

ゼロから学ぶ、ラズパイAI実装 ハンズオンセミナー

～セットアップから画像認識AI実装まで～

2018年1月27日

名古屋校 2011期 越智 由浩



今日の心がまえ、スタンス

- 他人の知見(ブログや記事)を参考に“真似ながら動かしてみる”を実践できるようになることを目指します
- コマンドや各種ツールなど次々に新しいことが出てくると思いますが、手順はgithubに掲載しますので、いきなり全部覚えようとせず、まずは進みましょう
- AIを実装して動かすまでの一通りを体感してもらうことで、どのような要素で成り立っているのか、これからこういったことを深掘りして学べばよいかを考えるきっかけになればと思います

今日の時間配分

10:00 – 12:00	<ul style="list-style-type: none">• ラズパイ基本セットアップ• カメラを使った画像配信
13:00 – 15:00	<ul style="list-style-type: none">• ニューラルネットワーク概説• 手書き文字認識システムの実装とテスト
15:30 – 16:30	<ul style="list-style-type: none">• 物体識別システムの実装とテスト
16:30 – 17:00	<ul style="list-style-type: none">• 学びの振り返り

今日の時間配分

10:00 – 12:00	<ul style="list-style-type: none">• ラズパイ基本セットアップ• カメラを使った画像配信
13:00 – 15:00	<ul style="list-style-type: none">• ニューラルネットワーク概説• 手書き文字認識システムの実装とテスト
15:30 – 16:30	<ul style="list-style-type: none">• 物体識別システムの実装とテスト
16:30 – 17:00	<ul style="list-style-type: none">• 学びの振り返り

基本セットアップ／カメラを使った画像配信

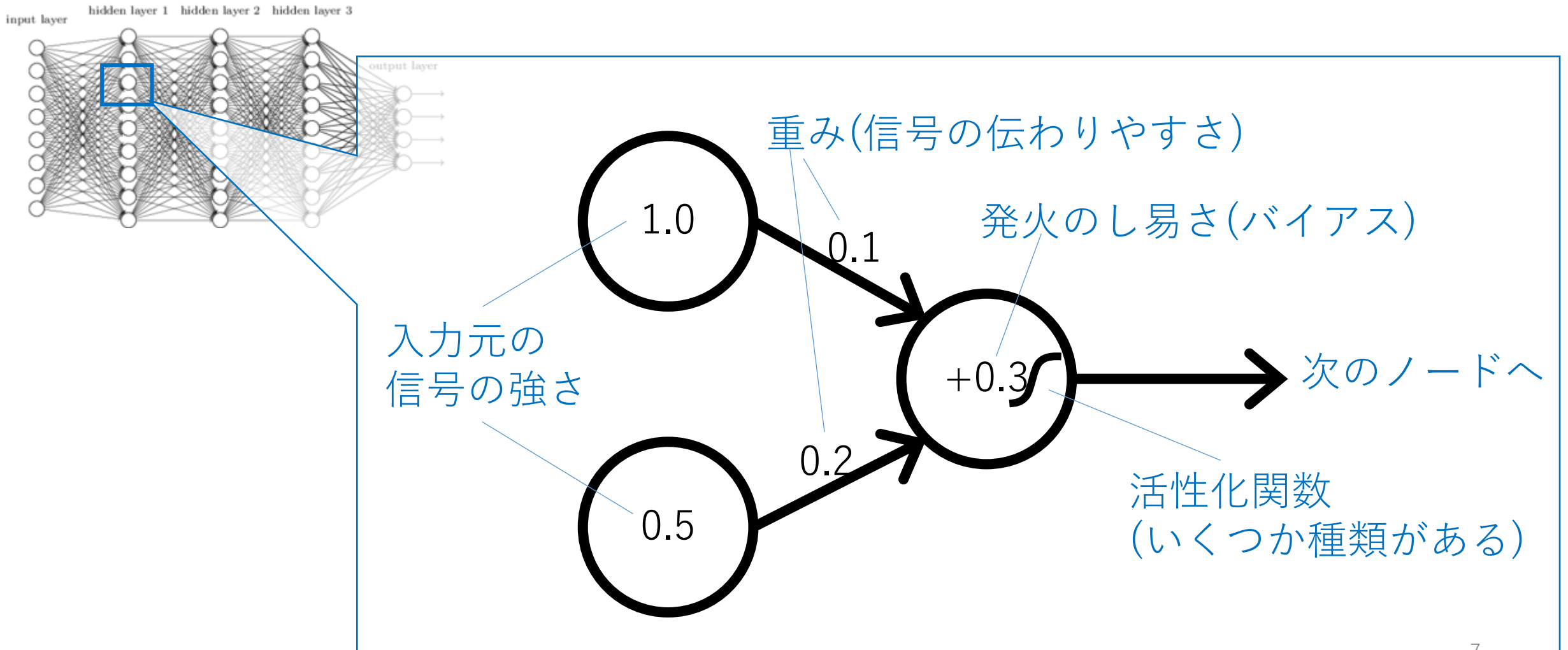


https://github.com/yoshihiroo/programming-workshop/tree/master/rpi_ai_handson

今日の時間配分

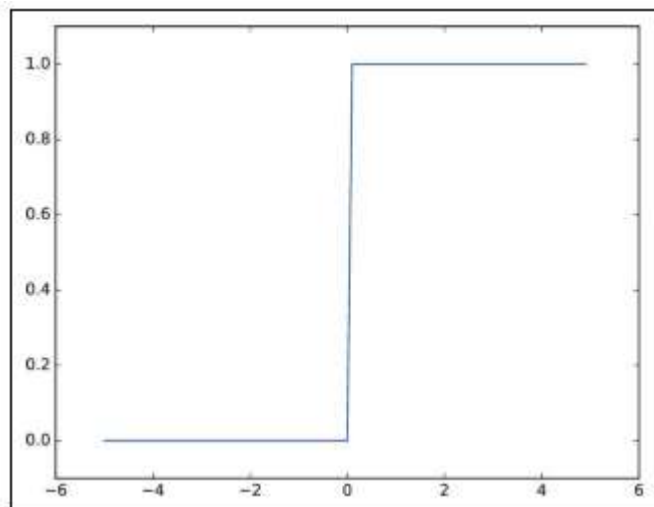
10:00 – 12:00	<ul style="list-style-type: none">• ラズパイ基本セットアップ• カメラを使った画像配信
13:00 – 15:00	<ul style="list-style-type: none">• ニューラルネットワーク概説• 手書き文字認識システムの実装とテスト
15:30 – 16:30	<ul style="list-style-type: none">• 物体識別システムの実装とテスト
16:30 – 17:00	<ul style="list-style-type: none">• 学びの振り返り

ニューラルネットワークモデルの計算 ルール

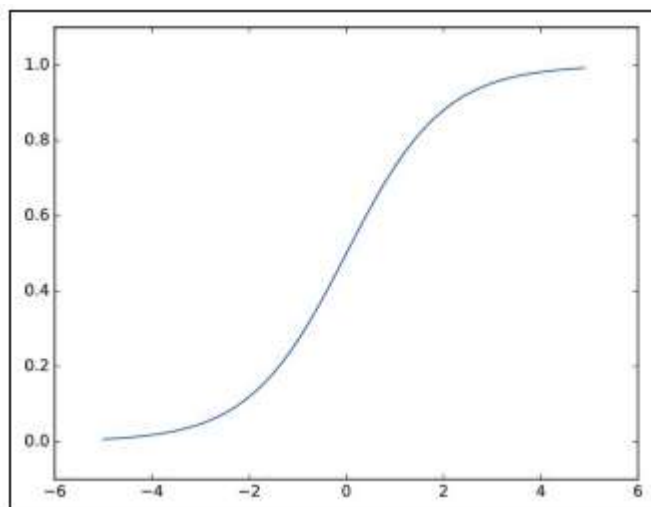


活性化関数

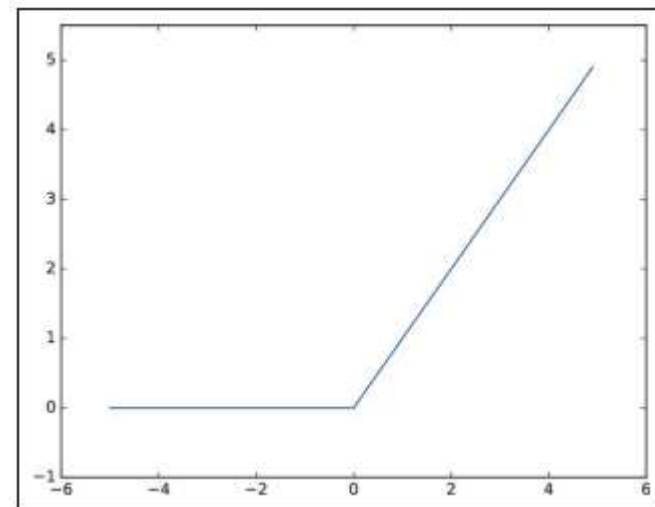
ステップ関数



シグモイド関数



ReLU関数
(Rectified Linear Unit)



