第4回
y=ax+bから始める
初心者向けML講座

これまでのあらすじ

タスク別の出力層の設計

- 基本的なAIは出力層の設計を変えることで様々なタスクに応用できる。
- 具体的には大きく分けて回帰（数値予測）、2分類問題、多分類問題の3種類がある。
- だいたいこれを使ったり、組み合わせたりすれば（データさえあれば）基本的な判断するAIが作れる。
  （ただし、画像生成・文章生成・強化学習などはさらに別の仕組みを追加する必要がある。）

- **回帰**は出力層は**1個**、**活性化関数無し**
- **2分類**は出力層は**1個**、活性化関数は**シグモイド関数**
- **多分類**は出力層は**n個**、活性化関数は**ソフトマックス関数**

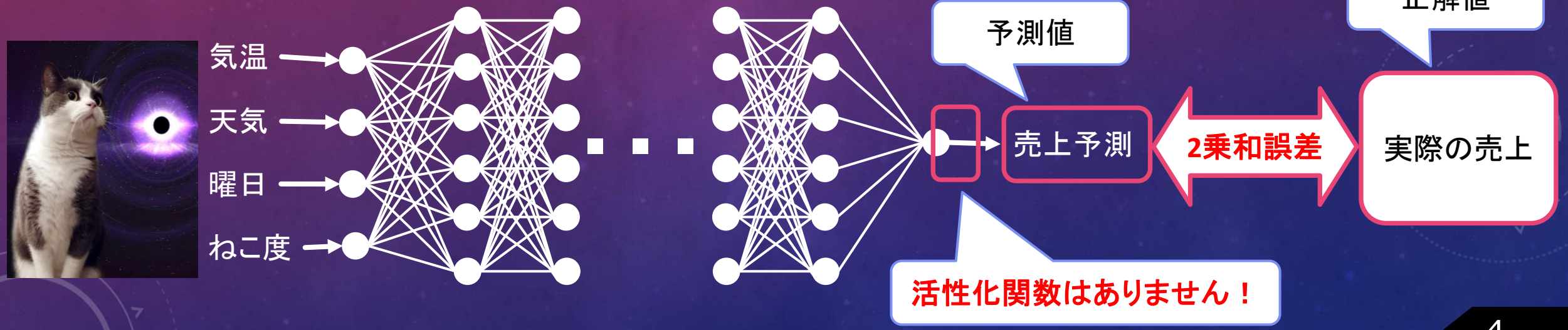
- **y=ax+bは流石に今回は関係なかった…**

回帰（数値予測）のタスク

- 数値予測の場合は**出力層は1個**
- **出力層に活性化関数は使用しない**（そのまま予測値として使う）≒恒等関数ともいう
- 誤差の求め方は**2乗和誤差**

予測値：y'　正解値：y　$(y' - y)^2$

例：宇宙猫屋の売上予測



2分類問題のタスク

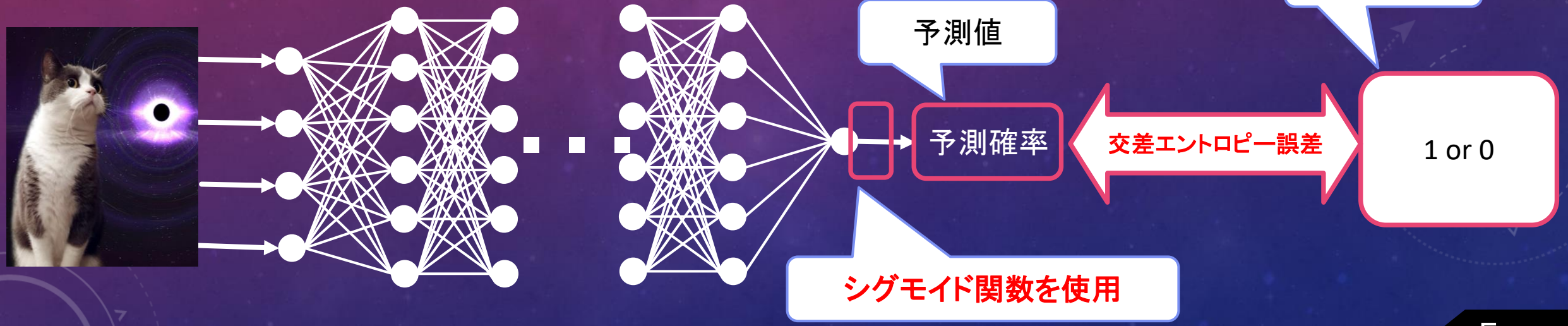
- 2分類予測の場合は**出力層は1個**
- 出力層の活性化関数は**シグモイド関数**を使用
- 誤差の求め方は**交差エントロピー誤差**

交差エントロピー誤差
予測確率がどれだけ出にくい
値なのかを表す計算式。

予測値：y'　正解値：y

$$-y\log y' - (1-y)\log(1-y')$$

例：画像が猫かどうか判定する（猫である＝1、猫でない＝0とする）



多分類問題のタスク

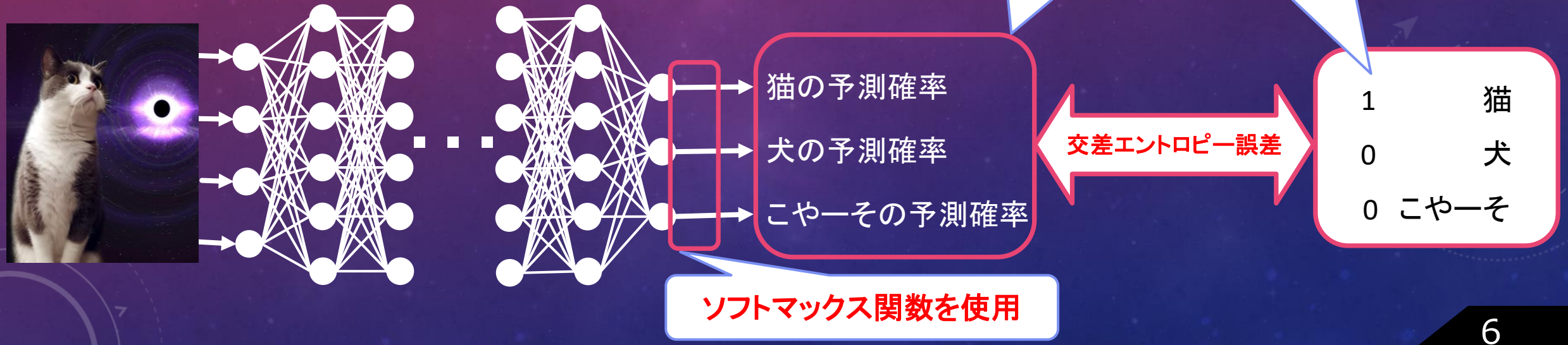
- 多分類予測の場合は**出力層はn個（nは予測したい分類数）**
- 活性化関数は**ソフトマックス関数**を使用
- 誤差の求め方は**交差エントロピー誤差**

