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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студент гр. 8383 _____ Дейнега В. Е.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2021

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).

Задание.

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)
- Построить и обучить сверточную нейронную сеть
- Исследовать работу сеть без слоя Dropout
- Исследовать работу сети при разных размерах ядра свертки

Выполнение работы

1) Создадим модель ИНС.

Была реализована модель ИНС из методических указаний.

```
inp=Input(shape=(depth,height,width))
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_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)
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)
```

Модель была обучена на 10 эпохах в связи с нехваткой ресурсов пк для выставления большего количества эпох. Точность на обучаемых данных составила 0.783, на валидационных - 0.784, на тестовых - 0.5193, графики точности и потерь представлены на рис. 1.

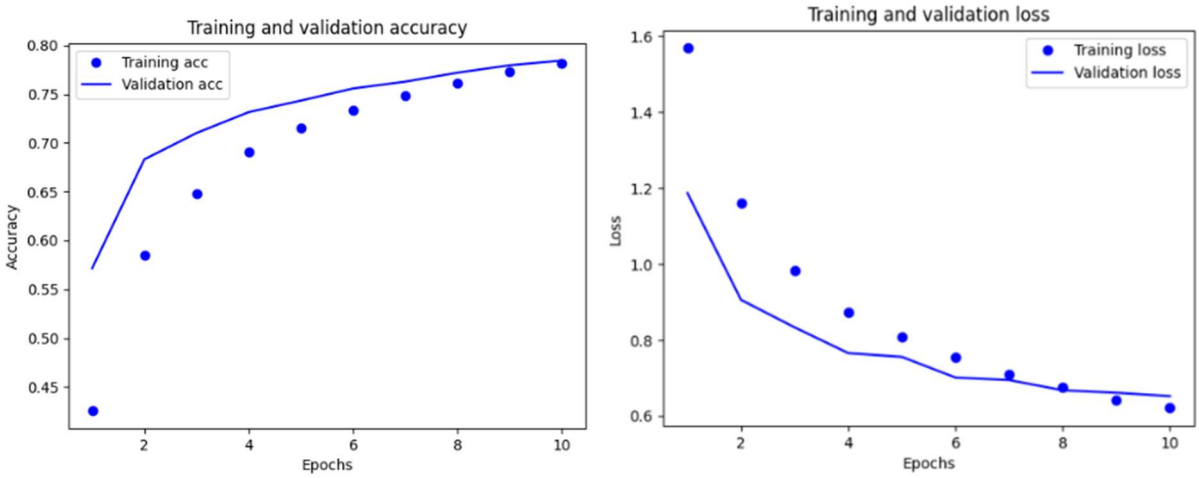

Рисунок 1 – Точность и потери ИНС 1.

2) Исследуем работу сети без слоя Dropout.

На обучаемых данных точность значительно выросла и составила 0.9738, на валидационных - 0.7338, на тестовых 0.5911, графики точности и потерь представлены на рис. 2.

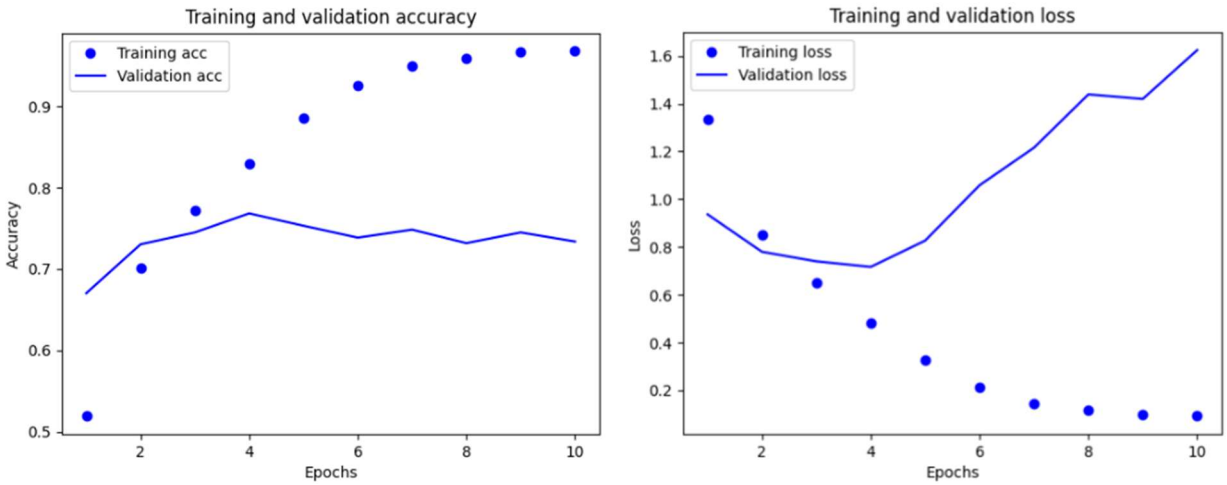

Рисунок 2 – Точность и потери ИНС без слоя Dropout.

