ГБПОУ МО «Долгопрудненский физико-технический колледж»

ОТЧЁТ

По нейронной сети

Выполнил:

Студент группы ИСП 3-1

Смирнов Игорь

Преподаватель:

Глянцев Денис Яковлевич

Долгопрудный 2021

Содержание

1. Введение.
2. Код и его объяснение.
3. Статистика/Аналитика (Игра с данными).
4. Графики.
5. Приложения.

Введение

В данной работе представлена нейронная сеть, связанная с отзывами к фильмам. Разработана по методичке (смотреть в источнике) в такой рабочей среде как Visual Studio code на языке программирования Python

Сам код:

from keras.datasets import imdb

import numpy as np

from keras import models

from keras import layers

from keras import optimizers

from keras import losses

from keras import metrics

import matplotlib.pyplot as plt

from tensorflow import optimizers

(train\_data, train\_labels), (test\_data, test\_labels) = imdb.load\_data(num\_words=8000) #Загрузка набора данных IMDB Аргумент num\_words= означает, что в обучающих данных будет сохранено только n слов, которые наиболее часто встречаются в обучающем наборе отзывов

print(train\_data[0]) #train\_data и test\_data — списки отзывов; каждый отзыв — список индексов слов

print(train\_labels[0]) #train\_labels и test\_labels — списки 0 и 1, где 0 - отрицательные отзывы, а 1 — положительные отзывы

print(max([max(sequence) for sequence in train\_data])) #выводит максимальный индекс, зависит от используемых данных: сколько слов => столько индексов

print("Тест", len(test\_data))

#декодирование одного из отзывов в последовательность слов на английском языке:

word\_index = imdb.get\_word\_index() #word\_index — словарь, который отображает слова в индексы (целочисленные)

reverse\_word\_index = dict([(value, key) for (key, value) in word\_index.items()]) # Здесь мы получаем обратное представление словаря, отображающее все реверснуто (то есть индекы в слова)

decoded\_review = ' '.join([reverse\_word\_index.get(i - 3, '?') for i in train\_data[0]]) #Здесь отзыв декодируется. Индексы смещены на 3, так как индексы 0, 1 и 2 зарезервированы для слов

#Кодирование последовательностей целых чисел в бинарную матрицу

def vectorize\_sequences(sequences, dimension=8000): #Создание матрицы с формой (len(sequences), dimension)

results = np.zeros((len(sequences), dimension))

for i, sequence in enumerate(sequences):

results[i, sequence] = 1 #Запись единицы в элемент с данным индексом

return results

x\_train = vectorize\_sequences(train\_data) #Обучающие данные в векторном виде (векторизация типа. Наверное...)

x\_test = vectorize\_sequences(test\_data) #Контрольные данные в векторном виде (векторизация типа. Наверное...)

print(x\_train[0])

y\_train = np.asarray(train\_labels).astype('float32') #Векторизация меток

y\_test = np.asarray(test\_labels).astype('float32') #Векторизация меток

epoch = 15 #задается кол-во эпох

epochs = epoch #для использования заданных эпох(удобство, как по мне)

#Определение модели

model = models.Sequential()

model.add(layers.Dense(3, activation='softmax', input\_shape=(8000,))) #Активация, добавление нейронных слоев. 3 - кол-во нейронов, activation - ф-ция активации input\_shape - взымаемые данные

model.add(layers.Dense(3, activation='softmax')) #Активация, добавление нейронных слоев. 3 - кол-во нейронов, activation - ф-ция активации

model.add(layers.Dense(2, activation='tanh')) #Активация, добавление нейронных слоев. 2 - кол-во нейронов, activation - ф-ция активации

#model.compile(optimizer='rmsprop', loss='binary\_crossentropy', metrics=['accuracy']) #Компиляция модели

#model.compile(optimizer=optimizers.RMSprop(learning\_rate=0.001), loss='binary\_crossentropy', metrics=['accuracy']) #Настройка оптимизатора

model.compile(optimizer=optimizers.RMSprop(learning\_rate=0.001), loss=losses.binary\_crossentropy, metrics=[metrics.binary\_accuracy]) #Использование нестандартных функций потерь и метрик

