МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание объектов на фотографиях»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Распознавание объектов на фотографиях (Object Recognition in Photographs)

CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).

Порядок выполнения работы.

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

Требования.

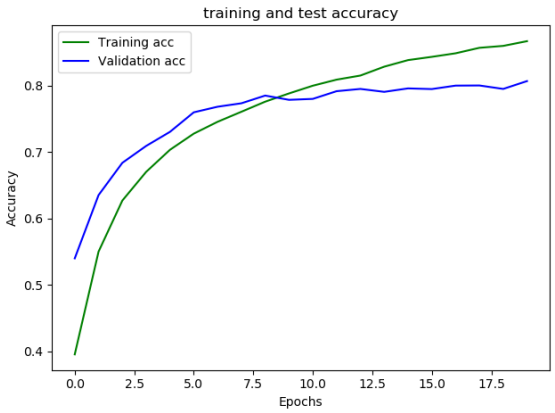
- Построить и обучить сверточную нейронную сеть
- Исследовать работу сеть без слоя Dropout
- Исследовать работу сети при разных размерах ядра свертки

Ход работы.

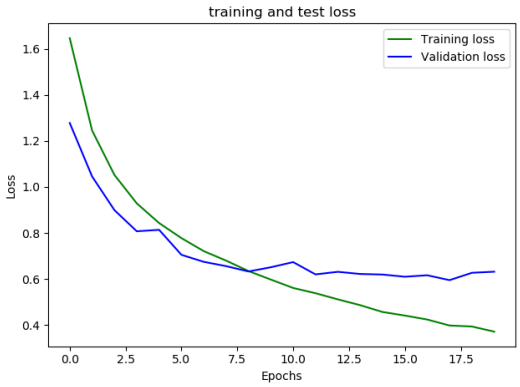
Набор данных СИФАР-10 состоит из 60'000 цветных рисунков следующих десяти классов: самолеты, легковые автомобили, птицы, кошки, олени, собаки, лягушки, лошади, корабли, грузовики; размер каждого образа – 32*32 пикселей. В обучающую выборку входят 50'000 рисунков, а 10'000 – в тестовую.

1. Была построена сверточная нейронная сеть, использующая слои maxpooling и dropout со следующей архитектурой:
    - Оптимизатор – adam
    - `batch_size=128`
    - `loss='categorical_crossentropy'`
    - `epochs=20`

      Точность ~80%



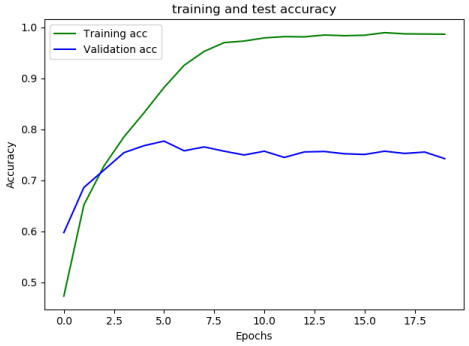
а



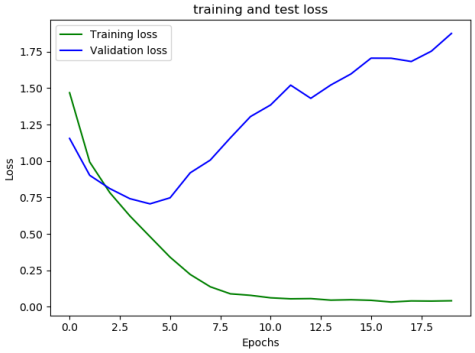
б

Рисунок 1 – Графики точности и потерь данной архитектуры

2. Исследуем работу сети без слоя Dropout:



а

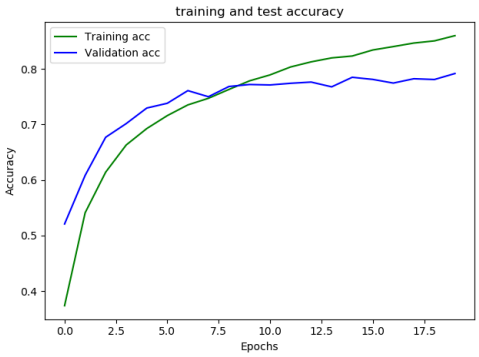


б

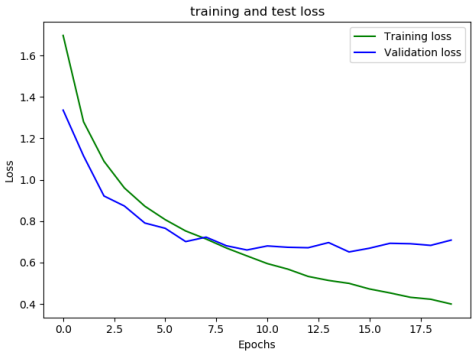
Рисунок 2 – Графики точности и потерь без слоев разреживания.

Видим, что наблюдается переобучение после ~5 эпохи. Dropout как раз используется для решения этой проблемы путем случайного исключения нейронов во время итераций.

