AIは幸せにするか？
コンデンサーの性能を判別できるか？

山形大学　大学院理工学研究科
伊藤智博
tomohiro@yamagata-u.ac.jp

2018/3/16 14:00～ 山形大学工学部百周年記念会館

一般向けコンピュータの性能向上

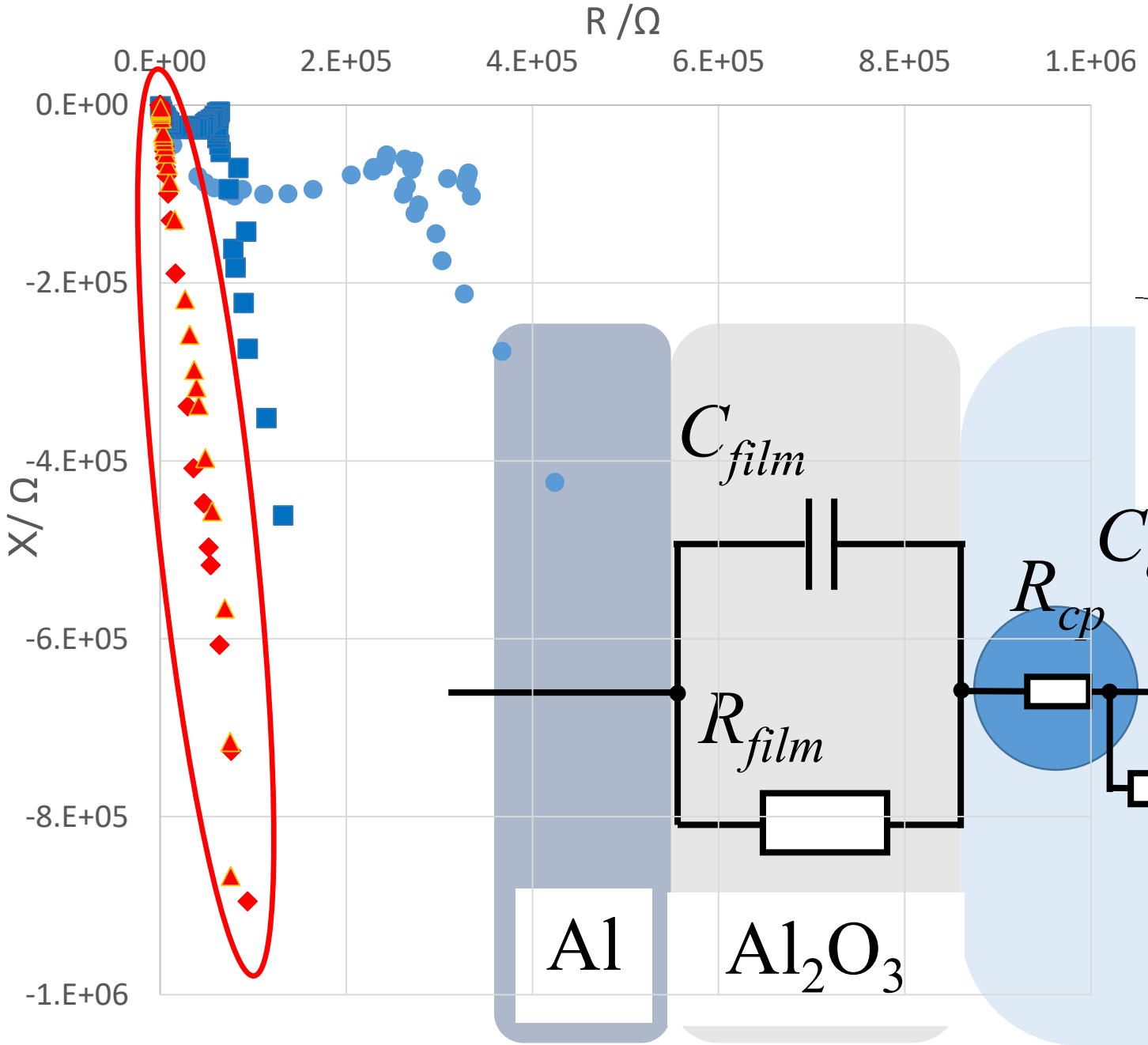
年代	メモリ	価格／万円	CPU
1995	256MB	500	Super SPARC, Single Core
1997	1GB	500	Ultra SPARC, Single Core
2007	32GB	500	Opteron, Dual Core
2014	768GB	1400	Xeon, 10 core
2018	1024GB	1000	EPYC,32 core

