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

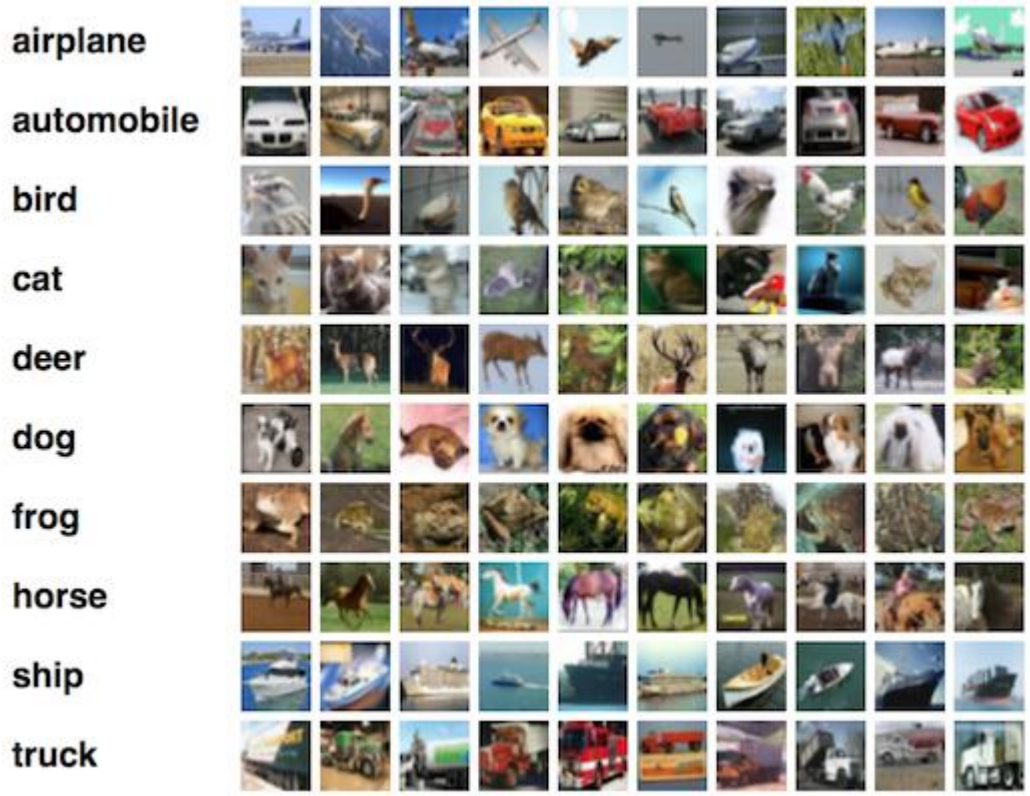
Студентка гр. 8383 _____ Максимова А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы

Реализовать распознавание объектов на фотографиях (Object Recognition in Photographs) CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, корабль и грузовик).



Задачи

- 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 сети без слоя Dropout
3. Исследовать работу сети при разных размерах ядра свертки

Основные теоретические положения

Для задач, связанных с распознаванием изображений, обычно не используют многослойный полносвязанный персептрон. Во-первых, это связано с большим количеством весов для обучения, быстро увеличивающемся при незначительном увеличении размеров изображений, что в свою очередь увеличивает время обучения и требуемое количество данных, используемых в процессе обучения. Во-вторых, изображения представляются в виде тензоров первого ранга, что приводит к потери топологии изображения.

Сверточные нейронные сети

Решением проблемы является использование сверточных нейронных сетей, в которых используются принципы локального восприятия, разделяемости весов и уменьшения размерности. Проще говоря, идея использования данного типа ИНС заключается в первичной обработке изображения с выявлением полезной информации - свертка изображения и последующее понижении качества изображения - субдискретизация, не вызывающие при этом взрывного роста количества параметров модели.

