Introduction to image recognition with deep learning

2020-11-02

Bioconductor Virtual Conference Asia 2021

Jianqiang Sun
https://aabbdd.jp

物体分类

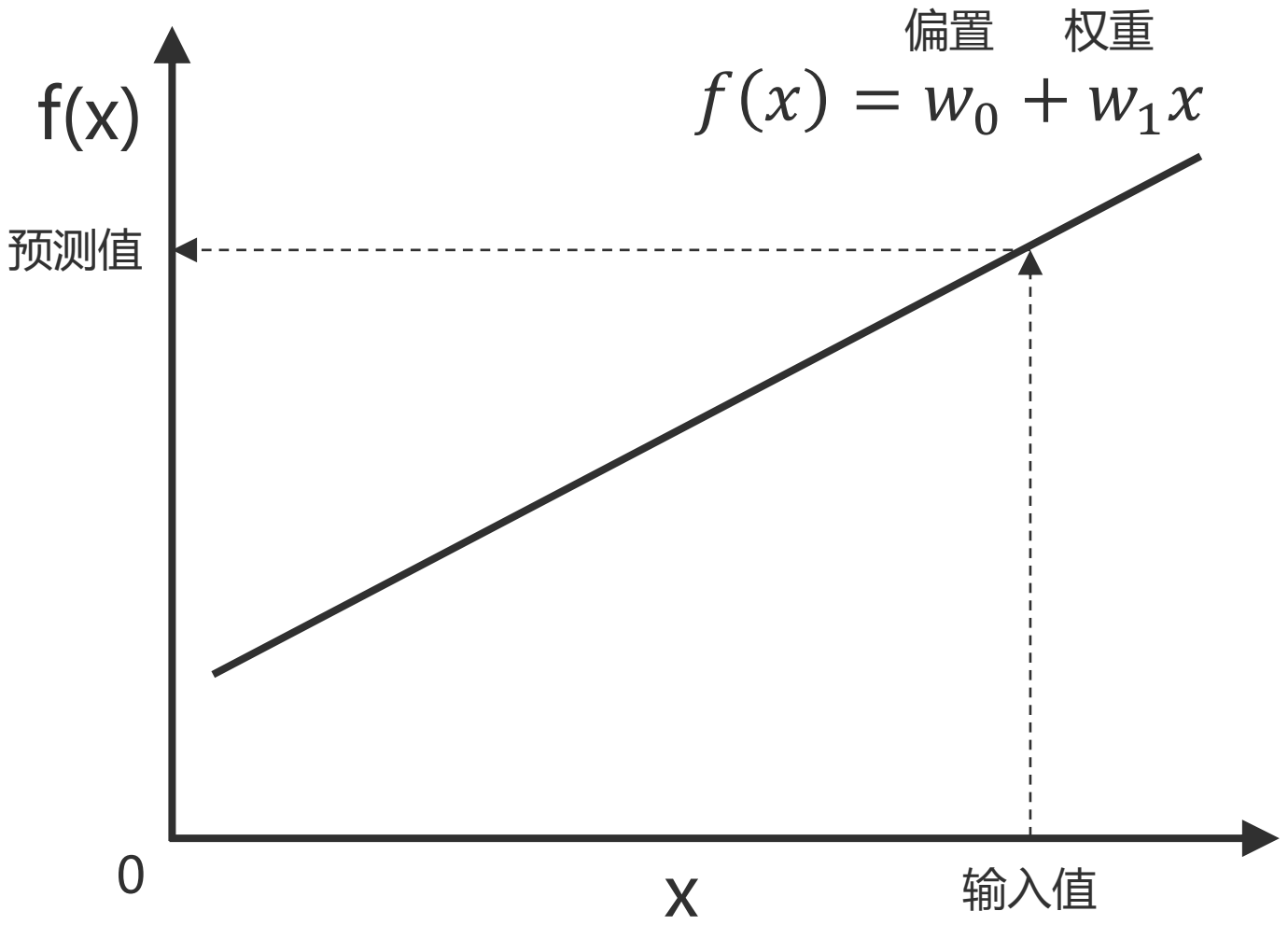
- 神经网络
- 卷积神经网络
- torch

函数

假设某种农作物的产量只被气温影响，我们可以尝试使用简单的线性回归来预测产量。譬如：

$$f(x) = w_0 + w_1 x$$

产量　　气温



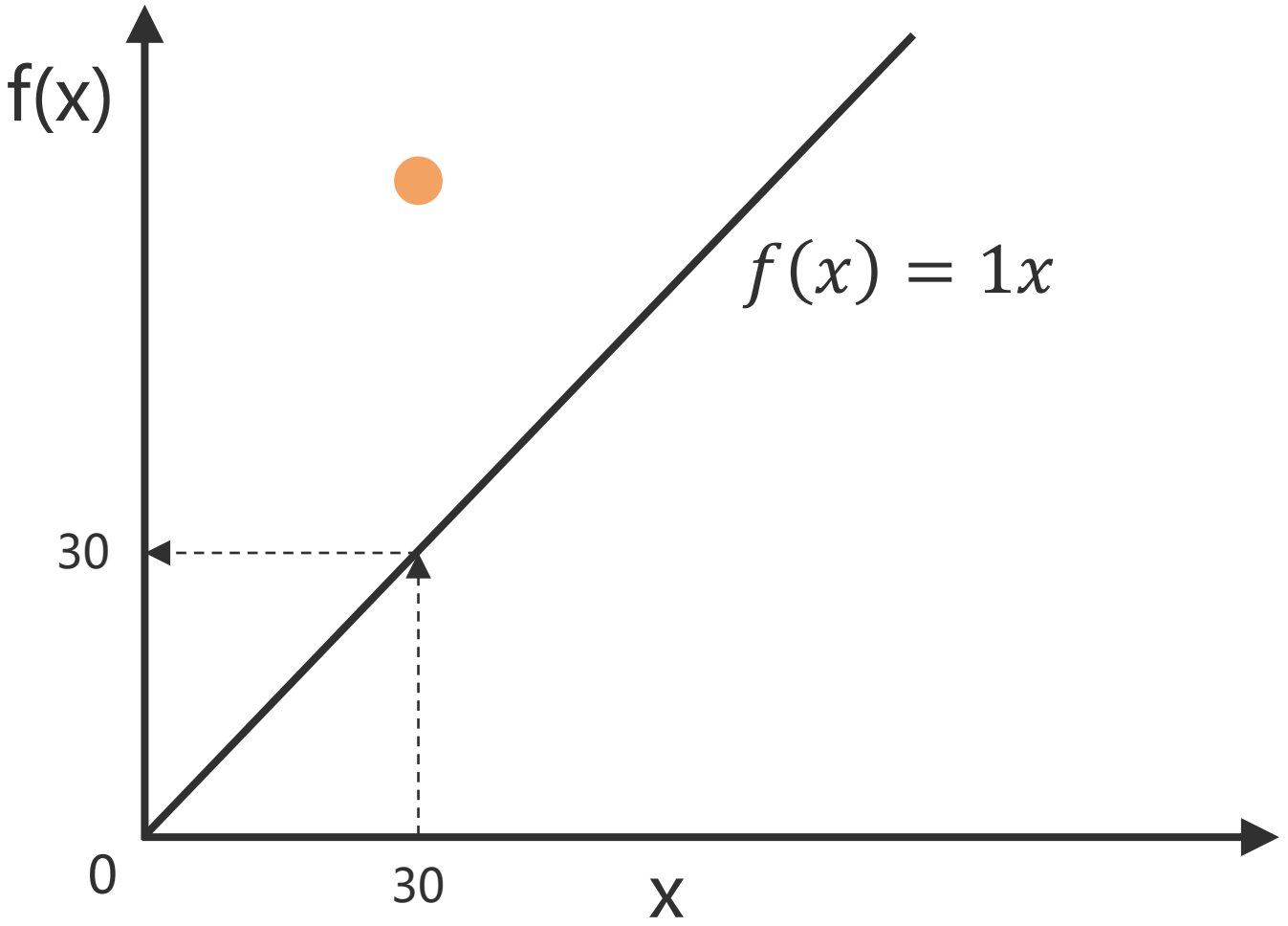
函数

假设某种农作物的产量只被气温影响，我们可以尝试使用简单的线性回归来预测产量。譬如：

$$f(x) = w_0 + w_1 x$$

产量　　气温

实际产量	气温	w_0	w_1	预测值
70	30	0	1	30



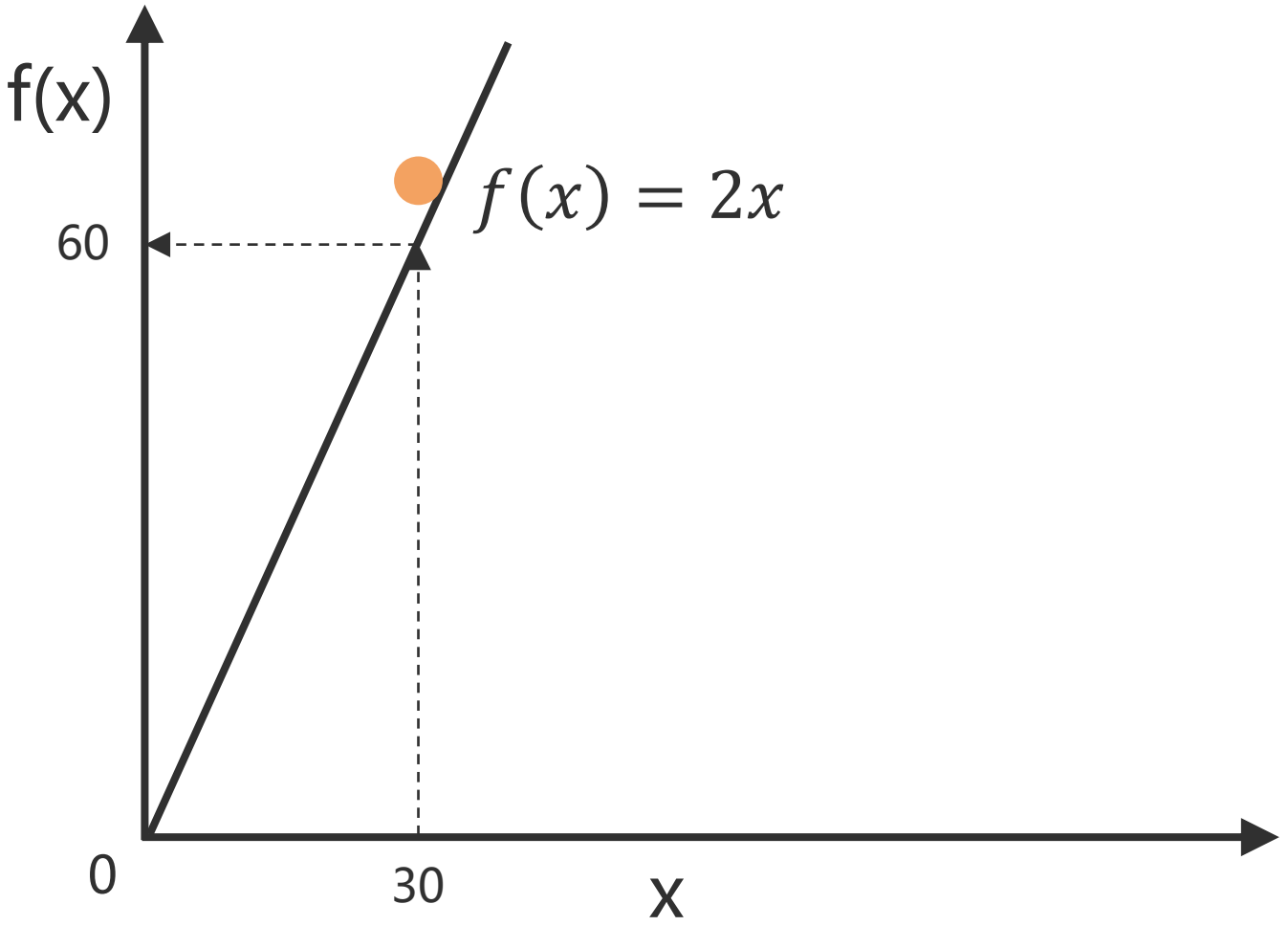
函数

假设某种农作物的产量只被气温影响，我们可以尝试使用简单的线性回归来预测产量。譬如：

$$f(x) = w_0 + w_1 x$$

产量　　气温

实际产量	气温	w_0	w_1	预测值
70	30	0	1	30
70	30	0	2	60



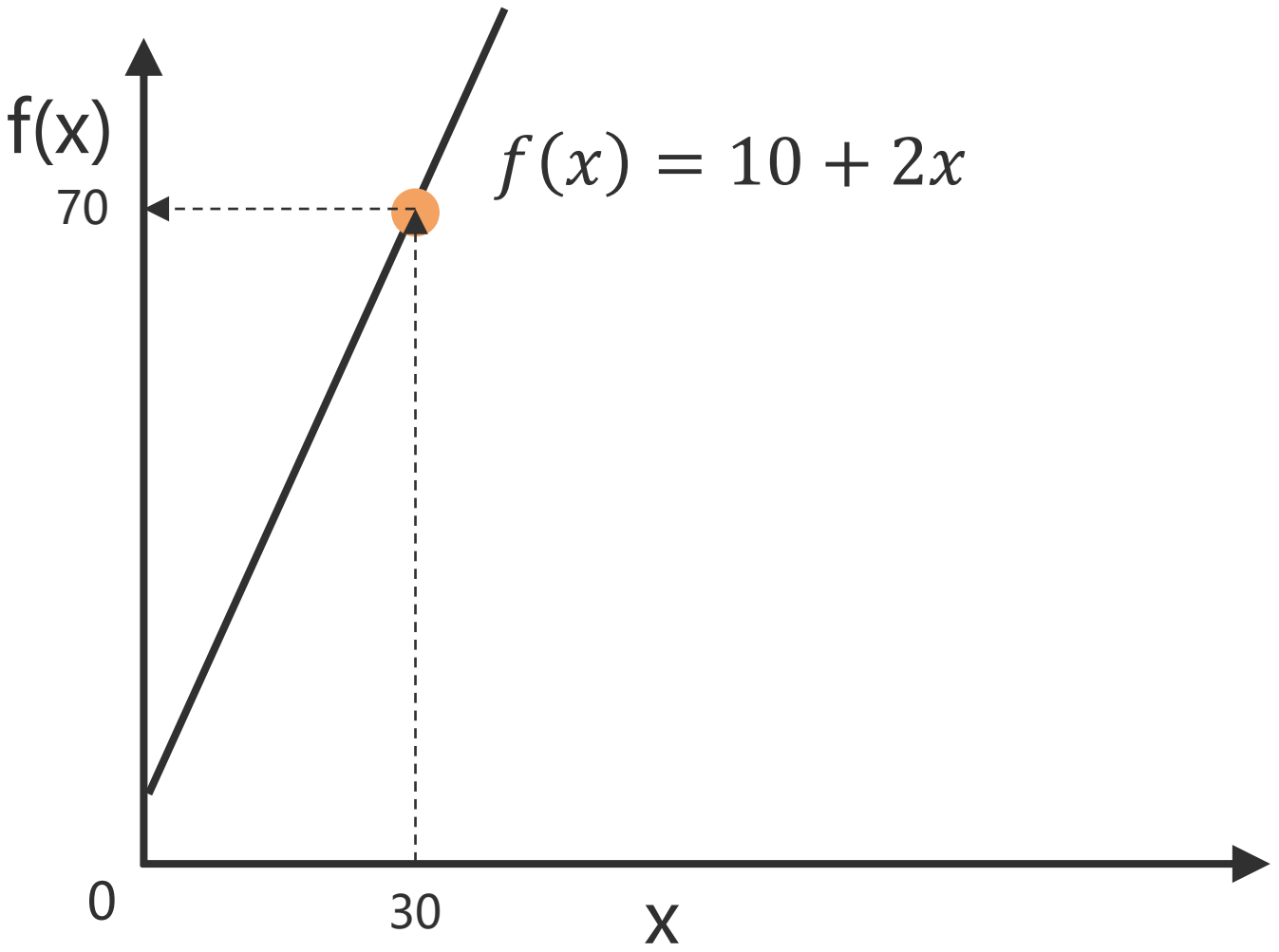
函数

假设某种农作物的产量只被气温影响，我们可以尝试使用简单的线性回归来预测产量。譬如：

$$f(x) = w_0 + w_1 x$$

产量　　气温

实际产量	气温	w_0	w_1	预测值
70	30	0	1	30
70	30	0	2	60
70	30	10	2	70



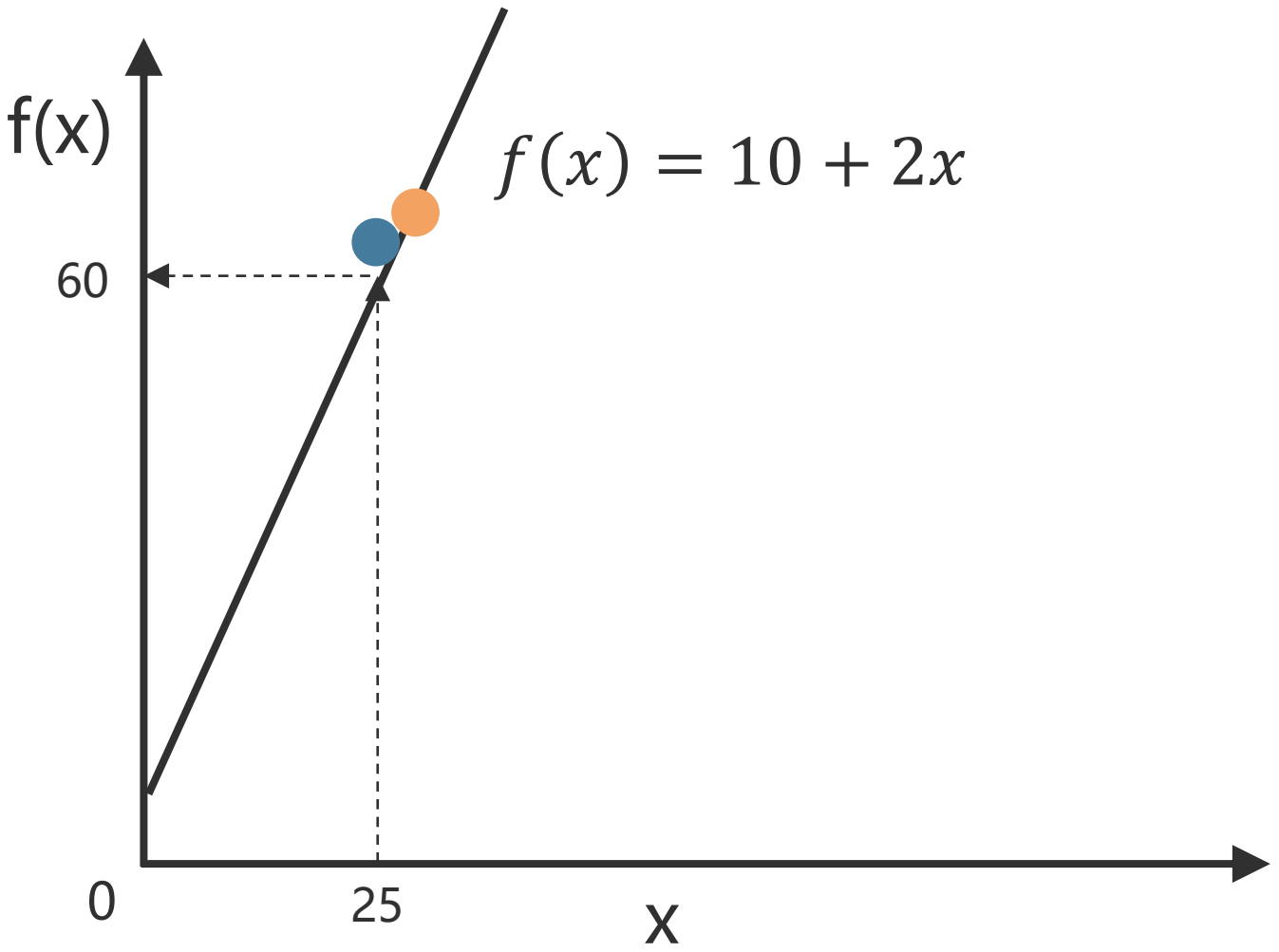
函数

假设某种农作物的产量只被气温影响，我们可以尝试使用简单的线性回归来预测产量。譬如：

$$f(x) = w_0 + w_1 x$$

产量　　气温

实际产量	气温	w_0	w_1	预测值
70	30	0	1	30
70	30	0	2	60
70	30	10	2	70
65	25	10	2	60

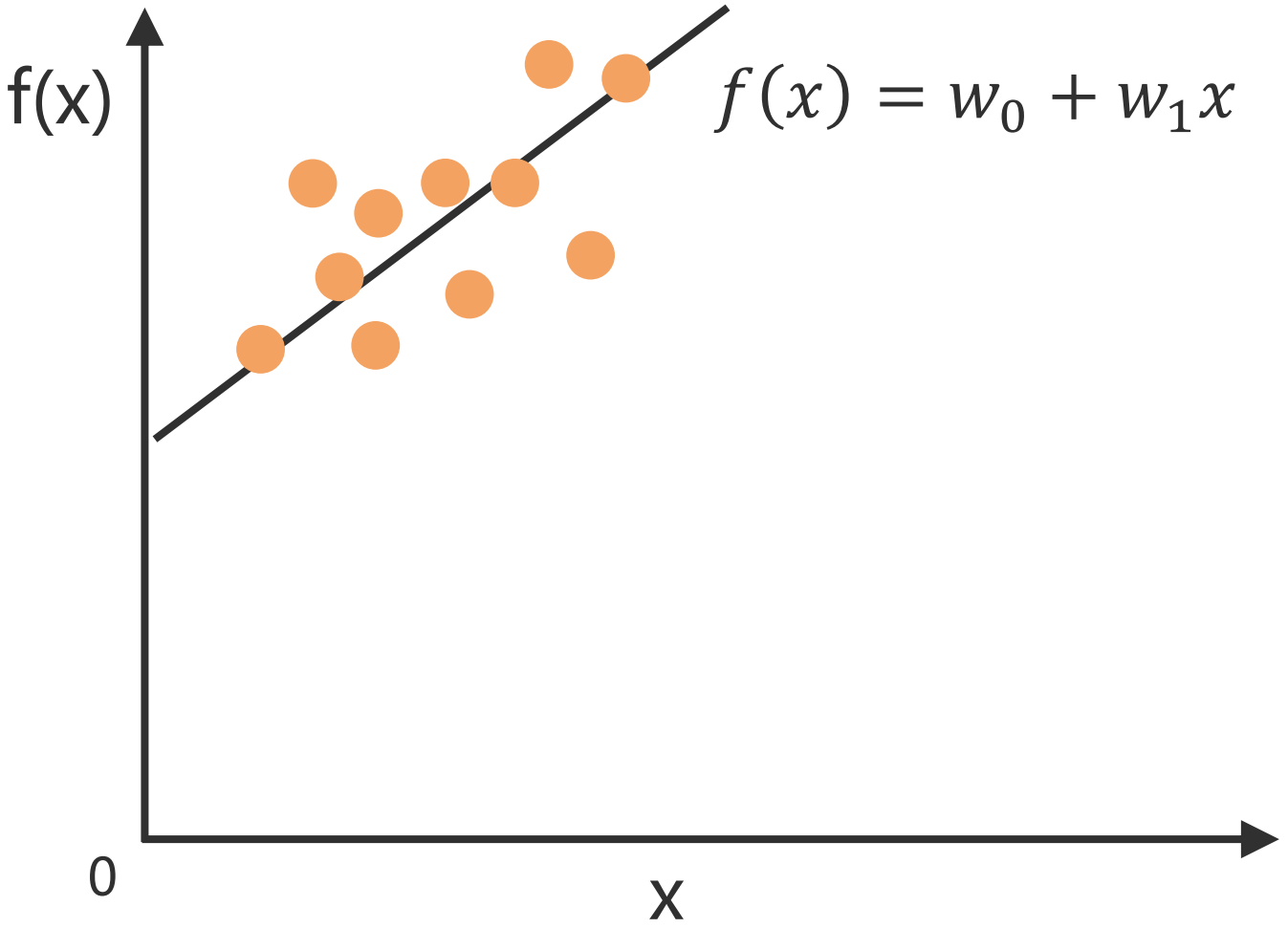


函数

假设某种农作物的产量只被气温影响，我们可以尝试使用简单的线性回归来预测产量。譬如：

$$f(x) = w_0 + w_1 x$$

产量　　气温



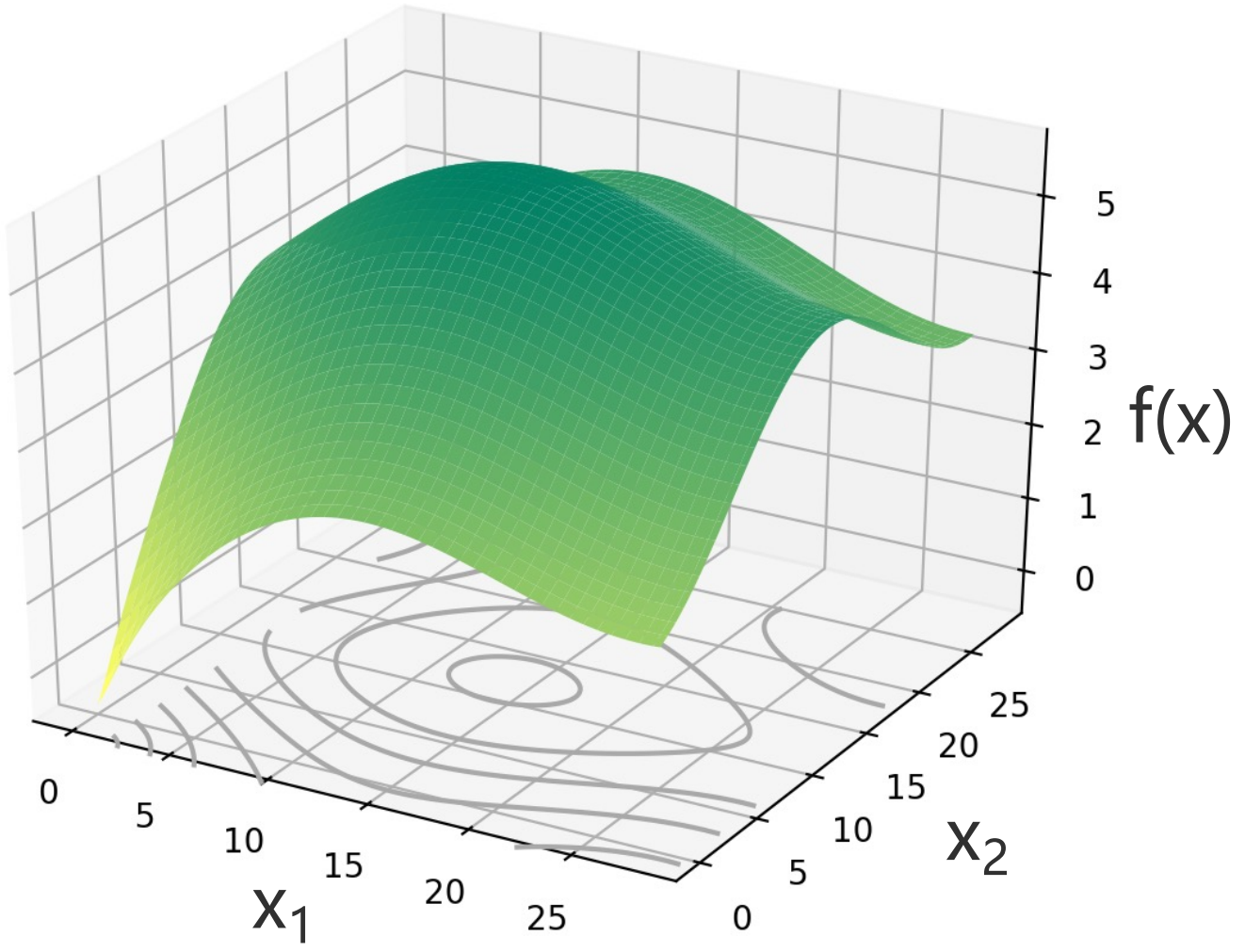
- 最小二乘法
- 梯度下降法

函数

假设某种农作物的产量被气温以及施肥量影响，
我们还可以尝试用多变量的线性函数来预测产量。
譬如：

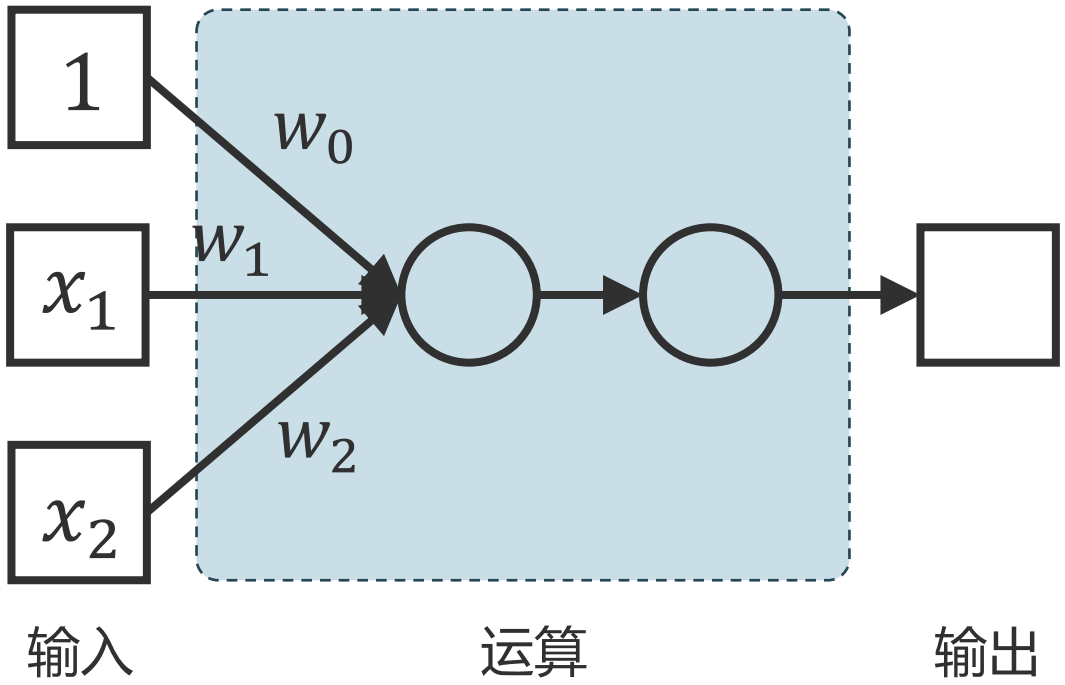
$$f(x) = w_0 + w_1 x_1 + w_2 x_2$$

▲ 产量　▲ 气温　▲ 施肥量



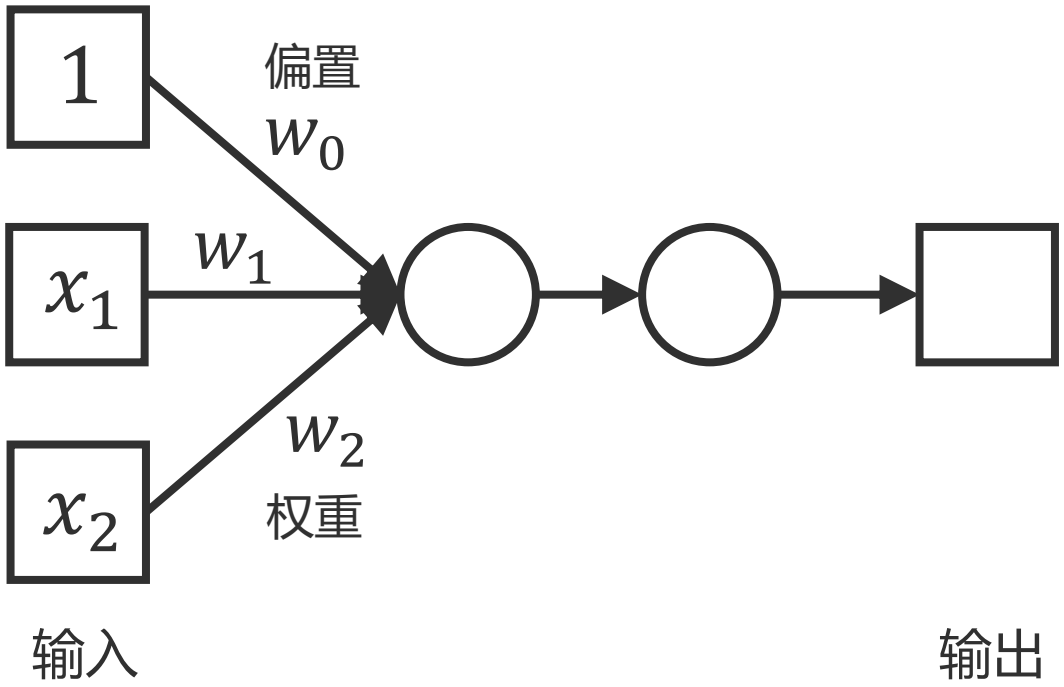
神经网络

神经网络是机器学习算法之一，它不仅可以像线性函数一样用来做回归分析，还可以应用在分类问题上。早在 20 世纪 50 年代就已经出现神经网络的基本概念。它的运算方式与线性回归相似，即接受多个输入，然后乘上权重，最后输出一个数值。



