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»

Студентка гр. 7381 _____ Судакова П.С.

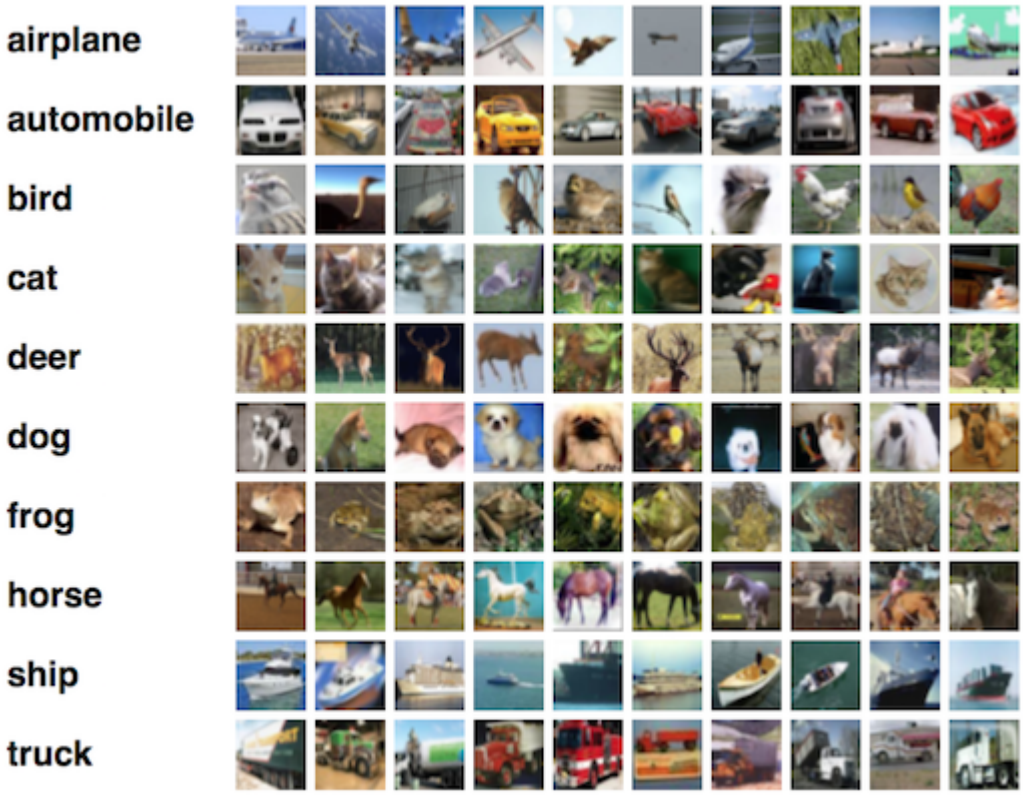
Преподаватель _____ Жукова Н.А.

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).



Набор данных содержит 60,000 изображений для обучения и 10,000 изображений для тестирования.

Порядок выполнения работы.

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

Требования.

1. Построить и обучить сверточную нейронную сеть
2. Исследовать работу сеть без слоя Dropout
3. Исследовать работу сети при разных размерах ядра свертки

Ход работы.

Построим сверточную нейронную сеть со слоем разреживания и обучим ее на 10 эпохах и размером батча 70. Результат точности и ошибок приведены на рис. 1- 2.