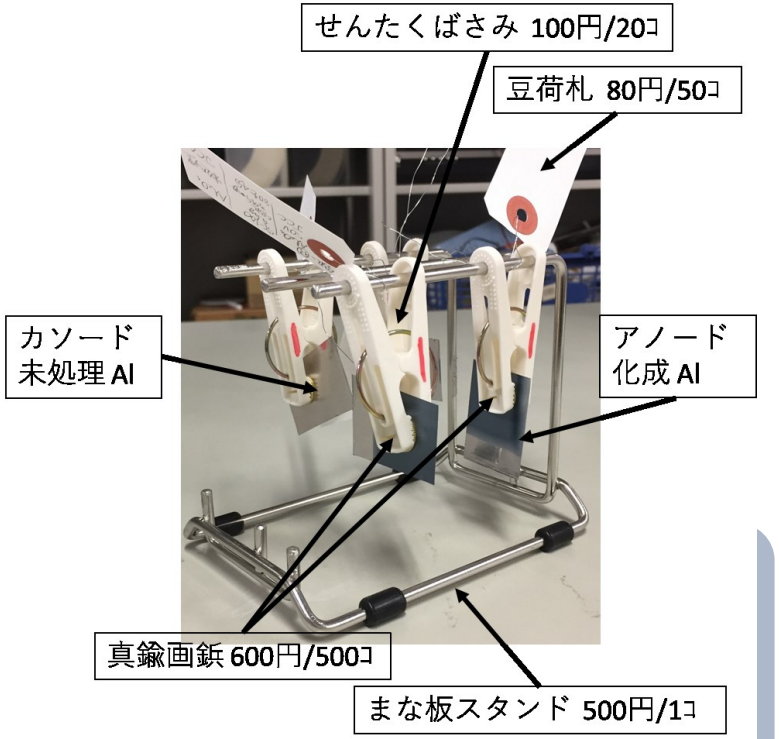
機械学習用のライブラリーの充実

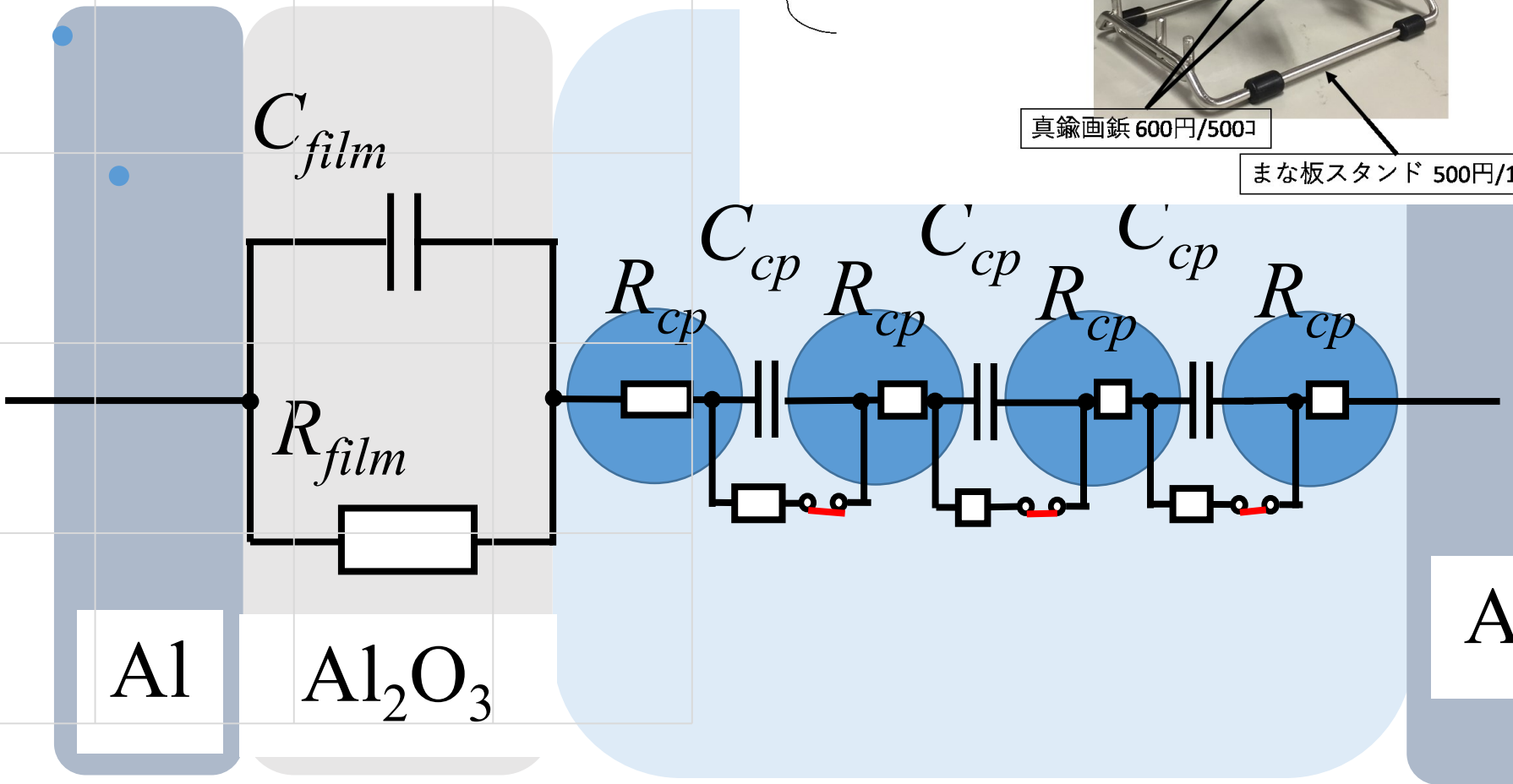
名前	言語	
tensorflow	Python, C++, Java	https://www.tensorflow.org/
Caffe	C++,Python, MATLAB	http://caffe.berkeleyvision.org/
Chainer	Python	http://chainer.org/
Theano	Python(パイソン)	http://deeplearning.net/software/theano/index.html

無償で利用できます.

