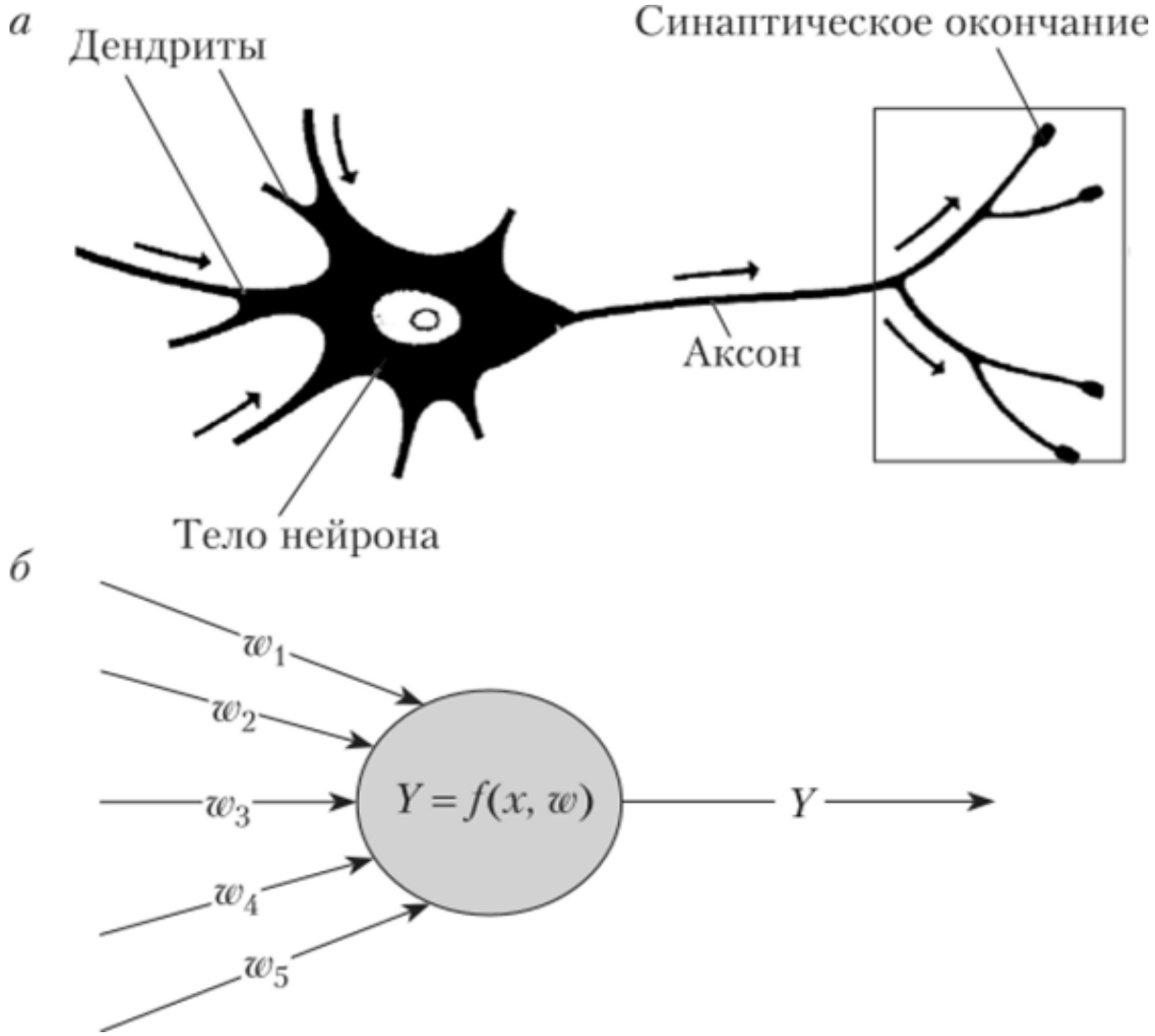
# Нейронные сети

# Модель нейрона



```
from __future__ import absolute_import, division,
print_function, unicode_literals
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import imageio
```

