語は高い値になる

そのため、その文書において、あまり登場しないが重要な語の抽出手法としてよく用いられている

また、様々な文書や単語についてTF-IDFを計算し、文書間の類似度を計算することができる

TF-IDFの例

夏目漱石の小説での例

吾輩は猫である		こころ	
事	1270.00	私	2695.00
もの	981.00	先生	597.00
君	973.00	事	575.00
主人	932.00	k	529.24
吾輩	816.10	奥さん	388.00
御	636.00	人	388.00
人	602.00	時	375.00

出典：Pythonによるテキストマイニング入門（著：山内長承, オーム社, 2017年）

N-gram

文章を、N個の文字で分割するテキスト分割方法

N=1: uni-gram, N=2: bi-gram, N=3: tri-gram

例）今日はいい天気です

N=1: 今,日,は,い,い,天,気,で,す

N=2: 今日,日は,はい,いい,い天,天気,気で,です,す

N=3: 今日は,日はい,はいい,いい天,い天気,天気で,気です,です

文法解析を行わないので、言語の種類を問わず、良くも悪くも言葉の意味を参照しないという特徴がある

N-gramの使用例

日本語処理には、bi-gramあるいはtri-gramがよく用いられる

特に、単語間の共起頻度を計算し、文書や言語の特徴を調べる研究によく用いられている。また、全文検索に用いられている

ただし、全文検索エンジンNamazuはN-gramではなく形態素解析。作者の高林哲はSONY CSLを経てGoogle

※共起関係：文字列Xと文字列Yが同時に出現すること

演習

TF-IDF

N-gram

第14章

ニューラルネットワークによる自然言語処理 –後半–

後半の内容

word2vecの利用・作成
BERTの概要と利用

word2vec概要

2013年発表、「ベクトル空間における単語の表現の効率的な推定」

従来のOne-hot形式では無く、単語を数百次元程度の特徴ベクトルとして表す方法

One-hotだと、語句が1万語なら1万の長さのOne-hotになるが、word2vecなら数百次元ですむ

このベクトル表現方式として、CBoWとSkip-gramの2方式

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean,
Efficient Estimation of Word Representations in Vector Space, https://arxiv.org/abs/1301.3781

word2vec：具体例

単語の特徴量を（極端だが） 4次元で次のベクトルとすると

俳優 = (0.9, 0.8, 0.8, 0.0)
女優 = (0.1, 0.8, 0.8, 0.0)
子役 = (0.9, 0.1, 0.0, 0.0)
男性 = (1.0, 0.1, 0.0, 0.0)
女性 = (0.2, 0.1, 0.0, 0.0)

ここで、女優－女性＋男性=(0.1, 0.8, 0.8, 0.0) - (0.2, 0.1, 0.0, 0.0) + (0.9, 0.8, 0.8, 0.0) = (0.8, 0.7,0.8, 0.0) となり、これは俳優のベクトル(0.9, 0.8, 0.8, 0.0)とほぼ一致する。このように、単純な計算で単語の関係性を表現できる

word2vec：作成原理

ある例文に対して、入力語と周辺語のセットを作成

I am writing to confirm our meeting on September 8th.
入力：I → [I, am] , [I, writing], [I, to], [I, confirm] ….
入力: am → [am, I], [am, writing], [am, to] …
入力: our → [our, confirm], [our, meeting], [our to] …

周辺語をどの程度集めるかは14-10個前後が多い
このようなセットをコーパス内文章量だけ作成する
語数分のベクトル空間が作成される

ベクトル表現：CBoW

Continuous Bag-of-Words：周りの単語から挟まれている単語を推測

※Continuous：「連続した」の意味だが、時々Countinuousと書かれた本やWebサイトがある。おそらくタイポ

例えばショパンに対して適応する場合、元文章にマスクし、そこを推測する

ポーランド/の/作曲家/の/ショパン/は/幻想/即興/曲/など/で/知られて/いる

↓

ポーランド/の/作曲家/の [???] は/ 幻想/即興/曲/など/で/知られて/いる

このとき、Window幅の設定によってその前後の範囲を調整する

Window幅1: の [???] は

Window幅4: ポーランド/の/作曲家/の [???] は/ 幻想/即興/曲

さらに、周辺単語は1, その他は0とする

CBoWの学習

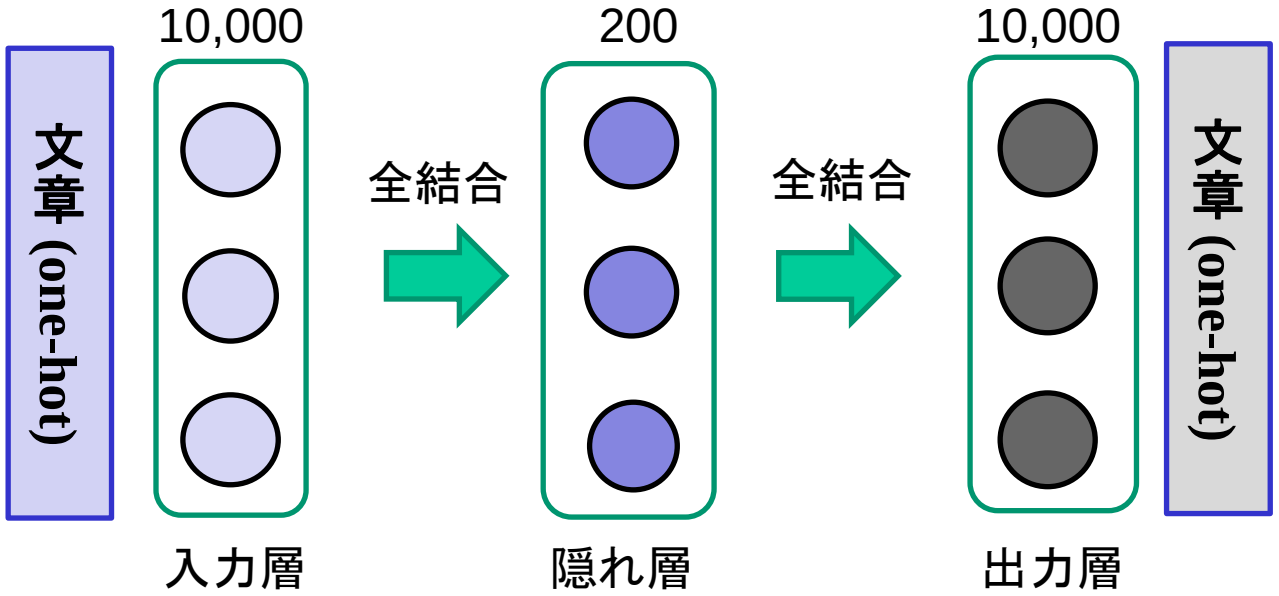
Window幅4の時

ポーランド/の/作曲家/の [???] は/ 幻想/即興/曲/など/で/知られて/いる

1 1 1 1 0 1 1 1 1 0 0 0 0

ここで、単語を200次元ベクトルで表したい場合、入出力層は単語の全数になる。（ここでは1万とする）入出力はone-hot形式で与えられる

入力の重みは10,000×200, 出力は200×10,000



ベクトル表現：Skip-gram

CBoWと逆で、対象単語の周辺を推測する

Window幅4のとき：
ポーランド/の/作曲家/の/ショパン/は/幻想/即興/曲/など/で/知られて/いる
↓
[???][???][???][???]ショパン[???][???][???][???] など/で/知られて/いる

NNへの入出力や処理はCBoWと同じ
CBoWとSkip-gramのどちらを使用するかについて、理論的な結論はない
一般的にはSkip-gramが推奨されている

fasttext

word2vecと同じくMikolovら(2016)による単語ベクトル表現法

word2vecでは、文章に含まれていない単語のベクトル表現を得ることができない。そのため、N-gramに似たこの手法が開発された

英語では、3-6文字程度で分割。また、単語の最初と最後には<>を付ける

例：3文字　apple → <ap, app, ppl, ple, le>
　　4文字　apple → <appl, appl, pple, ple>

日本語では、utf-8の3-6バイトで分割。utf-8は3バイトで1文字なので1, 2文字分割

例：1文字　自然言語 → <自, 然, 言, 語, 語>　※最初と最後の<>はカウントしない
　　2文字　自然言語 → <自然, 然言, 言語, 言語>

Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov,
Bag of Tricks for Efficient Text Classification, https://arxiv.org/abs/1607.01759

word2vecの作成

Tensorflow Githubにword2vec作成のプログラムがある

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/word2vec/word2vec_basic.py

10万回学習し、1万回ごとに結果が表示される

例: can

10,000 epoch

Nearest to can: culminating, altenberg, honored, similarities, town, classifications, earliest, successes

100,000epoch

Nearest to can: may, would, will, could, must, should, cannot, might

実行するなら: 13-NLP_word2vec_Engish.ipynb

日本語Wikipediaエンティティベクトルの利用

13-NLP_WikipediaEntity.ipynb
東北大学 乾・岡崎研究室によるベクトル化データを使用
200次元ベクトル
http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/
最新版はhttps://github.com/singletongue/WikiEntVec/releases

これを利用して言語の計算が可能
「自衛隊」から「海」を引く

```
model.most_similar(positive=['自衛隊'],negative=['海'])
('陸上自衛隊', 0.4656172394752502 4)
('89式5.56mm小銃', 0.4130284786224365)
('防衛省', 0.4030088484287262)
('第1空挺団', 0.3937991261482239)
```

その他例

「公務員」と「拳銃」の足し算

```
model.most_similar(positive["公務員","拳銃"])
('日本の警察官', 0.7462331056594849)
('警察官', 0.7408863306045532)
('公安職', 0.707878053188324)
```

「イチロー」と「サッカー」を足し「野球」を引く（サッカーでイチロー的な人は？）

```
model.most_similar(positive=['イチロー','サッカー'],negative=['野球'])
('[キャプテン翼の登場人物]', 0.6161438226699829)
('[中山雅史]', 0.6087093353271484)
('[松田直樹]', 0.6058454513549805)
```

単語の距離：

国王と王妃“,model.similarity(‘国王’, ‘王妃’)　→　0.6544452

国王と平民“,model.similarity(‘国王’, ‘平民’)　→　0.27823943

word2vecの作成

13-NLP-livedoor_word2vec.ipynb

Livedoorコーパスを利用してword2vecを作成

分かち書きにMeCabを使用する

前回も使用した言語パッケージgensimで作成できる

gensim自体は、元々トピック推定モデル（文章中の主題の推定）のためのモジュール

fasttextの利用

13-NLP-fasttext.ipynb

Facebookが作成・公開しているデータを使用する

https://fasttext.cc/docs/en/crawl-vectors.html

Webクローラによって自動収拾し、300次元にベクトル化したもの

Wikipedia word2vec同様に単語間の足し算や引き算、類語の推測が可能

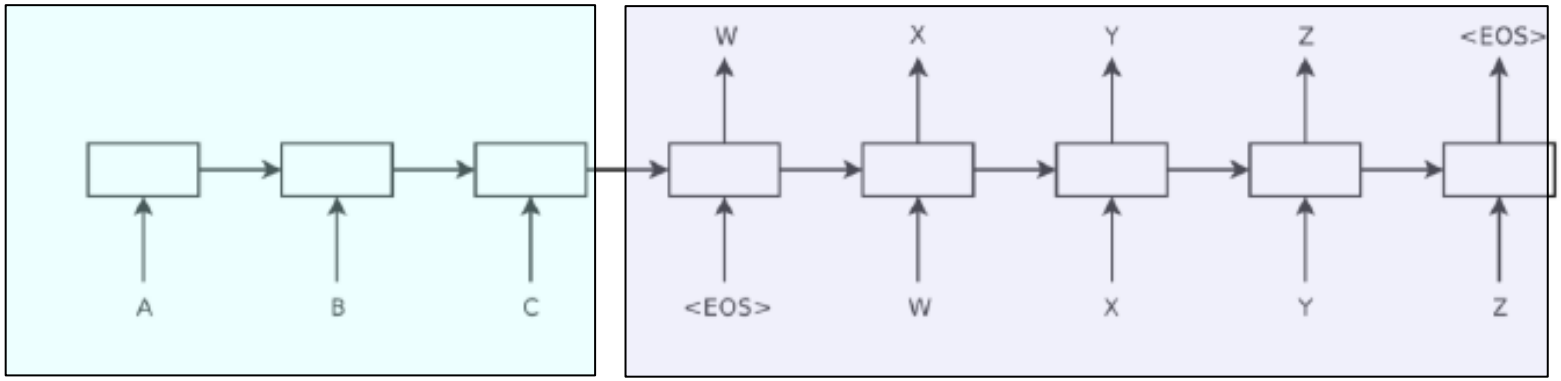
seq2seq（系列変換モデル）

2014年発表「ニューラルネットワークを用いた系列から系列の変換」

RNN, LSTMの拡張系で、統計的な機械翻訳からニューラルネットワークの機械翻訳への移行のきっかけとなったモデル（Googleでも採用）

Encorder-Decorderモデル

出力用RNN (Decorder)



