

# 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](mailto: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

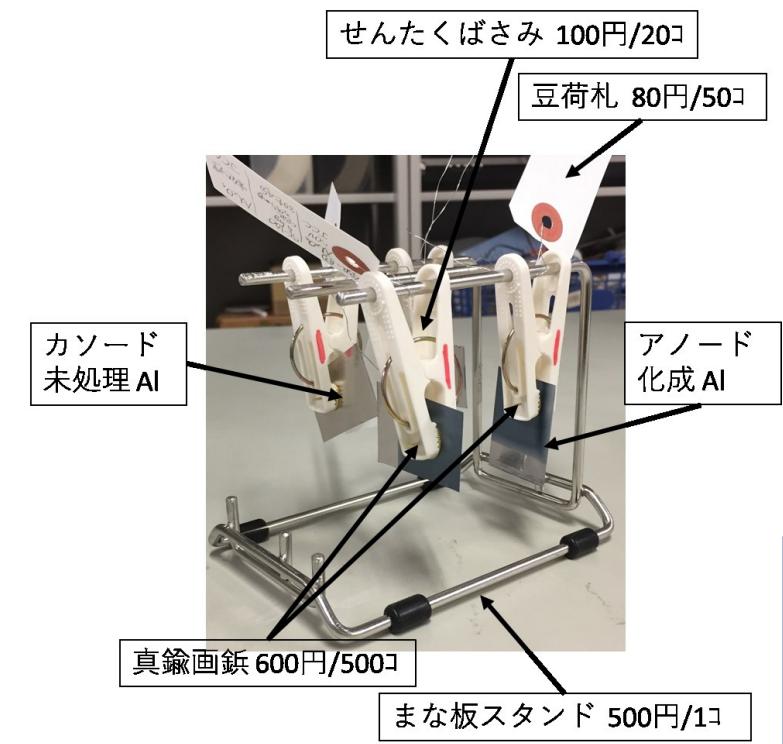
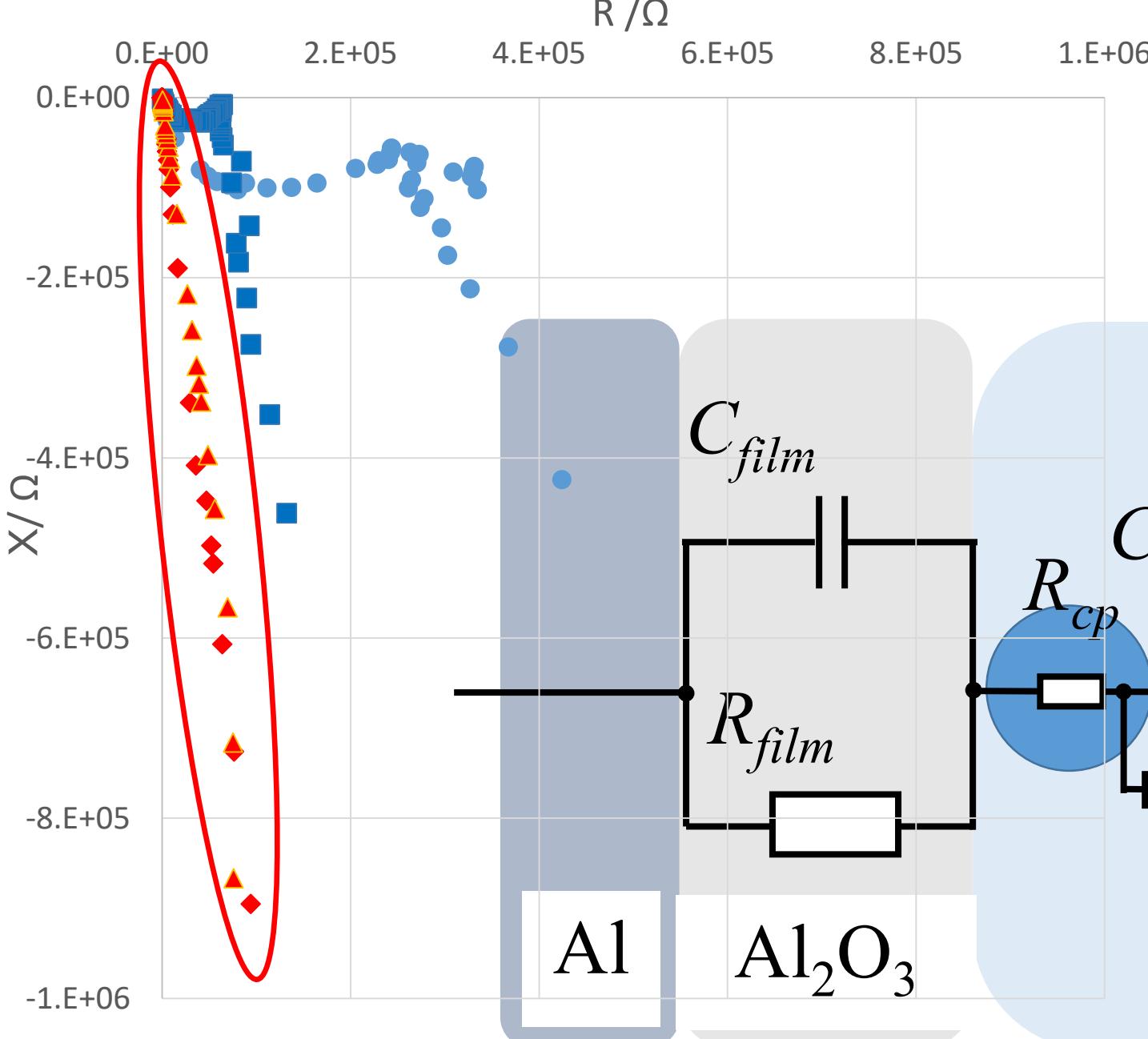
年代	メモリ	価格／万円	CPU
1995	256MB	500	Super SPARC, Single Core
1997	1GB	500	Ultra SPARC, Single Core
2007	32GB	500	Opteron, Dual Core
2014	768GB	1400	Xeon, 10 core
2018	1024GB	1000	EPYC, 32 core