#создание проверочного набора, выбрав n-ое кол-во образцов из оригинального набора обучающих данных

x\_val = x\_train[:8000]

partial\_x\_train = x\_train[8000:]

print("Проверочные", len(partial\_x\_train))

y\_val = y\_train[:8000]

partial\_y\_train = y\_train[8000:]

#обучение модели в течении n эпох пакетами по 512 образцов, и слежение за потерями и точностью на n oтложенных образцов

#model.compile(optimizer='rmsprop',loss='binary\_crossentropy',metrics=['acc'])

history = model.fit(partial\_x\_train,partial\_y\_train,epochs=epoch,batch\_size=512,validation\_data=(x\_val, y\_val))

results = model.evaluate(x\_test, y\_test)

#словарь с данными обо всем происходившем в процессе обучения

history\_dict = history.history

history\_dict.keys()

print(history\_dict.keys())

#Формирование графиков потерь на этапах обучения и проверки

loss\_values = history\_dict['loss']

val\_loss\_values = history\_dict['val\_loss']

epochs = range(1, (epoch) + 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()

print(results)

#Формирование графиков точности на этапах обучения и проверк

plt.clf()

acc\_values = history\_dict['binary\_accuracy']

val\_acc\_values = history\_dict['val\_binary\_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()

#Обучение новой модели с нуля

model = models.Sequential()

model.add(layers.Dense(3, activation='softmax', input\_shape=(8000,)))

model.add(layers.Dense(3, activation='softmax'))

model.add(layers.Dense(2, activation='tanh'))

model.compile(optimizer='rmsprop',loss='binary\_crossentropy',metrics=['accuracy'])

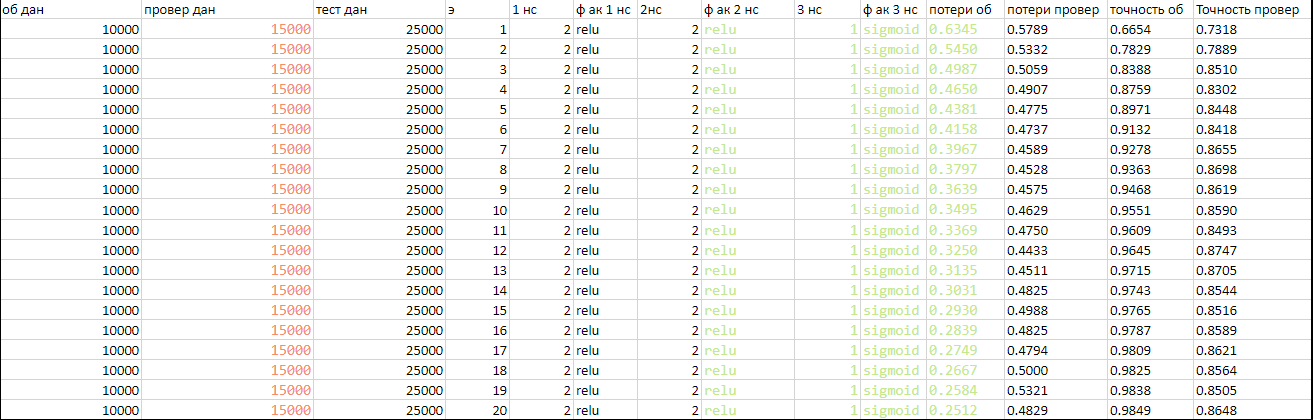
model.fit(x\_train, y\_train, epochs=4, batch\_size=512)

resultss = model.evaluate(x\_test, y\_test)

print(resultss)

АНАЛИТИКА ДАННЫХ

1 проход:



В источниках будет указана ссылка на exсel таблицу для более подробного рассмотрения проходов по измененным данным.

В первом проходе мы использовали изначально 10000 обрабатываемых данных. Количество проверочных данных составило 15000, а количество тестовых данных составляло около 25000. Проходимых эпох было принято решение поставить 20 для достижения наиболее подробного и лучшего обучения нейронной сети. Использовано 3 нейронных слоя с таким набором нейронов: 2/2/1. Функции активации использовались на первых двух слоях такая как relu а на третьем sigmoid.