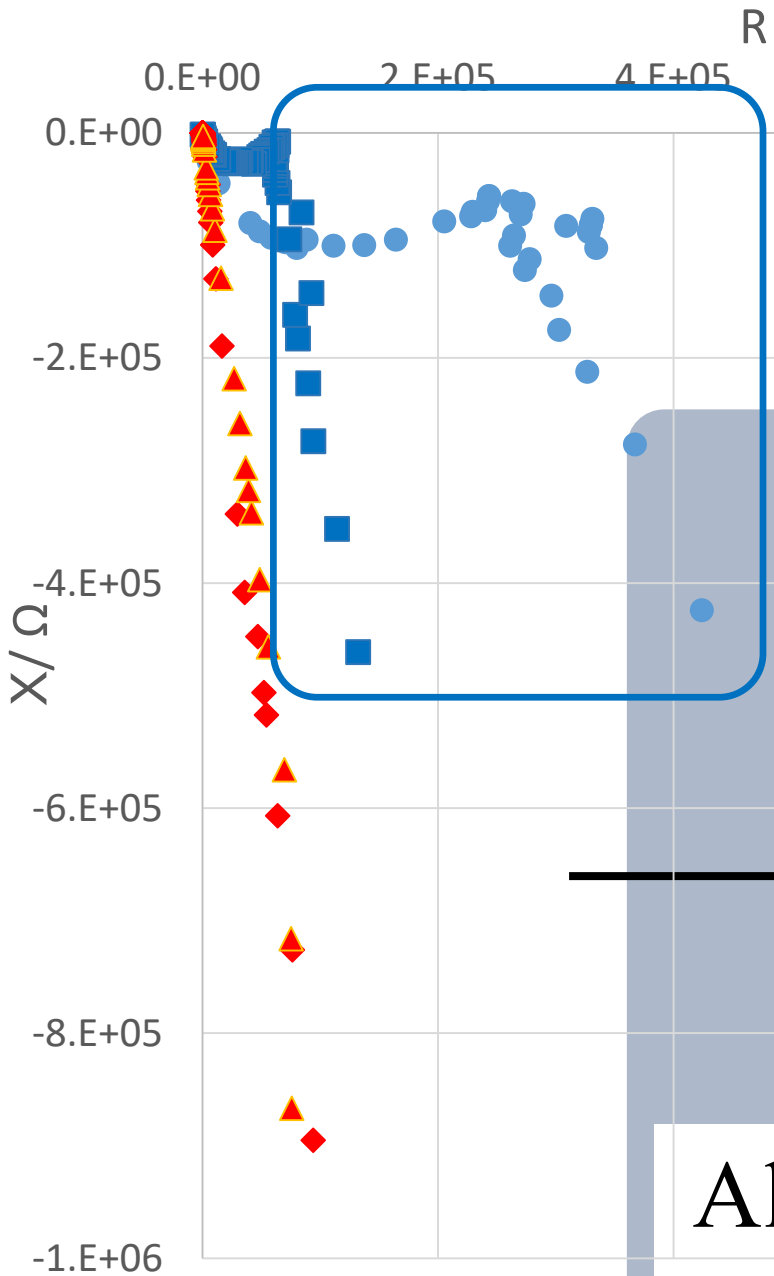
高周波特性に優れた電解コンデンサー

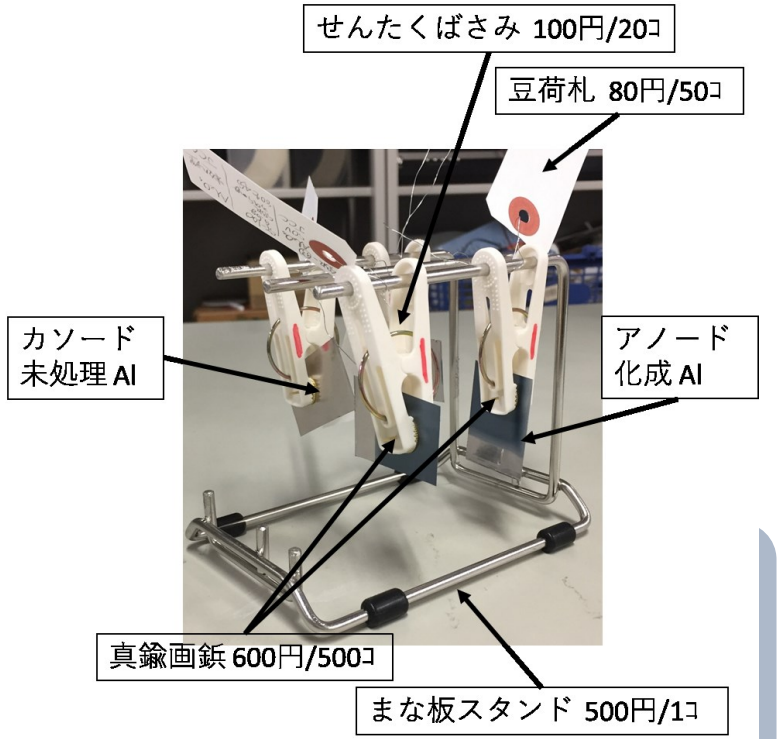


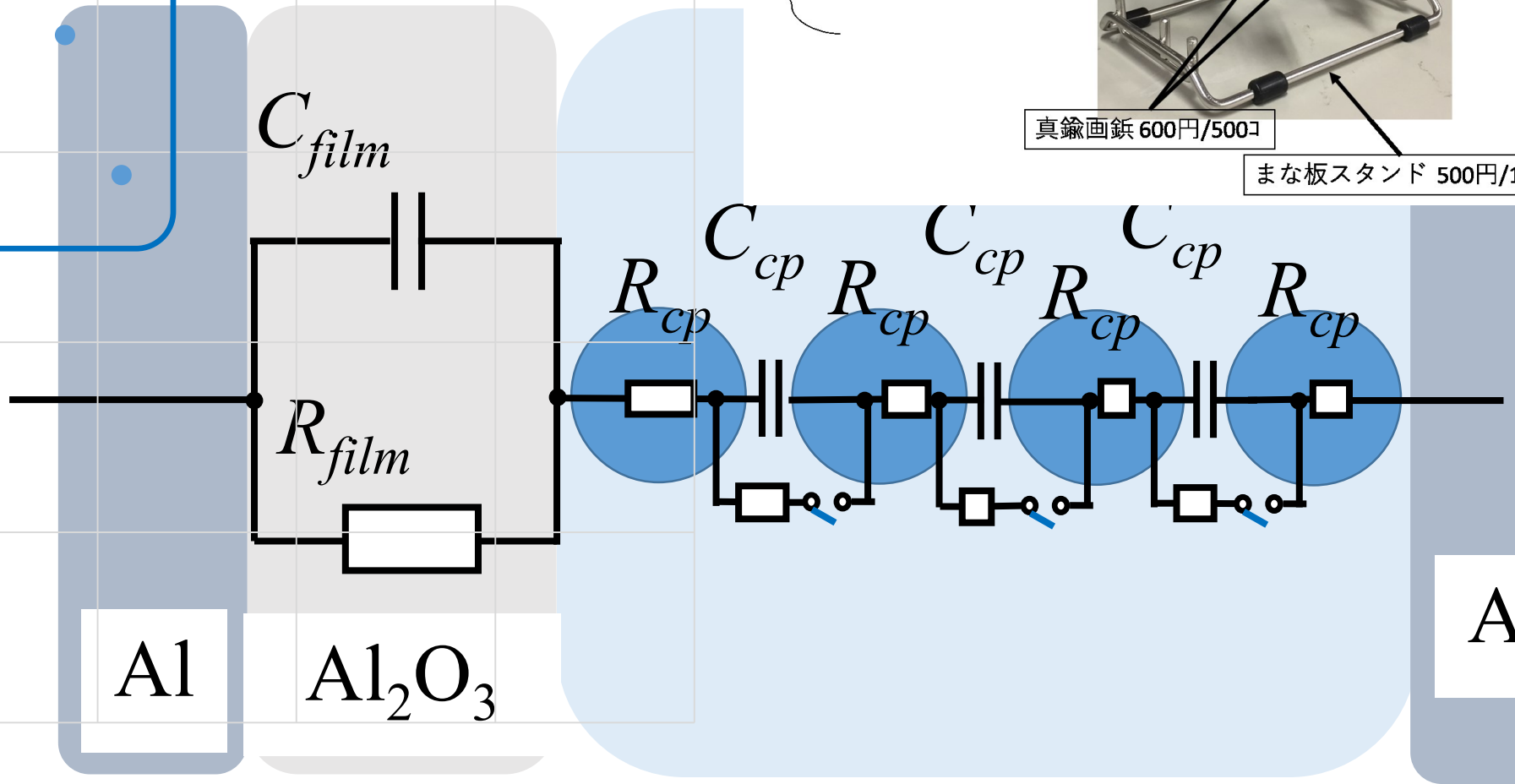




高周波特性に劣る電解コンデンサー







機械学習

教師データ

Bad　Good

0　1

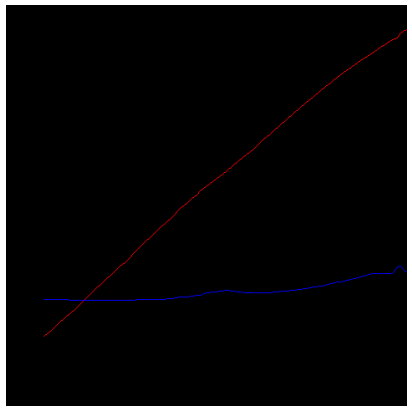
1　0

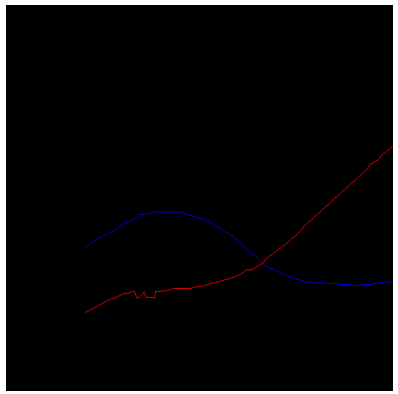
y (2行2列)

=

学習データ

480*480*3=691200





x
(2行691200列)

機械学習で赤線内の***w***と***b***の行列を近似する.

Bad　Good

480*480*3=691200

W(691200行2列)

\+

Bad　Good

b_1　b_2

b_1　b_2

b (2行2列)
(ブロードキャストルール)

機械学習

教師データ

Bad　　Good

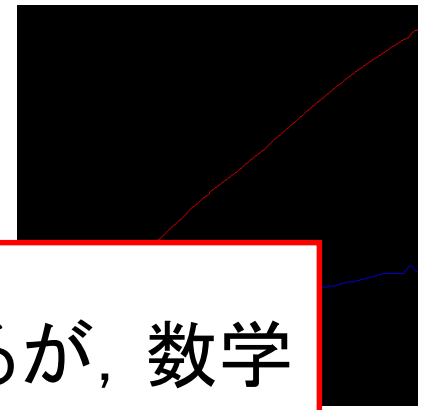
0　　1

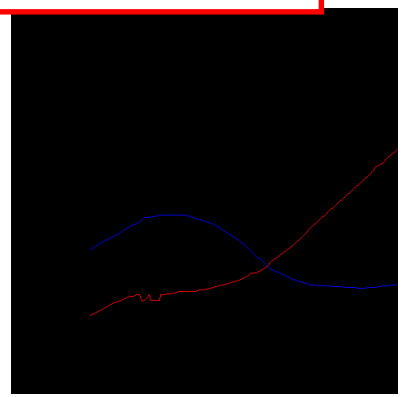
1　　0

y (2行2列)

学習データ

480*480*3=691200





x
(2行691200列)

機械学習で赤線内の***w***と***b***の行列を近似する.

Bad　　Good

480*480*3=691200

W(691200行2列)

\+

Bad　Good

b_1　b_2

b_1　b_2

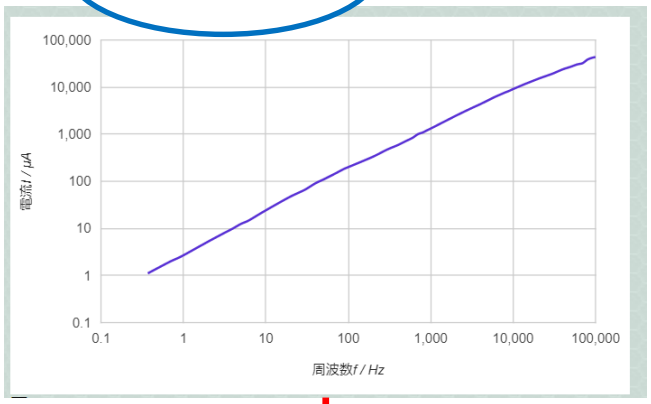
b (2行2列)