# 機械学習用のライブラリーの充実

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

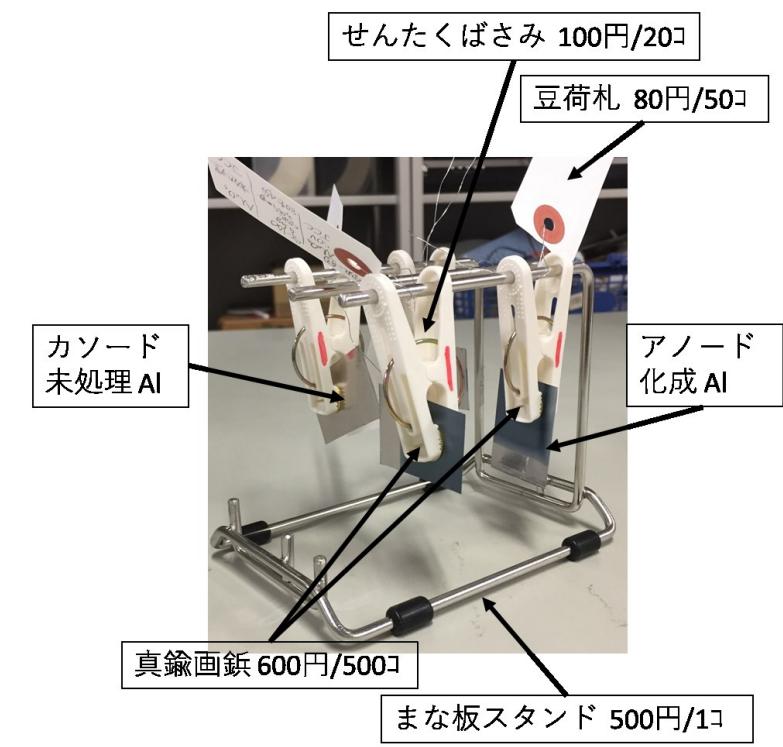
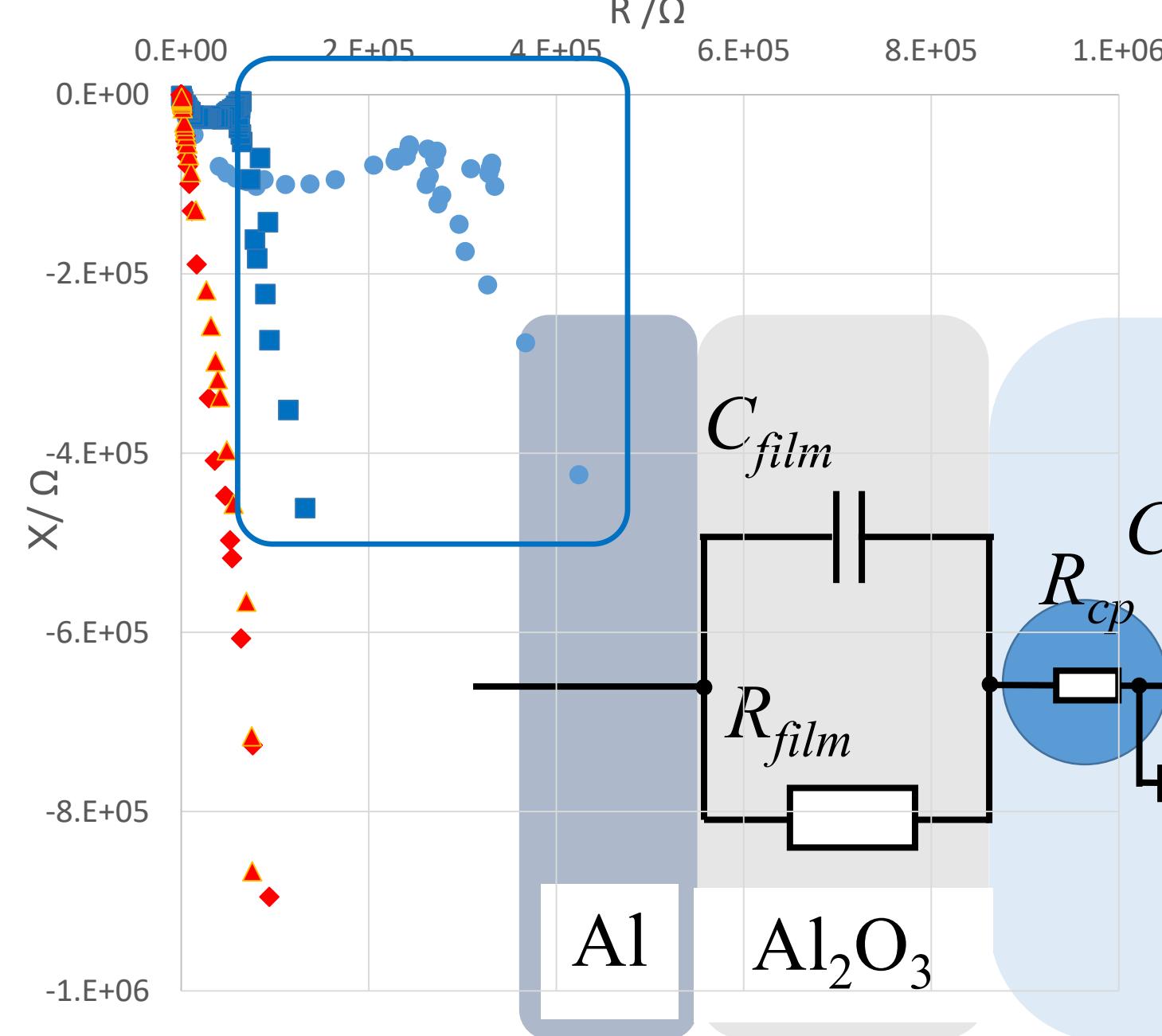
無償で利用できます。

# 高周波特性に優れた電解コンデンサ-



Al

# 高周波特性に劣る電解コンデンサー



Al

# 機械學習

# 教師データ

Bad Good

0 1

1 0

$y$  (2行2列)

学習データ

The diagram illustrates the convolutional process from input  $x$  to output  $y$ . The input  $x$  is a 2D matrix of size 480x480x3, represented by a black square divided into 3x3 smaller squares. A blue line labeled  $x$  indicates the receptive field of a specific output unit. The output  $y$  is a 2D matrix of size 100x100, represented by a black square divided into 5x5 smaller squares. A red line labeled  $y$  shows the result of the convolution operation. The diagram also shows the stride of the convolution, indicated by a blue arrow pointing to the right.