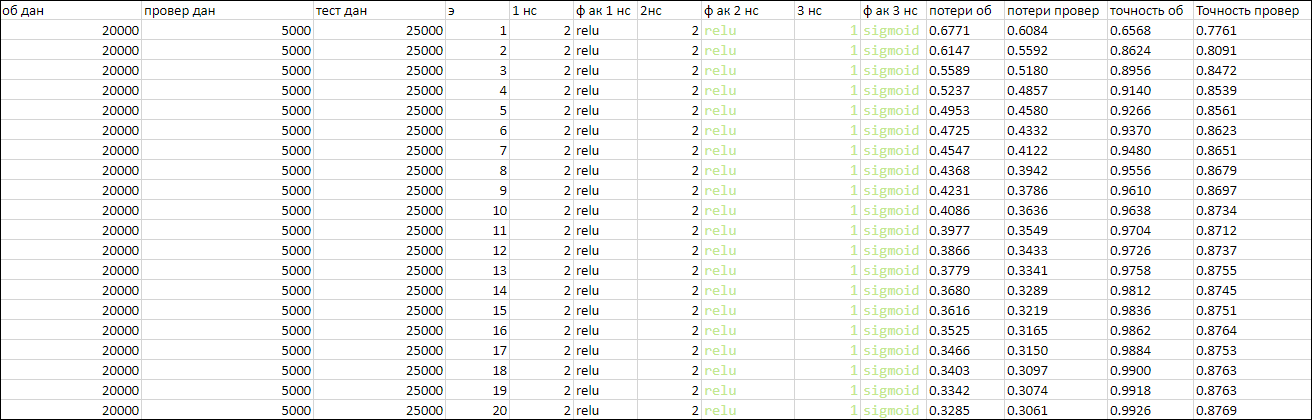
При таких входных данных потеря обрабатывающих данных в 1 эпохе составила 0.6345(64%), а в последней 0.2512(25%). В совокупности всех этих факторов можно сказать, что с каждой эпохой потеря обрабатывающих данных все время уменьшалась. При этом точность всегда росла и в сравнении с первой эпохой, где точность составляла 0.6654(67%), в конечном итоге она составляла 0.9849(целых 98-99%).

Проверочные данные:

Их потеря составляла на начальном этапе 0.5789(58%), а в конечном результате, т.е. на последней эпохе потеря составила 0.4829(48%) => меньше 50%. Исходя из выше предоставленных данных о работе нейронной сети можно сказать, что потеря проверочных данных была снижена. При этом точность тоже не стояла на месте: 1 эпоха - 0.7318(73%), а 20 эпоха - 0.8648(87%).

Можно сделать вывод, что нейронная сеть постепенно/поэтапно обучается и причем вполне успешно.

2 проход:



Во втором проходе мы использовали изначально 20000 обрабатываемых данных. Количество проверочных данных составило 5000, а количество тестовых данных составляло около 25000. То есть количество тестовых данных не изменилось. Проходимых эпох было принято решение оставить 20 для достижения наиболее подробного и лучшего обучения нейронной сети. Использовано 3 нейронных слоя с таким набором нейронов: 2/2/1. Функции активации использовались на первых двух слоях такая как relu а на третьем sigmoid.

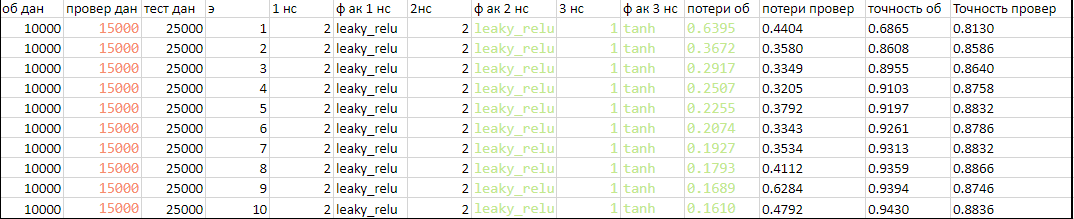
При таких входных данных потеря обрабатывающих данных в 1 эпохе составила 0.6771(68%), а в последней 0.3285(33%). В совокупности всех этих факторов можно сказать, что с каждой эпохой потеря обрабатывающих данных все время уменьшалась. Но в сравнении с 1 проходом потеря составила чуть больший процент. При этом точность всегда росла и в сравнении с первой эпохой, где точность составляла 0.6568(66%), в конечном итоге она составляла 0.9926(целых 99%).