AIとか，機械学習といっているが，数学的には，トライアンドエラーでW行列とb行列を決めているだけ！

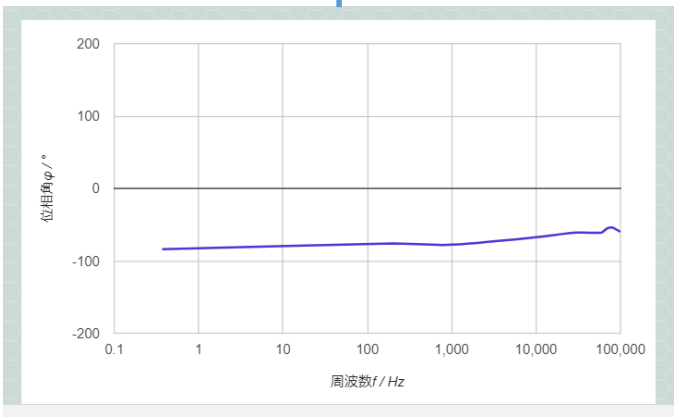
教師データと学習データの規格化

学習データ

Good

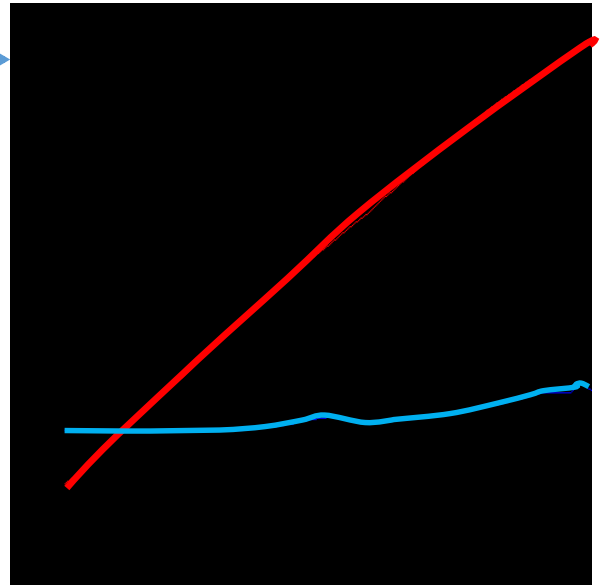


＋



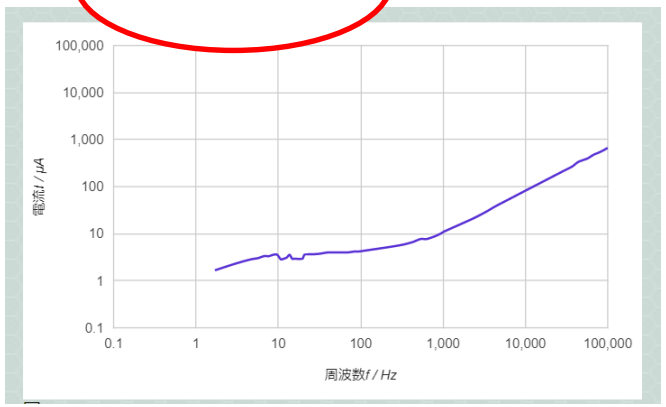
B(255)で画像にプロット

画像に変換

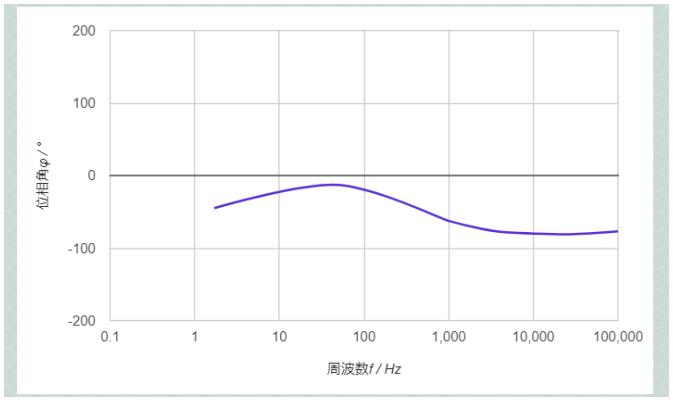


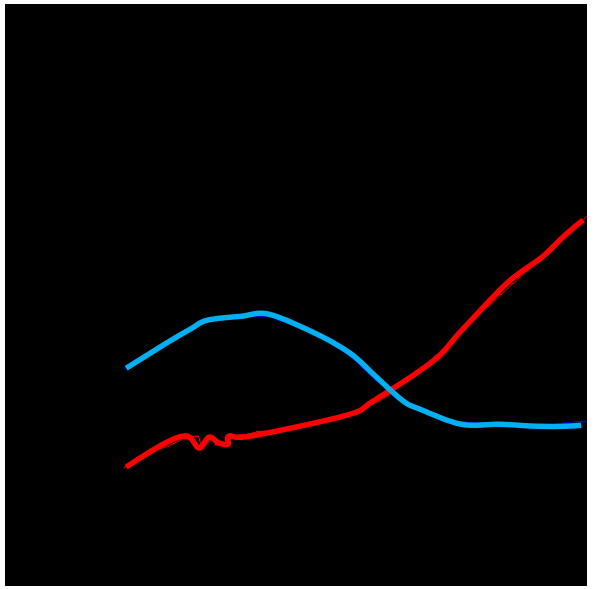
R(255)で画像にプロット

Bad



＋





機械学習

教師データ

Bad　　Good

0　　1

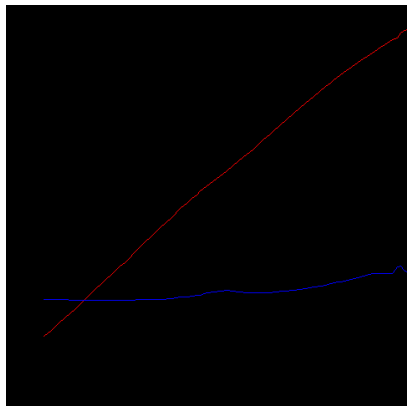
1　　0

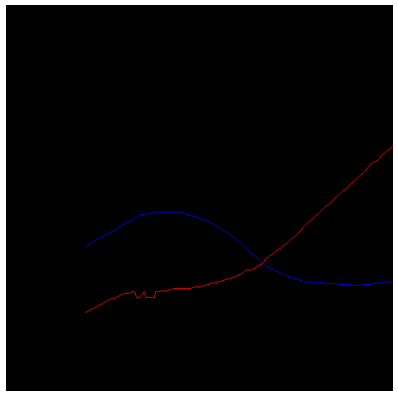
y (2行2列)

=

学習データ

480*480*3=691200





x
(2行691200列)

機械学習で赤線内の***w***と***b***の行列を近似する.

Bad　　Good

480*480*3=691200

W(691200行2列)

\+

Bad　Good

b_1　b_2

b_1　b_2

b (2行2列)

機械学習-ソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp001.csv"

import tensorflow as tf
import numpy as np
import requests
import os

## 定数
IMG_SIZE = 480 ## 学習させる画像の縦幅・横幅
IMG_LENGTH = IMG_SIZE * IMG_SIZE * 3 ## 学習させる画像データ長
LABEL_CNT = 2 ## ラベルの種類の数

## 学習に必要な変数の初期化
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_entropy)

## CSVファイルをワークキューとして設定
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])

myconfig = tf.ConfigProto(
   intra_op_parallelism_threads=16 )

saver = tf.train.Saver()
sess = tf.Session(config=myconfig)
sess.run(tf.global_variables_initializer())
print(os.path.exists(ckptdir))
if os.path.exists(ckptdir)==True:
   saver.restore(sess,ckptfile)

## バッチ処理の準備
batch_url, batch_label = tf.train.batch([url, label], batch_size=2)
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)
image2 = []
label2 = []

urls, labels = sess.run([batch_url, batch_label])

for url in urls :
        r=requests.get(url)
        image = r.content

        ## 画像をTensorFlowで処理できるように変換
        image = tf.image.decode_jpeg(image, channels=3)
        image = tf.image.resize_image_with_crop_or_pad(image, IMG_SIZE, IMG_SIZE)
        image = tf.reshape(image, [-1])
        image_val = sess.run(image).astype(np.float32) / 255.0
        image2.append(image_val)

for label in labels :
        tmp = np.zeros(LABEL_CNT)
        tmp[label] = 1
        label2.append(tmp)
image2 = np.asarray(image2)
lable2 = np.asarray(label2)
image2_len = len(image2)
```

コードの長さは，50～100行

```
 sess.run(accuracy, feed_dict={x: image2, y_: label2})
 print('Step %d, Loss %f' % (i, loss))
if os.path.exists(ckptdir)==False:
   os.makedirs(ckptdir)
saver.save(sess,ckptfile)
sess.close()
```

機械学習-ソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp001.csv"

import tensorflow as tf
import numpy as np
import requests
import os

## 定数
IMG_SIZE = 480 ## 学習させる画像の縦幅・横幅
IMG_LENGTH = IMG_SIZE * IMG_SIZE * 3 ## 学習させる画像デ
LABEL_CNT = 2 ## ラベルの種類の数

## 学習に必要な変数の初期化
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(

## CSVファイルをワークキューとして設定
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])

myconfig = tf.ConfigProto(
   intra_op_parallelism_threads=16 )

saver = tf.train.Saver()
sess = tf.Session(config=myconfig)
sess.run(tf.global_variables_initializer())
print(os.path.exists(ckptdir))
```

```
# 学習データを格納する行列 xを定義
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
#学習結果を格納する行列 W, bを定義
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
# y = x・W + b　である計算式を定義
y = tf.nn.softmax(tf.matmul(x, W) + b)
#教師データである結果を格納する変数 y_を定義
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
#∑((y_n − y_n)^2)を計算する
#GradientDescentOptimizer = 最急降下法(Gradient Descent)でWを最適化
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_entropy)

Ref: tensorflow Math Function
tf.square: https://www.tensorflow.org/api_docs/python/tf/square
tf.reduce_sum : https://www.tensorflow.org/api_docs/python/tf/reduce_sum
```

```
if os.path.exists(ckptdir)==True:
   saver.restore(sess,ckptfile)

## バッチ処理の準備
 correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
 sess.run(accuracy, feed_dict={x: image2, y_: label2})
 print('Step %d, Loss %f' % (i, loss))
if os.path.exists(ckptdir)==False:
   os.makedirs(ckptdir)
saver.save(sess,ckptfile)
sess.close()
```