Diagram illustrating the convolutional process from input  $x$  to output  $y$ .

Input  $x$ :  
 $480 * 480 * 3 = 691200$

Output  $y$ :  
 $100 * 100 = 10000$

Stride: 2

機械学習で赤線内の  $w$  と  $b$  の行列を近似する。

The diagram illustrates the calculation of the number of combinations for a convolutional layer. It shows two sets of input channels (Bad and Good) and their corresponding output channels ( $b_1$  and  $b_2$ ). The calculation is as follows:

$$480 * 480 * 3 = 691200$$

**W(691200行2列)**

**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

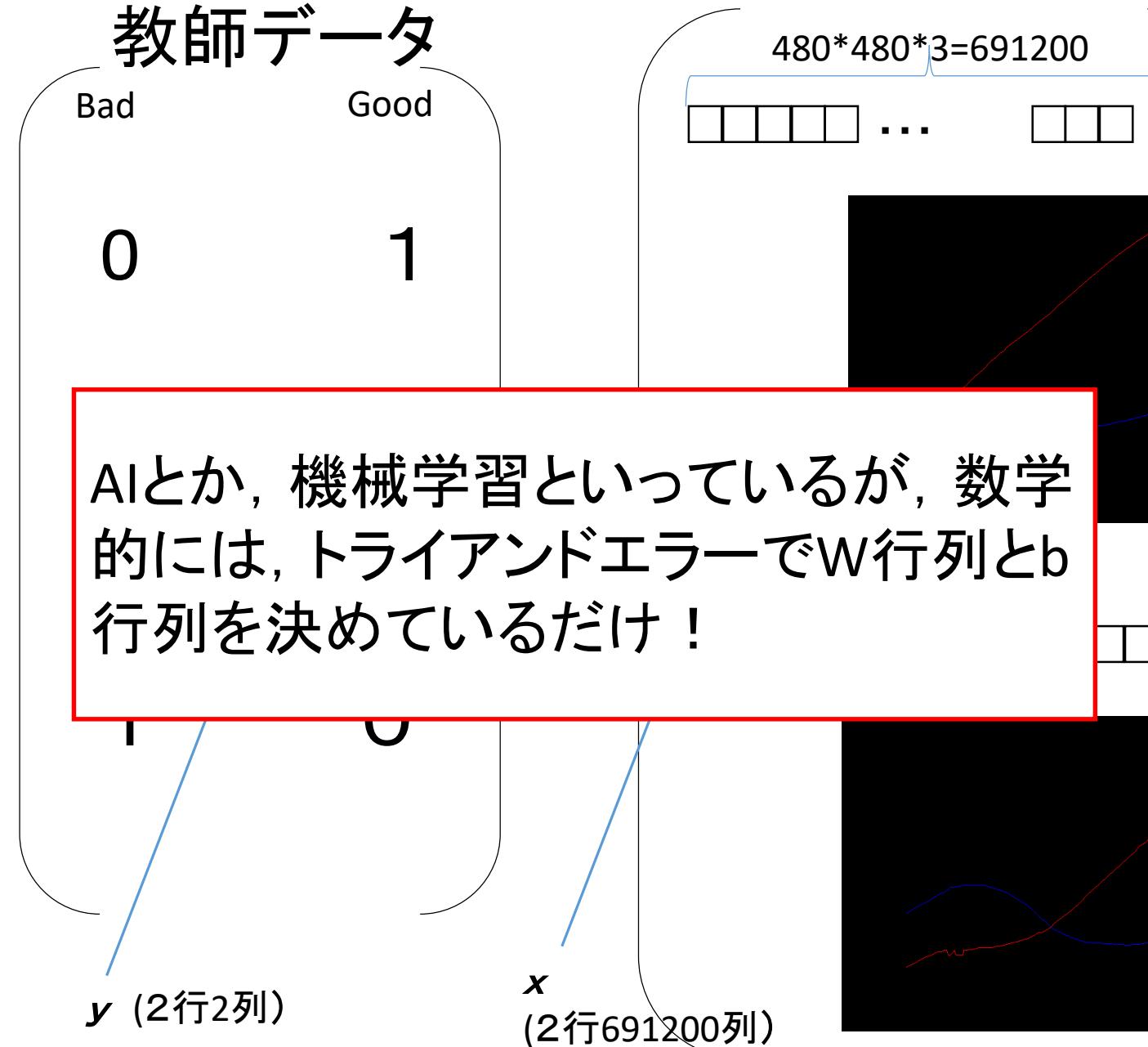
	Bad	Good
0		
1		

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！

$y$  (2行2列)

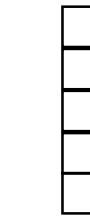
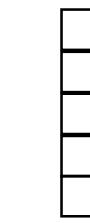
$x$  (2行691200列)

## 学習データ



機械学習で赤線内のwとbの行列を近似する。

Bad Good

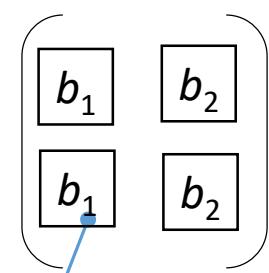


:

:

