



ディープラーニングを試してみる

演習のターゲット

ディープラーニングで「0」～「9」の手書き数字を見分ける

データセットの選択: mnist.mnist_training

☒ Main ☒ Shuffle ☒ Image Normalization(1.0/255.0) << < 1 / 6000 > >>

Index	x:image	y:label
1	1,28,28 	5
2	1,28,28 	0
3	1,28,28 	4
4	1,28,28 	1
5	1,28,28 	9
6	1,28,28 	2
7	1,28,28 	1
8	1,28,28 	3
9	1,28,28 	1
10	1,28,28 	4

- MNISTの手書き数字データセットを利用
- 2つのシンプルなモデルで精度比較

本日講義のゴール

ゼロから手書き数字識別モデルを構築し、ネットワークをカスタマイズ

1. プロジェクト作成
2. Affineを使ったシンプルなモデルを構築し、検証
3. CNNを使った少し高度なモデルを構築し、検証
4. CNNのネットワークをカスタマイズし、精度の向上を検討
 - 1つのモデルの精度検証に30分程度（実時間）必要になります.
 - 実行計画を立て、精度が最高になるネットワークを見つけてください.

Neural Network Consoleクラウド版にサインイン

Chromeブラウザを起動し、Neural Network ConsoleのWebからクラウドにサインイン。

The screenshot shows the Neural Network Console web interface in a browser. The address bar shows 'dl.sony.com/ja/'. The page has a black header with the 'SONY' logo. Below the header, there's a navigation bar with 'Neural Network Console', a search bar, 'サポート' (Support), 'プロジェクト' (Project), and a blue 'サインイン' (Sign In) button. The main content area features the text 'Neural Network Console' in large font, followed by 'ドラッグ&ドロップで 簡単ニューラルネットワーク設計' (Simple neural network design with drag&drop), and 'コーディング無しでディープラーニングを用いた高度なAI開発を実現' (Achieving advanced AI development using deep learning without coding). On the right side of the main content, there's a complex neural network diagram with various layers like Convolution, Batch Normalization, ReLU, and Pooling.

1. "dl.sony.com/ja/" にアクセス

2. "サインイン"をクリック

プロジェクト作成

ProjectタブのNew Projectから新規プロジェクトを作成する。

Neural Network Console

Dashboard

Project

Da

Job

Sample Project

Public Project

Service Settings

+ New Project ↑ Upload Project Date uploaded ▾ ALL ▾ 🔍 Search

Modified Action

2020-07-09T04:12:0...

☐ tutorial.basics.01_logistic_regression 2020-07-09T01:57:3...
📄 0.37MB

tutorial.basics.01_logistic_regression
📄 7607 ❤️ 55 SAMPLE

tutorial.basics.02_binary_cnn
📄 1336 ❤️ 26 SAMPLE

tutorial.basics.06_auto_encoder
📄 960 ❤️ 18 SAMPLE

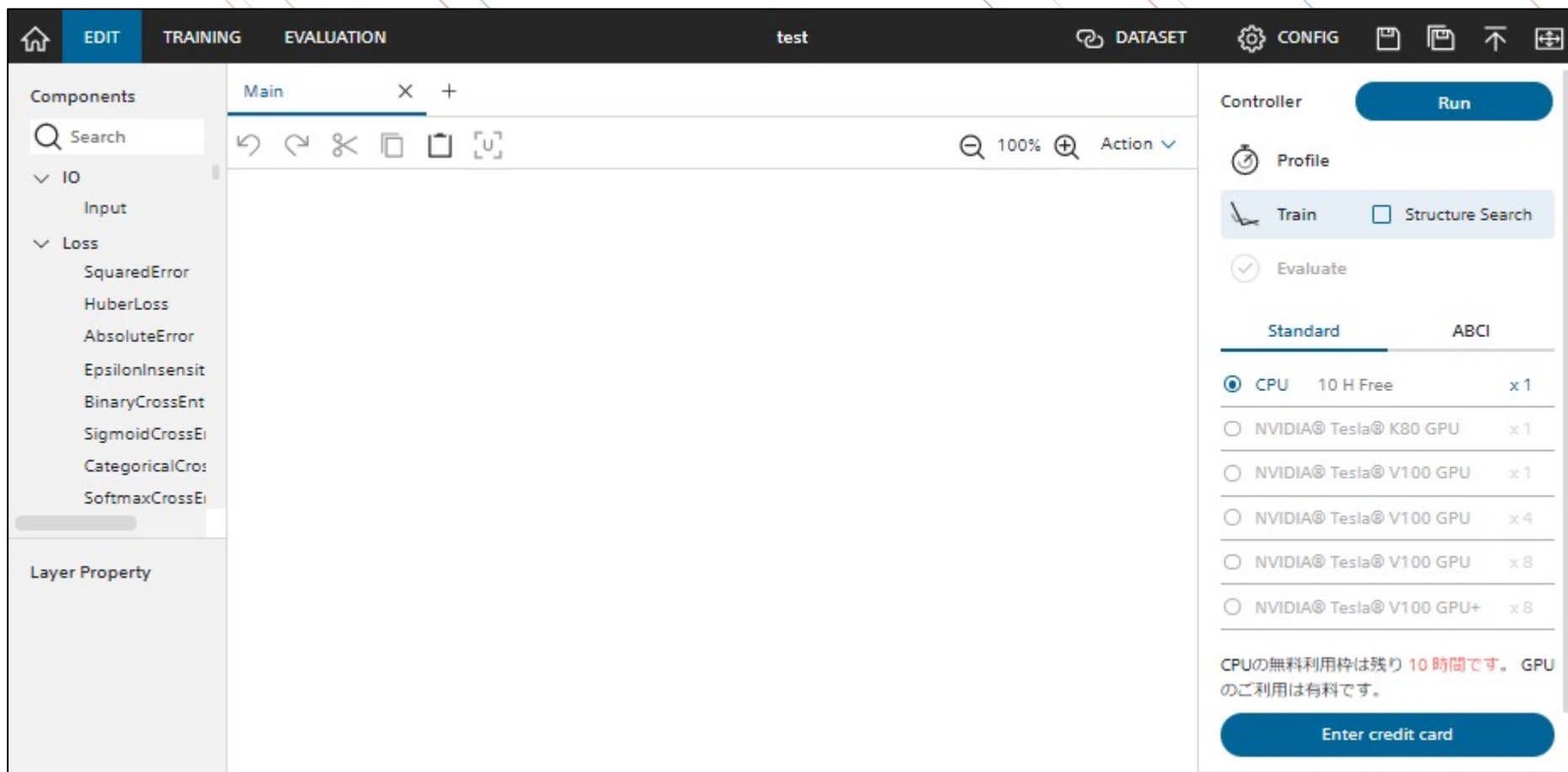
新しいプロジェクト名を入力してください。1～255文字以内で以下の文字は使用できません。
(¥, /, :, *, ?, ", <, >, |, ;)

Cancel OK

3. 半角英数字で任意のプロジェクト名を入力し"OK"をクリック

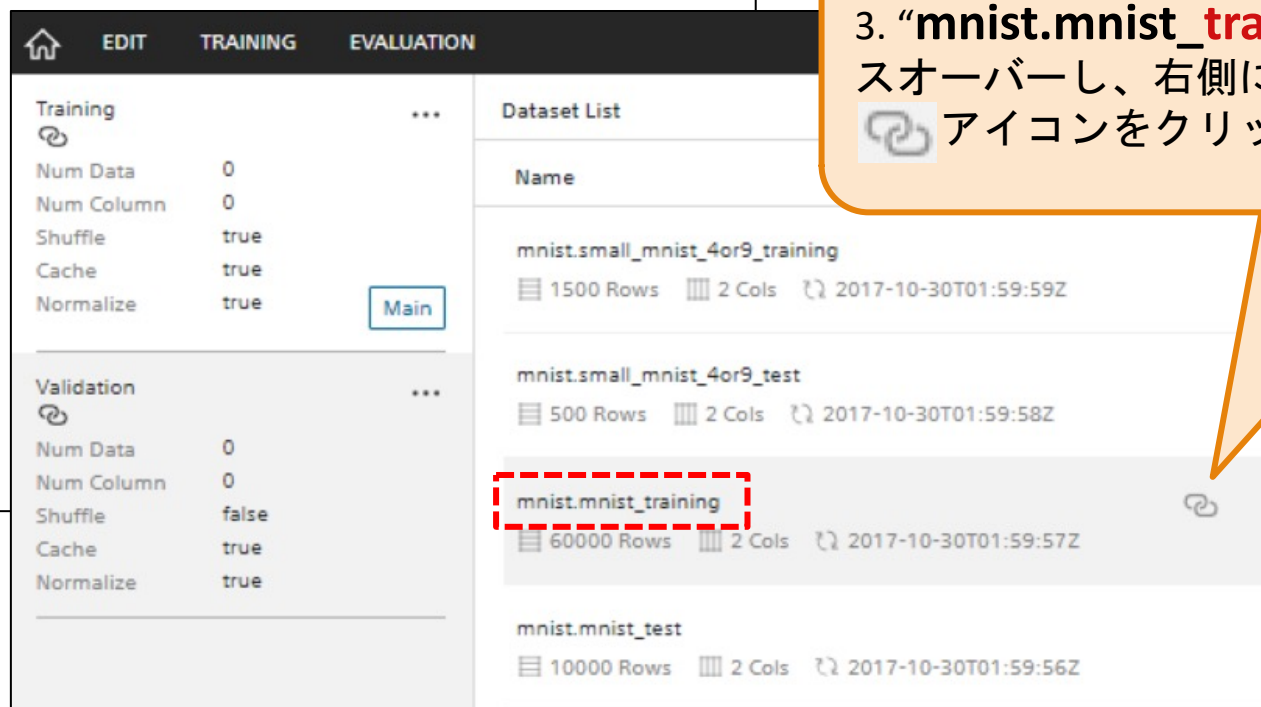
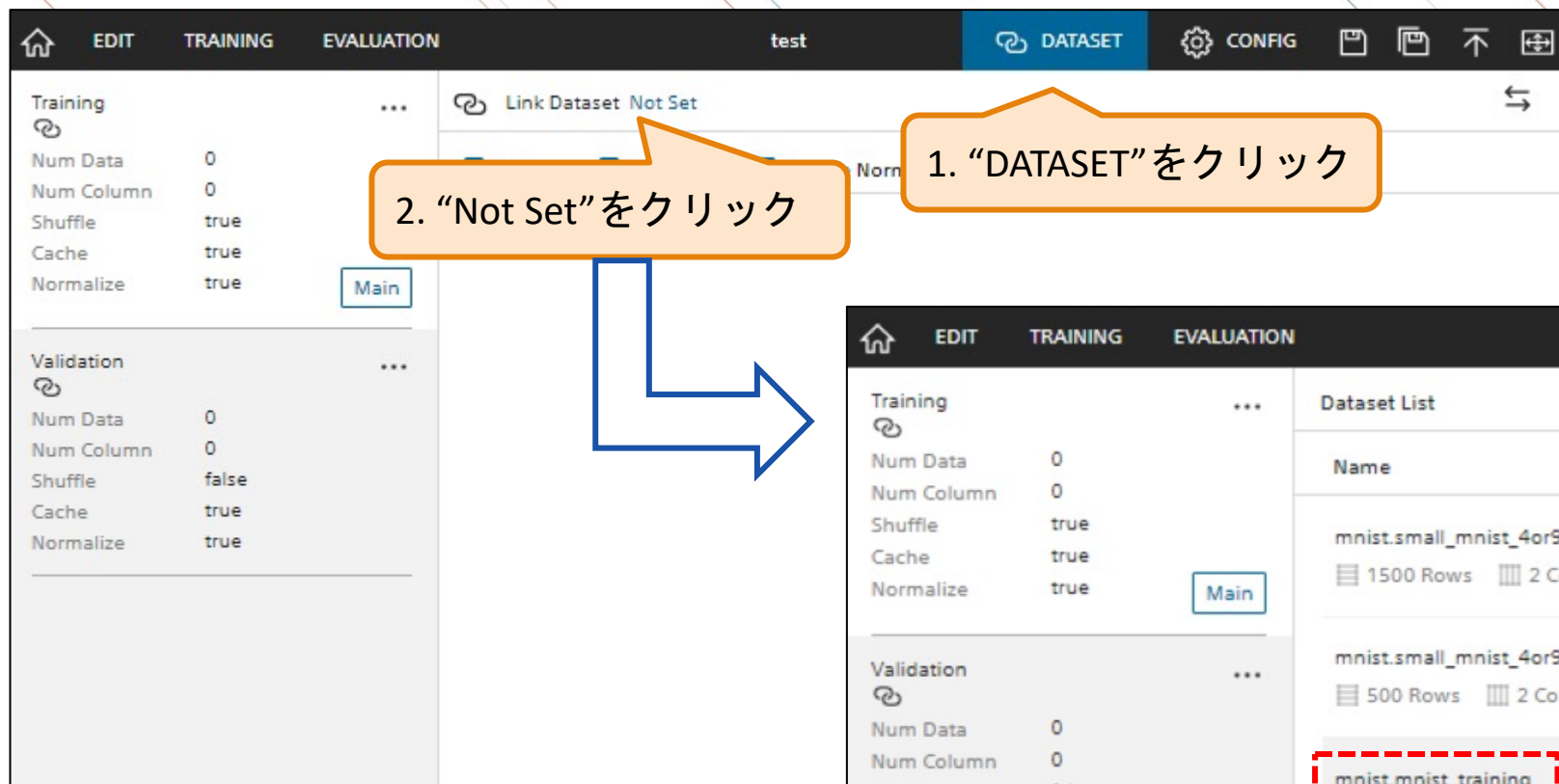
ネットワーク編集画面の確認

ネットワークの編集画面が表示されることを確認。



データセットの連携

学習用データセットの設定。



データセットの連携

評価用データセットの設定。

1. "Validation"をクリック

2. "Not Set"をクリック

3. "mnist.mnist_test"をマウスオーバーし、右側に表示されるアイコンをクリック

test

Link Dataset Not Set

Normalization(1.0/255.0)

Training

mnist.mnist_training

Num Data 60000

Num Column 2

Shuffle true

Cache true

Normalize true

Main

Validation

Num Data 0

Num Column 0

Shuffle false

Cache true

Normalize true

test

Dataset List

ALL Search

Name

mnist.small_mnist_4or9

1500 Rows 2 Co

mnist.small_mnist_4or9

500 Rows 2 Co

mnist.mnist_training

60000 Rows 2 Co 2017-10-30T01:59:57Z

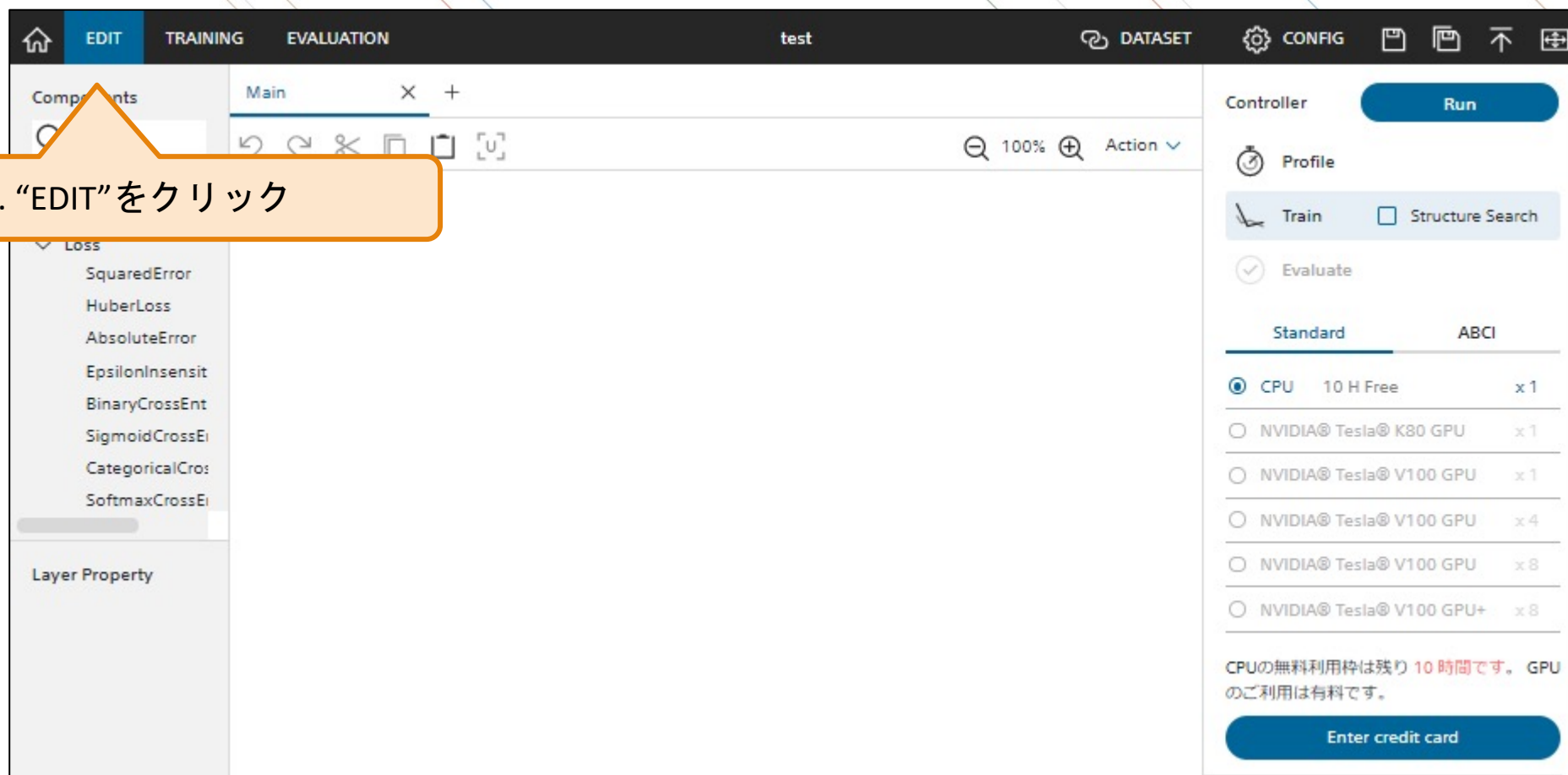
mnist.mnist_test

10000 Rows 2 Co 2017-10-30T01:59:56Z

ネットワーク編集画面の表示

ネットワーク編集画面を表示。

1. “EDIT”をクリック



ネットワーク作成

レイヤー選択パネルからネットワーク構造パネルにレイヤーをドラック＆ドロップする。
 すでにあるレイヤーと連結し、ネットワークを構成する。
 レイヤーパラメータパネルで各レイヤーの詳細を設定する。