機械学習-ソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp001.csv"

import tensorflow as tf
import numpy as np
import requests
import os

## 定数
IMG_SIZE = 480 ## 学習させる画像の縦幅・横幅
IMG_LENGTH = IMG_SIZE * IMG_SIZE * 3 ## 学習させる画像データ長
LABEL_CNT = 2 ## ラベルの種類の数

## 学習に必要な変数の初期化
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_en

## CSVファイルをワークキューとして設定
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])

myconfig = tf.ConfigProto(
   intra_op_parallelism_threads=16 )

saver = tf.train.Saver()
sess = tf.Session(config=myconfig)
sess.run(tf.global_variables_initializer())
print(os.path.exists(ckptdir))
```

```
if os.path.exists(ckptdir)==True:
    saver.restore(sess,ckptfile)
```

```
# 学習データを格納する行列 xを定義
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
#学習結果を格納する行列 W, bを定義
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
# y = x・W + b　である計算式を定義
y = tf.nn.softmax(tf.matmul(x, W) + b)
#教師データである結果を格納する変数 y_を定義
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
#∑((y_n − y_−n)^2)を計算する
#GradientDescentOptimizer = 最急降下法(Gradient Descent)でWを最適化
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_entropy)
```

Training.Optimizerは，tensorflowでは，約10種類ある.

tf.train.Optimizer
tf.train.GradientDescentOptimizer
tf.train.AdadeltaOptimizer
tf.train.AdagradOptimizer
tf.train.AdagradDAOptimizer
tf.train.MomentumOptimizer
tf.train.AdamOptimizer
tf.train.FtrlOptimizer
tf.train.ProximalGradientDescentOptimizer
tf.train.ProximalAdagradOptimizer
tf.train.RMSPropOptimizer　Ref:　https://www.tensorflow.org/api_guides/python/train#Optimizers

機械学習-ソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp001.csv"

import tensorflow as tf
import numpy as np
import requests
import os

## 定数
IMG_SIZE = 480 ## 学習させる画像の縦幅・横幅
IMG_LENGTH = IMG_SIZE * IMG_SIZE * 3 ## 学習させる画像データ長
LABEL_CNT = 2 ## ラベルの種類の数

## 学習に必要な変数の初期化
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_entropy)

## CSVファイルをワークキューとして設定
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])

myconfig = tf.ConfigProto(
    intra_op_parallelism_threads=16 )

saver = tf.train.Saver()
sess = tf.Session(config=myconfig)
sess.run(tf.global_variables_initializer())
print(os.path.exists(ckptdir))
if os.path.exists(ckptdir)==True:
    saver.restore(sess,ckptfile)

## バッチ処理の準備
batch_url, batch_label = tf.train.batch([url, label], b
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, c
image2 = []
label2 = []

urls, labels = sess.run([batch_url, batch_label])

for url in urls :
        r=requests.get(url)
        image = r.content

        ## 画像をTensorFlowで処理できるように変換
        image = tf.image.decode_jpeg(image, channels=3)
        image = tf.image.resize_image_with_crop_or_pad(image, IMG_SIZE, IMG_SIZE)
        image = tf.reshape(image, [-1])
        image_val = sess.run(image).astype(np.float32) / 255.0
        image2.append(image_val)

for label in labels :
        tmp = np.zeros(LABEL_CNT)
        tmp[label] = 1
        label2.append(tmp)
image2 = np.asarray(image2)
lable2 = np.asarray(label2)
image2_len = len(image2)
```

ウェブページから画像を読み込む

```
r=requests.get(url)
image = r.content
```

```
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])
```

url

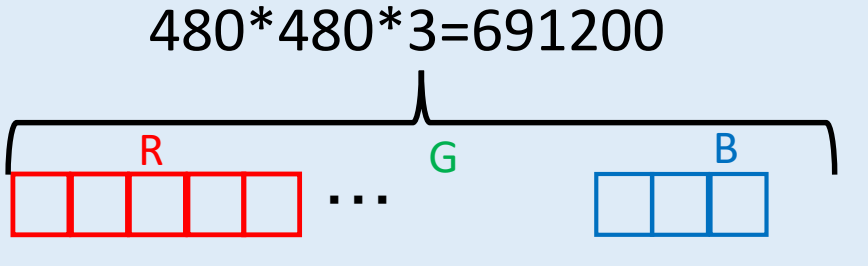
label

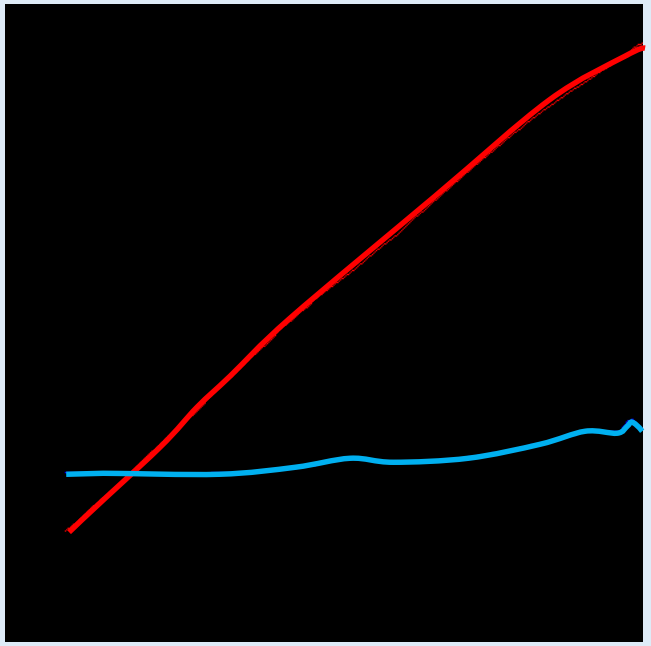
```
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_BAD.png,0
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_GOOD.png,1
```

機械学習-ソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp001.csv"

import tensorflow as tf
import numpy as np
import requests
import os
```

480*480*3=691200





```
image = tf.image.decode_jpeg(image, channels=3)
    image = tf.image.resize_image_with_crop_or_pad(image, IMG_SIZE, IMG_SIZE)
    image = tf.reshape(image, [-1])
    image_val = sess.run(image).astype(np.float32) / 255.0
    image2.append(image_val)
```

```
if os.path.exists(ckptdir)==True:
    saver.restore(sess,ckptfile)

## バッチ処理の準備
    r=requests.get(url)
    image = r.content

    ## 画像をTensorFlowで処理できるように変換
    image = tf.image.decode_jpeg(image, channels=3)
    image = tf.image.resize_image_with_crop_or_pad(image, IMG_SIZE, IMG_SIZE)
    image = tf.reshape(image, [-1])
    image_val = sess.run(image).astype(np.float32) / 255.0
    image2.append(image_val)

for label in labels :
    tmp = np.zeros(LABEL_CNT)
    tmp[label] = 1
    label2.append(tmp)
image2 = np.asarray(image2)
lable2 = np.asarray(label2)
image2_len = len(image2)
```

画像を一行行列に変換するコード