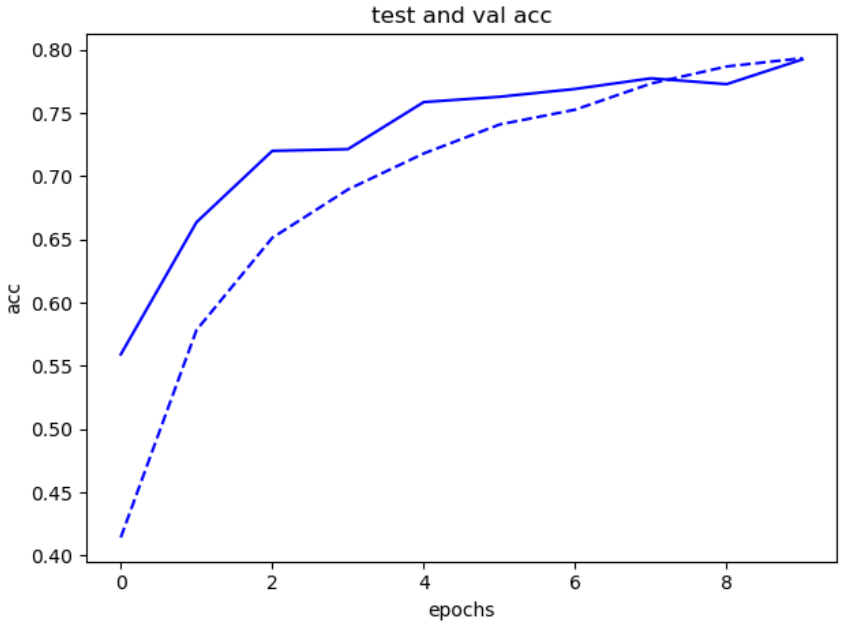


Рисунок 1 – График точности модели с ядром 3x3 с dropout

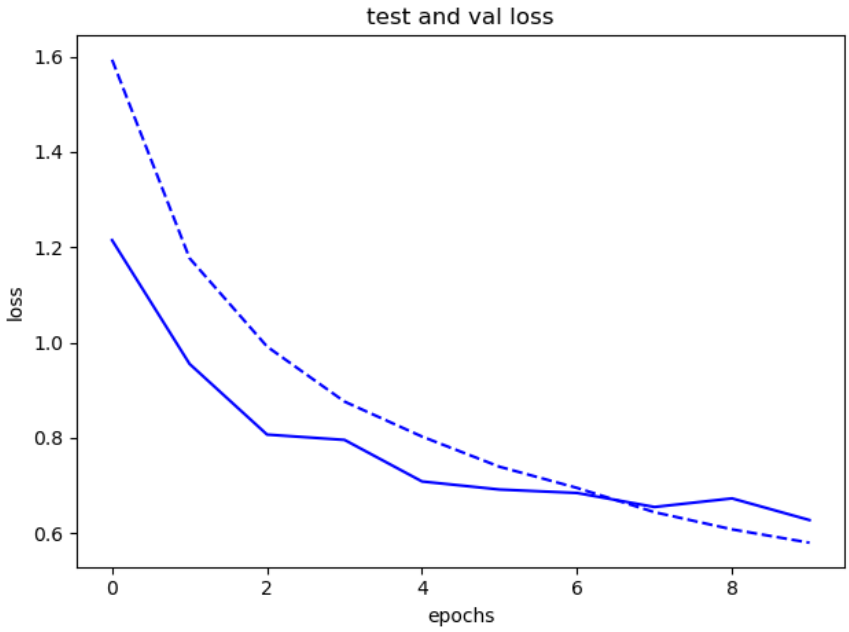


Рисунок 2 – График ошибок модели с ядром 3x3 с dropout

Переобучение не наблюдается.

Построим сверточную нейронную сеть без слоя разреживания и обучим ее на 10 эпохах и размером батча 70. Результат точности и ошибок приведены на рис. 3- 4.