1. レイヤーを選択

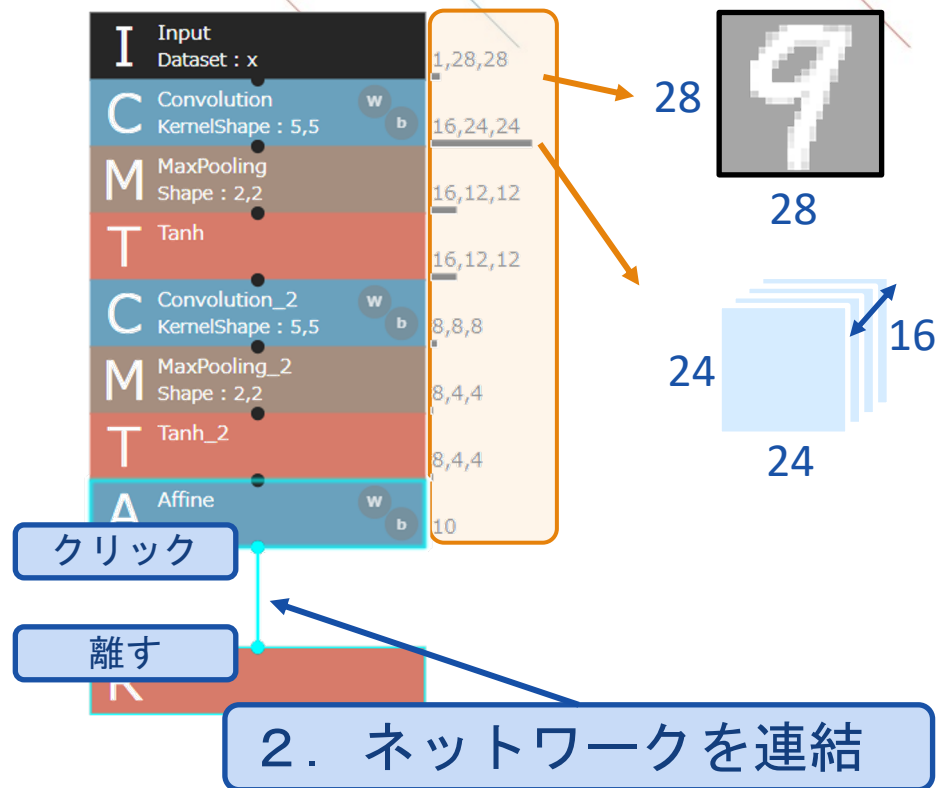
レイヤー選択

レイヤー設定

ネットワーク構造

3. 詳細設定

データの次元を表現



最小のモデル

0～9の手書き数字文字を分類するための分類モデルを作成する。

Layer Property

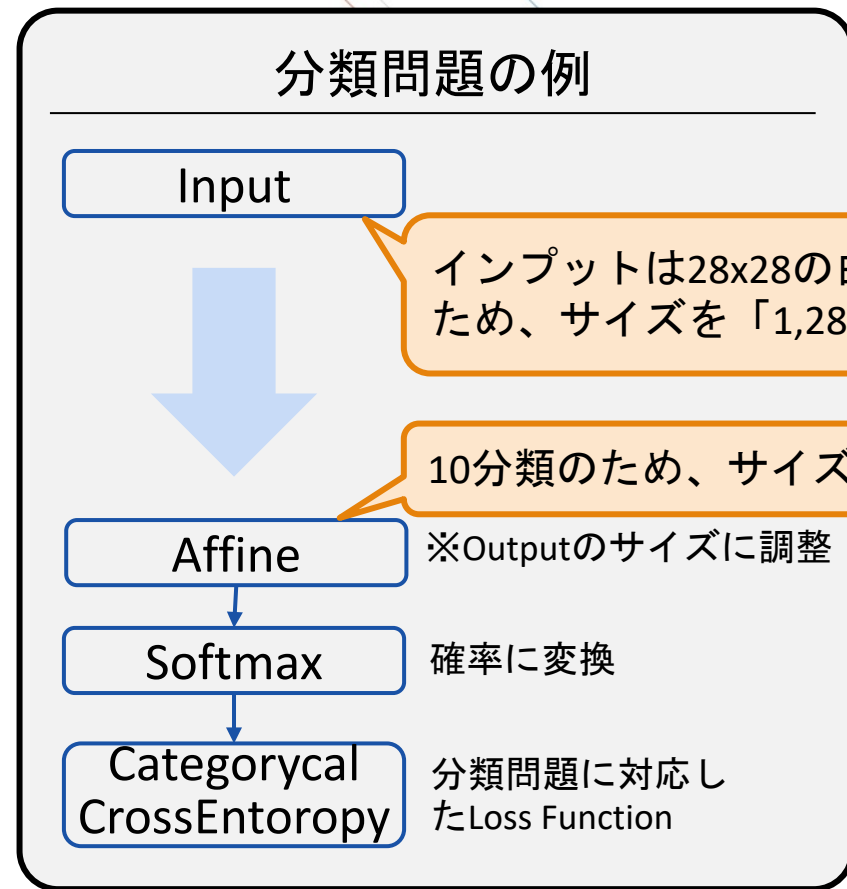
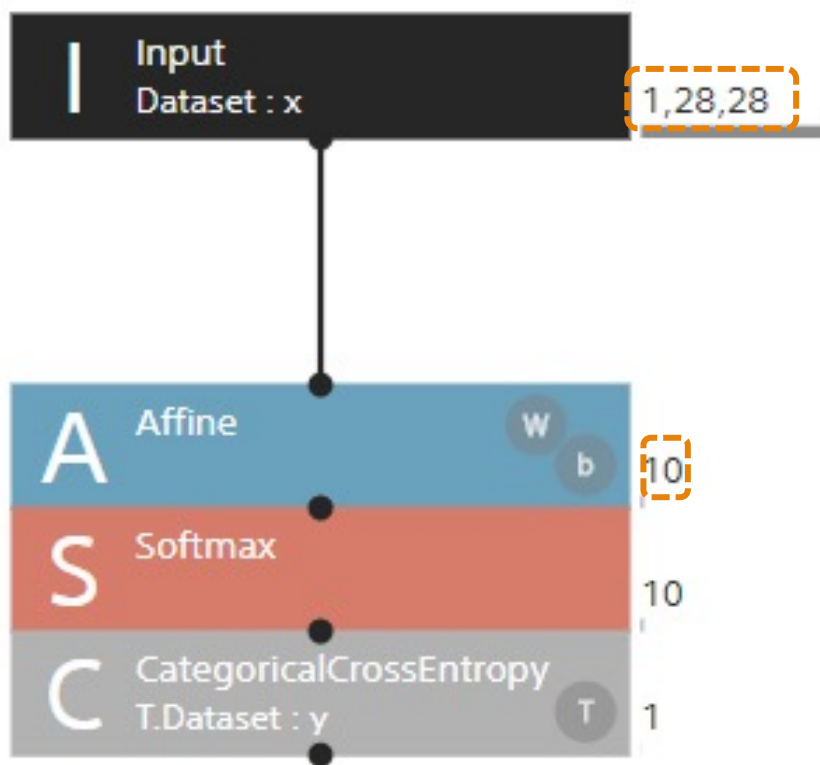
I Input

Name	Input
Size	1,28,28
Dataset	x
Generator	None
GeneratorMultiplier	1
Output	1,28,28

Layer Property

A Affine

Name	Affine
Input	1,28,28
OutShape	10
BaseAxis	0
WithBias	☒
ParameterScope	Affine



学習回数の設定

学習の繰り返し回数を設定する。

The screenshot shows the 'CONFIG' tab of a machine learning interface. The left sidebar contains a 'Config' section with three expandable categories: 'Global Config' (Global), 'Optimizer' (Optimizer), and 'train_error' (Monitor). The main area displays the 'Global Config' settings. A callout box with the text '1. Max Epochを“10”に変更' points to the 'Max Epoch' input field, which currently contains the value '10'. Other visible settings include 'Batch Size' (64), 'Precision' (Float), and 'Monitor Interval' (10 epoch). The 'Save Best' checkbox is checked.

※変更しないと100に設定されているため、計算が1時間以上かかることがあります。

学習の実行

学習の実行を行い、結果が表示されることを確認。

1.“EDIT”をクリック

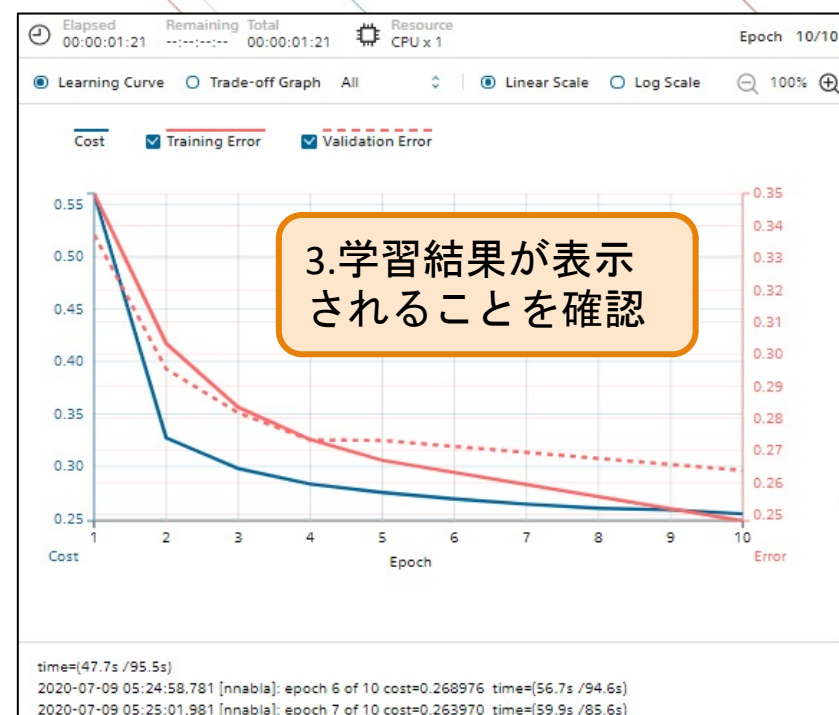
Controller

Run

Profile

Standard	ABCI
☒ CPU 10 H Free	x 1
☐ NVIDIA® Tesla® K80 GPU	x 1
☐ NVIDIA® Tesla® V100 GPU	x 1
☐ NVIDIA® Tesla® V100 GPU	x 4
☐ NVIDIA® Tesla® V100 GPU	x 8
☐ NVIDIA® Tesla® V100 GPU+	x 8

2.“Run”をクリック



3.学習結果が表示されることを確認

レイヤー説明：Affine

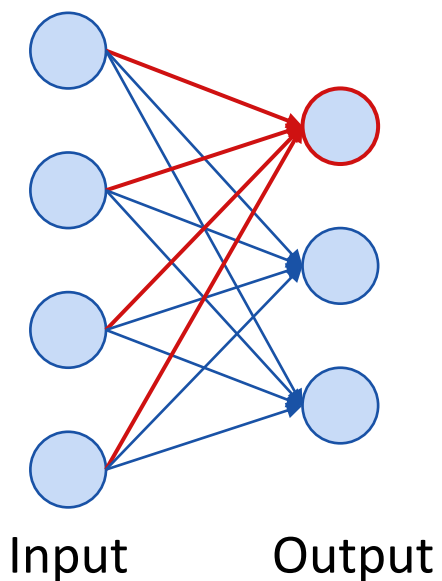
入力層の全ての値を用いた線形変換を行うレイヤーである。

Outputのサイズを任意に指定でき、レイヤーのサイズを自在に調整できる。

Outputの各セルはInputの全ての値が入力値として利用される。

$$y = Wx + b$$

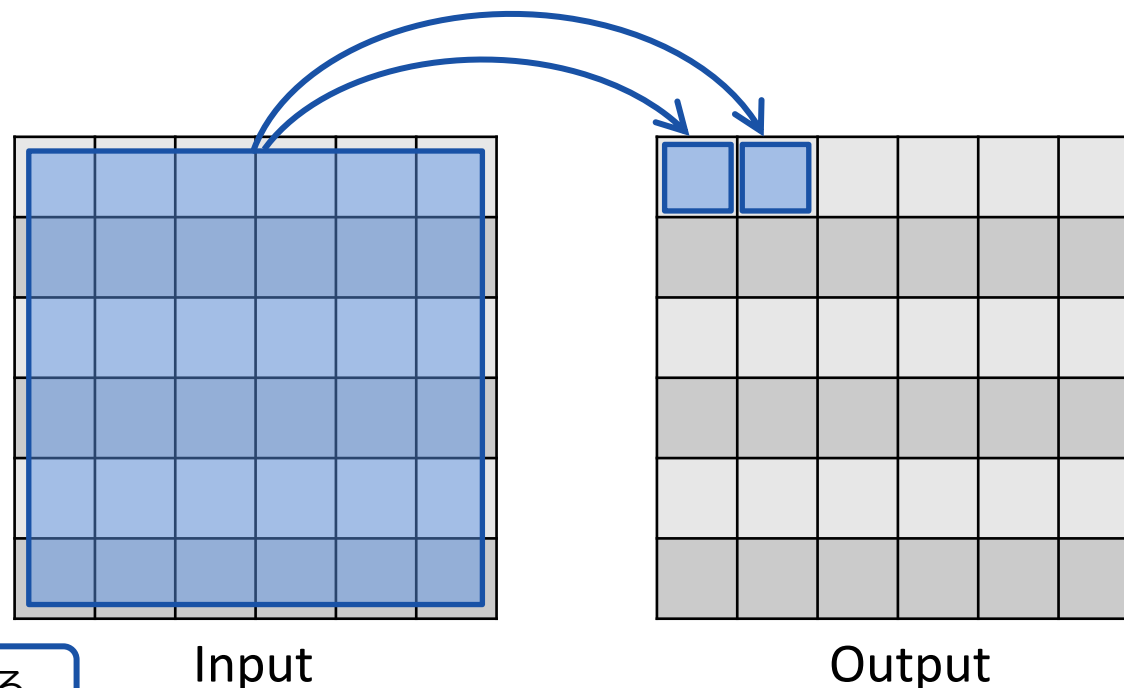
1次元→1次元の例



全結合層

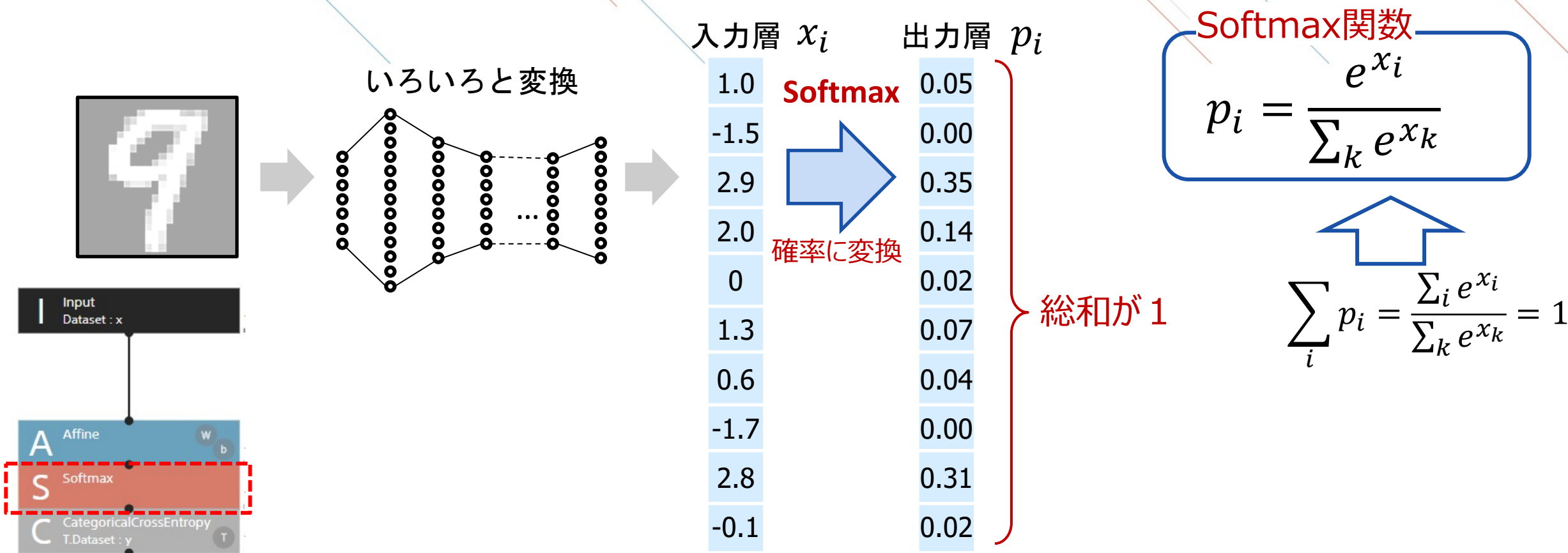
出力ニューロンは全ての入力ニューロンの信号を受け取る

2次元→2次元の例



レイヤー説明：Softmax

入力層のそれぞれの値を各セルの確率に変換するレイヤーである。
出力層の各セルの値を0～1に、セルの総和を1にすることが可能である。
分類問題でのネットワークの最後に使用する。



レイヤー説明：Loss Function（分類問題の場合）

予測値と真値との誤差で、ネットワークの性能の悪さを示す指数を出力するレイヤーである。
学習ではこれを小さくするようにパラメータをアップデートする。

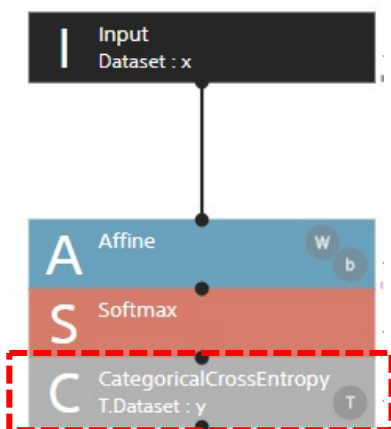
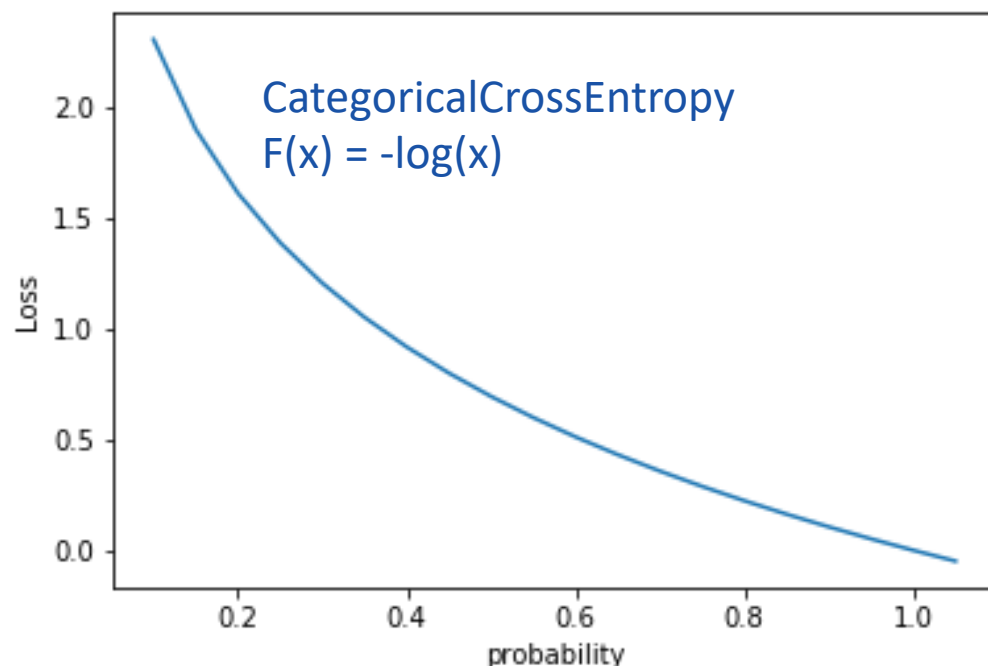
CategoricalCrossEntropy

Loss Function: $L = -\log(x_y)$

x_y : 真のクラスの予測確率

L が大きい $\rightarrow$ 予測が真のクラスから遠い

L が小さい $\rightarrow$ 予測が真のクラスに近い





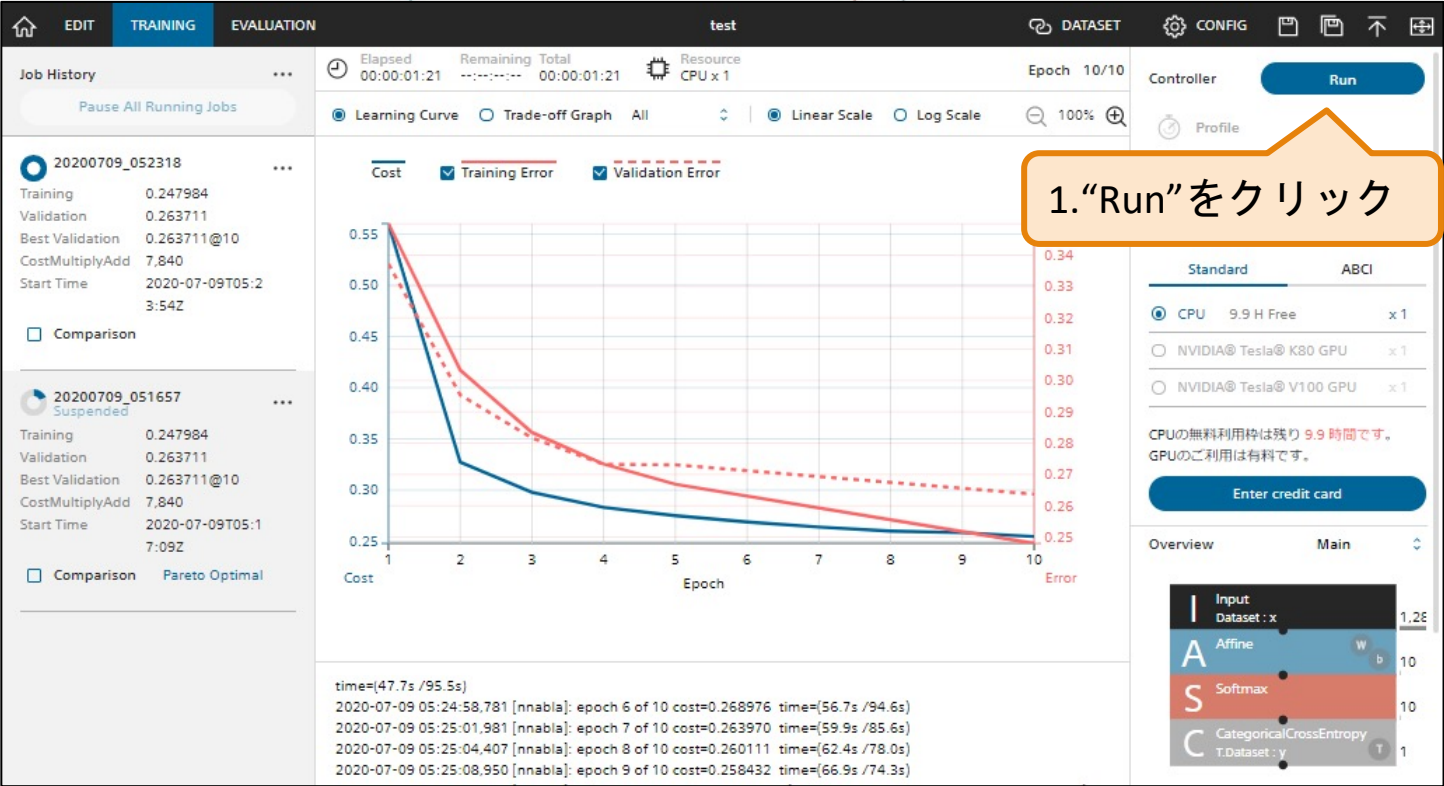
ネットワークの最後の活性化関数とロス関数

課題に合わせて最後の活性化関数とロス関数を設定する

	2 値分類問題	多値分類問題	回帰問題 (連続値などの数値の予測)
最後の活性化関数	Sigmoid 入力値を0.0～ 1 .0 (確率) にする	Softmax 入力値の合計が1.0となる0.0～ 1.0 (確率) にする	(使用しない)
ロス関数	BinaryCrossEntropy 出力と正解が全体としてどれくらい離れているかを計算	CategoricalCrossEntropy 出力と正解カテゴリーが全体としてどれくらい離れているかを計算	SquareError 出力と正解の値の差の 2 乗 (2 乗誤差) を計算
代表的なネットワークの例			

評価の実行

学習結果が表示されている画面で、再度”Run”をクリックし評価を実行。



The screenshot shows the evaluation results table with columns: Index, x:image, y:label, y'_0, y'_1, y'_2, y'_3. An orange callout box labeled '2. 評価結果が表示されることを確認' points to the table.

