Лабораторная работа №5

Выполнил студент группы БВТ2003 Глазков Даниил

Задание

1. Ознакомиться со сверточными нейронными сетями
2. Изучить построение модели в Keras в функциональном виде
3. Изучить работу слоя разреживания (Dropout)

In [1]:
```
from keras.datasets import cifar10
from keras.models import Model
from keras.layers import Input, Convolution2D, MaxPooling2D, Dense, Dropout, Fla
from tensorflow.python.keras.utils import np_utils
import numpy as np
```

```
WARNING:tensorflow:From C:\Users\loprz\AppData\Local\Programs\Python\Python311\Li
b\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross
_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entro
py instead.
```

In [2]:
```
batch_size = 32
num_epochs = 200
kernel_size = 3 # использование ядра 3x3
pool_size = 2 #   использование объединения 2x2
conv_depth_1 = 32 #  32 ядра на слой преобразования
conv_depth_2 = 64 # переход на 64 после первого уровня объединения
drop_prob_1 = 0.25 #  вероятность 0,25
drop_prob_2 = 0.5 #  в плотном слое вероятность 0,5
hidden_size = 512 # в плотном слое 512 нейронов

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)


inp = Input(shape=(depth, height, width)) # N.B. глубина занимает первое место в
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(inp) #устарело
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)  #Для предотвращения переобучения
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
```

```
padding='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size),padding='same')(conv_4) #
#размер исходного изображения, рисунок дополняется нулями по краям.
drop_2 = Dropout(drop_prob_1)(pool_2)
#Теперь выровняйте до 1D, примените Плотный -> ReLU (с выпадением) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)

drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out) # Чтобы определить модель, просто укажите
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, verbose=0,
model.evaluate(X_test, Y_test, verbose=1)
```

```
WARNING:tensorflow:From C:\Users\loprz\AppData\Local\Programs\Python\Python311\Li
b\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_
functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functi
ons instead.

WARNING:tensorflow:From C:\Users\loprz\AppData\Local\Programs\Python\Python311\Li
b\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max
_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\loprz\AppData\Local\Programs\Python\Python311\Li
b\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer
is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\loprz\AppData\Local\Programs\Python\Python311\Li
b\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorV
alue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\loprz\AppData\Local\Programs\Python\Python311\Li
b\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_e
agerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly
_outside_functions instead.

313/313 [==============================] - 1s 2ms/step - loss: 443.9073 - accurac
y: 0.2014
```

Out[2]: [443.9073486328125, 0.2013999968767166]

Без слоя DropOut

In [3]:
```
batch_size = 32
num_epochs = 200
kernel_size = 3 # использование ядра 3x3
pool_size = 2 #   использование объединения 2x2
conv_depth_1 = 32 #  32 ядра на слой преобразования
conv_depth_2 = 64 # переход на 64 после первого уровня объединения
drop_prob_1 = 0.25 #  вероятность 0,25
drop_prob_2 = 0.5 #  в плотном слое вероятность 0,5
hidden_size = 512 # в плотном слое 512 нейронов

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
```

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)


inp = Input(shape=(depth, height, width)) # N.B. глубина занимает первое место б
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(inp) #устарело
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
#drop_1 = Dropout(drop_prob_1)(pool_1)  #Для предотвращения переобучения
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(pool_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size),padding='same')(conv_4) #
#размер исходного изображения, рисунок дополняется нулями по краям.
#drop_2 = Dropout(drop_prob_1)(pool_2)
#Теперь выровняйте до 1D, примените Плотный -> ReLU (с выпадением) -> softmax
flat = Flatten()(pool_2)
hidden = Dense(hidden_size, activation='relu')(flat)

#drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(hidden)
model = Model(inputs=inp, outputs=out) # Чтобы определить модель, просто укажите
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, verbose=0,
model.evaluate(X_test, Y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 1935.6566 - accura
cy: 0.2774
```

Out[3]:
```
[1935.6566162109375, 0.2773999869823456]
```

Измененное количество ядер в сверточных слоях

In [4]:
```
batch_size = 32
num_epochs = 200
kernel_size = 3
pool_size = 2
conv_depth_1 = 16
conv_depth_2 = 32
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
```

```
X_test /= np.max(X_train)