出力
↑
入力の総和 + バイアス
→

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

$$h(x) = \frac{1}{1 + \exp(-x)}$$

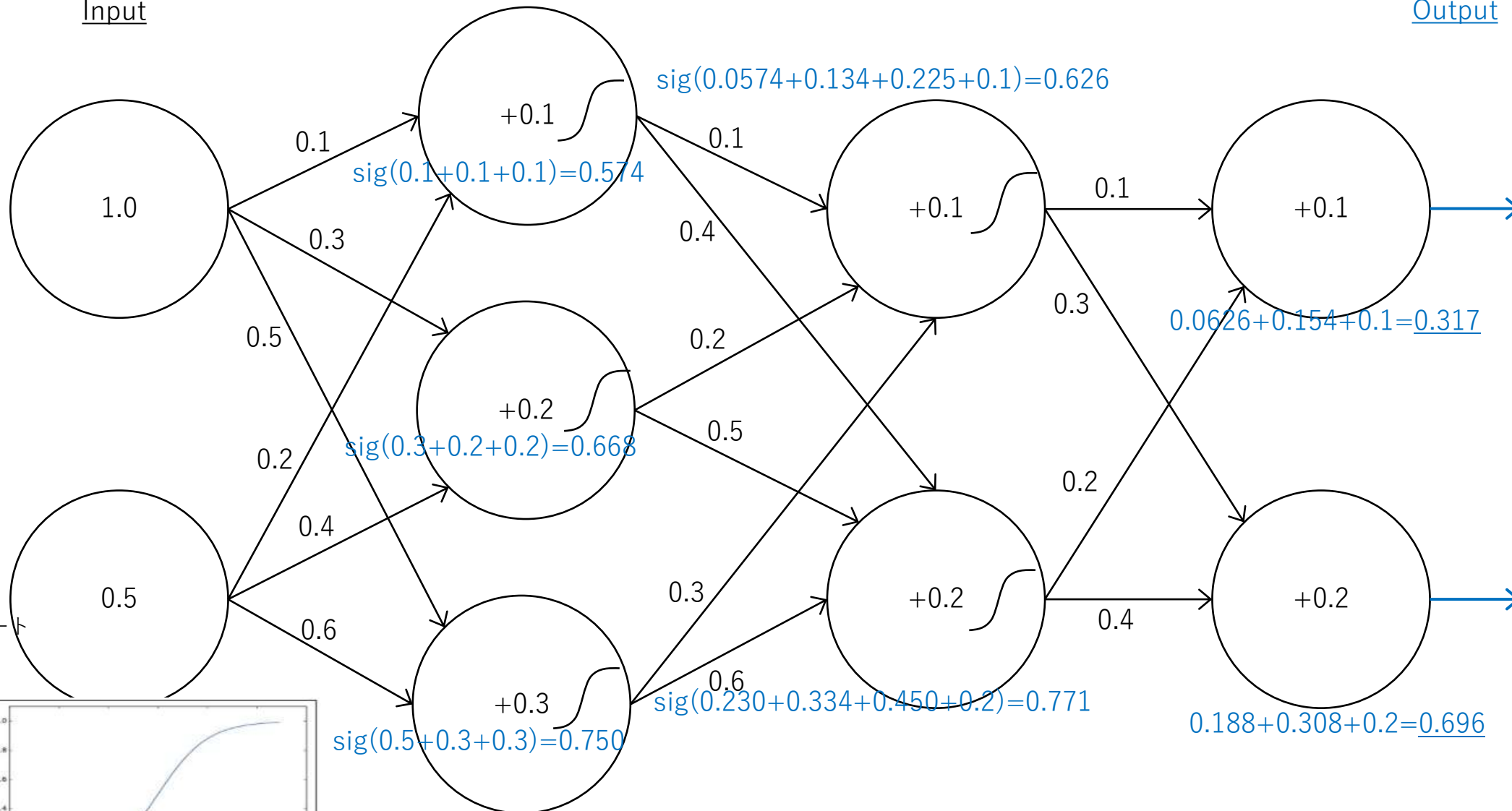
$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