神经网络

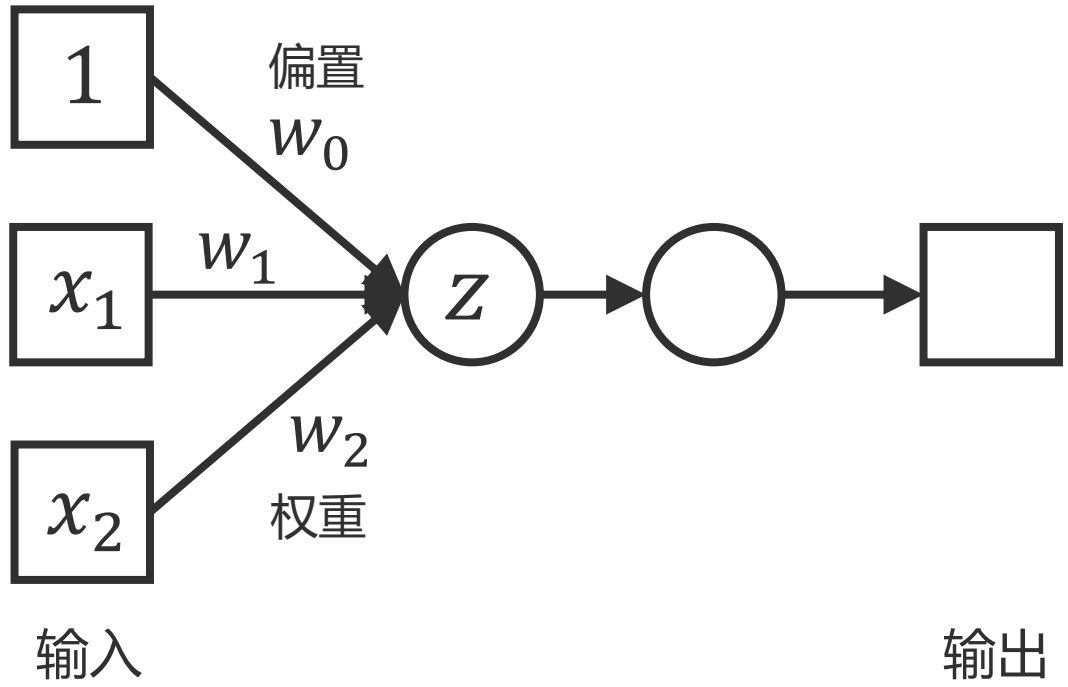
输入 x_1 x_2



神经网络

输入　x_1　　x_2

处理　$z = w_0 + w_1 x_1 + w_2 x_2$

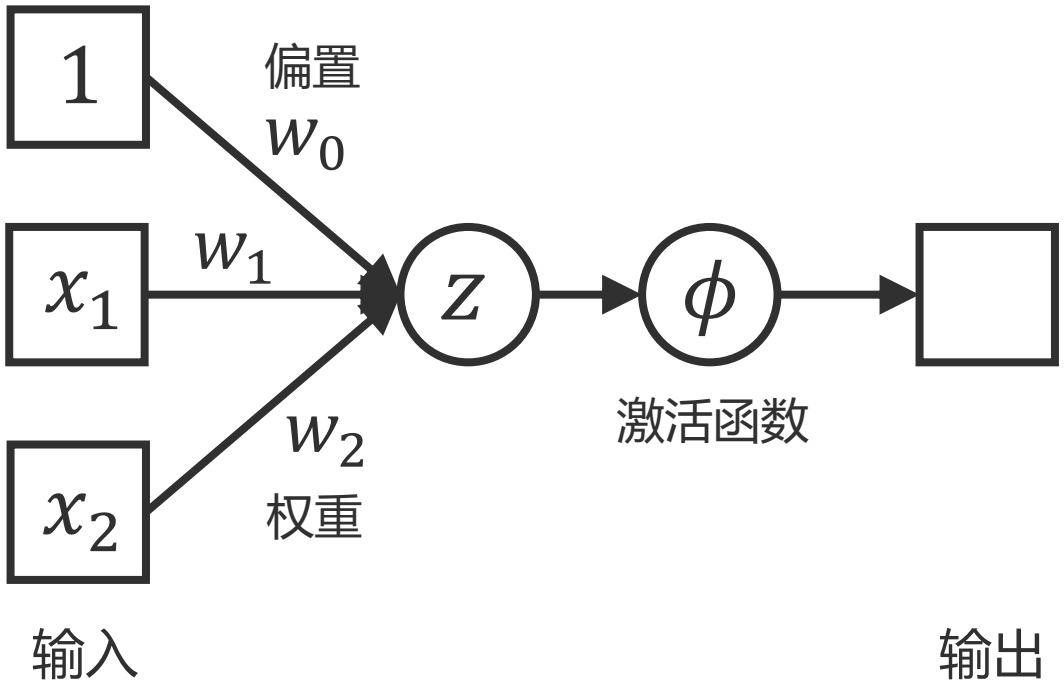


神经网络

输入　x_1　x_2

处理　$z = w_0 + w_1 x_1 + w_2 x_2$

处理　$\phi(z) = z$



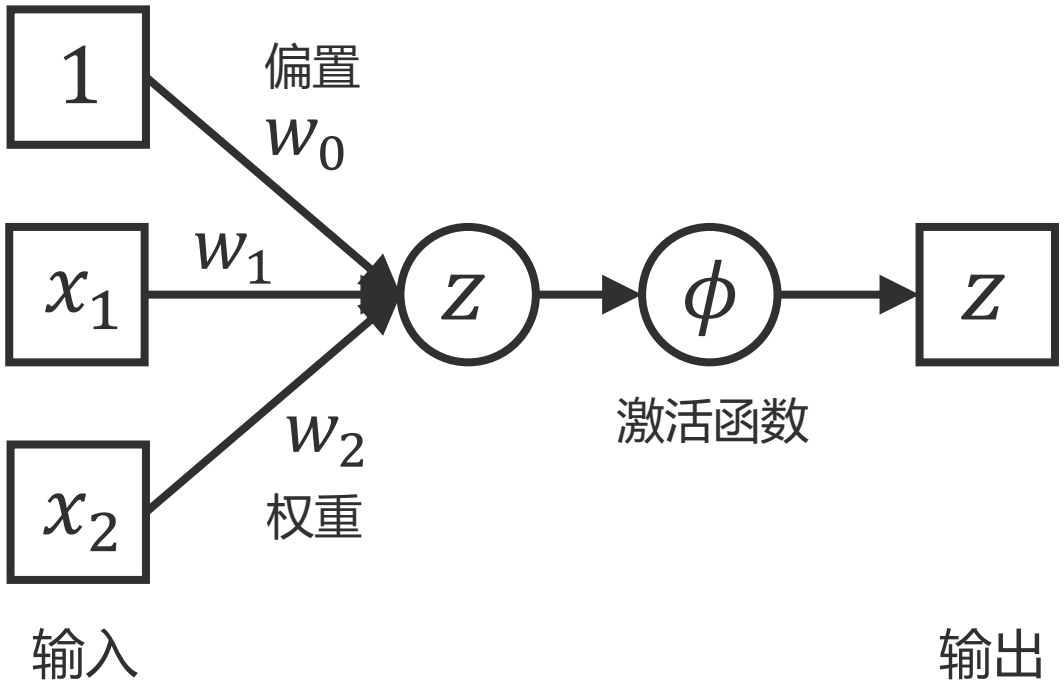
神经网络

输入　x_1　x_2

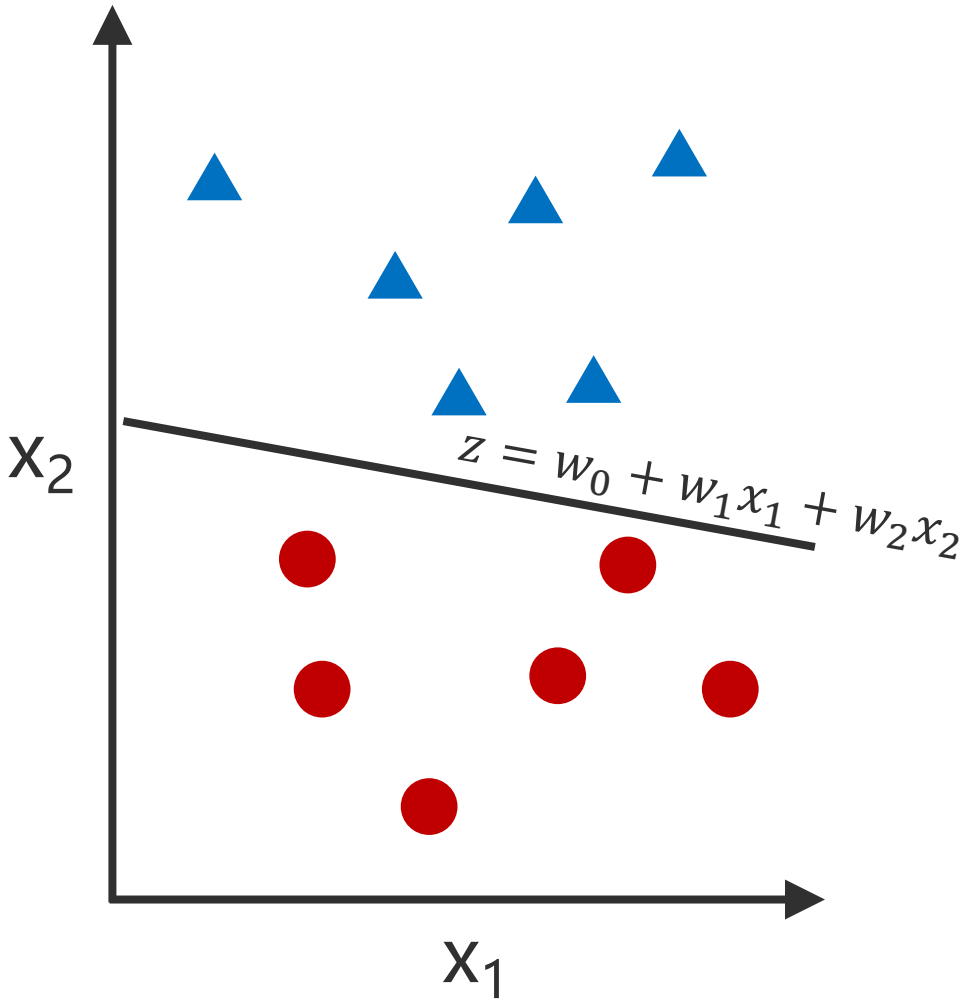
处理　$z = w_0 + w_1 x_1 + w_2 x_2$

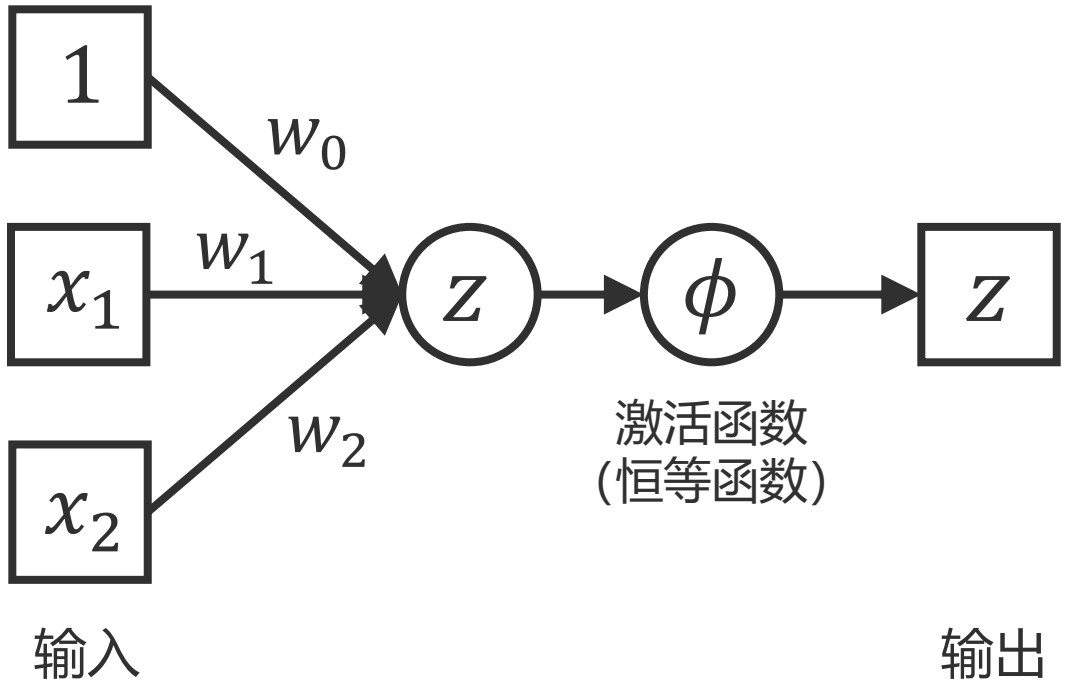
处理　$\phi(z) = z$

输出　z

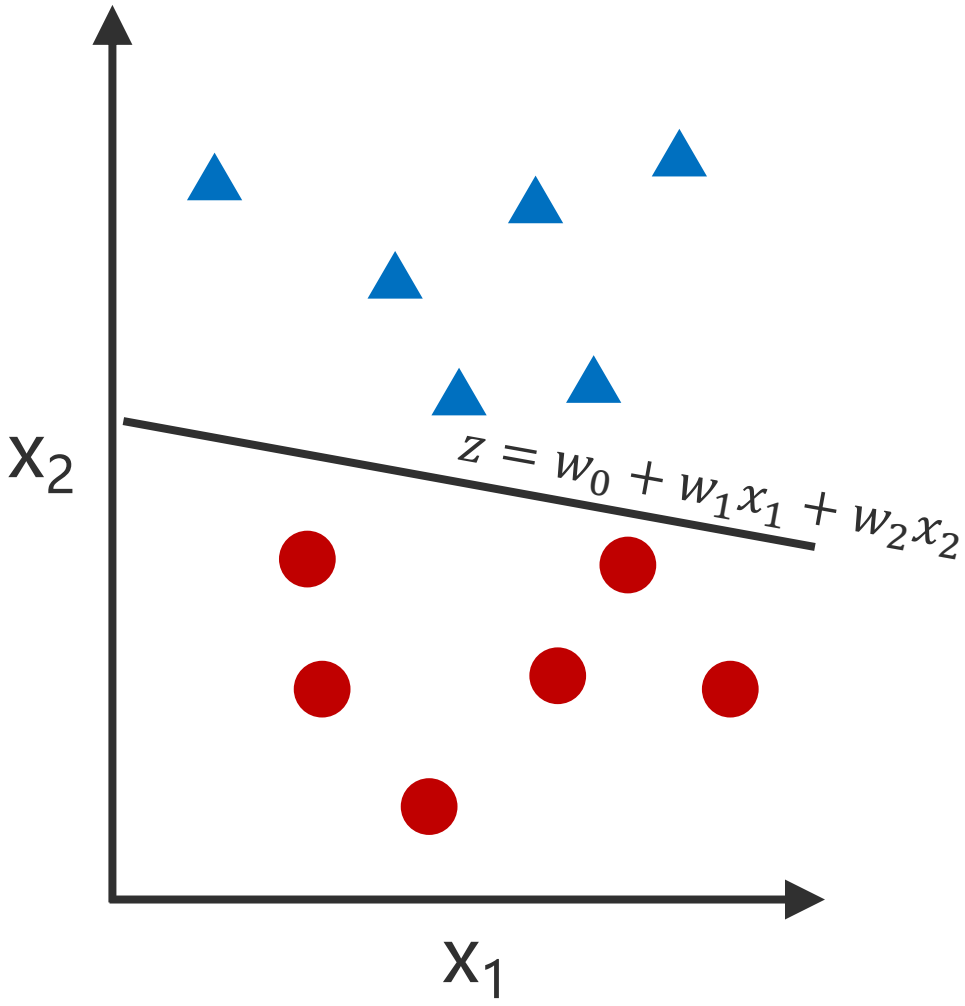


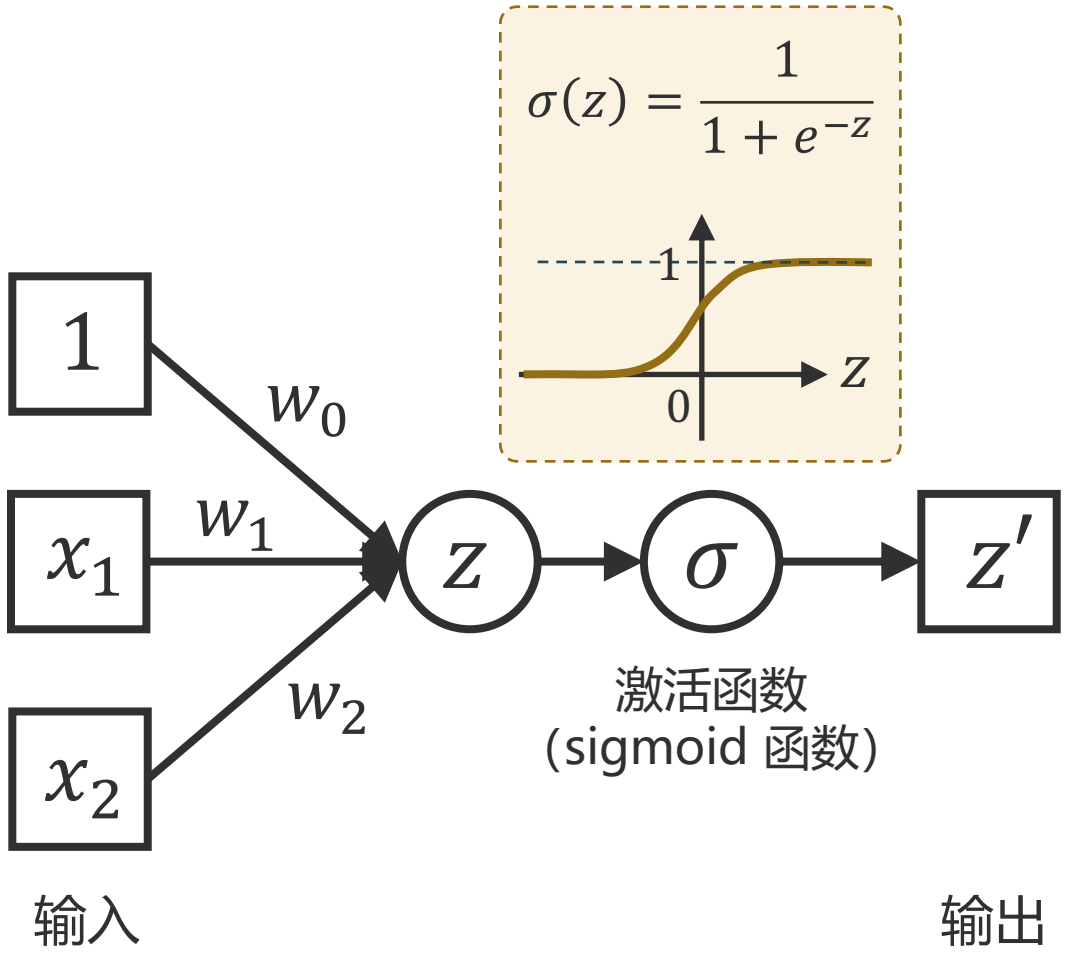
线性可分



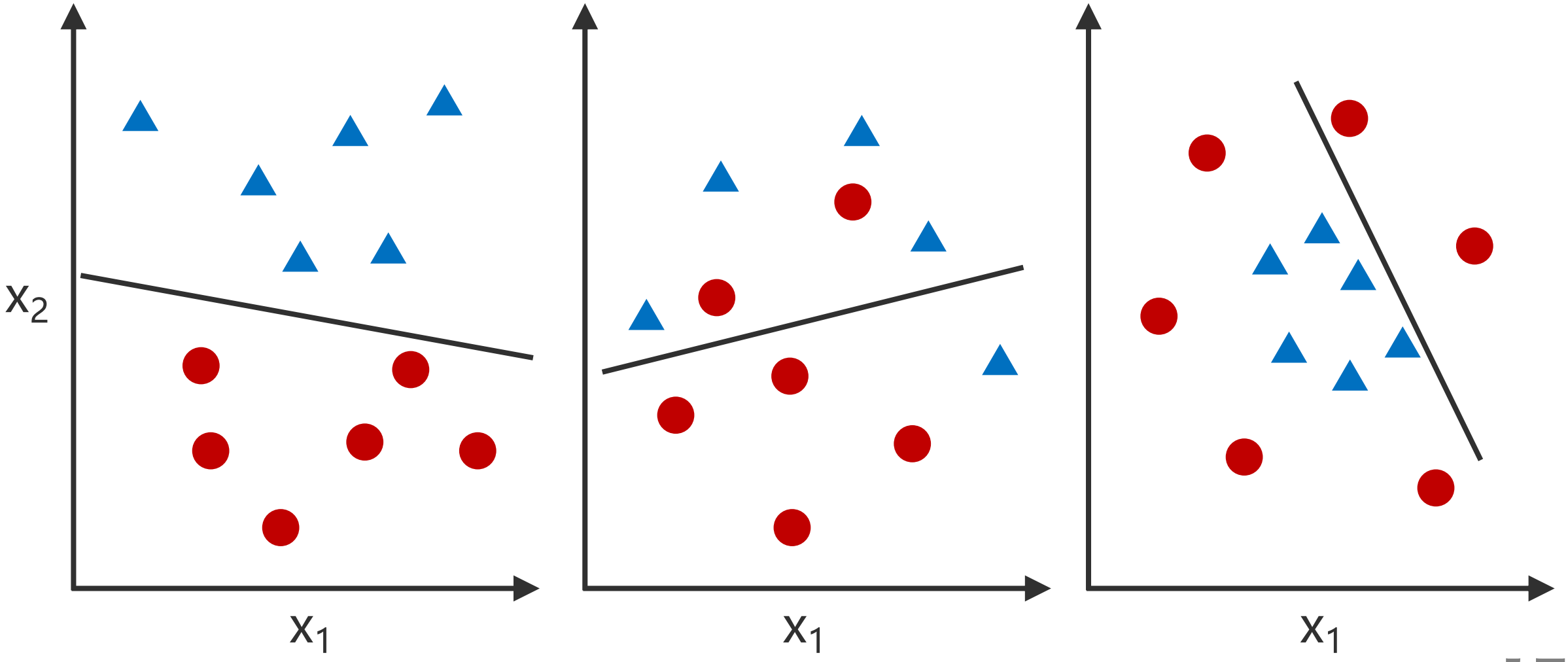


线性可分

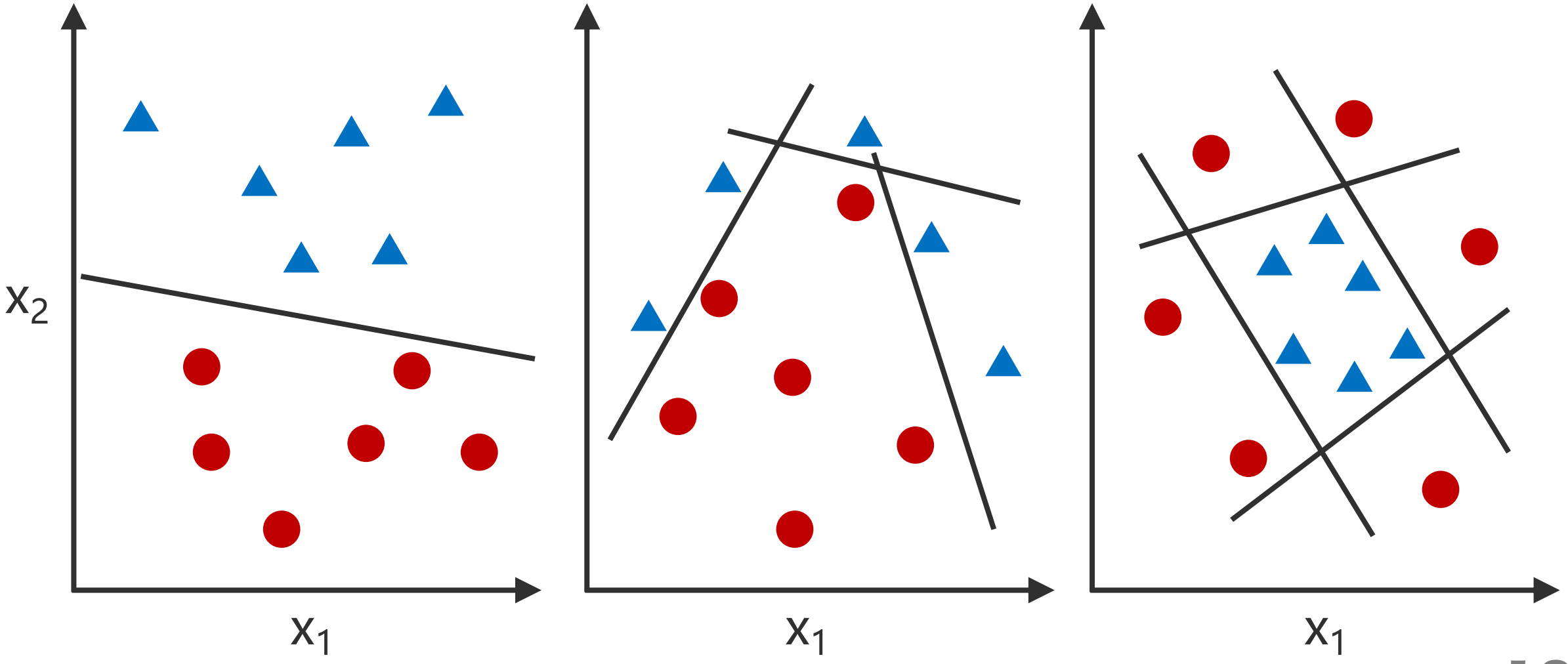

x2
$z = w_0 + w_1 x_1 + w_2 x_2$
x1


$\sigma(z) = \frac{1}{1+e^{-z}}$
1
z
0
1
w_0
x_1
w_1
z
σ
z'
w_2
x_2
激活函数
(sigmoid 函数)
输入
输出

