Свёрточные сети

Одномерная свёртка

In
```
def convolve(sequence, weights):
    convolution = np.zeros(len(sequence) - len(weights) + 1)
    for i in range(convolution.shape[0]):
        convolution[i] = np.sum(weights * sequence[i:i + len(weights)])
```

Двумерный свёрточный слой в keras

In
```
# filters — количество фильтров, которому равна величина выходного тензора.
# kernel_size — пространственный размер фильтра K
# strides — размер шага свёртки (по умолчанию 1)
# padding — толщина отбивки из нулей.
# Есть два типа паддинга: valid и same.
# Тип паддинга по умолчанию — valid, то есть нулевой.
# Тип same означает автоматический подбор величины паддинга так,
# чтобы ширина и высота выходного тензора была равна ширине и высоте входного тензора.
# activation — функция активации, которая применяется сразу же после свёртки с фильтром.

keras.layers.Conv2D(filters, kernel_size, strides, padding, activation)
```

Слой сглаживания - преобразование многомерного слоя в одномерный

In
```
keras.layers.Flatten()
```

Слой пулинга

In
```
# pool_size — размер пулинга
# strides — величина шага. Если указано *None*, то шаг равен размеру пулинга.
# padding — толщина отбивки из нулей.

keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid', ...)
keras.layers.AveragePooling2D(pool_size=(2, 2), strides=None, padding='valid', ...)
```

Инициализация модели архитектуры ResNet50

In
```
# input_shape — размер входного изображения
# classes=1000 — количество нейронов в последнем полносвязном слое
# weights='imagenet' — инициализация весов.
# ImageNet — название большого датасета, на котором сеть
# обучалась классифицировать изображения на 1000 классов.
# Чтобы инициализация весов была случайной, напишите weights=None.
# include_top=True — указание на то, что в конце архитектуры ResNet есть два слоя:
# GlobalAveragePooling2D и Dense. Если задать False, то этих слоёв не будет.

from tensorflow.keras.applications.resnet import ResNet50

model = ResNet50(input_shape=None,
                 classes=1000,
                 include_top=True,
                 weights='imagenet')
```

Создание своей сети на основе ResNet50

In
```
backbone = ResNet50(input_shape=(150, 150, 3),
                    weights='imagenet',
                    include_top=False)

# замораживаем ResNet50 без верхушки (опционально)
backbone.trainable = False

model = Sequential()
model.add(backbone)
model.add(GlobalAveragePooling2D())
model.add(Dense(12, activation='softmax'))
```

Применение загрузчиков данных

In
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator()

# Загрузка из папки
datagen_flow = datagen.flow_from_directory(
    # папка, в которой хранится датасет
    '/dataset/',
    # к какому размеру приводить изображения
    target_size=(150, 150),
    # размер батча
    batch_size=16,
    # в каком виде выдавать метки классов
    class_mode='sparse',
    # указываем, что это загрузчик для обучающей выборки (при необходимости)
    subset='training',
    # или для валидационной выборки (при необходимости)
    subset='validation',
    # фиксируем генератор случайных чисел (от англ. random seed)
    seed=12345)

# получение следующего батча
features, target = next(datagen_flow)

# обучение модели с помощью загрузчиков
# на полном датасете
model.fit(datagen_flow, steps_per_epoch=len(datagen_flow))

# с выделением тренировочной и валидационной выборки
model.fit(train_datagen_flow,
          validation_data=val_datagen_flow,
          steps_per_epoch=len(train_datagen_flow),
          validation_steps=len(val_datagen_flow))
```

Добавление аугментации в загрузчике данных

In
```
# horizontal_flip — горизонтальное отражение
# vertical_flip — вертикальное отражение

datagen = ImageDataGenerator(validation_split=0.25,
                             rescale=1./255,
                             horizontal_flip=True,
                             vertical_flip=True)
```

Словарь

Свёртка (convolution)
функция, которая ко всем пикселям применяет одинаковые операции, чтобы извлечь важные для классификации элементы изображения.

Свёрточная нейронная сеть (convolution neural network)
класс нейронных сетей, использующих свёрточные слои. Они выполняют большую часть вычислительной работы сети.

Свёрточный слой (convolution layer)
слой, в котором операция свёртки применяется к входным изображениям.

Фильтр (filter)
компонент свёрточного слоя, набор весов, которые применяются к изображению.

Padding
техника добавления к краям матрицы нулей (zero padding), чтобы крайние пиксели участвовали в свёртке не меньше раз, чем центральные.

Striding, или **Stride**
техника перемещения фильтра не на один пиксель, а на большее количество пикселей.

Пулинг (pooling)
уплотнение группы пикселей до одного с применением некоторого преобразования: например, вычисление максимума или среднего арифметического.

Аугментация (augmentation)
увеличение объёма данных через трансформацию существующего датасета. Причём меняется только обучающая выборка, тестовая и валидационная остаются прежними. Суть техники заключается в преобразовании исходного изображения, но с сохранением его целевого признака.