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)


inp = Input(shape=(depth, height, width)) # N.B. глубина занимает первое место б
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(inp) #устарело
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)  #Для предотвращения переобучения
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size),padding='same')(conv_4) #
#размер исходного изображения, рисунок дополняется нулями по краям.
drop_2 = Dropout(drop_prob_1)(pool_2)
#Теперь выровняйте до 1D, примените Плотный -> ReLU (с выпадением) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)

drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out) # Чтобы определить модель, просто укажите
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, verbose=0,
model.evaluate(X_test, Y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 278.0039 - accurac
y: 0.2201
```

Out[4]: `[278.00390625, 0.22010000050067902]`

In [5]:
```
batch_size = 32
num_epochs = 200
kernel_size = 3 # использование ядра 3x3
pool_size = 2 #   использование объединения 2x2
conv_depth_1 = 128 #  32 ядра на слой преобразования
conv_depth_2 = 256 # переход на 64 после первого уровня объединения
drop_prob_1 = 0.25 #  вероятность 0,25
drop_prob_2 = 0.5 #  в плотном слое вероятность 0,5
hidden_size = 512 # в плотном слое 512 нейронов

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)
```

```
inp = Input(shape=(depth, height, width)) # N.B. глубина занимает первое место б
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(inp) #устарело
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)  #Для предотвращения переобучения
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size),padding='same')(conv_4) #
#размер исходного изображения, рисунок дополняется нулями по краям.
drop_2 = Dropout(drop_prob_1)(pool_2)
#Теперь выровняйте до 1D, примените Плотный -> ReLU (с выпадением) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)

drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out) # Чтобы определить модель, просто укажите
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, verbose=0,
model.evaluate(X_test, Y_test, verbose=1)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 703.2024 - accurac
y: 0.2361
```

Out[5]: [703.202392578125, 0.2361000031232834]

Изменение размера ядра в сверточных слоях

In [6]:
```
batch_size = 32
num_epochs = 200
kernel_size = 1 # использование ядра 1x1
pool_size = 2 #   использование объединения 2x2
conv_depth_1 = 32 #  32 ядра на слой преобразования
conv_depth_2 = 64 # переход на 64 после первого уровня объединения
drop_prob_1 = 0.25 #  вероятность 0,25
drop_prob_2 = 0.5 #  в плотном слое вероятность 0,5
hidden_size = 512 # в плотном слое 512 нейронов

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)


inp = Input(shape=(depth, height, width)) # N.B. глубина занимает первое место б
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
```

```
padding='same', activation='relu')(inp) #устарело
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)  #Для предотвращения переобучения
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size),padding='same')(conv_4) #
#размер исходного изображения, рисунок дополняется нулями по краям.
drop_2 = Dropout(drop_prob_1)(pool_2)
#Теперь выровняйте до 1D, примените Плотный -> ReLU (с выпадением) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)

drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out) # Чтобы определить модель, просто укажите
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, verbose=0,
model.evaluate(X_test, Y_test, verbose=1)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 5683.3452 - accura
cy: 0.1068
```

Out[6]:
```
[5683.34521484375, 0.10679999738931656]
```

In [7]:
```
batch_size = 32
num_epochs = 200
kernel_size = 5 # использование ядра 5x5
pool_size = 2 #   использование объединения 2x2
conv_depth_1 = 32 #  32 ядра на слой преобразования
conv_depth_2 = 64 # переход на 64 после первого уровня объединения
drop_prob_1 = 0.25 #  вероятность 0,25
drop_prob_2 = 0.5 #  в плотном слое вероятность 0,5
hidden_size = 512 # в плотном слое 512 нейронов

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)

Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)


inp = Input(shape=(depth, height, width)) # N.B. глубина занимает первое место в
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(inp) #устарело
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)  #Для предотвращения переобучения
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
```

```
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
padding='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size),padding='same')(conv_4) #
#размер исходного изображения, рисунок дополняется нулями по краям.
drop_2 = Dropout(drop_prob_1)(pool_2)
#Теперь выровняйте до 1D, примените Плотный -> ReLU (с выпадением) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)

drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out) # Чтобы определить модель, просто укажите
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.fit(X_train, Y_train, batch_size=batch_size, epochs=num_epochs, verbose=0,
model.evaluate(X_test, Y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 627.9857 - accurac
y: 0.2883
```

Out[7]: [627.9856567382812, 0.2883000075817108]

In [ ]: