5.コンピュータビジョンのためのディープラーニング

17t4055g 築地俊平

5.1 畳み込みニューラルネットワークの紹介

本節は
CNNの簡単なプログラムを動かし、内容を理解していく

5.1 畳み込みニューラルネットワークの紹介

- 基本的なCNNのインスタンス化

```
>>> model1 = models.Sequential()
>>>
model1.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
>>> model1.add(layers.MaxPooling2D((2,2)))
>>> model1.add(layers.Conv2D(64,(3,3),activation='relu'))
>>> model1.add(layers.MaxPooling2D((2,2)))
>>>
model1.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
```

5.1 畳み込みニューラルネットワークの紹介

```
>>> model1.summary()
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 3, 32)          18464
=================================================================
Total params: 37,280
Trainable params: 37,280
Non-trainable params: 0
_________________________________________________________________
```

5.1 畳み込みニューラルネットワークの紹介

```
>>> model1.add(layers.Flatten())
>>> model1.add(layers.Dense(64,activation='relu'))
>>> model1.add(layers.Dense(10,activation='softmax’))
```

10個の出力を持つ最終層とソフトマックス活性化関数を使用することで10ユニットの分類を行う

5.1 畳み込みニューラルネットワークの紹介

```
>>> model1.summary()
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 3, 32)          18464
_________________________________________________________________
flatten_1 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                18496
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 56,426
Trainable params: 56,426
Non-trainable params: 0
_________________________________________________________________
```

5.1 畳み込みニューラルネットワークの紹介

```
17  from keras.datasets import mnist
18  from keras.utils import to_categorical
19
20  (train_images,train_labels),(test_images,test_labels)=mnist.load_data()
21
22  train_images=train_images.reshape((60000,28,28,1))
23  train_images=train_images.astype('float32')/255
24
25  test_images=test_images.reshape((10000,28,28,1))
26  test_images=test_images.astype('float32')/255
27
28  train_images = to_categorical(train_labels)
29  test_images = to_categorical(test_labels)
30
31  model.compile(optimizer = 'rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
32  model.fit(train_images,train_labels,epochs=5,batch_size=64)
33
```

5.1 畳み込みニューラルネットワークの紹介

>>>test_loss,test_acc=model.evaluate(test_images.test_labels)

>>>test_acc

0.9931999999999997

2章の全結合ネットワークの正解率97.8%なので、精度が上がっている。

5.1.1 畳み込み演算

- 全結合層と畳み込み層の違い

→全結合層…入力特徴空間からパターンを学習

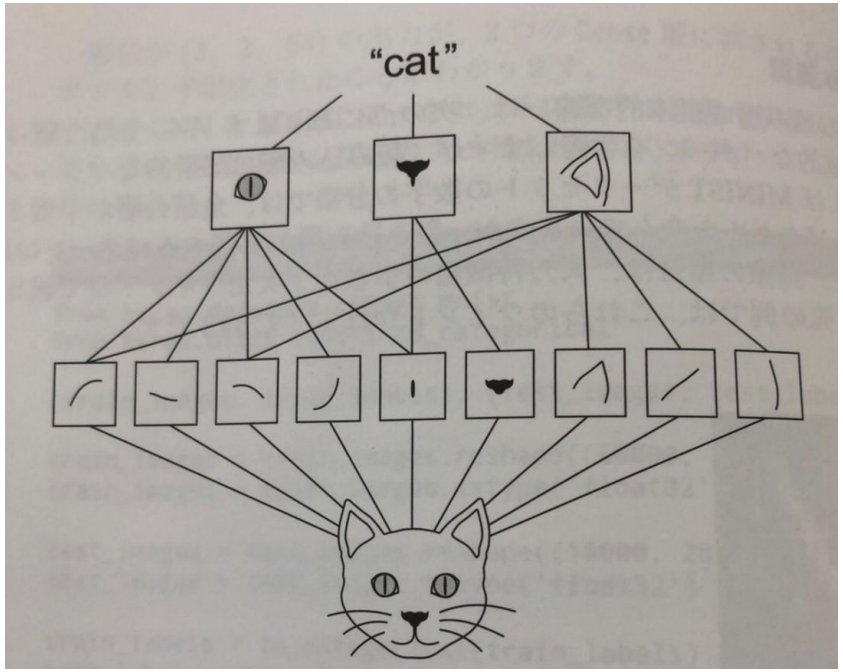
→畳み込み層…局所的なパターンを学習

5.1.1 畳み込み演算

- CNNの特徴

→CNNが学習するパターンは移動不変

→CNNはパターンの区間階層を学習できる



局所的なエッジが目や耳といった局所的な物体と結合される→猫

5.1.1　畳み込み演算

畳み込み演算は

2つの空間軸（幅と高さ）と1つの軸（チャネル軸）に基づき

特徴マップ（feature map）と呼ばれる3次元テンソルに対して実行される。

5.1.1 畳み込み演算

MNISTの例

入力…畳み込み層はサイズが(28,28,1)の特徴マップ

出力…サイズ(26,26,32)と特徴マップ

32個の出力フィルターはそれぞれ26×26グリッドの値を含んでいる

5.1.1 畳み込み演算

畳み込みは二つのパラメータで定義される

・入力から抽出されたパッチのサイズ
→通常は3×3か5×5

・出力特徴マップの深さ
→畳み込みによって計算されるフィルタの数

KerasConv2D層では
Conv2D(output_depth,(window_height,window_width))で定義される

5.1.1 畳み込み演算

サイズが3×3のウインドウをスライド
↓
3次元パッチを抽出
↓
1次元ベクトルに変換
→畳み込みカーネル(convokution kernle)
↓
すべてのベクトルが3次元の出力マップ
として空間的に再結合