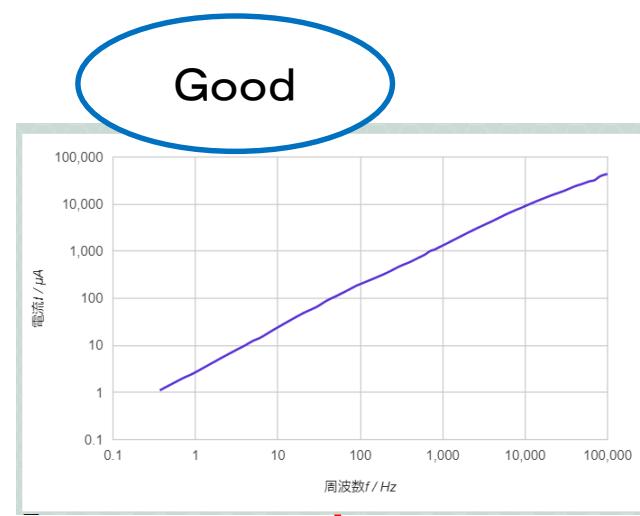
480\*480\*3=691200

Bad Good

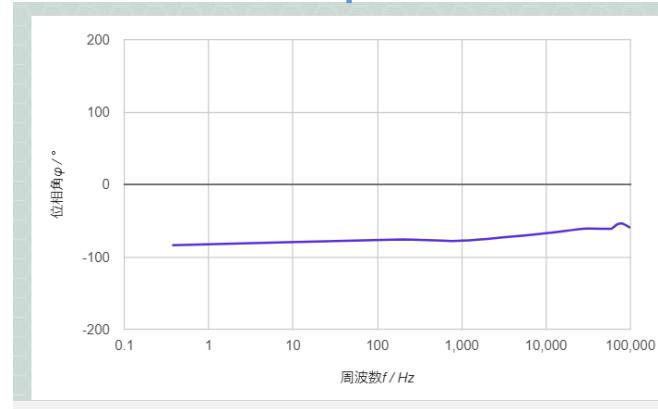


# 教師データと学習データの規格化

## 学習データ

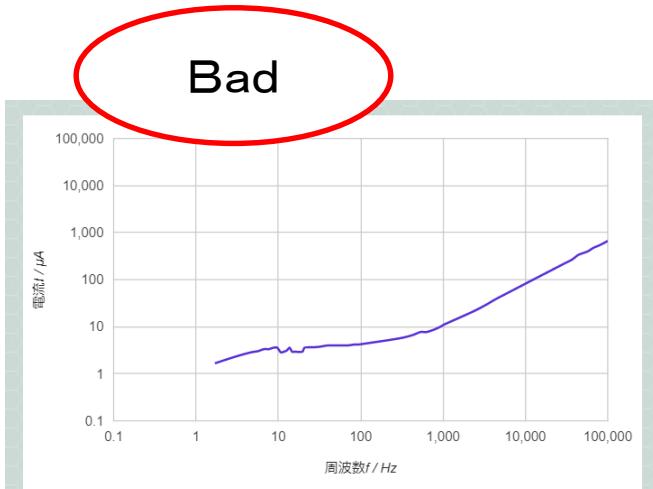
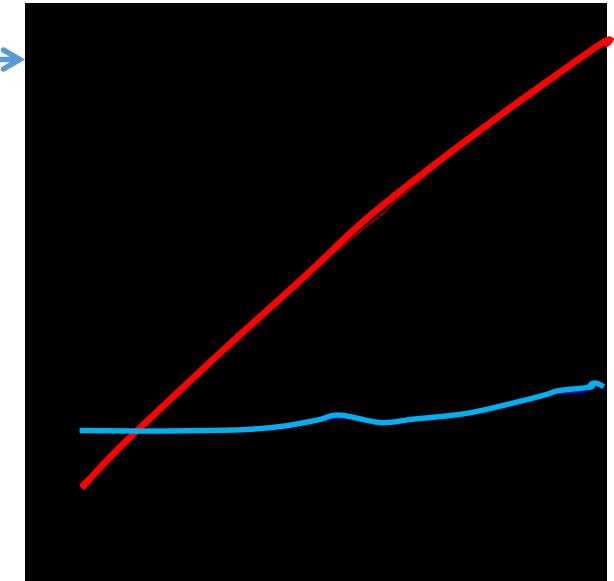