Используемые данные

CIFAR-10: набор из 60 тысяч изображений размера 32 на 32 пикселя, на каждом их которых находится один объект из 10 возможных. 50 тысяч изображений отводится на обучение и 10 тысяч - на тестирование. Изображения - цветные, в представлении RGB. Кроме того, на изображениях отсутствуют пересечения по классам.

Задача классификации

Задача, в которой имеется множество объектов, где каждый объект, исходя из его свойств и параметров, можно отнести к конкретному классу. Таким образом, нейронная сеть, получая на вход объект, возвращает на выход вероятность (дискретную величину) его принадлежности к каждому из классов. В процессе обучения ИНС стремимся достигнуть результата, когда вероятность

принадлежности к правильному классу имеет значение единицы, к другим классам - нуля.

Выполнение работы

1. Были импортированы все необходимые для работы классы и функции.

```
from keras.datasets import cifar10
import matplotlib.pyplot as plt
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
```

2. Были заданы следующие гиперпараметры (параметры, выбираемые до обучения):

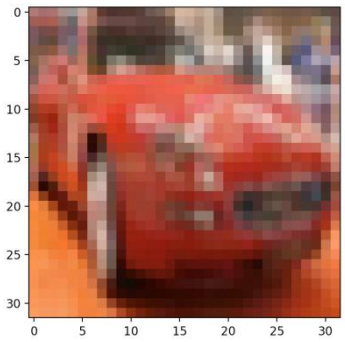
Параметр	**Означает**	**Значение**
batch_size	Количество обучающих образцов, обрабатываемых одновременно за одну итерацию алгоритма градиентного спуска	32
num_epochs	Количество итераций обучающего алгоритма по всему обучающему множеству	20
kernel_size	Размер ядра в сверточных слоях	3
pool_size	Размер подвыборки в слоях подвыборки	2
conv_depth	Количество ядер в сверточных слоях	conv_depth _1 = 32 conv_depth _2 = 64
drop_prob	Применяем dropout после каждого слоя подвыборки, а также после полносвязного слоя	drop_prob_1 = 0.25 drop_prob_2 = 0.5
hidden_size	Количество нейронов в полносвязаном слое MLP	512

3. Загрузка данных была выполнена с помощью функции load_data(), которая возвращает кортеж из 4 массивов NumPy: данные для обучения и тестирования.

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

Для проверки успешности загрузки было открыто одно из изображений.

```
plt.imshow(x_train[5])
plt.show()
```



4. Была выполнена нормализация входных данных.

```
x_train = x_train / np.max(x_train)  # [0, 1]
x_test = x_test / np.max(x_test)
```

Как видно, нормализация выполнена успешно.

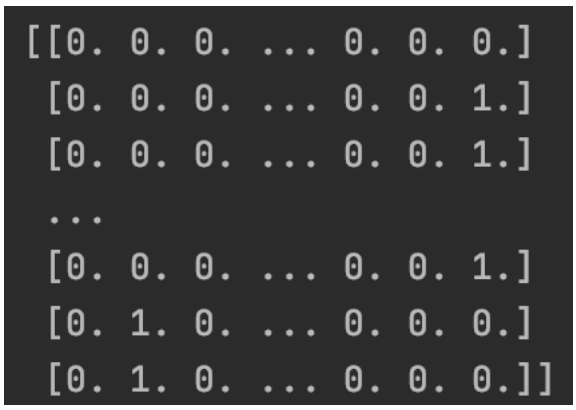
```
[[0.5372549  0.5176471  0.49411765]
 [0.50980395 0.49803922 0.47058824]
 [0.49019608 0.4745098  0.4509804 ]
 ...
 [0.70980394 0.7058824  0.69803923]
 [0.7921569  0.7882353  0.7764706 ]
 [0.83137256 0.827451   0.8117647 ]]

[[0.47843137 0.46666667 0.44705883]
 [0.4627451  0.45490196 0.43137255]
 [0.47058824 0.45490196 0.43529412]
 ...
 [0.7019608  0.69411767 0.6784314 ]
 [0.6431373  0.6431373  0.63529414]
 [0.6392157  0.6392157  0.6313726 ]]]
```