3. Исследуем работу сети при разных размерах ядра свертки. Рассмотрим размеры 5x5 и 7x7:

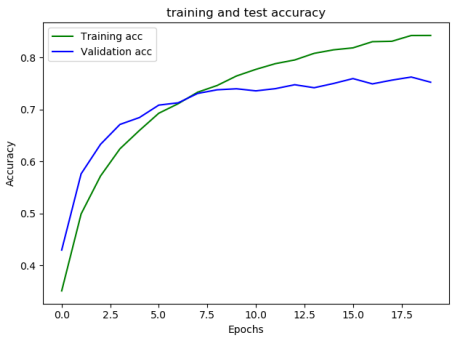


а

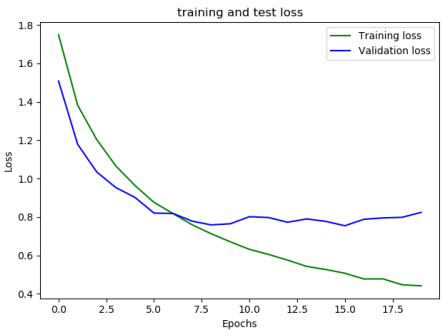


б

Рисунок 3 – Графики точности и потерь с размером ядра 5x5.



а



б

Рисунок 4 – Графики точности и потерь с размером ядра 7x7.

Как видим, при увеличении размера ядра переобучение начинает возникать немного раньше, а точность уменьшается.

Выводы.

В ходе выполнения данной работы ознакомились со сверточными нейронными сетями и на их основе получили представление о распознавании объектов на фотографиях. Было исследовано влияние слоя разреживания и размера ядра свертки на нейронную сеть.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
      from keras.datasets import cifar10
      from keras.models import Model
      from keras.layers import Input, Convolution2D, MaxPooling2D,
Dense, Dropout, Flatten
      from keras.utils import np_utils
      import numpy as np
      import matplotlib.pyplot as plt

      batch_size = 128  # in each iteration, we consider 32 training
examples at once
      num_epochs = 20  # we iterate 20 times over the entire training
set
      kernel_size = 3  # we will use 3x3 kernels throughout
      pool_size = 2  # we will use 2x2 pooling throughout
      conv_depth_1 = 32  # we will initially have 32 kernels per conv.
layer...
      conv_depth_2 = 64  # ...switching to 64 after the first pooling
layer
      drop_prob_1 = 0.25  # dropout after pooling with probability 0.25
      drop_prob_2 = 0.5  # dropout in the dense layer with probability
0.5
      hidden_size = 512  # the dense layer will have 512 neurons

      (X_train, y_train), (X_test, y_test) = cifar10.load_data()  #
fetch CIFAR-10 data
      num_train, depth, height, width = X_train.shape  # there are
50000 training examples in CIFAR-10
      num_test = X_test.shape[0]  # there are 10000 test examples in
CIFAR-10
      num_classes = np.unique(y_train).shape[0]  # there are 10 image
classes

      X_train = X_train.astype('float32')
      X_test = X_test.astype('float32')
      X_train /= np.max(X_train)  # Normalise data to [0, 1] range
      X_test /= np.max(X_train)  # Normalise data to [0, 1] range
      Y_train = np_utils.to_categorical(y_train, num_classes)  # One-
hot encode the labels
      Y_test = np_utils.to_categorical(y_test, num_classes)  # One-hot
encode the labels
```

```
      inp = Input(shape=(depth, height, width))  # N.B. depth goes
first in Keras
      # Conv [32] -> Conv [32] -> Pool (with dropout on the pooling
layer)
      conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
border_mode='same', activation='relu')(inp)
      conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
border_mode='same', activation='relu')(conv_1)
      pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
      drop_1 = Dropout(drop_prob_1)(pool_1)
      # Conv [64] -> Conv [64] -> Pool (with dropout on the pooling
layer)
      conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(drop_1)
      conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(conv_3)
      pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
      drop_2 = Dropout(drop_prob_1)(pool_2)
      # Now flatten to 1D, apply Dense -> ReLU (with dropout) ->
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(input=inp, output=out)  # To define a model, just
specify its input and output layers
      model.compile(loss='categorical_crossentropy',  # using the
cross-entropy loss function
                    optimizer='adam',  # using the Adam optimiser
                    metrics=['accuracy'])  # reporting the accuracy
      h = model.fit(X_train, Y_train,  # Train the model using the
training set...
                    batch_size=batch_size, nb_epoch=num_epochs,
                    verbose=1, validation_split=0.1)  # ...holding out
10% of the data for validation
      score = model.evaluate(X_test, Y_test, verbose=1)  # Evaluate the
trained model on the test set!
      print('Test loss:', score[0])
      print('Test accuracy:', score[1])

      plt.title("training and test accuracy")
      plt.plot(h.history['accuracy'], 'g', label='Training acc')
      plt.plot(h.history['val_accuracy'], 'b', label='Validation acc')
      plt.xlabel('Epochs')
      plt.ylabel('Accuracy')
```

```
plt.legend()
plt.show()

plt.clf()
plt.title("training and test loss")
plt.plot(h.history['loss'], 'g', label='Training loss')
plt.plot(h.history['val_loss'], 'b', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()
```