На графике потерь хорошо заметно расхождение графиков, что свидетельствует о переобучении модели. Вернем модель к исходной конфигурации.

3) Протестируем разные размеры ядра свертки.

Уменьшим размер ядра до 2x2.

Точность на обучаемых данных составила 0.7785, на валидационных - 0.7596, на тестовых - 0.4307, графики точности и потерь представлены на рис. 3.

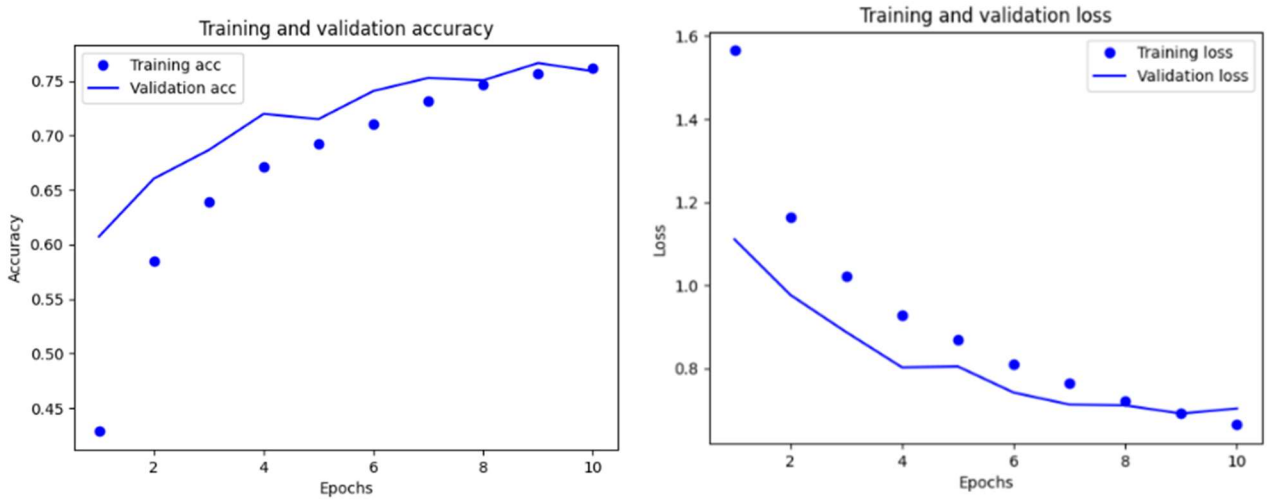

Рисунок 3 – Точность и потери ИНС с размером ядра свертки 3.

Точность модели на валидационных и обучаемых данных уменьшилось, однако эти изменния не критичны. Точность же на тестовых данных катострафически упала, что делает применение модели невозможным.

Протестируем модель ИНС с размером ядра свертки 4x4

Точность на обучаемых данных составила 0.7686, на валидационных - 0.7544, на тестовых - 0.5183, графики точности и потерь представлены на рис. 4.