5. Кроме того, было необходимо подготовить правильный формат выходных значений.

```
# изменение формата правильных ответов
y_train = np_utils.to_categorical(y_train, num_classes)
y_test = np_utils.to_categorical(y_test, num_classes)
```

Как можно заметить, преобразование из вектора в матрицу было выполнено успешно.



6. Далее была определена модель нейронной сети, имеющая следующую структуру:

inp → conv_1[32] → conv_2[32] → pool_1 → drop_1 → conv_3[64] → conv_4[64] → pool_2→ drop_2 → flatten → hidden → drop_3 → out,

где

- inp - Input: входной слой, принимающий изображение размера 32 на 32 пикселя, представленное в RGB;
- conv_i - Convolution2D: слой свертки;
- pool_i - MaxPolling2D: слой подвыборки;
- flatten: слой, преобразующий двумерные данные в одномерные;
- hidden - Dense: полносвязанный слой из 512 нейронов;
- drop_i - Dropout: слой, используемый для предотвращения переобучения ИНС (исключение из сети нейронов с вероятностью p);
- out - выходной слой из 10 нейронов (вероятности принадлежности изображения к каждому из классов).

На всех слоях кроме выходного используется функция активации $ReLU = max(0, x)$, на выходном - $SoftMax$ $\frac{e^{x_i}}{\sum_{i=0}^{k} e^{x_i}}$ (преобразует вектор действительных чисел в вектор вероятностей).

7. Были определены следующие параметры обучения сети: в качестве функции потерь используется "categorical_crossentropy", которую предпочтительно использовать в задачах классификации, когда количество классов больше двух, метрика - точность, оптимизатор - "adam".

8. После было запущено обучение сети с помощью метода fit, адаптирующего модель под обучающие данные.

9. Для проверки работоспособности программы она была запущена. В процессе обучения нейронной сети отображаются четыре величины: loss/val_loss - потери сети на обучающих/контрольных данных соответственно и accuracy/val_accuracy - точность на обучающих/контрольных данных. Как видно, программа работает.

```
Epoch 1/200
1407/1407 [==============================] - 108s 76ms/step - loss: 1.7598 - accuracy: 0.3512 - val_loss: 1.1743 - val_accuracy: 0.5814
Epoch 2/200
1407/1407 [==============================] - 105s 74ms/step - loss: 1.1582 - accuracy: 0.5904 - val_loss: 0.9409 - val_accuracy: 0.6660
Epoch 3/200
1407/1407 [==============================] - 107s 76ms/step - loss: 0.9704 - accuracy: 0.6565 - val_loss: 0.8068 - val_accuracy: 0.7298
Epoch 4/200
1407/1407 [==============================] - 105s 74ms/step - loss: 0.8603 - accuracy: 0.7001 - val_loss: 0.7518 - val_accuracy: 0.7406
Epoch 5/200
1407/1407 [==============================] - 105s 75ms/step - loss: 0.7873 - accuracy: 0.7245 - val_loss: 0.7328 - val_accuracy: 0.7476
```

Как видно из рисунка, время прохождения одной эпохи составляет более полторы минуты, поэтому исходное значение параметра количества эпох равное 200, была уменьшено в 10 раз (5 экспериментов по 200 эпох заняли бы приблизительно 28 часов).