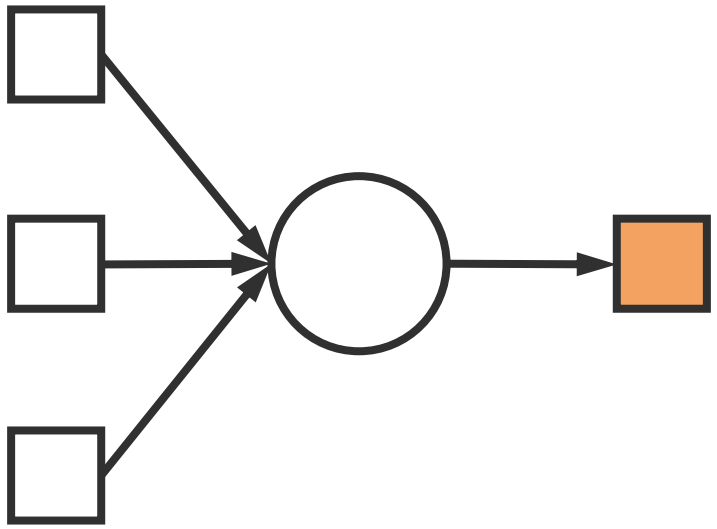
线性可分



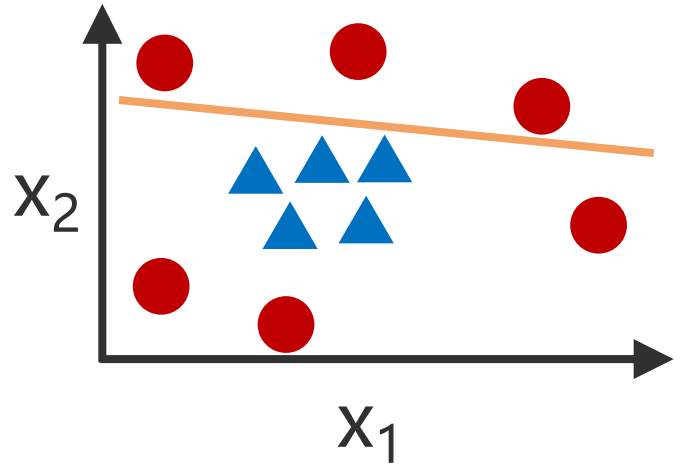
线性可分



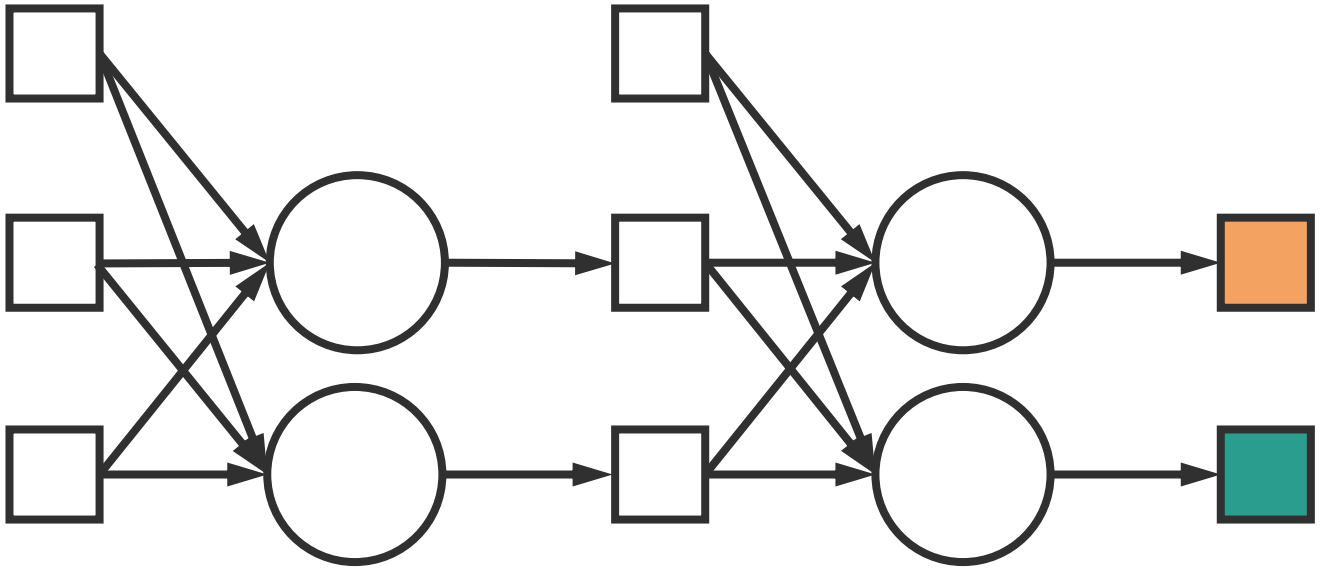
神经网络



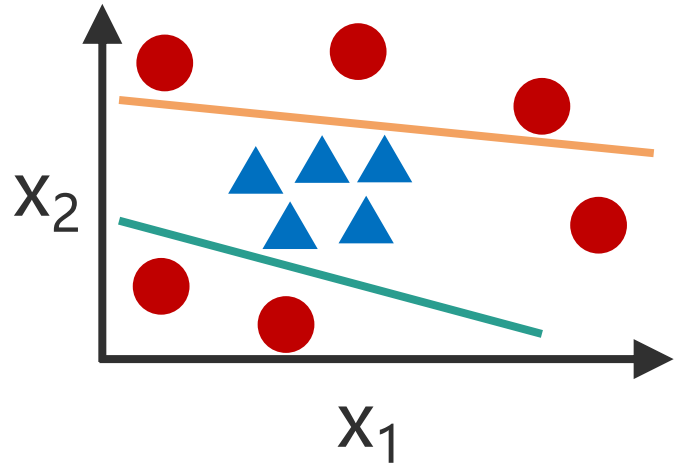
图片仅供参考，以实物为准



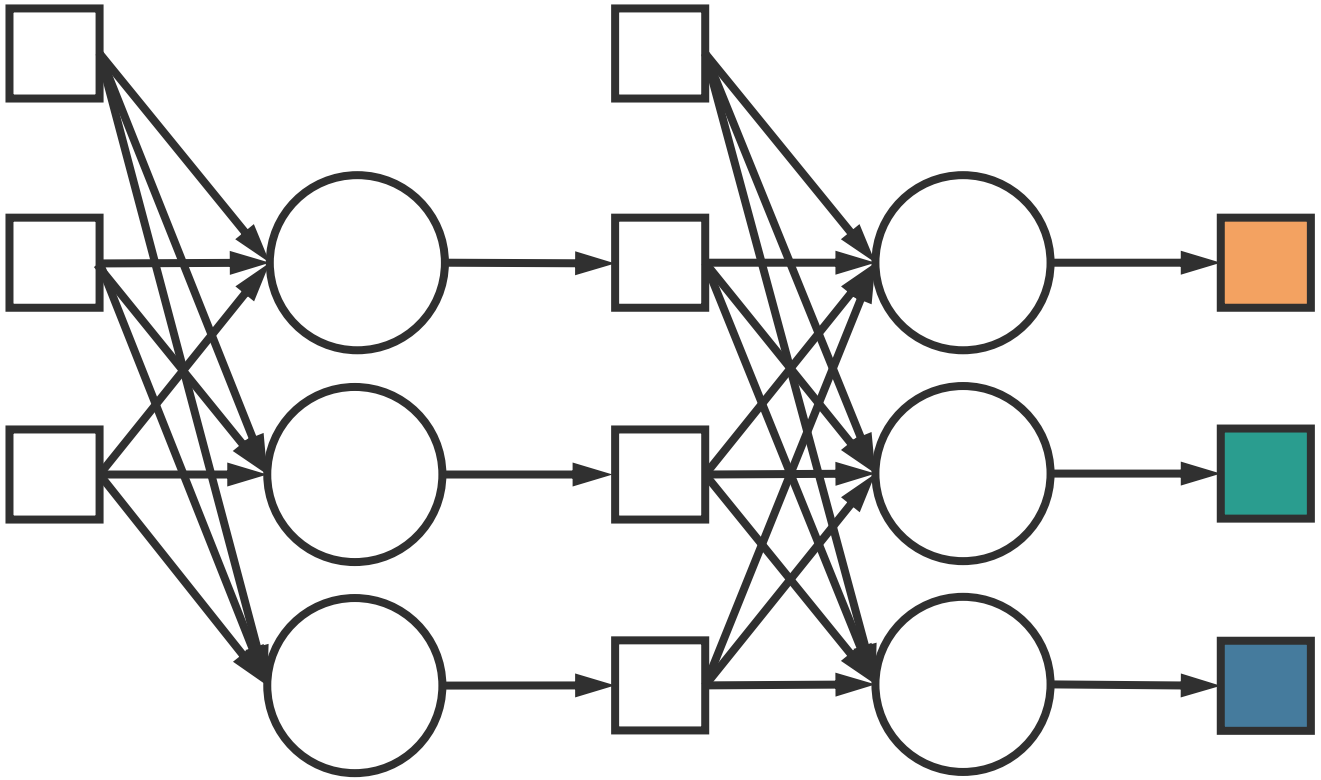
神经网络



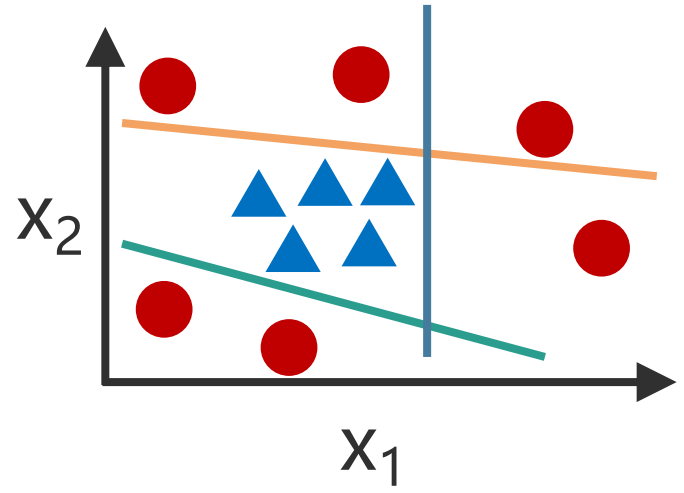
图片仅供参考，以实物为准



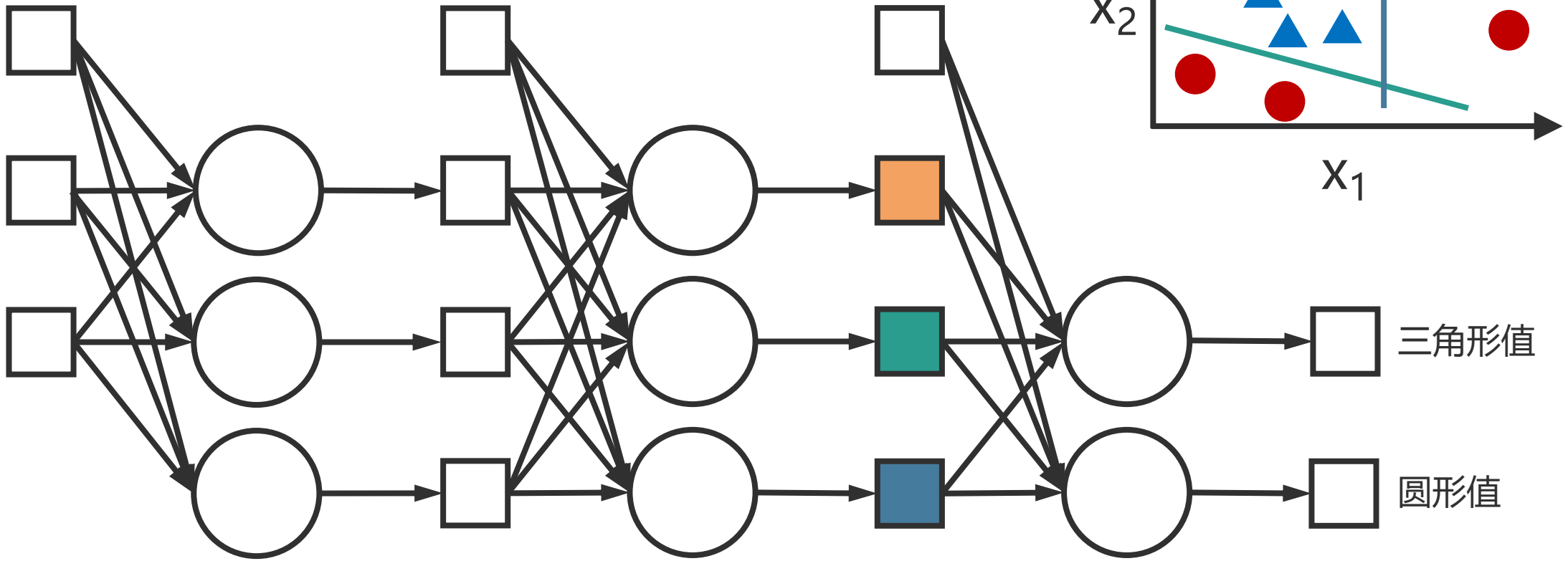
神经网络



图片仅供参考，以实物为准

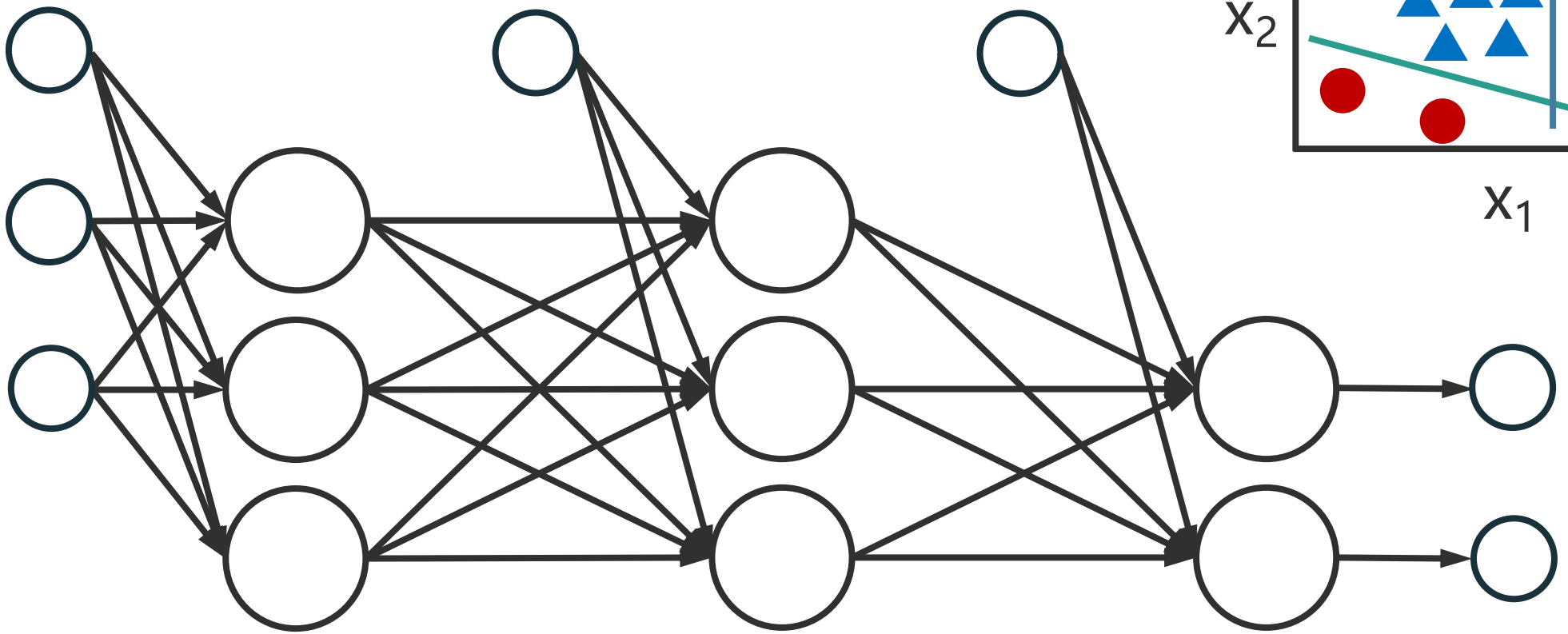


神经网络

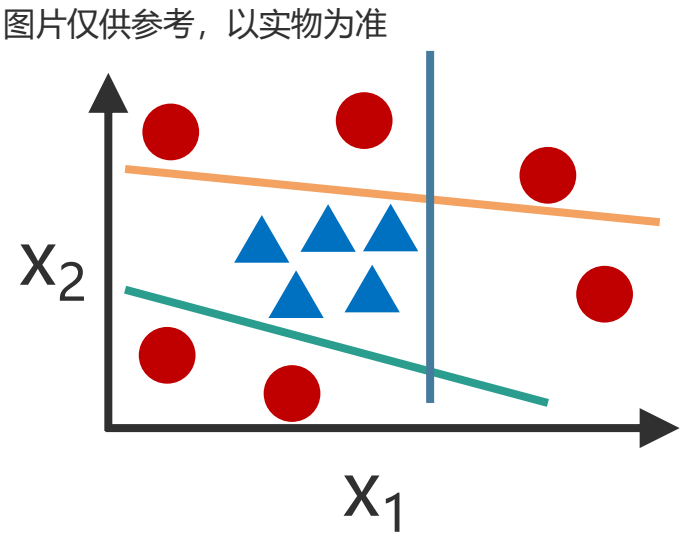


图片仅供参考，以实物为准

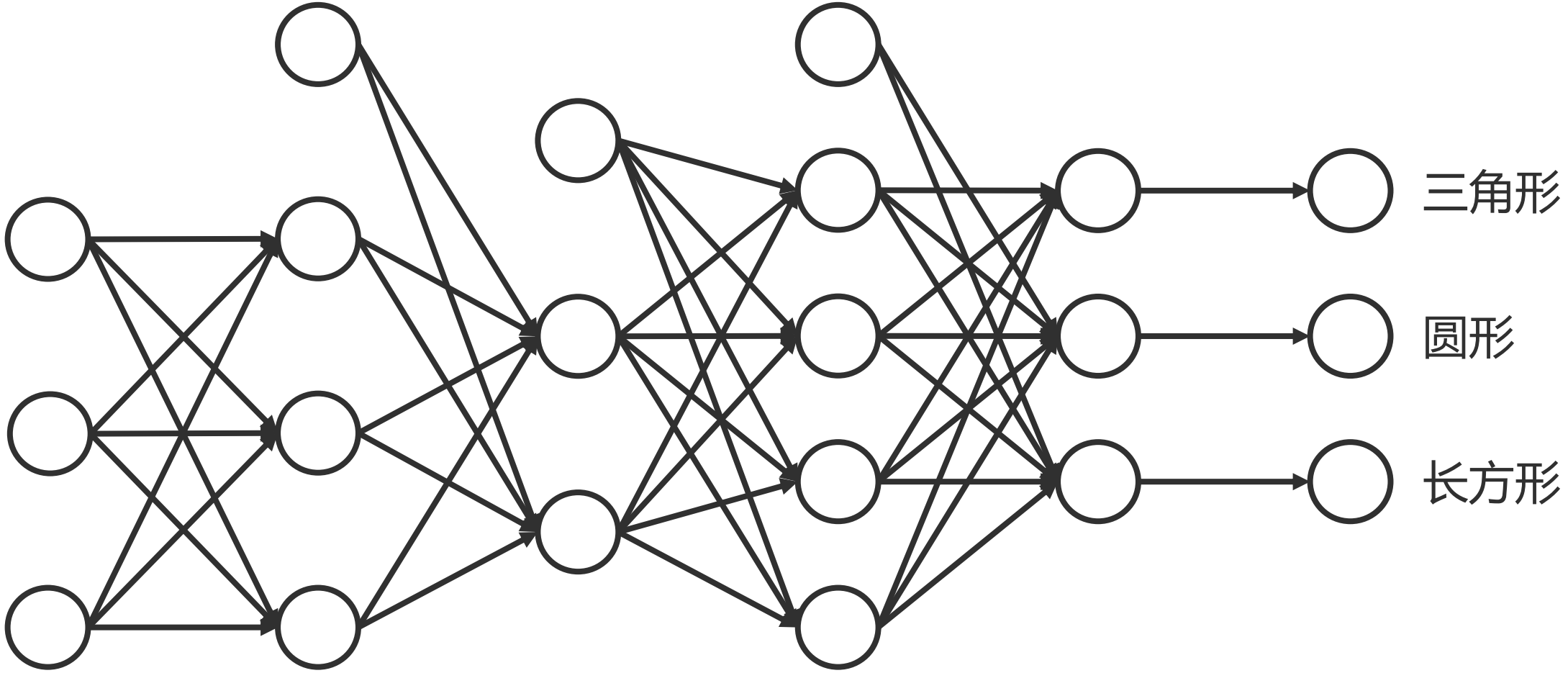
神经网络



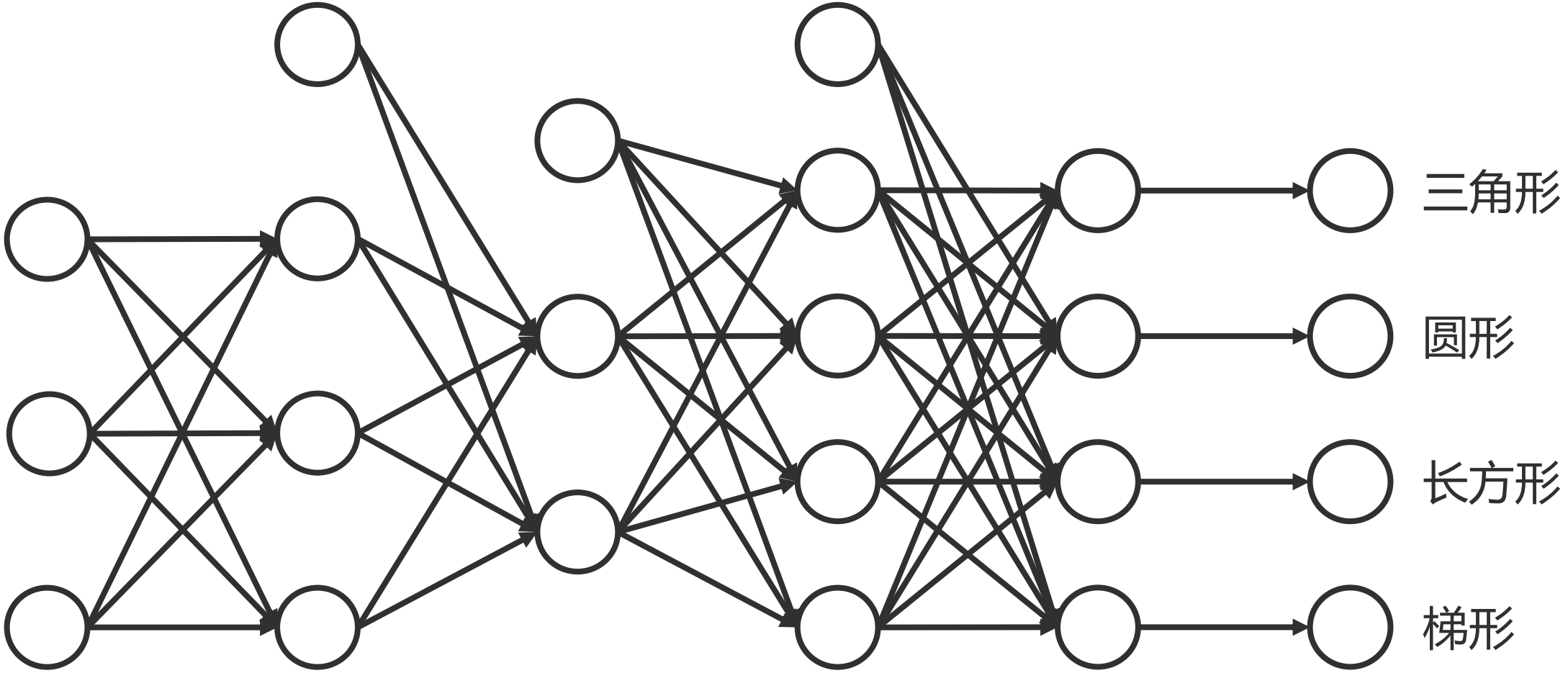
图片仅供参考，以实物为准



神经网络



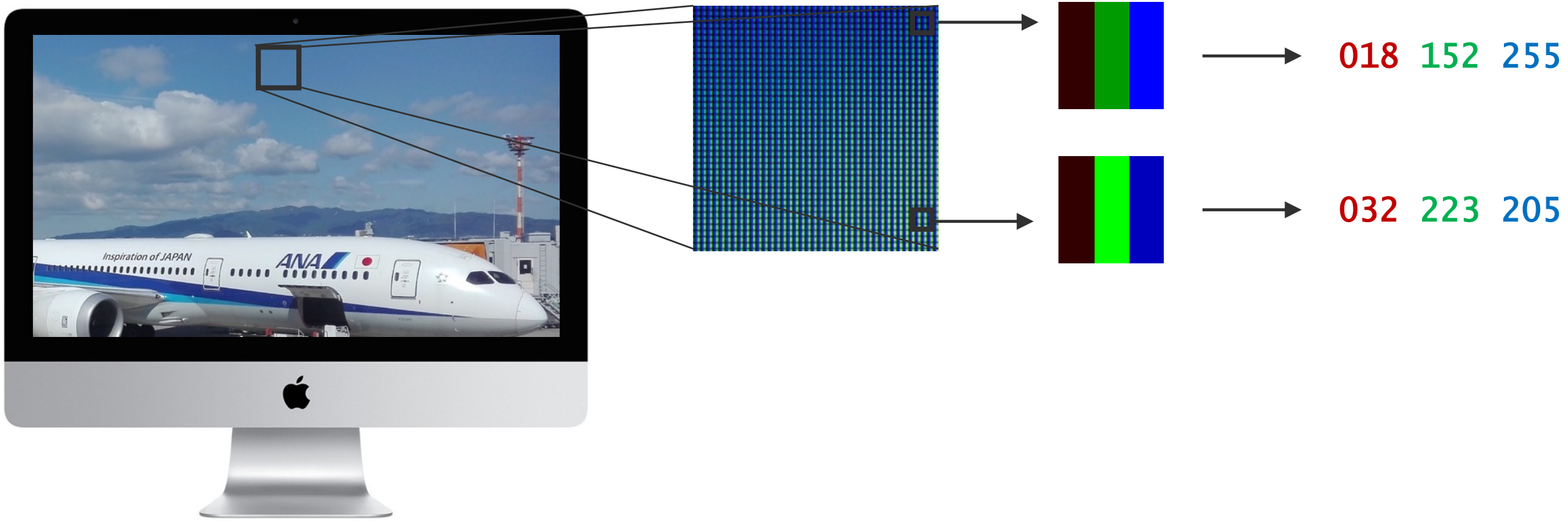
神经网络



物体分类

- 神经网络
- 
  卷积神经网络
- torch

物体识别



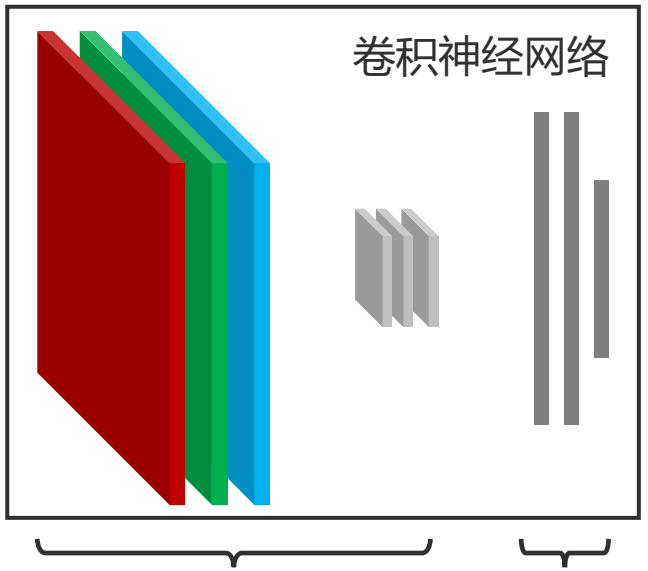
物体识别

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

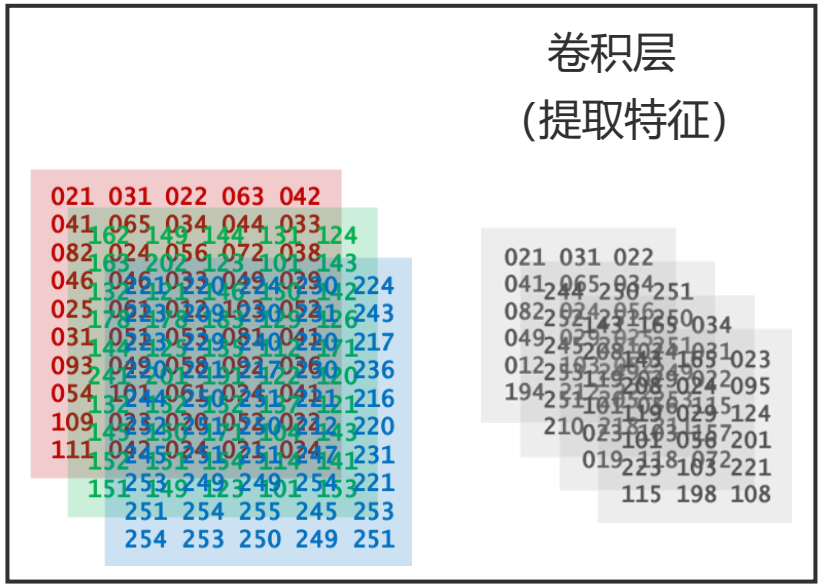
162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

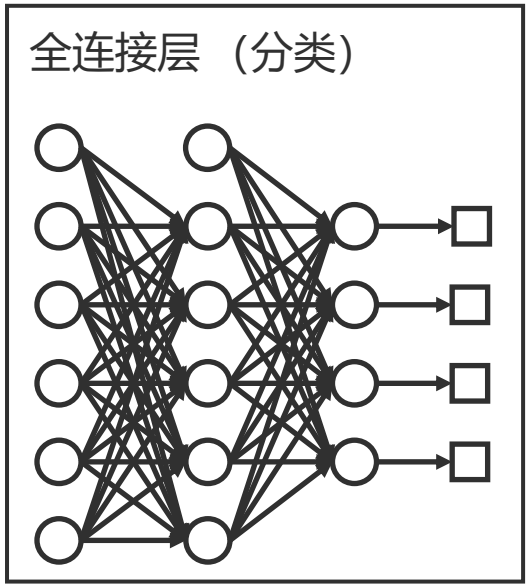
221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

卷积神经网络




```
airplane 56.1%
sky      25.0%
cloud    12.4%
mountain  6.5%
```





黑白图像

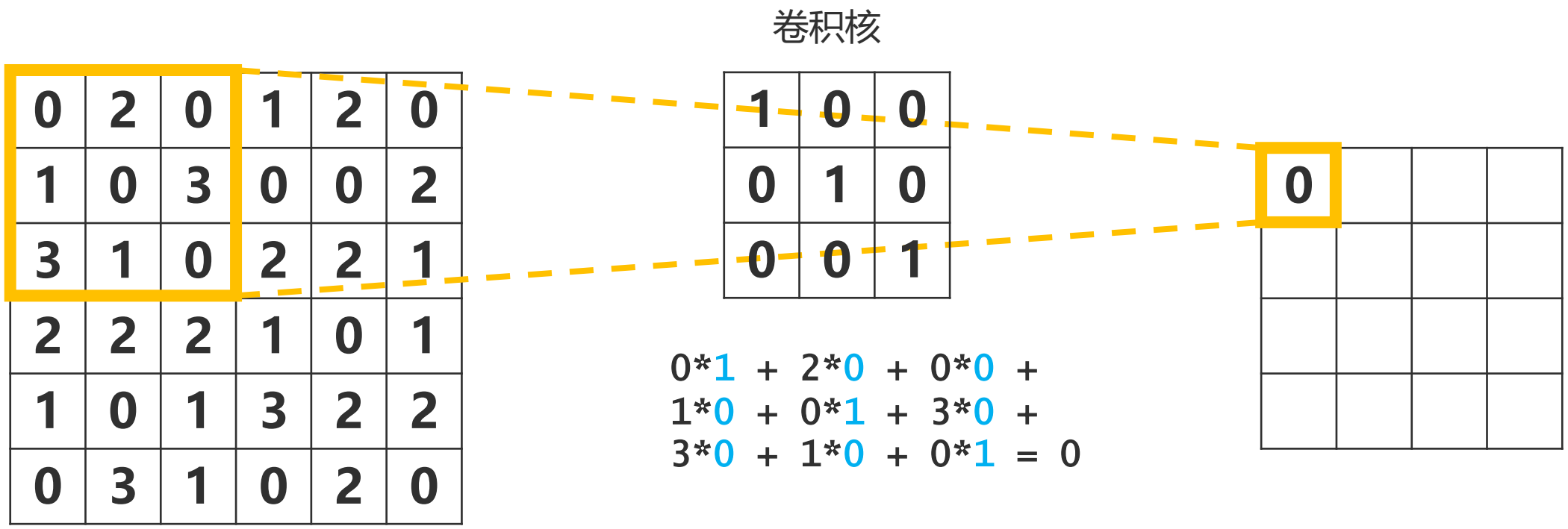
162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