+

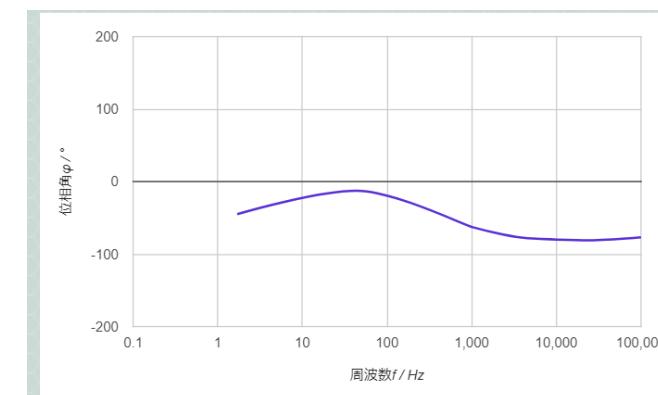


B(255)で画像にプロット

画像に変換

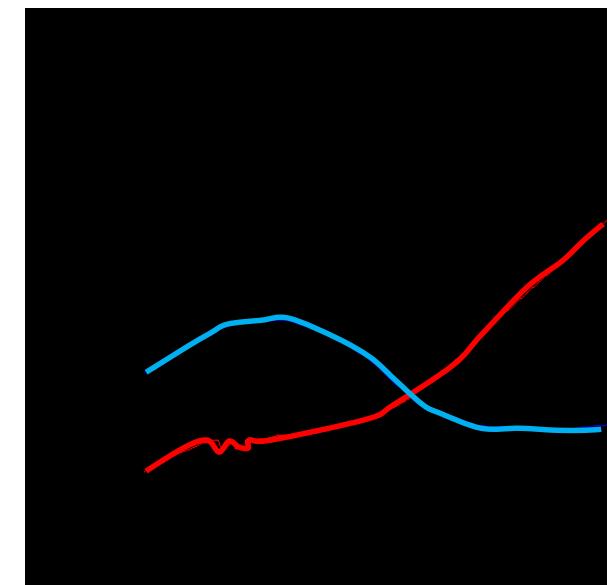


+



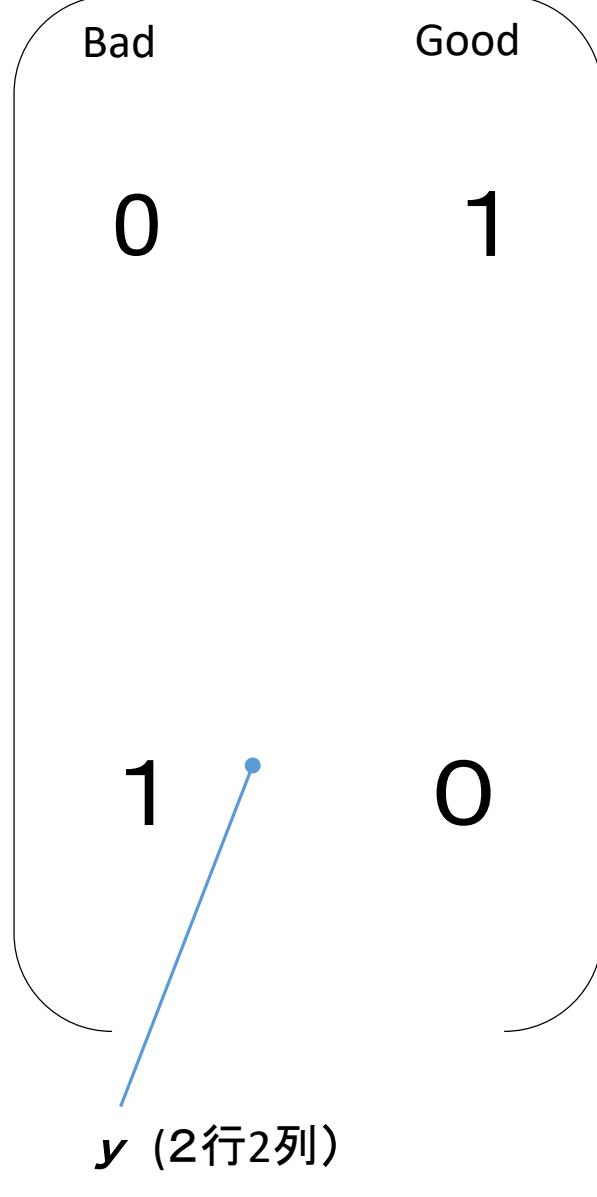
R(255)で画像にプロット

→

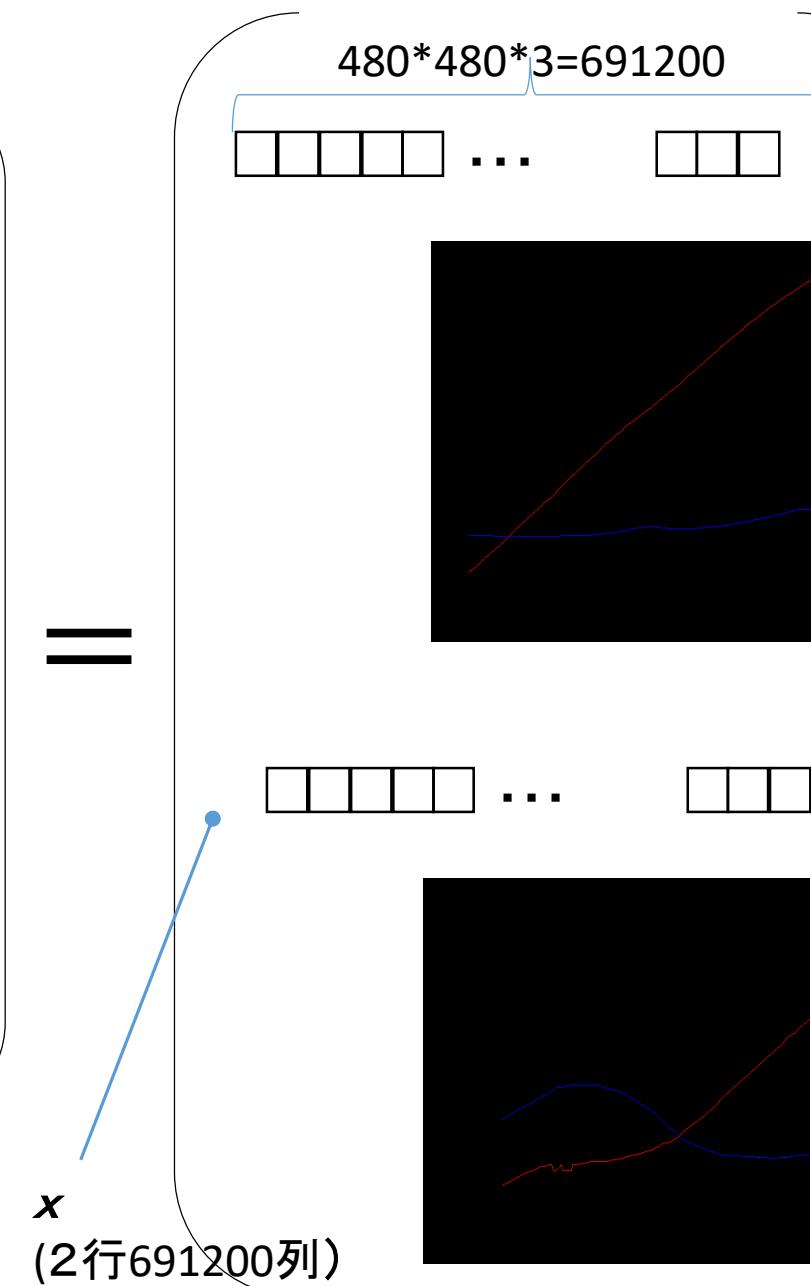


# 機械学習

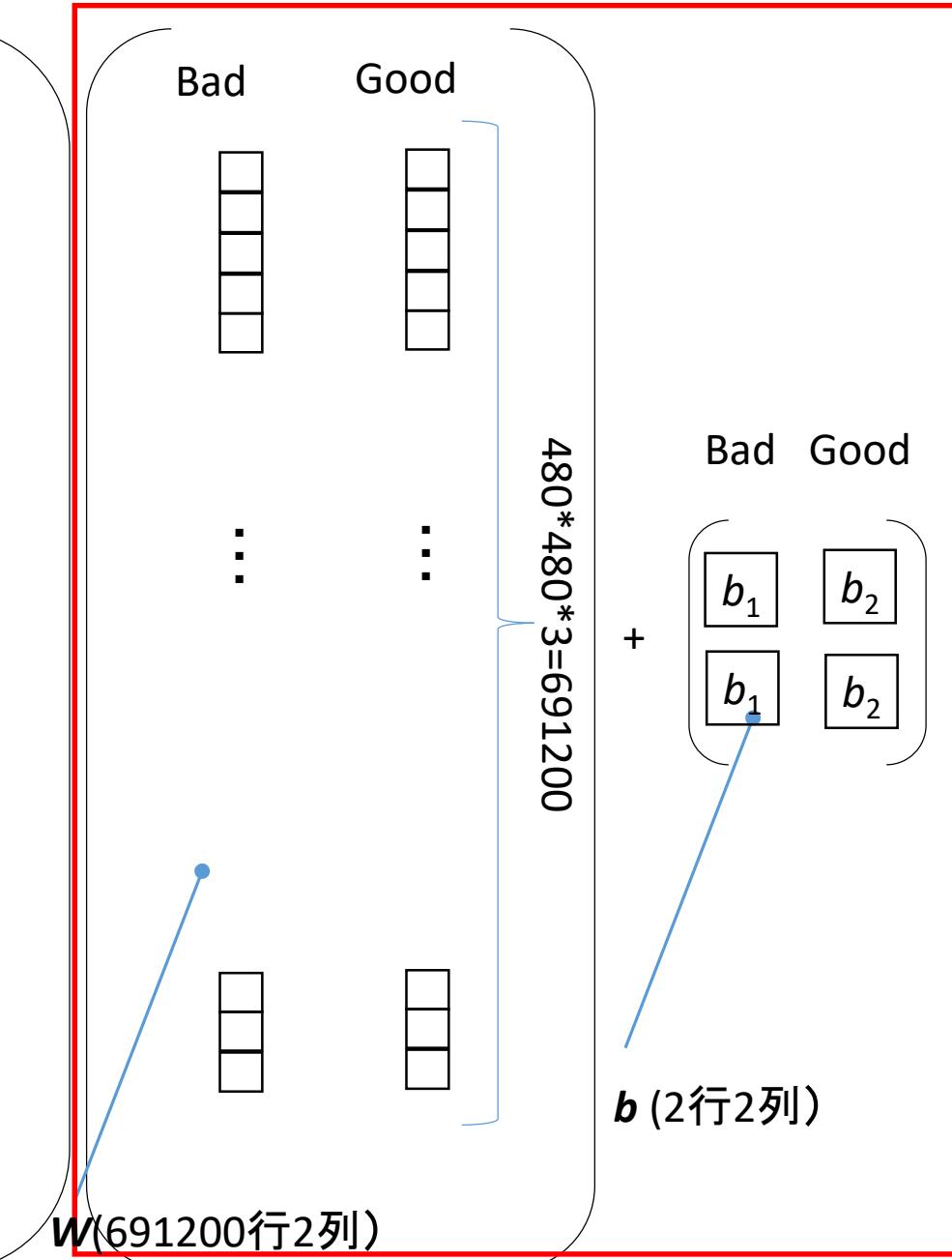
## 教師データ



## 学習データ



機械学習で赤線内の  $w$  と  $b$  の行列を近似する。



# 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)
label2 = 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 ## 学習させる画像データ  
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)
```

\*\*\* バイナリ加工の準備

```
# 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]))  
#  $y = x \cdot W + b$  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])  
#  $\sum (y_n - y_{-n})^2$  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](https://www.tensorflow.org/api_docs/python/tf/square)