10. Для отображения графиков ошибок и точности был использован следующий фрагмент кода:

```
loss = hist.history['loss']
acc = hist.history['accuracy']
val_loss = hist.history['val_loss']
val_acc = hist.history['val_accuracy']
epochs = range(1, len(loss) + 1)
# построение графика ошибки
plt.plot(epochs, loss, label='Training loss', linestyle='--', linewidth=2, color="red")
plt.plot(epochs, val_loss, 'b', label='Validation loss', color="blue")
plt.title('Training and Validation loss')                    #оглавление на рисунке
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

#Построение графика точности
plt.clf()
plt.plot(epochs, acc, label='Training accuracy', linestyle='--', linewidth=2, color="red")
plt.plot(epochs, val_acc, 'b', label='Validation accuracy', color="blue")
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Исследование работы сети

Эксперимент 1: *исходная модель, размер ядра свертки: 3x3*

Табл. 1 - Результаты запуска модели № 1

Потери сети на обучающих данных	Потери сети на данных валидации	Потери сети на контрольных данных
0.4429	0.6706	0.6931
Точность сети на обучающих данных	Точность сети на данных валидации	Точность сети на контрольных данных
84%	79%	78%

График ошибки на обучающих и контрольных данных:

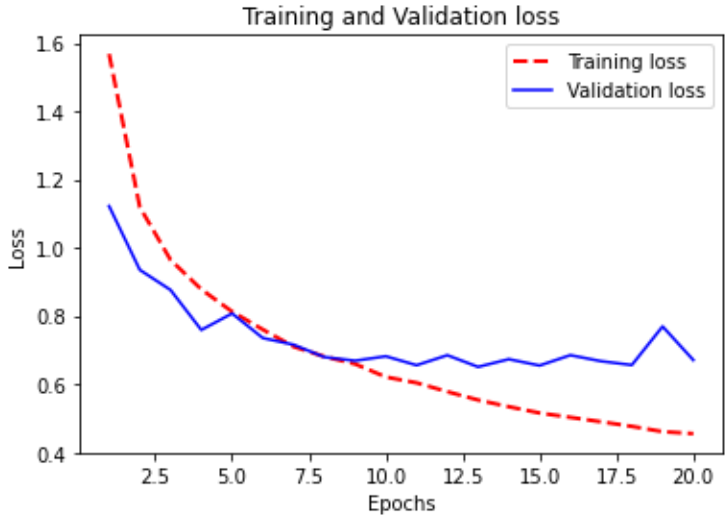
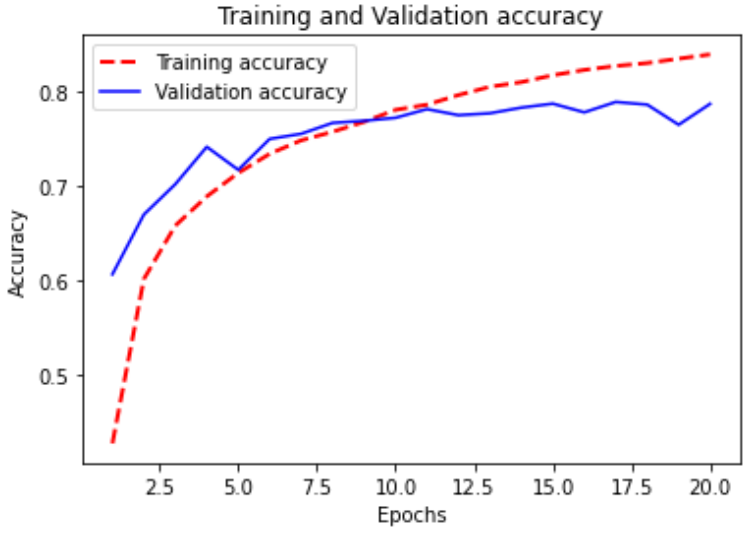