```
 sess.run(accuracy, feed_dict={x: image2, y_: label2})
 print('Step %d, Loss %f' % (i, loss))
if os.path.exists(ckptdir)==False:
    os.makedirs(ckptdir)
saver.save(sess,ckptfile)
sess.close()
```

機械学習-ソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp001.csv"

import tensorflow as tf
import numpy as np
import requests
import os

## 定数
IMG_SIZE = 480 ## 学習させる画像の縦幅・横幅
IMG_LENGTH = IMG_SIZE * IMG_SIZE * 3 ## 学習させる画像データ長
LABEL_CNT = 2 ## ラベルの種類の数

## 学習に必要な変
x = tf.placeholder(t
W = tf.Variable(tf.z
b = tf.Variable(tf.ze
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_entropy)

## CSVファイルをワークキューとして設定
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])

myconfig = tf.ConfigProto(
   intra_op_parallelism_threads=16 )

saver = tf.train.Saver()
sess = tf.Session(config=myconfig)
sess.run(tf.global_variables_initializer())
print(os.path.exists(ckptdir))
if os.path.exists(ckptdir)==True:
   saver.restore(sess,ckptfile)

## バッチ処理の準備
batch_url, batch_label = tf.train.batch([url, label], batch_size=2)
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)
image2 = []
label2 = []

urls, labels = sess.run([batch_url, batch_label])

for url in urls :
        r=requests.get(url)
        image = r.content

        ## 画像をTensorFlowで処理できるように変換
        image = tf.image.decode_jpeg(image, channels=3)
        image = tf.image.resize_image_with_crop_or_pad(image, IMG_SIZE, IMG_SIZE)
        image = tf.reshape(image, [-1])
        image = tf.cast(image, tf.float32) / 255.0
        image = sess.run(image)
        image2.append(image)
        tmp = np.zeros(LABEL_CNT)
        tmp[label] = 1
        label2.append(tmp)
image2 = np.asarray(image2)
lable2 = np.asarray(label2)
image2_len = len(image2)
print (image2_len)

i = 0
for i in range(100):
 _,loss = sess.run([train_step, cross_entropy], feed_dict={x: image2[0:image2_len], y_: label2[0:image2_len]})
 correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
 sess.run(accuracy, feed_dict={x: image2, y_: label2})
 print('Step %d, Loss %f' % (i, loss))
if os.path.exists(ckptdir)==False:
   os.makedirs(ckptdir)
saver.save(sess,ckptfile)
sess.close()
```

学習命令（トレイニングステップ）

```
_,loss = sess.run([train_step, cross_entropy], feed_dict={x: image2[0:image2_len], y_: label2[0:image2_len]})
```

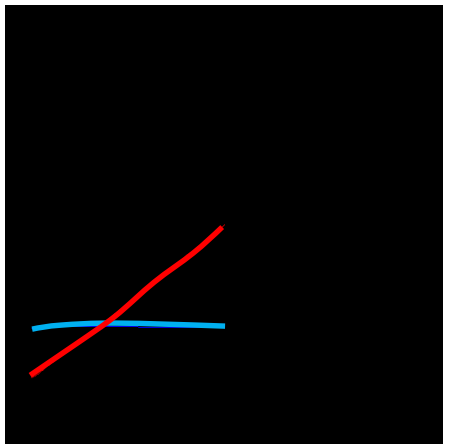
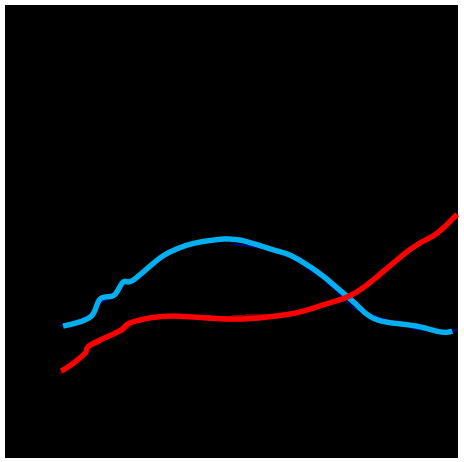
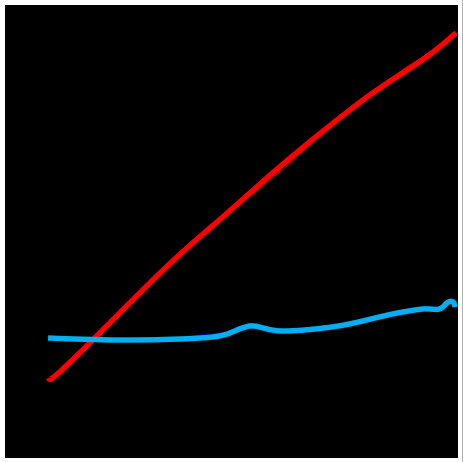
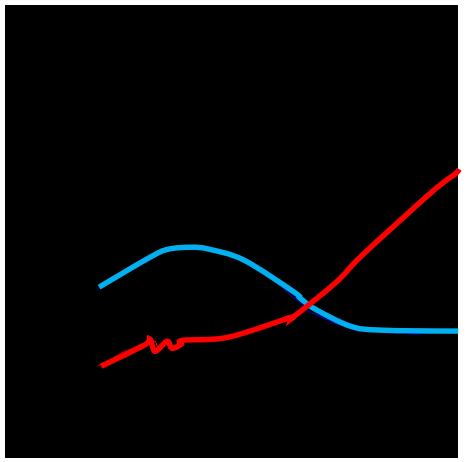
判定プログラムのソースコード

```
##基本変数
ckptdir="/mnt/ckpts/cp001/"
ckptfile="/mnt/ckpts/cp001/cp001.ckpt"
csvfile="/mnt/teacherdata/capacitor/cp001/cp100-test.csv"

import tensorflow as tf
import numpy as np
import requests
import os

## 定数
IMG_SIZE = 480 ## 学習さ
IMG_LENGTH = IMG_SIZE
LABEL_CNT = 2 ## ラベル

## 学習に必要な変数の初期化
x = tf.placeholder(tf.float32, shape=[None, IMG_LENGTH])
W = tf.Variable(tf.zeros([IMG_LENGTH, LABEL_CNT]))
b = tf.Variable(tf.zeros([LABEL_CNT]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, shape=[None, LABEL_CNT])
cross_entropy = tf.reduce_sum(tf.square(y-y_))
train_step = tf.train.GradientDescentOptimizer(1e-7).minimize(cross_entropy)

## CSVファイルをワークキューとして設定
queue = tf.train.string_input_producer([csvfile])
reader = tf.TextLineReader()
key, val = reader.read(queue)
url, label = tf.decode_csv(val, [[''], [0]])

myconfig = tf.ConfigProto(
   intra_op_parallelism_threads=16 )

saver = tf.train.Saver()
sess = tf.Session(config=myconfig)
lizer())

atch([url, label], batch_size=4)

nners(sess=sess, coord=coord)

batch_label])

    tmp = np.zeros(LABEL_CNT)
    r=requests.get(url)
    image = r.content

    ## 画像をTensorFlowで処理できるように変換
    image = tf.image.decode_jpeg(image, channels=3)
    image = tf.image.resize_image_with_crop_or_pad(image, IMG_SIZE, IMG_SIZE)
    image = tf.reshape(image, [-1])
    image_val = sess.run(image).astype(np.float32) / 255.0
    p=sess.run(y, feed_dict={x: [image_val], y_: [tmp]})[0]
    print ("{0}:".format(url.decode('utf-8')),end='')
    print (np.argmax(p),end='')
    print ("(value = {0})".format(np.max(p)))
sess.close()
```

```
#判別ステップ
 p=sess.run(y, feed_dict={x: [image_val], y_: [tmp]})[0]
#表示部分
print ("{0}:".format(url.decode('utf-8')),end='')
 print (np.argmax(p),end='')
 print ("(value = {0})".format(np.max(p)))
```

判定結果(2017/12/28)