ニューラルネットワーク、計算練習

sig() : シグモイド関数

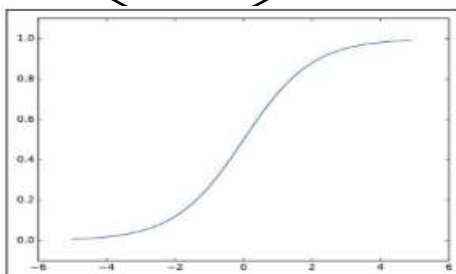
Input

Output



シグモイド関数、参照シート

x	y
0.3	0.574
0.516	0.626
0.7	0.668
1.1	0.750
1.214	0.771

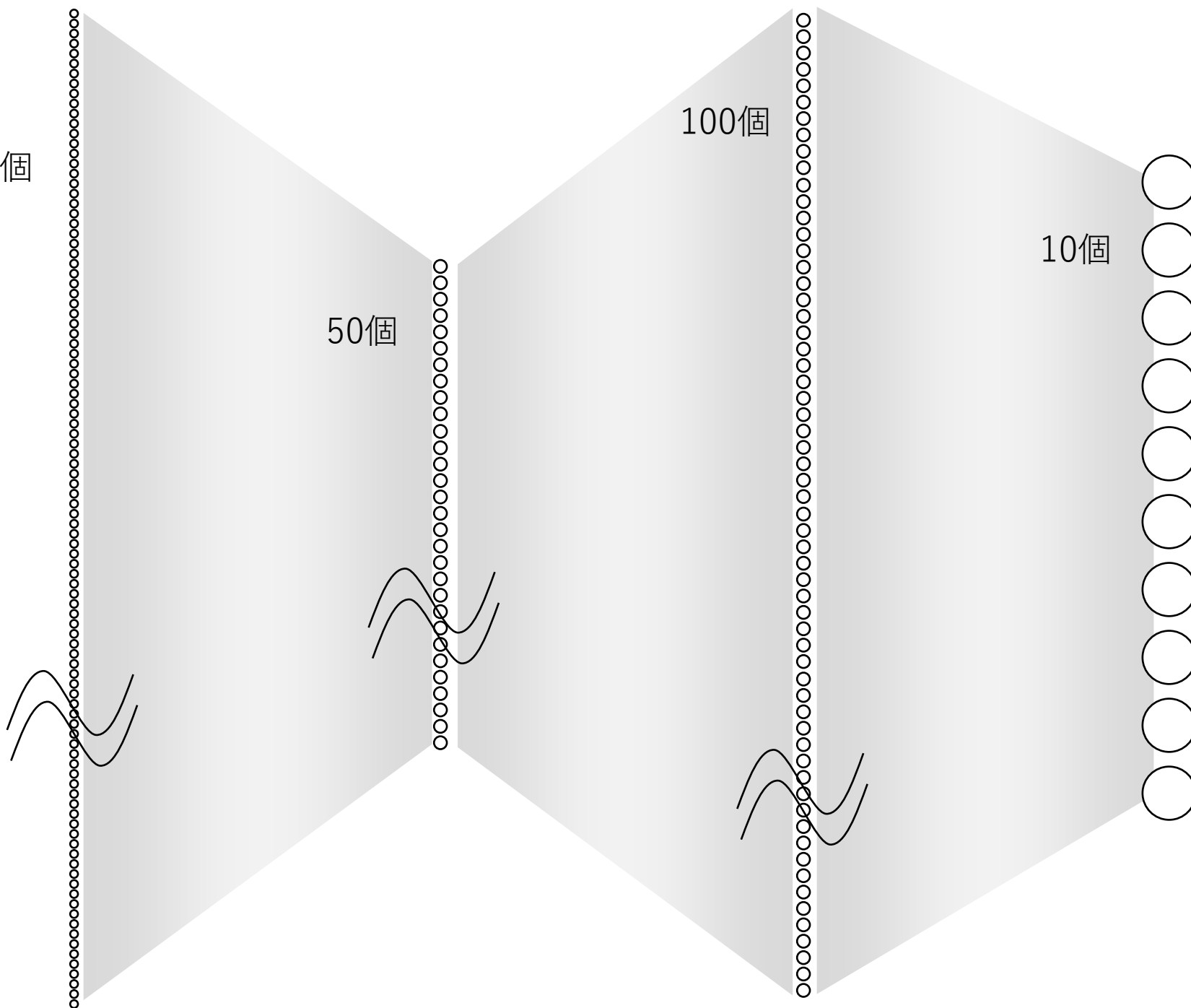


784個

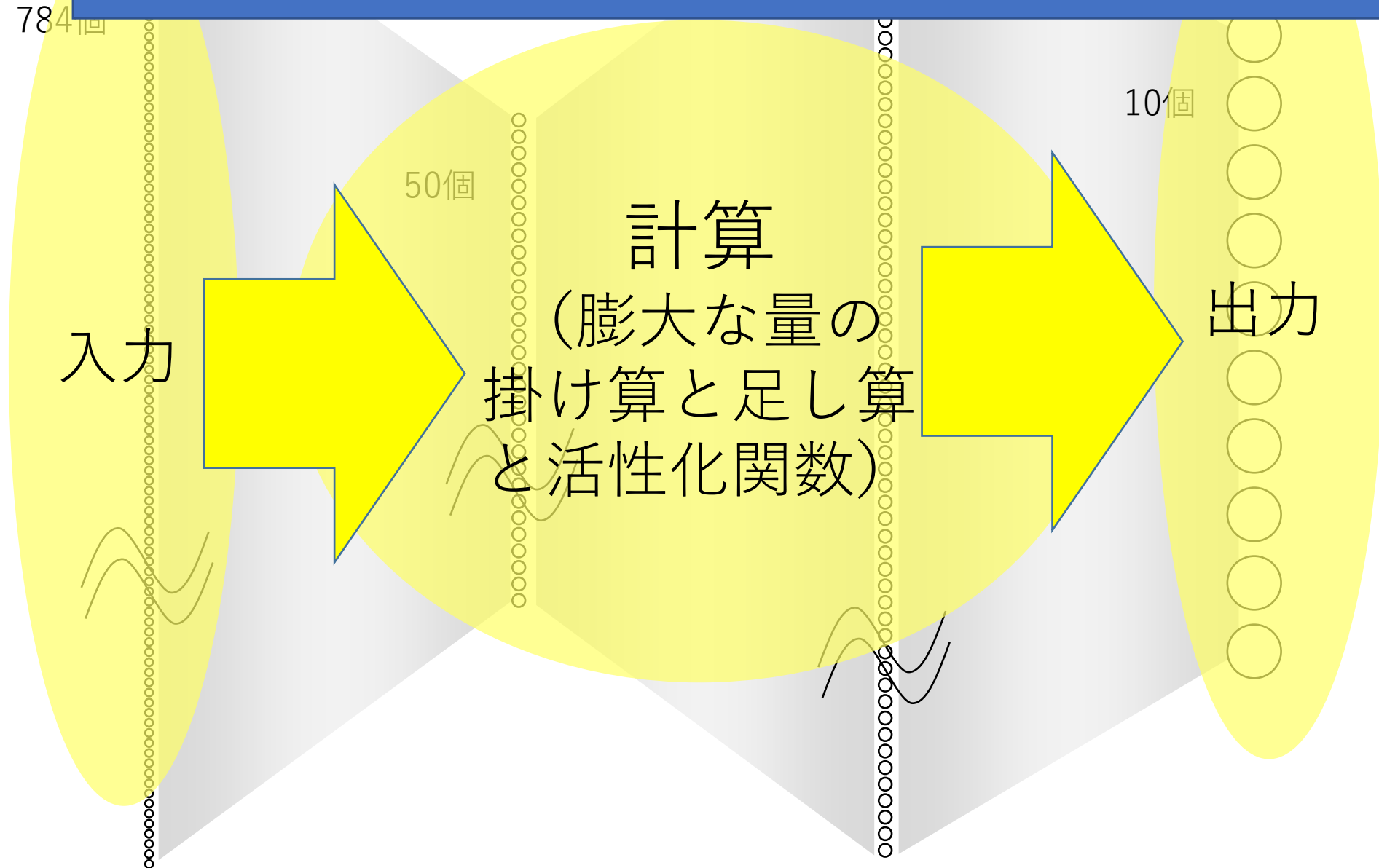
50個

100個

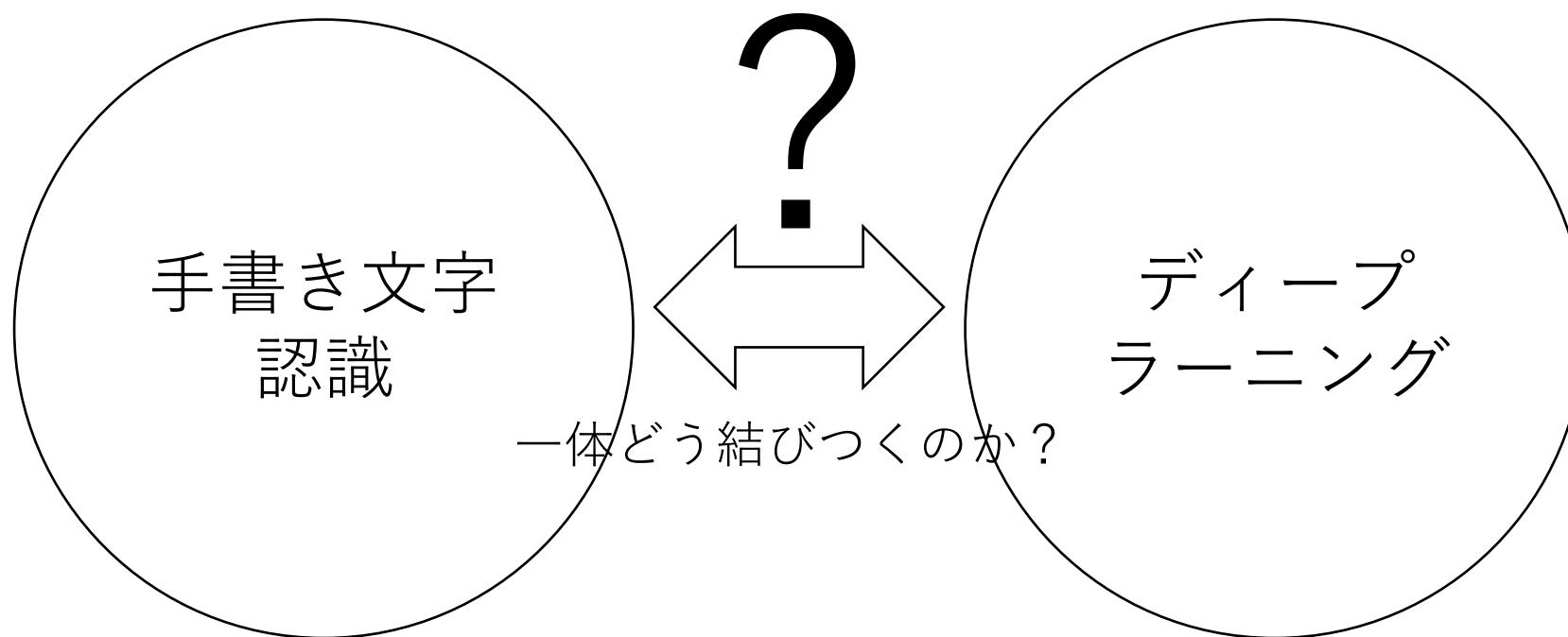
10個



このくらいの数のニューラルネットワークを使うと、
手書きの文字認識ができちゃいます。



手書き文字(数字)認識をさせてみる



人間が文字認識する、をホワイトボードでやってみる

ディープラーニングの全体の流れ

データの準備

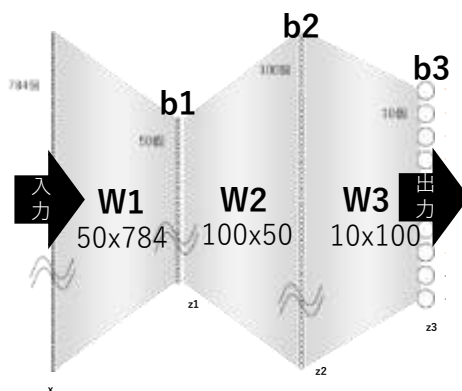


MNIST手書き文字データ