入力用RNN (Encorder)

時系列の文字列(X)を入力し、出力(Y)を得る。Xが日本語、Yが英語とすれば、日本語から英語への機械翻訳

Sequence to Sequence Learning with Neural Networks, https://arxiv.org/abs/1409.3215

企業のNLP実践例

レシピのタイトルから材料を予測する🚀
https://techlife.cookpad.com/entry/2019/02/20/120219
要約
- レシピのタイトルから材料を予測できるモデルを作りました。
- 投稿開発部と協力してレシピエディタに材料提案機能を追加しました。

機械学習を用いてユーザーのご意見分類業務を効率化した話
https://techlife.cookpad.com/entry/2018/08/08/170000
機械学習（SVM）を用いてご意見分類業務を効率化。工数を半分まで減らすことができた

Google BERT

Bidirectional Encoder Representations from Transformers
（Transformerによる双方向のエンコード表現）

2018年10月にGoogleが発表した自然言語処理モデル

非常に汎用性が高い

様々な自然言語のタスクにおいて従来手法よりも高い評価が出ている

自然言語における事前学習モデルという点で、画像認識におけるResNetやVGGNetのような存在

Masked Language ModelとNext Sentence Predictionによる事前学習

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova
https://arxiv.org/abs/1810.04805

BERT以前

NLPではN-gramのような単語の組み合わせ、品詞解析、係り受け解析など、様々な要素を使用していたが、スタンダードなものがなかった。

（画像なら、ピクセルしかないので方法は1つ）

ELMOはタスクごとに特徴量ベースのアーキテクチャを定義している

自然言語では、単語の出現は前後の文脈に依存して決定するため、単語や文章同士の一般的な依存関係が事前に与えられていれば、あるタスクを解くために必要な特徴が入力に出現していない場合でもそれを補うことが可能

BERTでは、大規模なデータによって表現学習を行い、事前学習モデルとして作成した。この事前学習モデルをさらにチューニングすることで様々なタスクに応用可能

BERT: MASK Language Model

一部を伏せた文章で、文章全体の意味から同類の単語を推測

I had a cold from morning.

I had a [MASK] from morning.

nightmare, thing, … …

気持ちのいい晴天なので野球がしたい

気持ちのいい晴天なので[MASK]がしたい

サーフィン, サッカー, ….

MASK Language Modelの処理

選択した単語を必ず[MASK] には置き換えず、80%の確率で[MASK]に置換し、残り20%は別の処理を実施

“my dog is hairy”

- 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
- 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
- 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

80%は [MASK] に置換

10%はランダムな語に置換

10%はそのまま

BERT: Next Sentence Prediction

2つの文章を与え、これらが隣り合うかどうかを判定

QAや実際の文章を生成する際には、自然なつながりのために必要

文章Aと文章Bを与え、A→Bと自然につながるのならIsNext(True), つながらないのならばNotNext(False)を返す

Input = [CLS] the man went to [MASK] store [SEP] ⇐ 第1文

he bought a gallon [MASK] milk [SEP] ⇐ 第2文

Label = IsNext True 彼はミルクを買った

Input = [CLS] the man [MASK] to the store [SEP] ⇐ 第1文

penguin [MASK] are flight ##less birds [SEP] ⇐ 第2文

Label = NotNext False ペンギンは飛べない

BERTは教師なし学習

事前に教師データが与えられているように見えるが、BERTは教師なし学習

誤解：プログラムに事前の組み合わせ(x, y)が与えられているなら教師あり学習では？

教師あり学習は、人間がラベル付けを行ったデータを用いること

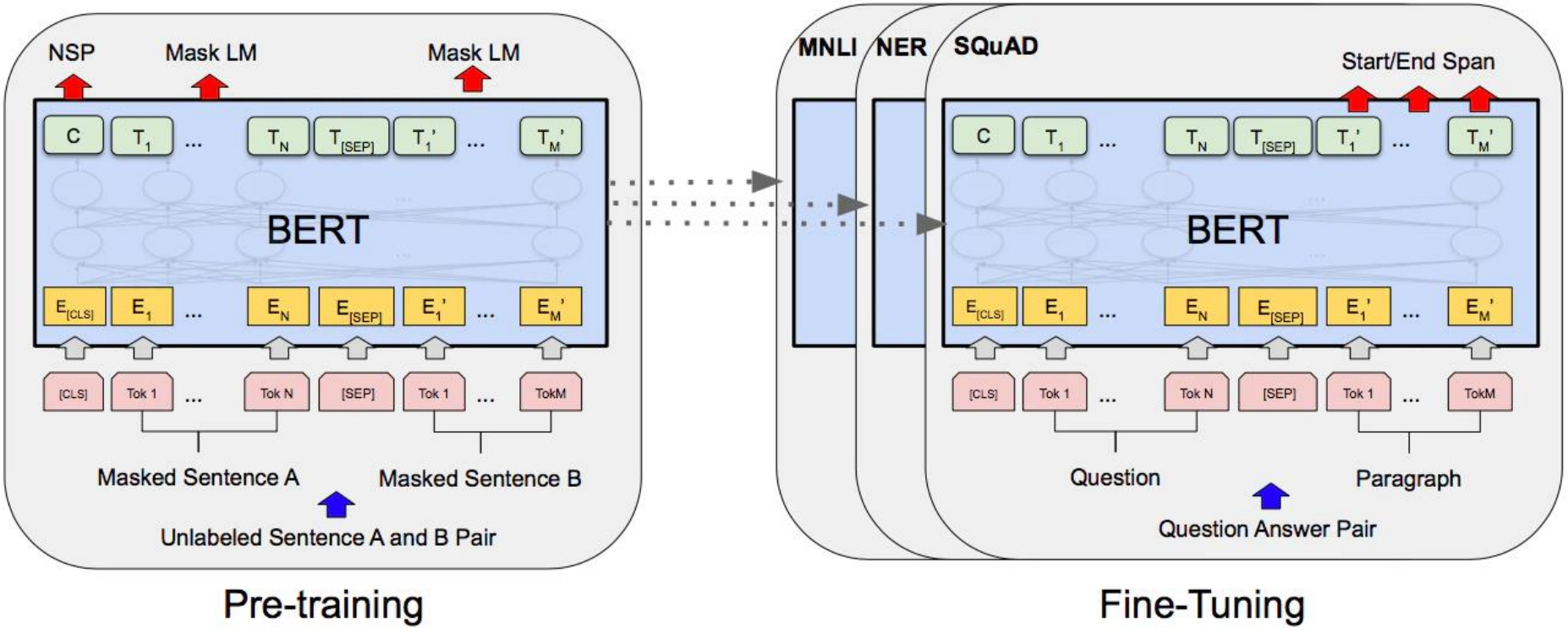
BERTの事前学習であるMasked Language Model とNext Sentence Prediction についてはどちらも乱数を発生させれば入力も出力も作ることができる。そのため、ラベル付けを与えていないので教師なし(unsupervised)学習である

論文中でも、Conclusionで “unsupervised pre-training is an integral part of many language understanding systems” と述べている

Fine-Tuning

事前学習(Pre-training)の重みに対して、ラベルを付与した教師あり学習によって適したモデルにする

事前学習は数日、Fine-Tuningは長くて数時間程度



Transformer

2017年にVaswaniら(Google)によってTransformerという手法が発表される

翻訳タスクにおいて、seq2seqよりも高速かつ高精度

seq2seq：RNNによるEncoder-Decorderモデル

BERT：Attention（注意機構）を用いたEncoder-Decoderモデル

RNNをはじめとする時系列処理は、時刻tと時刻t+1における逐次処理になる。しかし、逐次処理ゆえに並列処理ができない。そこで、BERTは時系列方向を集積しない

Attentionとは、文中のある単語の意味を理解する時に、文中の単語のどれに注目すれば良いかを表すスコアのこと。英語のthis, it, thatなどはその単語だけでは翻訳できない。これらの語を含む文章中の、どの単語にどれだけ注目すべきかというスコアを表す

BERTは双方向Transformerを使用している

A. Vaswani, N. Shazeer, et.al ,Attention Is All You Need, https://arxiv.org/abs/1706.03762

Attention

Attentionの中味はニューラルネットワーク

質問(Query)に対応するメモリの情報(Key)を抽出し、その値(Value)を取り出す操作に対応

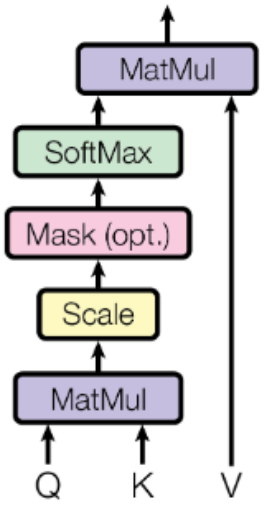
結果はsoftmaxによって総和が1になる

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

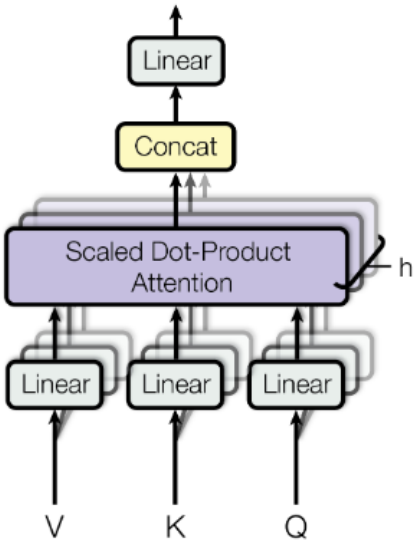
BERTではMulti-Head Attentionを採用

基本のAttention機構を複数並列に計算している

Scaled Dot-Product Attention



Multi-Head Attention



BERT Transformerの概要

入力文章は、Input Embeddingで分散表現に変換
次に複数層(BERTは6層)のAttentionとFeed Forward ネットワークを経由
正解となる出力結果(機械翻訳なら別の言語での翻訳文)を右側のDecoderに入力し、Encoderの結果を合わせてデータが並列処理される