График точности на обучающих и контрольных данных:



Выводы:

Из графиков потери сети и ее точности видно, что после 10 эпохи происходит расхождение кривых обучающих и валидационных данных: точность сети на тренировочных данных повышается, а на контрольных - падает; потери сети на данных для обучения уменьшаются, на тестовых - остаются неизменными (с небольшими колебаниями). Таким образом, можно сделать вывод, что несмотря на применение слоев *Dropout* происходит переобучение нейронной сети.

Из таблицы результатов запуска видно, что в целом точность сети на контрольных данных достаточно высокая.

Эксперимент 2: *исходная модель, размер ядра свертки: 3x3, исключение слоев Dropout*

Табл. 2 - Результаты запуска модели № 2

Потери сети на обучающих данных	Потери сети на данных валидации	Потери сети на контрольных данных
0.0515	2.0226	2.1356
Точность сети на обучающих данных	Точность сети на данных валидации	Точность сети на контрольных данных
98%	75%	73%

График ошибки на обучающих и контрольных данных:

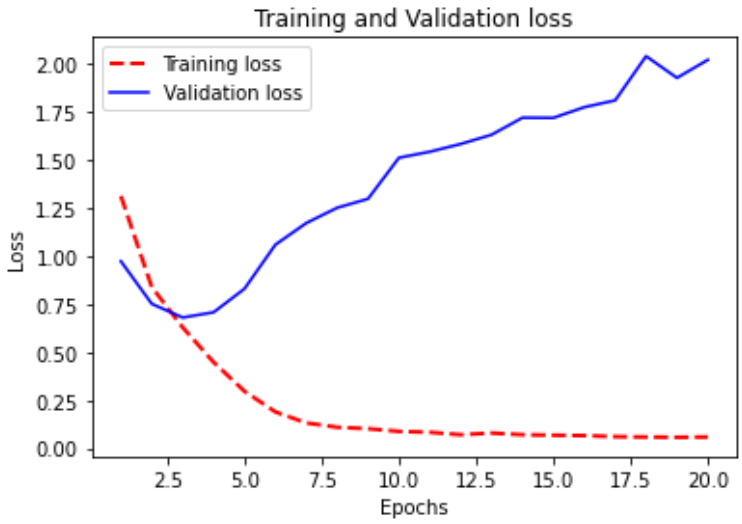
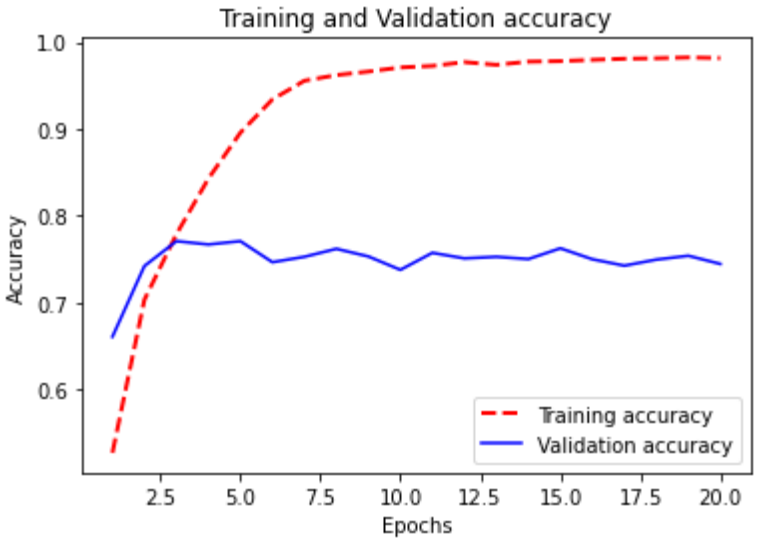


График точности на обучающих и контрольных данных:



Выводы:

Из графиков потери сети и ее точности видно, что уже после 3-4 эпохи происходит переобучение сети, что на 6 эпох раньше, чем в предыдущей модели и связано с удалением слоев Dropout, «борющихся» с переобучением сети.

Из таблицы результатов видно, что разность между точностью на обучающих и контрольных данных возросла до 25%. Такая разность достигается за счет возрастания точности сети на обучающих данных и уменьшения данного показателя на контрольных.

Таким образом, можно сделать вывод, что удаление слоев Dropout приводит к более быстрому переобучению нейронной сети, то есть данная модель хуже справляется с поставленной задачей, а, следовательно, необходимо вернуть удаленные слои.

Эксперимент 3: *исходная модель, размер ядра свертки: 2x2*

Табл. 3 - Результаты запуска модели № 3

Потери сети на обучающих данных	Потери сети на данных валидации	Потери сети на контрольных данных
0.4389	0.6653	0.7247
Точность сети на обучающих данных	Точность сети на данных валидации	Точность сети на контрольных данных
85%	78%	77%

График ошибки на обучающих и контрольных данных:

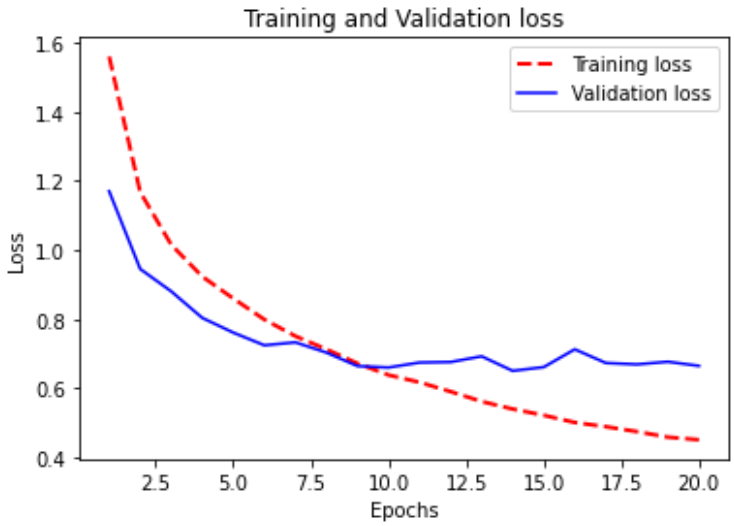
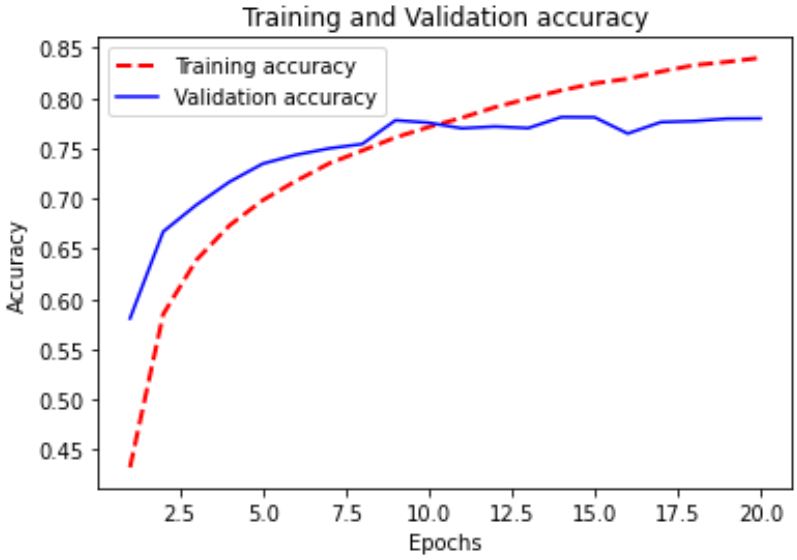


График точности на обучающих и контрольных данных:



Выводы:

Как видно из таблицы результатов запуска модели, точность нейронной сети по сравнению с моделью №1 уменьшилась, а потери сети - возросли. При этом эти изменения незначительны, что, вероятно, связано с тем, что размер ядра был уменьшен на единицу по обоим измерениям. Уменьшение точности и увеличение потерь сети может объясняться тем, что размера 2x2 ядра может не хватать для выявления отличительных признаков.