予測値：y'　　正解値：y

$$-\sum_{k=0}^{K-1} y_k \, log \, y'_k$$

正解yの1がある所の
$-\log y'_k$
が誤差になる

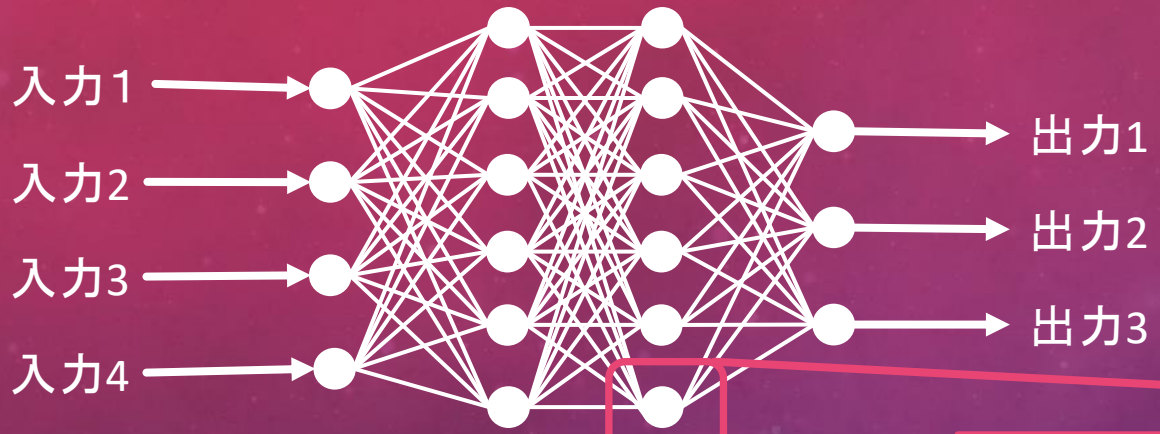
例：画像が猫かどうか判定する（「猫」「犬」「こやーそ」のどれかを予測）



画像を使ったニューラルネットワーク

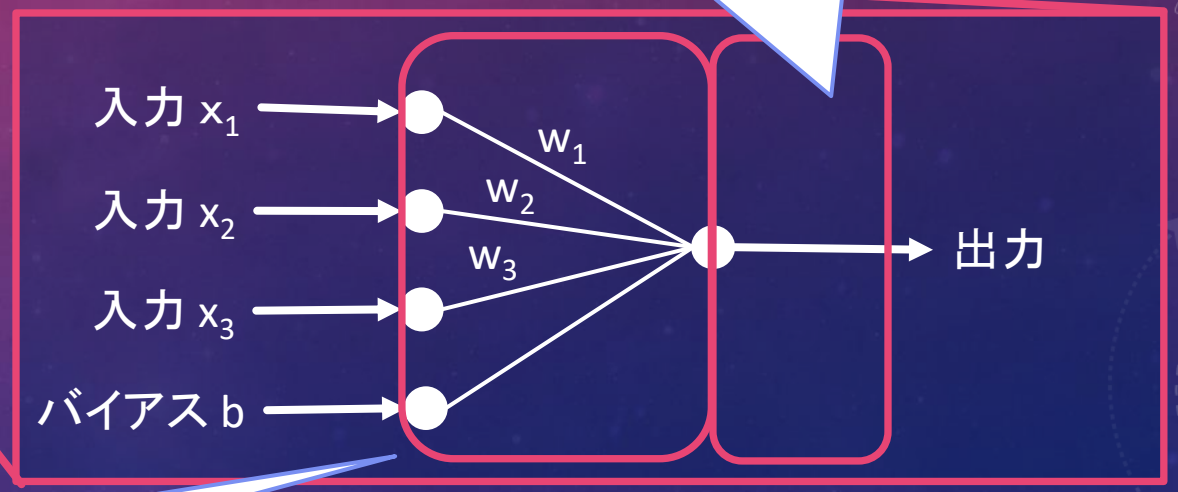
畳み込みニューラルネットワーク（CNN）

ディープラーニングの基本はニューラルネットワーク（NN）



ノード＝パーセプトロン

活性化関数を通して出力

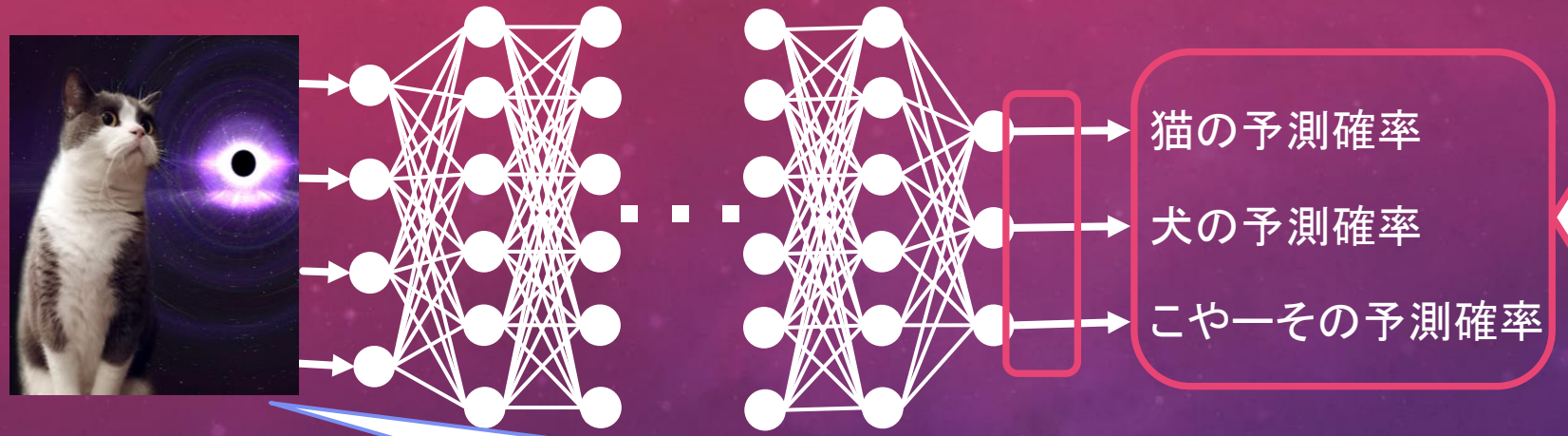


入力値×重み
を全て足す。
※重みwが学習するパラメータ

$$\begin{cases} a=x_1*w_1+x_2*w_2+x_3*w_3+b \\ y=f(a) \end{cases}$$

DNNにどのように画像を入れるか？

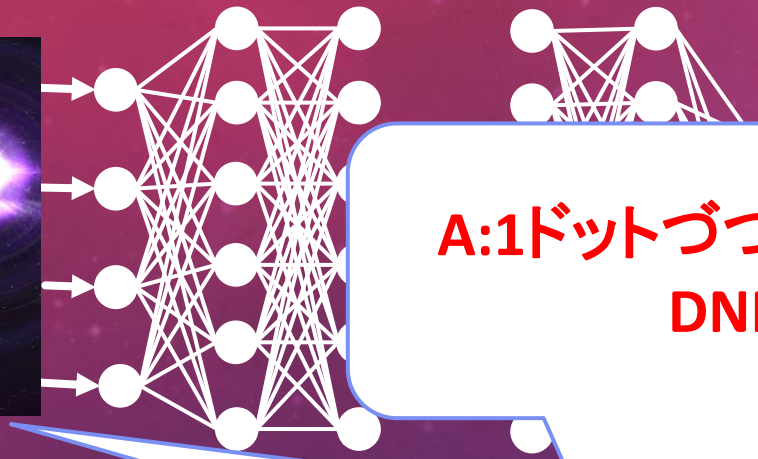
例：画像が猫かどうか判定する（「猫」「犬」「こやーそ」のどれかを予測）



Q:この画像をどのようにすれば計算できるように
DNNに入れる事が出来るでしょうか？

DNNにどのように画像を入れるか？

例：画像が猫かどうか判定する（「猫」「犬」「こやーそ」のどれかを予測）



A:1ドットづつ色の強弱のパラメータを
DNNに入れていく！

ントロピー誤差

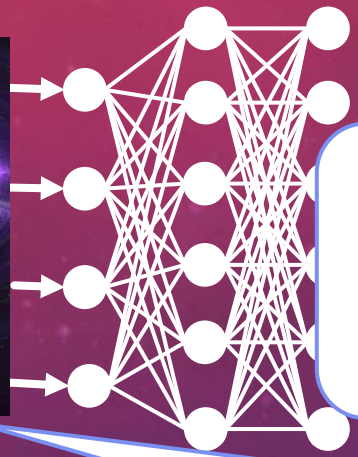
猫が答えの場合
の正解値

1	猫
0	犬
0	こやーそ

Q:この画像をどのようにすれば計算できるように
DNNに入れる事が出来るでしょうか？

DNNにどのように画像を入れるか？

例：画像が猫かどうか判定する（「猫」「犬」「こやーそ」のどれかを予測）



猫が答えの場合
の正解値

1	猫
0	犬
0	こやーそ

ントロピー誤差

**A:1ドットづつ色の強弱のパラメータを
DNNに入れていく！**

Q:この画像をと
DNNに入

**100px×100pxのカラー(RGB)画像の場合
100×100×3=3万個の入力データになる！**

DNN…るか？

そいつは素敵だ

例：画像が…ーそ」のどれかを予測）

猫が答えの場合
の正解値

1	猫
0	犬
0	こやーそ

…ントロピー誤差

A:1ドットづつ色の強弱のパラメータを
DNNに入れていく！

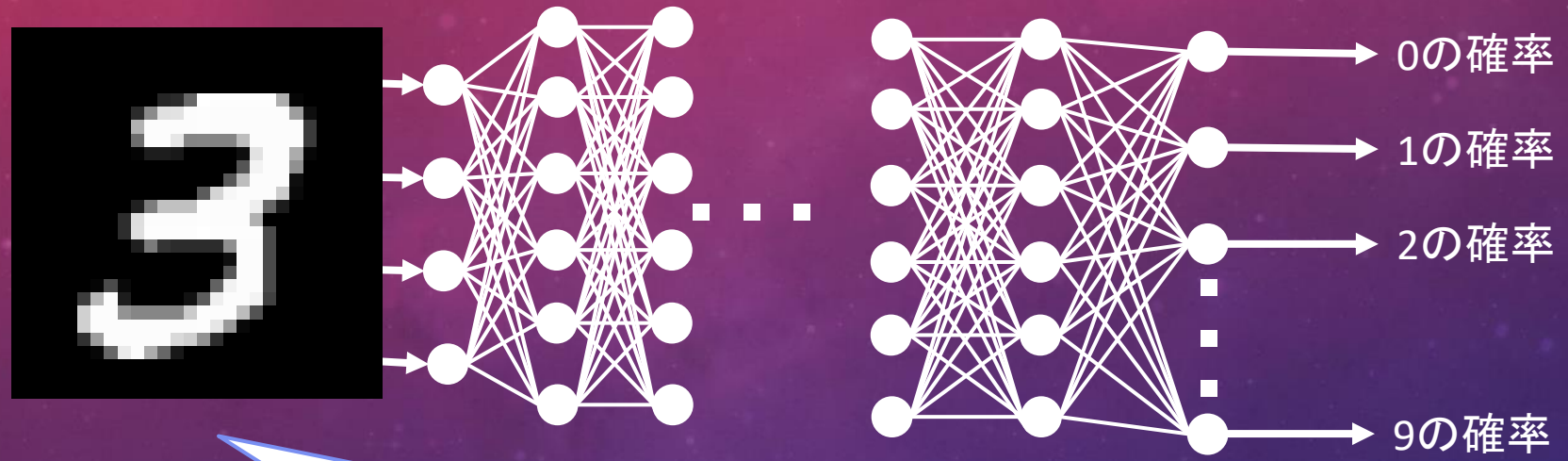
…の画像をと…
…DNNに入…

100px×100pxのカラー(RGB)画像の場合
100×100×3=3万個の入力データになる！

面白くなってきた

DNNにどのように画像を入れるか？モノクロ画像の場合

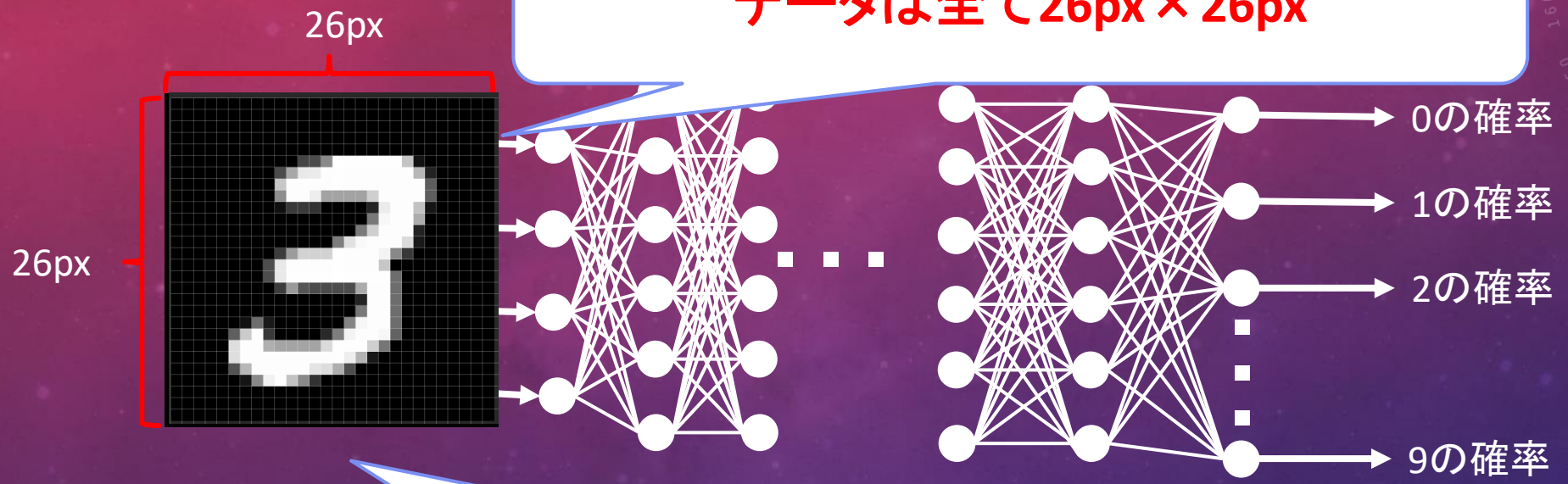
例：mnistを使った画像判定AIのモデルイメージ