单通道（1 channel）

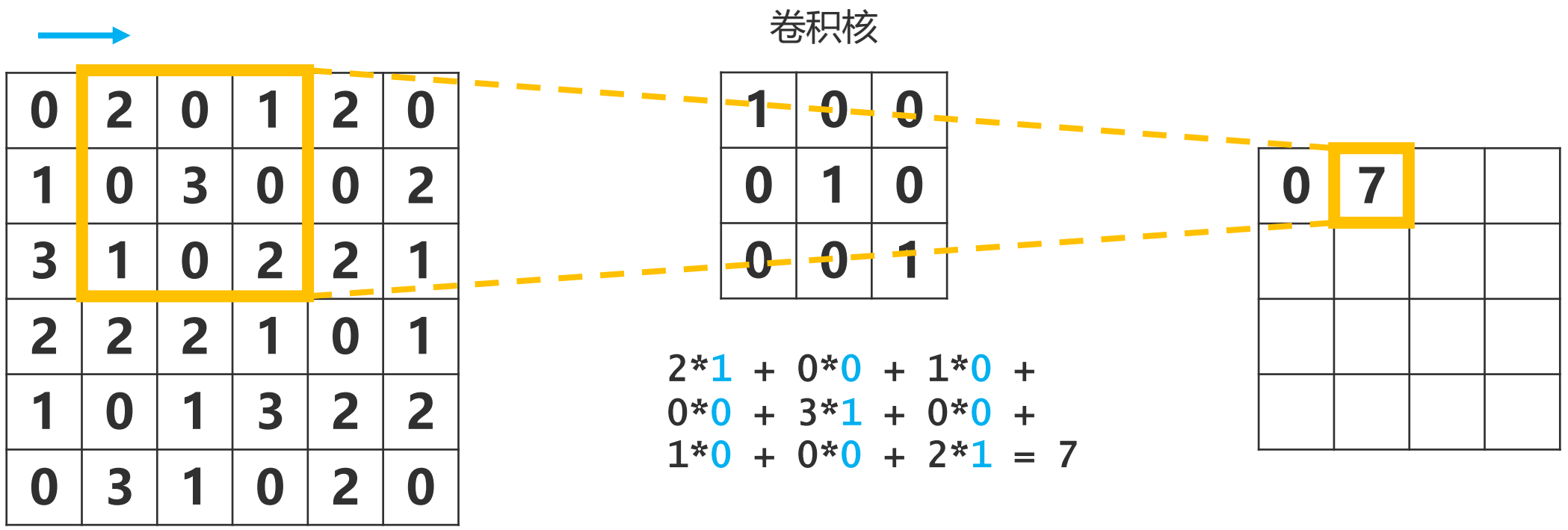
彩色图像

三通道（3 channels）

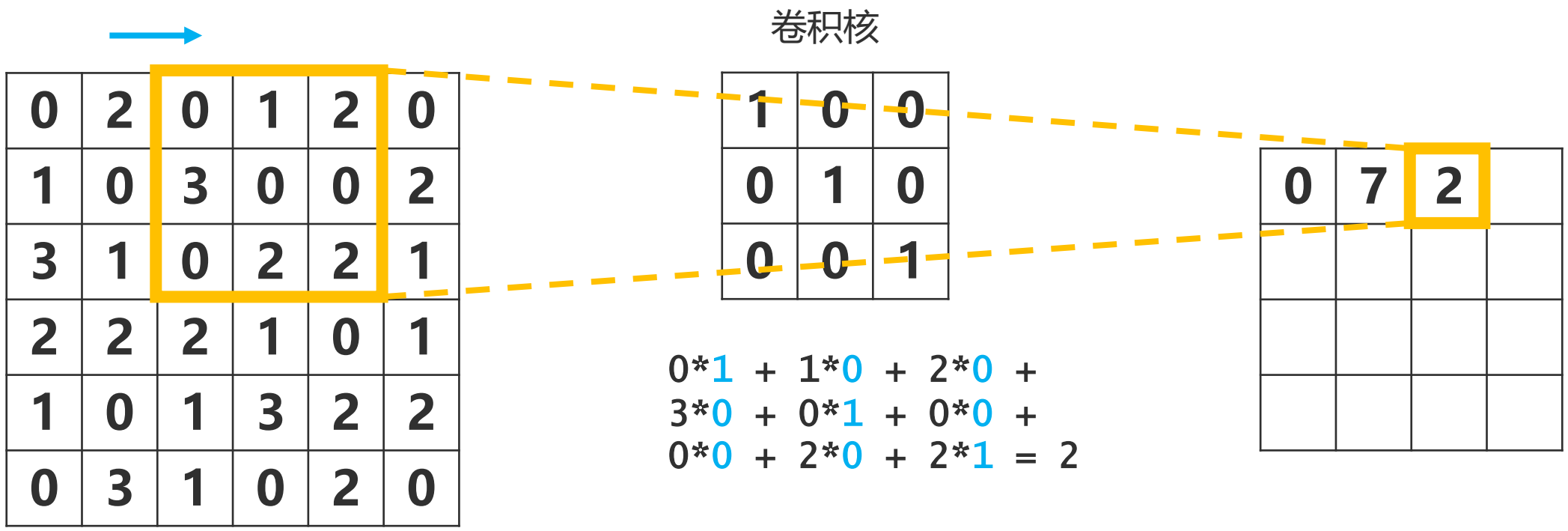
卷积计算


卷积核
0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0*1 + 2*0 + 0*0 +
1*0 + 0*1 + 3*0 +
3*0 + 1*0 + 0*1 = 0
0

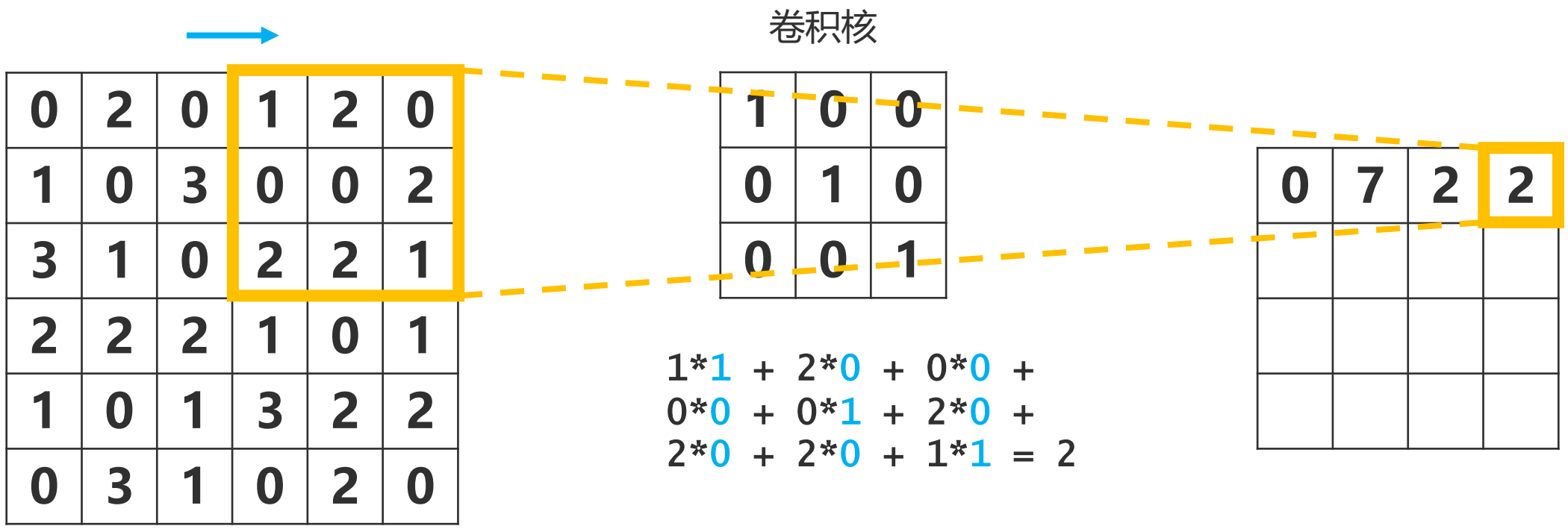
卷积计算


卷积核
0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
2*1 + 0*0 + 1*0 +
0*0 + 3*1 + 0*0 +
1*0 + 0*0 + 2*1 = 7
0 7

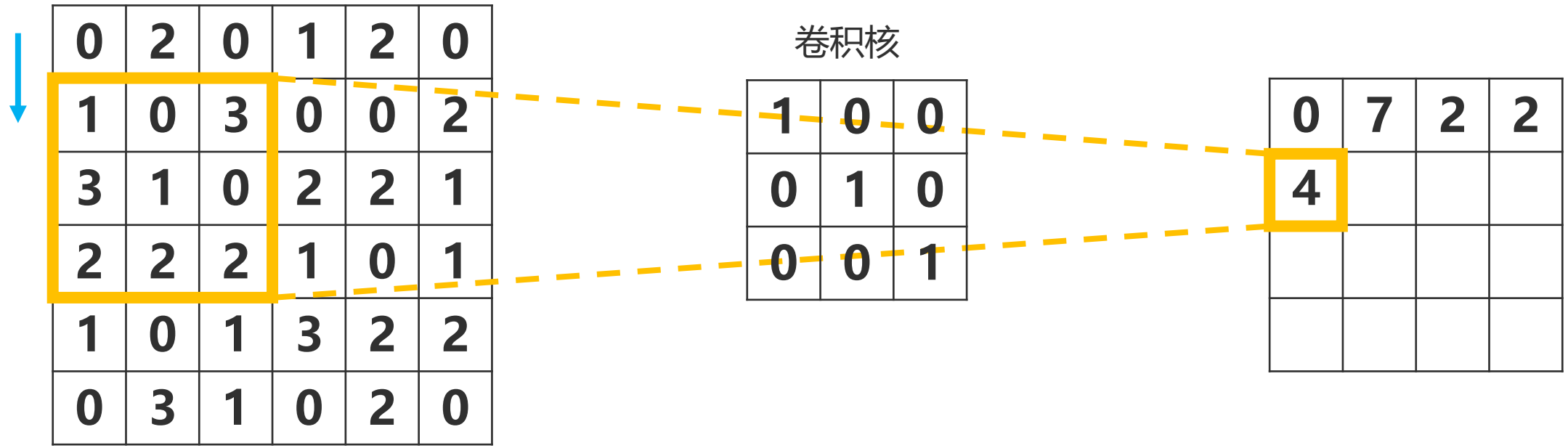
卷积计算


卷积核
0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0*1 + 1*0 + 2*0 +
3*0 + 0*1 + 0*0 +
0*0 + 2*0 + 2*1 = 2
0 7 2

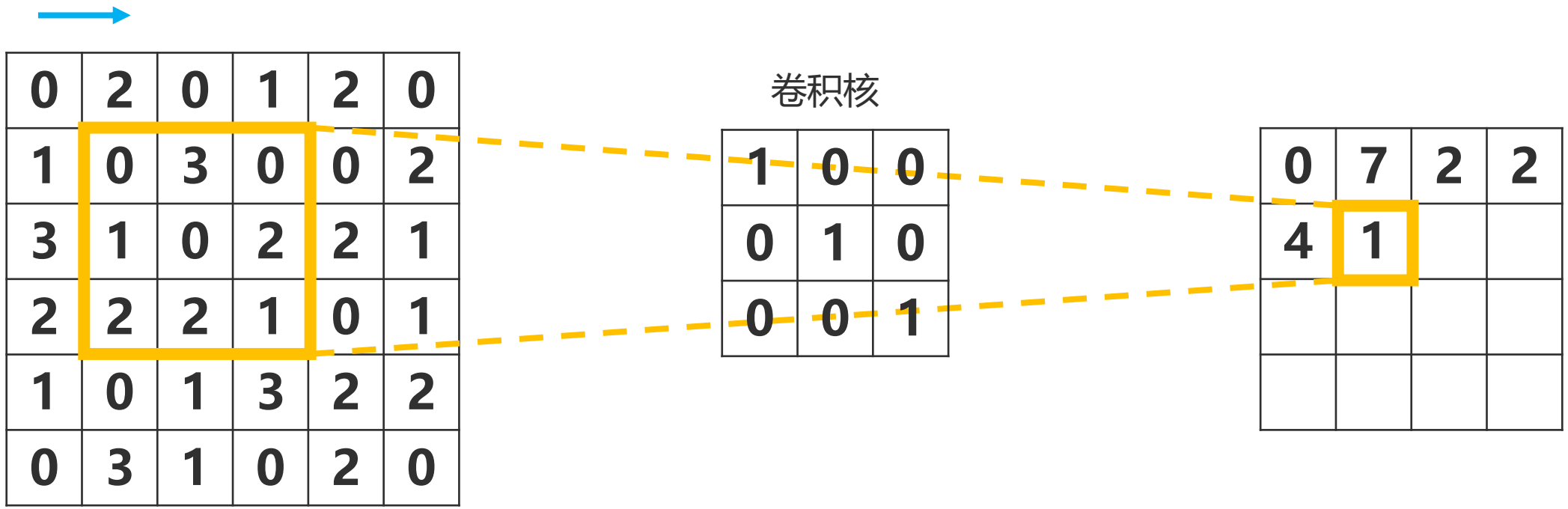
卷积计算


卷积核
0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
1*1 + 2*0 + 0*0 +
0*0 + 0*1 + 2*0 +
2*0 + 2*0 + 1*1 = 2
0 7 2 2

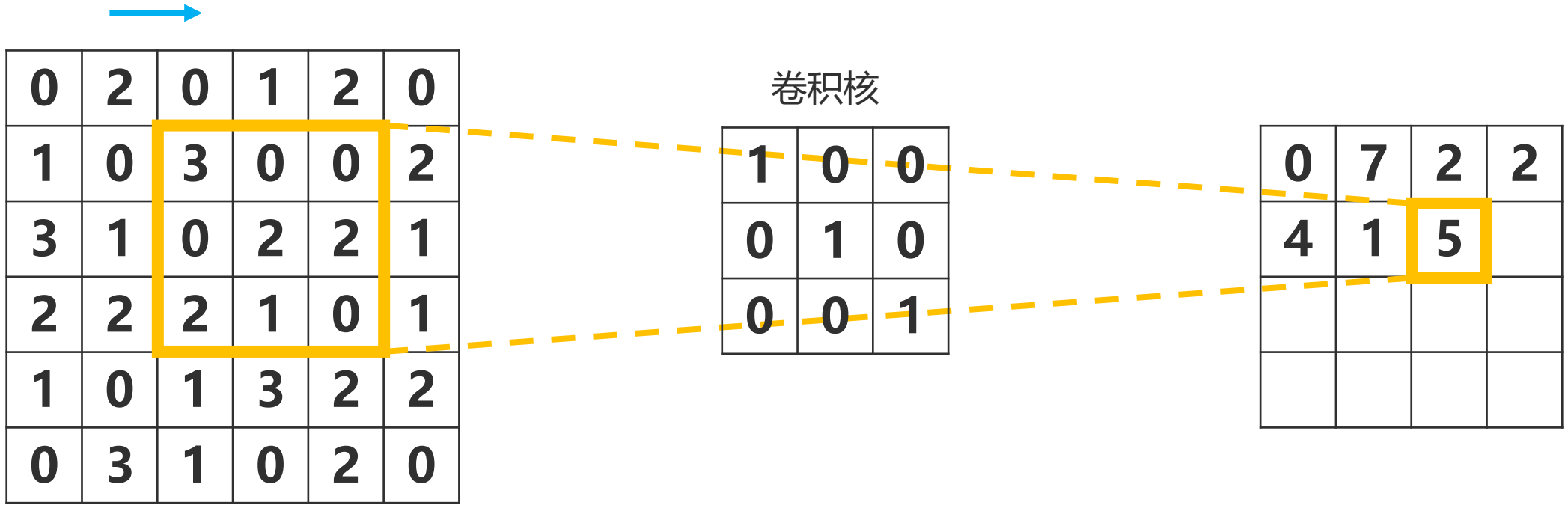
卷积计算



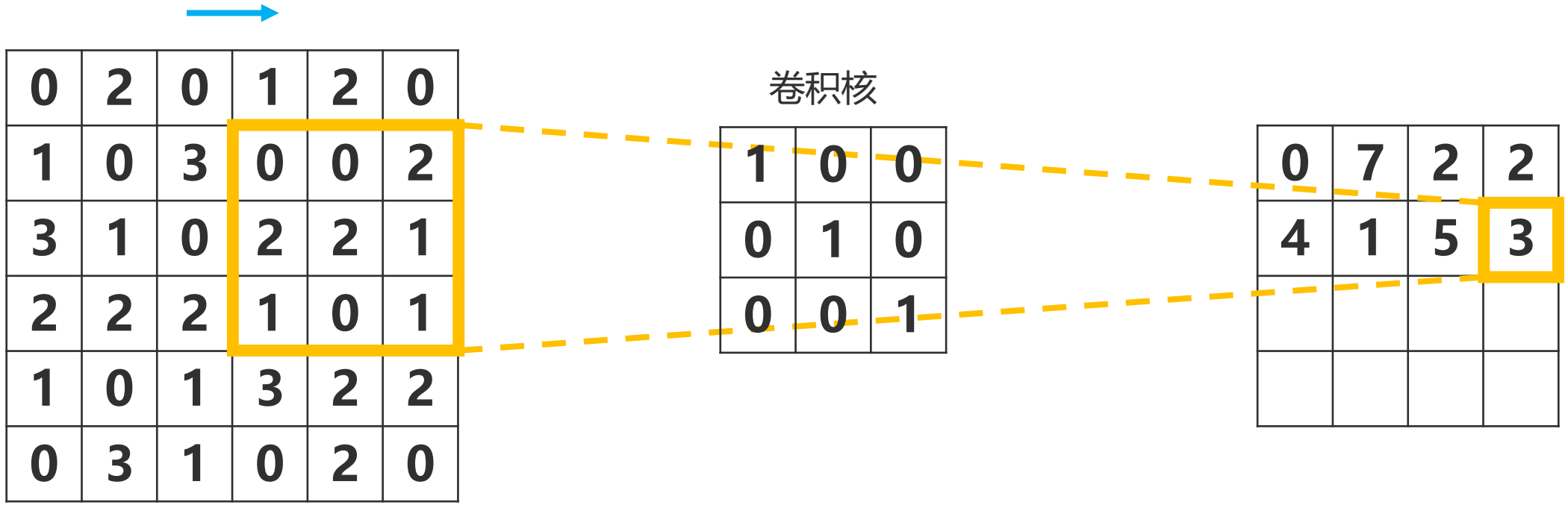
卷积计算


卷积核
0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1

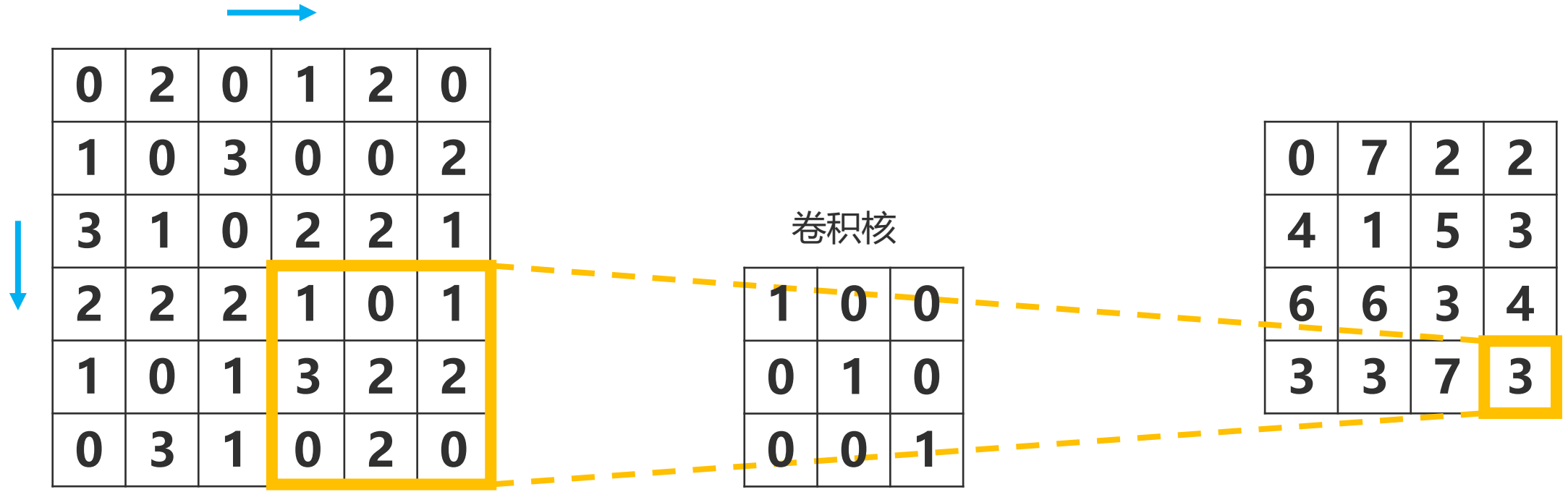
卷积计算



卷积计算


卷积核
0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1 5 3

卷积计算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
卷积核
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1 5 3
6 6 3 4
3 3 7 3

卷积计算

0	0	0	0	0	0	0	0
0	0	2	0	1	2	0	0
0	1	0	3	0	0	2	0
0	3	1	0	2	2	1	0
0	2	2	2	1	0	1	0
0	1	0	1	3	2	2	0
0	0	3	1	0	2	0	0
0	0	0	0	0	0	0	0

卷积核

1	0	0
0	1	0
0	0	1

0	5	0	1	2	0
2	0	7	2	2	4
5	4	1	5	3	1
2	6	6	3	4	3
4	3	3	7	3	2
0	4	1	1	5	0

黑白图像

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

单通道（1 channel）

彩色图像

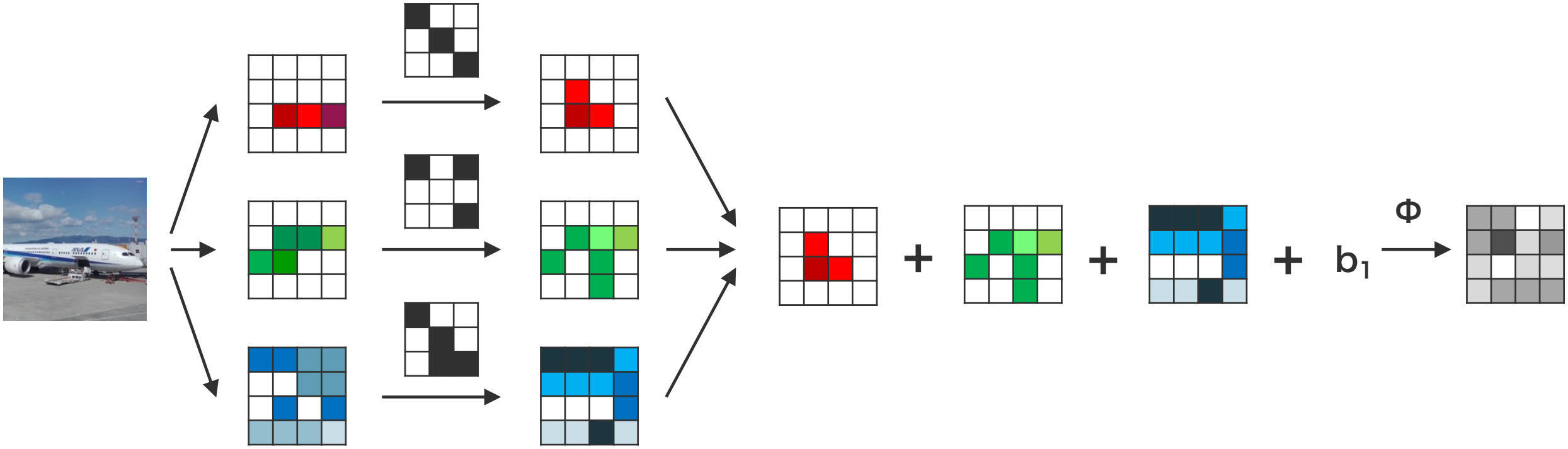
021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	013	103	053
031	051	053	081	041
093	049	058	093	036
054	101	061	074	041
109	023	026	053	022
111	042	024	031	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