Из графиков видно, что расхождение кривых также имеет место быть, но в данной модели при возрастании точности обучающих данных тот же показатель для контрольных остается неизменным, а не убывает, тоже самое можно сказать и про график потерь сети.

Эксперимент 4: *исходная модель, размер ядра свертки: 4x4*

Табл. 4 - Результаты запуска модели № 4

Потери сети на обучающих данных	Потери сети на данных валидации	Потери сети на контрольных данных
0.4700	0.7107	0.7794
Точность сети на обучающих данных	Точность сети на данных валидации	Точность сети на контрольных данных
84%	79%	77%

График ошибки на обучающих и контрольных данных:

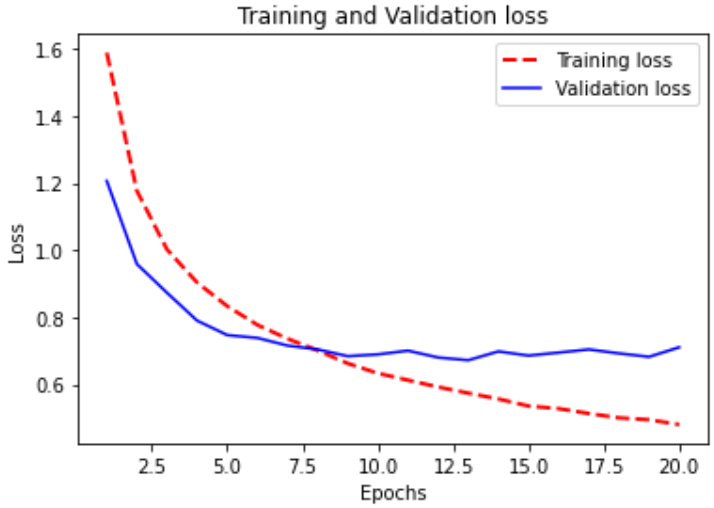
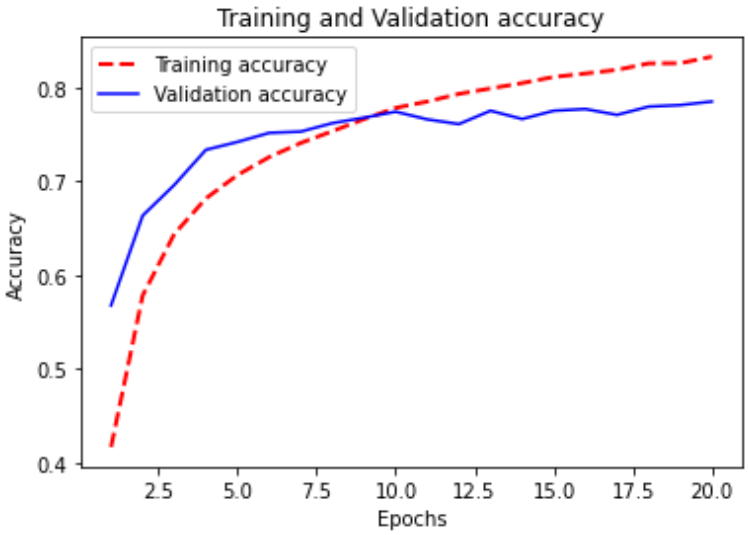


График точности на обучающих и контрольных данных:



Выводы:

Как видно из таблицы результатов запуска модели, а также графиков точности и ошибок, значения точности и потерь сети практически не изменились, что наблюдалось в модели под номером 3. Попробуем изменить размер ядра более явно.

Эксперимент 5: *исходная модель, размер ядра свертки: 6x6*

Табл. 5 - Результаты запуска модели № 5

Потери сети на обучающих данных	Потери сети на данных валидации	Потери сети на контрольных данных
0.6823	0.8555	0.9085