<http://yann.lecun.com/exdb/mnist/>

- 6万文字分の学習用データ
- 1万文字分の検証用データ

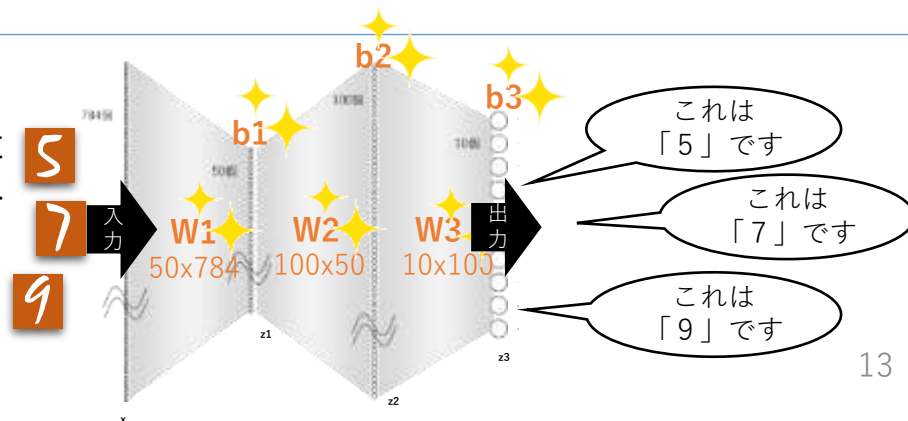
学習



6万文字分の学習用データを使って、入力した手書き文字に対応した出力が得られるようにパラメータ $W1$, $W2$, $W3$, $b1$, $b2$, $b3$ を調整

検証／適用

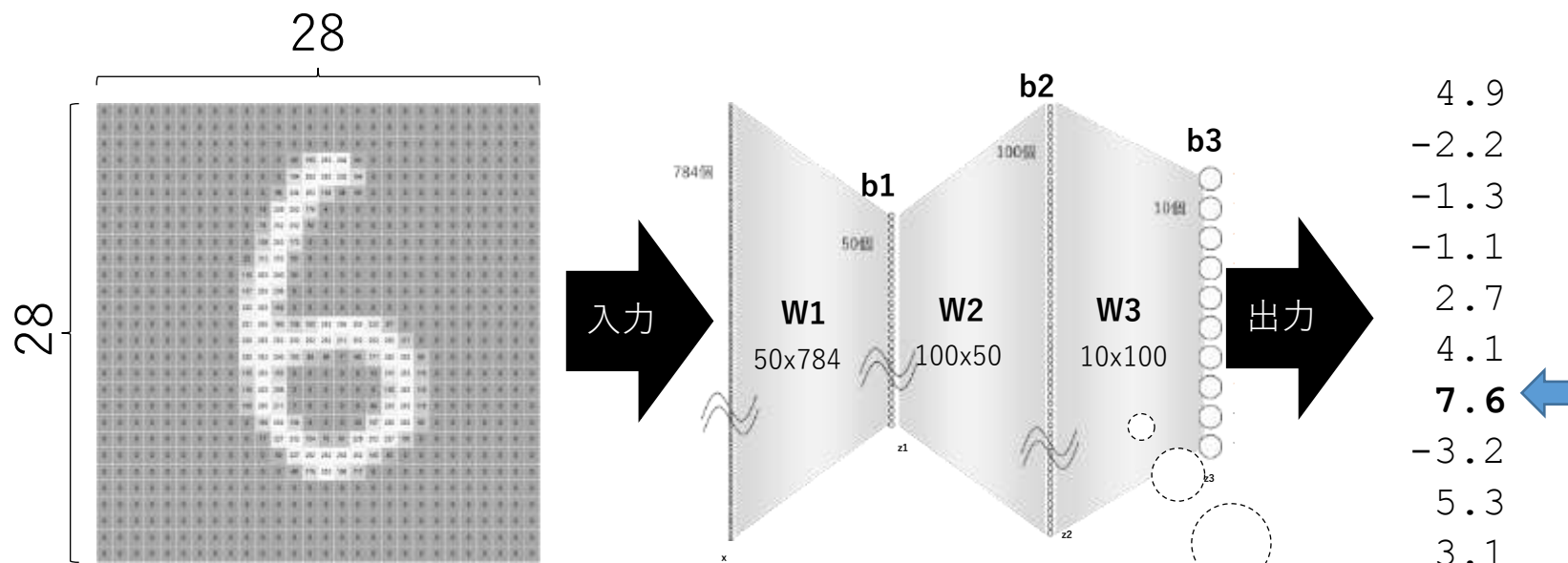
学習済(うまくパラメータが調整された状態)のニューラルネットに検証用データを入力し、うまく認識されることを検証する



ディープラーニングにおける学習とは

手書き文字を数値化し、

ニューラルネットに食わせ..

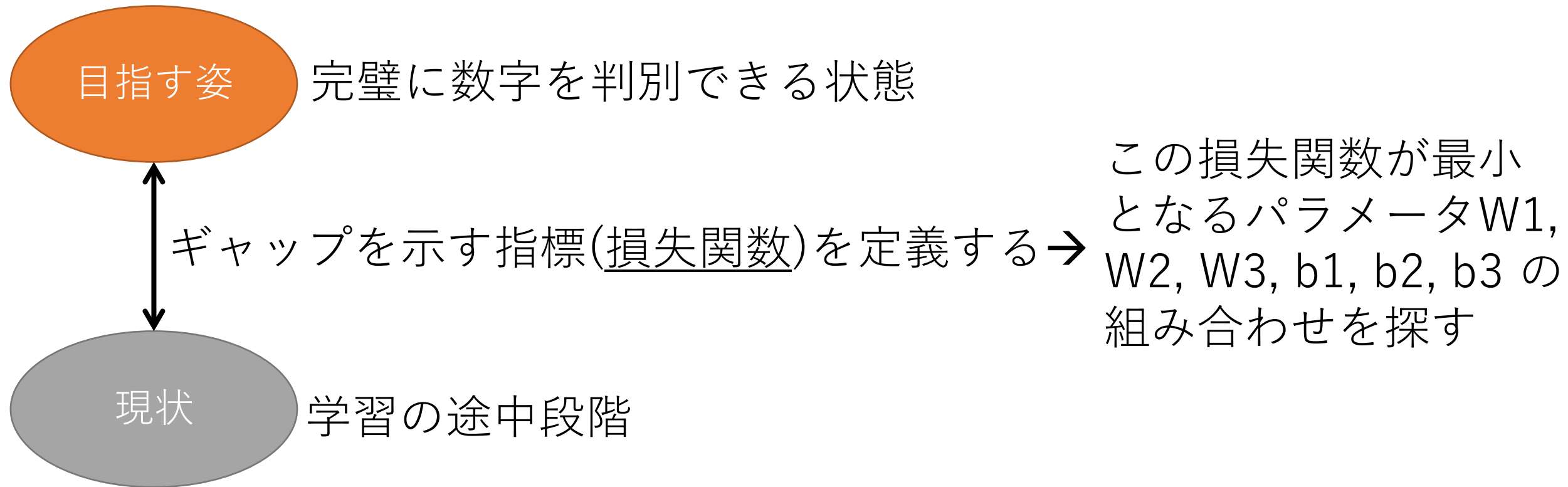


パラメータ(W1, W2, W3, b1, b2, b3 -全部で45,350個の数字)を少しずつ変えながら、入力に対応した箇所が大きな数値を示すような**絶妙な組み合わせ**を探すプロセス