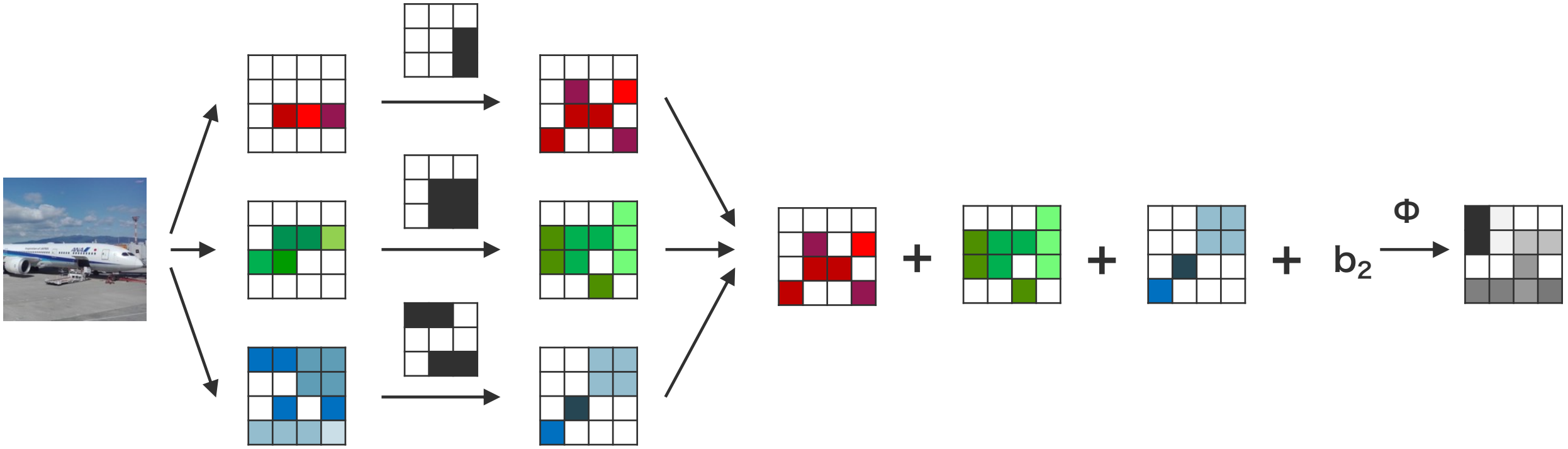
224	243	217	236	216
220	231	221	254	253
251	254	255	245	253
254	253	250	249	251