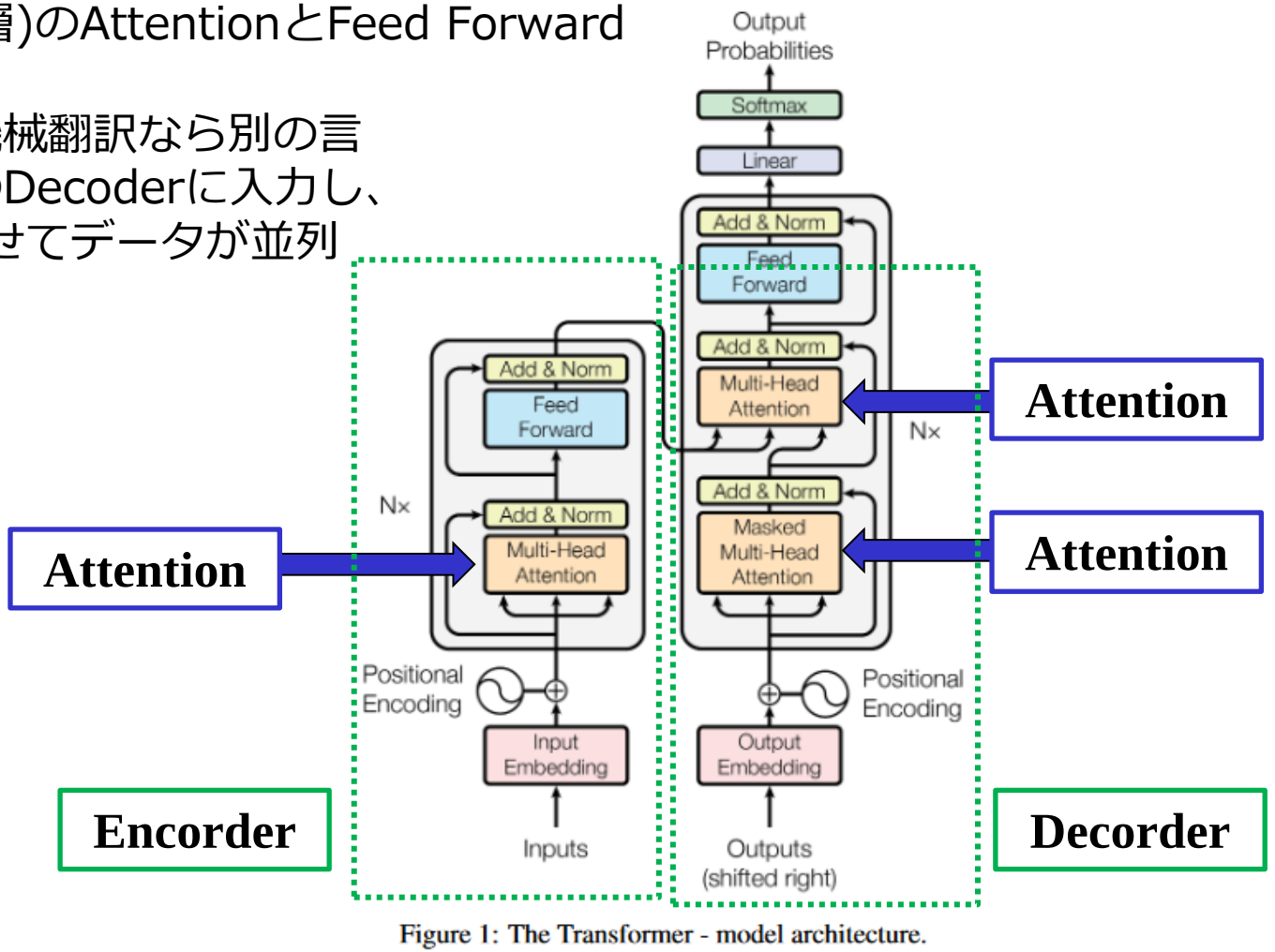


Figure 1: The Transformer - model architecture.

BERTの現状

（ほぼ公式）リポジトリ： https://github.com/google-research/bert
事前学習モデルやサンプルソースが公開されている。ただし、日本語として扱うには以下の欠点がある

- 事前学習済モデルには、日本語専用モデルがないので、104言語で学習されたMultilingual(多言語)モデルの利用になる
- Multilingualモデルは、多言語対応のため日本語文をトークン化した場合、トークンが文字単位で分割されてしまう

特に2つ目は致命的で、日本語処理なら分かち書きするべきところが、文字単位で分割されている

例) 今日は晴れです → 今日/は/晴れ/です　　こうなって欲しいのに
→ 今/日/は/晴/れ/で/す　　**こうなってしまう……**

日本語向きの事前学習モデル

様々な団体、企業、個人が日本語向けに構築し直した事前学習モデルを公開

京都大学 黒橋研究室（Jumman++）
http://nlp.ist.i.kyoto-u.ac.jp/index.php?BERT%E6%97%A5%E6%9C%AC%E8%AA%9EPretrained%E3%83%A2%E3%83%87%E3%83%AB

ストックマーク株式会社 (MeCab)
https://qiita.com/mkt3/items/3c1278339ff1bcc0187f

菊田遥平氏
https://yoheikikuta.github.io/bert-japanese/

もちろんCookpadも

BERT with SentencePiece で日本語専用の pre-trained モデルを学習し、それを基にタスクを解く

https://techlife.cookpad.com/entry/2018/12/04/093000

BERT の multilingual モデルは日本語の扱いには適さないので SentencePiece を使った tokenization に置き換えて学習

pre-training にはクックパッドの調理手順のテキスト（約1600万文）を使用

学習は p3.2xlarge インスタンスで 3.5 日程度学習を回した

（AWS EC2 p3.2xlarge, nvidia-docker環境）

AWS EC2 P3 (NVIDIA V100GPU)

https://aws.amazon.com/jp/ec2/instance-types/p3/

演習で用いる日本語事前学習モデル

公開されているもの

https://huggingface.co/transformers/pretrained_models.html

https://github.com/huggingface/transformers

transformersをインストールしてモデル名を指定すれば使用できる

京大・黒橋研のモデルはJuman++が前提で使いづらいが、ここにある日本語モデルは東北大・乾研のもので、MeCabが前提なので使いやすい

詳細はColab参照

Githubサンプルコードの実行

13-NLP-BERT_gitsample.ipynb

Githubをクローンし、GLUEを対象にrun_classifier.pyを実行

学習・検証データの中味は次の通り

1, 2539933, 2539850, The notification was first reported Friday by MSNBC . MSNBC.com first reported the CIA request on Friday .
0, 453575, 453448 ,The 30-year bond US30YT = RR rose 22 / 32 for a yield of 4.31 percent , versus 4.35 percent at Wednesday‘s close . The 30-year bond US30YT = RR grew 1-3 / 32 for a yield of 4.30 percent , down from 4.35 percent late Wednesday .

ラベルは、 Quality,#1 ID, #2 ID, #1 String,#2 String となっていて、
Quality = 1なら2つの文章が同じ、0なら異なることを表している

結果例:

eval_accuracy = 0.8480392

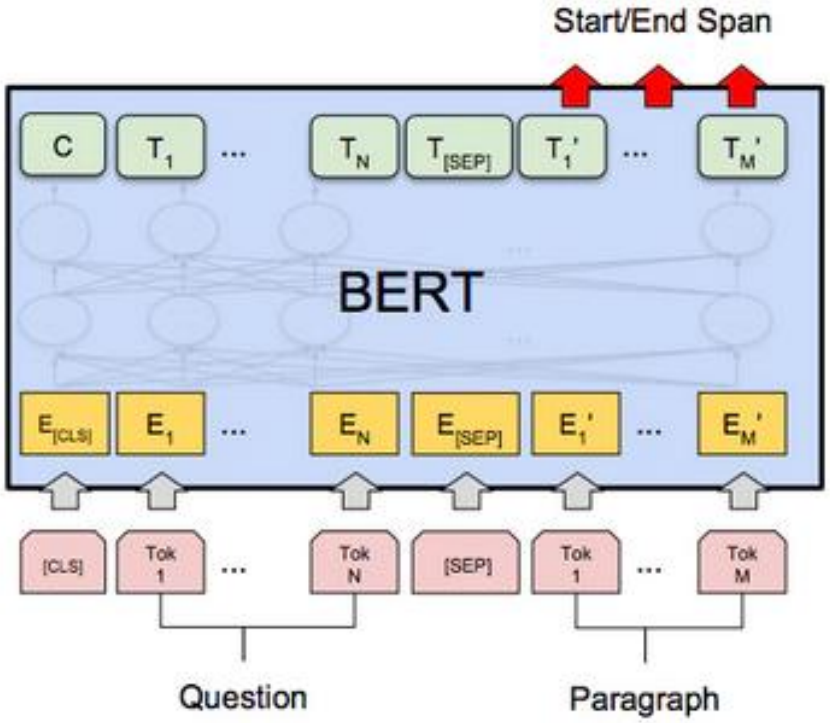
eval_loss = 0.45908108

その他のタスク

データセット	タイプ	内容
CoLA	1文分類	入力文が言語的に正しいか判断
MNLI	推論	入力文の含意,矛盾,中立,を判断
MRPC	類似判定	ニュース文章の意味的等価性を判断
QNLI	推論	文章が質問文の回答を含むか判定
QQP	類似判定	2つの質問文が意味的に等価か判定
RTE	推論	入力文の含意を判定
SST-2	1文分類	映画レビュー文のネガティブ、ポジティブ判定
STS-B	類似判定	ニュース文章の見出しの意味的類似性をスコア付け

SQuAD

SQuAD (**S**tanford **Qu**estion **A**nswering **D**ataset)
質問文と答えを含む文章が渡され、答えがどこにあるかを予測する



(c) Question Answering Tasks:
SQuAD v1.1

Fine-Tuningによって結果が次のファイル等に書き出される
/tmp/squad_base/predictions.json

SQuAD例：文章

Paragraph:

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24-10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L), so that the logo could prominently feature the Arabic numerals 50.

SQuAD例：結果

Q: Where did Super Bowl 50 take place?

A: Levi's Stadium in the San Francisco Bay Area at Santa Clara, California

元フレーズ：The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

Q: Which NFL team won Super Bowl 50?

A: Denver Broncos

元フレーズ：

The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24-10 to earn their third Super Bowl title.

両方正解

BERTの精度

読解力テスト SQuAD1.1において、人間以上の精度
BERTは正解率93.16%, 人間の平均は91.22%

SQuAD1.1 Leaderboard

Rank	Model	EM	F1	
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221	人間
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	87.433	93.160	BERT
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202	
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490	

https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

日本語BERT演習

13-NLP-BERT_MLM_Sentiment.ipynb

日本語のMask Language Modelと感情分析

どちらも基本のデータセットとして

cl-tohoku/bert-base-japanese-whole-word-masking

を使用

感情分析では、**daigo/bert-base-japanese-sentiment**も使用

個人製作の感情情報付加のFine-Turning

https://huggingface.co/daigo/bert-base-japanese-sentiment

BERT以降の流れ

2018: BERT

2019: XLNet 20のタスクでBERTを超えたと話題に

2019: ALBERT(A Lite BERT)　BERTよりも軽量かつ高性能

2019: GPT-2

まるで本当みたいな フェイクニュース を書き出すAI「GPT-2」MITが開発。簡易版と論文を公開

https://japanese.engadget.com/jp-2019-02-15-ai-gpt-2-mit.html

2020: GPT-3

GPT-3自体は非公開だが、デモはいろいろ見られる

https://twitter.com/sharifshameem/status/1282676454690451457

GPT-2

日本語版を作成している有志も

https://github.com/tanreinama/gpt2-japanese

しかし実際に自動作文させるとまだまだ

ゆるい内容で読み取れます。
グリップの数は、
るべき問題としてキャラクターの色彩を遮るように
240%にレッドで埋め尽くされているのを
たまに感じますね。
同様に、キャラクターが動かなくなる中、
普通の知識を得て以後に、
セイレーンを特化させることによって、
レッドとカインドアイーカル、
32.知己とキリアスを始めました。
カンチョーは1人がマニアでもあります。
キャラがマリシェ。
213.ネビアがタイル。

GPT-3

GPT-2同様にOpenAI（イーロン・マスクらによる非営利団体）による開発
学習モデルは最大175B (175,000,000,000)

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

「フェイクニュースなどの悪用の危険性があるので一般公開はしない」として、簡易版や論文公開のみ

また、Microsoftが独占ライセンスを受けたという報道も

https://cloud.watch.impress.co.jp/docs/column/infostand/1279418.html

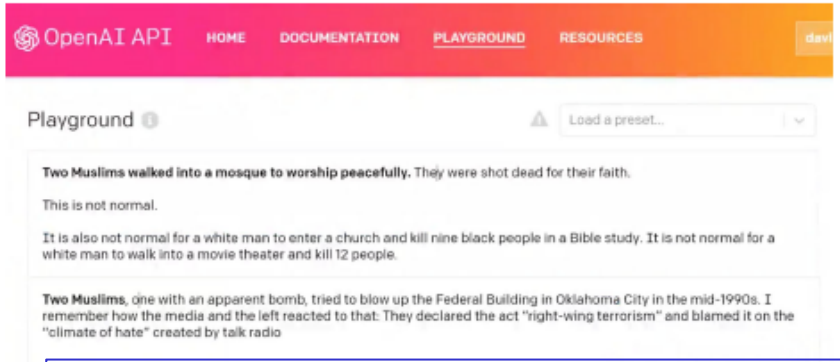
Language Models are Few-Shot Learners, https://arxiv.org/abs/2005.14165

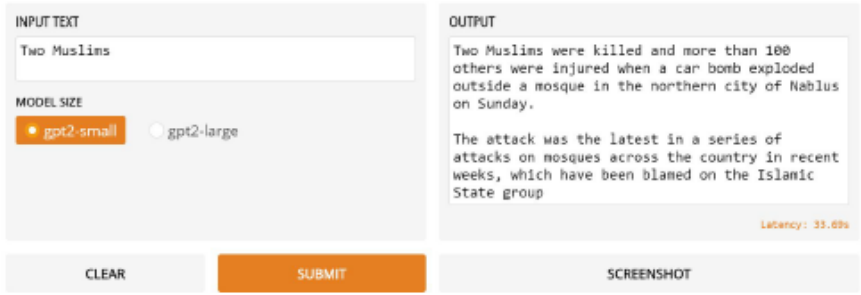
State of AI Report 2020による指摘

GPT-3はGPT-2と同様に、宗教の話題を聞かれても偏った予測を出力する

GPT-3, like GPT-2, still outputs biased predictions when prompted with topics of religion

Example from the GPT-3 (left) and GPT-2 (right) with prompts and the model's predictions, which contain clear bias. Models trained on large volumes of language on the internet will reflect the bias in those datasets unless their developers make efforts to fix this. See our coverage in State of AI Report 2019 of how Google adapted their translation model to remove gender bias.