28x28=784個の格子(ピクセル)ごとに0~1の255段階の値で明るさを示すことで手書き文字を表現

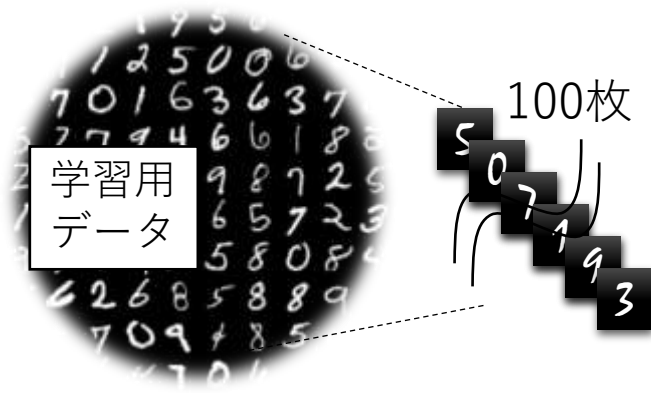


具体的にどうアプローチするか—指標の定義

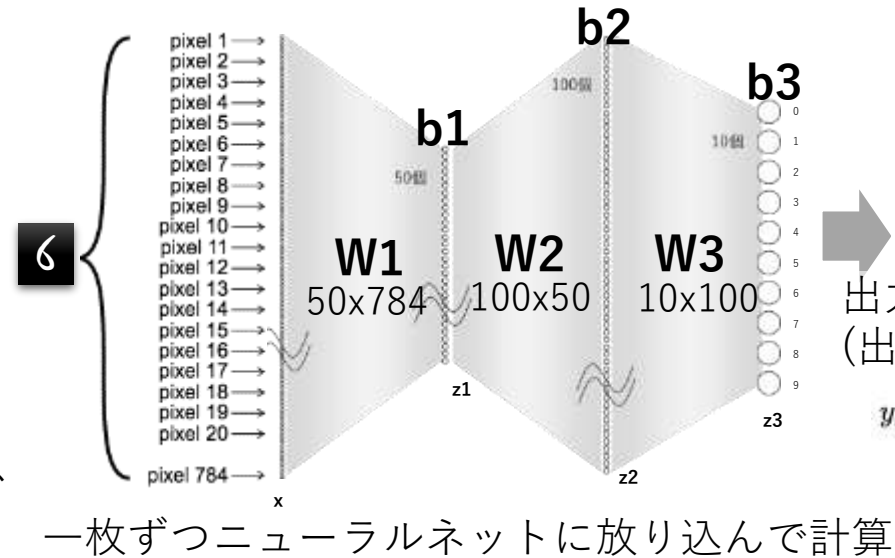




損失関数～当たってなさ具合の指標



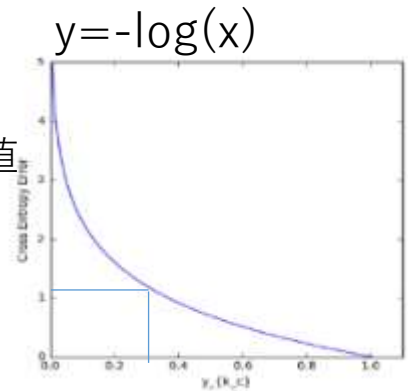
MNIST計6万枚のデータ群から、ランダムに100枚を選び出す。



出力を正規化
(出力総和=1)

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

正解箇所の値の
エントロピー -
-logを計算



100枚分計算して平均を求める。
これが損失関数の値となる。

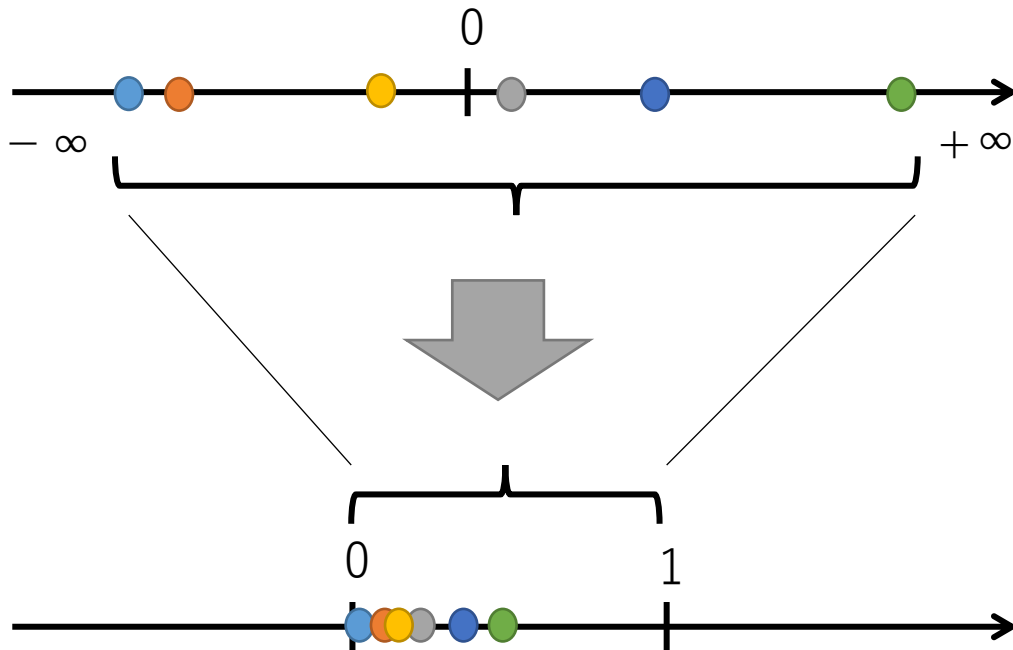
↑
パラメータW1, W2, W3, b1, b2, b3
のある組み合わせ(学習の途中段階)
における当たってなさ具合

