Deep Learning

https://www.streamingnology.com

@streamingnology

https://github.com/streamingnology

streamingnology

@streamingnology

卷积神经网络：卷积层的作用

提取图像的特征值

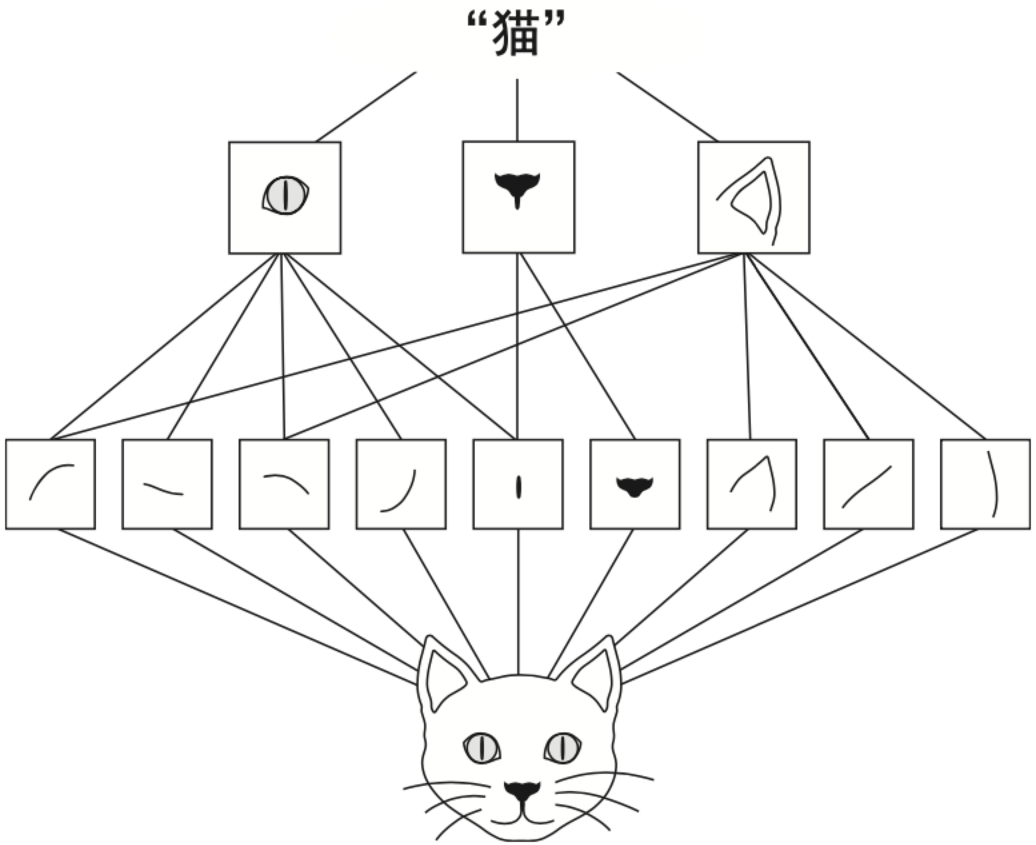
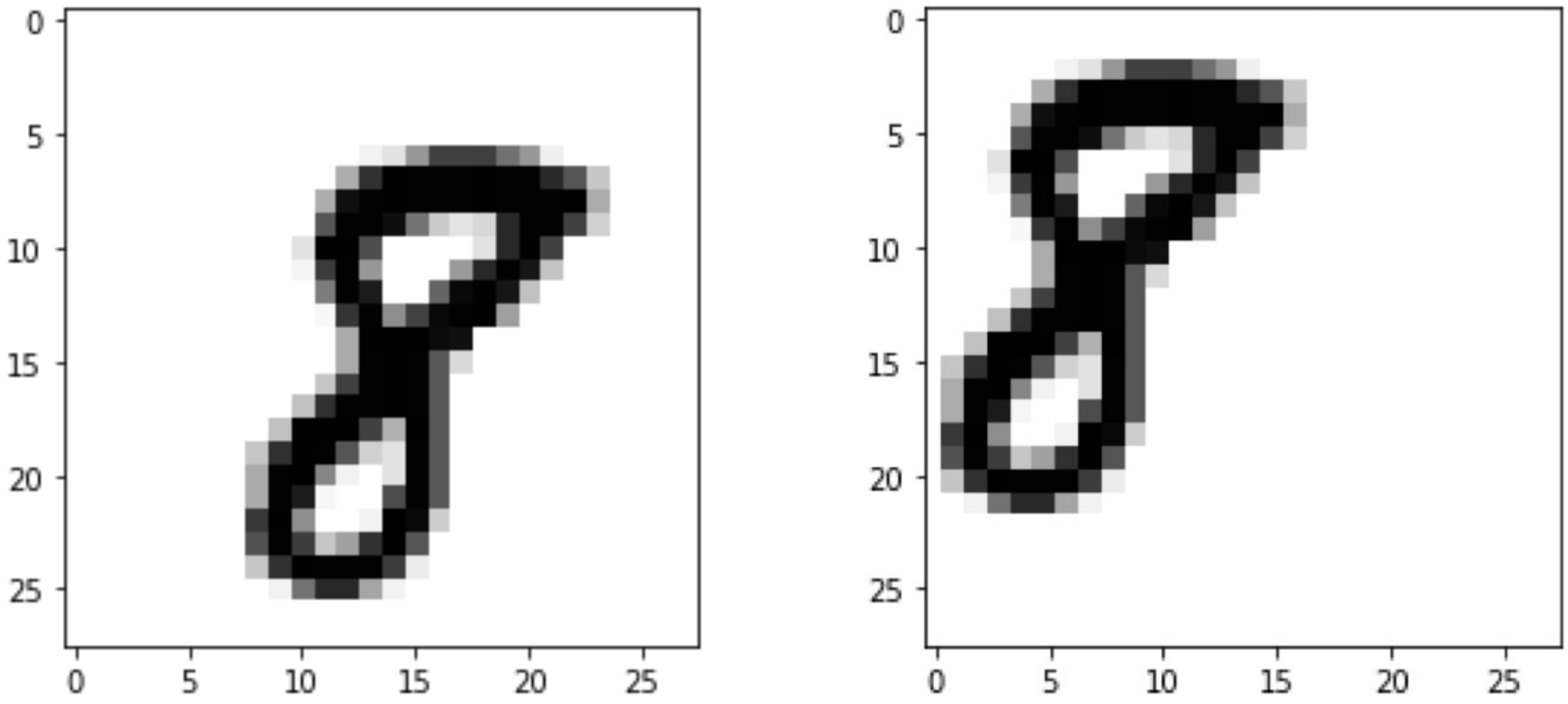