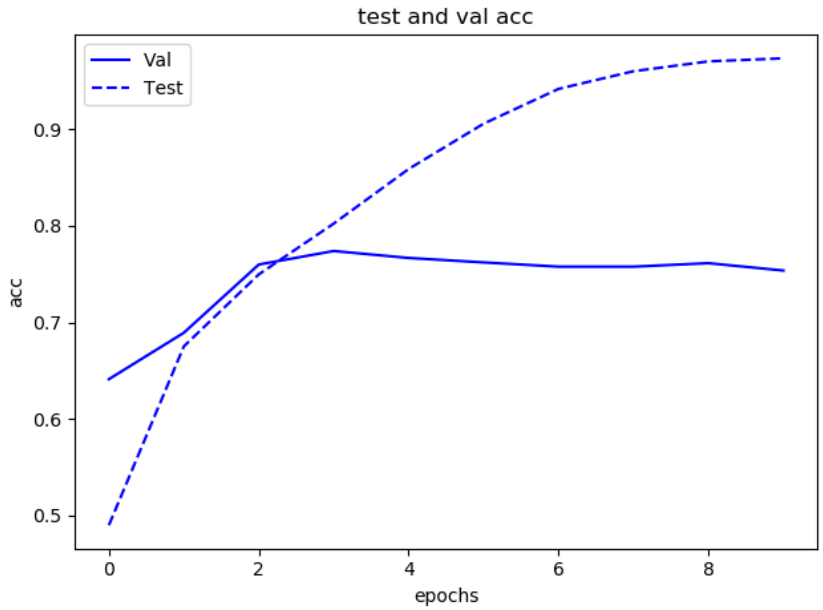


Рисунок 3 – График точности модели с ядром 3x3 без dropout

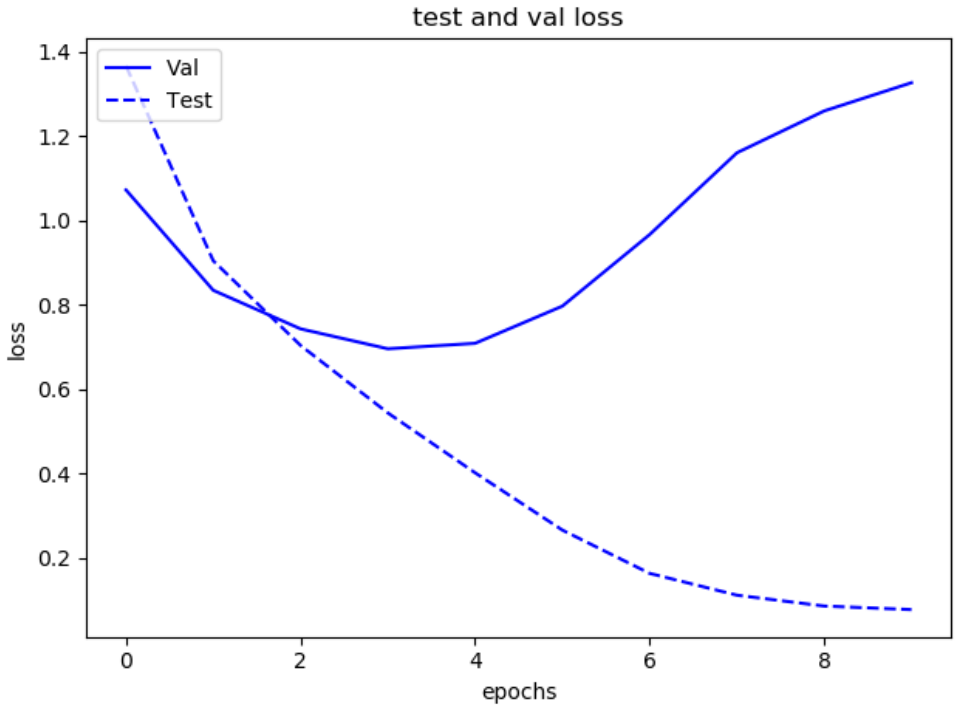


Рисунок 4 – График ошибок модели с ядром 3x3 без dropout

Наблюдаем переобучение на 3 эпохе.

Обучим модель с размерами ядра свертки 5x5 с dropout. Результат точности и ошибок приведены на рис. 5- 6.