Index	x:image	y:label	y'_0	y'_1	y'_2	y'_3
1		7	4.351314e-06	1.9141853e-11	7.842366e-06	0.0
2		2	0.0	0.0	0.0	0.0
3		1	2.3040302e-06	0.9779643	0.012287794	0.0

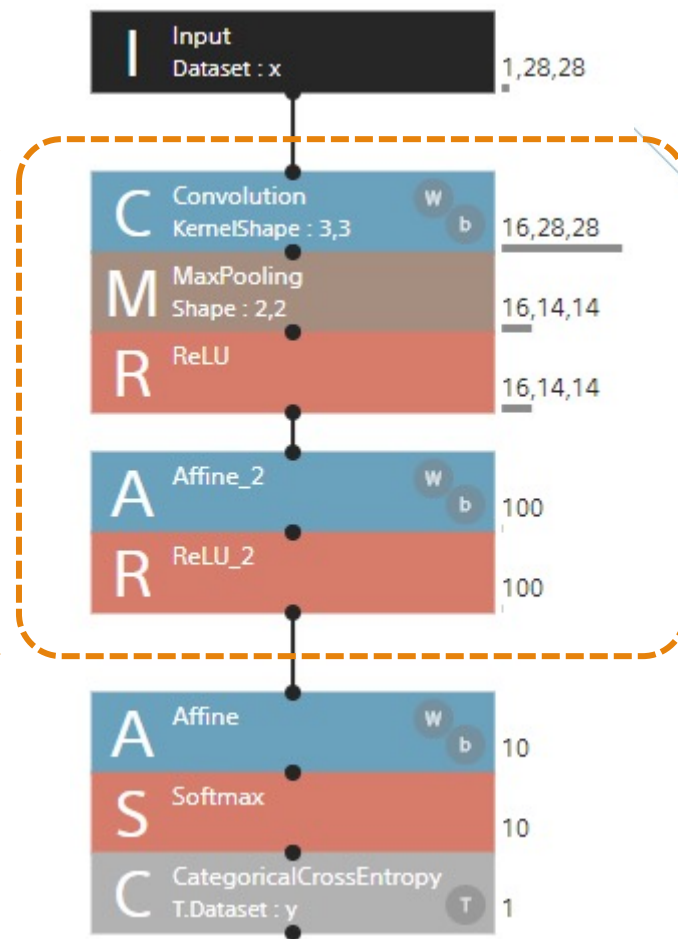
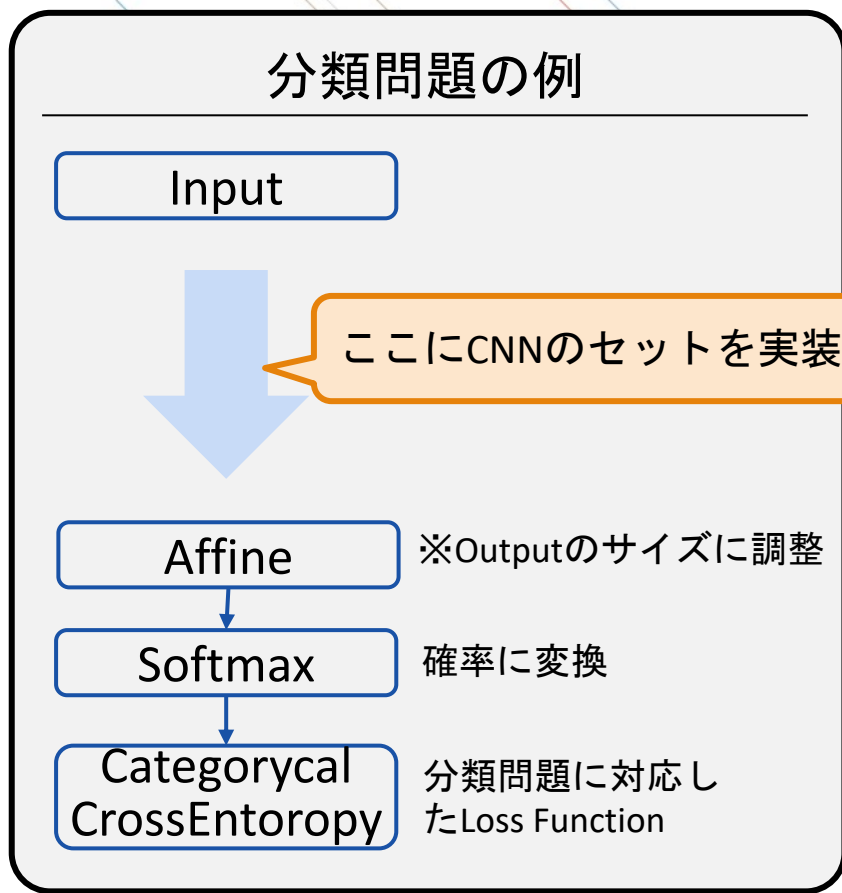
The screenshot shows the evaluation results table with columns: Accuracy, Avg.Precision, Avg.Recall, Avg.F-Measures, Recall, y'_0, y'_1, y'_2, y'_3, y'_4, y'_5, y'_6, y'_7, y'_8, y'_9. An orange callout box labeled '3. 混同行列' points to the table.

Metric	Value
Accuracy	0.9271
Avg.Precision	0.9263
Avg.Recall	0.9259
Avg.F-Measures	0.926

CNNの実装

ネットワークを変更し試行錯誤を実施。

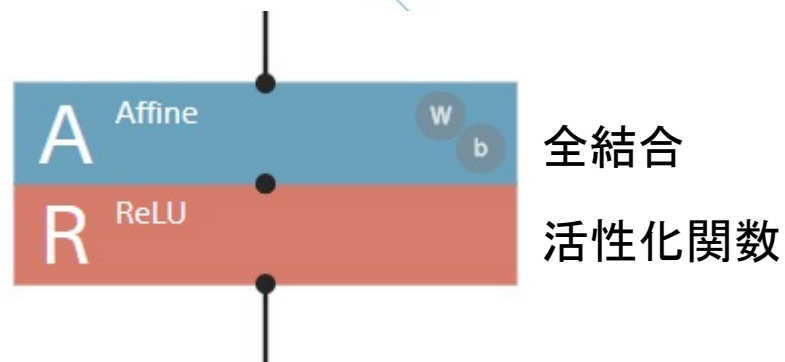
画像分類で用いられることが多いConvolutional Neural Network (CNN) を実装する。



よく利用されるレイヤーの組み合わせ

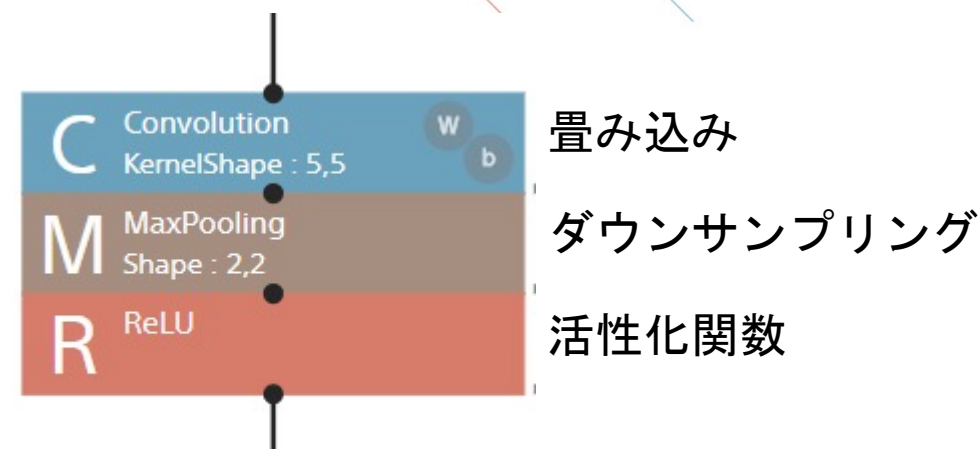
ネットワークを作成する際は、レイヤーを組み合わせたセットを最小単位として考えることが多い。

Affineのセットの例



- 活性化関数はtanhなど他を使う場合もある

Convolutionのセットの例



- 活性化関数はtanhなど他を使う場合もある
- Pooling層はSumPoolingなど他を使う場合もある

レイヤー説明： Convolution

入力データを一定サイズごとと順番に線形変換を行うレイヤーである。
 画像の局所的な特徴を抽出可能である。

$$y = \sum_{i,j} w_{ij} x_{ij} + b$$

※ 1 枚のlayerに対しては同じ変換

変換する入力
 (カーネル)のサイズ
 (KernelShape)

⇒ 3x3

カーネルの移動量
 (Stride)

⇒ 1

Padding
 (1,1)

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

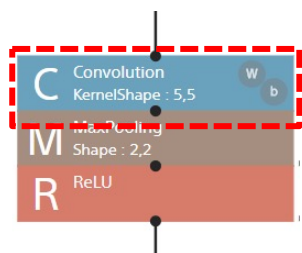
各カーネルに
 重みをかけて
 和をとる

Output

Outputの枚数
 (OutMaps)

畳み込み層

局所的なニューロンの入力を元に
 出力ニューロンの値を求める

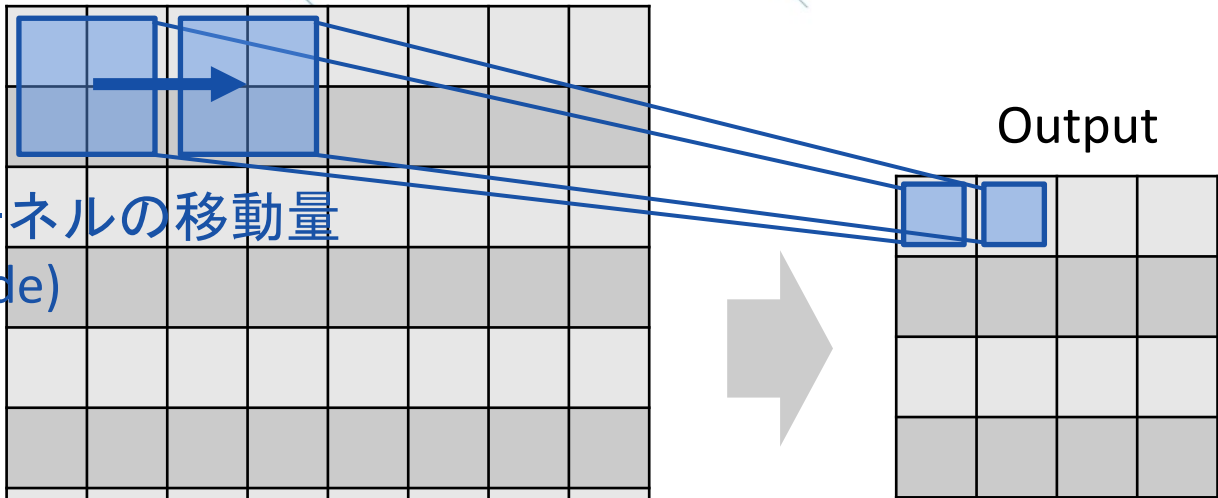


レイヤー説明： Pooling

入力データを一定サイズごとに順に変換するレイヤーである。
カーネルのサイズと移動量をそろえて、画像サイズを圧縮することが一般的である。

カーネルのサイズ
(KernelShape) Input

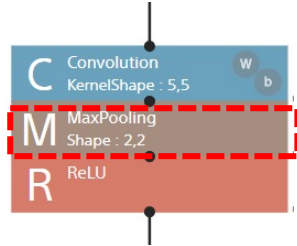
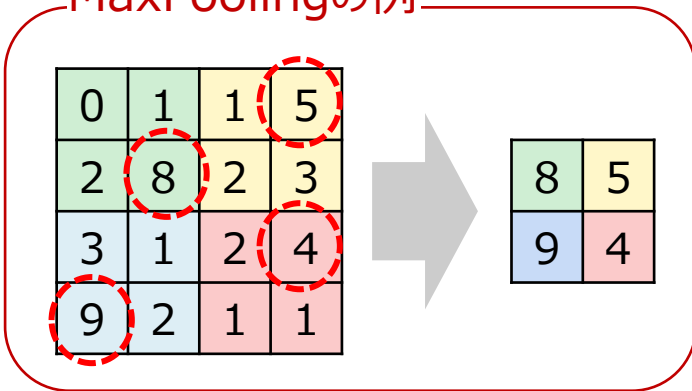
カーネルの移動量
(Stride)



Pooling層の種類

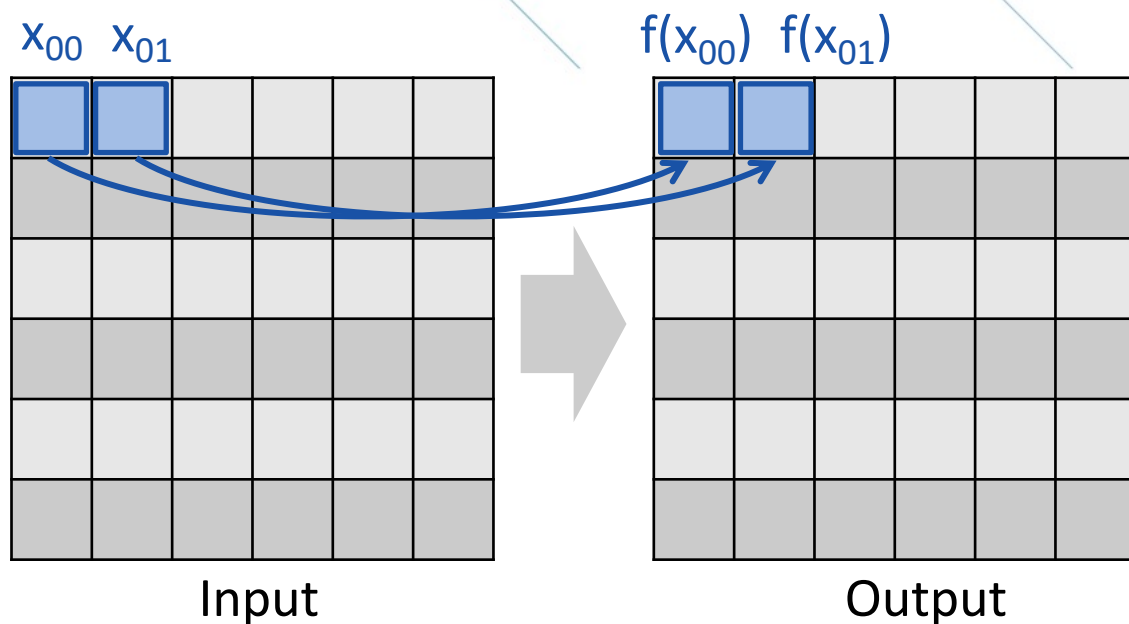
Layer	変換内容
MaxPooling	カーネルの中の最大値を取得
AveragePooling	カーネルの中の平均値を取得
SumPooling	カーネルの中の総和を取得

MaxPoolingの例



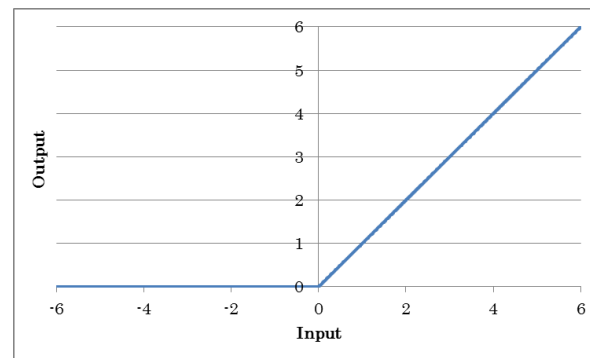
レイヤー説明：Activate (ReLU, Tanh)

インプットの各セルに対して、非線形変換※を行うレイヤーである。
 ネットワークの途中に、このActivate Functionを入れることが精度を出すためには必須である。
 挿入する場所はConvolution層やPooling層の後に挿入することが一般的である。



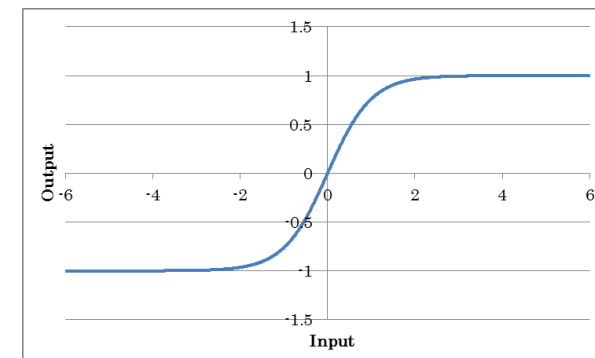
ReLU

$$f(x) = \max(0, x)$$



Tanh

$$f(x) = \tanh(x)$$



※非線形変換とはAffineやConvolutionとは異なり、1次関数で表すことができない変換

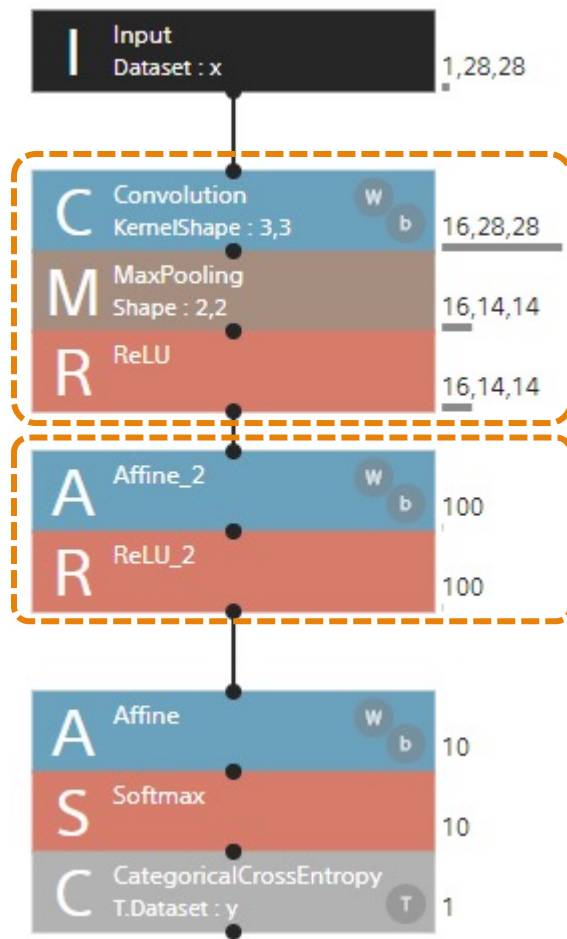
CNNの実装

Convolution層とAffine-ReLU層の組み合わせとConvolutionのパラメータ(OutMaps, KernelShape)及び、Affineのサイズを変更し、試行錯誤することができる。

各ユニットをクリックすると
左下に設定されている
パラメータが表示されます。

Layer Property

C Convolution	
Name	Convolution
Input	1,28,28
OutMaps	16
KernelShape	3,3
BorderMode	same
Padding	1,1
Strides	1,1
Dilation	1,1



- Convolution層の数
- Convolutionのパラメータ (OutMaps, KernelShape)

- Affine-ReLU層の数
- Affineのサイズ

ネットワークの試行錯誤による精度向上

精度向上をするための戦略を決める

- パラメタは以下の5種類

層数

1. Convolution層(CMRのセット)の数
2. Affine-ReLU層(ARのセット)の数

各Convolutionのパラメタ

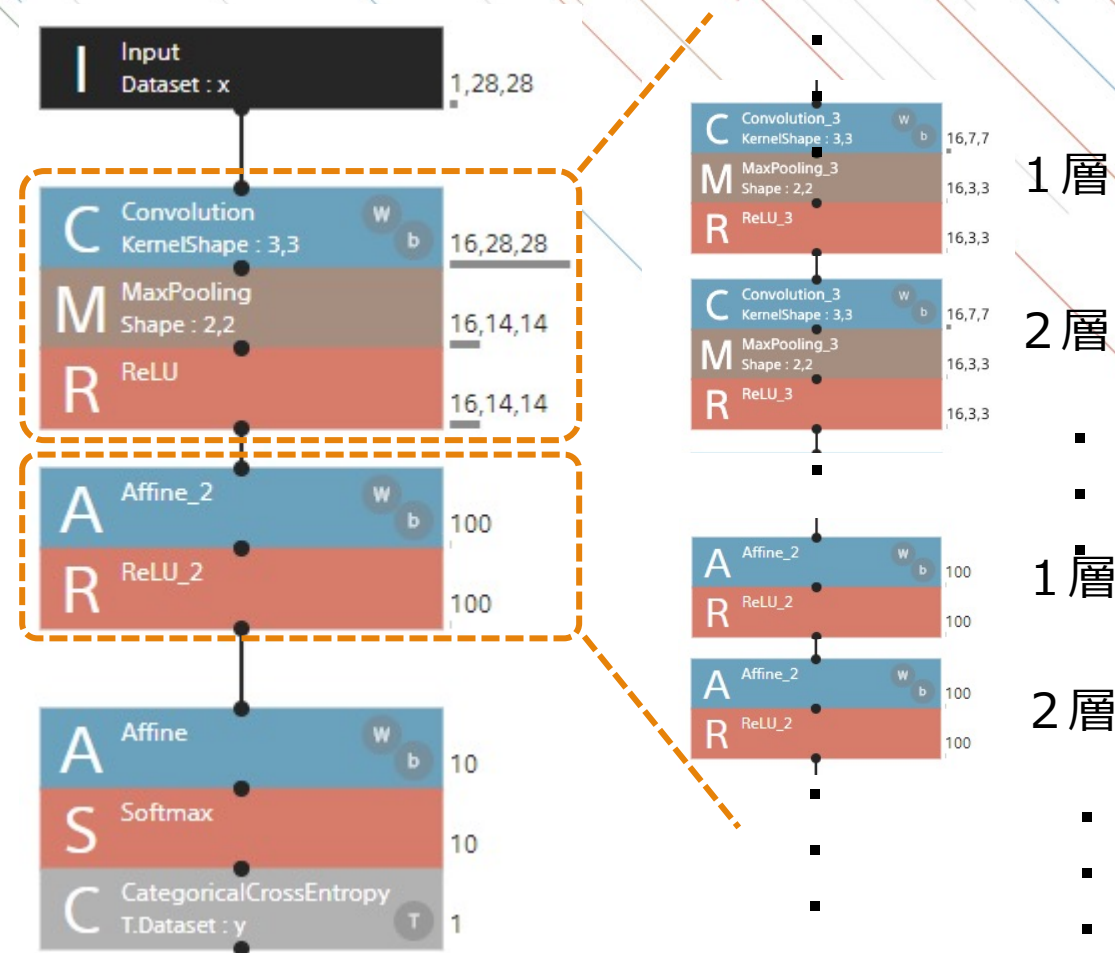
3. OutMaps 初期値: 16
4. KernelShape 初期値:(3,3)

各Affineのサイズ

5. OutShape 初期値: 100

- 例えば...