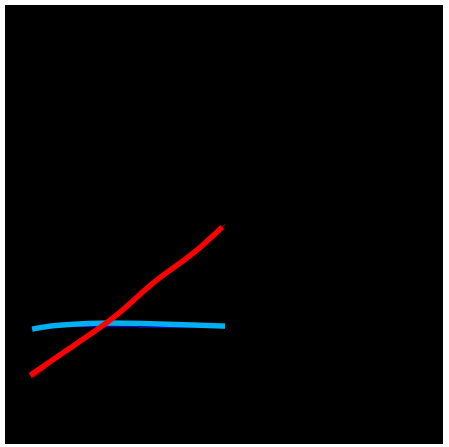
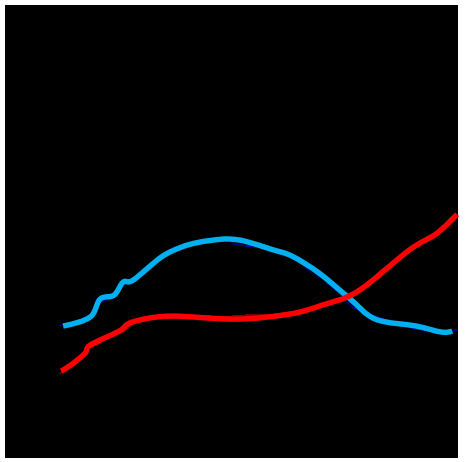
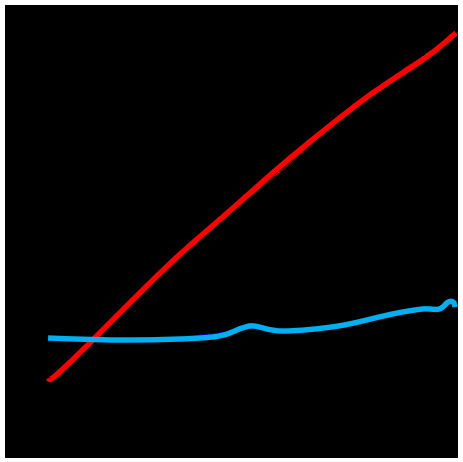
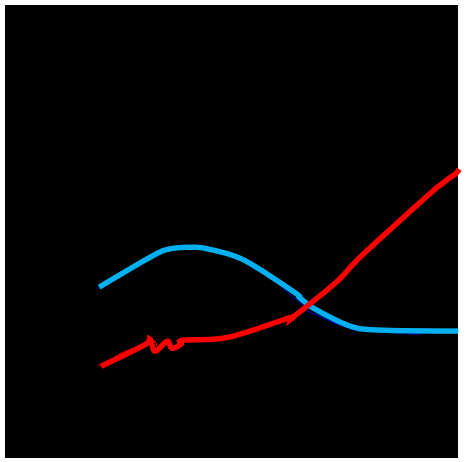
未知のデータの判定　　機械学習で使った教師データ

ファイル名	CapacitorDLImage_GOODcp001-50.png	CapacitorDLImage_BADcp100-50.png	CapacitorDLImage_GOOD.png1 (性能：Good)	CapacitorDLImage_BAD.png (性能：Bad)
画像				
判定結果	1 = Good	0 = Bad	1 = Good	0 = Bad
確率/-	0.53120577335357 67	0.5198063850402832	0.9391119480133057	0.935306191444397

プログラムの表示結果

http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_BADcp100-50.png:0(value = 0.5198063850402832)
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_GOODcp001-50.png:1(value = 0.5312057733535767)
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_BAD.png:0(value = 0.935306191444397)
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_GOOD.png:1(value = 0.9391119480133057)

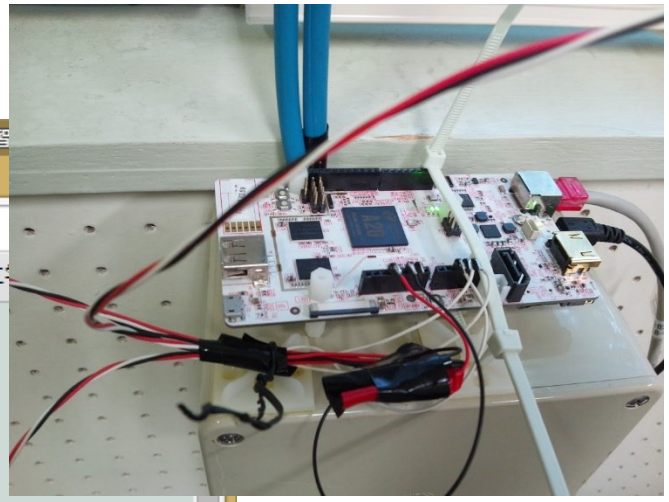
判定結果(2018/1/8)

	未知のデータの判定		機械学習で使った教師データ	
ファイル名	CapacitorDLImage_GOODcp001-50.png	CapacitorDLImage_BADcp100-50.png	CapacitorDLImage_GOOD.png1 (性能：Good)	CapacitorDLImage_BAD.png (性能：Bad)
画像				
判定結果	1 = Good	0 = Bad	1 = Good	0 = Bad
確率 *P*/-	0.5460538268089294	0.530039370059967	0.9827914834022522	0.9827913641929626

プログラムの表示結果

http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_BADcp100-50.png:0(value = 0.530039370059967)
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_GOODcp001-50.png:1(value = 0.5460538268089294)
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_BAD.png:0(value = 0.9827913641929626)
http://localhost/mnt/teacherdata/capacitor/cp001/CapacitorDLImage_GOOD.png:1(value = 0.9827914834022522)

サーバ室モニタリング



サーバルーム温度監視・温度

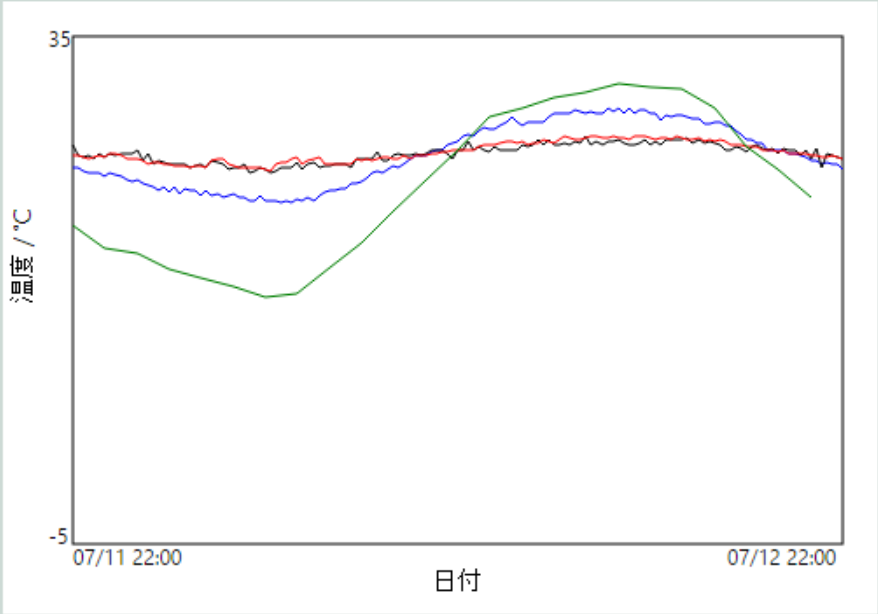
https://a.yamagata-u.ac.jp/amenity/network/ServerRoomViewWeb.aspx

アプリ　サーバルーム温度履歴　IdM実験室: WIF E…　ASP.NET MVC 5 で…　Protecting an ASP…　Java OpenID Con…　files　山形大学工…

計算機室の概略図 T_i = 24.5 °C , T_a = 25.4 °C , T_o = 25.3 °C, P_{sp} = 1.87 kW

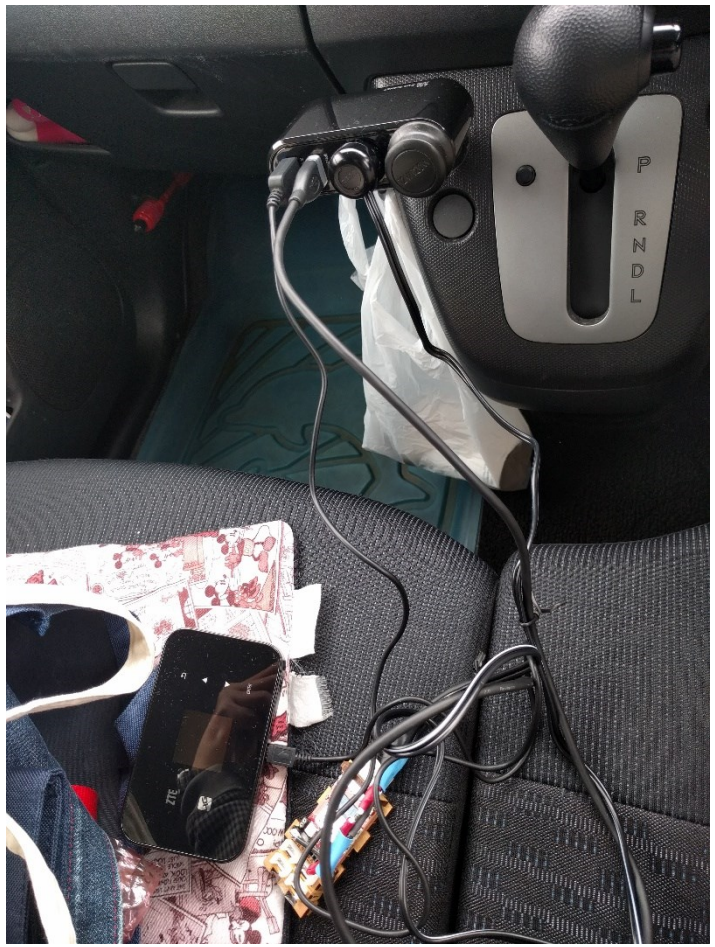
太陽光発電+Li電池蓄電システムの発電量は-0.47 kWで、計算機室への送電量はP_{sp} = 1.87 kW(力率 f = cosφ = 0.81 ･)です。推定ですが、現在の外気吸入による冷却能力P_{oa}は.1 kWです。アメダスによると，2015年07月12日 21時00分の山形県米沢市の1時間の平均気温は，22.3℃です．アメダスデータは、MetBrokerからMetXML形式のデータを定期的に取得して、更新しております。