(補足) 指数関数を用いた正規化

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

正規化の様子を数直線で表現すると、
大小さまざまな数字について、それぞれの
位置関係は保ったままで、

- 0から1のあいだにギュッと押し込む
- 且つ、値の総和が1になる



Excelシートで実際に試してみた例

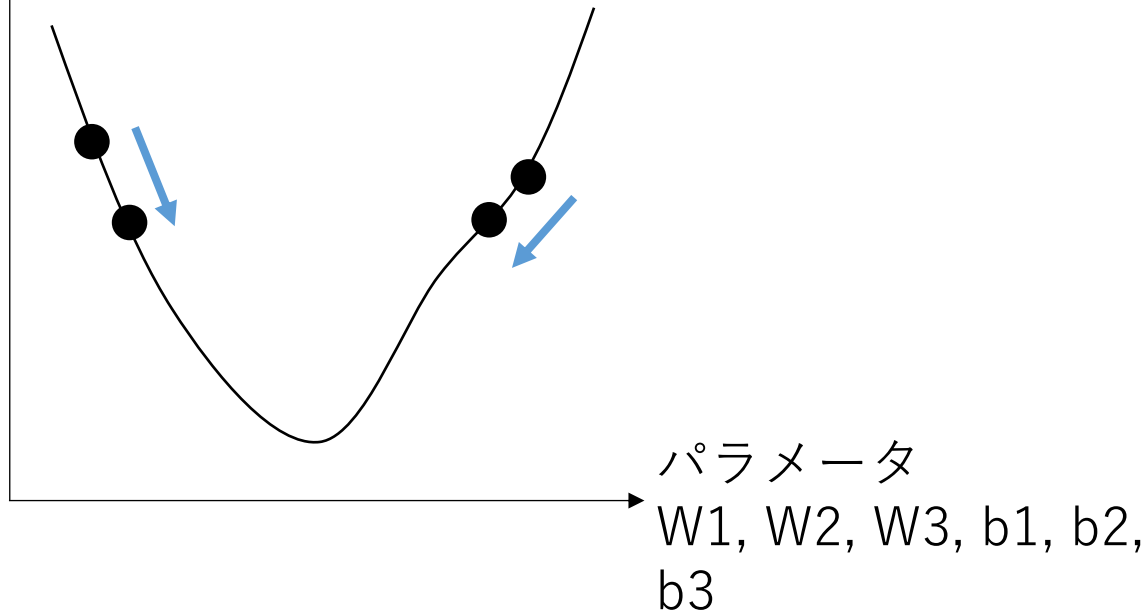
適当な数字	=EXP(左セル)		左セルを%表示
-1.2	0.301194	0.037784	4%
-0.3	0.740818	0.092934	9%
1.3	3.669297	0.460304	46%
0.1	1.105171	0.138641	14%
-1.1	0.332871	0.041758	4%
0.6	1.822119	0.22858	23%
	=SUM(列の合計)		列のSUM 確かに足したら1
	7.97147		100%

= (左セル) ÷ SUMの値

損失関数が最小となるパラメータを探す

損失関数 E

当たってなさ
具合の指標

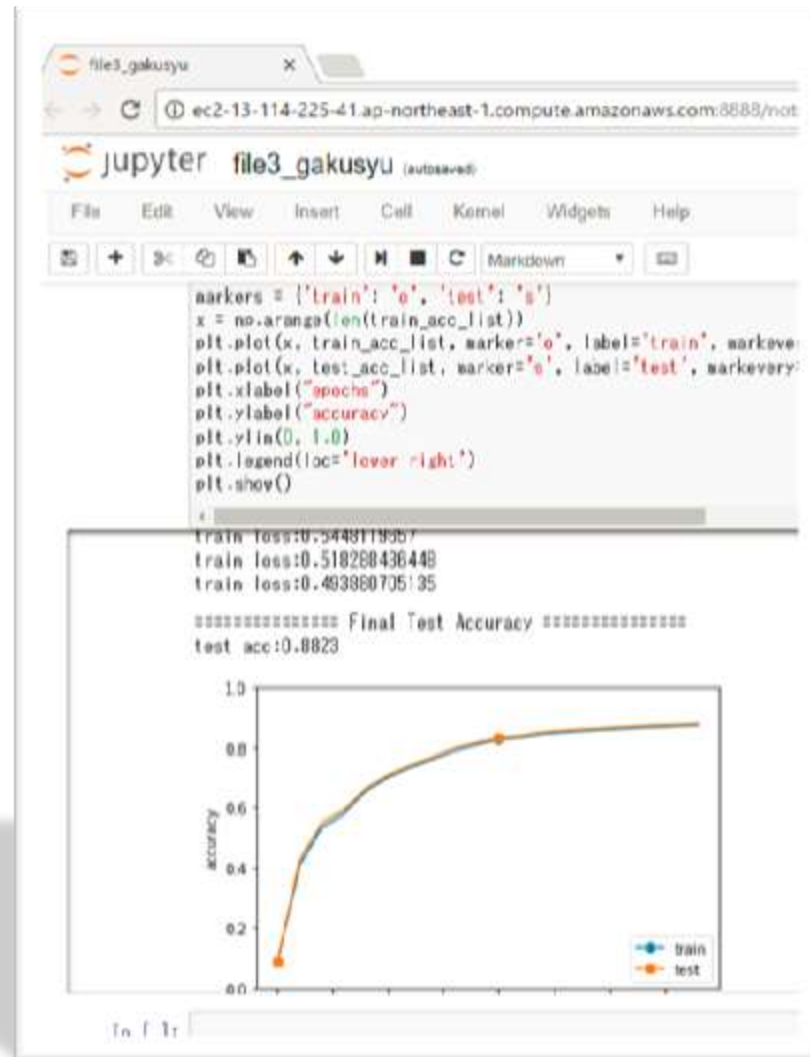


$$W_{\text{代入}} \leftarrow W - \alpha \frac{\partial E}{\partial W}$$

$$b_{\text{代入}} \leftarrow b - \alpha \frac{\partial E}{\partial b}$$

足元の坂の傾きを調べて、
その傾きの大きさに従って一歩進む。
それを繰り返して、 E の一番低いところに
たどり着く

実際の学習の過程を見してみる



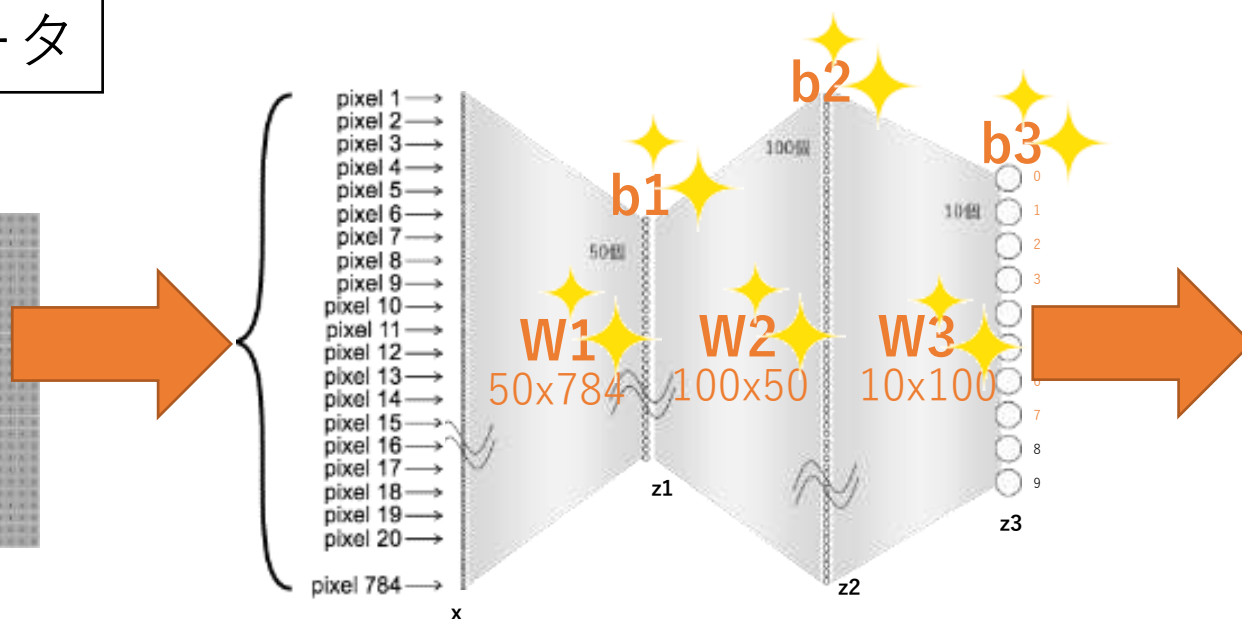
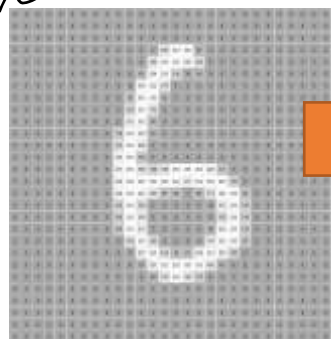
ディープラーニングの学習 ～ 奥深い世界

- 学習をいかに効率よく行うかが、実際にディープラーニングを使う上で大きな課題
- 学習(コンピューターの数値計算)の手法はそれ自体が奥深い研究テーマであり、誤差逆伝搬法(バックプロパゲーション)、SGD、Momentum、AdaGrad、Adam、…など、専門用語がバンバン出てくる領域。今日はそのあたりの深入りはやめときます