tf.reduce\_sum : [https://www.tensorflow.org/api\\_docs/python/tf/reduce\\_sum](https://www.tensorflow.org/api_docs/python/tf/reduce_sum)

```
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]))  
#  $y = x \cdot W + b$  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])  
# $\sum((y_n - y_{-n})^2)$ 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], batch_size=1)  
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)  
label2 = np.asarray(label2)  
image2_len = len(image2)
```

```
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
```

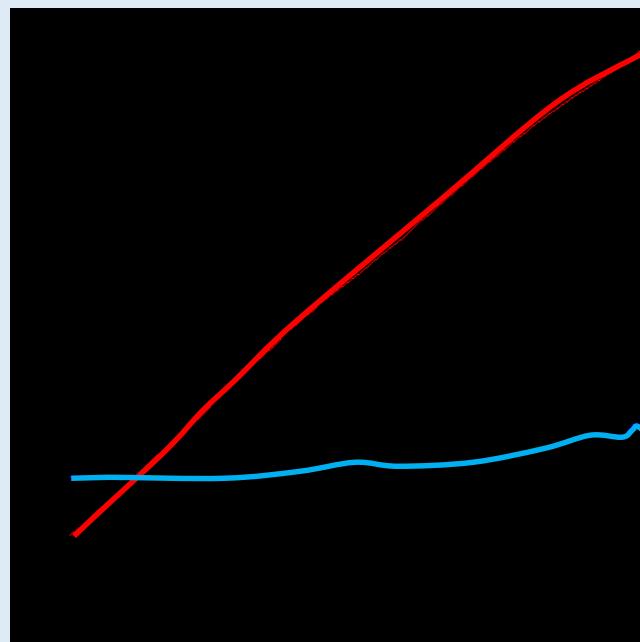
ウェブページから画像を読み込む  
r=requests.get(url)  
image = r.content

# 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
```

$480 \times 480 \times 3 = 691200$



```
if os.path.exists(ckptdir)==True:  
    saver.restore(sess,ckptfile)  
  