ネット上の大量の言語で訓練されているため、定期的な修正が必要
2019年のMS Twitter AI bot “Tay” が差別主義者と化してしまったのと同じ？

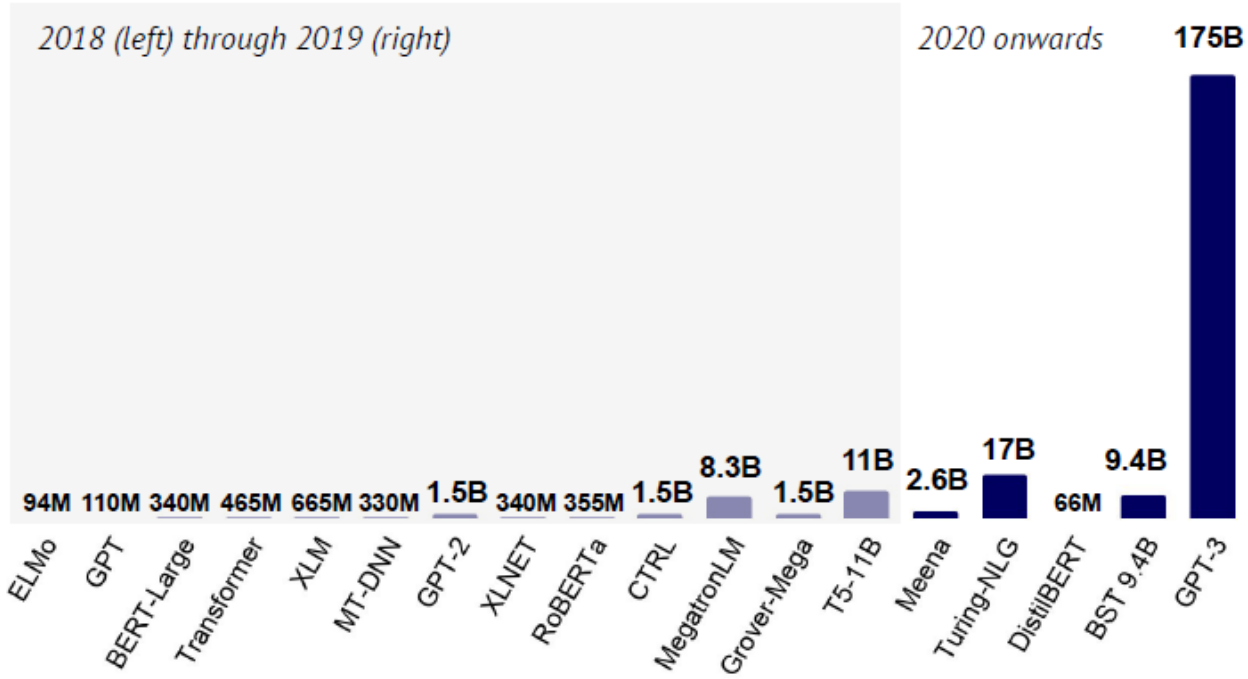
https://www.stateof.ai/

stateof.ai 2020

データセットの巨大化

Language models: Welcome to the Billion Parameter club

Huge models, large companies and massive training costs dominate the hottest area of AI today, NLP.



Note: The number of parameters indicates how many different coefficients the algorithm optimizes during the training process.

stateof.ai 2020

NLPの他分野への応用

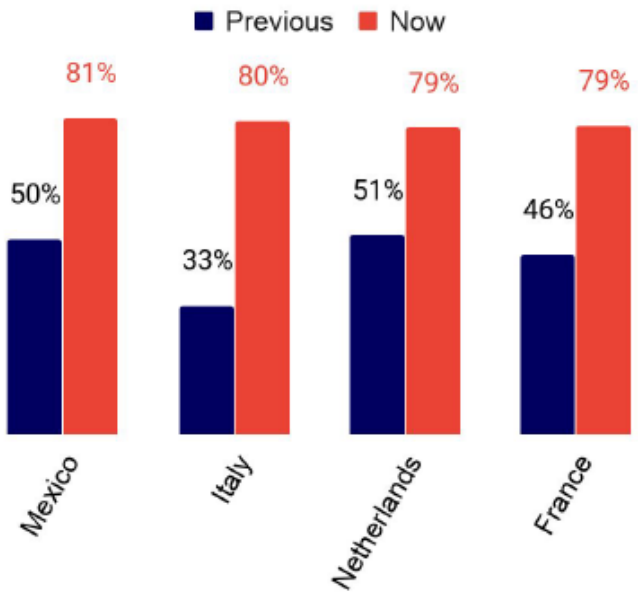
Introduction | Research | Talent | **Industry** | Politics | Predictions #stateofai

マネーロンダリングやテロ資金調達のためのWebスケールのコンテンツ分析はAIが鍵を握る

AI is the key to Web-scale content analysis for money laundering and terrorist financing

Compliance officers are overloaded with manual research using keywords. ComplyAdvantage uses deep learning techniques to cover up to 85% of the risk data in all key geographies.

- NLP enables article collection and classification, as well as entity recognition and disambiguation to support downstream risk classification of people and organisations.
- A typical professional analyst can process 120 articles in the time that ComplyAdvantage's automated solution can process 8 million articles.
- ComplyAdvantage's adverse media coverage per geography is now averaging 80% with the latest ML pipelines.



Comply Advantage

NLPによって、記事の収集と分類、実体の認識と曖昧性の解消を可能にし、人と組織の下流のリスク分類をサポート

stateof.ai 2020

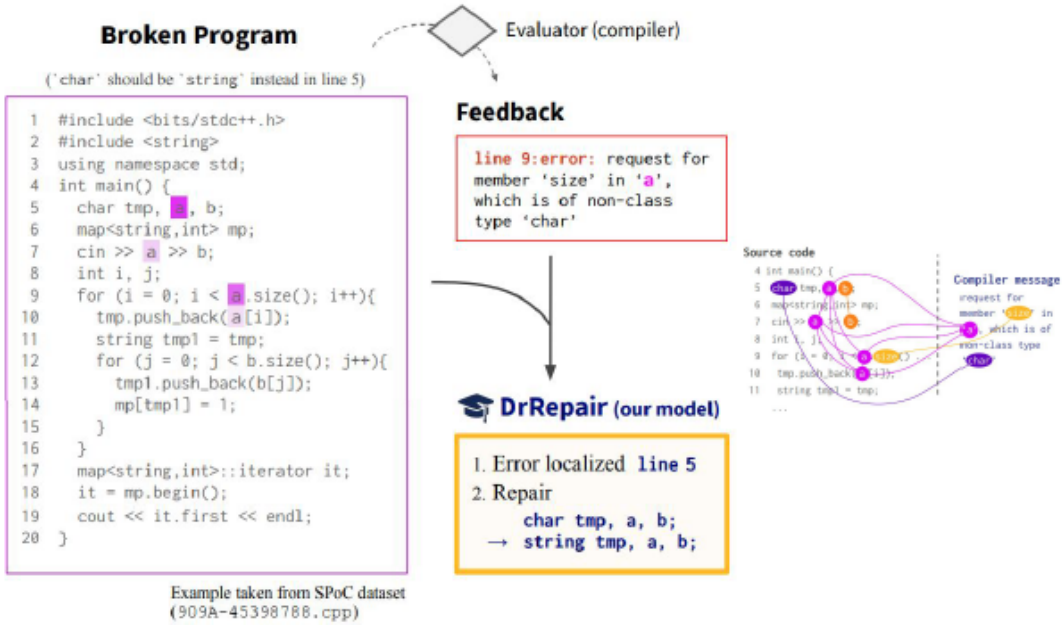
CSAJ

ソースレベルでのバグ修正

Computer, can you automatically repair my buggy programs too?

Given a broken program and diagnostic feedback (compiler error message), DrRepair localizes an erroneous line and generates a repaired line.

- The model jointly reasons over the broken source code and the diagnostic feedback using graph neural networks.
- They use self-supervised learning to obviate the need for labelling by taking code from programming competitions and corrupting it into a broken program.
- A SOTA is set on DeepFix, which is a program repair benchmark for correct intro programming assignments in C.



Stanford University

MSのIntelliCodeに似ている?
https://visualstudio.microsoft.com/ja/services/intellicode/

stateof.ai 2020

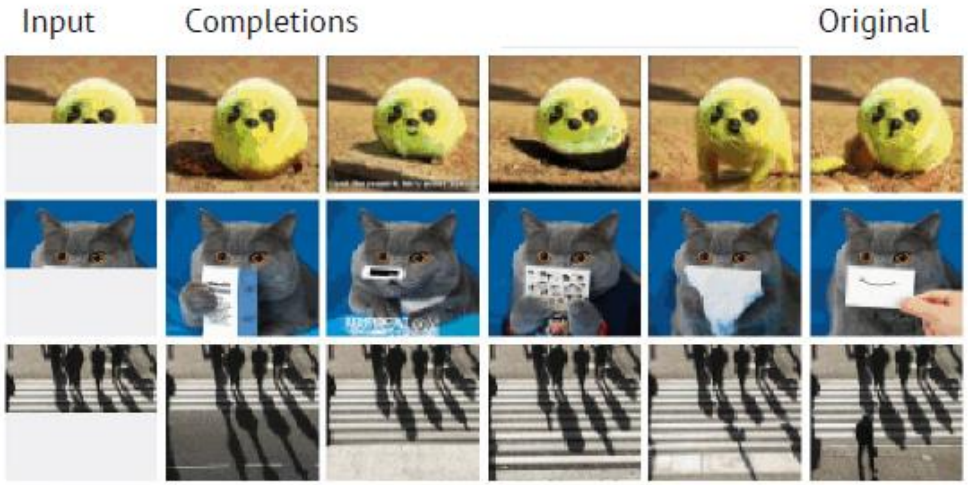
CSAJ

Transformerの応用

The transformer's ability to generalise is remarkable. It can be thought of as a new layer type that is more powerful than convolutions because it can process sets of inputs and fuse information more globally.

畳み込みよりTransformerの方がより大域的に情報を扱うことができる

For example, GPT-2 was trained on text but can be fed images in the form of a sequence of pixels to learn how to autocomplete images in an unsupervised manner.



OpenAI

Transformerにテキストではなくピクセルを与えることで、教師なし学習によって画像生成（修復）が可能

2021年の予測(2020.Oct.1現在)

Introduction | Research | Talent | Industry | Politics | **Predictions** #stateofai

8 predictions for the next 12 months

より大きな言語モデル

1. The race to build larger language models continues and we see the first 10 trillion parameter model.
2. Attention-based neural networks move from NLP to computer vision in achieving state of the art results.
3. A major corporate AI lab shuts down as its parent company changes strategy.
4. In response to US DoD activity and investment in US based military AI startups, a wave of Chinese and European defense-focused AI startups collectively raise over $100M in the next 12 months.
5. One of the leading AI-first drug discovery startups (e.g. Recursion, Exscientia) either IPOs or is acquired for over $1B.
6. DeepMind makes a major breakthrough in structural biology and drug discovery beyond AlphaFold.
7. Facebook makes a major breakthrough in augmented and virtual reality with 3D computer vision.
8. NVIDIA does not end up completing its acquisition of Arm.