学習済のパラメータを使って、文字認識が正しく行われていることを確かめる

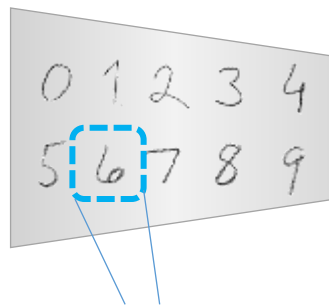
検証用のデータ

例えば「6」



「6」と認識できるか？
実際にPythonで動かして
試してみよう！

digit_recognition_NN.pyの概要

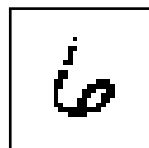


撮影



224x224
カラー

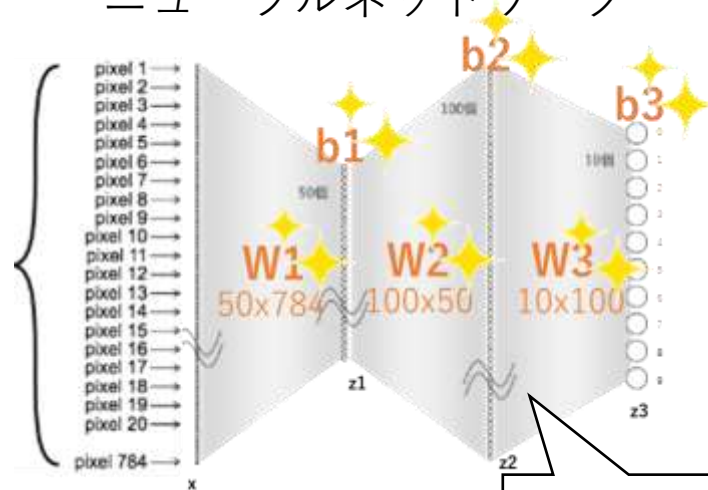
画像前処理



28x28
白黒(2値)

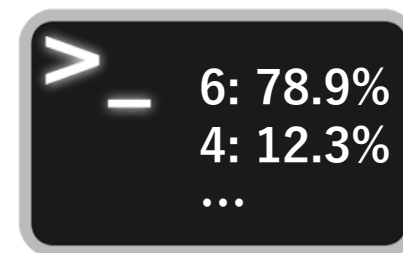


ニューラルネットワーク



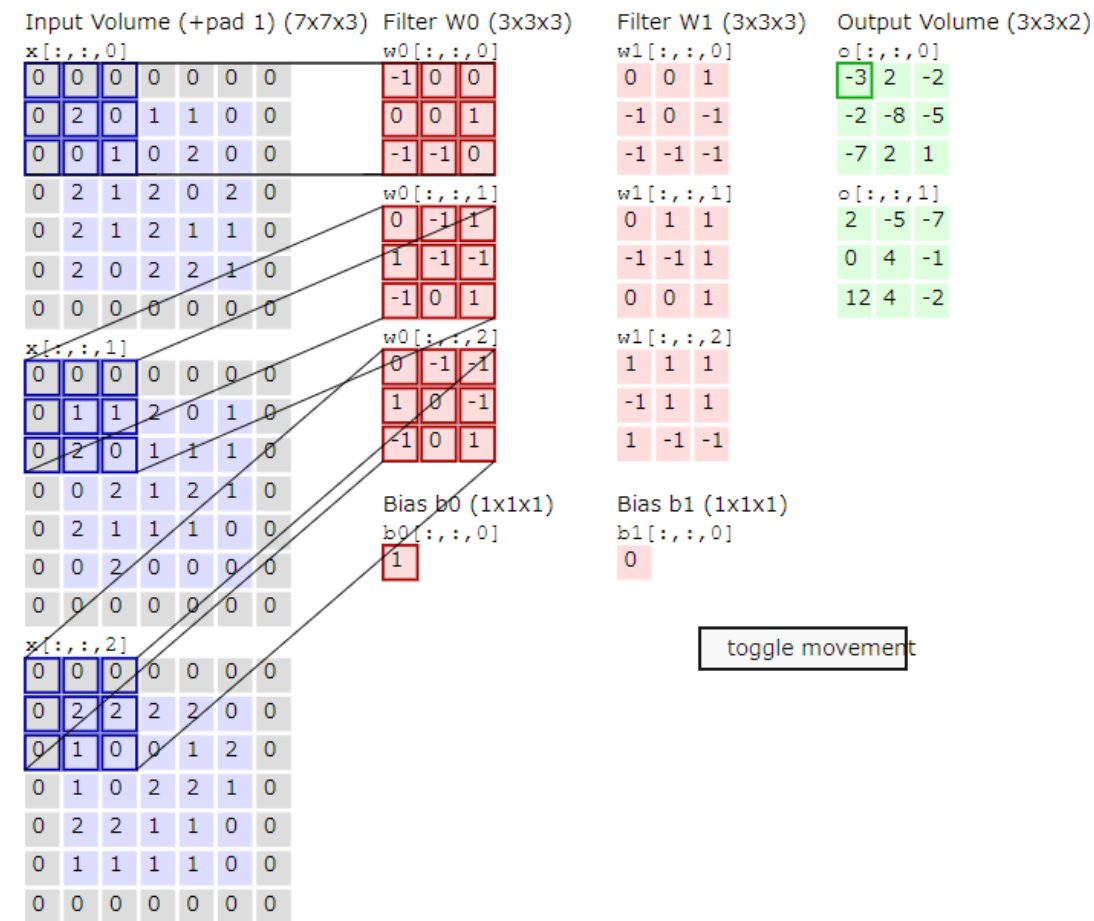
MNISTテスト
データで93%の
認識率

結果の出力

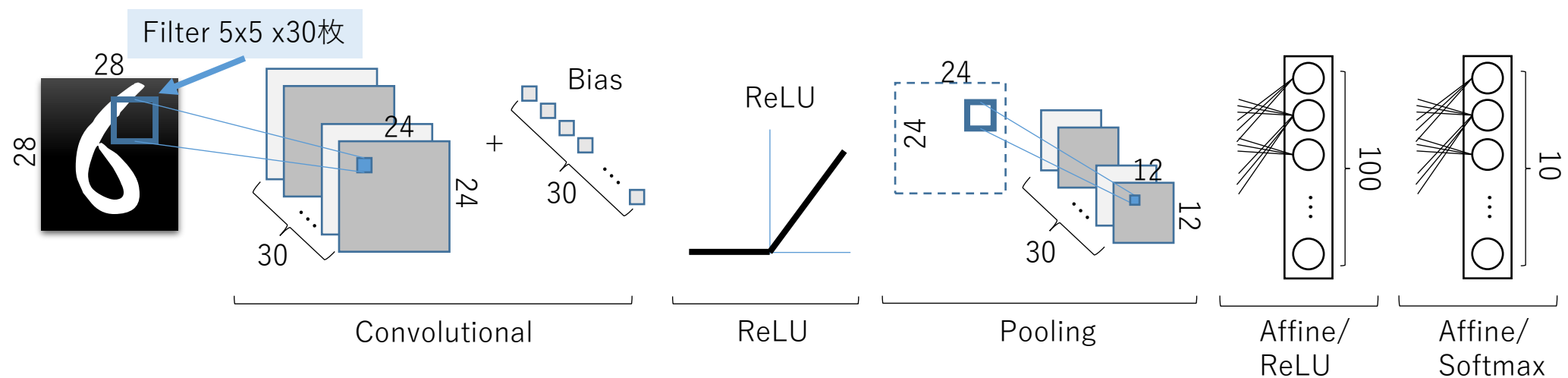


さらに精度を上げる ~畳み込みニューラルネットワーク~

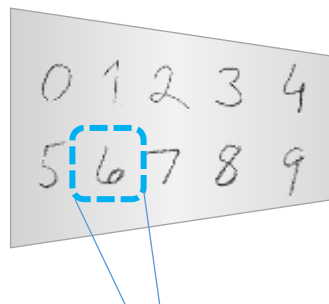
アニメーションでざっくりとしたイメージを理解する
<http://cs231n.github.io/convolutional-networks/>