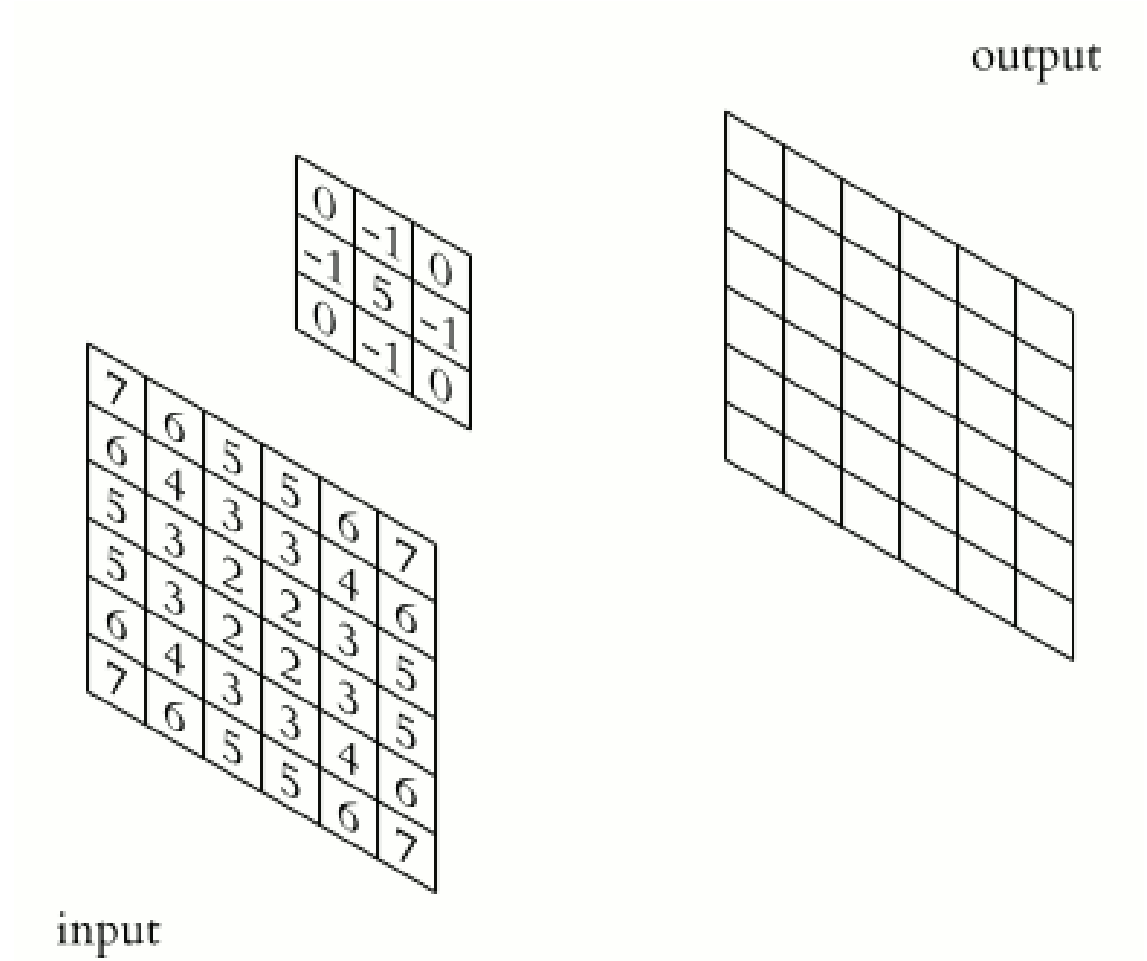
```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) =
mnist.load_data()
print(type(x_train))
print(x_train.shape)
print(y_train.shape)
```