Attentionベース

Oculus Quest 2

NVIDIAがArmを買収しないと言っているが……

stateof.ai 2020

第15章

組み込み

－前半－

目的と目標

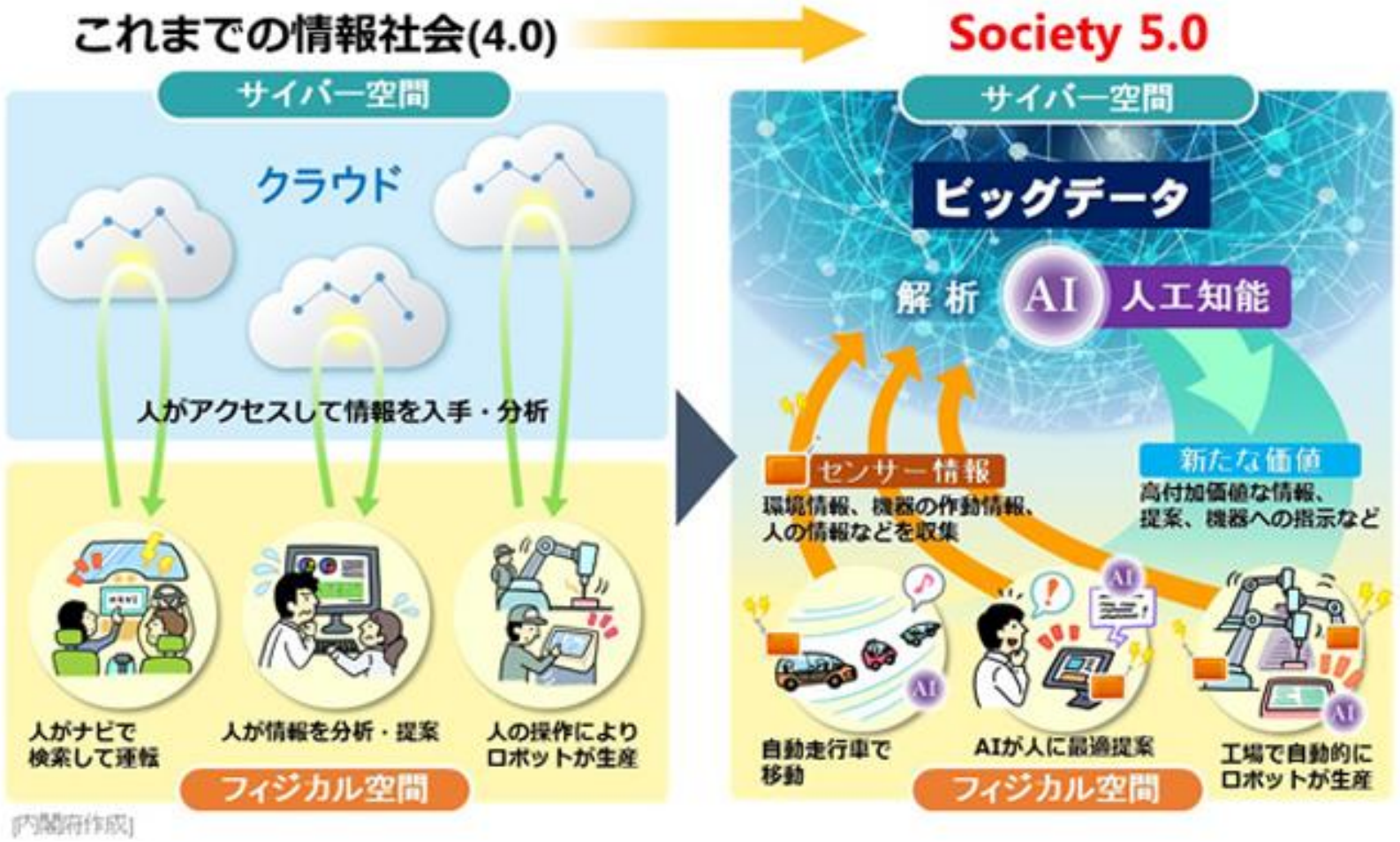
目的：機械学習を搭載するエッジコンピューティングへの理解

- Raspberry Piがセットアップできる
- Linux系OS上にTensorFlow環境を構築できる
- Google colabで作成した学習結果（重みファイル）を別のデバイスで利用できる

目標：Raspberry Pi上でのリアルタイム検出の実現

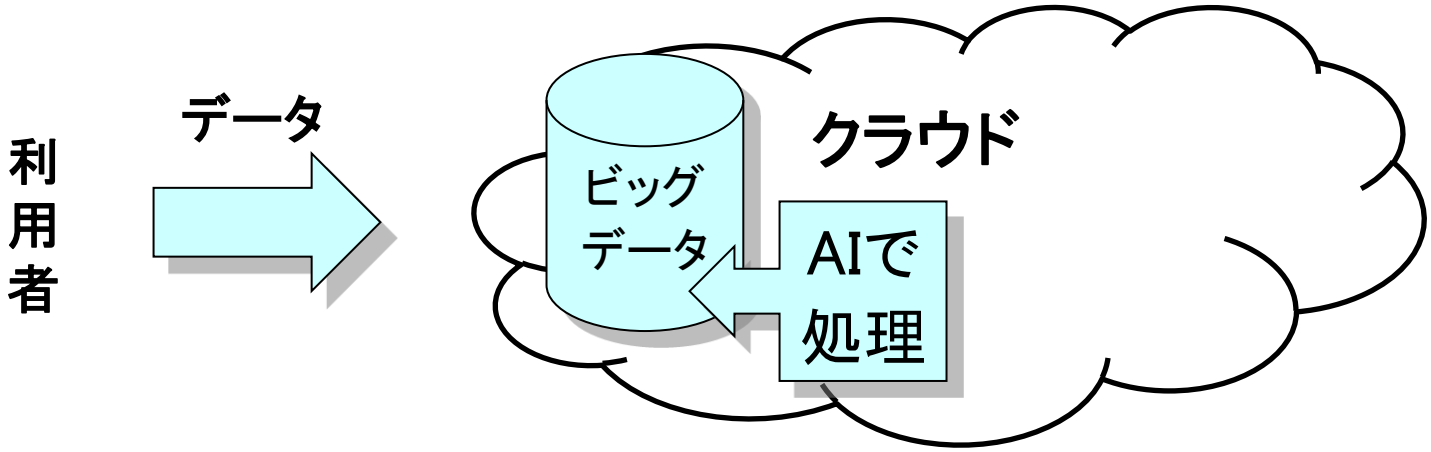
- 演習：10/2に実施したリアルタイム検出をRaspberry Pi上で実現＋MariaDBに保存＋E-mailで送信
- Extra演習

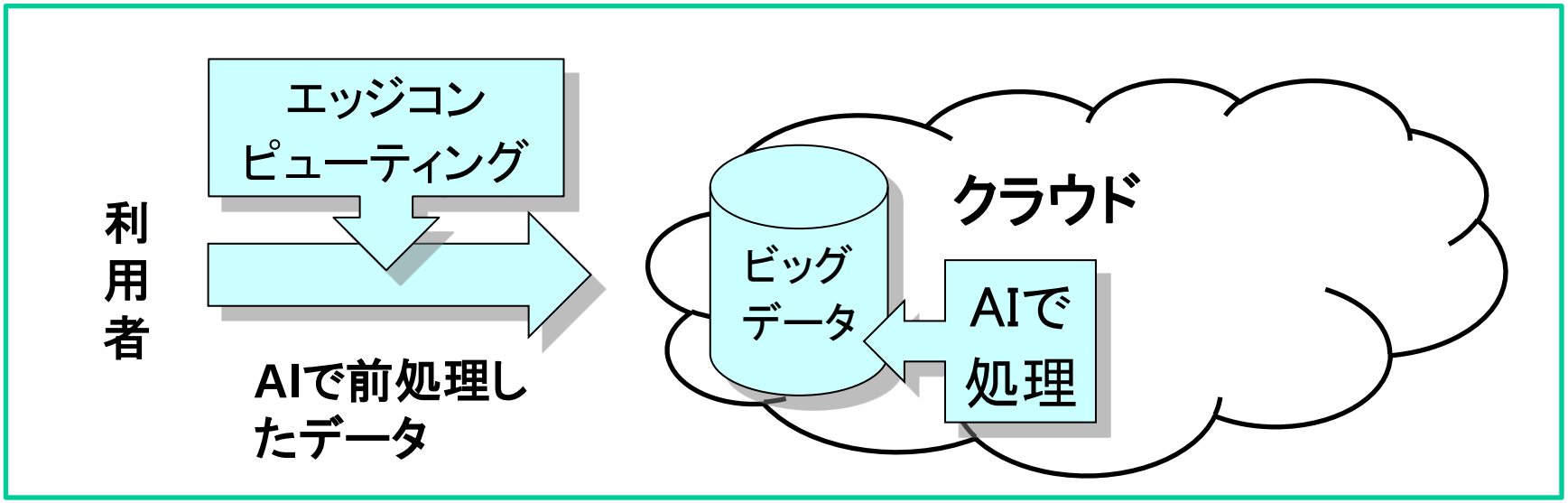
概要



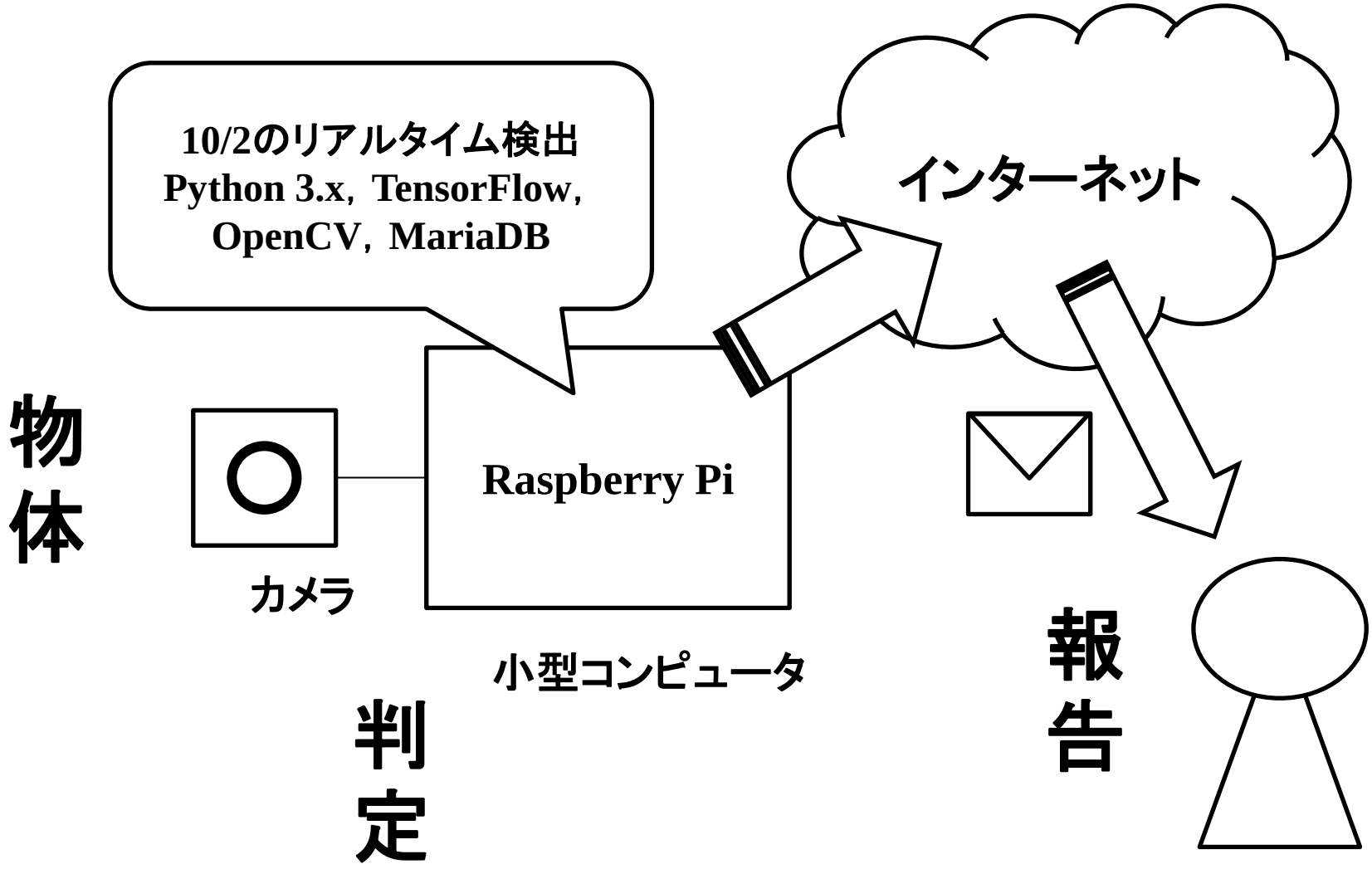
内閣府　Society 5.0　https://www8.cao.go.jp/cstp/society5_0/index.html