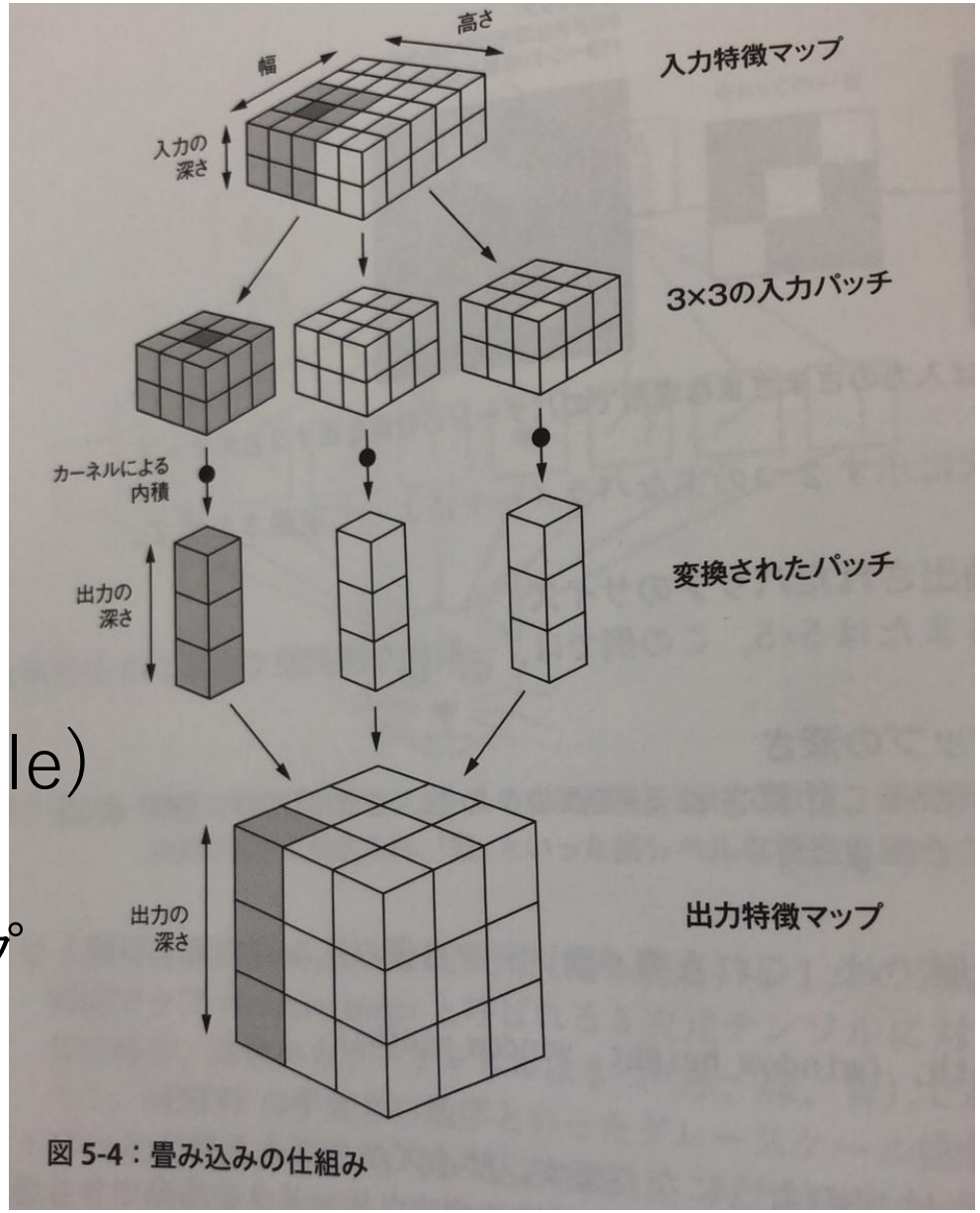


図 5-4：畳み込みの仕組み

5.1.1 畳み込み演算

出力の幅と高さは、入力の幅と高さとは異なる場合がある

理由としては
- 周辺効果
- ストライドの使用

5.1.1 畳み込み演算

周辺効果とパンディング

入力と同じ空間次元を持つ出力特徴マップを取得したい場合

→パンディング(padding)