三通道（3 channels）

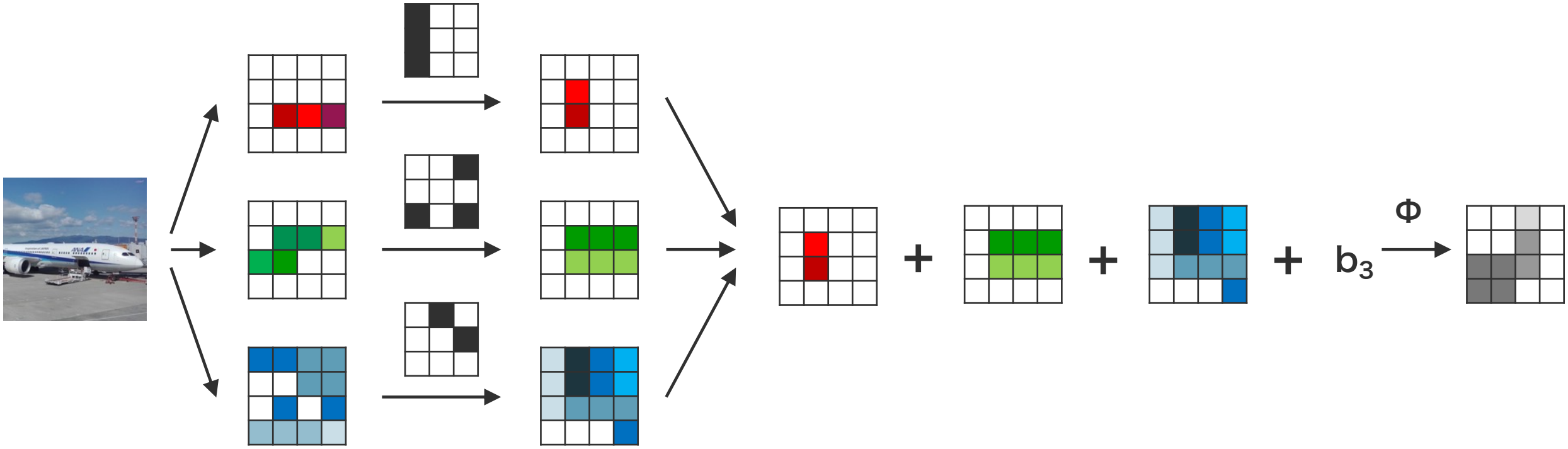
卷积计算



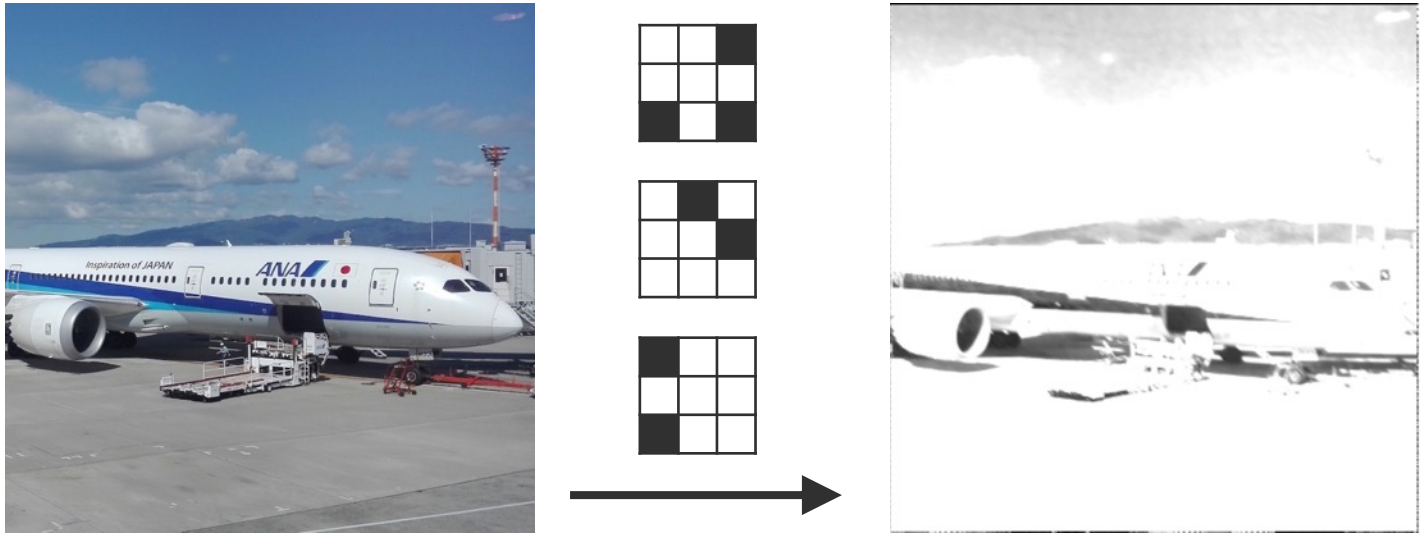
卷积计算



卷积计算

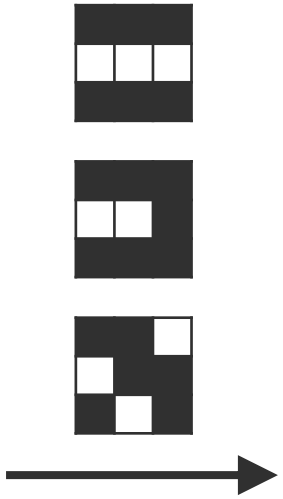


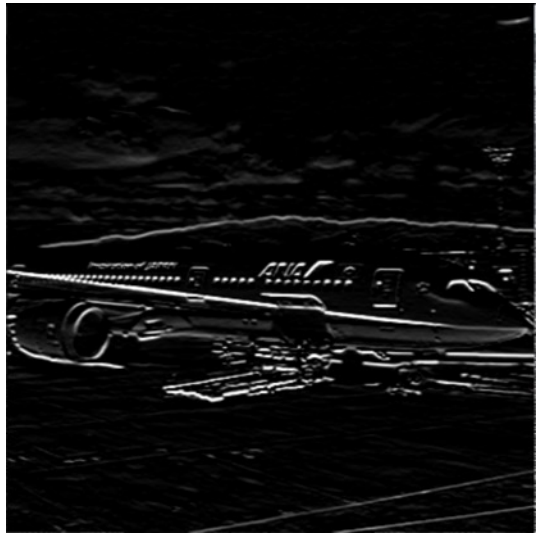
卷积计算



使用多个不同的卷积核可以从一张图片中产生多张黑白图片。

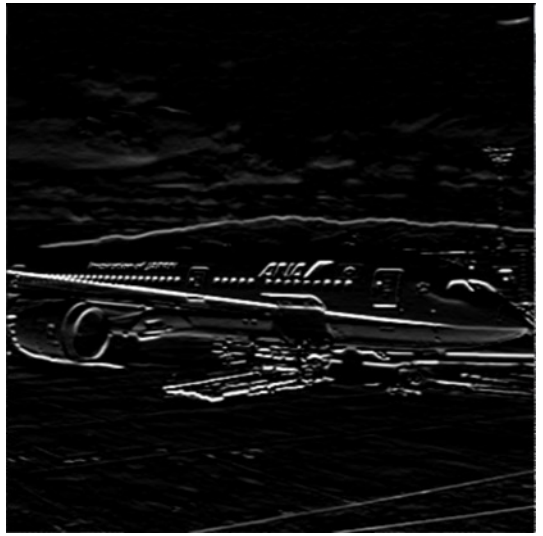
卷积计算





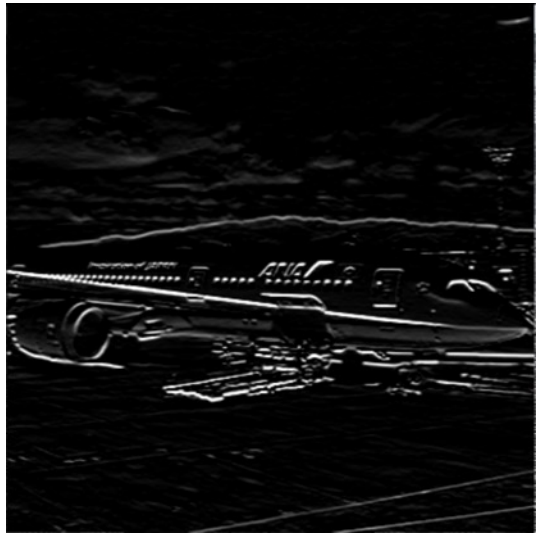
使用多个不同的卷积核可以从一张图片中产生多张黑白图片。

卷积计算



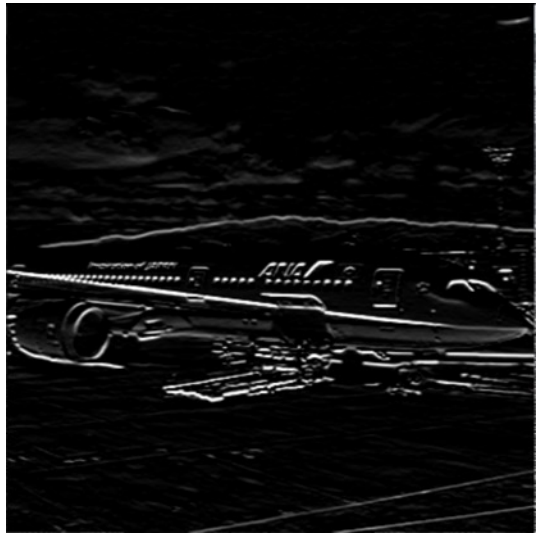
使用多个不同的卷积核可以从一张图片中产生多张黑白图片。

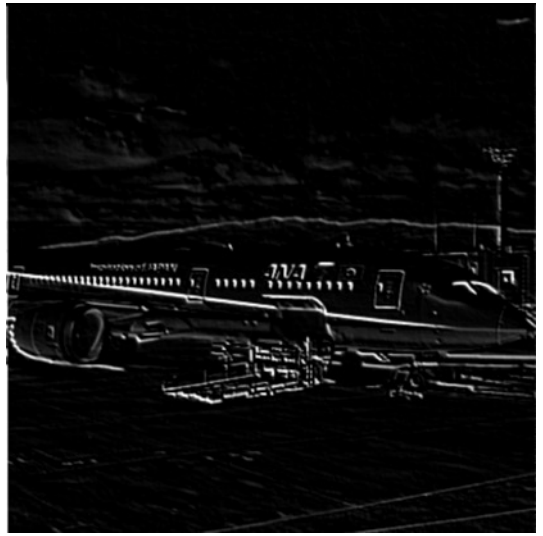
卷积计算



使用多个不同的卷积核可以从一张图片中产生多张黑白图片。

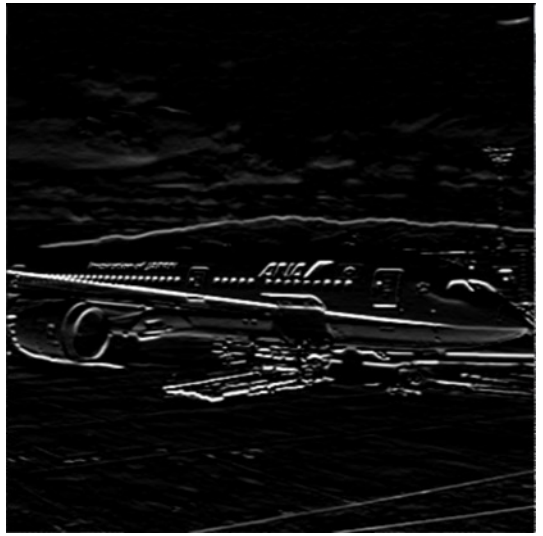
卷积计算

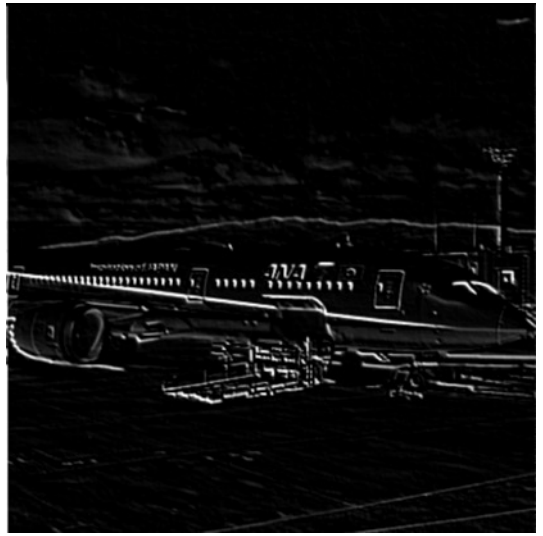




使用多个不同的卷积核可以从一张图片中产生多张黑白图片。

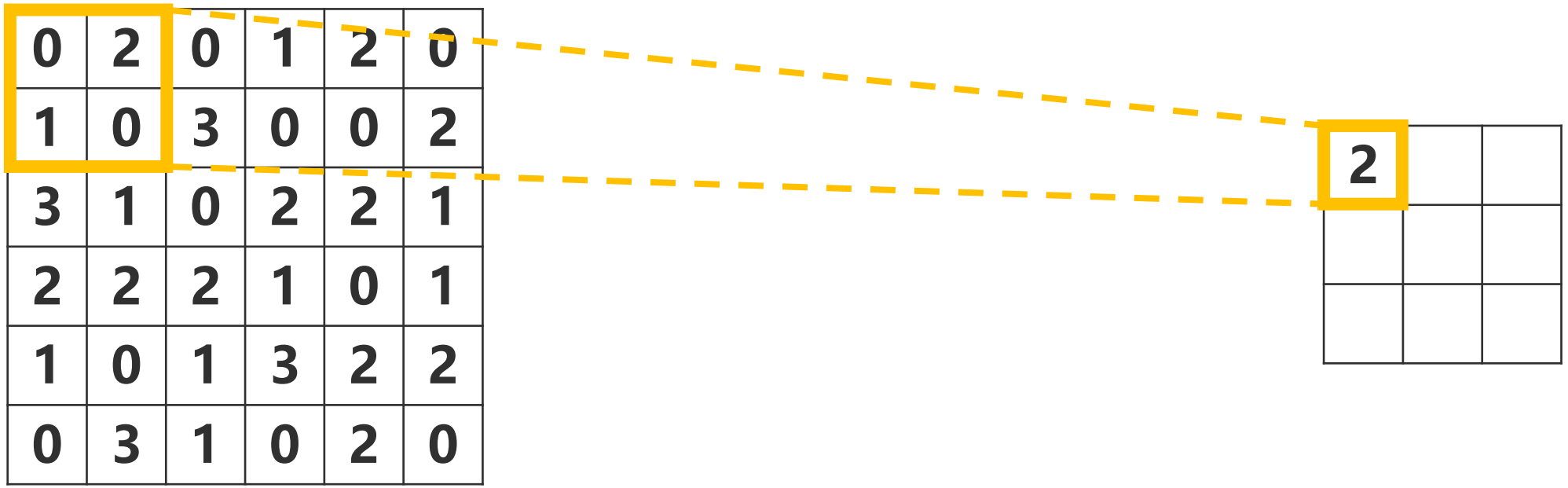
卷积计算



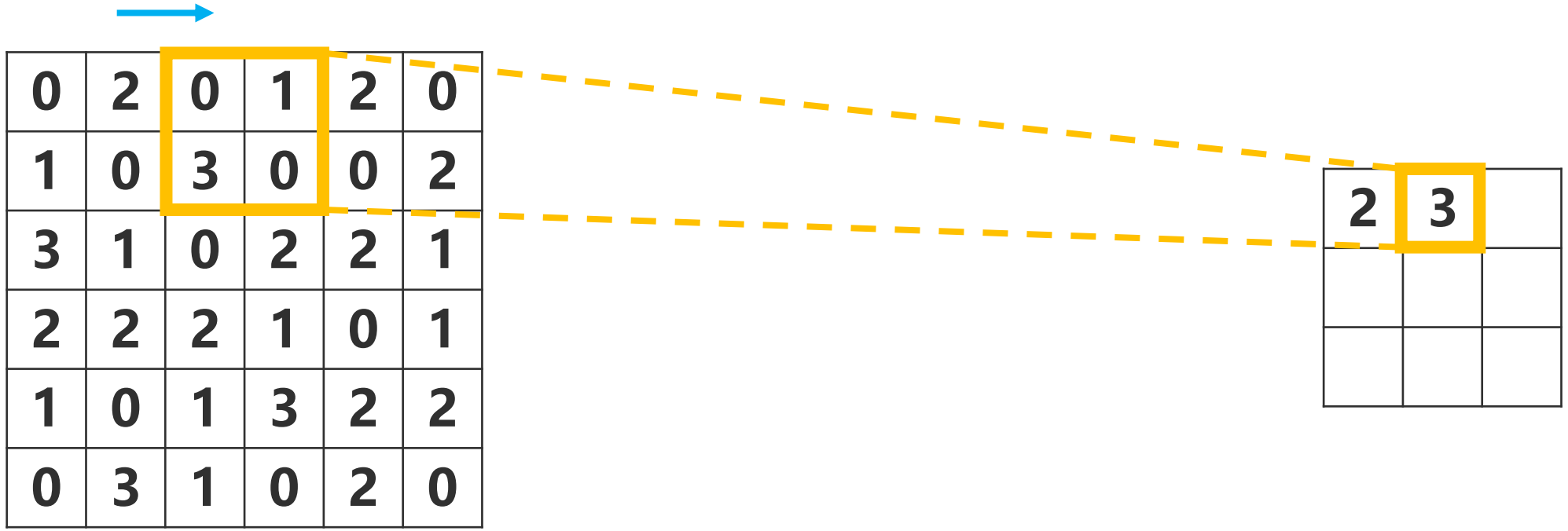


使用多个不同的卷积核可以从一张图片中产生多张黑白图片。

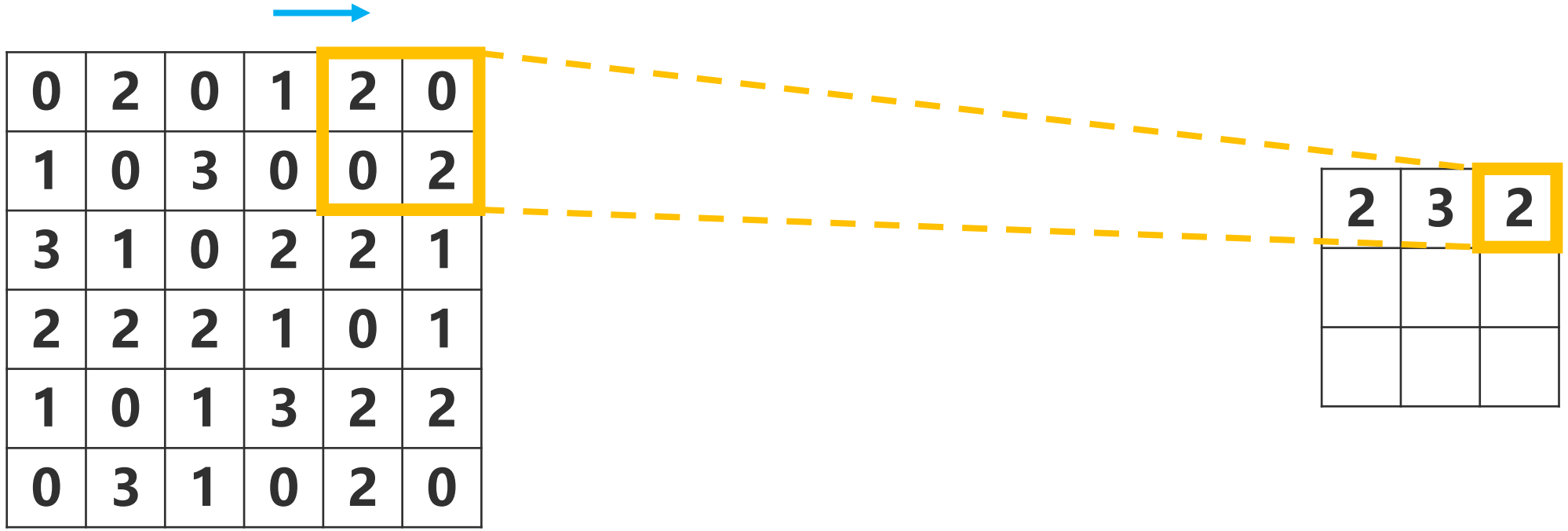
汇聚层



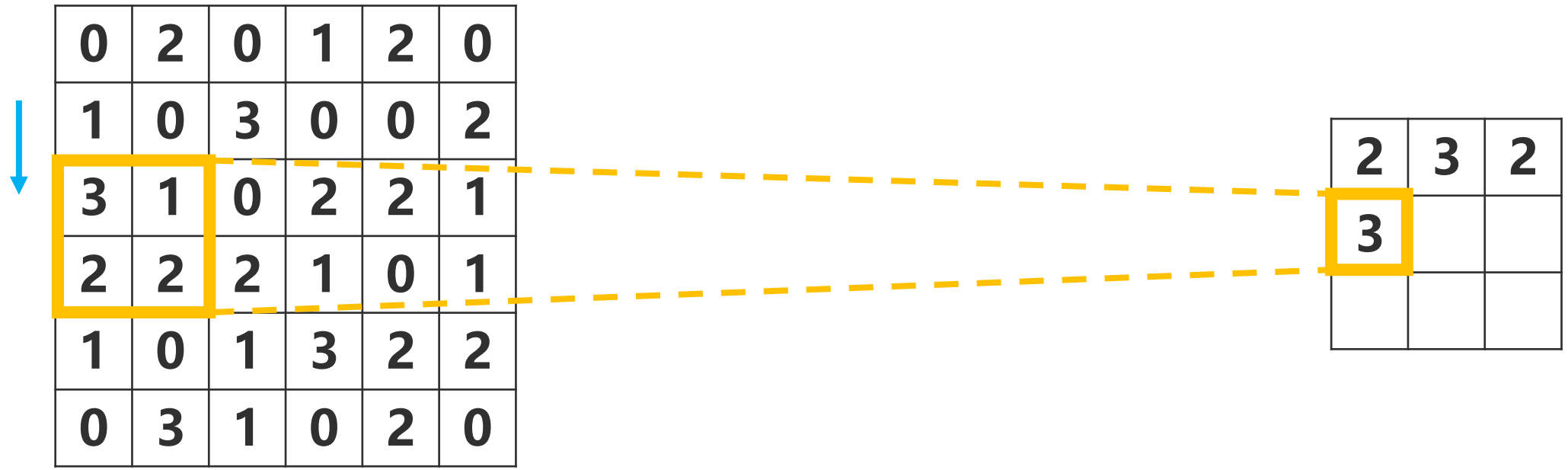
汇聚层



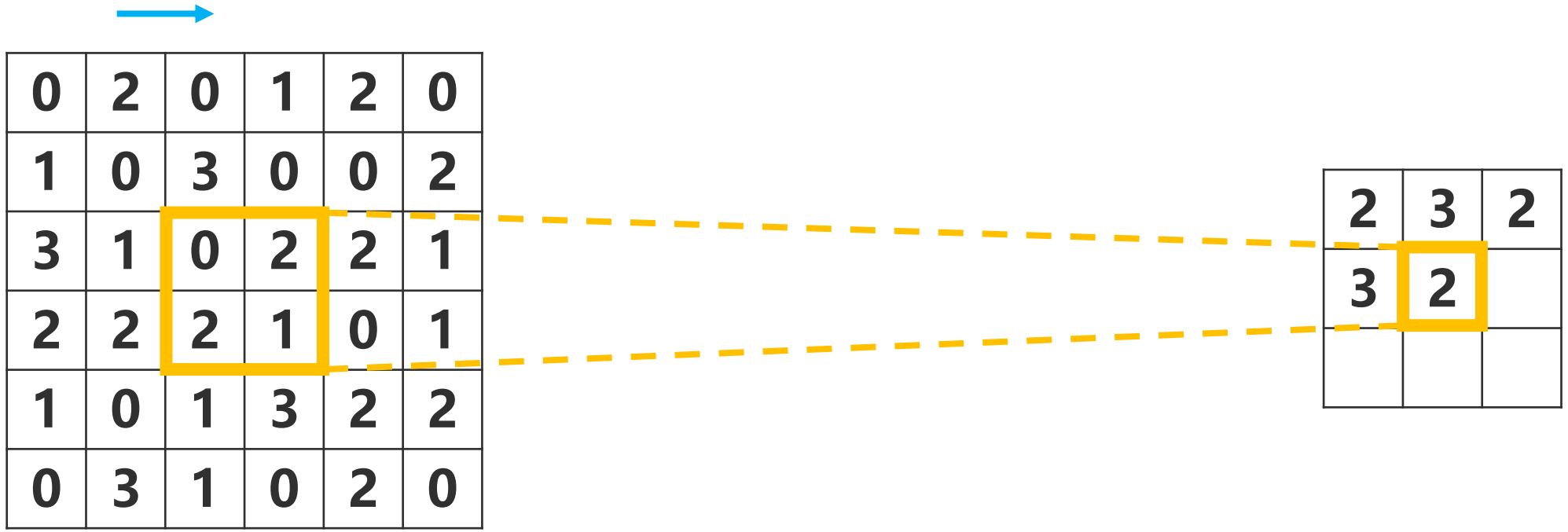