Mnistとは手書きの数字画像を学習用に6万枚、
テスト用に1万枚集めたデータセット
もちろん正解データもついている

DNNにどのように画像を入れるか？モノクロ画像の場合

例：mnistを使った画像判定AIのモ



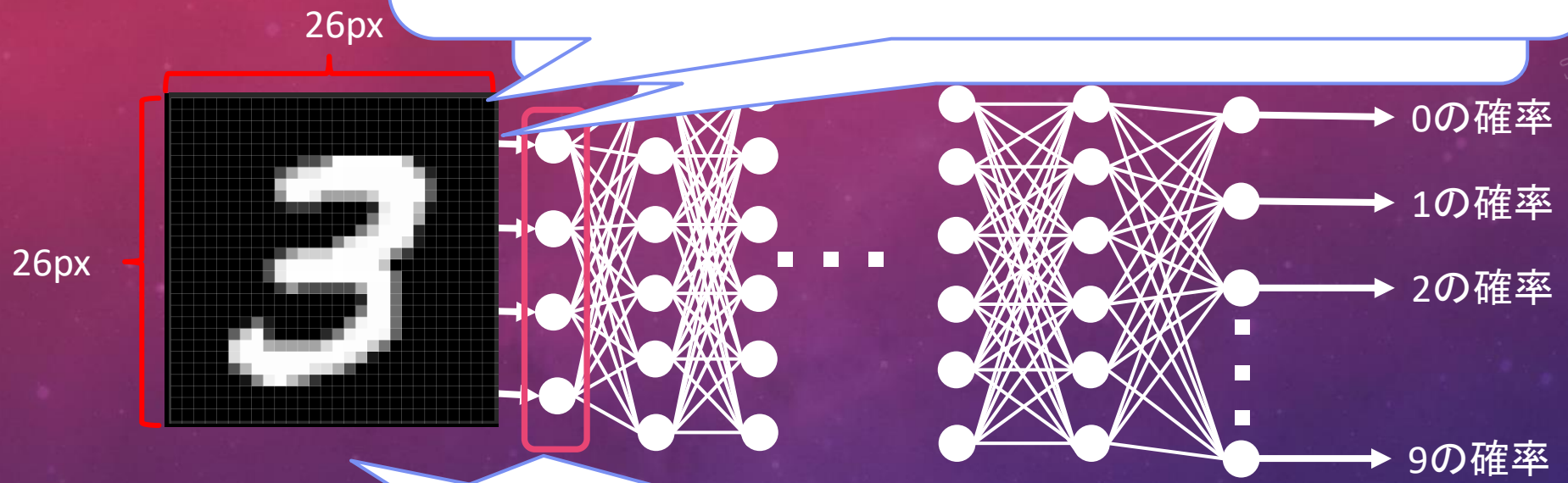
データは全て26px×26px

Mnistとは手書きの数字画像を学習用に6万枚、
テスト用に1万枚集めたデータセット
もちろん正解データもついている

DNNにどのよう　合

例：mnistを使った画像判定

1ドットづつ黒い部分が0、白い部分が255の
0～255までの255段階で色の強さのパラメーターがあります。
つまり、入力は26×26=676個のデータを入力



**※一般的には0～255では学習する重みが大きくなりすぎるため
前処理で255で割ってからNNに入力していきます。**

DNNにどのよう　合

例：mnistを使った画像判定

の確率
の確率
の確率
9の確率

1ドットづつ黒い部分が0、白い部分が255の
0～255までの255段階で色の強さのパラメーターがあります。
つまり、入力は26×26=676個のデータを入力

後は自由にモデルの層とノード数を
決めるだけ！

ね、かんたんでしょ？

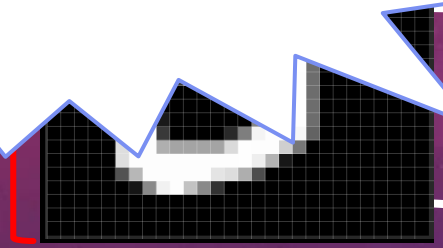
**※一般的には0～255では学習する重みが大きくなりすぎるため
前処理で255で割ってからNNに入力していきます。**