## バッチ加工の準備  
  
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)
```

```
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)  
label2 = 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
```

## 学習命令(トレーニングステップ)

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

```
# 学習に必要な変数  
x = tf.placeholder(tf.float32)  
W = tf.Variable(tf.zeros([IMG_SIZE*IMG_SIZE*3, 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_mean(tf.square(y - y_))  
train_step = tf.train.GradientDescentOptimizer(1e-3).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, [[1], [0]])
```

```
myconfig = tf.ConfigProto()  
myconfig.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])
```

```
tmp = np.zeros(LABEL_CNT)  
tmp[label] = 1  
label2.append(tmp)  
image2 = np.asarray(image2)  
label2 = 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()
```

# 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 ## 学習用
IMG_LENGTH = IMG_SIZE
LABEL_CNT = 2 ## ラベル

## 学習に必要な変数の定義
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)

#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)))

## 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])

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

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_GOO Dcp001-50.png	CapacitorDLImage_BAD cp100-50.png	CapacitorDLImage_GOO D.png1 (性能:Good)	CapacitorDLImage_BA D.png (性能:Bad)
画像				
判定結果	1 = Good	0 = Bad	1 = Good	0 = Bad
確率/-	0.5312057733535767	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

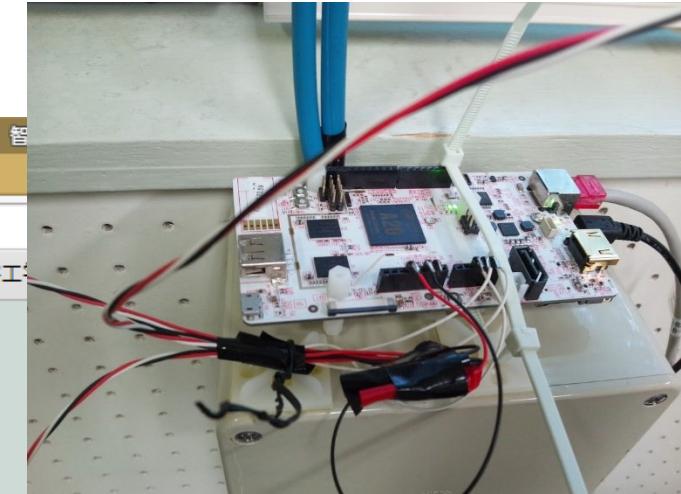
サーバルーム温度監視・温度 ×

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

計算機室の概略図  $T_i = 24.5^{\circ}\text{C}$ ,  $T_a = 25.4^{\circ}\text{C}$ ,  $T_o = 25.3^{\circ}\text{C}$ ,  $P_{sp} = 1.87 \text{ kW}$   