图 5-2　视觉世界形成了视觉模块的空间层次结构：超局部的边缘组合成局部的对象，比如眼睛或耳朵，这些局部对象又组合成高级概念，比如“猫”

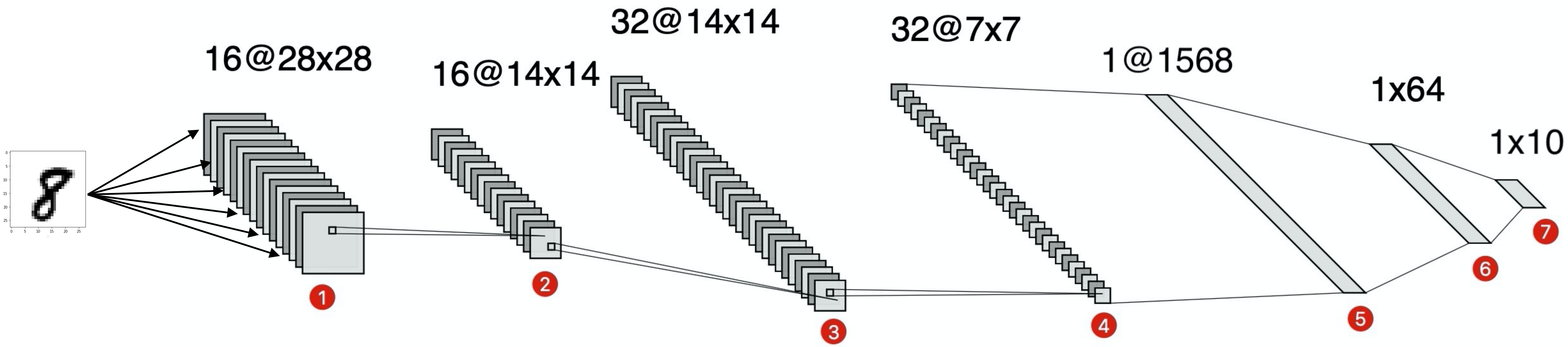
上图来自右边这本书	Deep Learning with Python Python深度学习