- (CMR:1~3層) x (AR:1,3,5,7,9層)
→ 15通り → 5人で1人3通り



※「構造自動探索」は時間がかかり、ネットワーク構造自体を探索するので、今回は用いない。



参考資料

ネットワーク作成

レイヤー選択パネルからネットワーク構造パネルにレイヤーをドラック＆ドロップする。
 すでにあるレイヤーと連結し、ネットワークを構成する。
 レイヤーパラメータパネルで各レイヤーの詳細を設定する。

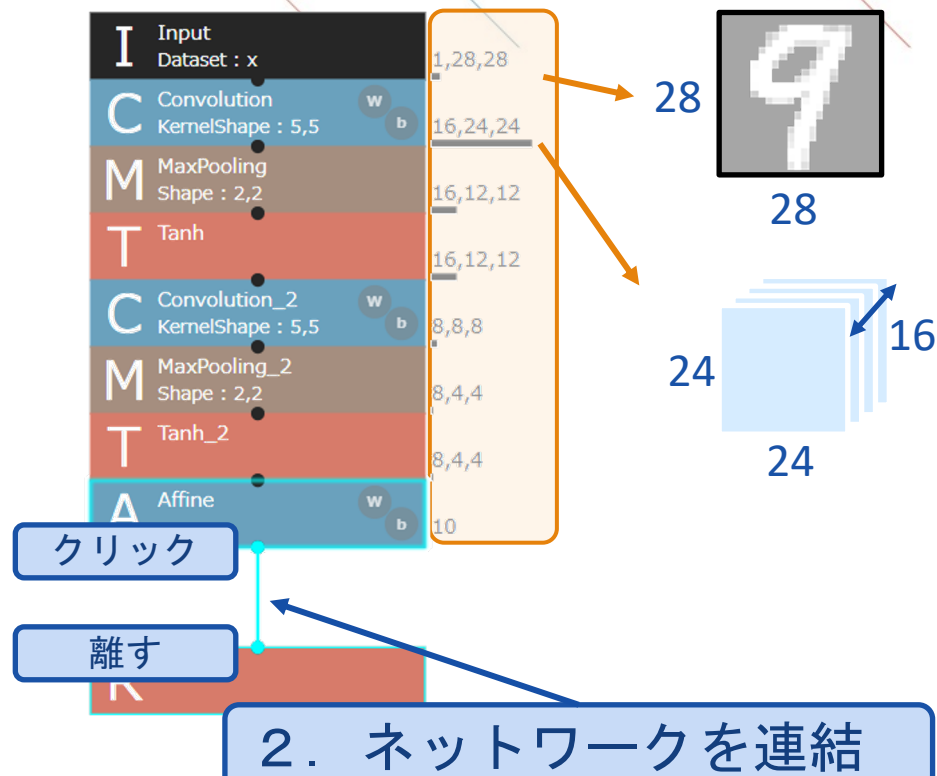
The screenshot shows the neural network builder interface with three main panels:

- Components Panel (Left):** Contains a search bar and a list of layers and loss functions. A blue arrow points from the "Convolution" layer to the "1. レイヤーを選択" (Select Layer) instruction box.
- Main Panel (Center):** Displays the network structure as a vertical stack of layers: Input (Dataset: x), Convolution (KernelShape: 5,5), MaxPooling (Shape: 2,2), Tanh_2, Affine, Tanh_3, Affine_2, Sigmoid, and BinaryCrossEntropy (T.Dataset: y). A blue arrow points from the "レイヤー設定" (Layer Setting) instruction box to the "Convolution" layer.
- Layer Property Panel (Bottom Left):** Shows the detailed settings for the selected "Convolution" layer, including Name, Input, OutMaps, KernelShape, BorderMode, and Padding.
- Controller Panel (Right):** Contains a "Run" button, a "Profile" section, and a list of hardware configurations (CPU, NVIDIA Tesla K80, V100, etc.).
- Overview Panel (Bottom Right):** Provides a summary of the network structure and statistics.

Annotations on the screenshot include:

- 1. レイヤーを選択** (Select Layer): A blue box with an arrow pointing to the "Convolution" layer in the Main panel.
- レイヤー設定** (Layer Setting): A blue box with an arrow pointing to the "Convolution" layer in the Layer Property panel.
- ネットワーク構造** (Network Structure): A blue box with an arrow pointing to the Main panel.
- 学習指示** (Learning Instruction): A blue box with an arrow pointing to the "Run" button in the Controller panel.
- ネットワーク統計量** (Network Statistics): A blue box with an arrow pointing to the Overview panel.

データの次元を表現



ネットワークを形成するレイヤー

約100種類のレイヤーが実装されており、これらを組み合わせてネットワークを作成する。

IO
Input
Loss
SquaredError
HuberLoss
BinaryCrossEntropy
SigmoidCrossEntropy
CategoricalCrossEntropy
SoftmaxCrossEntropy
KLMultinomial
Parameter
Parameter
WorkingMemory
Basic
Affine
Convolution
Deconvolution
Embed
Pooling
MaxPooling
AveragePooling
SumPooling
Unpooling

Activation
Tanh
Sigmoid
Abs
ReLU
CReLU
PReLU
ELU
CELU
Softmax
LoopControl
RepeatStart
RepeatEnd
ReccurentInput
ReccurentOutput
Delay
Binary
BinaryConnectAffine
BinaryConnectConvolution
BinaryWeightAffine
BinaryWeightConvolution
BinaryTanh
BinarySigmoid
Unit
LSTM

Math
Sum
Mean
Prod
Max
Min
Log
Exp
Sign
Arithmetic(Scalar)
AddScalar
MulScalar
RSubScalar
RDivScalar
PowScalar
RPowScalar
MaximumScalar
MinimumScalar
Arithmetic(2 Inputs)
Add2
Sub2
Mul2
Div2
Pow2
Maximum2
Minimum2

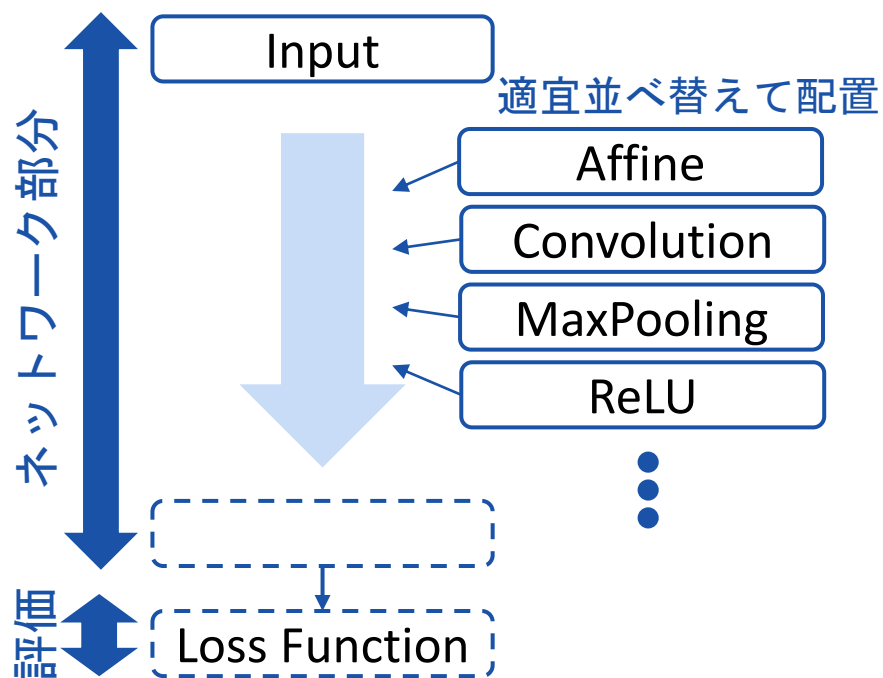
Logical
LogicalAnd
LogicalOr
LogicalXor
Equal
NotEqual
GreaterEqual
Greater
LessEqual
Less
LogicalAndScalar
LogicalOrScalar
LogicalXorScalar
EqualScalar
NotEqualScalar
GreaterEqualScalar
GreaterScalar
LessEqualScalar
LessScalar
LogicalNot
Validation
BinaryError
TopNError

Others
BatchNormalization
Dropout
Concatenate
Reshape
Broadcast
Flip
Shit
Transpose
Slice
Stack
VATNoise
Unlink
Identity
Comment
Others(Pre Process)
OneHot
RandomCrop
RandomFlip
RandomShift
MeanSubtraction
ImageAugmentation

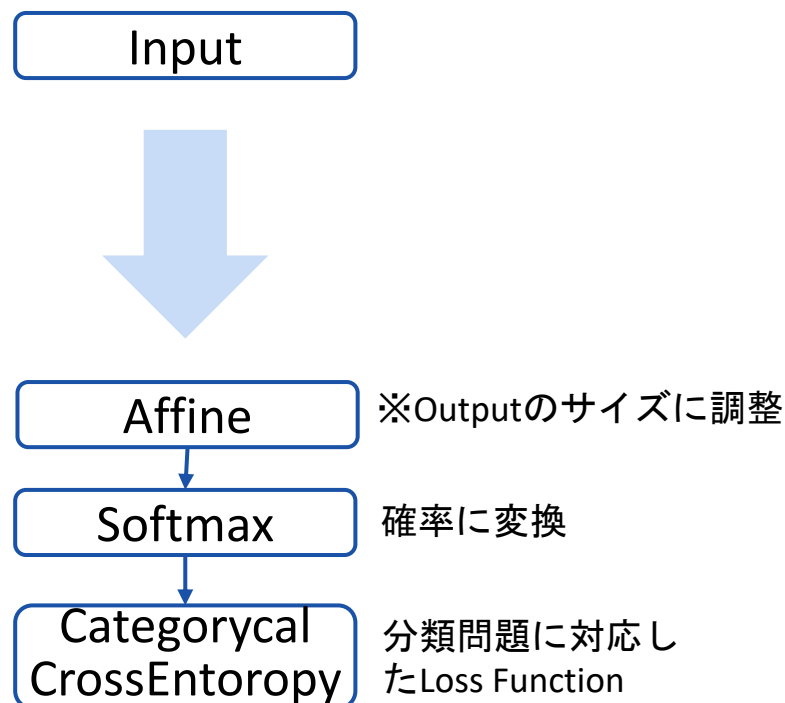
レイヤーの組み合わせ方

Inputから始まって、Loss Function(ネットワークの精度を評価する層)で終わる。
 Loss Functionとその直前のレイヤーは問題設定によって適宜修正する必要がある。
 それ以前の部分に関しては、問題設定によらず使いまわすことが可能である。

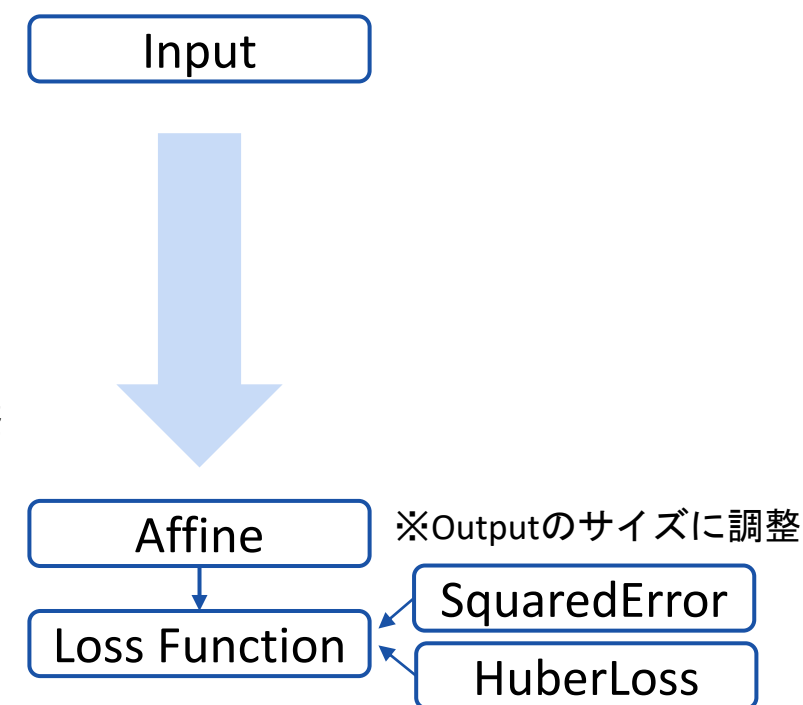
一般的な配置方法



分類問題の例



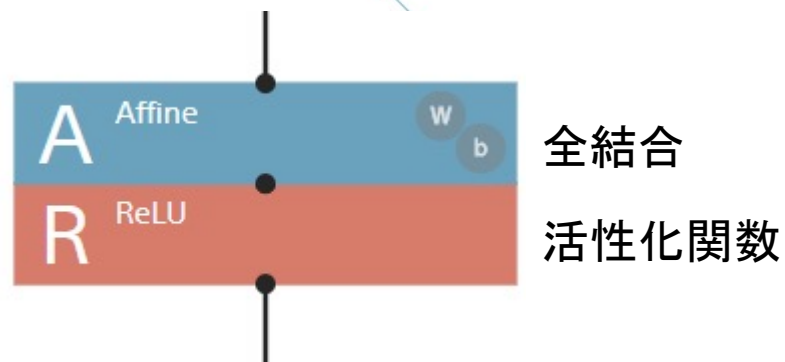
回帰問題の例



よく利用されるレイヤーの組み合わせ

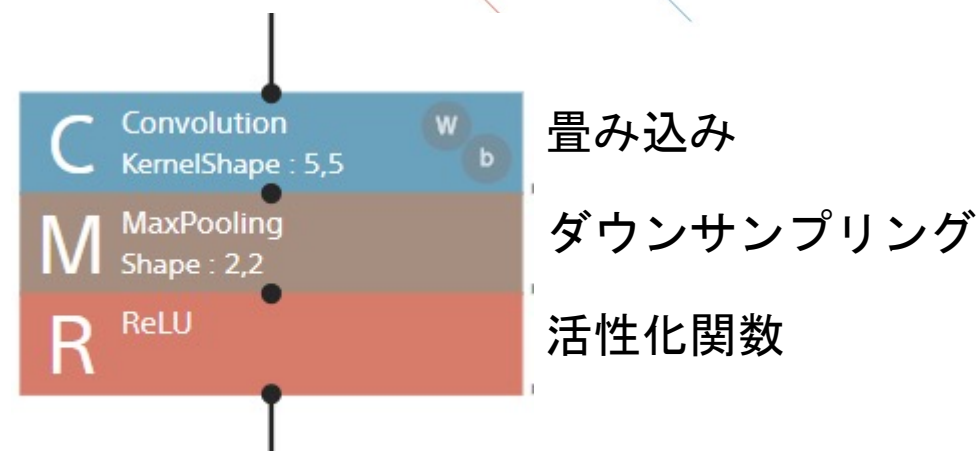
ネットワークを作成する際は、レイヤーを組み合わせたセットを最小単位として考えることが多い。

Affineのセットの例



- 活性化関数はtanhなど他を使う場合もある

Convolutionのセットの例



- 活性化関数はtanhなど他を使う場合もある
- Pooling層はSumPoolingなど他を使う場合もある

ネットワークを形成するレイヤー

約100種類のレイヤーが実装されており、これらを組み合わせてネットワークを作成する。
代表的なレイヤー（青枠で囲った部分）を次頁以降で説明する。

IO
Input
Loss
SquaredError
HuberLoss
BinaryCrossEntropy
SigmoidCrossEntropy
CategoricalCrossEntropy
SoftmaxCrossEntropy
KLMultinomial
Parameter
Parameter
WorkingMemory
Basic
Affine
Convolution
Deconvolution
Embed
Pooling
MaxPooling
AveragePooling
SumPooling
Unpooling

Activation
Tanh
Sigmoid
Abs
ReLU
CReLU
PReLU
ELU
CELU
Softmax
LoopControl
RepeatStart
RepeatEnd
ReccurentInput
ReccurentOutput
Delay
Binary
BinaryConnectAffine
BinaryConnectConvolution
BinaryWeightAffine
BinaryWeightConvolution
BinaryTanh
BinarySigmoid
Unit
LSTM

Math
Sum
Mean
Prod
Max
Min
Log
Exp
Sign
Arithmetic(Scalar)
AddScalar
MulScalar
RSubScalar
RDivScalar
PowScalar
RPowScalar
MaximumScalar
MinimumScalar
Arithmetic(2 Inputs)
Add2
Sub2
Mul2
Div2
Pow2
Maximum2
Minimum2

Logical
LogicalAnd
LogicalOr
LogicalXor
Equal
NotEqual
GreaterEqual
Greater
LessEqual
Less
LogicalAndScalar
LogicalOrScalar
LogicalXorScalar
EqualScalar
NotEqualScalar
GreaterEqualScalar
GreaterScalar
LessEqualScalar
LessScalar
LogicalNot
Validation
BinaryError
TopNError

Others
BatchNormalization
Dropout
Concatenate
Reshape
Broadcast
Flip
Shit
Transpose
Slice
Stack
VATNoise
Unlink
Identity
Comment
Others(Pre Process)
OneHot
RandomCrop
RandomFlip
RandomShift
MeanSubtraction
ImageAugmentation

レイヤー説明：Input

ネットワーク全体の入力層にあたるレイヤーである。
パラメータとして入力データのサイズを入力する必要がある。
指定したデータサイズと入力データのサイズが合わない場合、学習時にエラーが発生する。

データサイズの例

表データ

入力サイズ：100

個人属性情報 100カラム						予測値
ID	性別	年齢	居住地	収入	...	購入頻度
00001	男性	20代	東京	700万		高
00002	男性	30代	埼玉	600万		低
00003	女性	20代	東京	300万		中
00004	女性	10代	東京	NA		中
00005	男性	20代	千葉	400万		高

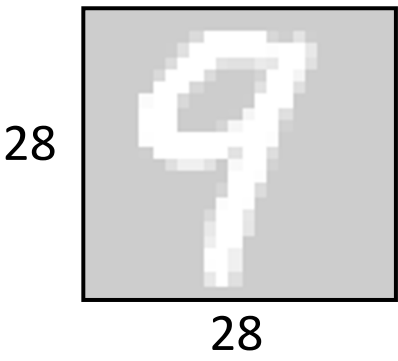
RGB画像

入力サイズ：(3, 256, 128)



白黒画像

入力サイズ：(1, 28, 28)