Проверочные данные:

Их потеря составляла на начальном этапе 0.6084(61%), а в конечном результате, т.е. на последней эпохе потеря составила 0.3061(31%). Исходя из выше предоставленных данных о работе нейронной сети можно сказать, что потеря проверочных данных была снижена и весьма эффективно. При этом точность тоже не стояла на месте: 1 эпоха - 0.7761(78%), а 20 эпоха - 0.8769(88%).

В общем, при изменении стартовых данных можно заметить и изменение результатов в целом.

3 проход:



В третьем проходе мы вернулись к 10000 обрабатываемых данных. Количество проверочных данных составило 15000, а количество тестовых данных составило неизменных - 25000. Проходимых эпох было принято решение сделать 10 для того, чтобы проверить как обучится нейронная сеть за меньшее количество эпох. Использовано 3 нейронных слоя с таким набором нейронов: 2/2/1, но функции активации были изменены: на первых двух слоях такая как leaky\_relu а на третьем tanh.

При таких входных данных потеря обрабатывающих данных в 1 эпохе составила 0.6395(64%), а в последней 0.1610(16%). В совокупности всех этих факторов можно сказать, что с каждой эпохой потеря обрабатывающих данных все время уменьшалась, причем существенно. В сравнении с прошлыми проходами потеря составила средний процент на начальном этапе и существенно меньший на последнем. Точность не переставала расти и в сравнении с первой эпохой, где точность составляла 0.6865(69%), в конечном итоге она составляла 0.9430(целых 94%).

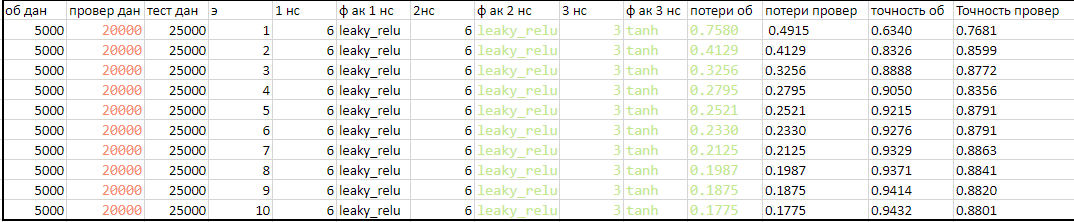
Проверочные данные:

Их потеря составляла на начальном этапе 0.4404(44%), а в конечном результате, т.е. на последней эпохе потеря составила 0.4792(48%). Судя по таблице, можно высказать, что потеря данных какое-то время начала падать, но после возросла и стала больше, чем на 1 эпохе - это не есть хорошо. Точность составила на 1 эпохе 0.8130(81%), а на последней 0.8836(88%).

Вывод:

Судя по полученным результатам из данного прогона, изучив полностью полученные данные, можно высказать следующее утверждение: Уменьшение эпох позволило достичь минимальную потерю обрабатываемых данных и оставить вполне сносный процент точности, но в проверочных данных получилось так, что их потеря стала весьма нестабильной. Точность при этом не дала особых результатов, так как совсем немного ушла вверх.

4 проход:



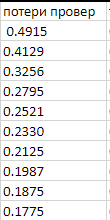
В 4 раз было решено сделать 5000 обрабатываемых данных. В связи с этим изменилось и количество проверочных данных (20000), а вот количество тестовых данных по-прежнему 25000. В этой ситуации изменения произошли и в нейронах: 6/6/3. Функции активации остались те же: leaky\_relu, tanh. Эпох было решено оставить 10 для более точного изучения.

Обрабатывающие данные:

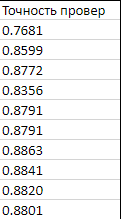
Проведя небольшую статистику по обрабатывающим данным, можно сказать, что, как и в предыдущих проходах, потери с каждой эпохой становятся меньше. Грубо говоря, первая эпоха - потери составляют 0.7580(76%), в то время как двадцатая эпоха - потери составляют 0.1775(18%). Из этого следует, что несмотря на малое кол-во эпох можно достичь меньших потерь (Ну на это еще влияет кол-во вводных данных, но это детали). Точность обрабатываемых данных отличается существенно лишь на начальной эпохе, но в конечном итоге все приходит плюс-минус к одному и тому же проценту: 1 эпоха - 0.6340(63%), 20 эпоха - 0.9432(94%)

