Введение в нейронные сети

Урок 4. Сверточные нейронные сети

ДЗ - поиграться с кодом соревнования по Северстали, получить как можно лучший результат.
Альтернативное ДЗ - рассмотреть постановку и решение аналогичного индустриального кейса.
Сдавать как обычно блокнот.

Взяла пример из методички.

Для обучения нейросети будем использовать датасет cifar-10.

Изначальные данные по data augmentation:

```
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.,
        zoom_range=0.2,
        channel_shift_range=0.,
        fill_mode='nearest',
        cval=0.,
        horizontal_flip=True,
        vertical_flip=False,
        rescale=None,
        preprocessing_function=None,
        data_format=None,
        validation_split=0.0)


    train_gen = datagen.flow(X_train,
                             y_train,
                             batch_size=batch_size)

    # запуск data augmentation через fit
    model.fit(train_gen,
              epochs=epochs,
              validation_data=(X_test, y_test))
```

```
Использование data augmentation
Epoch 1/5
98/98 [==============================] - 35s 288ms/step - loss: 2.3024 - accuracy: 0.1091 - val_loss: 2.2972 - val_accuracy: 0.1570
Epoch 2/5
98/98 [==============================] - 27s 276ms/step - loss: 2.2964 - accuracy: 0.1190 - val_loss: 2.2900 - val_accuracy: 0.1677
Epoch 3/5
98/98 [==============================] - 28s 284ms/step - loss: 2.2852 - accuracy: 0.1365 - val_loss: 2.2711 - val_accuracy: 0.1658
Epoch 4/5
98/98 [==============================] - 27s 270ms/step - loss: 2.2564 - accuracy: 0.1560 - val_loss: 2.2133 - val_accuracy: 0.1962
Epoch 5/5
98/98 [==============================] - 27s 278ms/step - loss: 2.1806 - accuracy: 0.1859 - val_loss: 2.0936 - val_accuracy: 0.2409
```

Поменяем количество эпох и параметры сдвига:

```
else:
    print('Использование data augmentation')
    # Препроцессинг и data augmentation в реальном времени:
    datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        zca_epsilon=1e-06,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        channel_shift_range=0.,
        fill_mode='nearest',
        cval=0.,
        horizontal_flip=True,
        vertical_flip=False,
        rescale=None,
        preprocessing_function=None,
        data_format=None,
        validation_split=0.0)
```

Точность повысилась до 0,28 на 9 эпохе и упала до 0,25 на последней эпохе.

```
            validation_data=(X_test, y_test))

Использование data augmentation
Epoch 1/10
98/98 [==============================] - 29s 283ms/step - loss: 2.3062 - accuracy: 0.1023 - val_loss: 2.2968 - val_accuracy: 0.1157
Epoch 2/10
98/98 [==============================] - 27s 276ms/step - loss: 2.2960 - accuracy: 0.1178 - val_loss: 2.2872 - val_accuracy: 0.1424
Epoch 3/10
98/98 [==============================] - 28s 283ms/step - loss: 2.2838 - accuracy: 0.1399 - val_loss: 2.2644 - val_accuracy: 0.1700
Epoch 4/10
98/98 [==============================] - 28s 282ms/step - loss: 2.2507 - accuracy: 0.1617 - val_loss: 2.2004 - val_accuracy: 0.2127
Epoch 5/10
98/98 [==============================] - 28s 284ms/step - loss: 2.1762 - accuracy: 0.1907 - val_loss: 2.0980 - val_accuracy: 0.2300
Epoch 6/10
98/98 [==============================] - 28s 286ms/step - loss: 2.1277 - accuracy: 0.2007 - val_loss: 2.0447 - val_accuracy: 0.2477
Epoch 7/10
98/98 [==============================] - 28s 283ms/step - loss: 2.1026 - accuracy: 0.2095 - val_loss: 2.0913 - val_accuracy: 0.2272
Epoch 8/10
98/98 [==============================] - 28s 287ms/step - loss: 2.0872 - accuracy: 0.2209 - val_loss: 2.0037 - val_accuracy: 0.2742
Epoch 9/10
98/98 [==============================] - 28s 282ms/step - loss: 2.0697 - accuracy: 0.2254 - val_loss: 1.9873 - val_accuracy: 0.2809
Epoch 10/10
98/98 [==============================] - 27s 280ms/step - loss: 2.0601 - accuracy: 0.2286 - val_loss: 2.0376 - val_accuracy: 0.2515
```

Усложняю структуру слоев сети:

```
# конфигурирование слоев нейросети
model = Sequential()

# слои нейросети отвественные за свертку и max-pooling
model.add(Conv2D(32, (3, 3), padding='same', input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# полносвязные слои нейронной сети
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))

model.summary()
# компиляция модели
model.compile(loss='categorical_crossentropy',
              optimizer='SGD',
              metrics=['accuracy'])
```

```
# слои нейросети отвественные за свертку и max-pooling
model.add(Conv2D(32, (3, 3), padding='same', input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(256, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# полносвязные слои нейронной сети
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

```
Epoch 1/10
98/98 [==============================] - 31s 294ms/step - loss: 2.3034 - accuracy: 0.1063 - val_loss: 2.3007 - val_accuracy: 0.1342
Epoch 2/10
98/98 [==============================] - 28s 284ms/step - loss: 2.3014 - accuracy: 0.1093 - val_loss: 2.2987 - val_accuracy: 0.1257
Epoch 3/10
98/98 [==============================] - 28s 289ms/step - loss: 2.2989 - accuracy: 0.1131 - val_loss: 2.2955 - val_accuracy: 0.1430
Epoch 4/10
98/98 [==============================] - 28s 289ms/step - loss: 2.2943 - accuracy: 0.1251 - val_loss: 2.2880 - val_accuracy: 0.1674
Epoch 5/10
98/98 [==============================] - 29s 291ms/step - loss: 2.2819 - accuracy: 0.1405 - val_loss: 2.2668 - val_accuracy: 0.1816
Epoch 6/10
98/98 [==============================] - 28s 285ms/step - loss: 2.2444 - accuracy: 0.1575 - val_loss: 2.1999 - val_accuracy: 0.2119
Epoch 7/10
98/98 [==============================] - 29s 290ms/step - loss: 2.1755 - accuracy: 0.1774 - val_loss: 2.1741 - val_accuracy: 0.1836
Epoch 8/10
98/98 [==============================] - 28s 284ms/step - loss: 2.1412 - accuracy: 0.1884 - val_loss: 2.1171 - val_accuracy: 0.2056
Epoch 9/10
98/98 [==============================] - 28s 284ms/step - loss: 2.1258 - accuracy: 0.1960 - val_loss: 2.0660 - val_accuracy: 0.2307
Epoch 10/10
98/98 [==============================] - 28s 287ms/step - loss: 2.1119 - accuracy: 0.1996 - val_loss: 2.0614 - val_accuracy: 0.2399
```

```
[12] # # сохранение модели и весов
```

Потери уменьшились, но точность примерно на таком же уровне как при вышеуказанных параметрах.

Вывод:

Существенно повысить точность распознавание образов cifar 10 сверточной нейронной сетью не удалось. Показатели точности при разных подходах были в диапазоне 0.23 - 0.28.