$$P_{oa} = (T_o - T_i) \times 0.288 \times 500 \times 4.185 \div 3600$$



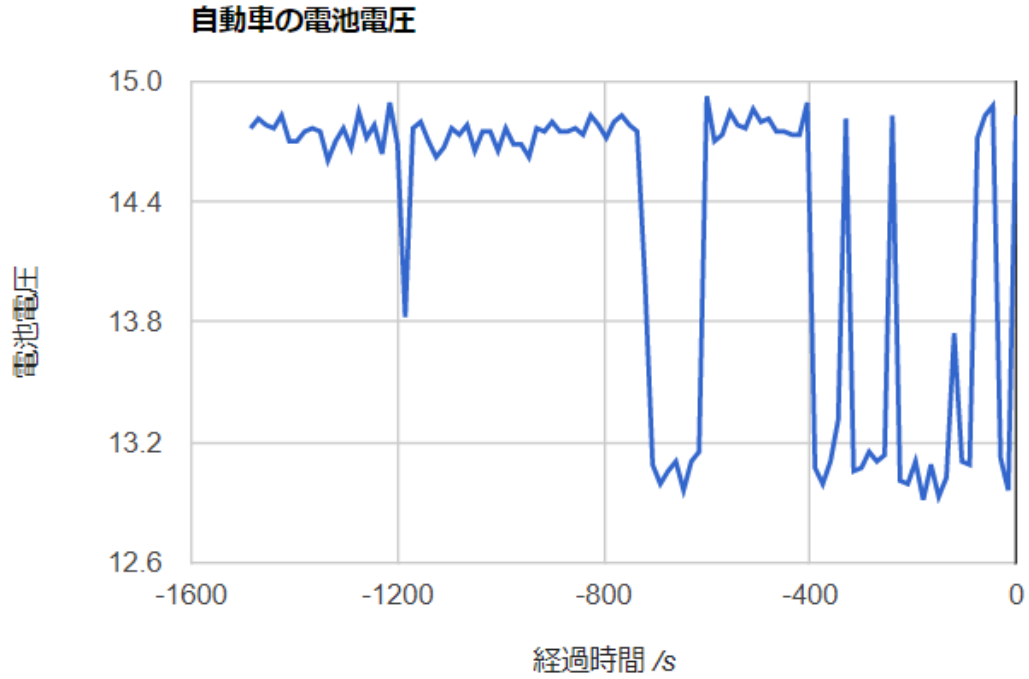
https://a.yamagata-u.ac.jp/amenity/network/ServerRoomViewWeb.aspx

自動車のバッテリーのモニタリング



自動車運転時のバッテリー電圧

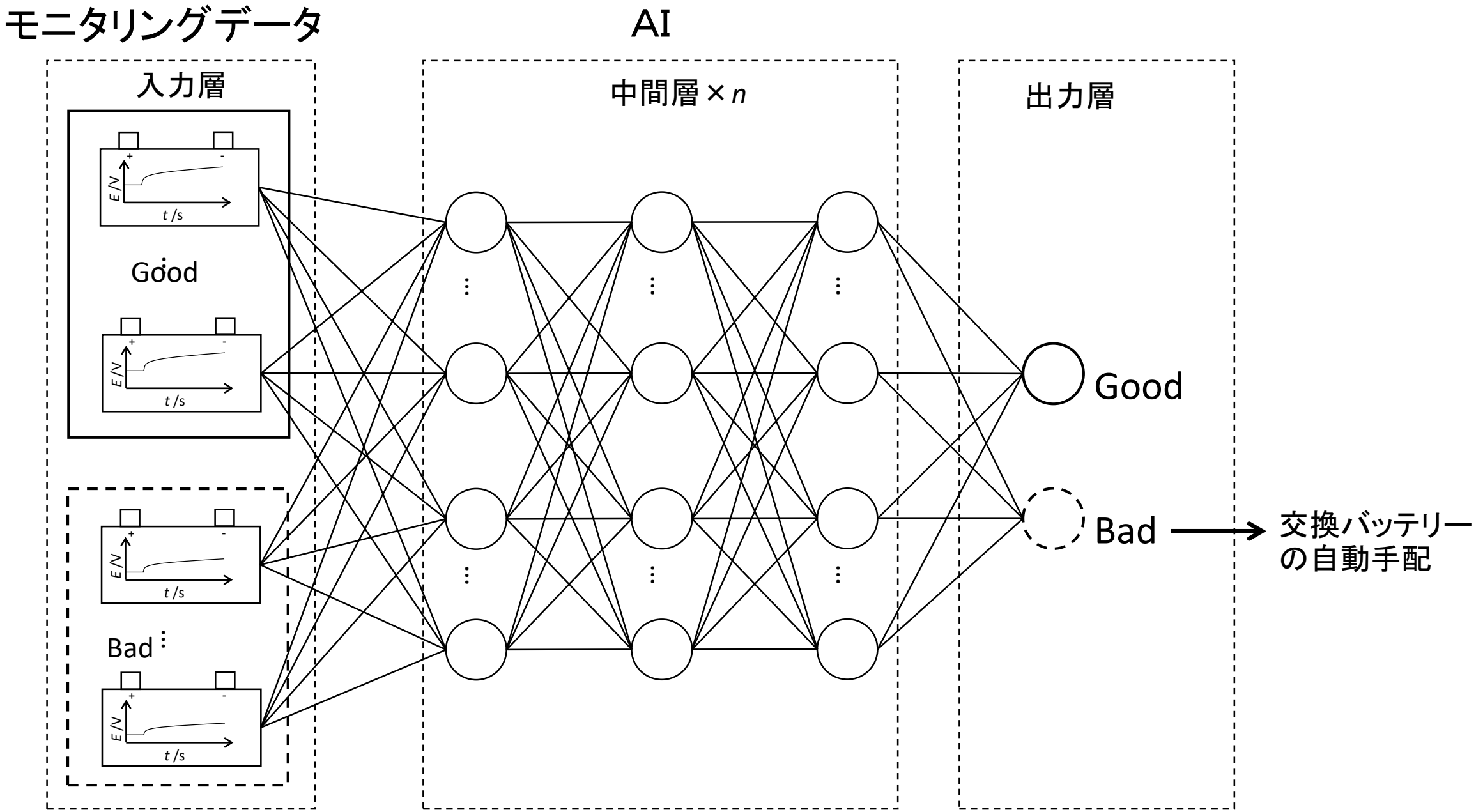
下図に自動車を運転しているときのバッテリーの電圧を示す.



自動車のバッテリーの電圧(最新数値データ)

https://edu.yz.yamagata-u.ac.jp/Public/54299/c1/IoT/BattryMonitoring/vehicle/

IoTとAIが連携した自動車の管理コストの低減



まとめ

- AIは，人件費を削減でき，人類を幸せにする.
  → 自動化できるものは，自動化しちゃいましょう.
  → 機器分析は，可能な限り自動化できます.

- 人は，データの規格化および教師データを準備するための基礎研究が重要になる.

- コンデンサーの性能をAIで判別できる.
  →品質管理への応用が可能

謝辞

- 導電性高分子アルミニウム電解コンデンサの基礎データを収集してくださった関口理希氏，白谷貴明氏，後藤武氏，および，研究室のスタッフの皆さまに感謝いたします．

- 研究するにあたり，アドバイスをいただいた仁科辰夫先生，立花和宏先生に感謝いたします．