→入力特徴マップの上下左右に適切な数の行と列を追加することで畳み込みウインドウの中心をすべてで移動できるようにする

5.1.1 畳み込み演算

周辺効果とパンディング

×を与えることで5×5の
出力特徴マップができる

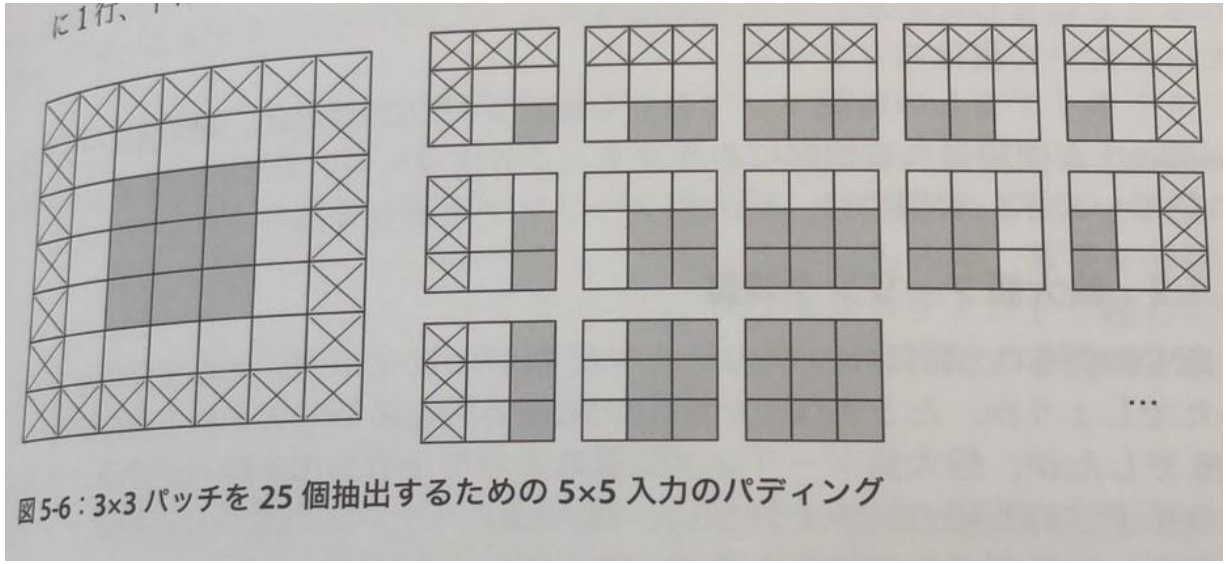


図5-6：3×3 パッチを 25 個抽出するための 5×5 入力のパディング

5.1.1 畳み込み演算

畳み込みのストライド
畳み込みの説明は中心のタイルが連続していないかもしれない
→連続しているときストライド1

ストライド2の3×3の畳み込み

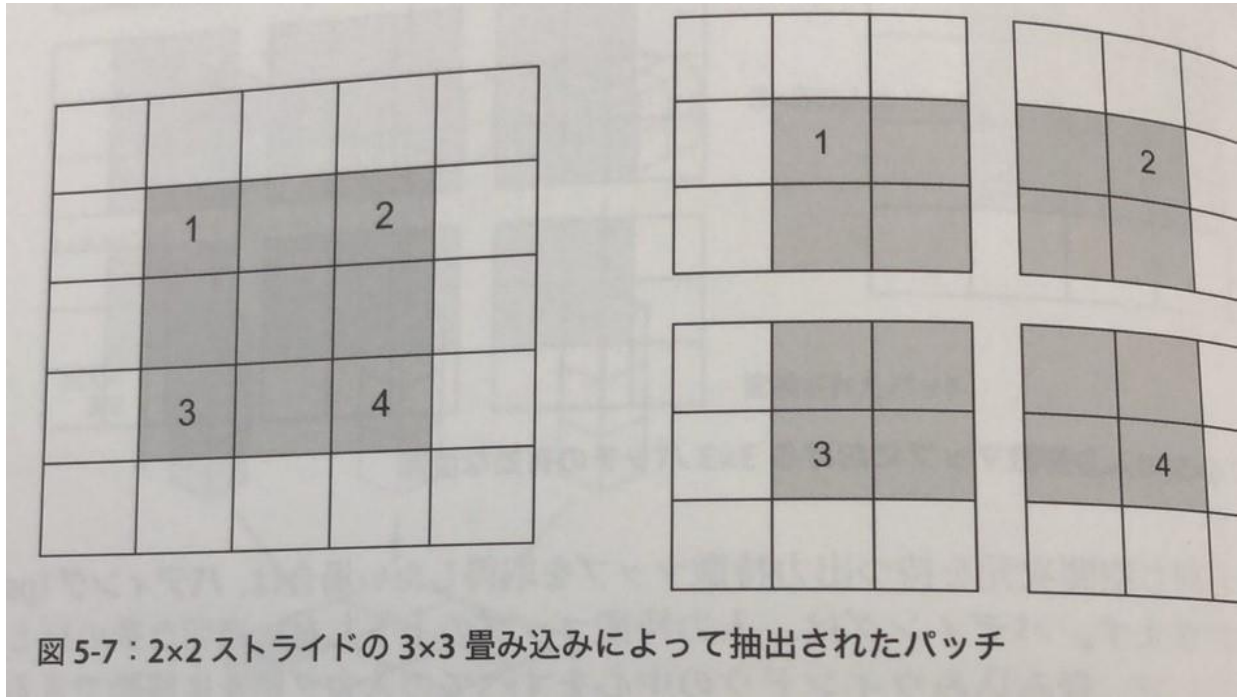


図 5-7：2×2 ストライドの 3×3 畳み込みによって抽出されたパッチ

5.1.2 最大プーリング演算

最大プーリング演算

→通常2×2ウインドウとストライド2を使って実行される

畳み込み

→3×3ウインドウとストライドなし（ストライド1）で実行される

5.1.2 最大値プーリング変換

```
>>> model_no_max_pool = models.Sequential()
>>> model_no_max_pool.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
>>> model_no_max_pool.add(layers.Conv2D(64,(3,3),activation='relu'))
>>> model_no_max_pool.add(layers.Conv2D(64,(3,3),activation='relu'))
>>> model_no_max_pool.summary()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 64)        36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

5.1.2 最大値プーリング変換

・特徴量の空間階層の学習に貢献しない

→3つ目の層の3×3ウインドウに含まれているのは最初の7×7の情報のみ。CNNによって学習される高レベルのパターンは最初の入力からすれば依然として非常に小さいものの、数字を分類されるための学習には不十分の可能性

・最終的な特徴マップが非常に大きい

→最終的な特徴マップのサンプル一つで22×22×64=30970個

サイズが512にし、平坦化したとき、1580万個のパラメータになる

5.1.2 最大値プーリング変換

要するに…

ダウンサンプリングの目的

→処理の対象となる特徴マップの係数の数を減らす

→連続する畳み込みが調べるウインドウを徐々に大きくすることで、空間フィルタ階層を抽出すること

他にも平均値プーリング(average pooling)という手法もある