汇聚层



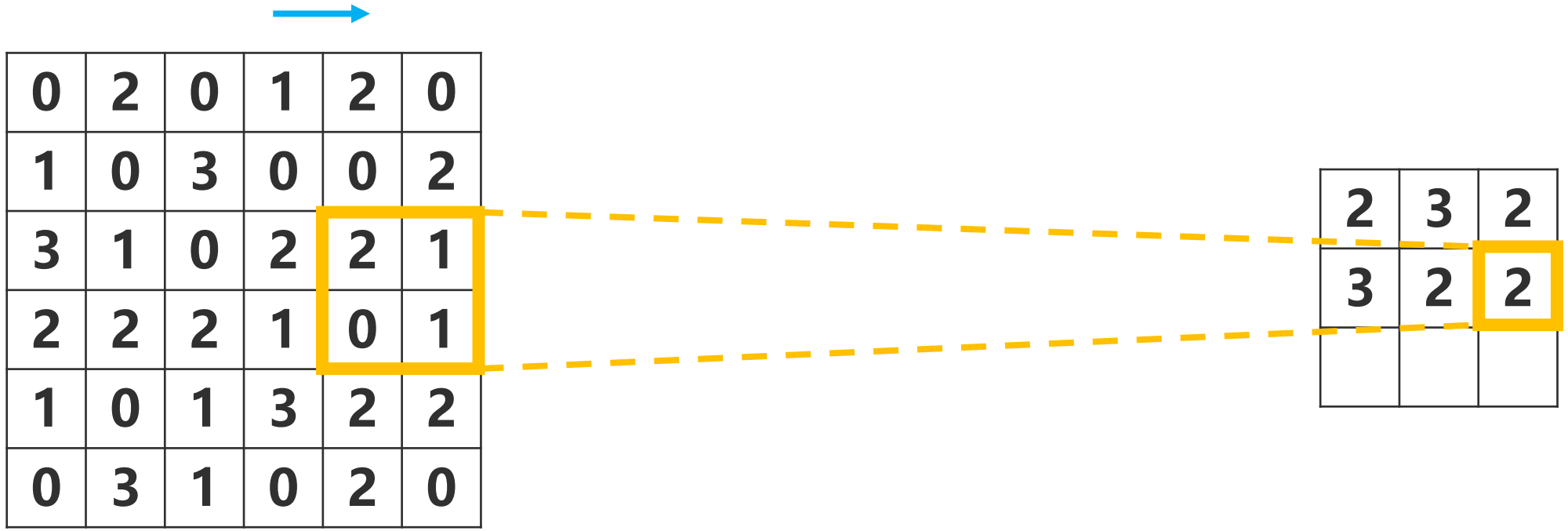
汇聚层



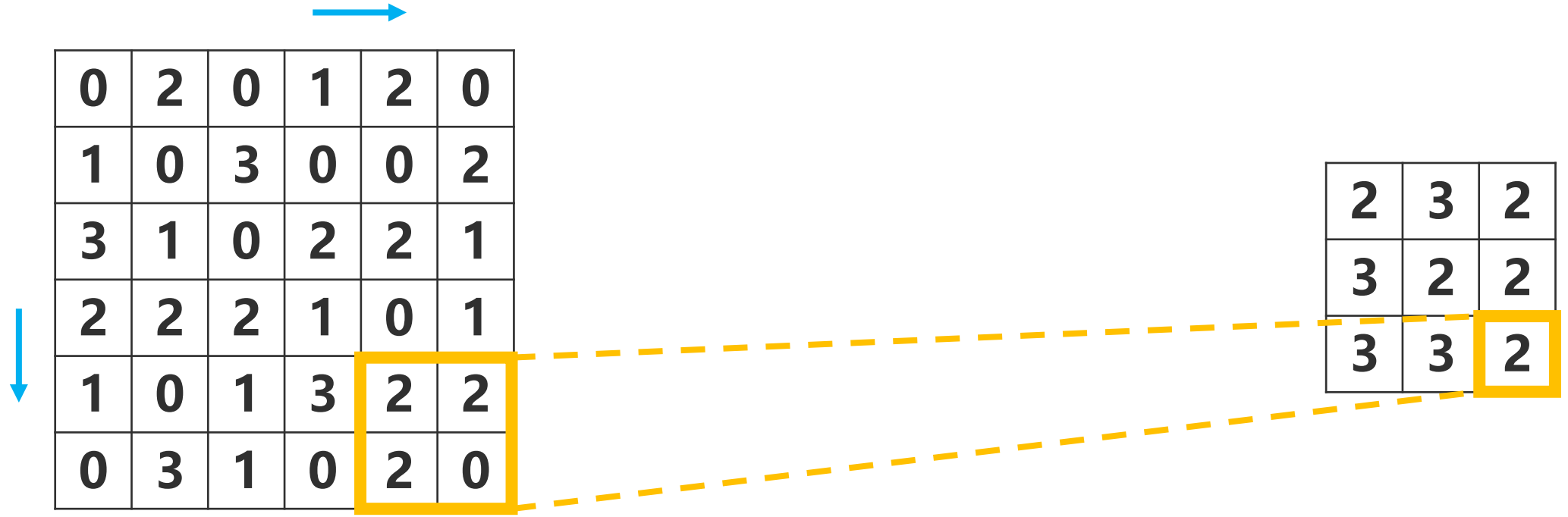
汇聚层



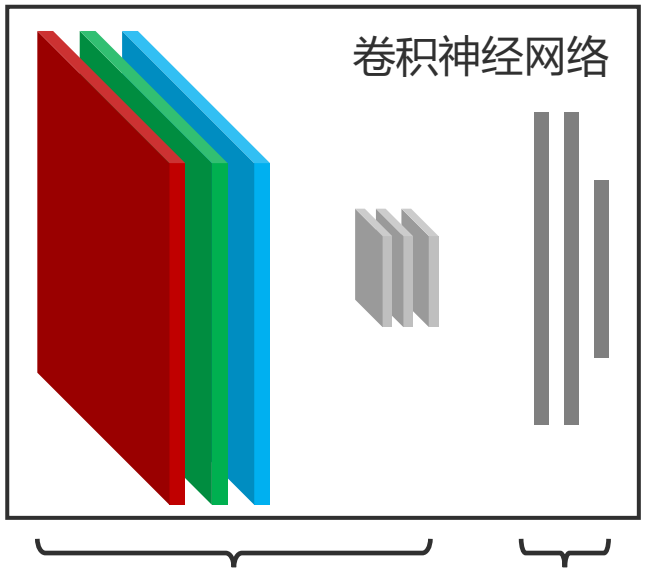
汇聚层



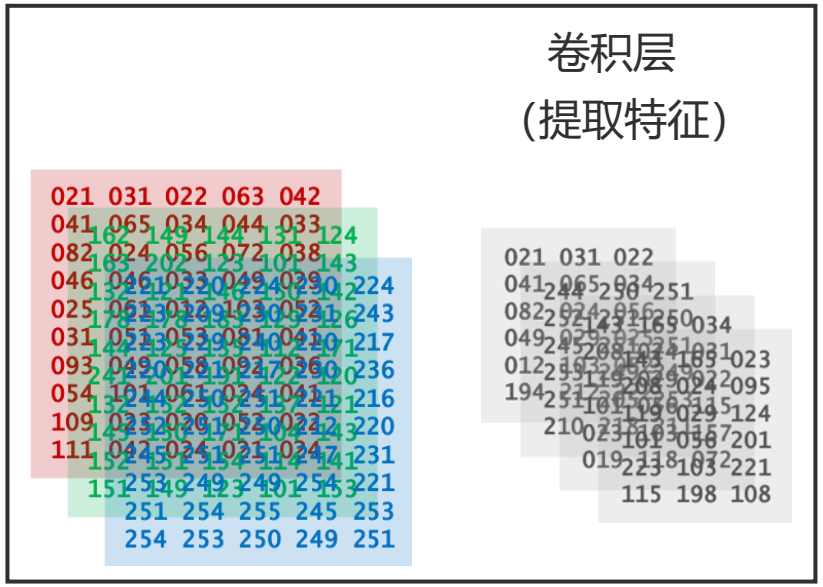
汇聚层

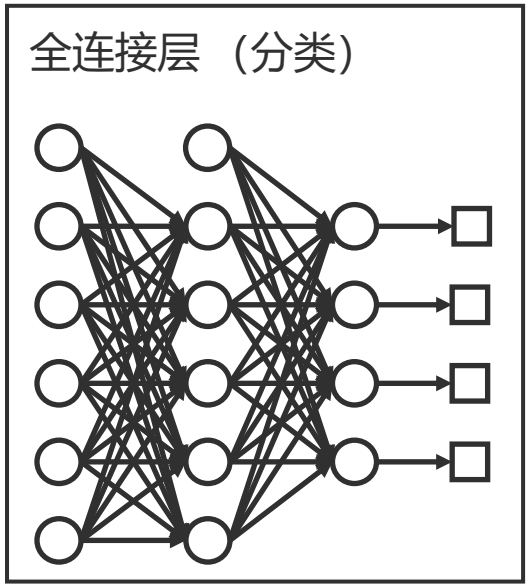


卷积神经网络

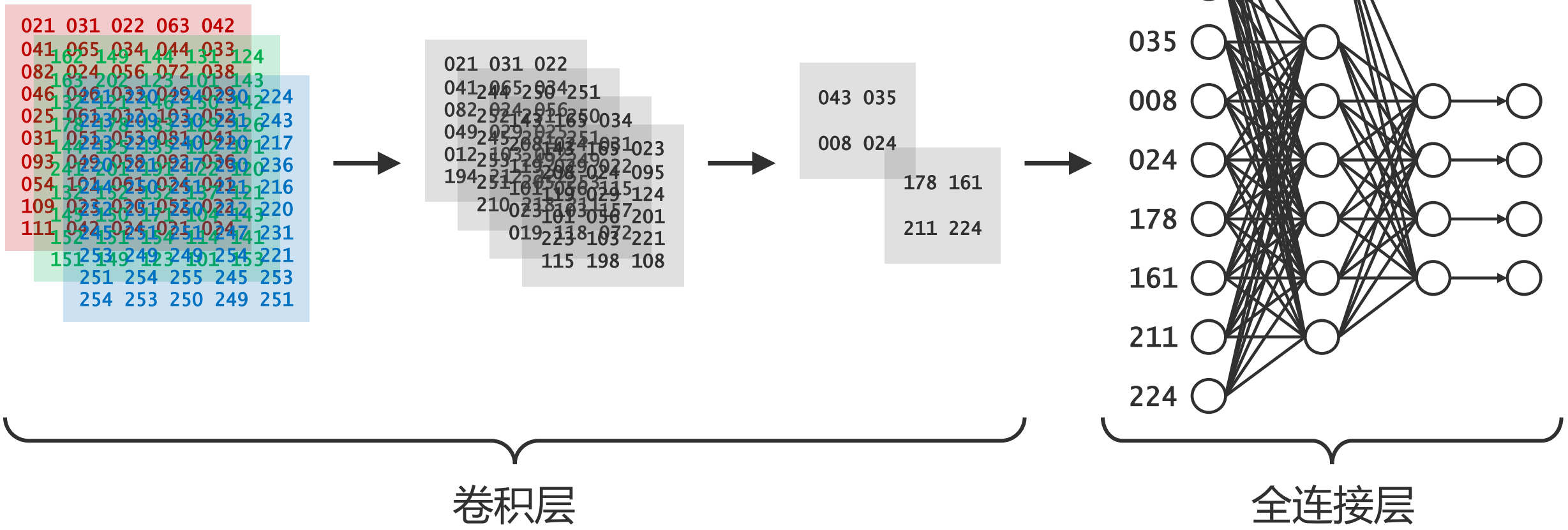



```
airplane 56.1%
sky      25.0%
cloud    12.4%
mountain  6.5%
```



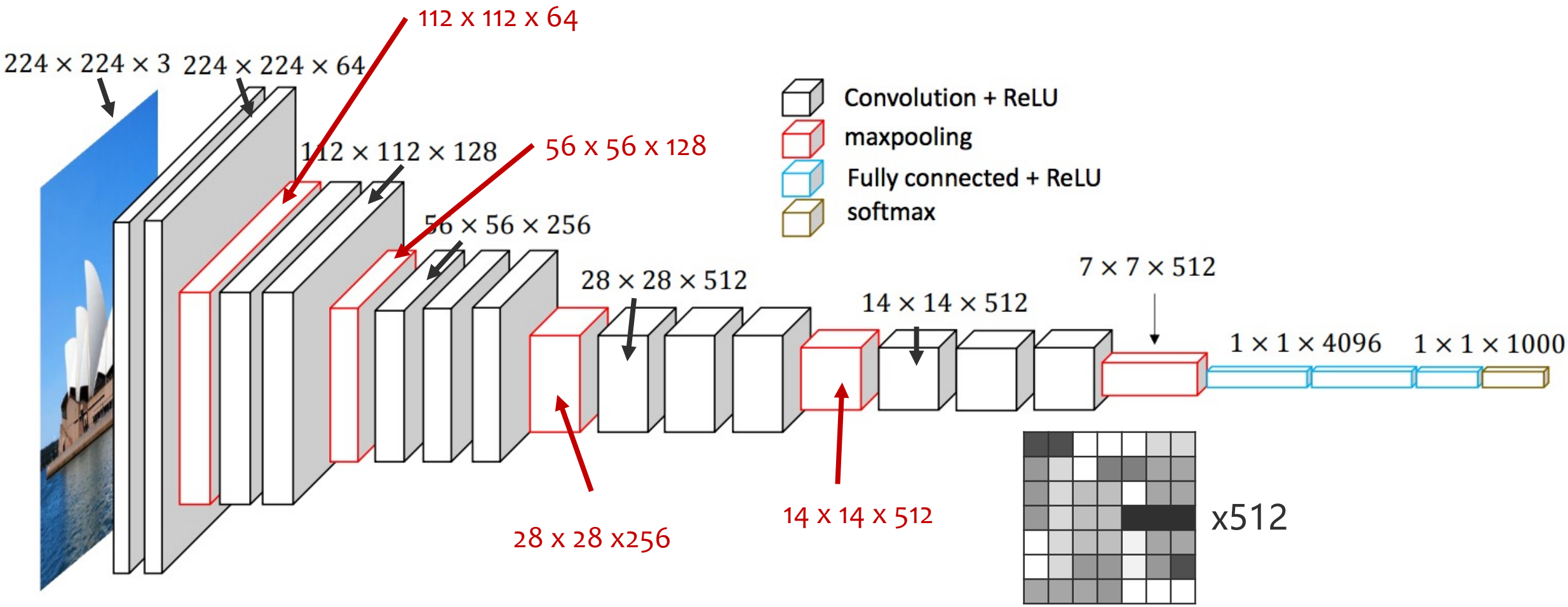


卷积神经网络


021 031 022 063 042
041 065 034 044 033
082 024 056 072 038
046 046 033 049 039
025 053 032 103 053
031 051 053 081 041
093 049 058 093 036
054 101 061 024 041
109 023 020 055 022
111 042 034 031 034
043 035
008 024
178 161
211 224
043
035
008
024
178
161
211
224
卷积层
全连接层

VGG16

https://doi.org/10.1145/3038912.3052638



特征提取

分类

物体分类

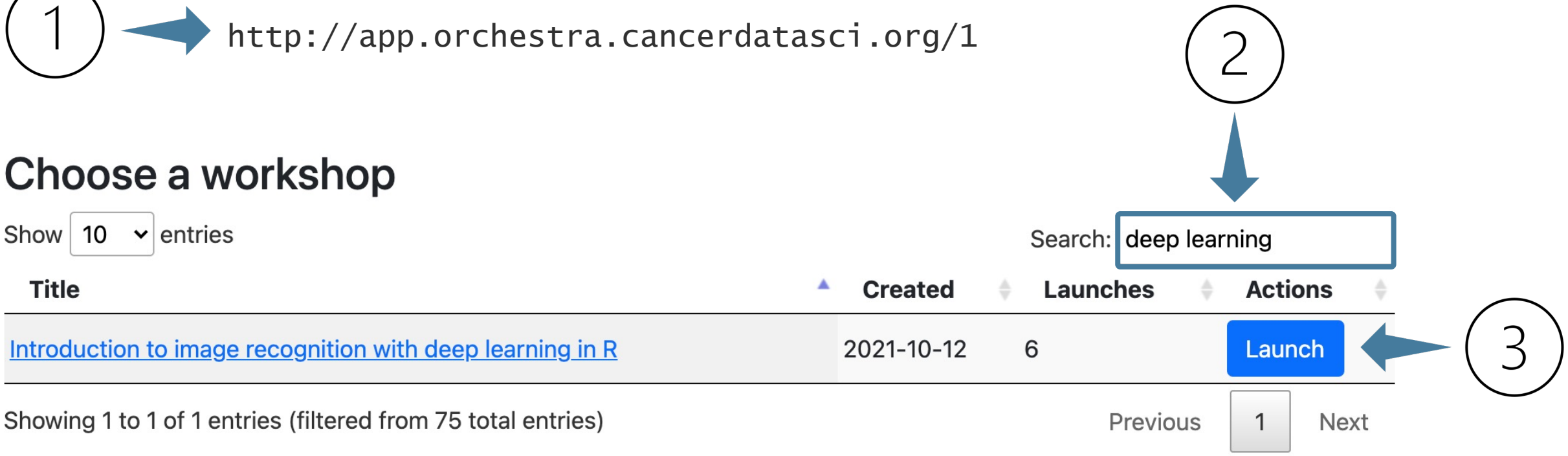
神经网络

卷积神经网络

torch

setup

1 → `http://app.orchestra.cancerdatasci.org/1`



setup

Get ready.