レイヤー説明：Affine

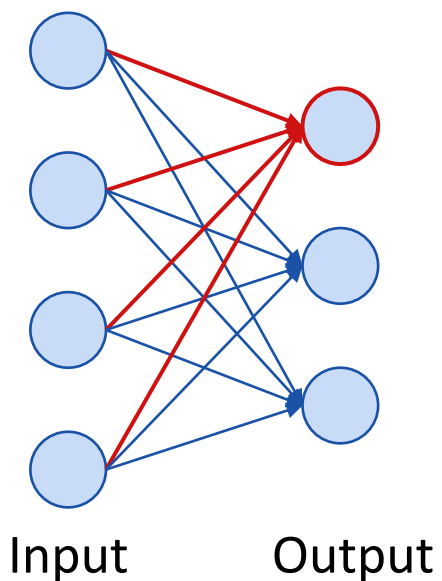
入力層の全ての値を用いた線形変換を行うレイヤーである。

Outputのサイズを任意に指定でき、レイヤーのサイズを自在に調整できる。

Outputの各セルはInputの全ての値が入力値として利用される。

$$y = Wx + b$$

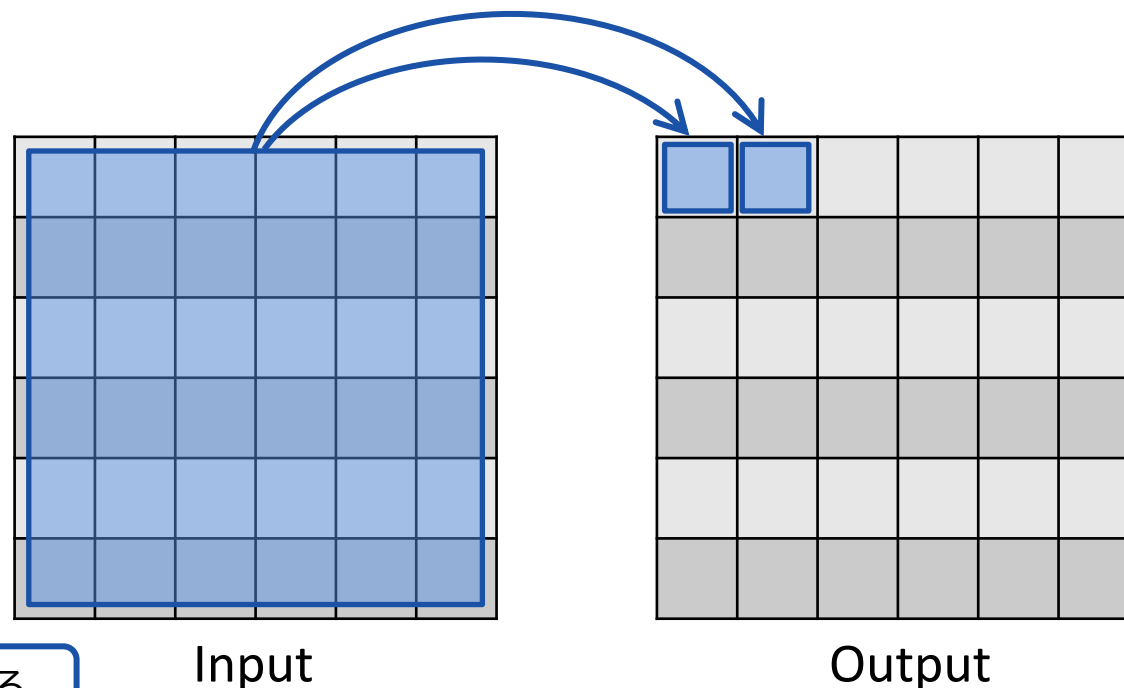
1次元→1次元の例



全結合層

出力ニューロンは全ての入力ニューロンの信号を受け取る

2次元→2次元の例



レイヤー説明： Convolution

入力データを一定サイズごとと順番に線形変換を行うレイヤーである。
 画像の局所的な特徴を抽出可能である。

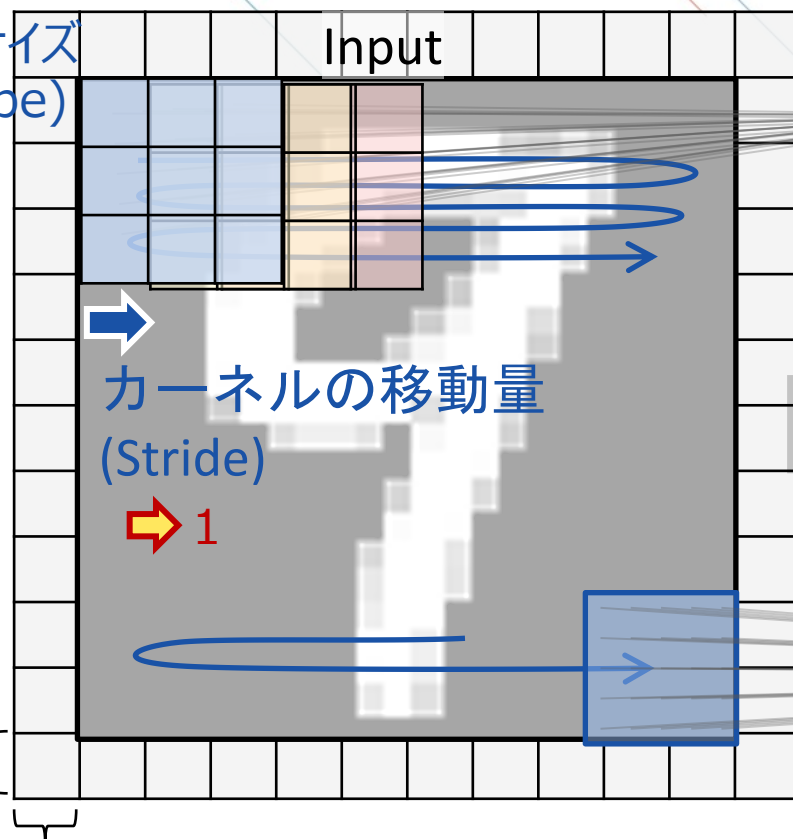
変換する入力
 (カーネル)のサイズ
 (KernelShape)

⇒ 3x3

カーネルの移動量
 (Stride)

⇒ 1

Padding
 (1,1)



Output

Outputの枚数
 (OutMaps)

$$y = \sum_{i,j} w_{ij} x_{ij} + b$$

※ 1 枚のlayerに対しては同じ変換

畳み込み層

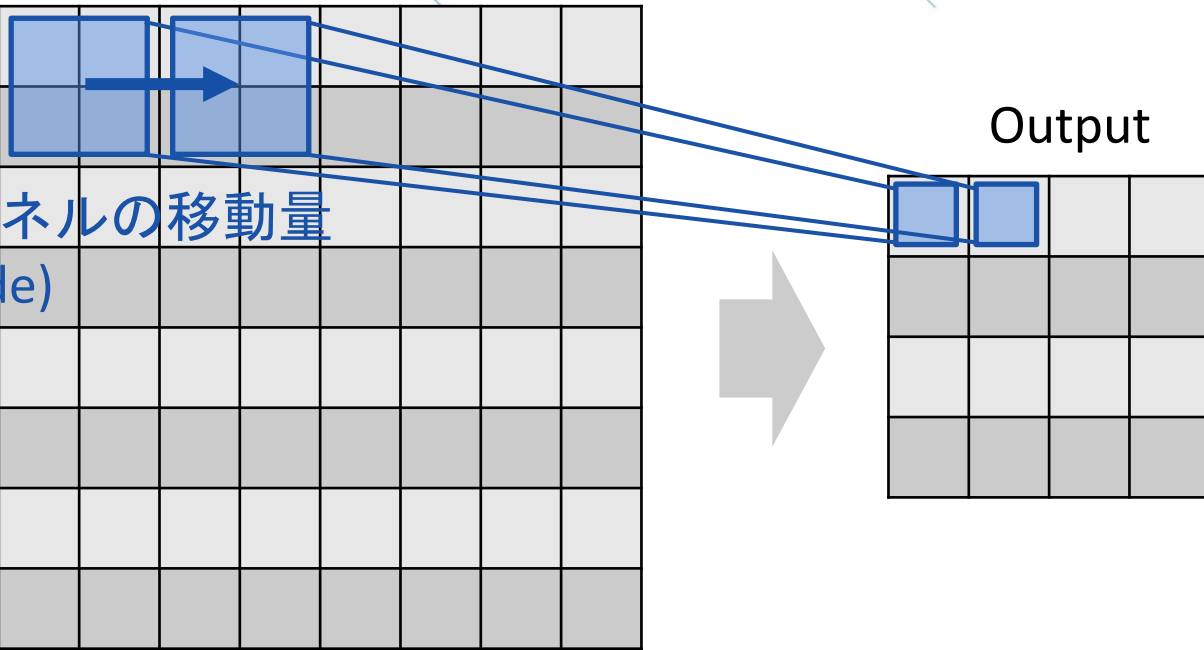
局所的なニューロンの入力を元に
 出力ニューロンの値を求める

レイヤー説明： Pooling

入力データを一定サイズごとに順に変換するレイヤーである。
カーネルのサイズと移動量をそろえて、画像サイズを圧縮することが一般的である。

カーネルのサイズ
(KernelShape) Input

カーネルの移動量
(Stride)



Pooling層の種類

Layer	変換内容
MaxPooling	カーネルの中の最大値を取得
AveragePooling	カーネルの中の平均値を取得
SumPooling	カーネルの中の総和を取得

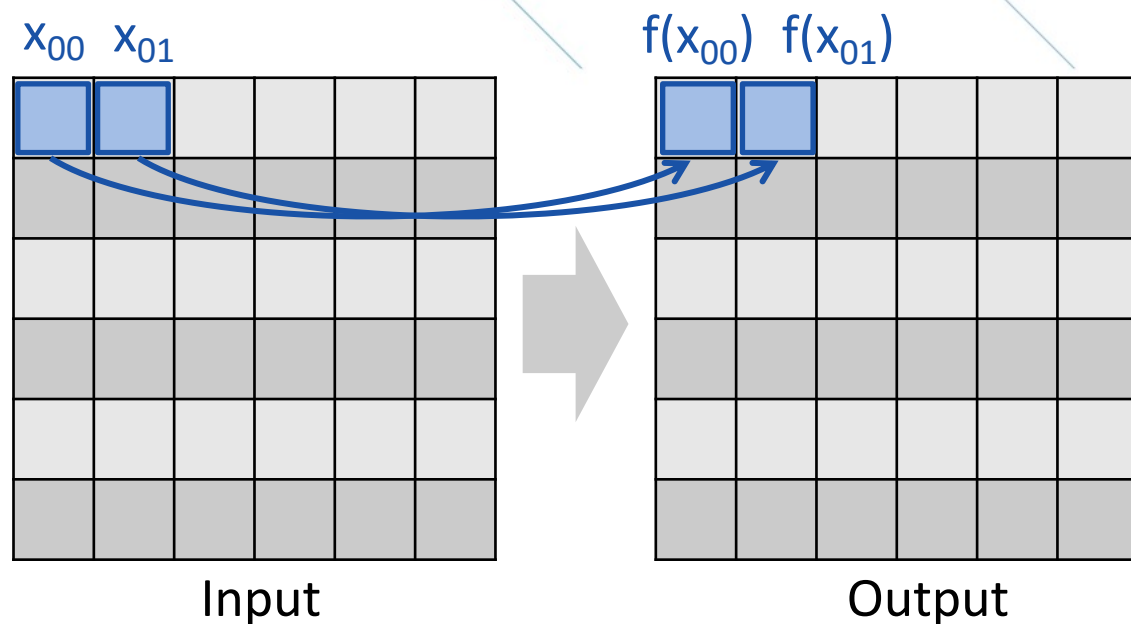
MaxPoolingの例

0	1	1	5
2	8	2	3
3	1	2	4
9	2	1	1

8	5
9	4

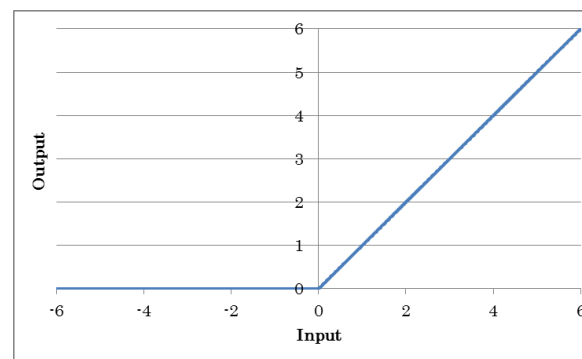
レイヤー説明：Activate (ReLU, Tanh)

インプットの各セルに対して、非線形変換※を行うレイヤーである。
 ネットワークの途中に、このActivate Functionを入れることが精度を出すためには必須である。
 挿入する場所はConvolution層やPooling層の後に挿入することが一般的である。



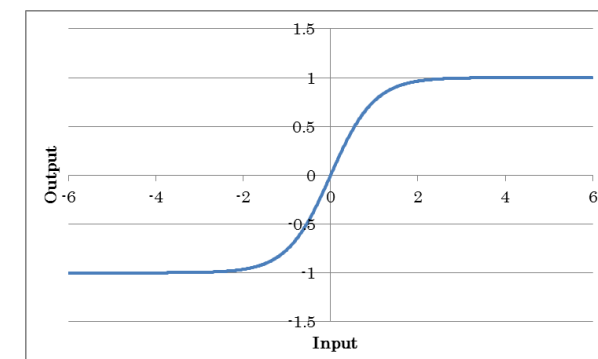
ReLU

$$f(x) = \max(0, x)$$



Tanh

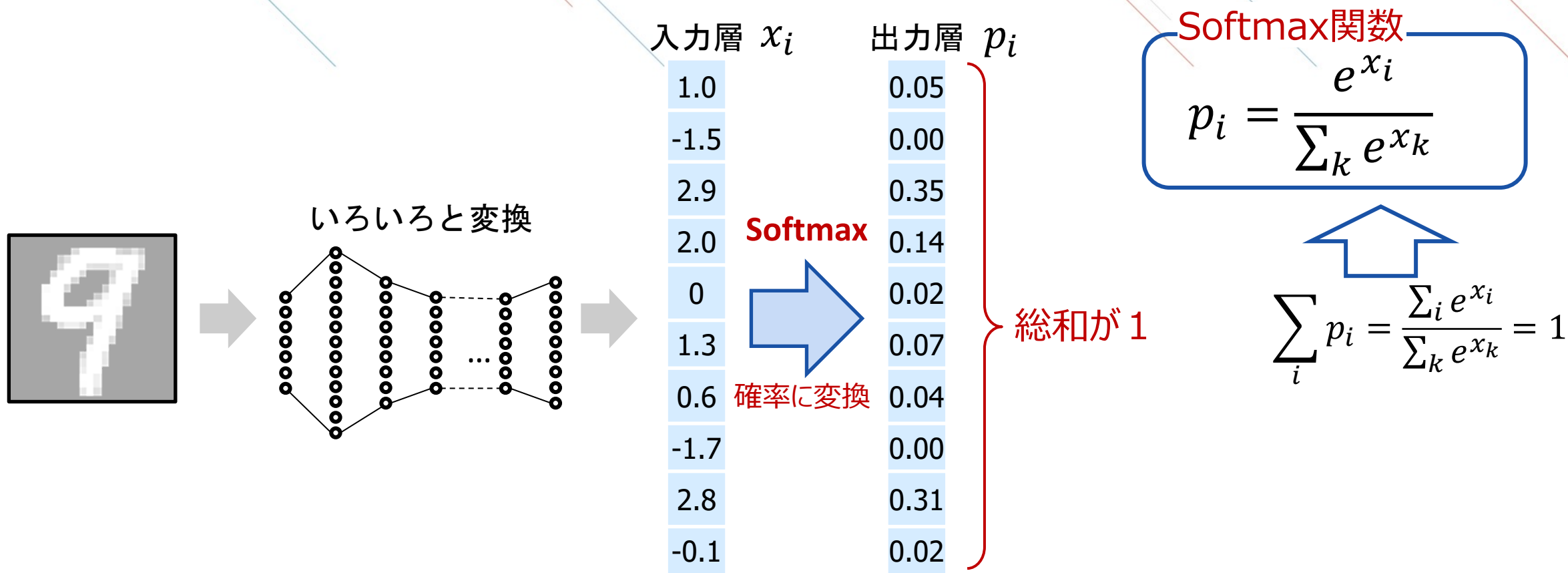
$$f(x) = \tanh(x)$$



※非線形変換とはAffineやConvolutionとは異なり、1次関数で表すことができない変換

レイヤー説明：Softmax

入力層のそれぞれの値を各セルの確率に変換するレイヤーである。
出力層の各セルの値を0～1に、セルの総和を1にすることが可能である。
分類問題でのネットワークの最後に使用する。



レイヤー説明：Loss Function

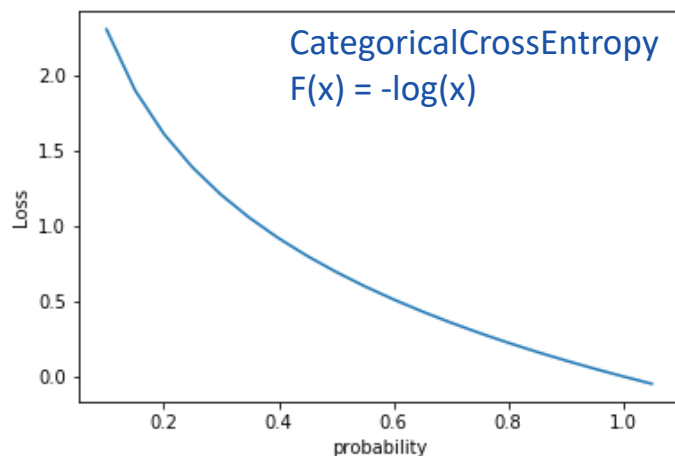
予測値と真値との誤差で、ネットワークの性能の悪さを示す指数を出力するレイヤーである。
学習ではこれを小さくするようにパラメータをアップデートする。

分類問題の場合

CategoricalCrossEntropy

$$L = -\log(x_y)$$

x_y : 真のクラスの予測確率

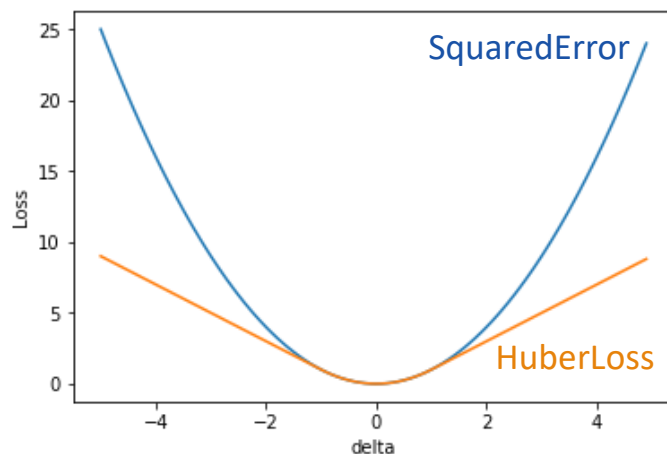


回帰問題の場合

SquaredError

$$L = (x - y)^2$$

x : 予測値、 y : 真の値



SquaredErrorに比べて、HuberLossのほうが外れ値に敏感でない

HuberLoss

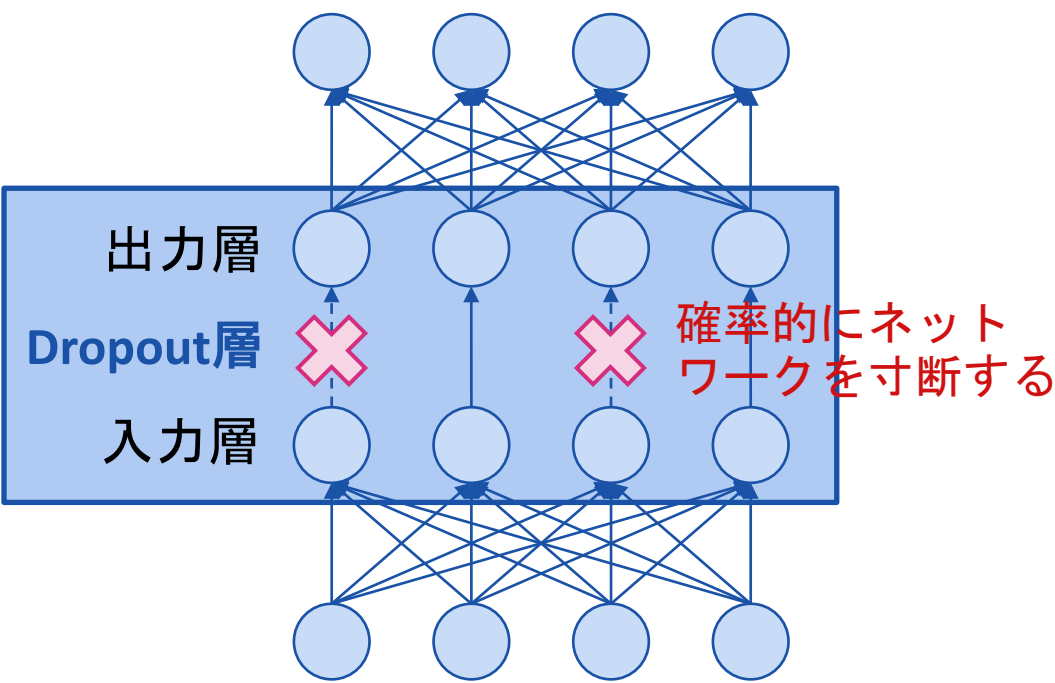
$$L = \begin{cases} (x - y)^2, & |x - y| < d \\ \delta\{2|x - y| - \delta\}, & otherwise \end{cases}$$

x : 予測値、 y : 真の値

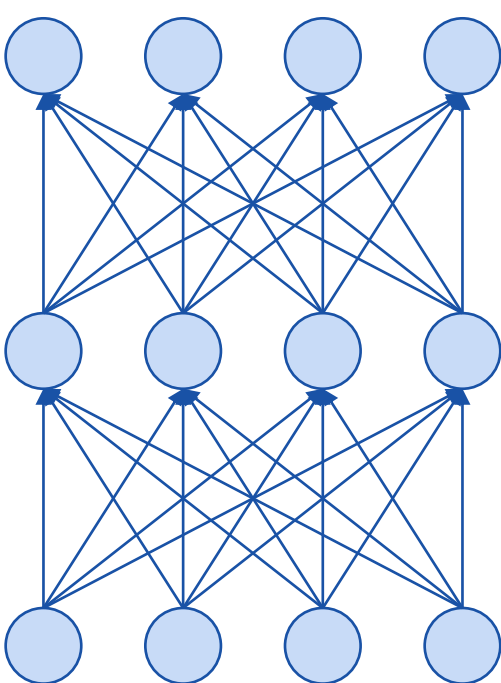
レイヤー説明：Dropout

確率 p で各セルの値をゼロにする変換を行うレイヤーである。
特定のネットワークの学習を阻害でき、過学習に対して有効である。
学習時にのみ作用し、推論時には作用しない。

