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: «Распознование объектов на фотографии»

Студент гр. 7381 _____ Кортев Ю. В.

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.

В листинге 1 показана инициализация модели.

```
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),
padding='same', activation='relu')(inp)
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)
# 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))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)
# Now flatten to 1D, apply FC -> 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(inp, 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
```

Листинг 1 - Инициализация модели

В листинге 2 показаны начальные параметры модели.

```
batch_size = 64 # in each iteration, we consider 32 training examples
at once
num_epochs = 40 # we iterate 200 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 FC layer with probability 0.5
hidden_size = 512 # the FC layer will have 512 neurons
```

Листинг 2 - Начальные параметры модели

На рисунках 1-2 показаны графики точности и потери модели.

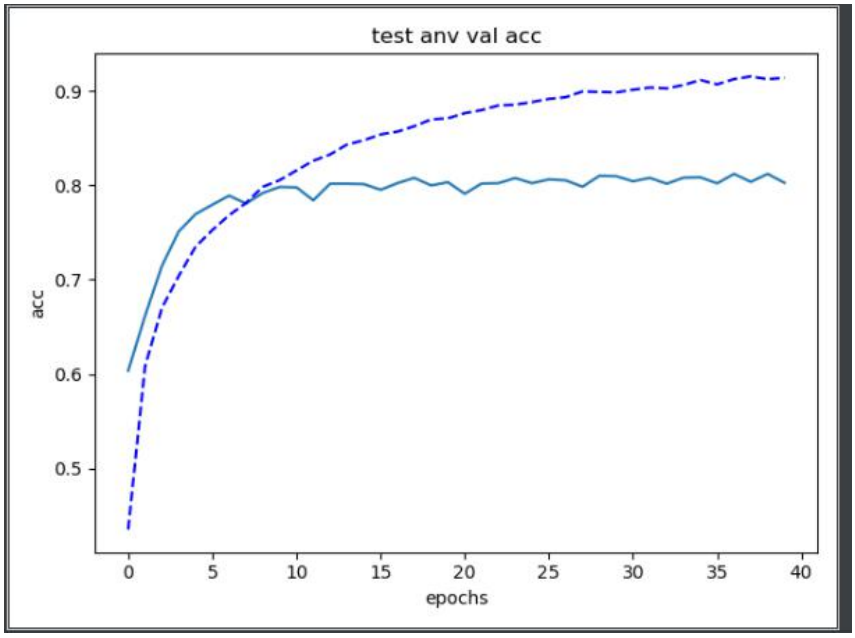

Рисунок 1 - Точность модели с ядром 3x3 и прореживанием

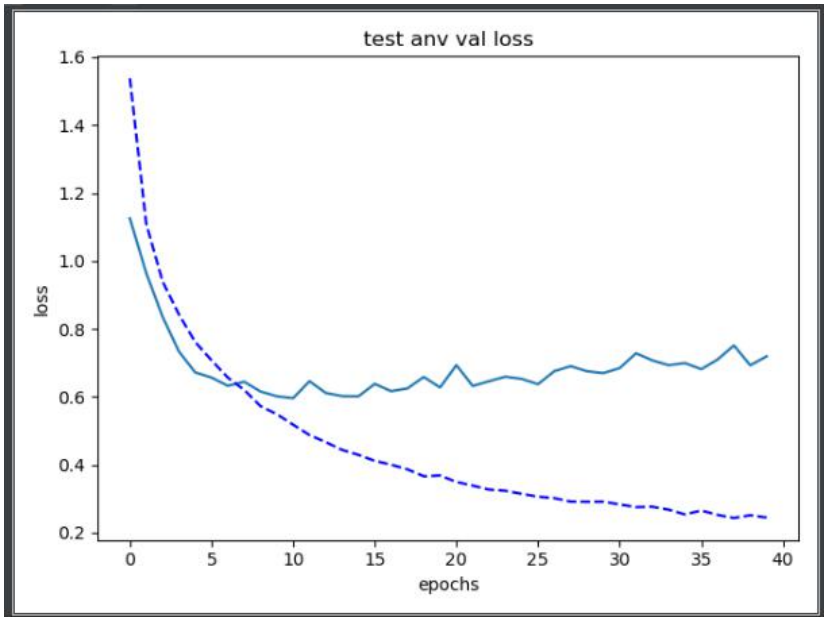

Рисунок 2 - Потери модели с ядром 3x3 и прореживанием

Теперь для того, чтобы исследовать работу сети без прореживания, уберу из модели слои Dropout. Графики полученной точности и потерь показаны на рисунках 3-4.