DNNにどのよう　合

例：mn　を使った　像判定

1ドットづつ黒い部分が0、白い部分が255の
0～255までの255段階で色の強さのパラメーターがあります。
つまり、入力は26×26=676個のデータを入力

うまくいくと思った？

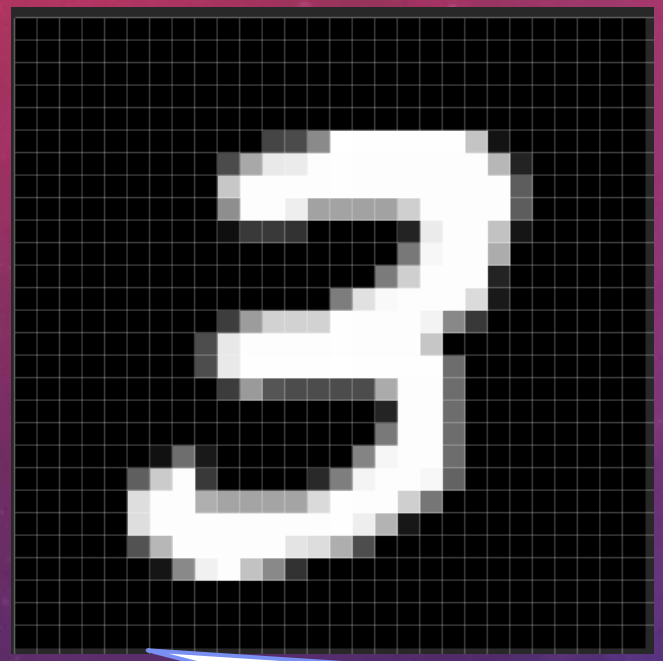


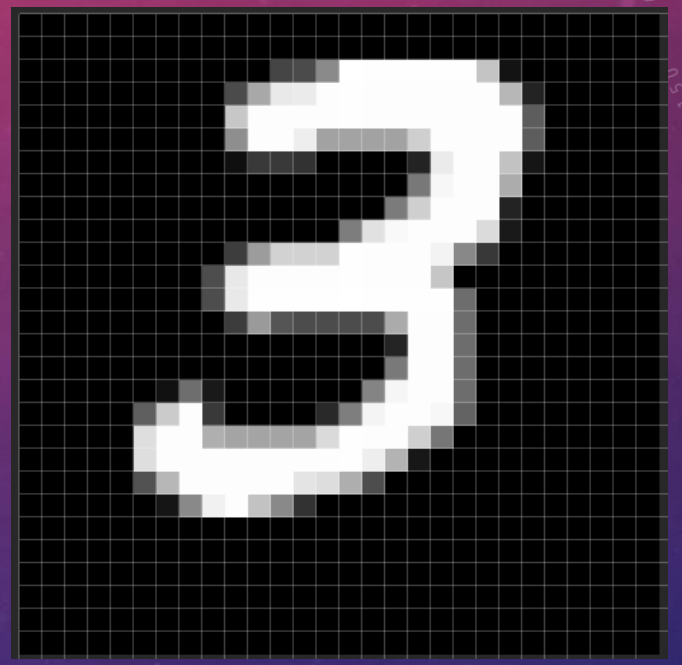
残念！
クルツちゃんでした!!!

※一般的には0～255で　　大きくなりすぎるため
前処理で255で　　ていきます。

DNNにどのように画像を入れるか？モノクロ画像の場合

人間的には全く同じ3の画像ですが、データ的には大きく違う





左は130個以上0のデータが続いた後に数値が出てくるが、
右は50個ちょいで数値が出てくる。
データの並びは全然違う！

DNNどのように画像を入れるか？モノクロ画像の場合

計算でこれを同じと認識？

データ的には大きく違う

できらぁ！

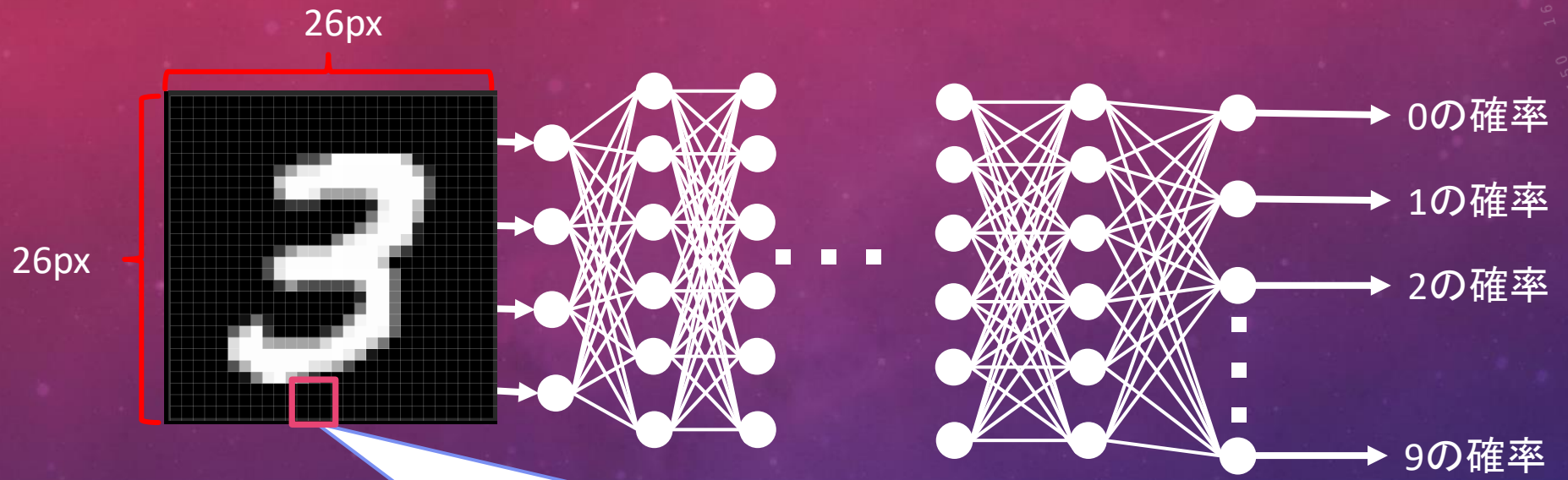
左は130個以上0のデータが続いた後に数値が出てくる。
右は50個ちょいで数値が出てくる。
データの並びは全然違う！

畳み込み(Convolutional)処理

畳み込みニューラルネットワーク（CNN)