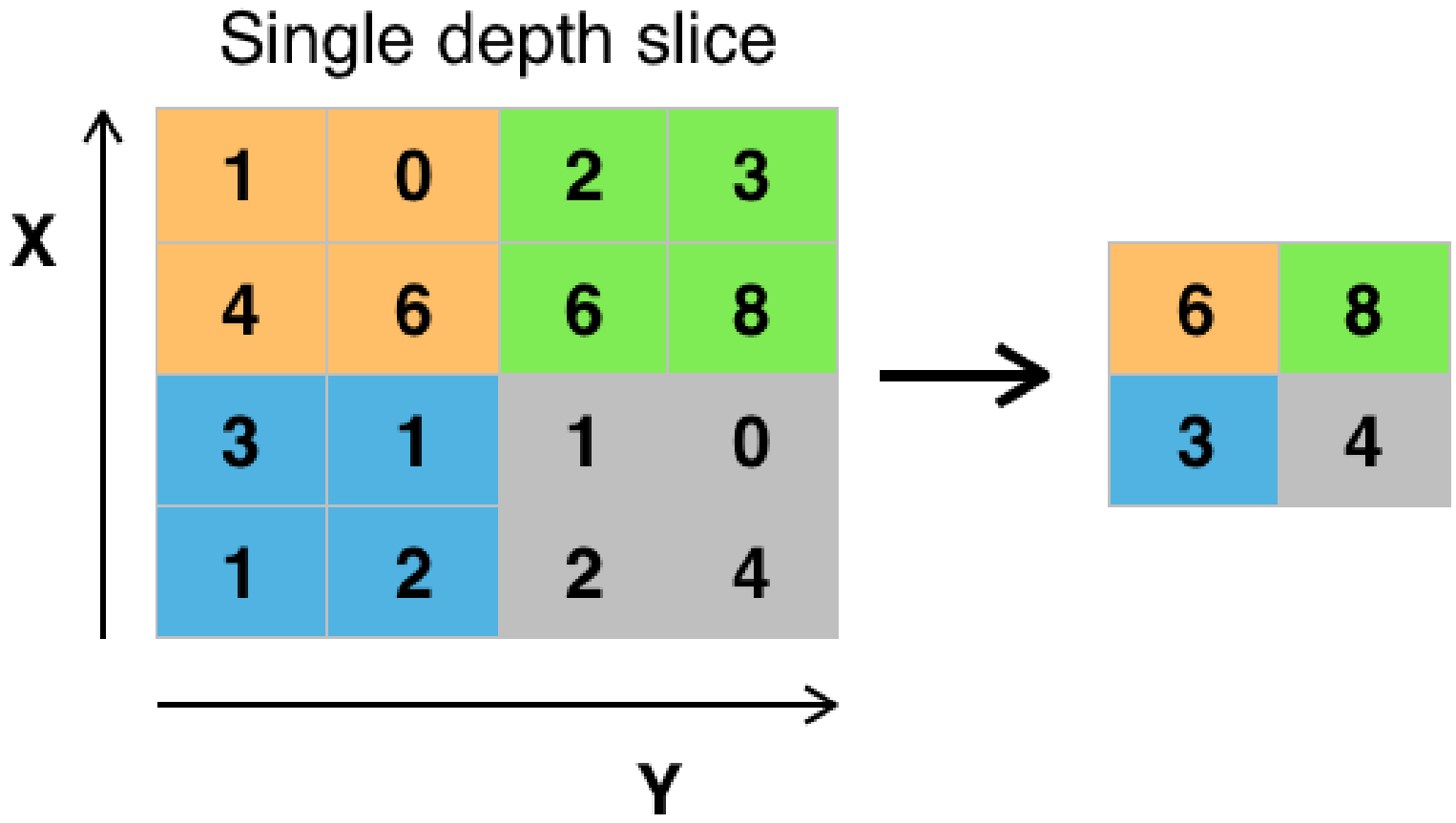
```
plt.figure(figsize=(8,8))
for i in range(16):
  plt.subplot(4, 4, i + 1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(x_train[i], cmap=plt.cm.binary)
  plt.colorbar()
  plt.xlabel(y_train[i])
plt.show()
```

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

# Конволюционный фильтр



# Слой подвыборки



```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(
  optimizer='adam’,
  loss='sparse_categorical_crossentropy’,
  metrics=['accuracy']
)
model.fit(x_train, y_train, epochs=5)
print(model.evaluate(x_test, y_test))
```

```
def model_answer(model, filename, display=True):
  image = imageio.imread(filename)
  image = np.mean(image, 2, dtype=float)
  image /= 255.0
  if display:
    plt.xticks([])
    plt.yticks([])
    plt.imshow(image, cmap=plt.cm.binary)
    plt.xlabel(filename)
    plt.show()
  image = np.expand_dims(image, 0)
  image = np.expand_dims(image, -1)
  return np.argmax(model.predict(image))
```

```
for i in range(10):
    filename = 'digit_examples/%d.png' % i
    print(
        'Имя файла:’,
        filename,
        '\tОтвет сети:’,
        model_answer(model, filename, False))
print()
print(model_answer(model, 'digit_examples/0.png'))
```