Точность сети на обучающих данных	Точность сети на данных валидации	Точность сети на контрольных данных
76%	72%	70%

График ошибки на обучающих и контрольных данных:

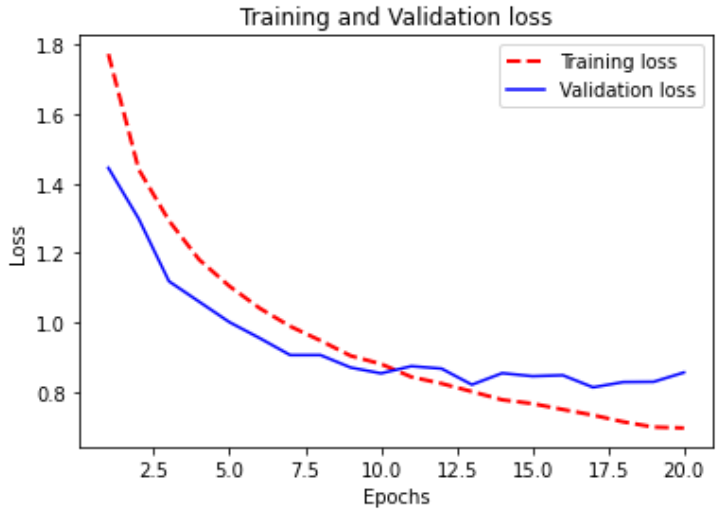
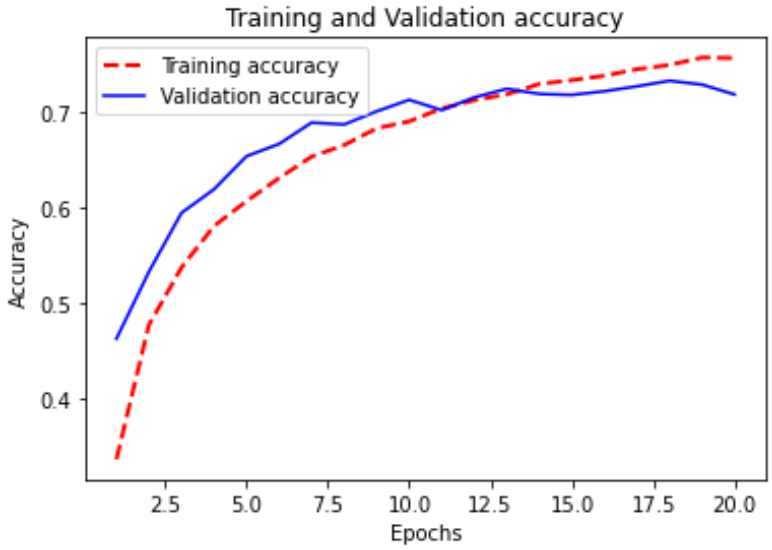


График точности на обучающих и контрольных данных:



Выводы:

Как видно из таблицы результатов запуска модели, потери сети увеличились по сравнению с рассматриваемыми прежде моделями, а точность - уменьшилась. Так что можно сделать вывод, что явное увеличение размера ядра свертки привело к ухудшению работу нейронной сети.

Из проведенных экспериментов можно сделать вывод, что удаление слоев Dropout приводит к более быстрому переобучению сети, увеличение или уменьшение размеров ядра свертки либо привело к ухудшению работы сети,

либо не привело ни к каким значительным изменениям, поэтому в качестве оптимальной модели примем модель под номером 3 со слоями Dropout и размером ядра свертки (фильтра) 3x3.

Выводы

В результате выполнения лабораторной работы был реализован механизм распознавания объектов на фотографиях из базы данных CIFAR-10. Была построена модель, решающая задачу классификации 10 видов изображений. Было изучено влияние слоев Dropout и размера ядра свертки на работу нейронной сети. Были построены графики ошибок и точности для всех рассмотренных моделей.