You will see a progress bar and a disabled "Launch Workshop" button below untill your workshop has successfully been deployed to our cluster. Once available, the progress bar will disappear and the button will become active. Click the button and you will be redirected to the RStudio login screen. Use these credentials to login:

Login Credentials

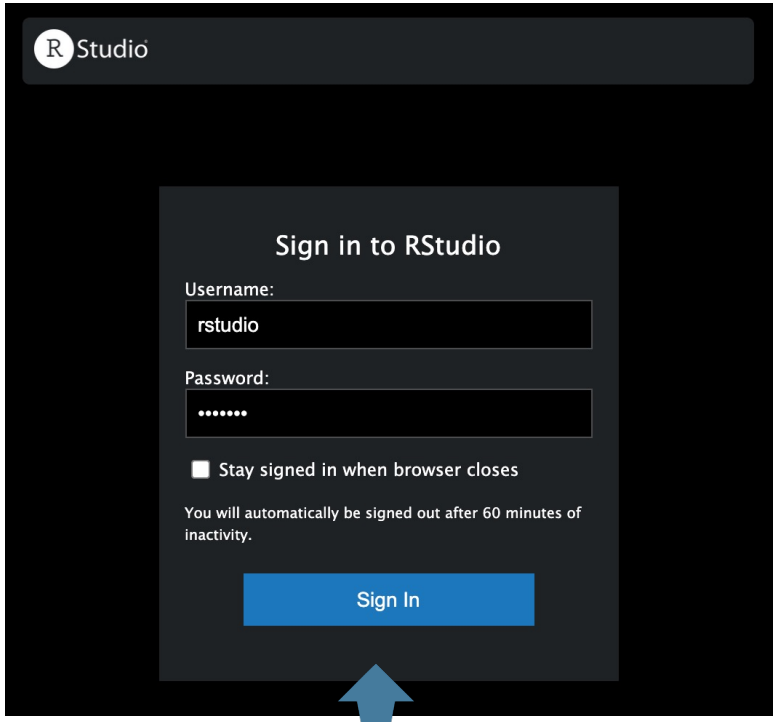
- Username: **rstudio**
- Password: **rstudio**

Make note of your personalized URL, which you can reuse for up to **8 hours** is:
http://mkvukbhj.orchestra.cancerdatasci.org

The button below will become active when your workshop is ready to run

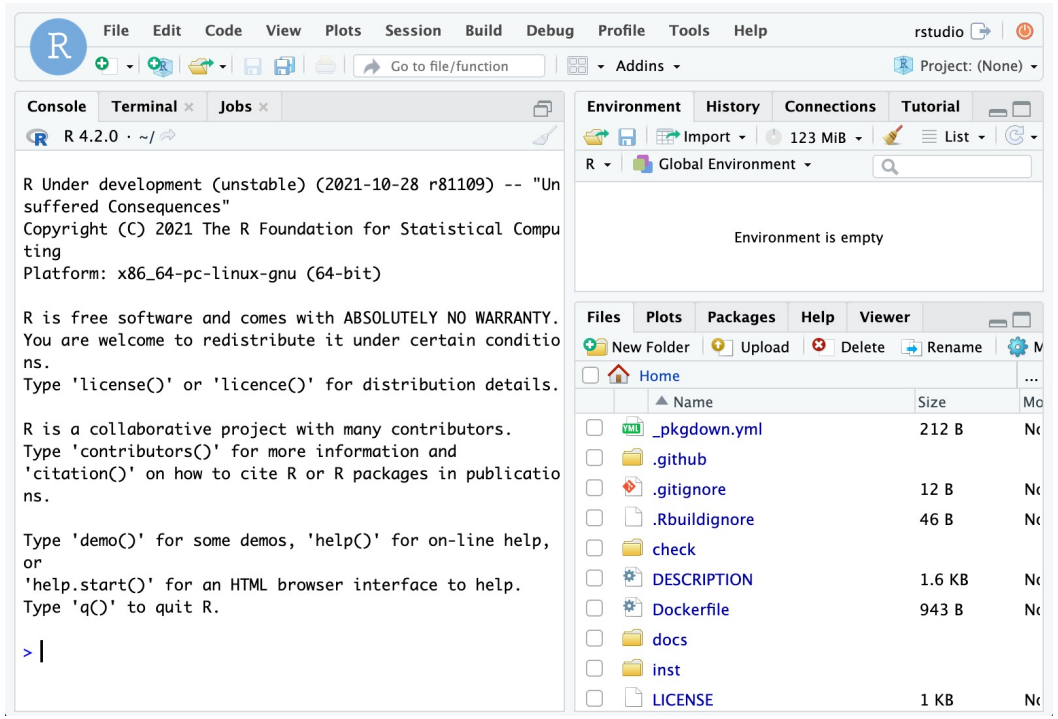
Launch Workshop

4

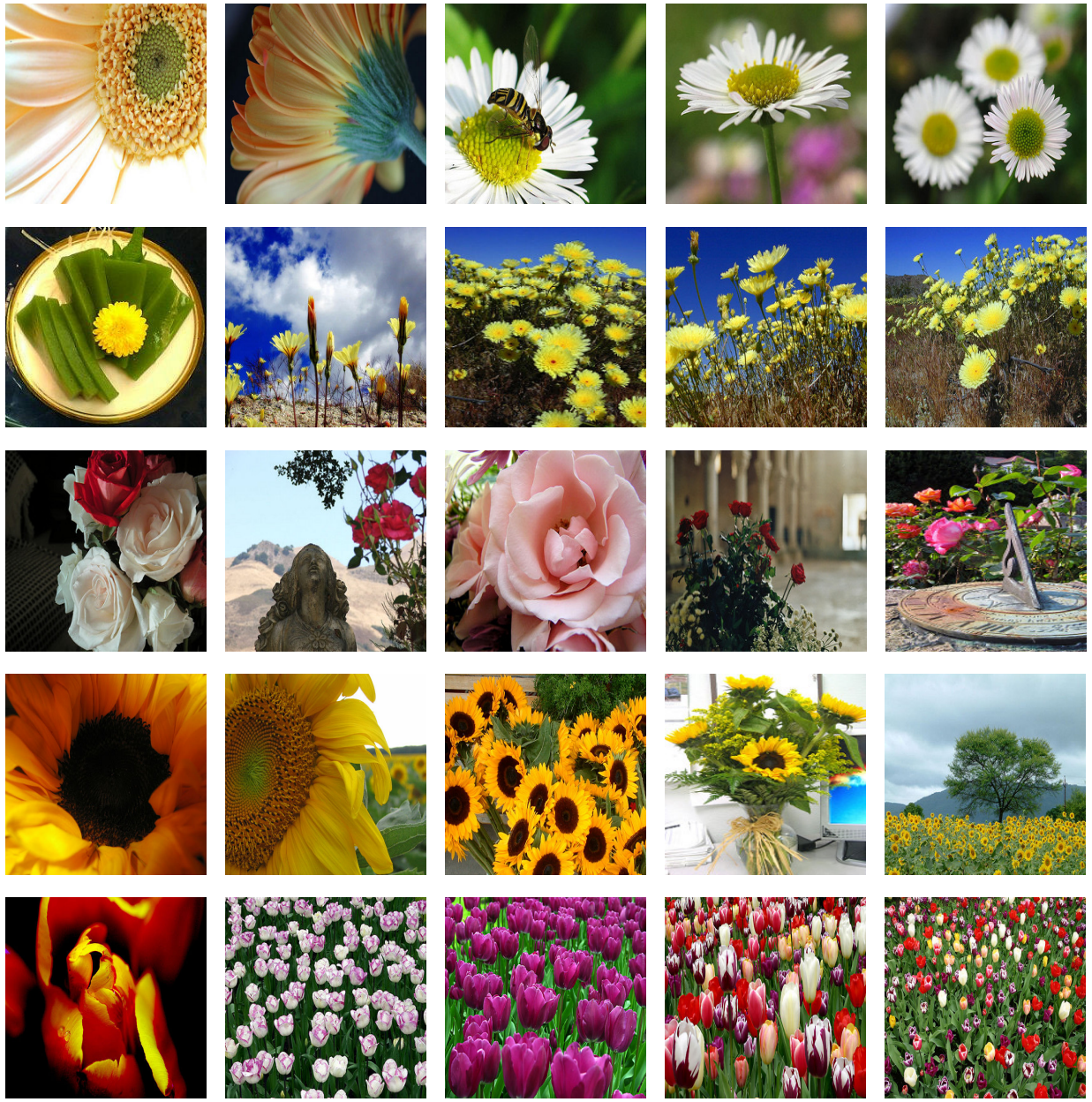


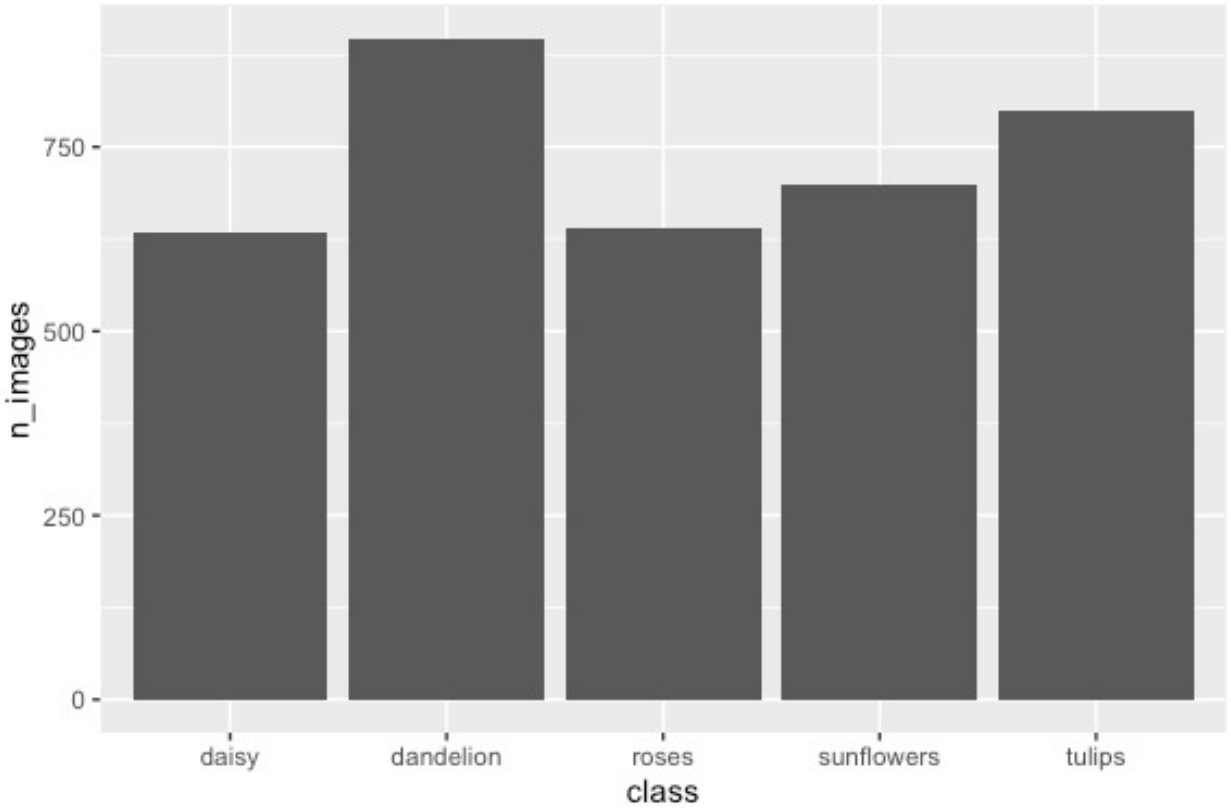
5

Username: rstudio
Password: rstudio



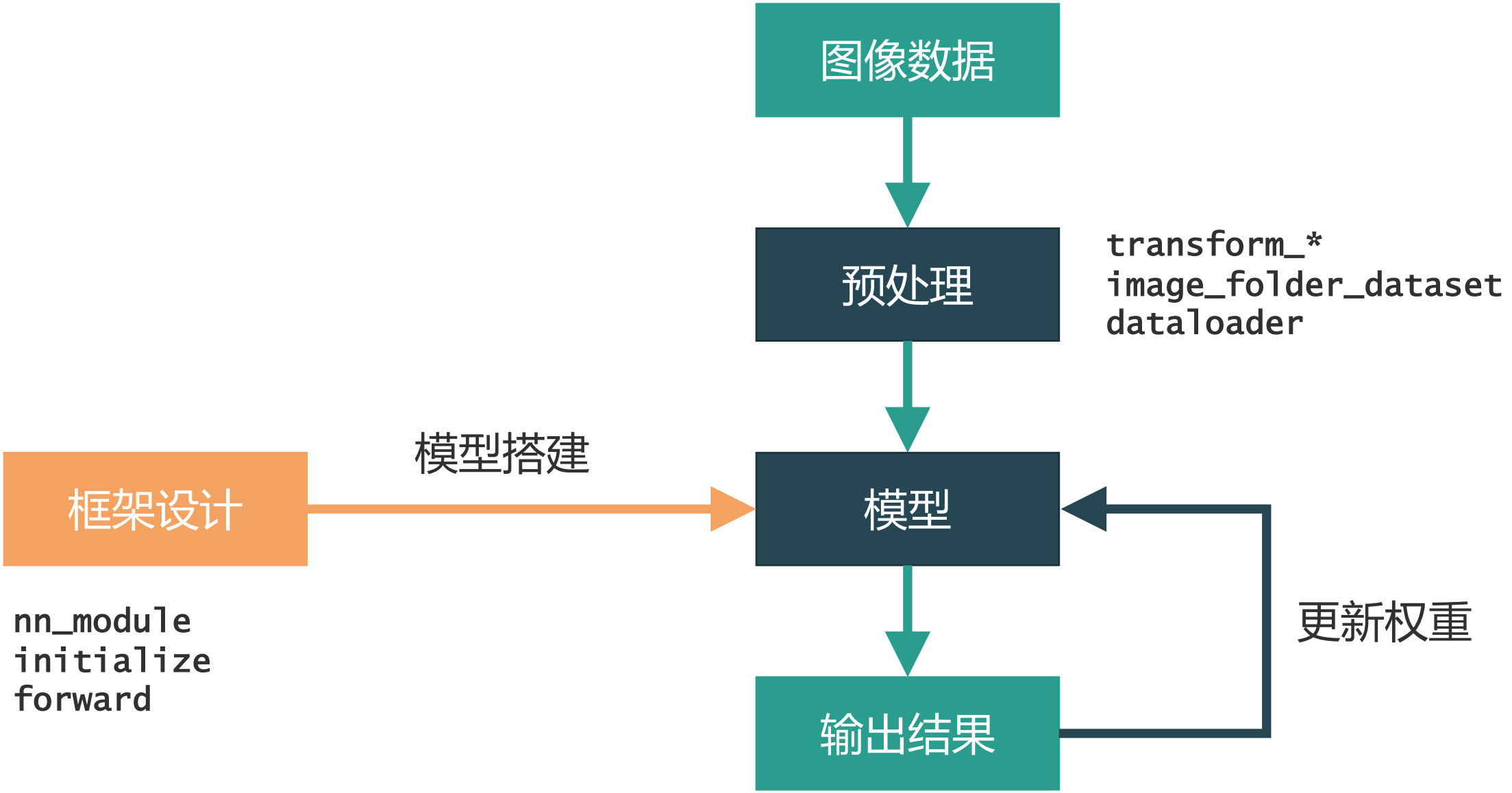
tf_flowers dataset





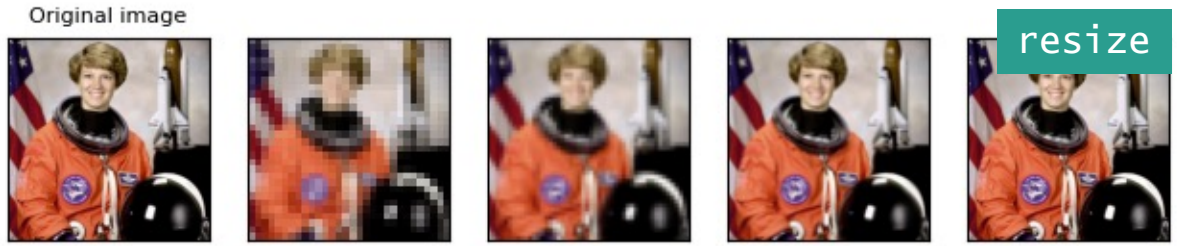
https://www.tensorflow.org/datasets/catalog/tf_flowers

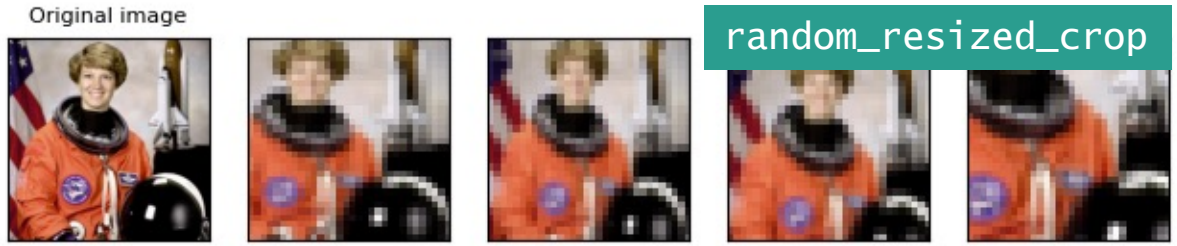
torch

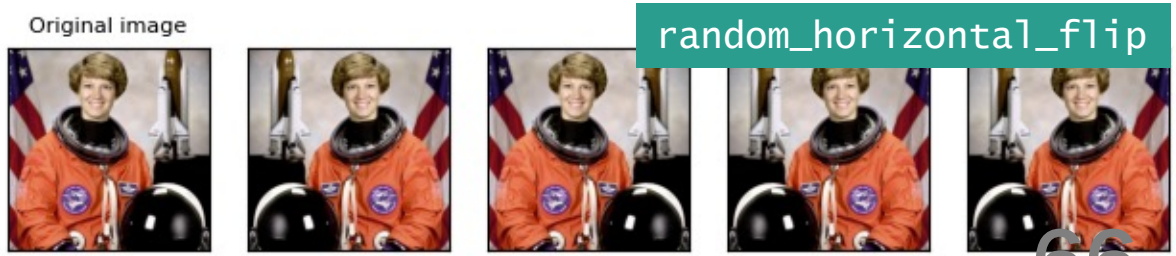

图像数据
预处理
transform_*
image_folder_dataset
dataloader
模型搭建
框架设计
模型
nn_module
initialize
forward
更新权重
输出结果

预处理

```
train_transforms <- function(img) {
    img <- transform_to_tensor(img)
    img <- transform_resize(img, size = c(512, 512))
    img <- transform_random_resized_crop(img, size = c(224, 224))
    img <- transform_color_jitter(img)
    img <- transform_random_horizontal_flip(img)
    img <- transform_normalize(img, mean = c(0.485, 0.456, 0.406),
                                    std = c(0.229, 0.224, 0.225))
    img
}
```



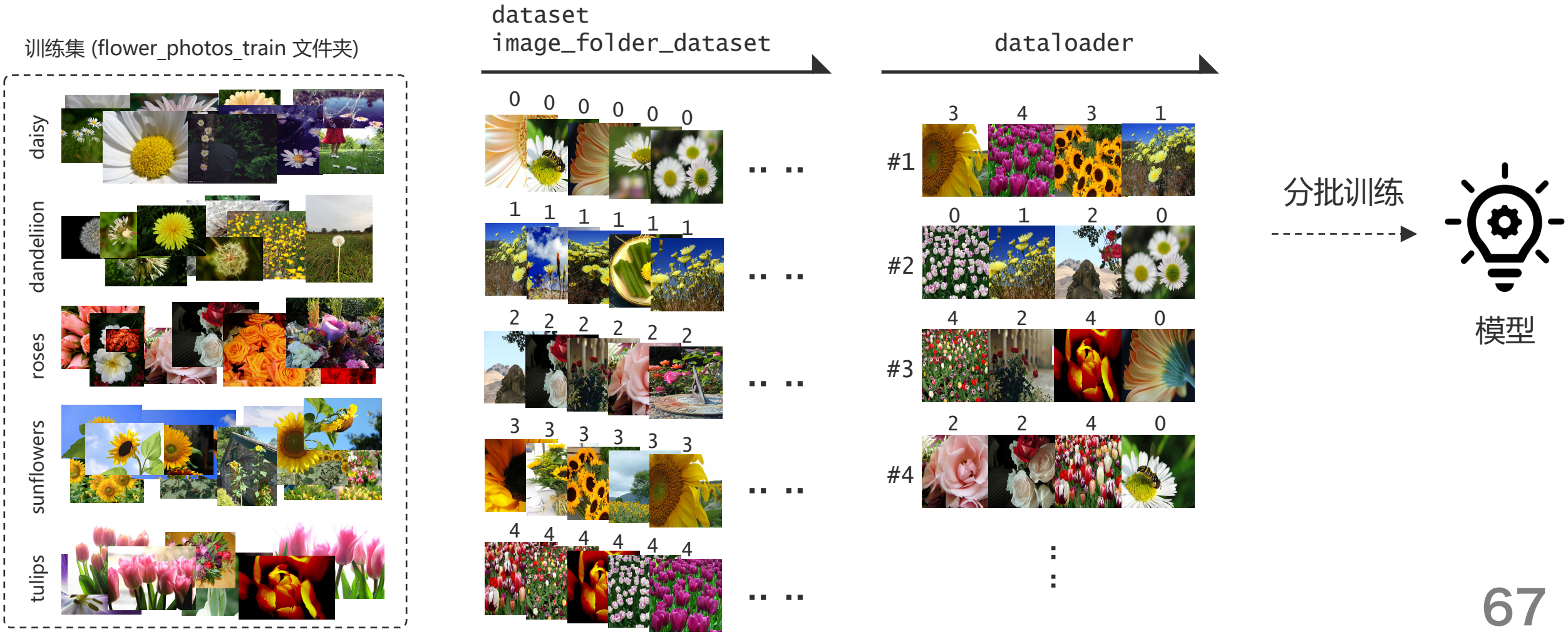




https://pytorch.org/vision/stable/auto_examples/plot_transforms.html

预处理

```
dataset_train <- image_folder_dataset('flower_photos_train',
                                      transform = train_transforms)
dataloader_train <- dataloader(dataset_train)
```



定义模型框架

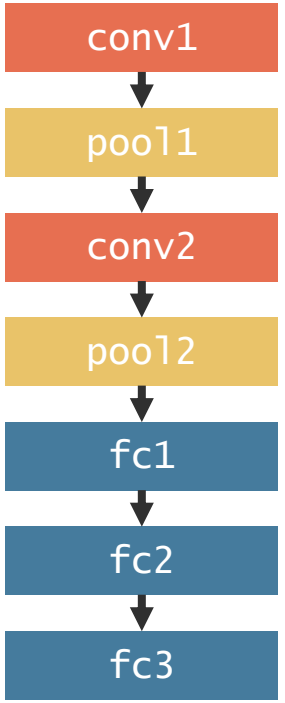
```
SimpleCNN <- nn_module(
    "SimpleCNN",
    initialize = function(n_classes) {
        self$conv1 <- nn_conv2d(3, 16, 5)
        self$pool1 <- nn_max_pool2d(2, 2)
        self$conv2 <- nn_conv2d(16, 32, 5)
        self$pool2 <- nn_max_pool2d(2, 2)

        n_inputs <- (((((224 - 5 + 1) / 2) - 5 + 1) / 2) ^ 2)* 32

        self$fc1 <- nn_linear(in_features = n_inputs, out_features = 512)
        self$fc2 <- nn_linear(in_features = 512, out_features = 64)
        self$fc3 <- nn_linear(in_features = 64, out_features = n_classes)
    },
    forward = function(x) {
        x <- self$conv1(x)
        x <- nnf_relu(x)
        x <- self$pool1(x)
        x <- self$conv2(x)
        x <- nnf_relu(x)
        x <- self$pool2(x)

        # convert a matrix to a vector
        x <- torch_flatten(x, start_dim = 2)

        x <- self$fc1(x)
        x <- nnf_relu(x)
        x <- self$fc2(x)
        x <- nnf_relu(x)
        x <- self$fc3(x)
        x
    })
```

准备卷积层的组件

准备全连接层的组件

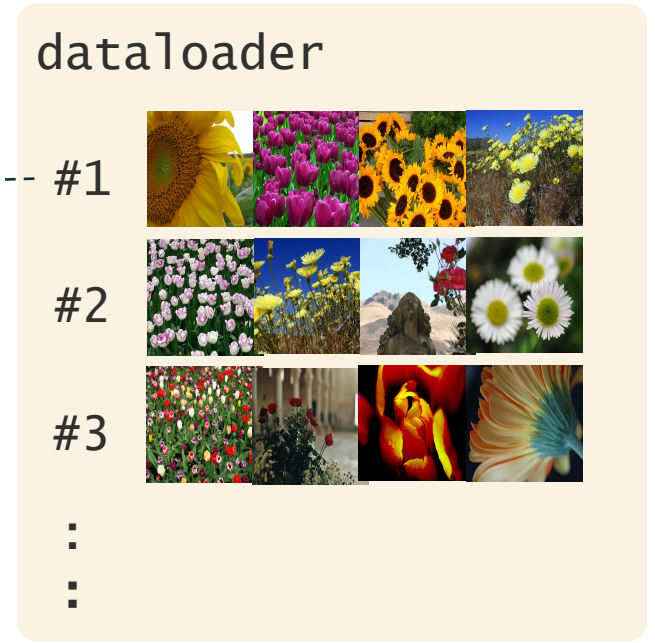
定义卷积层的组件的连接顺序

定义全连接层的组件连接顺序



训练模型

```
model <- SimpleCNN(length(dataset_train$classes))
model$to(device = 'cpu')
model$train()

loss_train <- c()

for (epoch in 1:5) {
    loss_running <- 0
    n_train_samples <- 0



    coro::loop(for ( b in dataloader_train) {

        optimizer$zero_grad()

        output <- model(b$x$to(device = 'cpu'))

        loss <- criterion(output, b$y$to(device = 'cpu'))

        loss$backward()

        optimizer$step()

        loss_running <- loss_running + loss$item() * nrow(b$x)
        n_train_samples <- n_train_samples + nrow(b$x)

    })

    loss_train <- c(loss_train, loss_running / n_train_samples)
    cat(sprintf("epoch %d  loss: %3f\n", epoch, loss_running / n_train_samples))
}
```



- 预测
- 计算损失
- 更新权重