評価に用いる畳み込みニューラルネットワークの構成



digit_recognition_CNN.pyの概要

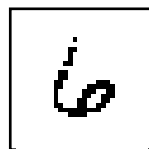


撮影



224x224
カラー

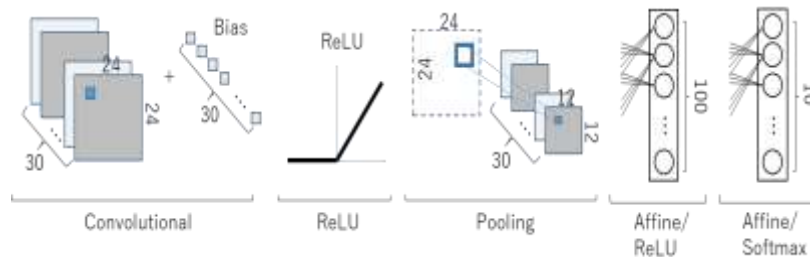
画像前処理



28x28
白黒(2値)

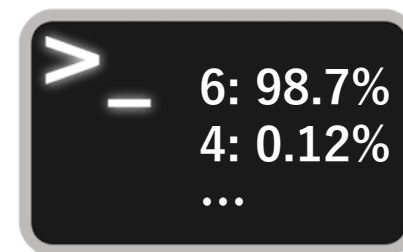


畳み込みニューラルネットワーク



MNISTテスト
データで99%の
認識率

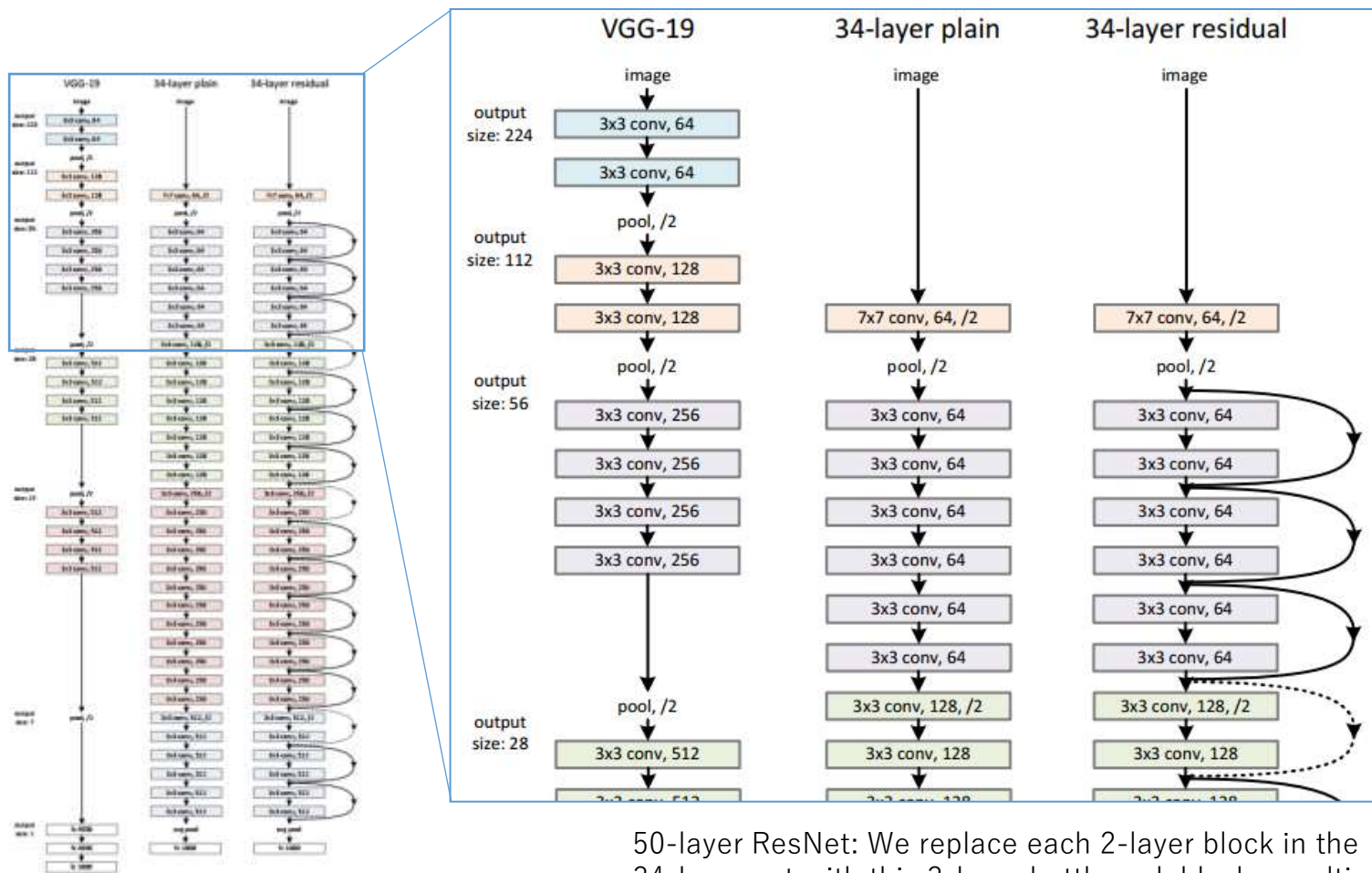
結果の出力



今日の時間配分

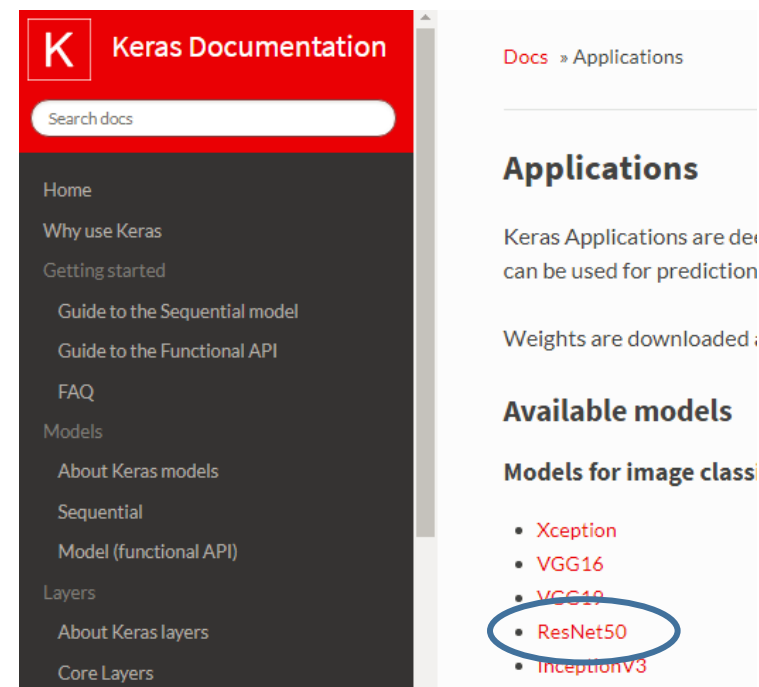
10:00 – 12:00	<ul style="list-style-type: none">• ラズパイ基本セットアップ• カメラを使った画像配信
13:00 – 15:00	<ul style="list-style-type: none">• ニューラルネットワーク概説• 手書き文字認識システムの実装とテスト
15:30 – 16:30	<ul style="list-style-type: none">• 物体識別システムの実装とテスト
16:30 – 17:00	<ul style="list-style-type: none">• 学びの振り返り

image_classification_resnet50.pyの概要



50-layer ResNet: We replace each 2-layer block in the 34-layer net with this 3-layer bottleneck block, resulting in a 50-layer ResNet

<https://keras.io/applications/>



上記サイトで公開されているKerasライブラリを用いたResNet50の実装コードをベースに、カメラ画像を取り込むように変更。