エッジコンピューティング





カメラ＋小型コンピュータ



機器

Raspberry Pi 4 Model B セット内容

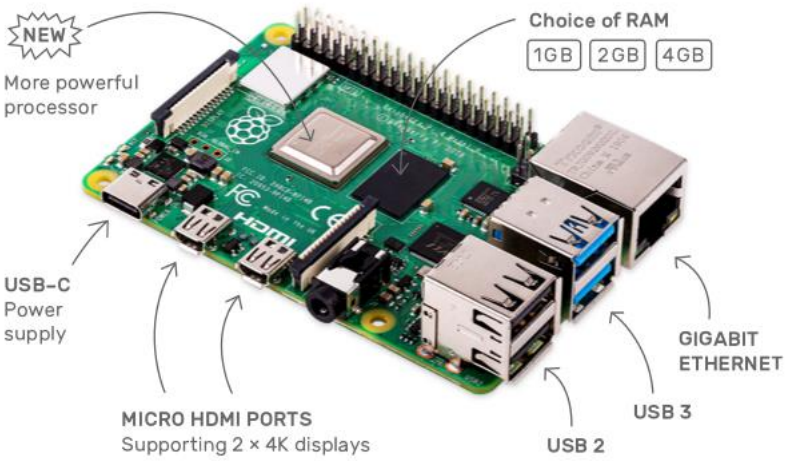
- ラズベリーパイ4B（4GB RAM）
  - セットによってはラズベリーパイ4B用ケース付き
- microSDカード（32GB）
  - 必要があればBackup用に別のカードを準備
- カードリーダー
- 放熱対策のためのヒートシンク、ファン
- 5.1V3.0A USB Type-C電源アダプタ
  - セットによってはスイッチ付き電源ケーブル付き
- MicroHDMI-to-HDMIケーブル
- その他（LANケーブル、プラスドライバなど）

セット以外

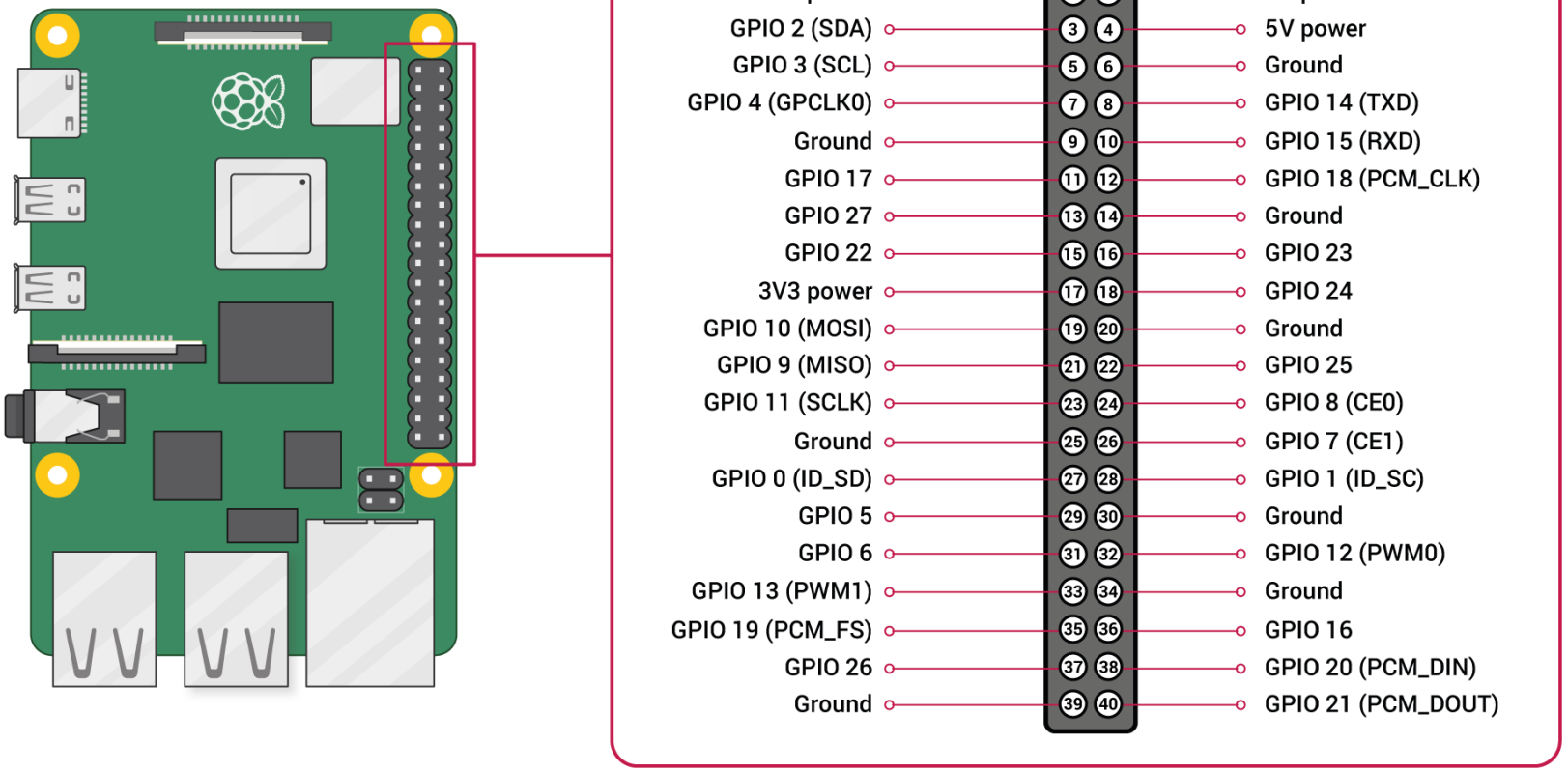
モニタ（HDMI）Camera Module V2、キーボード、マウス

Raspberry Pi 4 Model B

通常版：Model Aシリーズ（正方形）
　　Model Bシリーズ（長方形，高スペック，標準）
小型版：Zeroシリーズ
一般向け：Compute Moduleシリーズ
産業向け：第1世代～第4世代
Raspberry Pi 4は第4世代



PINOUT



https://www.raspberrypi.org/documentation/usage/gpio/

Raspberry Pi 4 Model B

発売日：2019年6月24日（2019年9月電波法認証）
メモリ（GPUと共用）：1GB($35), 2GB($45), 4GB ($55)
CPU：ARM Cortex-A72（1.5GHz）
GPU：Broadcom VideoCore VI（Dual Core 500MHz, OpenGL ES 3.0）
USBポート：2.0×2，3.0×2
映像入力：15ピンMIPIカメラインターフェース
映像出力：コンポジット RCA, micro-HDMI (up to 4kp60) x 2 2.0 , MIPI DSI
音声出力：3.5 mm ジャック, micro-HDMI, I²S
ストレージ：microSDカード
有線ネットワーク：Gigabit Ethernet
無線ネットワーク：2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 5.0, BLE
拡張コネクタ：GPIO 40 ピン
電源：3A(15W), USB Type-C, GPIO, Power over Ethernet

演習：10/2に実施したリアルタイム検出をRaspberry Pi上で実現＋MariaDBに保存＋E-mailで送信

詳細は別紙の演習資料

10/2の振り返り

リアルタイム検出

Raspberry Piのカメラを使用して、リアルタイムであるターゲットを認識させる

必要なもの：

- 対象画像（学習用、評価用）
- 背景画像（学習用、評価用）
- データセットを記述したcsvファイル

かなりの部分はこちらで用意してありますが（料理番組方式）、本当はこれらを全部自分で用意します。ぜひ1度自分でチャレンジしてください

10/2の振り返り

リアルタイム検出の手順

次の順番で行う

1. 対象物を撮影する
2. 背景画像を撮影する
3. 対象物画像を水増しして300枚用意する
4. 背景画像を分割して625枚用意する
5. 対象物画像と背景画像をcsvファイルに記述する
6. 用意したデータで学習を行う
7. 学習されたデータを用いて、リアルタイムで判定させる

対象物の撮影

対象物をカメラアプリで撮影し、128x128のサイズで切り出す
正方形で切り出すため、なるべく縦横比が極端でないものを選ぶ
顔は意外と認識されにくい（頑張って下さい）
手のひらは割と簡単。スマホケース、ペットボトルの模様、社員証など
ファイル名はtarget.jpgとする。（詳しい手順は、この後の対象画像作成手順を参照）

今度は、同じ位置で背景のみで撮影し、ファイル名をother.jpgとする。サイズ等は変更しない

10/2の振り返り

撮影

対象物が入った写真を撮影 →　対象物だけ切り抜いてtarget.jpg
次に対象物がない写真を撮影　→　other.jpg

認識させたい対象物

自分が映り込まないようにして撮影

10/2の振り返り

指定サイズでの切り出し(1)

正方形での切り出しが可能なXnViewを使用する

https://forest.watch.impress.co.jp/library/software/xnview/ などからダウンロード（XnViewで検索すると窓の杜が最初にヒット）

もちろん、オフィシャルサイトからDLしてもよい。
https://www.xnview.com/

XnViewを起動し、カメラアプリで撮影した対象物のファイルを読み込む
もし画像が鏡映しになっていたら、画像 → 反転 → 左右反転

10/2の振り返り

指定サイズでの切り出し(2)

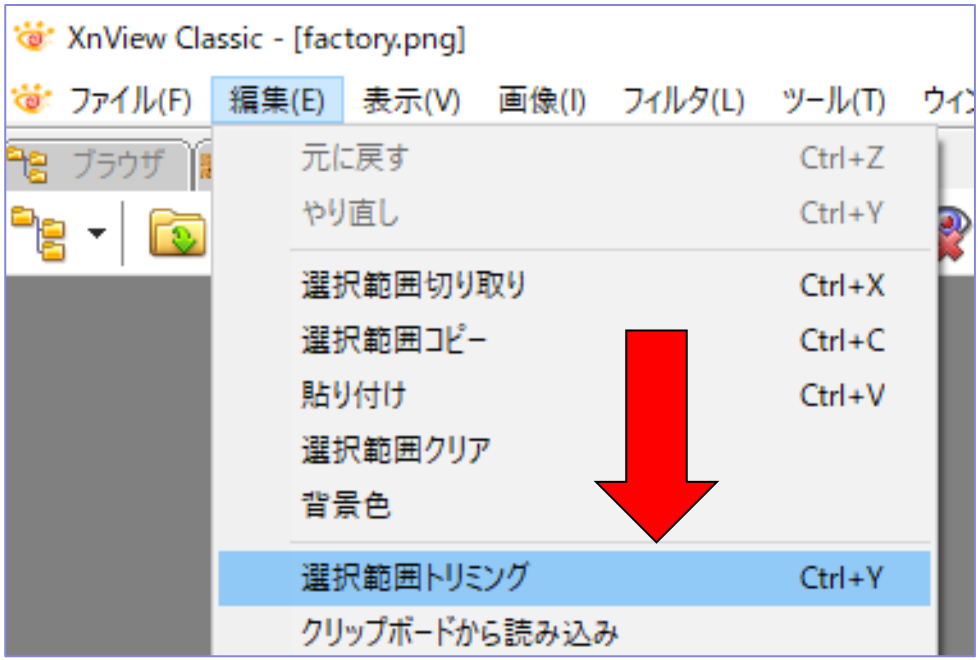
編集 → 選択範囲縦横率設定 → 1:1 (1.00) を選択
これで正方形で範囲選択が可能