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 $P_{sp} = 1.87 \text{ kW}$ (力率 $f = \cos\varphi = 0.81$ )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°C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。

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

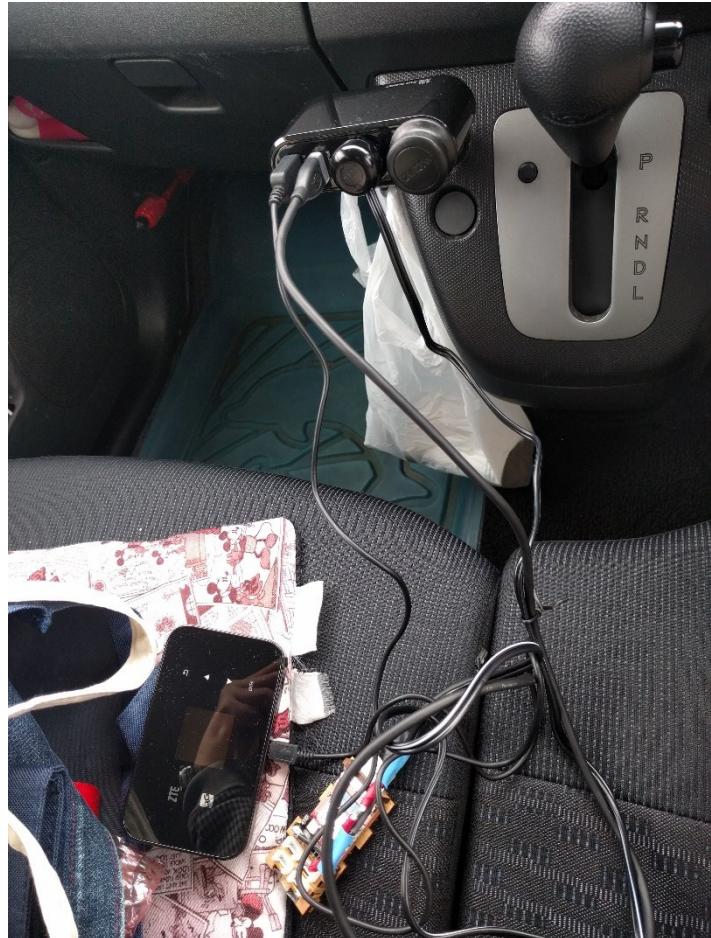
The graph displays temperature data over a 24-hour period. The y-axis represents temperature in degrees Celsius, ranging from -5 to 35. The x-axis shows dates: 07/11 22:00 and 07/12 22:00. There are four data series: a red line (室内), a blue line (外気), a green line (太陽光発電), and a black line (蓄電池). The indoor temperature (red) starts around 24.5°C, fluctuates slightly, and ends around 25.3°C. The outdoor temperature (blue) starts around 25.4°C, rises to a peak of about 26.5°C at 07/12 08:00, and then gradually decreases. The solar power generation (green) starts at approximately 25°C, drops to a minimum of about 23°C around 07/11 18:00, and then rises sharply to a peak of about 28°C around 07/12 08:00 before decreasing again. The battery storage (black) follows a similar pattern to the solar generation, starting at 25°C, dipping to 23°C, and then rising to a peak of about 28°C.



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

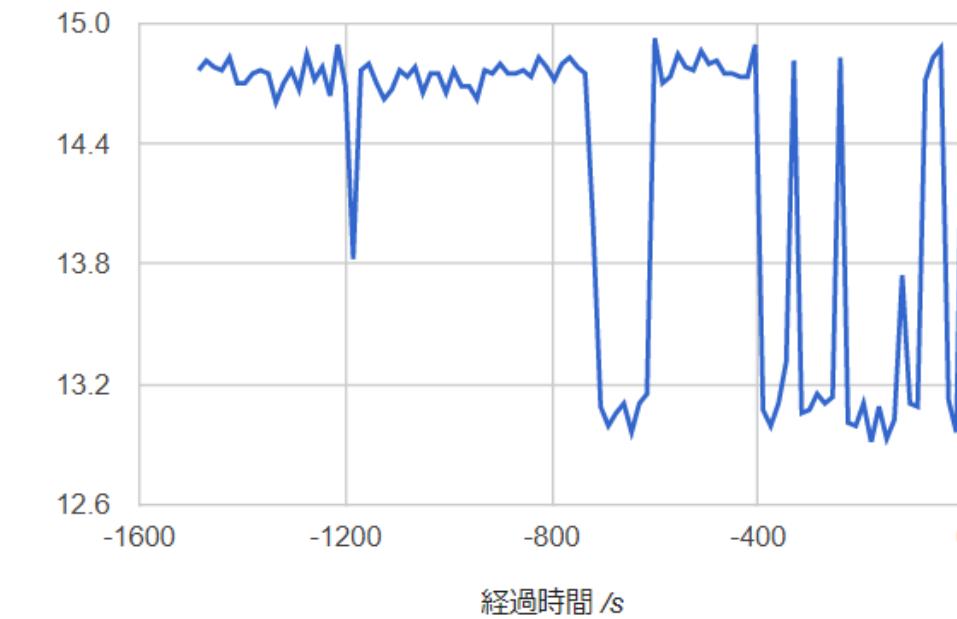
# 自動車のバッテリーのモニタリング

## 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。

自動車の電池電圧



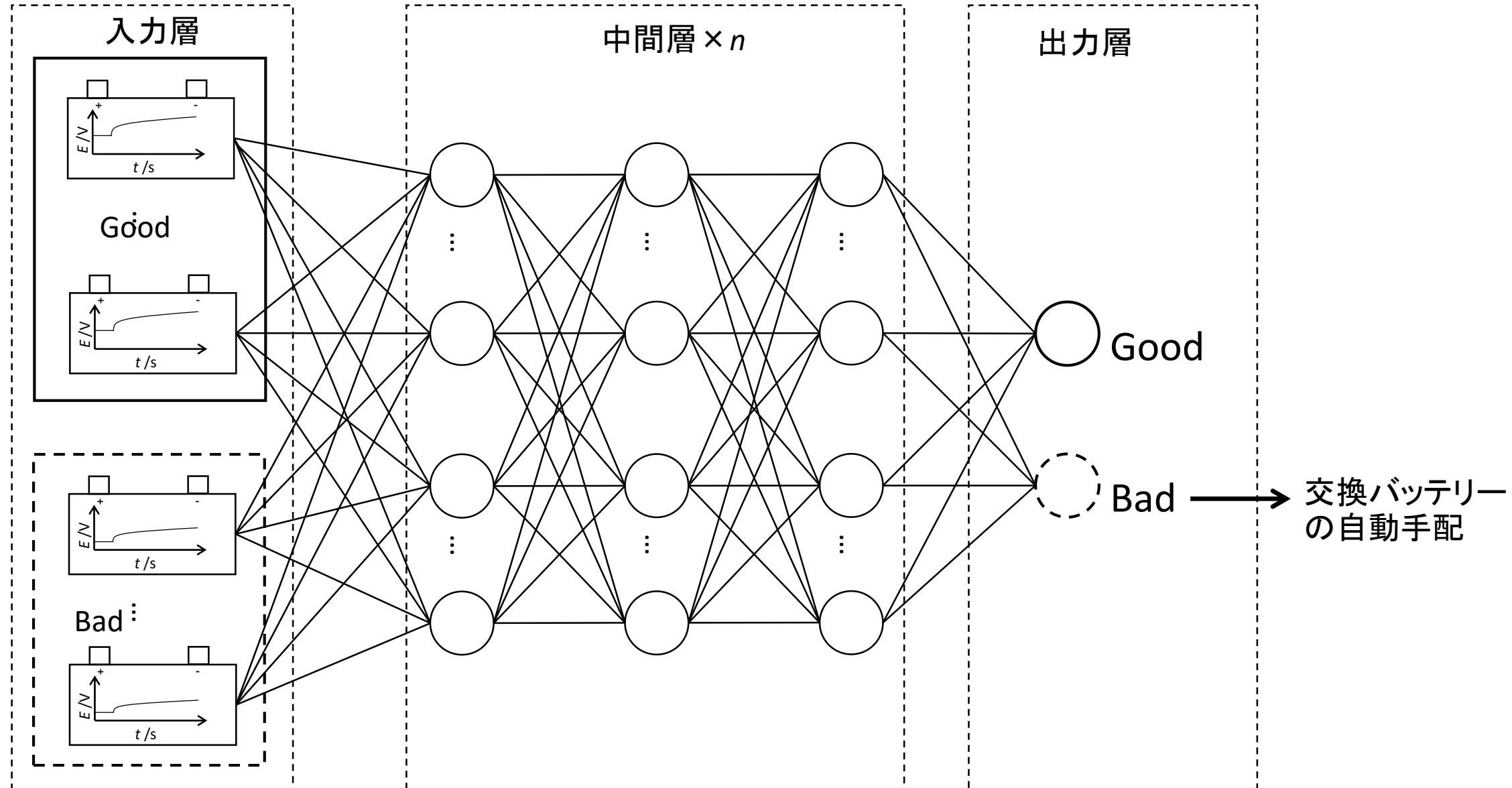
自動車のバッテリーの電圧(最新数値データ)

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

# IoTとAIが連携した自動車の管理コストの低減

モニタリングデータ

AI



# 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んに感謝いたします.
- ・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.