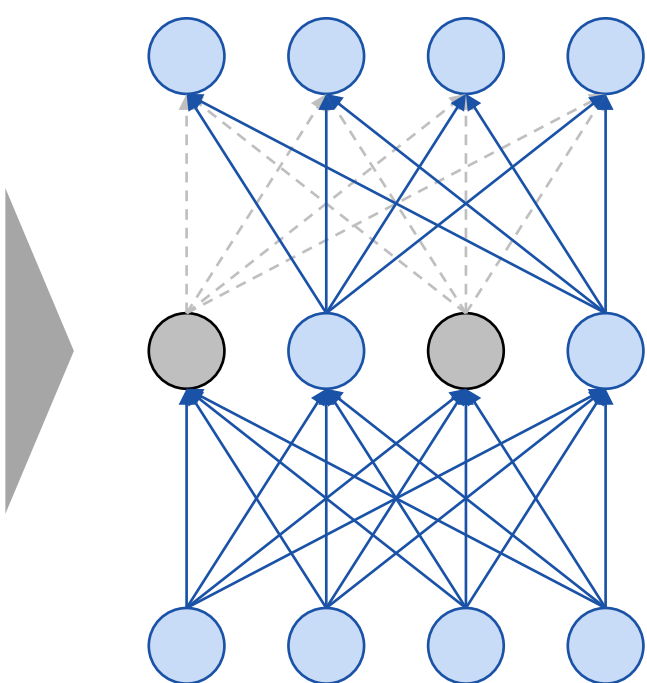
構造の概要



Dropoutなし

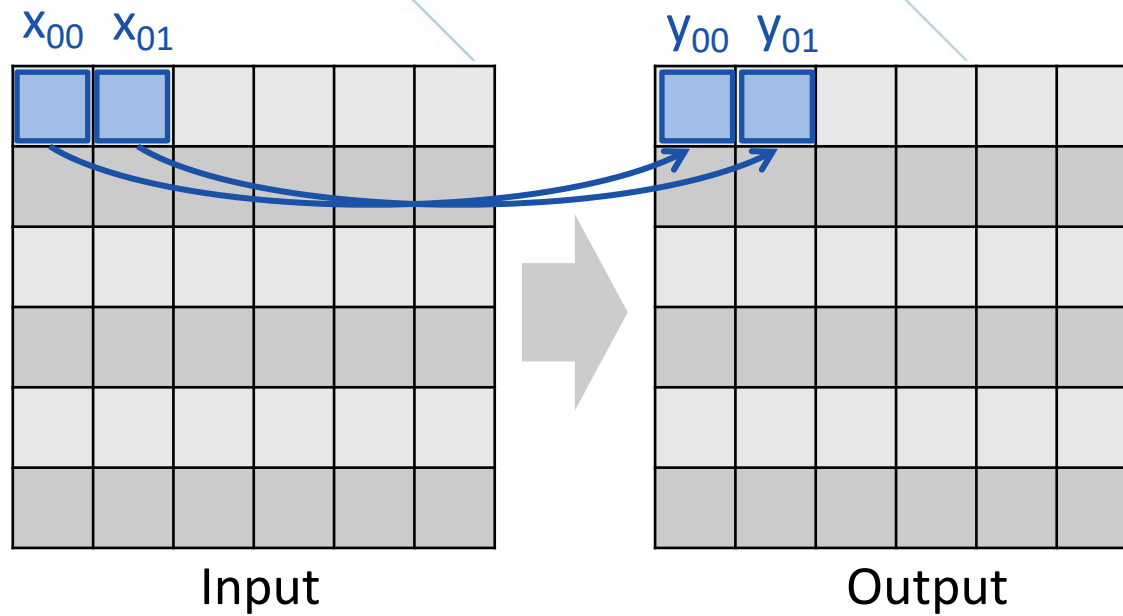


Dropoutあり



レイヤー説明：BatchNormalization

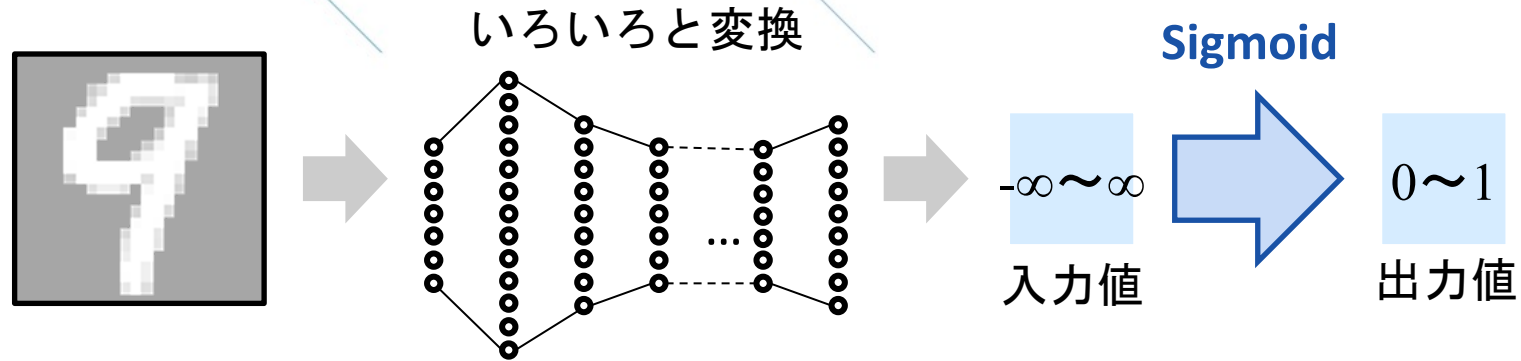
入力層の値をバッチサイズごとに平均を0、標準偏差を1に規格化するレイヤーである。
ネットワークの学習プロセスを全体的に安定化させ、学習速度を高めることに寄与する。



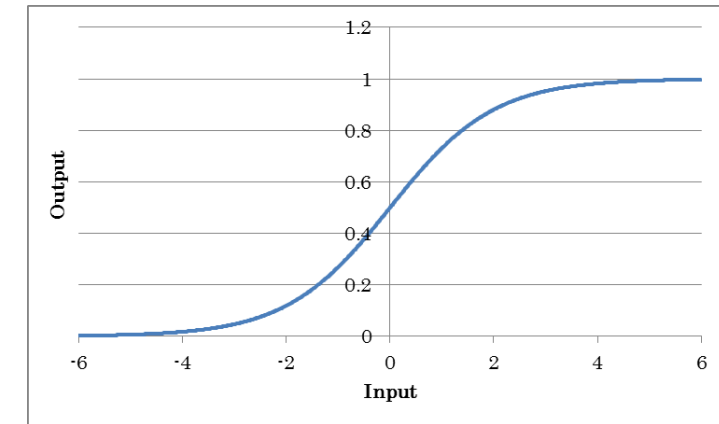
$$\mu = \frac{1}{M} \sum_i x_i$$
$$\sigma = \frac{1}{M} \sum_i (x_i - \mu)^2$$
$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$
$$y = \gamma \hat{x}_i + \beta$$

レイヤー説明：Sigmoid

入力層の値を0～1に変換するレイヤーである。
確率値の回帰問題でのネットワークの最後に使用される。

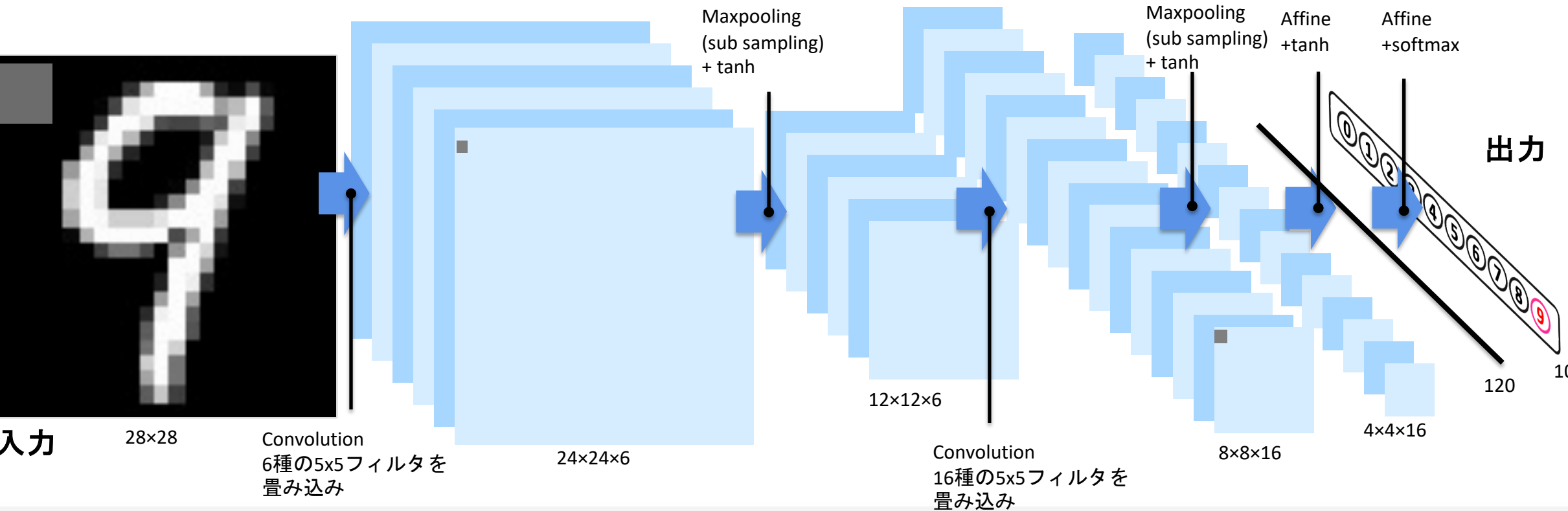
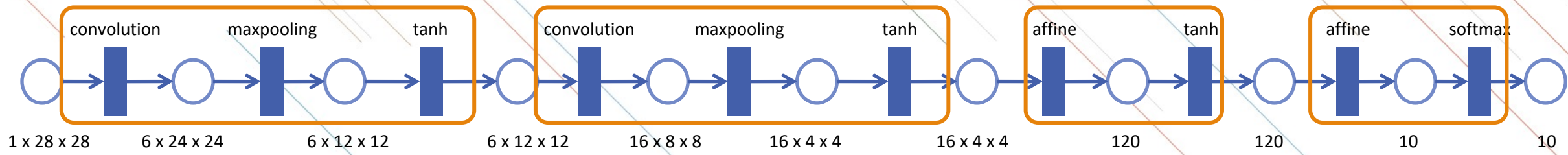


$$f(x) = \frac{1}{1 + e^{-ax}}$$



ネットワーク構造の例（LeNet5）

代表的なネットワークについても、レイヤーを組み合わせたセットで構成されている。





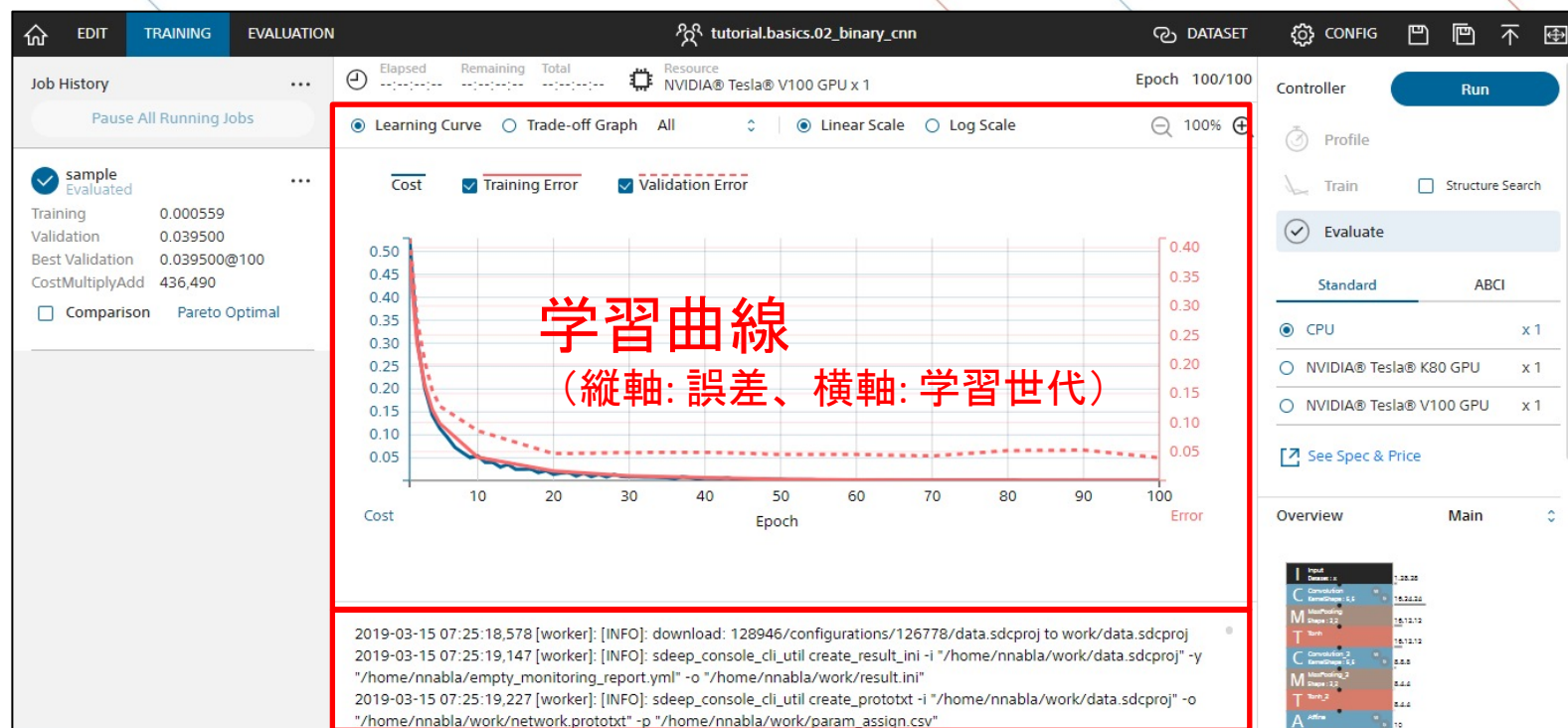
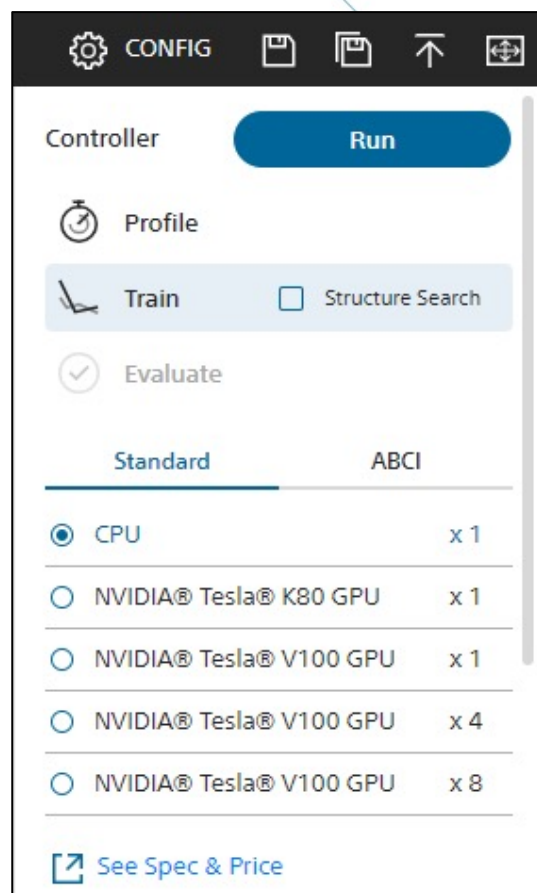
学習の実行

学習の実行

EDITページで学習に使用するインスタンスを選択し、Runボタンで学習が実行される。進捗や結果はTRAININGページで確認できる。

学習実行の方法

TRAININGページの概要



コアエンジンのログ出力

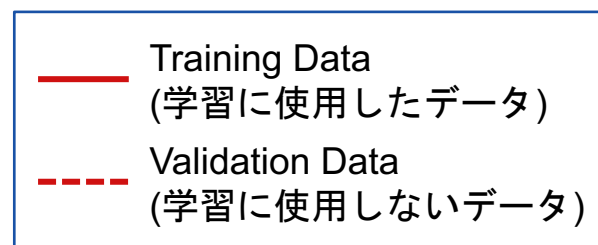
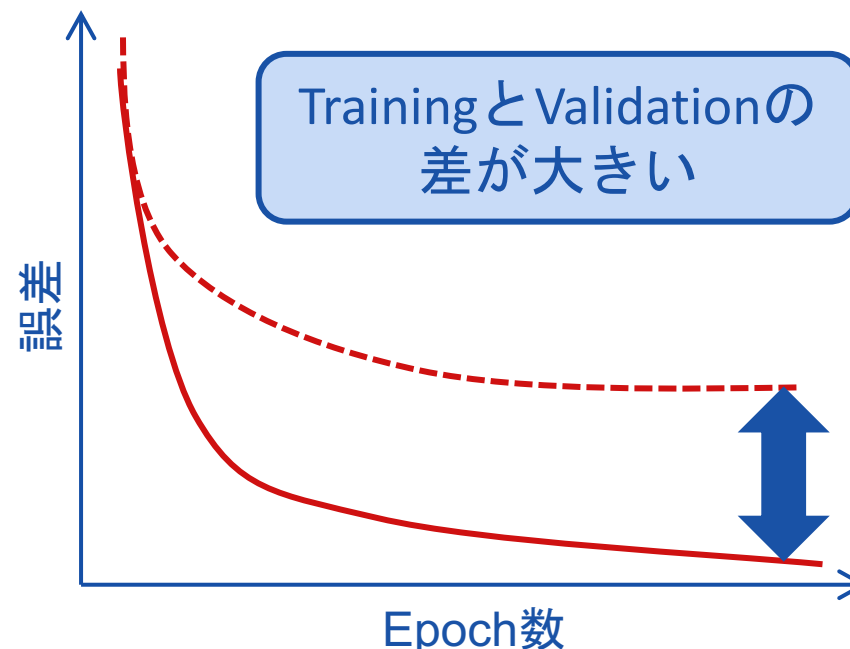
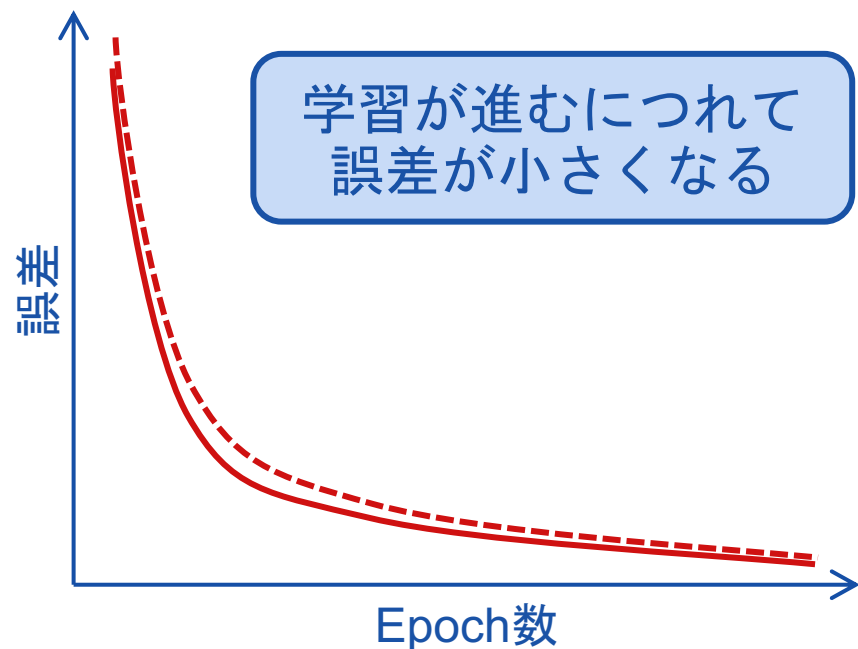
学習曲線の読み方