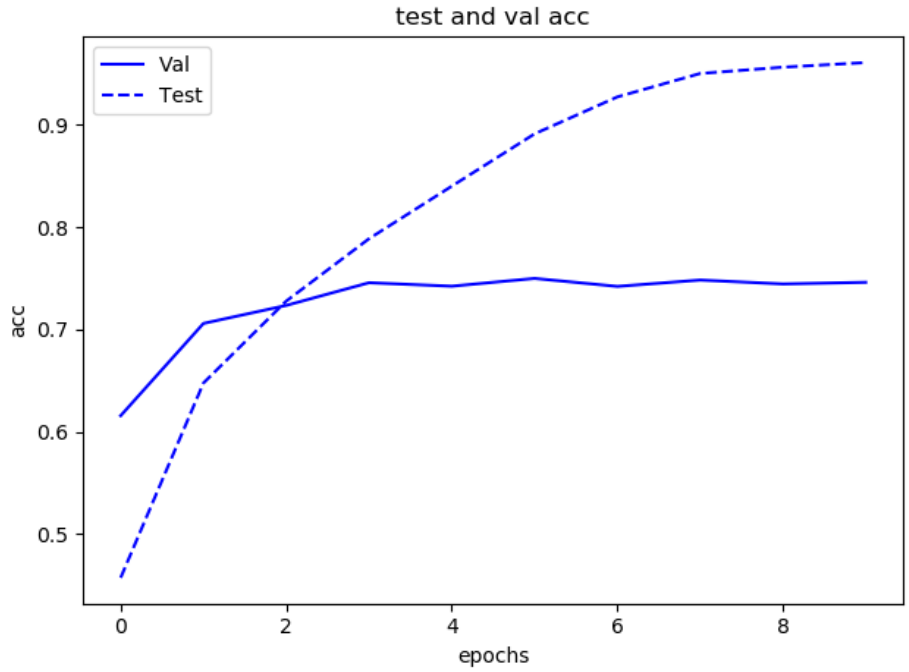


Рисунок 5 – График точности модели с ядром 5x5 с dropout

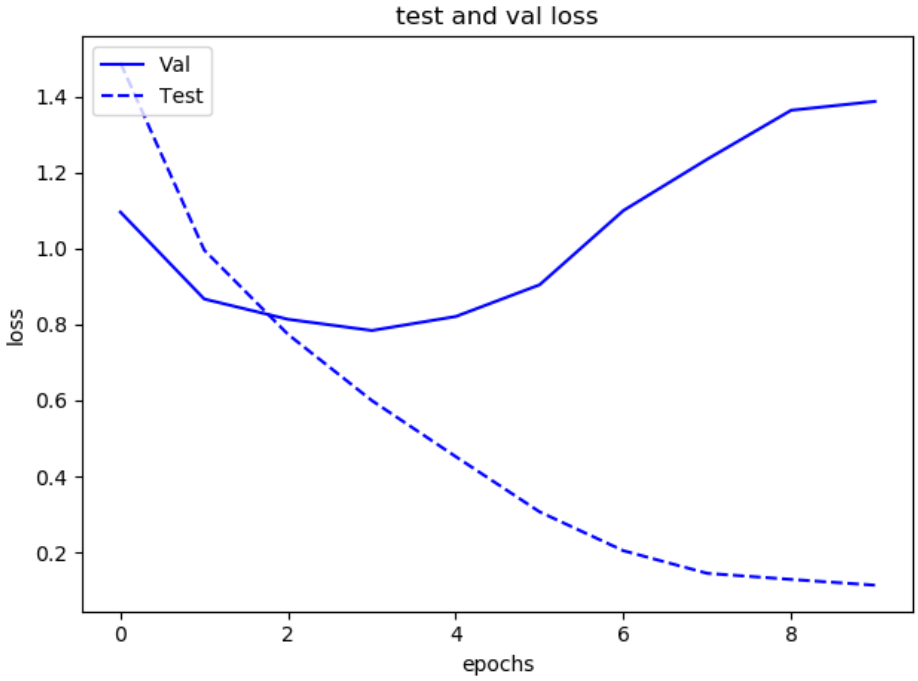


Рисунок 6 – График ошибок модели с ядром 5x5 с dropout

Выводы.

Было изучено влияние слоя разреживания(dropout) на обучение нейронной сети. Было доказано, что этот слой помогает бороться с переобучением. Также рассмотрели, как меняется результат обучения при изменении ядра свертки.

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
from matplotlib import pyplot as plt

batch_size = 70 # in each iteration, we consider 32 training
examples at once
num_epochs = 10 # we iterate 200 times over the entire training
set
kernel_size = 5 # 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')(pool_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()(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(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, epochs=num_epochs,
          verbose=1, validation_split=0.1) # ...holding out 10%
of the data for validation
print(model.evaluate(X_test, Y_test, verbose=1)) # Evaluate the
trained model on the test set!



plt.plot(range(num_epochs),h.history['val_accuracy'],'b-',label=
'val')
plt.plot(range(num_epochs),h.history['accuracy'],'b--',label='te
st')
plt.title('test and val acc')
plt.xlabel('epochs')
plt.ylabel('acc')
plt.legend(['Val', 'Test'], loc='upper left')
```

```
plt.show()

plt.plot(range(num_epochs),h.history['val_loss'],'b-',label='val
')
plt.plot(range(num_epochs),h.history['loss'],'b--',label='test')
plt.title('test and val loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['Val', 'Test'], loc='upper left')
plt.show()
```