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam’,
  loss='sparse_categorical_crossentropy’,
  metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

print(model.evaluate(x_test, y_test))
```

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam’,
  loss='sparse_categorical_crossentropy’,
  metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

print(model.evaluate(x_test, y_test))
```

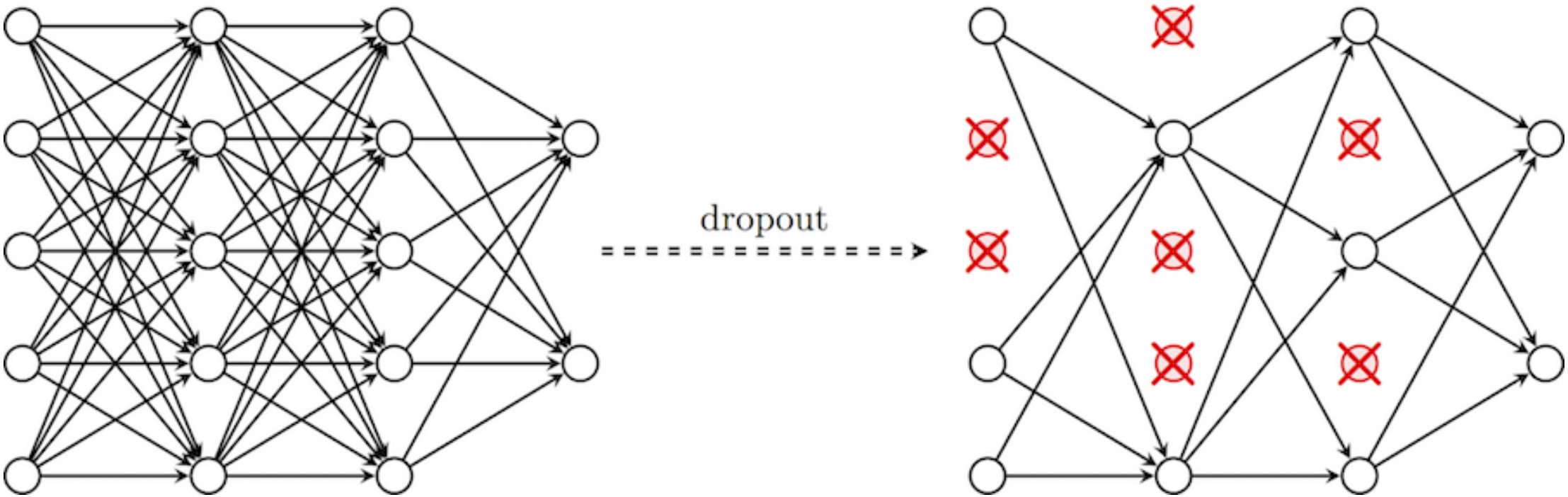
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(
    input_shape=(28, 28, 1),
    filters=16,
    kernel_size=(5, 5),
    strides=1,
    padding='same’,
    activation='relu’),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam’,
  loss='sparse_categorical_crossentropy’,
  metrics=['accuracy'])

model.fit(x_train.reshape(-1, 28, 28, 1), y_train, epochs=5)

model.evaluate(x_test.reshape(-1, 28, 28, 1), y_test)
```

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(
    input_shape=(28, 28, 1),
    filters=32,
    kernel_size=(5, 5),
    padding='same’,
    activation='relu’),
  tf.keras.layers.MaxPool2D(pool_size=[2, 2]),
  tf.keras.layers.Conv2D(
    filters=64,
    kernel_size=(5, 5),
    padding='same’,
    activation='relu’),
  tf.keras.layers.MaxPool2D(pool_size=[2, 2]),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(1024, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```


dropout

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(
    input_shape=(28, 28, 1),
    filters=32,
    kernel_size=(5, 5),
    padding='same’,
    activation='relu’),
  tf.keras.layers.MaxPool2D(pool_size=[2, 2]),
  tf.keras.layers.Conv2D(
    filters=64,
    kernel_size=(5, 5),
    padding='same’,
    activation='relu’),
  tf.keras.layers.MaxPool2D(pool_size=[2, 2]),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(1024, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.4),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```