TrainingとValidationの両データのErrorが低いことがよいモデルであると言える。

TrainingのErrorは低いものの、ValidationのErrorが高い状況を過学習と呼び、モデルがTrainingに特化しすぎて、汎化性能（未知のデータへの精度が高いこと）を失っている状態である。

よいモデルの学習結果

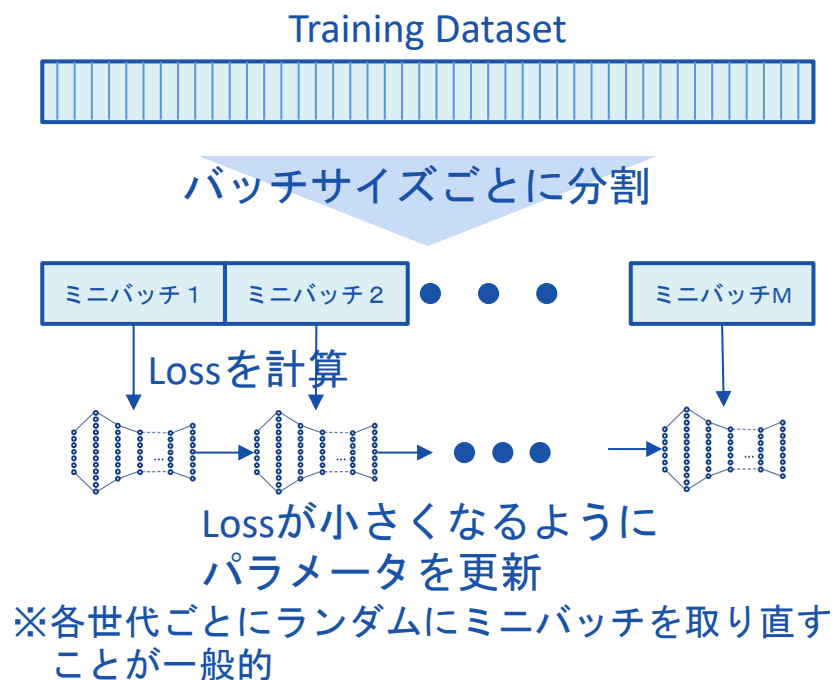
悪いモデルの学習結果（過学習）



学習パラメータの設定

Global Configページにおいて、学習世代数とバッチサイズを設定できる。
 学習曲線が収束していないと判断した場合には学習世代数を増やす。
 バッチサイズは小さすぎるとパラメータ更新が頻発し、学習が非効率になる。
 バッチサイズは一般に分類問題なら分類クラス数以上にする。

1 世代の学習の仕組み



学習パラメータの設定

Config

Global Config Global

Max Epoch = 100
Batch Size = 64

Optimizer Optimizer

Network = Main
Dataset = Training
Updater = Adam

train_error Monitor

Network = MainValidation
Dataset = Training

valid_error Monitor

Network = MainValidation
Dataset = Validation

Structure Search

Enable

Method: Random

Project Description: dataset-require=MNIST

Project Tag:

Max Epoch: 100 ☒ Save Best

Batch Size: 64

Precision: Float

Monitor Interval: 10 epoch

学習世代数
全学習データを使った
時点で1世代と数
える

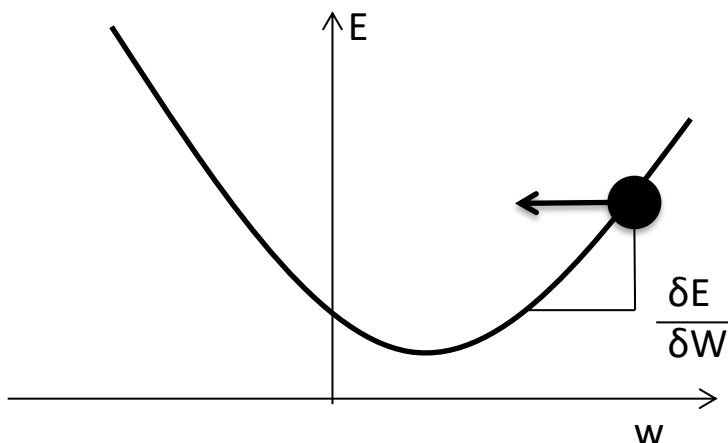
バッチサイズ
1回のパラメータ更
新に使うデータ数

パラメータの更新方法（学習方法）

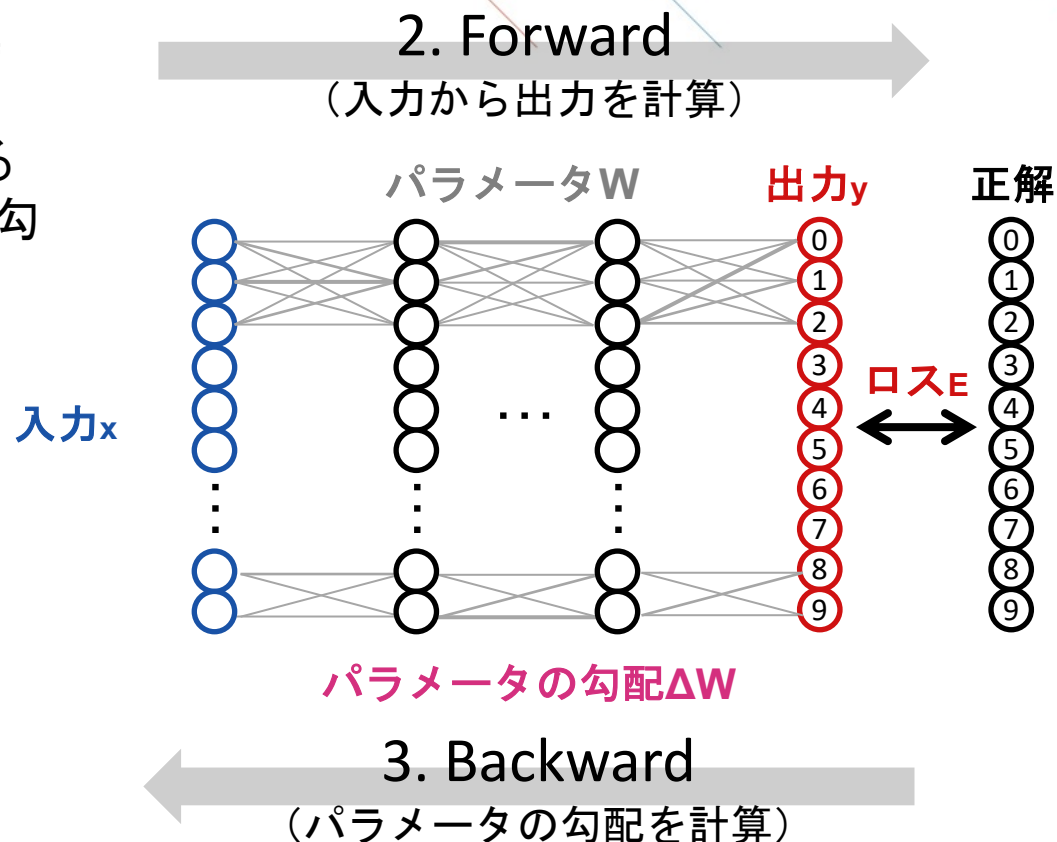
ニューラルネットワークでは、乱数で初期化したパラメータWをミニバッチ勾配降下法（Mini-Batch Gradient Descent）で最適化するのが一般的である。

Forward→Backward→Updateを繰り返し、パラメータWを最適化していく。

1. 学習データからミニバッチ（256個程度のデータ）を取得
2. 用意したデータを用いてForward計算を行い、現在のパラメータWによる出力yとロス（最小化したい値）Eを求める
3. Backward計算を（ロスEの逆伝播）行い、パラメータWの勾配 ΔW を求める
4. Updateを行う（求めた勾配 ΔW を元にWを更新）



$$W_{t+1} \leftarrow W_t - \eta \Delta W_t$$





評価の実行

評価の実行

TRAININGページのRunボタンをクリックするとEVALUTIONページに遷移し結果を表示する。
各データに対するモデルの予測結果や統計的な精度や指数、混同行列などを確認できる。

評価実行の方法	表示可能なグラフの概要		
	評価グラフ	内容	問題
	Output Result	各データの1つ1つの判定結果	分類/回帰
	Confusion Matrix	データセット全体の統計的な指標と混同行列	分類
	Classification Result	各データの判定確率上位3カテゴリーの確率	分類
	Classification Matrix	カテゴリーごとのモデルの判定傾向	分類
	Likelihood Graph	判定確率と正答率の傾向	分類

評価の見方：Output Result

検証用データ (DatasetタブのValidationデータ)の右側にDeep Learningの予測結果が追記される。
1つ1つの検証用データに対し、Deep Learningがどのように判断したかを確認できる。

Output Resultページのスナップショットとその見方

クリックして選択

Output Result

Confusion Matrix

Classification Result

Classification Matrix

Likelihood Graph

1 / 50

ページを変更し全ての結果を確認可能

Index	x:image	y:9	y'_0	y'_1
1		0	0.9930715	0.0069284593
2		0	0.99998736	1.2614053e-05
3		1	0.00012720657	0.9998728
4		1	0.0025280912	0.99747187

検証用データから転記

Deep Learningの予測結果

ラベル"0"の予想確率

ラベル"1"の予想確率

追記カラム名について

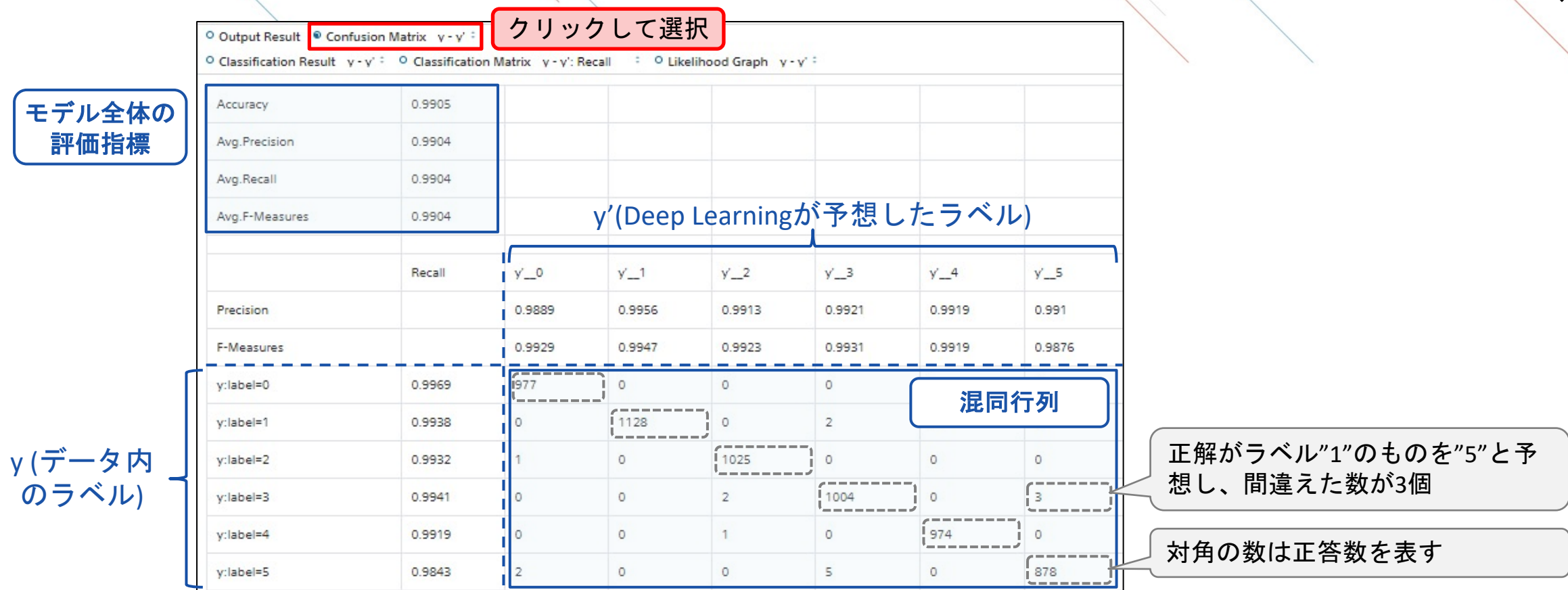
- 出力をアポストロフィー付きで表記
例: 学習時に $x \rightarrow y$ であれば、 y' を出力
- 分類問題の場合にはさらにindexを用い、各クラスの予想確率を出力
例: 2値分類の場合、 $y'_{_0}$ と $y'_{_1}$ を出力

評価の見方：Confusion Matrix

検証用データに対する統計的な評価指標と混同行列を表示する。

混同行列を用いて、全体の正答数や間違いやすいラベルの傾向などを確認できる。

Confusion Matrixページのスナップショットとその見方



評価の見方：Classification Result

各データごとにDeep Learningが確率が高いと予測した上位3つのラベルと確率を表示する。
第1候補の確率を昇順でソートすれば、Deep Learningが判定困難なデータを確認できる。

Classification Resultページのスナップショットとその見方

クリックして選択

ソート結果のリセット

Output Result

Confusion Matrix

Classification Result

Classification Matrix

Likelihood Graph

Reset Condition

Index	x:image	y:label ↑	y__1st ↑	y__1st_value ↑	y__2nd ↑	y__2nd_value ↑	y__3rd ↑	y__3rd_value ↑
1		9	7	0.24524365	3	0.20676546	9	0.1785611
2		7	7	0.24524365	3	0.20676546	9	0.1785611
3		8	7	0.24524365	3	0.20676546	9	0.1785611
4		9	8	0.3802535	9	0.34199467	4	0.19017594
5		3	3	0.43550995	5	0.28446183	8	0.18889885

検証用データから転記

Deep Learningの予測結果

第1候補のラベルと確率

第2候補のラベルと確率

第3候補のラベルと確率

列の↑をクリックでソートが可能

評価の見方：Classification Matrix (Recall)

ラベルごとにDeep Learningが間違えた上位3つの誤判定のラベルと数を表示する。
Recallでは、データセット内の正解ラベルを軸に誤判定結果の確認できる。

Classification Result (Recall) ページのスナップショットとその見方

Recallを選択

クリックして選択

Classification Matrix y - y': Recall

	Recall	true-positive	false-positive 1st	false-positive 2nd	false-positive 3rd	other
y:label=0	0.9969	y'_0: 977	y'_6: 1	y'_7: 1	y'_8: 1	0
y:label=1	0.9938	y'_1: 977	y'_0: 1	y'_2: 1	y'_3: 1	0
y:label=2	0.9932	y'_2: 977	y'_1: 1	y'_0: 1	y'_3: 1	0
y:label=3	0.9941	y'_3: 1004	y'_5: 3	y'_2: 2	y'_8: 1	0
y:label=4	0.9919	y'_4: 974	y'_9: 4	y'_6: 3	y'_2: 1	0
y:label=5	0.9843	y'_5: 878	y'_3: 5	y'_6: 5	y'_0: 2	2
y:label=6	0.9906	y'_6: 949	y'_0: 2	y'_1: 2	y'_4: 2	3
y:label=7	0.9932	y'_7: 1021	y'_2: 4	y'_0: 1	y'_1: 1	1
y:label=8	0.9877	y'_8: 962	y'_0: 3	y'_2: 2	y'_7: 2	5
y:label=9	0.9782	y'_9: 987	y'_7: 7	y'_4: 5	y'_5: 3	7

データ内のラベル

正解数

間違いやすい上位3件のラベルと件数

セルをクリックすると誤判定データの詳細にジャンプ

Index	x:image	y:label ↑	y'_1st ↑	y'_1st_value ↑	y'_2nd ↑	y'_2nd_value ↑	y'_3rd ↑	y'_3rd_value ↑
1		6	4	0.9877814	9	0.006788728	6	0.005254835
2		6	4	0.5159155	6	0.4840762	5	7.991703e-06

評価の見方：Classification Matrix (Precision)

ラベルごとにDeep Learningが間違えた上位3つの誤判定のラベルと数を表示する。
Precisionでは、Deep Learningが判定したラベルを軸に誤判定結果の確認できる。

Classification Result (Precision) ページのスナップショットとその見方

☐ Output Result ☐ Confusion Matrix ☒ Classification Matrix ☐ Likelihood Graph							
Precisionを選択 y - y': Precision							
	Precision	F-Measures	true-positive	false-negative 1st	false-negative 2nd	false-negative 3rd	other
y'_0	0.9889	0.9929	y:label=0: 977	y:label=8: 3	y:label=5: 2	y:label=6: 2	4
y'_1	0.9947	0.9947	y:label=1: 1025	y:label=9: 5	y:label=3: 2	y:label=6: 1	0
y'_2	0.9923	0.9923	y:label=2: 1025	y:label=5: 5	y:label=1: 2	y:label=6: 1	0
y'_3	0.9921	0.9931	y:label=3: 1004	y:label=5: 5	y:label=1: 2	y:label=6: 1	0
y'_4	0.9919	0.9919	y:label=4: 974	y:label=9: 5	y:label=6: 2	y:label=8: 1	0
y'_5	0.991	0.9876	y:label=5: 878	y:label=3: 3	y:label=9: 3	y:label=6: 1	1

Deep Learningが判定したラベル

正解数

間違いやすい上位3件のラベルと件数

評価の見方：Likelihood Graph (Softmax)

スコアや確率ごとの判定結果により、各ラベルをどの程度分離できているかを確認できる。
Softmaxを用いた分類問題の場合には、判定ラベルの予測確率とその正答の対応を表示する。