DNNにどのように画像を入れるか？モノクロ画像の場合

例：mnistを使った画像判定AIのモデルイメージ



画像で重要なのはデータの前後関係ではなく
あるドットを見たときに上下左右の矩形(四角形)
の関係性が重要

DNNにどのように

例：mnistを使った画像判定AIのモ

1層ごとに上下左右の関係を維持したまま
圧縮（畳み込み）するように計算していけば
上下左右の関係を維持したまま計算ができる！

画像で重要なのはデータの前後関係ではなく
あるドットを見たときに上下左右の矩形(四角形)
の関係性が重要

畳み込み処理の計算方法１

入力X=

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

フィルターF=

1	1	1
2	0	1
−1	−1	0

・入力値に対応したフィルターを用意する

・フィルターは畳み込みしたいサイズによって決まる（一般的には3x3、5x5などの小さい正方形サイズ）

・フィルターの値はDNNの重みと同じくランダムで決める。

畳み込み処理の計算方法２

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

×　フィルターF

1	1	1
2	0	1
-1	-1	0

＝　出力Y

4	

・入力Xの左上にフィルターFを重ねて掛け算をして全て足し合わせたものを出力Yとする。

・上図の場合1×1+2×1+3×1+5×2+6×0+7×1+9×-1+10×-1+11×0＝４

畳み込み処理の計算方法２

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＝ 出力Y

4	

・入力Xの左上にフィルターFを重ねて掛け算をして全て足し合わせたものを出力Yとする。

・上図の場合1×1+2×1+3×1+5×2+6×0+7×1+9×-1+10×-1+11×0＝４

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＝ 出力Y

4	8

・右に1ドットずらして同様にフィルターFと重ねて掛けて足し合わせる。

畳み込み処理の計算方法３

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

×　フィルターF

1	1	1
2	0	1
−1	−1	0

＝　出力Y

4	8
20	

・一番右まで行ったら次は1ドット下にずらして左端から同様に行う。

畳み込み処理の計算方法３

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＝ 出力Y

4	8
20	

・一番右まで行ったら次は1ドット下にずらして左端から同様に行う。

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＝ 出力Y

4	8
20	24

・右下まで1ドットづつずらしながらフィルターをかけます。

畳み込み処理の計算方法４

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＝ 出力Y

4	8
20	24

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＋ バイアスb

3

＝ 出力Y

7	11
23	27

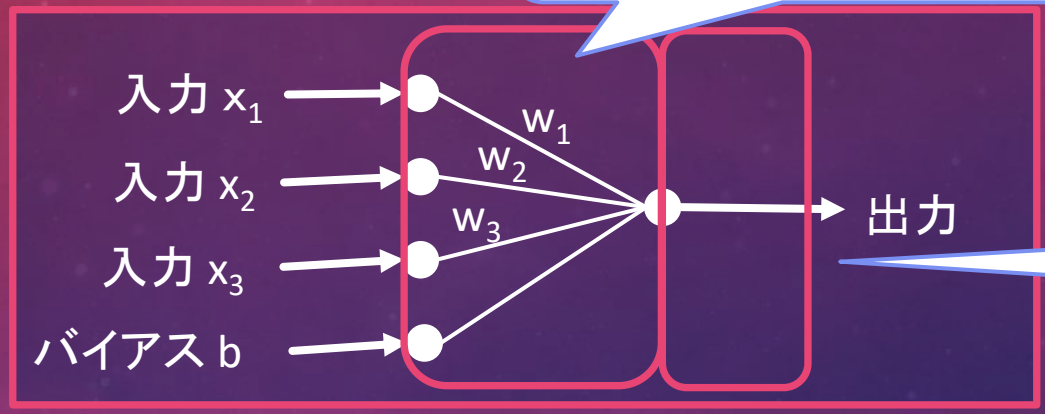
・最後に**バイアスｂを足し**たら**活性化関数**を通せば正式な出力Ｙとなります。

※本来は**ストライド**、**パディング**という処理もありますが、一気に覚えても大変なので省略します。

畳み込み処理の計算方法４

入力値×重み
を全て足す。
※重みwとbが学習するパラメータ

パーセプトロン



活性化関数

畳み込み処理

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

×

フィルターF

1	1	1
2	0	1
−1	−1	0

＋ バイアスb 3 ＝ 出力Y

7	11
23	27

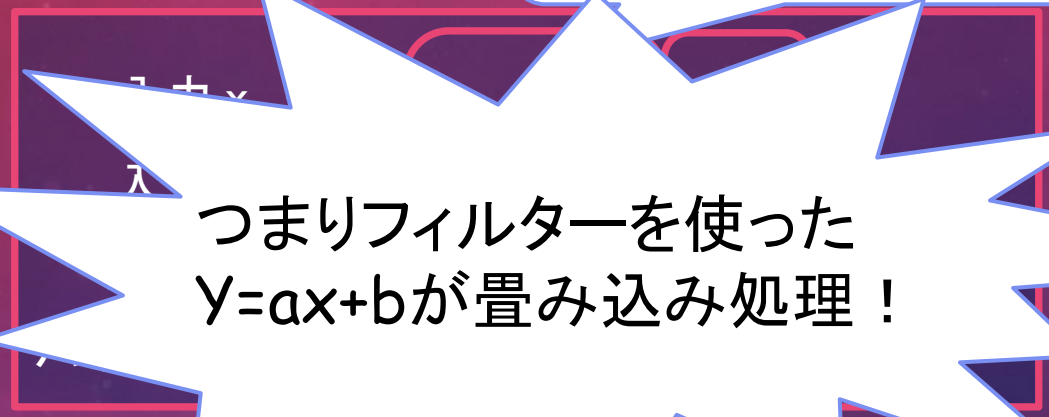
入力値×フィルター
を全て足す。
※フィルターとbが学習するパラメータ

出力Yを活性化関数に通す

畳み込み処理の計算方法４

入力値×重み
を全て足す。
※重みwとbが学習するパラメータ

パーセプトロン



つまりフィルターを使った
Y=ax+bが畳み込み処理！

活性化関数

畳み込み処理

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＋ バイアスb 3 ＝ 出力Y

7	11
23	27

入力値×フィルター
を全て足す。
※フィルターとbが学習するパラメータ

出力Yを活性化関数に通す

畳み込み処理の計算方法４

入力値×重み
を全て足す。
※重みwとbが学習するパラメータ

パーセプトロン

ね、かんたんでしょ？

つまりフィルターを使った
Y=ax+bが畳み込み処理！

活性化関数

処理

3	4
7	8
11	12
15	16

×

フィルターF

1	1	1
2	0	1
−1	−1	0

＋ バイアスb 3 ＝ 出力Y

7	11
23	27

入力値×フィルター
を全て足す。
※フィルターとbが学習するパラメータ

出力Yを活性化関数に通す

畳み込み処理の計算方法（おまけ）

入力X

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

× フィルターF

1	1	1
2	0	1
−1	−1	0

＝ 出力Y

4	8
20	

・面倒な計算をしているように思いますが、フィルターを転置（行と列を入れ替え）ると

入力X’

1	2	3
5	6	7
9	10	11

× フィルターF’

1	2	−1
1	0	−1
1	1	0

・入力X’は横方向、フィルターF’は縦方向に掛けて足し合わせる計算になります。
（数学の行列の掛け算）

プログラム上ではnumpyライブラリを使用し、np.dot(X’, F’)といった形で計算が出来ます。
（TensorFlowやPyTorchなどはこの辺りは勝手に良い感じにやってくれます）

CNNのモデルイメージ

例：mnistを使った画像判定AIのモデルイメージ

入力サイズがどのようなサイズでも同様に行います。
これにより**DNNよりも計算量**が少なく
上下左右の関係を維持したまま計算ができる！

CNNのモデルイメージ

例：mnistを使った画像判定AIのモデルイメージ

Convolutional Neural Network(CNN)
では各層で複数個のフィルターを用意して
複数パターンの出力を行い学習していきます。

CNNのモデルイメ

例：mnistを使った画像判定AIのモ

畳み込みを行うときの**フィルター1個**が
DNNの**パーセプトロン1個**に相当し、
1枚1枚の**出力画像**が**パーセプトロン1個分の出力**と
考えるとわかりやすいかもしれません。

Convolutional Neural Network(CNN)
では各層で複数個のフィルターを用意して
複数パターンの出力を行い学習していきます。

畳み込みの後はプーリング処理

- CNN畳み込みが終わったら次はプーリング処理を行うのが一般的です。
- 基本となるのはマックスプーリングとアベレージプーリングの2つが基本の処理です。

マックスプーリング

- マックスプーリングの場合もフィルター（枠のサイズだけ決める）を用意します。(今回は2×2のフィルターとします。)

フィルター（枠だけ）

入力x

1	2	3	4
5	6	7	8
16	15	14	13
12	11	10	9

マックスプーリング

- マックスプーリングの場合もフィルター（枠のサイズだけ決める）を用意します。(今回は2×2のフィルターとします。)
- 枠のサイズで入力を分割していきます。

フィルター（枠だけ）

入力x

1	2	3	4
5	6	7	8
16	15	14	13
12	11	10	9

マックスプーリング

- マックスプーリングの場合もフィルター（枠のサイズだけ決める）を用意します。(今回は2×2のフィルターとします。)
- 枠のサイズで入力を分割していきます。
- 分割した中で最も大きいものが出力となるのがマックスプーリングです。