Проверочные данные:

Здесь выходит такая ситуация:



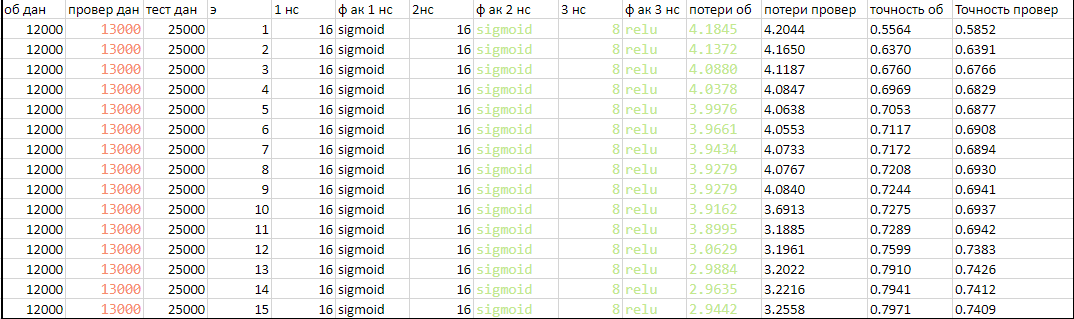
Изучив потери проверочных данных, можно отметить, что потеря изначально составляла меньше 50%, что не может не радовать, а в конечном итоге потеря составляла всего лишь 18%. Хорошенько все обдумав можно понять, что потери стали весьма небольшими под конец развития, так сказать.



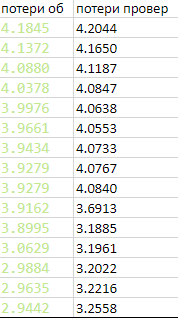
По точности проверочных данных можно сказать, что точность была весьма нестабильной и постоянно скакала от большего процента к меньшему, а между 5 и 6 эпохой произошло небольшое стагнирование и процент точности не вырос вовсе.

Сделав выводы и обдумав большинство факторов, есть вероятность ответить на вопрос о нестабильности точности проверочных данных. Возможно все зависит от вводных данных и чем их меньше, тем менее точно будет происходить “расчёт” (Либо можно сказать, что я балбес).

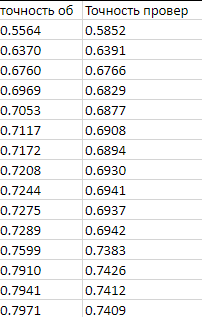
5 проход:



Точкой отправления стали 12000 обрабатываемых данных. В связи с этим изменились и проверочные данные - 13000. Тестовые данные все в том же духе, остаются неизменными себе (25000). Кол-во эпох составляет 15. Еще один небольшой эксперимент с эпохами. Также появились изменения и в количестве нейронов: 16/16/8. Еще можно отметить игру с функциями активации, теперь на первых двух слоях будет sigmoid, а на последнем слое relu.

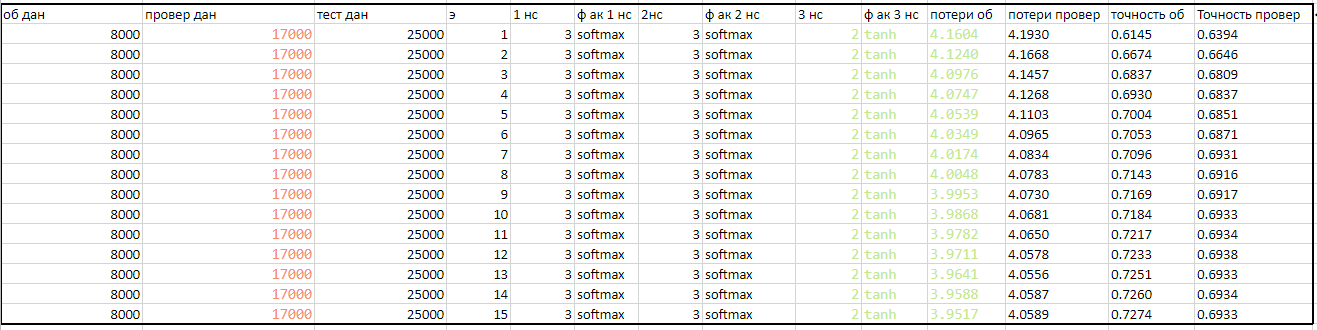


На этот раз потери вышли коллосальными. Абсолютный абсурд.