卷积神经网络

全连接方式的神经网络无法处理图片的这种变化



卷积神经网络



❶ 卷基层(3x3)

❷ 最大池化层(2x2)

❸ 卷基层(3x3)

❹ 最大池化层(2x2)

❺ 输入层 input layer(Flatten)

❻ 全连接层Dense

❼ 输出层 output layer

卷积神经网络

```
  model = models.Sequential()

❶ model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)))

❷ model.add(layers.MaxPooling2D((2, 2)))

❸ model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))

❹ model.add(layers.MaxPooling2D((2, 2)))

❺ model.add(layers.Flatten())

❻ model.add(layers.Dense(64, activation='relu'))

❼ model.add(layers.Dense(10, activation='softmax'))
```

卷积神经网络

```
model.summary()
```

Model: "sequential"

	Layer (type)	Output Shape	Param #
1	conv2d (Conv2D)	(None, 28, 28, 16)	160
2	max_pooling2d (MaxPooling2D)	(None, 14, 14, 16)	0
3	conv2d_1 (Conv2D)	(None, 14, 14, 32)	4640
4	max_pooling2d_1 (MaxPooling2	(None, 7, 7, 32)	0
5	flatten (Flatten)	(None, 1568)	0
6	dense (Dense)	(None, 64)	100416
7	dense_1 (Dense)	(None, 10)	650

Total params: 105,866
Trainable params: 105,866
Non-trainable params: 0

卷积神经网络

```
mnist = keras.datasets.mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
```

```
validation_images = training_images[50000:60000]
validation_labels = training_labels[50000:60000]

training_images = training_images[0:50000]
training_labels = training_labels[0:50000]

from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

training_images = training_images.reshape((50000, 28, 28, 1))
training_images = training_images.astype('float32') / 255

validation_images = validation_images.reshape((10000, 28, 28, 1))
validation_images = validation_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

training_labels = to_categorical(training_labels)
test_labels = to_categorical(test_labels)
validation_labels = to_categorical(validation_labels)
```