10/2の振り返り

指定サイズでの切り出し(3)

任意の範囲を選択後、編集 → 選択範囲トリミング を選択し、切り抜く

切り抜いた後、画像 → リサイズ、でサイズ変更画面が開くので、縦横128ピクセルを指定してサイズ変更し、target.jpg という名前で保存



CSVファイル

こちらで準備済み

- target_or_other_train.csv　：学習（訓練用）ファイルパスとラベル
- target_or_other_test.csv　：検証用ファイルパスとラベル

それぞれのファイル内に、各画像のパスとラベルがカンマ区切りで書かれている

エディタやExcelなどで1度中味をチェックすること

水増しと学習

Colab: 08-Augmentation_Learning.ipynb を開く

VM内にtarget.jpgとother.jpg, csvファイルをコピーし、実行する

先程のcsvファイルと、フォルダtrain_target, test_targetとの対応、実際のファイルの存在をチェックしておく

学習後、重みファイル(detection_weight.h5)をダウンロードし、後は実機で

ネットワークはある程度しか書いていないので、認識できるように、拡張して下さい

重みファイルはネットワークごとに名前を変えておいて、比較するとよい

結果

緑の枠内にターゲットを持っていく
無事に認識されると赤枠になる
どうにもダメなら、ネットワークやエポック数など変えてみる

Extra演習

extraA）TensorFlowを使った物体認証の復習1
各自でリアルタイム検出を行うことができるか、新しい物体を選んで各自で実施してみましょう。

extraB）TensorFlowを使った物体認証の復習2
microSDカードにraspberry pi OSのイメージを書き込んで、各自でインストールの最初からセットアップができるか実施してみましょう。

extraC）TensorFlowを使った物体認証（VGG-16版1）
10/02に実施した「VGG-16演習」を各自でラズベリーパイ上に再現してみましょう。

extraD）TensorFlowを使った物体認証（VGG-16版2）
10/02に実施した「VGG-16演習」の機能を今回実施したプログラムに組み込んでみましょう。※リアルタイム検出の中身をVGG-16に設定する

第16章

組み込み

－後半－

目的と目標

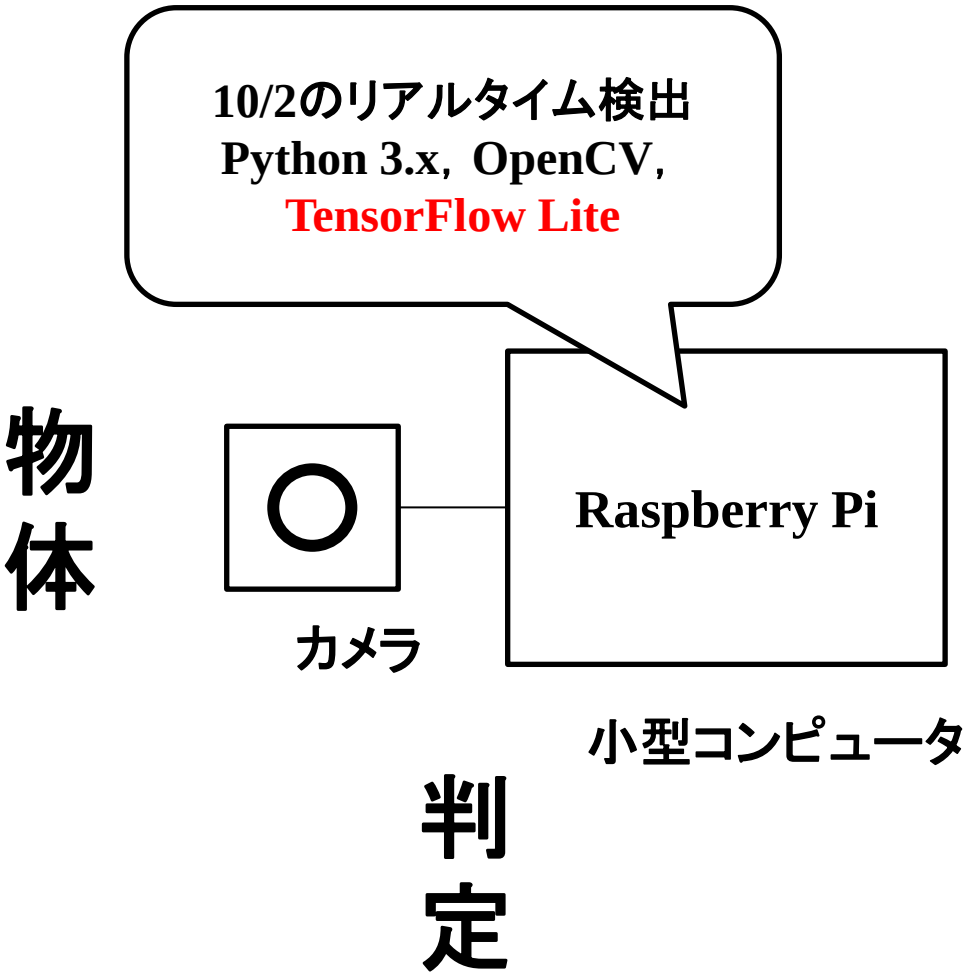
目的：機械学習を搭載するエッジコンピューティングへの理解（２）

- Raspberry Pi上にTensorFlow Lite環境を構築できる
- Google colabで作成した学習結果（重みファイル）をTensorFlow Liteに変換できる
- TensorFlow Lite モデルメーカーを使ってモデルを作成できる

目標：Raspberry Pi上でのリアルタイム検出の実現（Tensorflow Lite版）

- 演習：10/2に実施したリアルタイム検出をTensorflow Liteを使ってRaspberry Pi上で実現

カメラ＋小型コンピュータ



演習：10/2に実施したリアルタイム検出をTensorFlow Liteを使ってRaspberry Pi上で実現

詳細は別紙の演習資料

TensorFlow Lite

モバイル デバイス、組み込みデバイス、IoT デバイスで TensorFlow モデルを実行できるようにするツールセット

TensorFlow Lite インタープリタ:

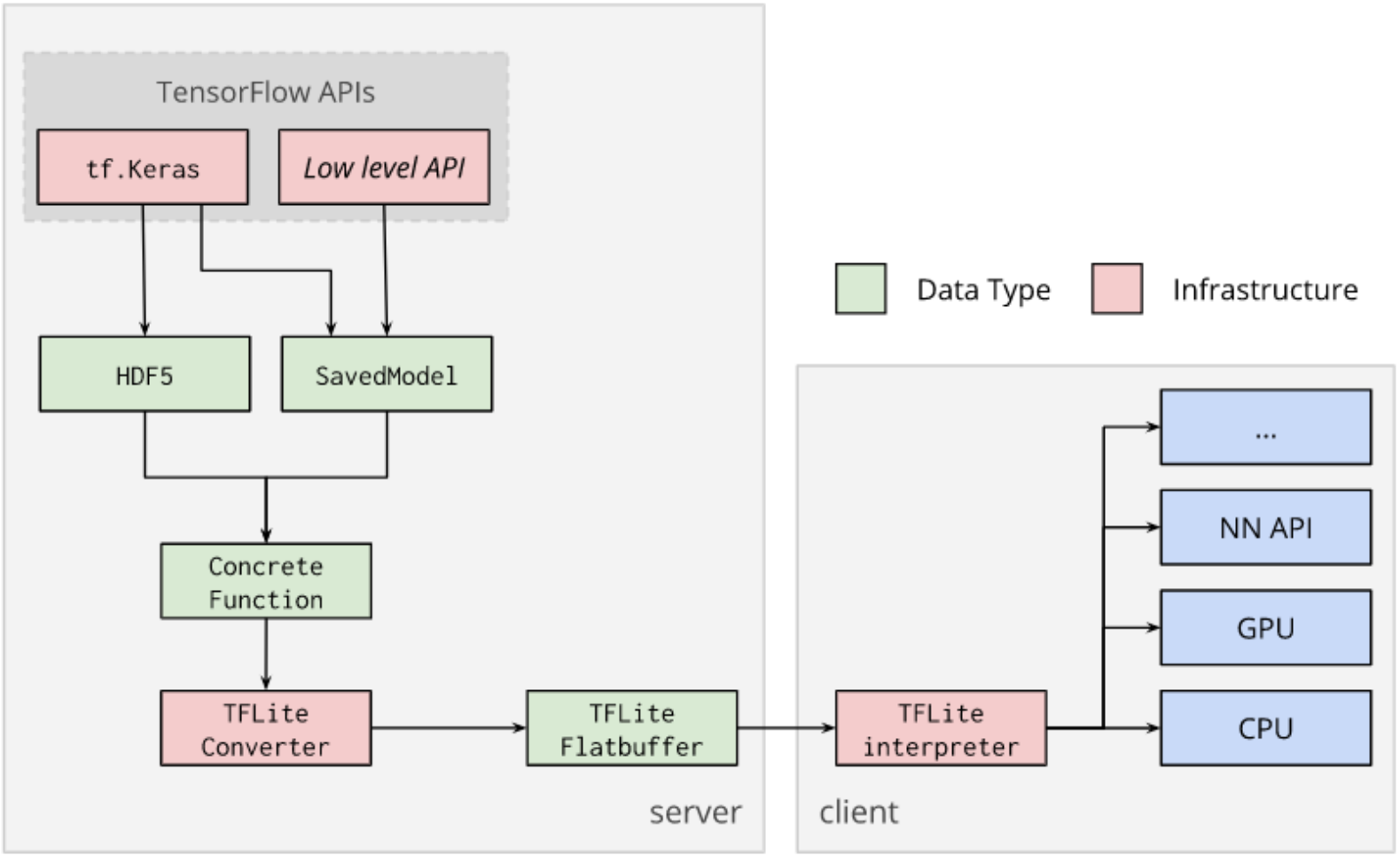
スマートフォン、組み込み Linux デバイス、マイクロコントローラなどのさまざまなハードウェア上で最適化されたモデルを実行できるようになるインタープリタ

TensorFlow Lite コンバータ:

TensorFlow モデルをインタープリタで使用するための効率的な形式に変換し、最適化を行うことができるコンバータ

https://www.tensorflow.org/lite?hl=ja

TensorFlow Lite



https://www.tensorflow.org/lite/convert?hl=ja

TensorFlow Liteモデルメーカー

概観

TensorFlow Liteモデルメーカーライブラリは、カスタムデータセットを使用してTensorFlow Liteモデルをトレーニングするプロセスを簡素化します。転移学習を使用して、必要なトレーニングデータの量を減らし、トレーニング時間を短縮します。