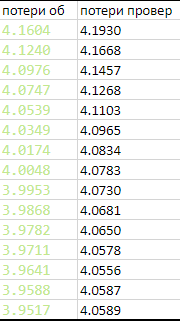


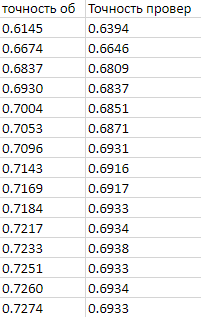
При этом точность очень и очень нестабильная, но даже в совокупности всех этих факторов выходит весьма причудливая ситуация. Точность еще куда не шла, но вот потери описать словами просто невозможно. Есть вероятность в ошибочных данных, но объяснить достаточно сложно.

6 проход:



Последний проход. Данных было решено взять 8000. При таком количестве были изменены проверочные данные и теперь они составляют 17000. Тестовые данные все так же составляют 25000 кровных. Эпох оставили 15, а вот нейронов было решено взять таким образом: 3/3/2. Были попытки поставить гораздо большее количество данных и нейронов, но, к сожалению, все было тщетно и все просто обрушивалось шквалом ошибок со стороны программы. Функции активации на этот раз тоже были изменены: softmax/softmax/tanh. Были попытки поставить на всех слоях разные функции активации, но тоже почему-то все сыпалось и не работало. Этому есть объяснение на самом деле, но я не был бы собой)  
Начнем аналитику:



Потери в этом случае тоже достигают каких-то просто невозможных отметок. Страшно наблюдать... Если так подумать, то это ведь средние 400%… Звучит вполне себе смешно и одновременно пугает, аж мурашки по коже.  


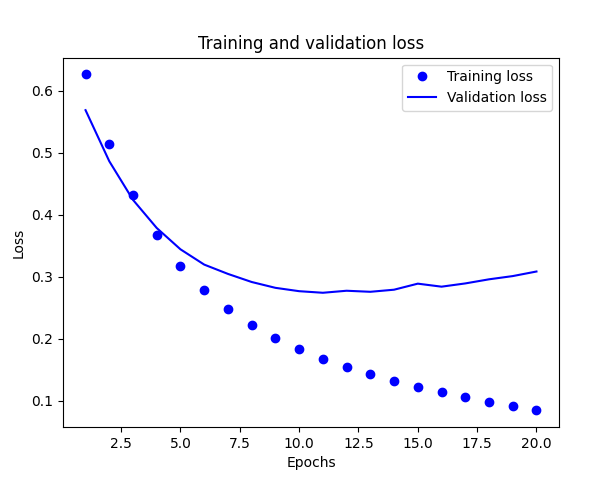
Точность на этот раз оказалась совсем не эффективной. Этот факт вовсе удручает в связи с последними тестами. Да, конечно, Точность росла вверх, но при этом точность от первоначальной эпохи не то чтобы изумительно отлична в конечной.

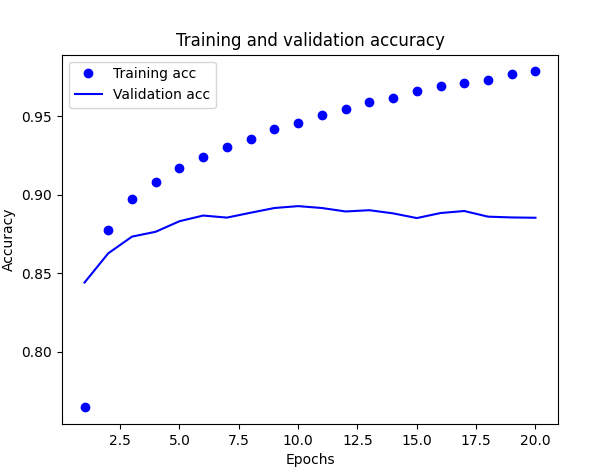
Итоги:

Подводя итоге по выше представленной аналитике/статистике данных, можно вполне себе представить, что каждое ваше изменение сильно может повлиять на результаты обучения. Изменив что-то одно - результат может сильно поменяться и отличаться от предыдущего.