フィルター（枠だけ）

入力X

1	2	3	4
5	6	7	8
16	15	14	13
12	11	10	9

→ 出力Y

6	8
16	14

アベレージプーリング

- アベレージプーリングの場合もフィルター（枠のサイズだけ決める）を用意します。(今回は2×2のフィルターとします。)
- 枠のサイズで入力を分割していきます。
- 分割した中の平均値を出力に使うのがアベレージプーリングになります。

フィルター（枠だけ）

入力X

1	2	3	4
5	6	7	8
16	15	14	13
12	11	10	9

→ 出力Y

3.5	5.5
13.5	11.5

アベレージプーリング

- アベレージプーリングの場合もフィルター（枠のサイズだけ決める）を用意します。(今回は2×2のフィルターとします。)
- 枠のサイズで入力を分割していきます。
- 分割した中の平均値を出力に使うのがアベレージプーリングになります。

フィルター（枠だけ）

入力X

1	2	3	4
5	6	7	8
16	15	14	13
12	11	10	9

→ 出力Y

3.5	5.5
13.5	11.5

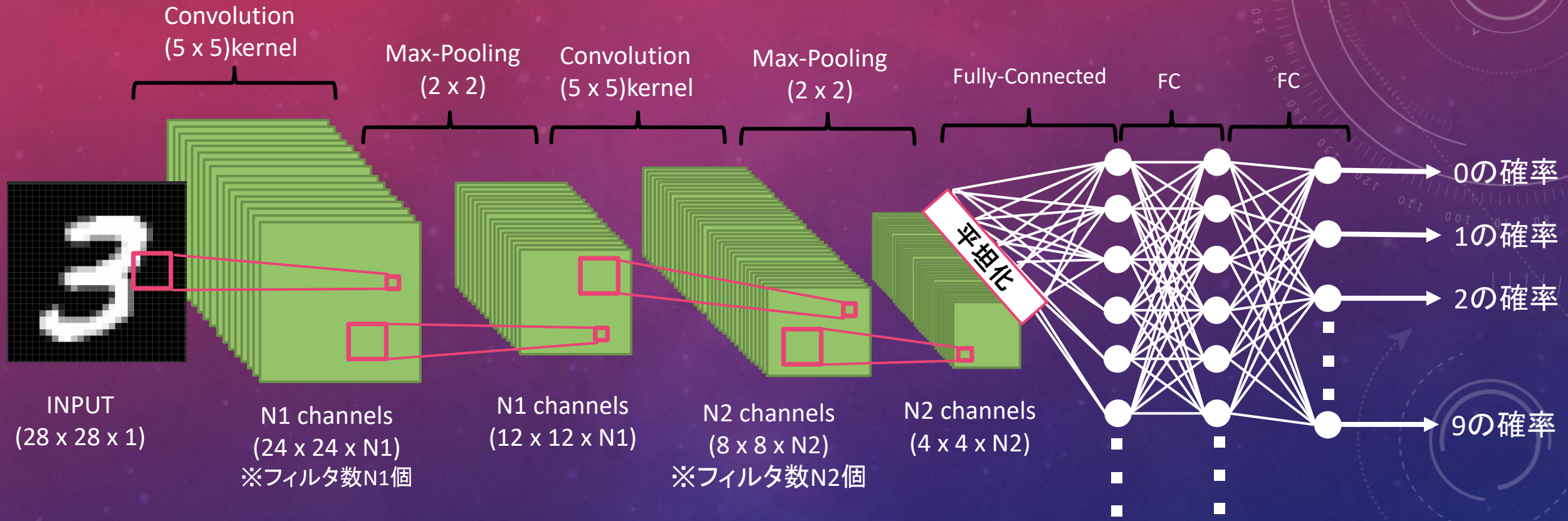
プーリングは単純なものなので
学習パラメータも活性化関数も
ありません。

最終的なCNN(Convolutional Neural Network)

- 最終的に**畳み込み処理**と**プーリング処理**を交互に行い、DNN（全結合層）とつなげるとCNNとなります。

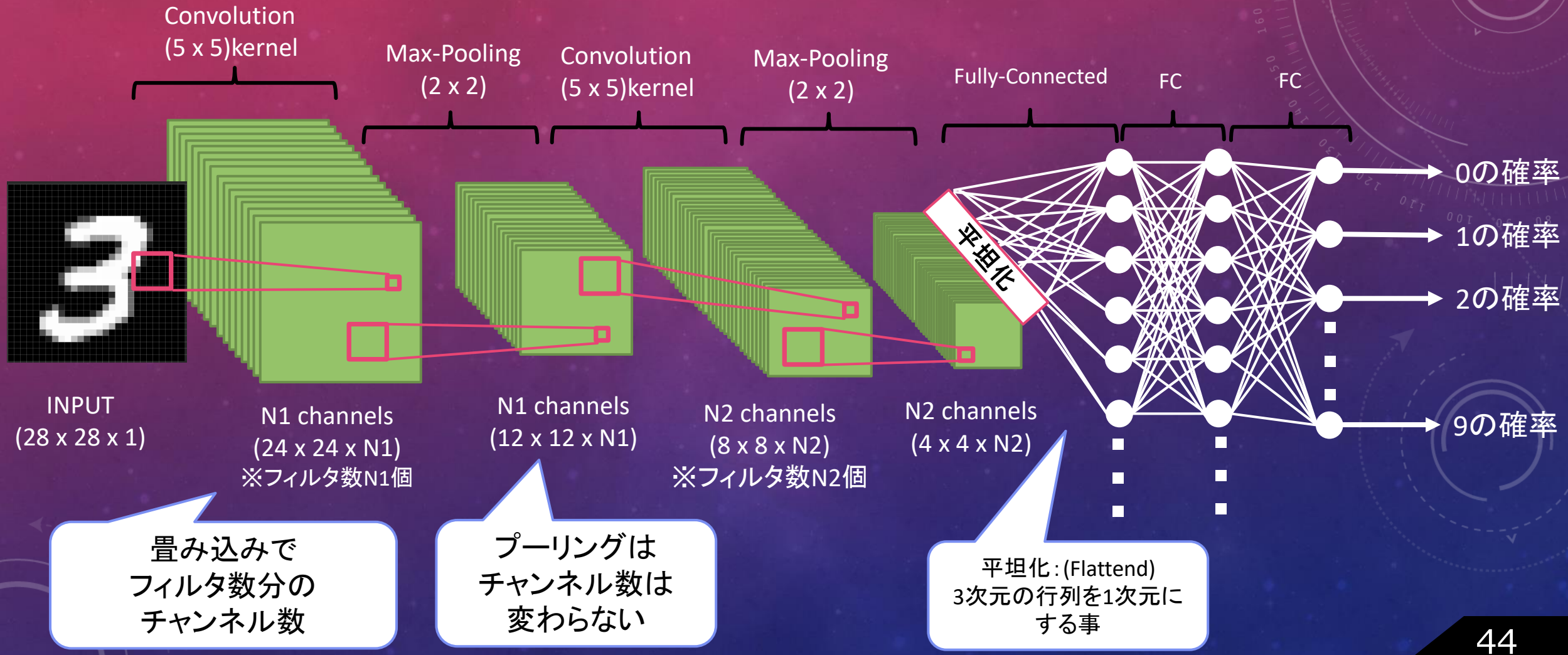
最終的なCNN(Convolutional Neural Network)