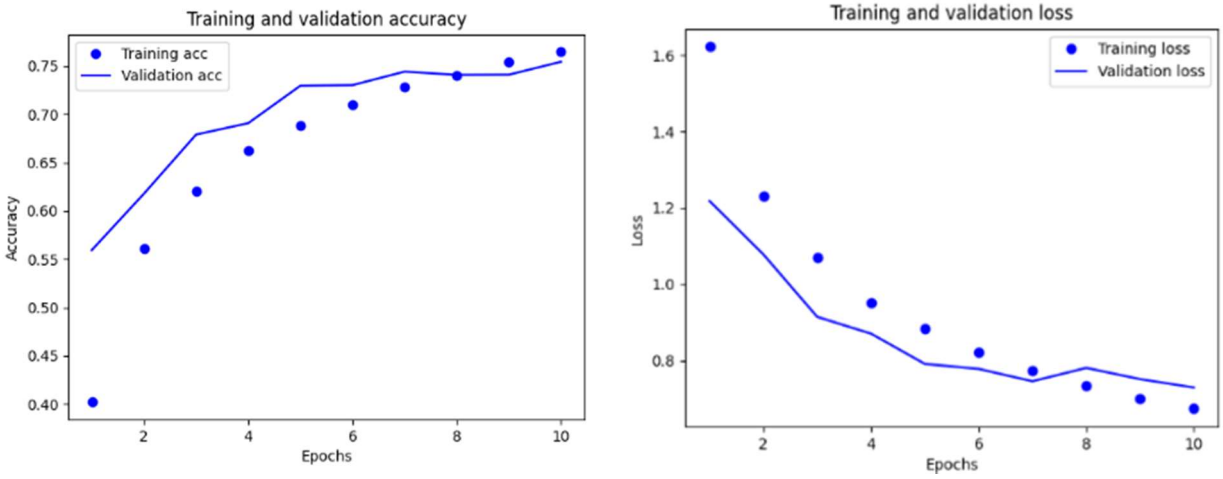

Рисунок 4 – Точность и потери ИНС с размером ядра свертки 4.

Точность этой модели на обучаемых, валидационных данных незначительно уменьшилась, однако эмпирически время, потраченное на обучение сети возросло и составила 31.8 мин.

Протестируем модель ИНС с размером ядра свертки 6x6

Точность на обучаемых данных составила 0.7211, на валидационных - 0.7242, на тестовых - 0.4953, графики точности и потерь представлены на рис. 5.

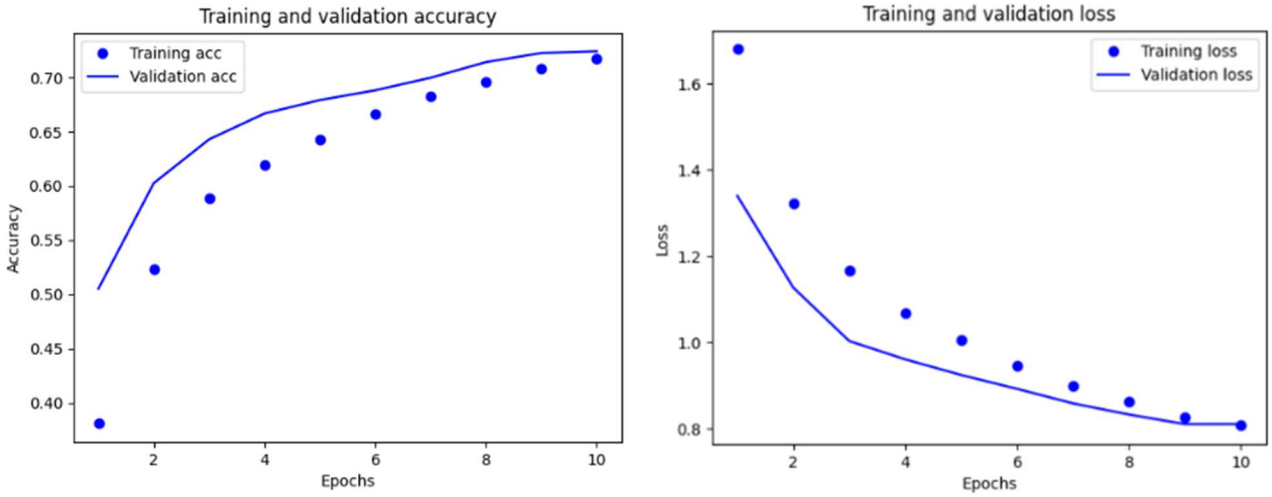

Рисунок 5 – Точность и потери ИНС с размером ядра свертки 6

Точность этой ИНС упала по всем параметрам, также увеличилось время обучения. ИНС с размером ядра свертки 6 обучалась 65.9 минут, что почти в два раза больше предыдущей ИНС. Ни одна предложенная нейросеть не добилась точности ИНС1, из чего можно сделать вывод, что конфигурация ИНС с ядром свертки размера 3x3 и слоем разреживания – оптимальная конфигруация сети. Вероятно, на большем колитчестве эпох ИНС достигла большей точности, однако аппаратные ограничения не позволяют мне испытать такую сеть.

Выводы.

В ходе лабораторной работы была реализована классификация объектов на фотографиях по 10 классам.

Приложение А

```
	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 = 32
	num_epochs = 10
	kernel_size = 3
	pool_size = 2
	conv_depth_1 = 32
	conv_depth_2 = 64
	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))
	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_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)

	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)

	model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy
'])

	hist = model.fit(X_train, Y_train,batch_size=batch_size, epochs=num_epochs
	                 ,verbose=1, validation_split=0.1)
```

```
results = model.evaluate(X_test, Y_test, verbose=1)
print(results)

hist_dict = hist.history
loss_values = hist_dict['loss']
val_loss_values = hist_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


plt.clf()
acc_values = hist_dict['accuracy']
val_acc_values = hist_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


model_json = model.to_json()
with open("model_5.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model_5.h5")
```