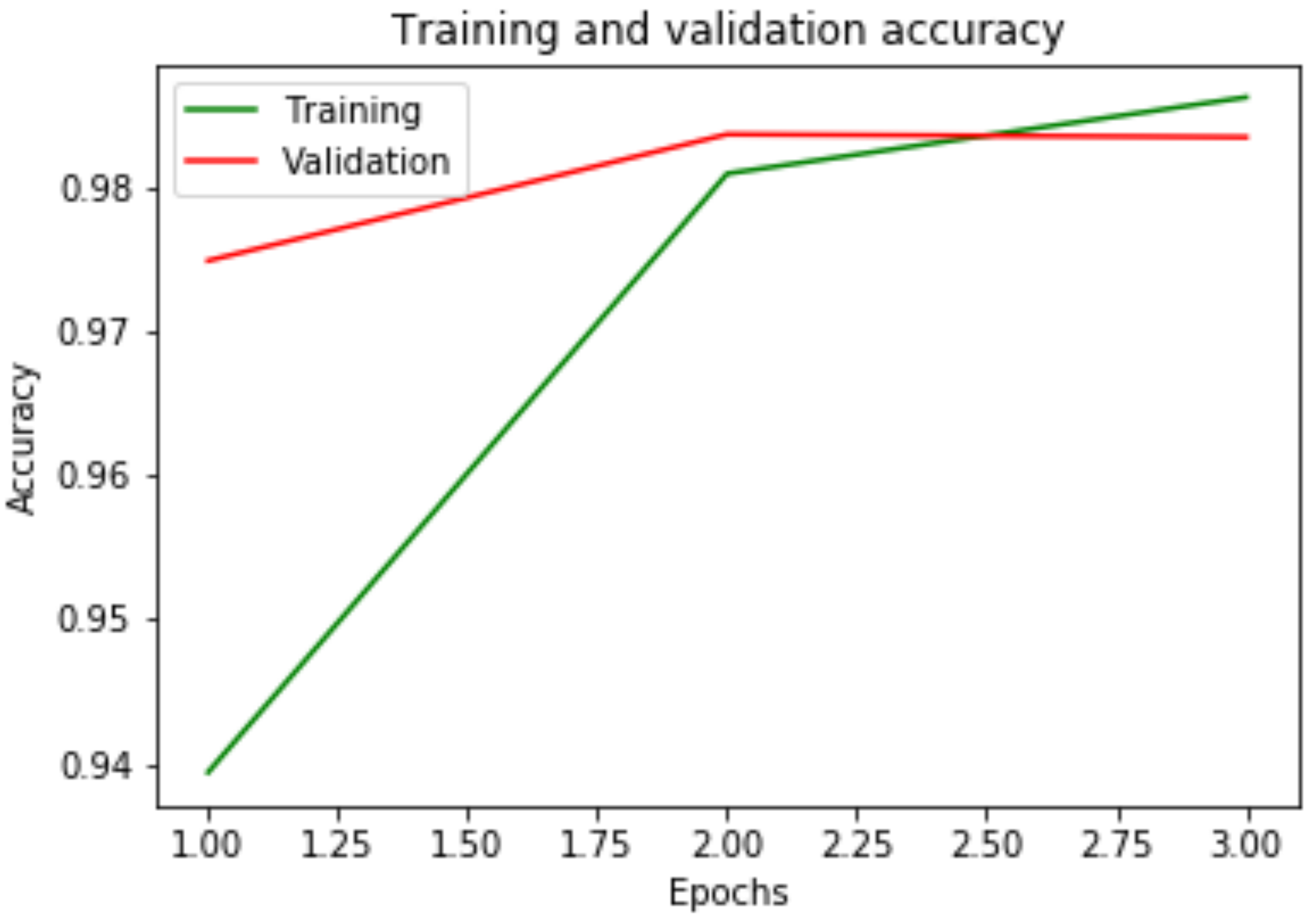
卷积神经网络

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
history = model.fit(training_images, training_labels, epochs=3, batch_size=64,
                    validation_data=(validation_images, validation_labels))
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/3
50000/50000 [==============================] - 19s 383us/sample - loss: 0.2015 - accuracy: 0.9394
- val_loss: 0.0765 - val_accuracy: 0.9750
Epoch 2/3
50000/50000 [==============================] - 19s 370us/sample - loss: 0.0614 - accuracy: 0.9811
- val_loss: 0.0541 - val_accuracy: 0.9838
Epoch 3/3
50000/50000 [==============================] - 19s 373us/sample - loss: 0.0439 - accuracy: 0.9864
- val_loss: 0.0555 - val_accuracy: 0.9836
```

卷积神经网络



Training and validation accuracy
Training
Validation
0.98
0.97
0.96
0.95
0.94
Accuracy
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Epochs

Reference

1. Publication-ready NN-architecture schematics.
   http://alexlenail.me/NN-SVG/LeNet.html

END

http://

https://www.streamingnology.com

@streamingnology

https://github.com/streamingnology

YouTube

streamingnology

微博

@streamingnology