Classification Result (Softmax) ページのスナップショットとその見方





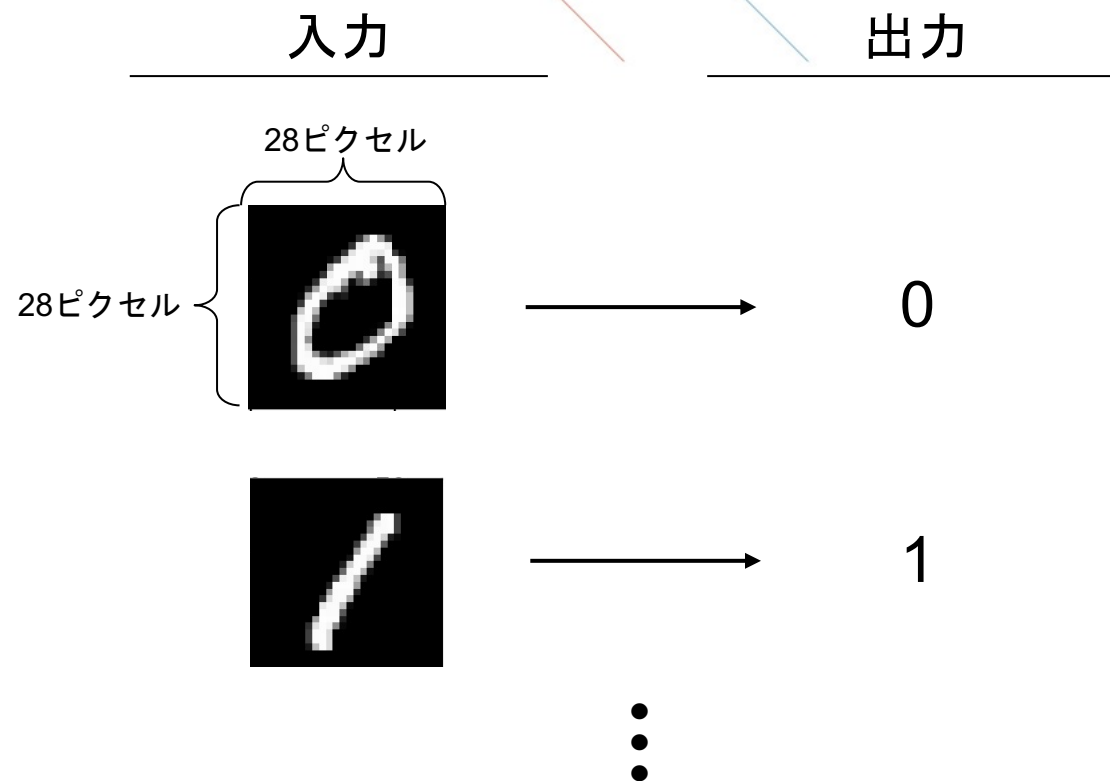
サンプルデータセットについて

手書き数字の分類

0～9の手書き数字の画像データを分類するデータセットです。

機械学習の画像分類で一般的なデータセットで、アメリカ国勢調査局が収集した手書き文字データから数字を抽出したものです。

データセットの例



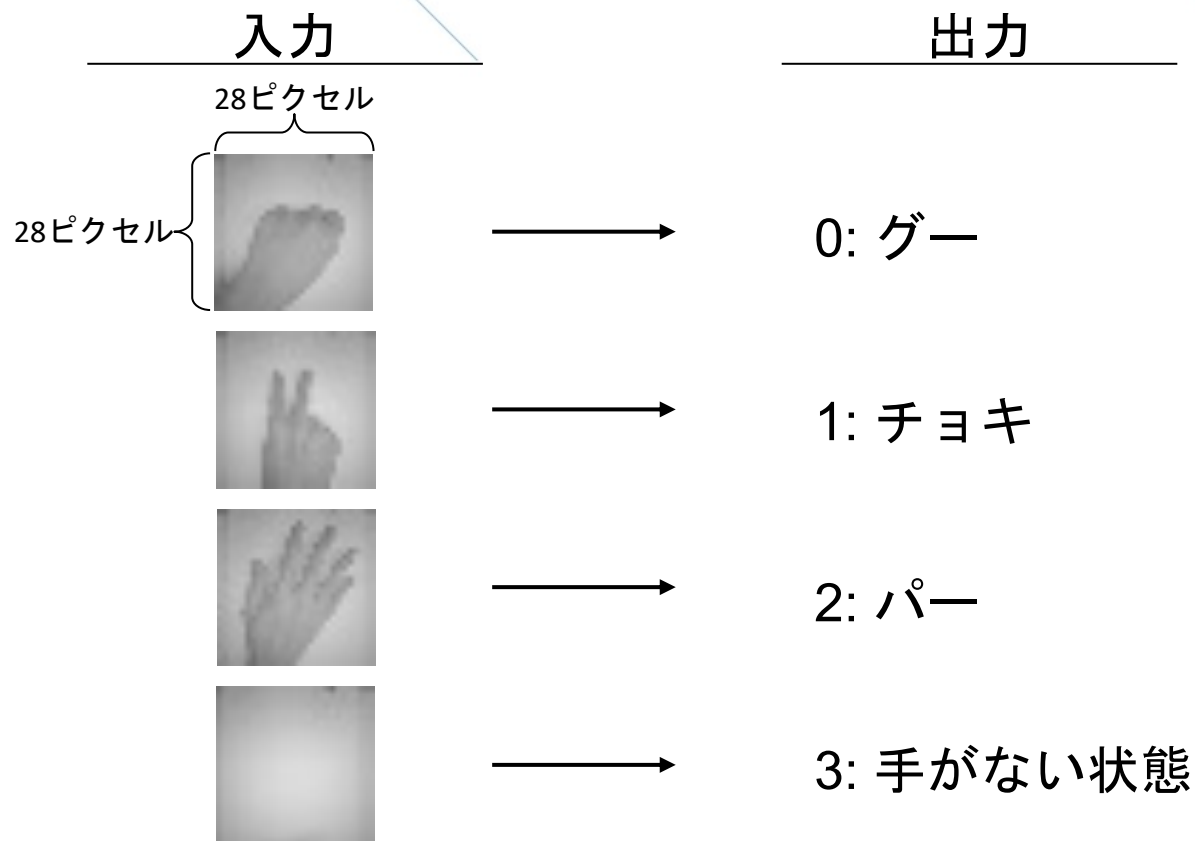
データセットの名称

Neural Network Console(以下NNC)ではMNISTに関して6つのデータセットを準備しています。
mnist.small_mnist_4or9は0～9の数字のうち、4と9だけを抽出したデータセットです。
簡単に画像分類を体験したい場合には、こちらのデータセットをご利用ください。
学習用と検証用はデータの重複なく分割をしたセットになります。

#	データセット名称	説明
1	mnist.mnist_training	MNISTの学習用データセット
2	mnist.mnist_test	MNISTの検証用データセット
3	mnist.small_mnist_4or9_training	MNISTの4と9の2つにデータを減らした際の学習用データセット
4	mnist.small_mnist_4or9_test	MNISTの4と9の2つにデータを減らした際の検証用データセット

グーチョキパーの分類

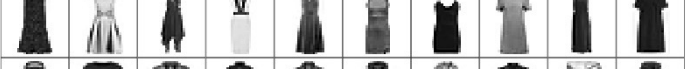
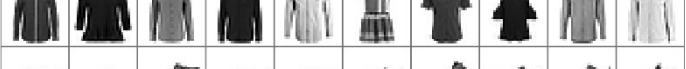
じゃんけんのグー・チョキ・パーの手と、手がない状態の4つを分類するデータセットです。
Neural Network Console(以下NNC)上ではsample.image.hand-signの名称で準備しております。
学習用(training)、検証用(test)には異なるデータがそれぞれ含まれています。。



ファッション画像の分類

ズボンやブーツ等の衣類の白黒画像を10種類に分類するデータセットです。
Neural Network Console(以下NNC)上ではfashion_mnistの名称で準備しております。
学習用(training)、検証用(test)には異なるデータがそれぞれ含まれています。

データセットの例

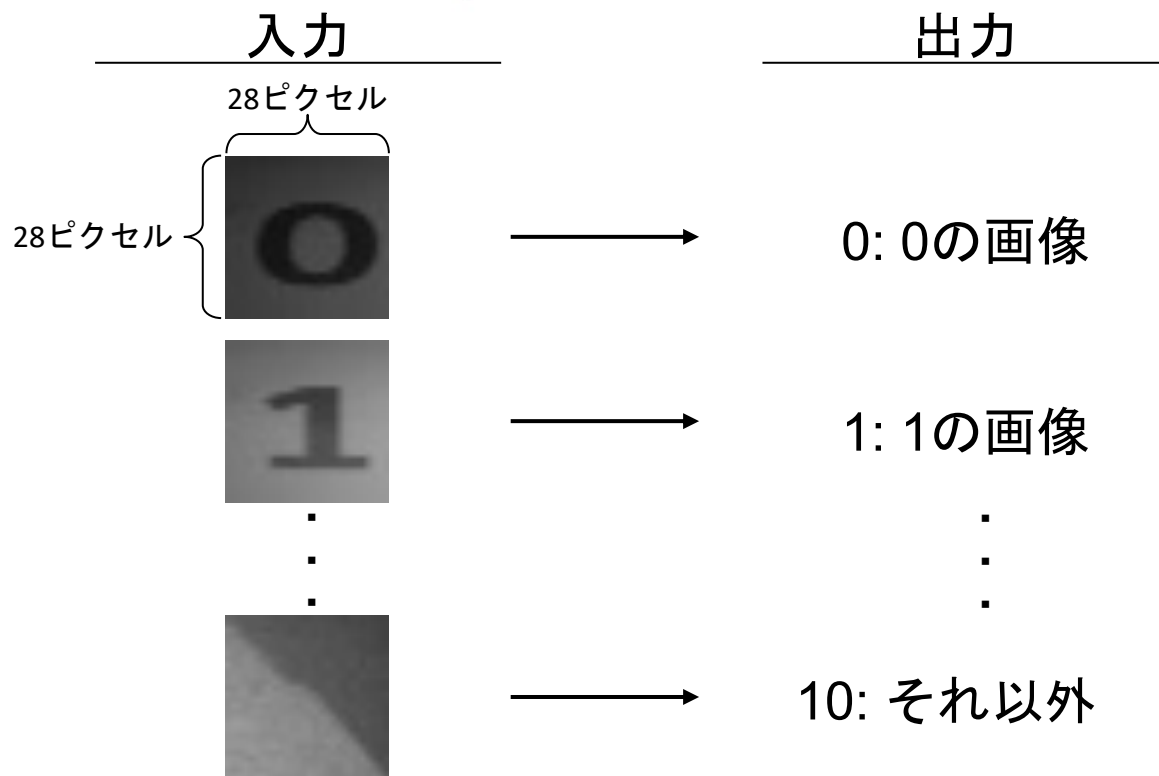
T-shirt/top	
Trouser	
Pullover	
Dress	
Coat	
Sandal	
Shirt	
Sneaker	
Bag	
Ankle boot	



出典: Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

印字数字の分類

印刷された0～9の数字とそれ以外を分類するデータセットです。
メーターの読み取り自動化のユースケースを想定して作られたものになります。
Neural Network Console(以下NNC)上ではimage_classification.digits.trainingの名称で準備しております。
学習用(training)、検証用(test)には異なるデータがそれぞれ含まれています。

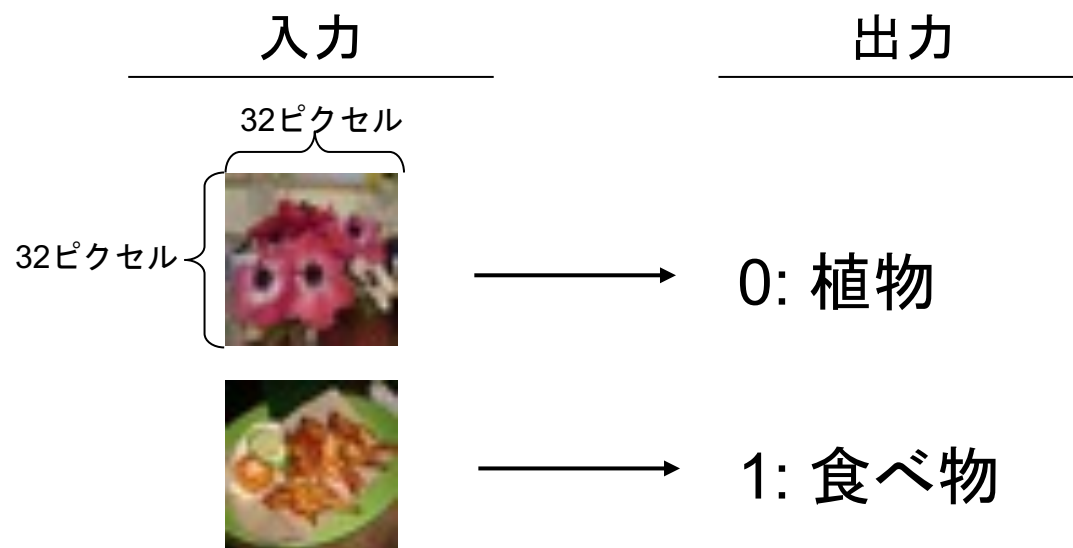


植物と食べ物の分類

植物と食べ物の画像を分類するデータセットです。

Neural Network Console(以下NNC)上ではflower_foodの名称で準備しております。

学習用(training)、検証用(test)には異なるデータがそれぞれ含まれています。。



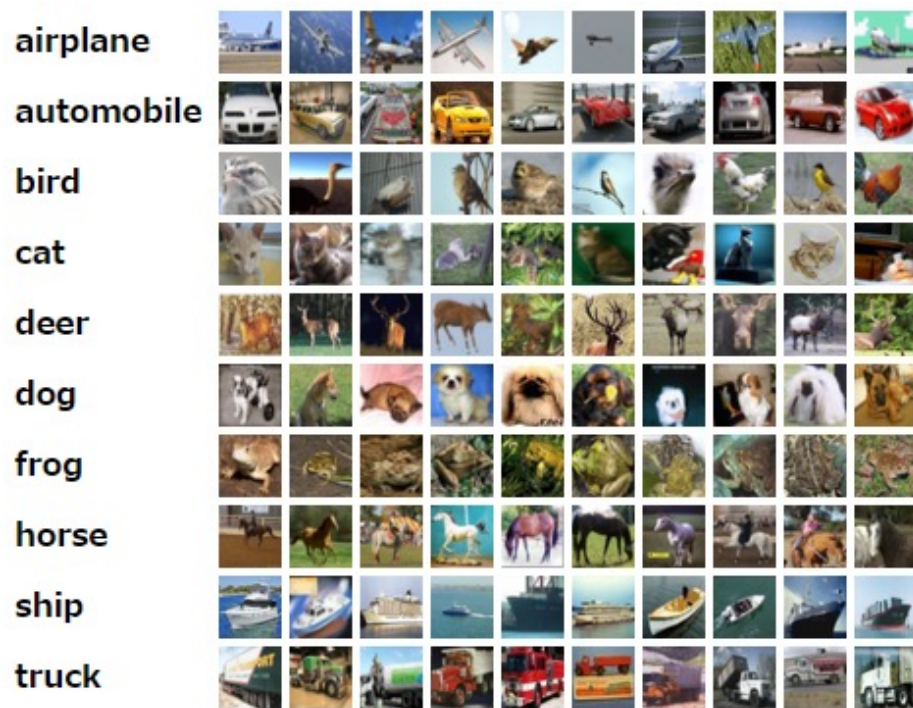
一般的な画像の分類(10分類のデータセット)

約8000万枚の画像データから6万枚を抽出し、飛行機、鳥、犬などの10種類の画像を分類したデータセットで、機械学習のベンチマークによく用いられるもの(CIFAR-10)です。

Neural Network Console(以下NNC)上ではCIFAR10.cifar10として準備しております。

学習用(training)、検証用(test)はデータの重複がないように分割したデータセットになります。。

データセットの例



出典: Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

一般的な画像の分類(100分類のデータセット)

前頁と同様の形式で、画像の分類数を100種類にしたデータセット(CIFAR-100)です。

NNC上ではCIFAR100.cifar100として準備しております。

学習用(training)、検証用(test)はデータの重複がないように分割したデータセットになります。

CIFAR-100のラベル一覧

#	ラベル
0	apple
1	aquarium fish
2	baby
3	bear
4	beaver
5	bed
6	bee
7	beetle
8	bicycle
9	bottle
10	bowl
11	boy
12	bridge
13	bus
14	butterfly
15	camel
16	can
17	castle
18	caterpillar
19	cattle

#	ラベル
20	chair
21	chimpanzee
22	clock
23	cloud
24	cockroach
25	couch
26	crab
27	crocodile
28	cup
29	dinosaur
30	dolphin
31	elephant
32	flatfish
33	forest
34	fox
35	girl
36	hamster
37	house
38	kangaroo
39	keyboard

#	ラベル
40	lamp
41	lawn mower
42	leopard
43	lion
44	lizard
45	lobster
46	man
47	maple tree
48	motorcycle
49	mountain
50	mouse
51	mushroom
52	oak tree
53	orange
54	orchid
55	otter
56	palm tree
57	pear
58	pickup truck
59	pine tree

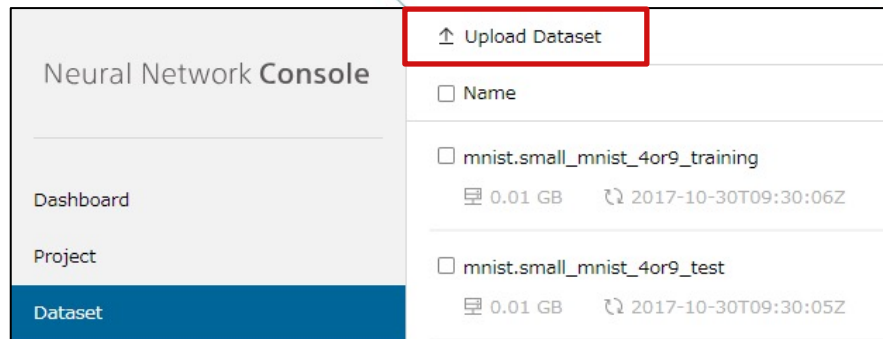
#	ラベル
60	plain
61	plate
62	poppy
63	porcupine
64	possum
65	rabbit
66	raccoon
67	ray
68	road
69	rocket
70	rose
71	sea
72	seal
73	shark
74	shrew
75	skunk
76	skyscraper
77	snail
78	snake
79	spider

#	ラベル
80	squirrel
81	streetcar
82	sunflower
83	sweet pepper
84	table
85	tank
86	telephone
87	television
88	tiger
89	tractor
90	train
91	trout
92	tulip
93	turtle
94	wardrobe
95	whale
96	willow tree
97	wolf
98	woman
99	worm

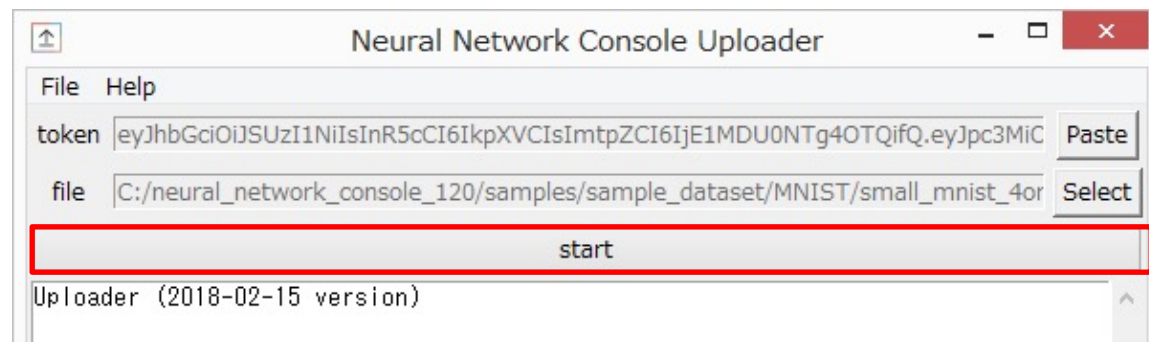
データセットのアップロード

専用のアップローダからデータセットをアップロードする。
Neural Network Console所定のCSVファイルを指定することで、自動的に対応する画像などのファイルもアップロードされる。

1. “Dataset” からUpload Datasetを選択



3. アップローダを用いてデータセット CSVファイルとデータをアップロード



2. アップローダをダウンロード (Windows / MacOS)



Upload Datasetで表示されるトークンをアップローダにPasteし、アップロードするデータセットCSVを指定してStartボタンを押すことでアップロードを開始

アップロード用のCSVファイルの準備

アップロード後にDatasetタブに表示される形式に沿ってCSVファイルを準備する必要がある。
 画像や時系列データなどのファイルはローカルでの保存場所を指定する。

準備するCSVファイルの例

mnist.mnist_training.csv

x:image	y:label
data/train/5/0001.jpg	5
data/train/0/0001.jpg	0
data/train/4/0001.jpg	4
data/train/1/0001.jpg	1
⋮	⋮

データセット一覧へのアップロード後の例

Neural Network Console

Group

Personal

Upload Dataset

Search

CSVのファイル名がデータ名になる

☐ mnist.mnist_training

☐ mnist.mnist_test

☐ mnist.mnist_training_100

☐ mnist.mnist_unlabeled

☐ iris_flower_dataset.iris_flower_dataset_training_dele

☐ iris_flowe

画像や時系列のファイルパスはデータが展開される

Preview

60000 Rows 2 Cols

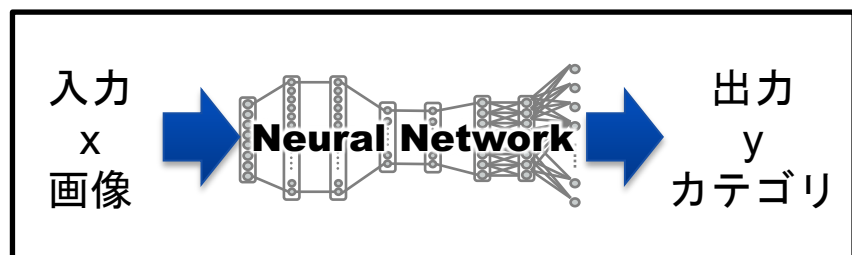
	x:image	y:label
1		5
2		0
3		4
4		1

データセットの準備（画像データ）

Neural Network Console所定のCSVファイルフォーマットでデータセットを準備

1行目	=ヘッダ	変数名[__次元Index][:ラベル名]
2行目以降	=データ	値 or ファイル名

画像認識機学習用データセットの例



入力xには画像ファイル名を指定

※Neural Network Consoleは2バイト文字に対応していないため、CSVファイル内やファイル名に漢字等を含めないようにする

ヘッダ

データ
(2行目以降)

xとy、2つの変数

	A	B
1	x:image	y
2	./training/5/0.png	5
3	./training/0/1.png	0
4	./training/4/2.png	4
5	./training/1/3.png	1
6	./training/9/4.png	9
7	./training/2/5.png	2
8	./training/1/6.png	1
9	./training/3/7.png	3
10	./training/1/8.png	1
11	./training/4/9.png	4

出力yには正解のカテゴリのIndexを記述

Neural Network Consoleに対応したデータセットファイルは簡単なスクリプトで作成可能

ローカルディレクトリとの対応例

Neural Network Console所定のCSVファイルフォーマットは学習用と検証用に分ける。
分類用のラベルなどは事前に数値化しておく必要がある。

画像分類のCSVファイル作成の例



train.csv		test.csv	
x	y	x	y
data/train/dog/0001.jpg	0	data/test/dog/0001.jpg	0
data/train/dog/0002.jpg	0	data/test/dog/0002.jpg	0
data/train/dog/0003.jpg	0	data/test/dog/0003.jpg	0
⋮		⋮	
data/train/cat/0001.jpg	1	data/test/cat/0001.jpg	1
data/train/cat/0002.jpg	1	data/test/cat/0002.jpg	1
data/train/cat/0003.jpg	1	data/test/cat/0003.jpg	1
⋮		⋮	

ラベルは事前に数値化

0: dog、1:cat