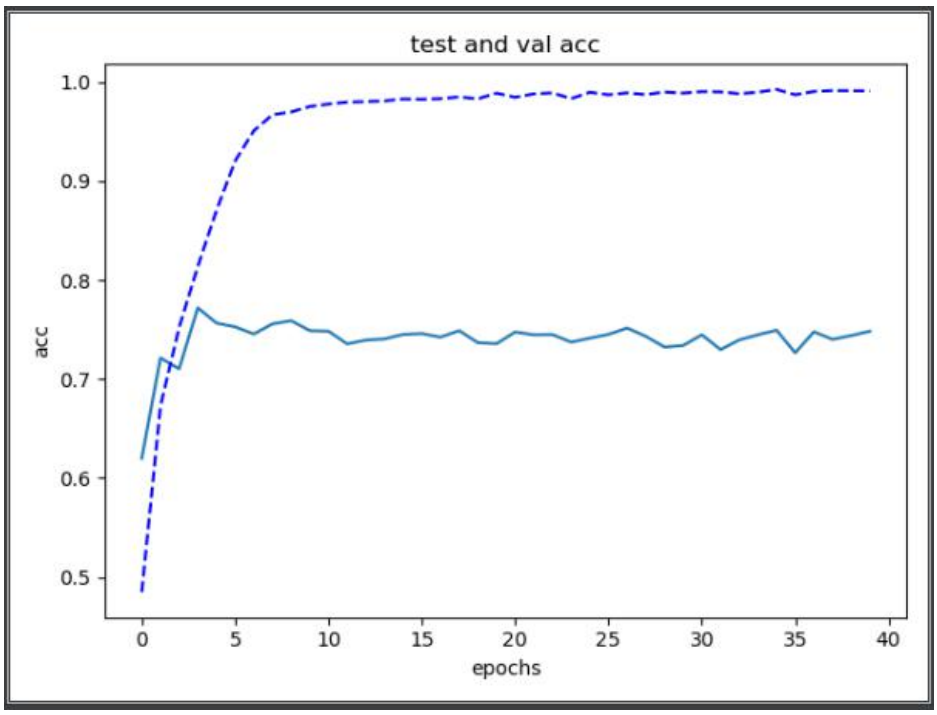

Рисунок 3 - Точность модели с ядром 3x3 и без прореживания

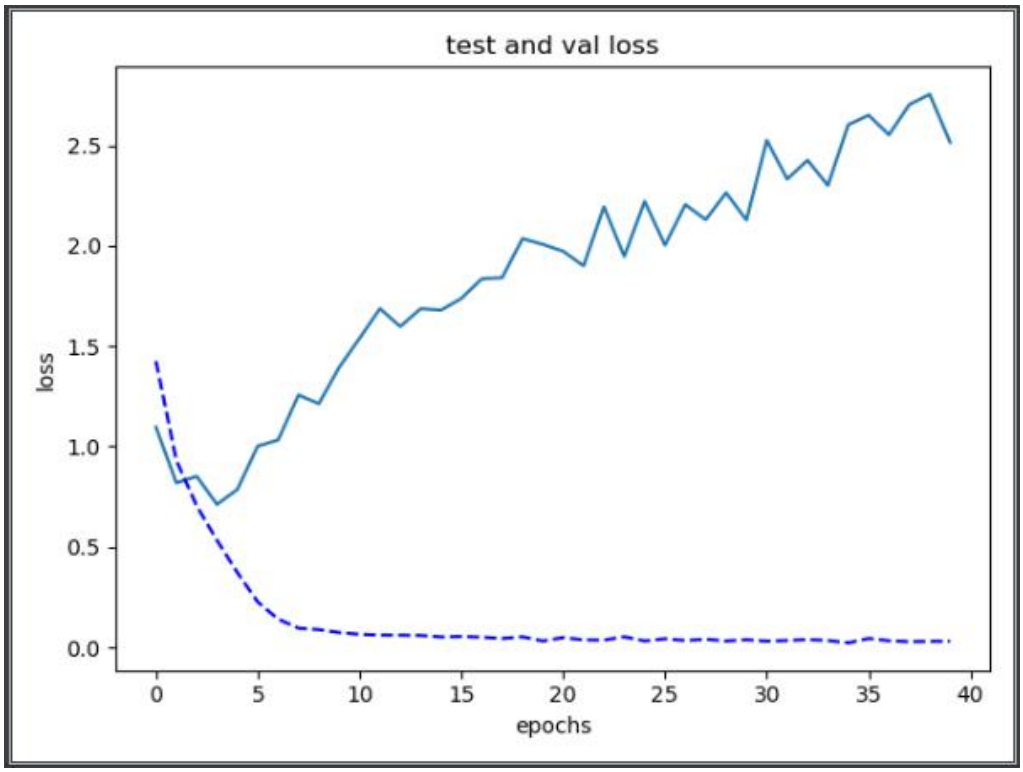

Рисунок 4 - Потери модели с ядром 3x3 и без прореживания

Как видно новая модель гораздо меньше устойчива к переобучению.