- 最終的に**畳み込み処理**と**プーリング処理**を交互に行い、DNN（全結合層）とつなげるとCNNとなります。



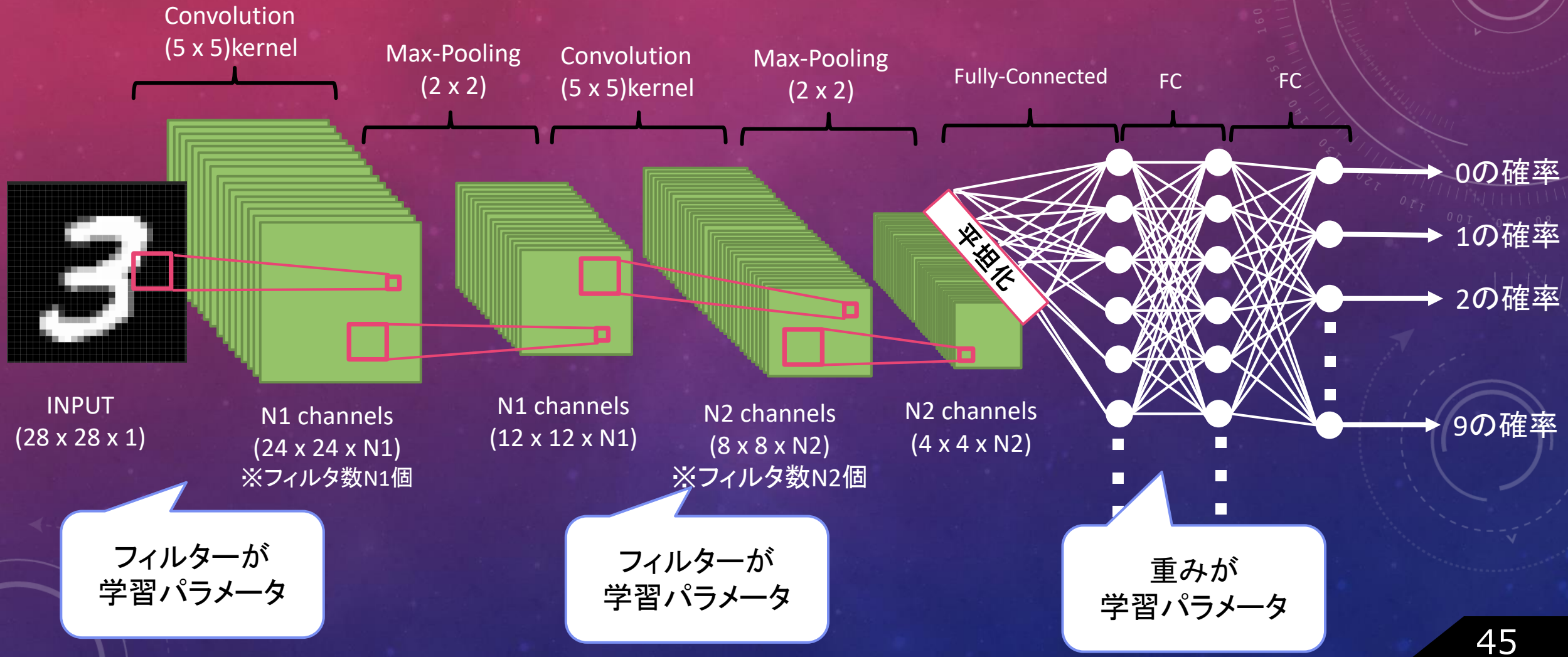
最終的なCNN(Convolutional Neural Network)

- 最終的に**畳み込み処理**と**プーリング処理**を交互に行い、DNN（全結合層）とつなげるとCNNとなります。



最終的なCNN(Convolutional Neural Network)

- 最終的に**畳み込み処理**と**プーリング処理**を交互に行い、DNN（全結合層）とつなげるとCNNとなります。



まとめ

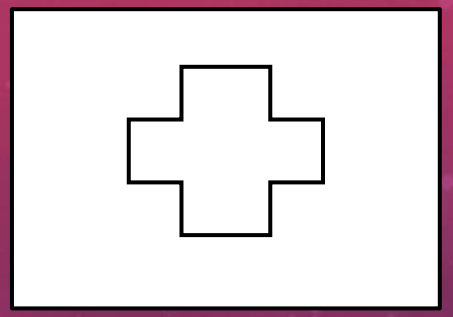
- CNNは画像を扱う際に適したNNである。
- CNNは畳み込み処理とプーリング処理を何度か行い画像を認識しやすくしたモデル
- 畳み込み処理はフィルタを使ったy=ax+b（フィルタのパラメーターを使って特徴を抽出する）
- プーリングは決めた枠のサイズの中で最大値や平均を取って出力する。（データのずれをぼやかす効果がある）
- 最終的に**畳み込み処理**と**プーリング処理**を交互に行い、DNN（全結合層）とつなげるとCNNとなります。

おわり

ありがとうございました

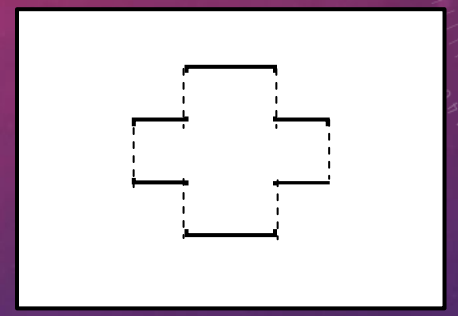
畳み込み処理の効果

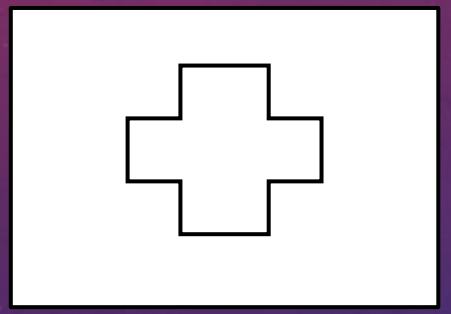
- 畳み込み処理は何らかの特徴を抽出する効果がある。



×　フィルターF=

0	0	0
1	1	1
0	0	0





×　フィルターF=

0	1	0
0	1	0
0	1	0