サポートされているタスク

モデルメーカーライブラリは現在、次のMLタスクをサポートしています。

サポートされているタスク	タスクユーティリティ
画像分類ガイド	画像を事前定義されたカテゴリに分類します。
テキスト分類ガイド	テキストを事前定義されたカテゴリに分類します。
質問回答ガイド	与えられた質問に対する特定のコンテキストで答えを見つけます

https://www.tensorflow.org/lite/guide/model_maker?hl=ja

転移学習

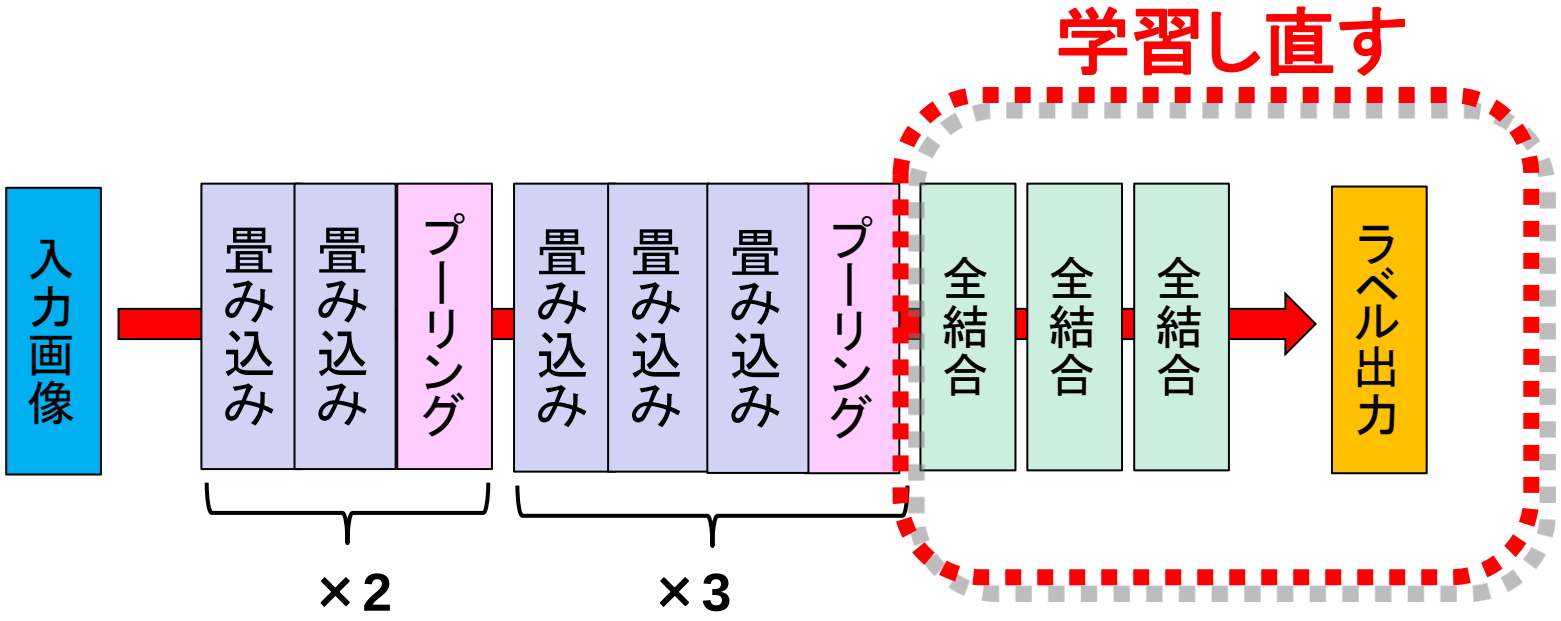
高度な深層学習モデルには大量のパラメータ（重み）があり、それらを何もないところから訓練するには大量の計算リソースのデータが必要

転移学習は関連するタスクについてすでに学習済みのモデルの一部を取り出し、新しいモデルの中で再利用することで、そのようなリソースの大部分を省略するテクニック

https://www.tensorflow.org/js/tutorials/transfer/what_is_transfer_learning?hl=ja

転移学習

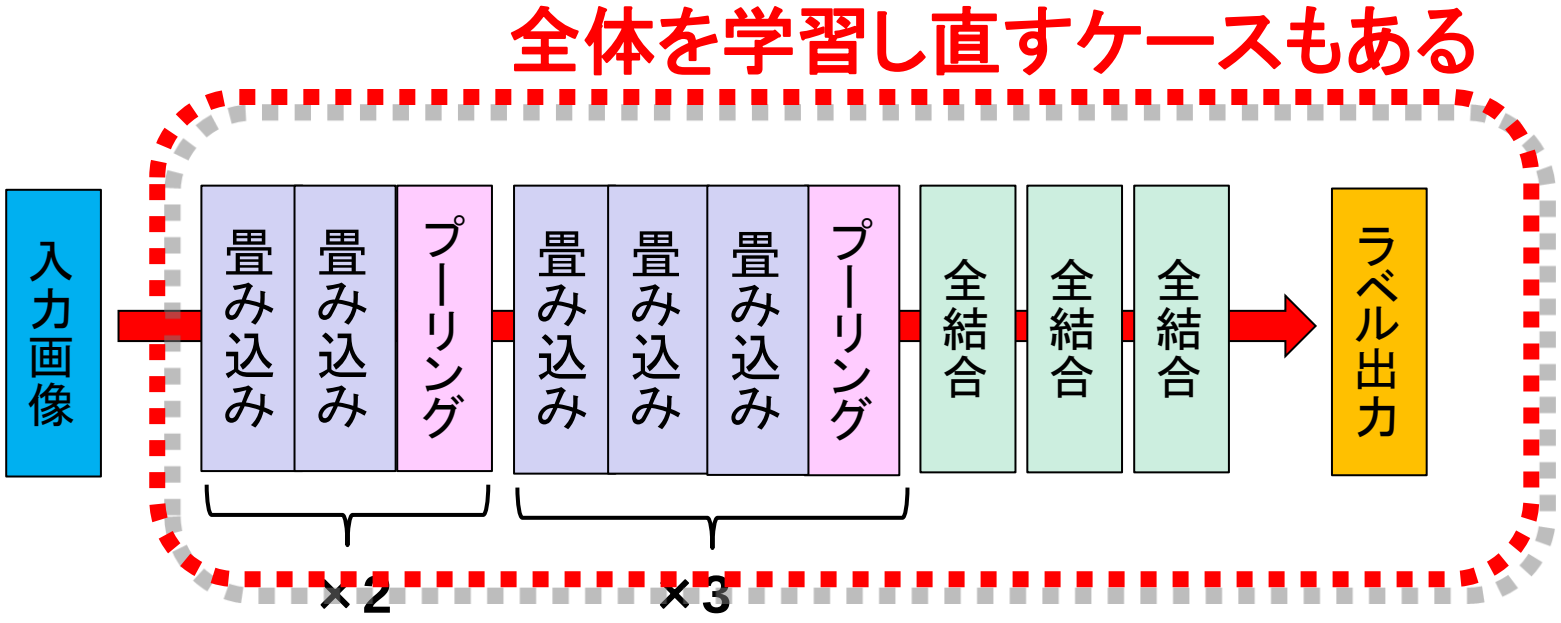
VGG16



Karen Simonyan, Andrew Zisserman, "**Very Deep Convolutional Networks for Large-Scale Image Recognition**", https://arxiv.org/abs/1409.1556, (2014)

転移学習

VGG16



Karen Simonyan, Andrew Zisserman, "**Very Deep Convolutional Networks for Large-Scale Image Recognition**", https://arxiv.org/abs/1409.1556, (2014)

CS231n Convolutional Neural Networks for Visual Recognition

When and how to fine-tune? How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset (small or big), and its similarity to the original dataset (e.g. ImageNet-like in terms of the content of images and the classes, or very different, such as microscope images). Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, here are some common rules of thumb for navigating the 4 major scenarios:

1. *New dataset is small and similar to original dataset.* Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns. Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the CNN codes.
2. *New dataset is large and similar to the original dataset.* Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.
3. *New dataset is small but very different from the original dataset.* Since the data is small, it is likely best to only train a linear classifier. Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features. Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.
4. *New dataset is large and very different from the original dataset.* Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch. However, in practice it is very often still beneficial to initialize with weights from a pretrained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

https://cs231n.github.io/transfer-learning/

いつどのようにFine-Turningするか？

新しいデータセットの「大きさ」と「類似度」が重要

１）新しいデータセットが小さい・元のデータセットに似ている
過学習が懸念される。最後の層の分類器を訓練するのがお勧め。

２）新しいデータセットが大きい・元のデータセットに似ている
全体をFine-Turningしても過学習しない可能性が高い。

３）新しいデータセットが小さい・元のデータセットに似ていない
分類器を訓練するのが良い。ネットワークの出力側よりも入力側のどこかの特徴量を使って訓練するのが良い。

４）新しいデータセットが大きい・元のデータセットに似ていない
ゼロから訓練しても大丈夫。訓練済みモデルの重みでネットワークを初期化すると有効であることが多い。

演習：targetを複数にしたモデルの作成

プログラム15d_TFL_ModelMaker.ipynbをもとに、targetを複数にして検出するモデルを各自で作成し、15e_detecRoi_mMaker.pyで動作するか確認してみましょう。

なお、フォルダが異なっていれば画像の名前は同じものがあっても特に問題ありません。

例えば、target1/target.1.jpgとtarget2/target.1.jpgは画像名が同じですが、フォルダが異なるので特に問題ありません。

（前回）Extra演習

extraA）TensorFlowを使った物体認証の復習1

各自でリアルタイム検出を行うことができるか、新しい物体を選んで各自で実施してみましょう

extraB）TensorFlowを使った物体認証の復習2

microSDカードにraspberry pi OSのイメージを書き込んで、各自でインストールの最初からセットアップができるか実施してみましょう

extraC）TensorFlowを使った物体認証（VGG-16版1）

10/02に実施した「VGG-16演習」を各自でラズベリーパイ上に再現してみましょう

extraD）TensorFlowを使った物体認証（VGG-16版2）

10/02に実施した「VGG-16演習」の機能を今回実施したプログラムに組み込んでみましょう。※リアルタイム検出の中身をVGG-16に設定

最終課題に向けて

これまでの内容を活かした、各自のアイディアによる自由製作です

例）

- 画像判定、動画判定器（非リアルタイムでもよい）
- 組み込み機を利用した何かリアルタイム判定機、あるいはWebサービス
- 自然言語処理を応用したネガティブ、ポジティブ判断、bot、類似判定
  （BERTは学習時間が大変かも。Word2vecによるコサイン類似など）

また、最終課題の制作物に対して、サービスの概要・仕様書を書いて下さい。
外部仕様、内部仕様、REST APIなどのしっかりとしたものではなく、他者がさっと目を通して概略が理解できる範囲の物でかまいません

第17章

最終課題, まとめ

最終課題

これまでの内容を活かした、各自のアイディアによる自由製作です

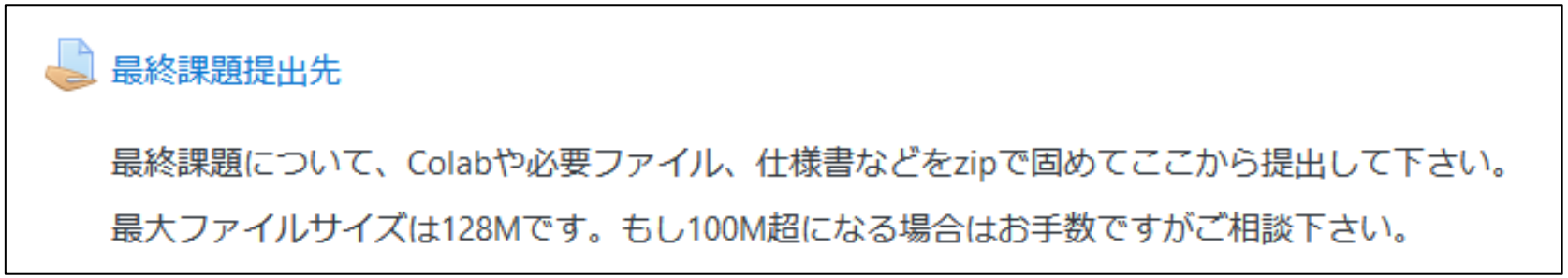
例）

- 画像判定、動画判定器（非リアルタイムでもよい）
- 組み込み機を利用した何かリアルタイム判定機、あるいはWebサービス
- 自然言語処理を応用したネガティブ、ポジティブ判断、bot、類似判定
  （BERTは学習時間が大変かも。Word2vecによるコサイン類似など）

また、最終課題の制作物に対して、サービスの概要・仕様書を書いて下さい。
外部仕様、内部仕様、REST APIなどのしっかりとしたものではなく、他者がさっと目を通して概略が理解できる範囲の物でかまいません

提出先

Moodleの「最終課題提出先」から提出してください
締切は11月末ですが、対象遅れても大丈夫です
プログラム(Colab)、簡単な仕様書などをzipで固めて提出して下さい



もし、考えていたことが実現できなかった場合は途中段階でもかまいません。
「××ということを考えていたが、○○までしかできなかった」という説明を添えて下さい
12月中に提出された方全員に何らかのコメントを返す予定です

この講座を通して

AI概要から古典的な機械学習、さらにCNN, LSTM, データ処理, 自然言語処理と、典型的なAI講座として一通り学びました

ここではGoogle ColabとTensorFlowという組み合わせで実施してきましたが、PyTorchの人気が最近特に向上中です。ひょっとしたらこの先、スタンダードなものはPyTorchになるかもしれません

すでに皆さんはわかっていると思うのですが、AIは万能ではないですし、また何でもAIで解決できるわけではありません。得手・不得手がありますし、適材適所です

年間を通じて、東京・大阪ではAI展示会が何度かありますので、ぜひ足を運んで下さい。そこで、「本当は何をやっているのか」と敢えて批判的な目で見て下さい