Вернусь к первоначальной архитектуры и изменю размерность ядра свертки. Размер ядра обычно берут в пределах от 3x3 до 7x7. Если размер ядра маленький, то оно не сможет выделить какие-либо признаки, если слишком большое, то увеличивается количество связей между нейронами. Установлю размерность 5x5 и сравню результат с первоначальной архитектурой. Графики точности и потерь показаны на рисунках 5-6.

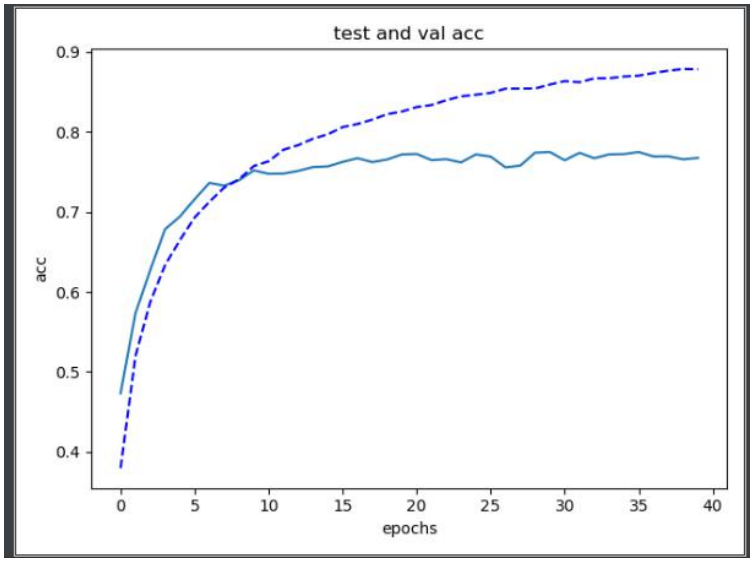

Рисунок 5 - Точность модели с ядром 5x5

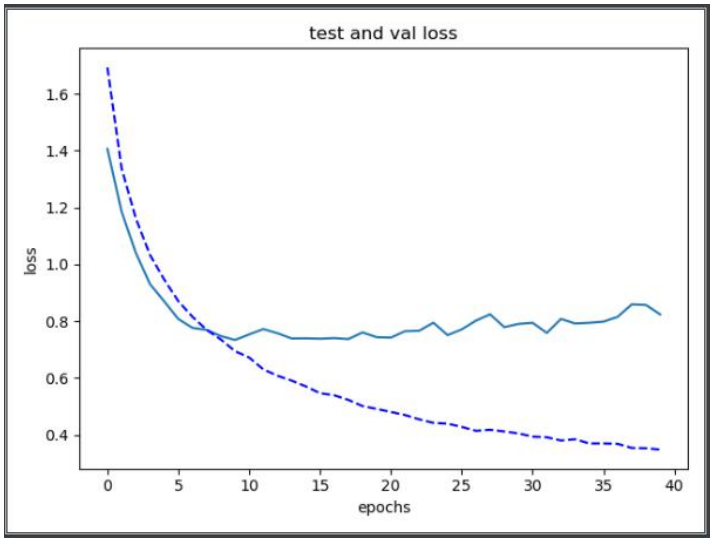

Рисунок 6 - Потери модели с ядром 5x5

Как видно максимум точности новой модели меньше, чем предыдущей. Делаю вывод, что начальные параметры и архитектура являются лучшими для данной задачи. По графику потерь на рисунке 2, делаю вывод, что итоговую модели необходимо обучить за 10 эпох.

Выводы.

В ходе выполнения лабораторной работы исследовано влияние прореживания в работе CNN, а также размерности ядра свертки, на процесс обучения. Построена и обучена модель распознования объектов на фотографиях.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras.layers import Input, Convolution2D, MaxPooling2D, Dropout,
Flatten, Dense
from keras.models import Model
from matplotlib import pyplot as plt
from keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder

from keras.datasets import cifar10

(X_train,y_train),(X_test,y_test)=cifar10.load_data()


batch_size = 64 # in each iteration, we consider 32 training examples
at once
num_epochs = 9 # we iterate 200 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 FC layer with probability 0.5
hidden_size = 512 # the FC layer will have 512 neurons


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 = to_categorical(y_train, num_classes) # One-hot encode the
labels
Y_test = 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),
padding='same', activation='relu')(inp)
```

```
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)
# 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))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)
# Now flatten to 1D, apply FC -> 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(inp, 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='test')
plt.title('test and val acc')
plt.xlabel('epochs')
plt.ylabel('acc')
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.show()
```