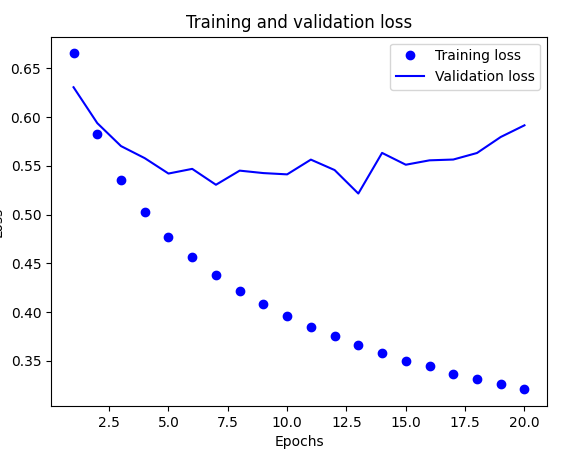
Предлагаю посмотреть на графики, которые прилагаются к табличным данным. На них можно увидеть наглядное представление потерь и точности:

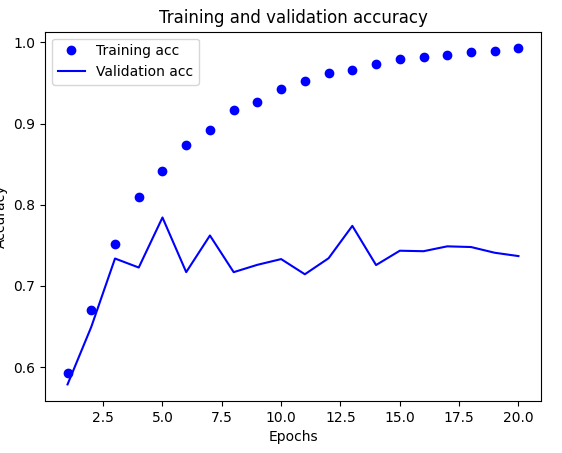
Проход 1:



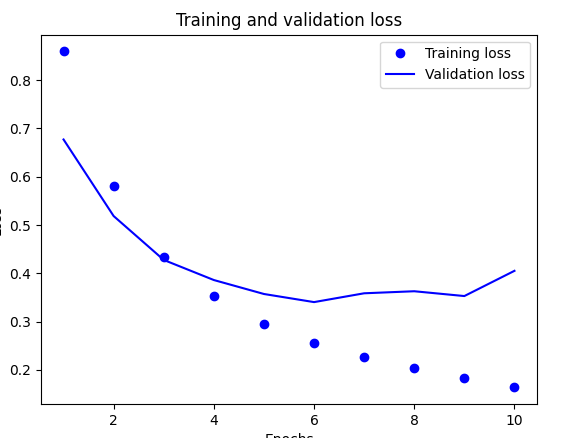


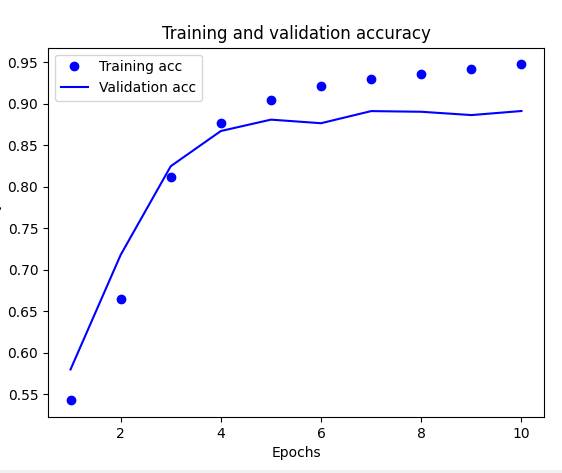
Проход 2:



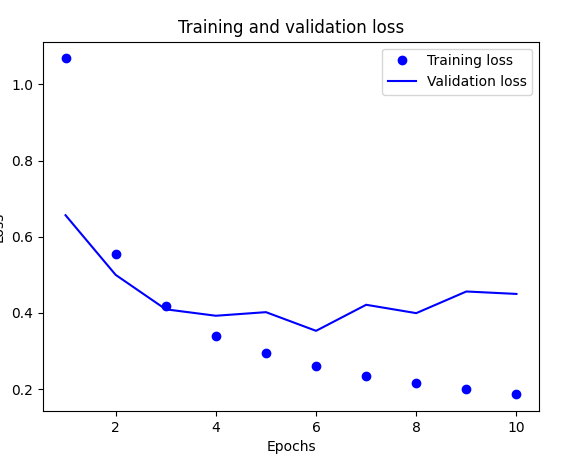


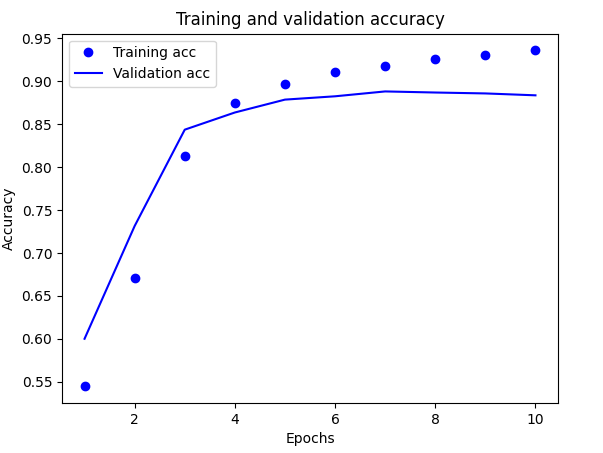
3 проход:



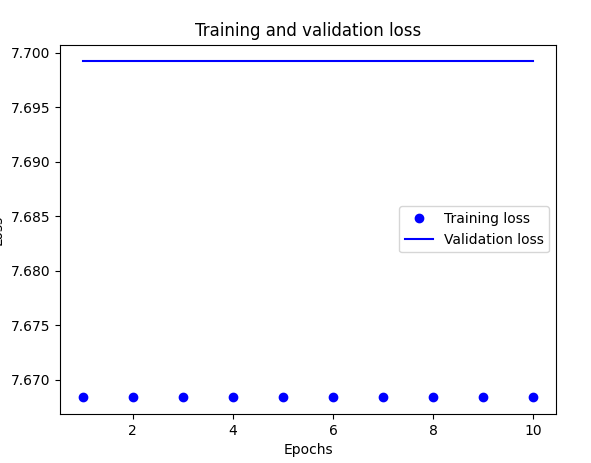


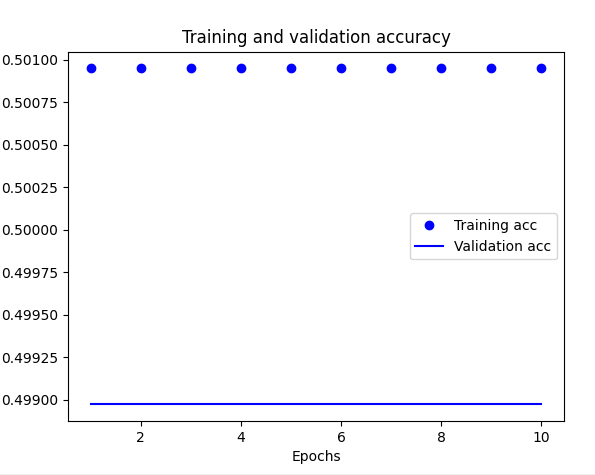
4 проход:



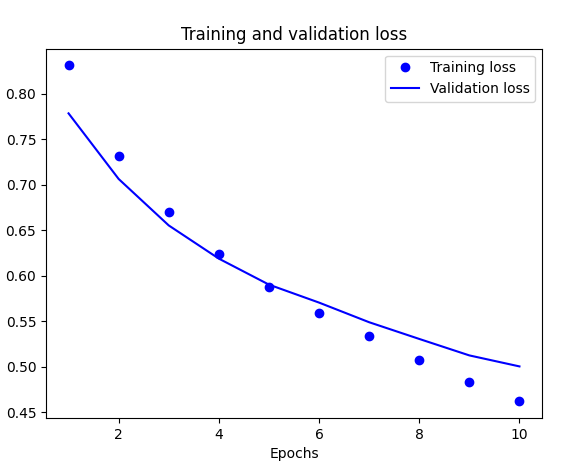


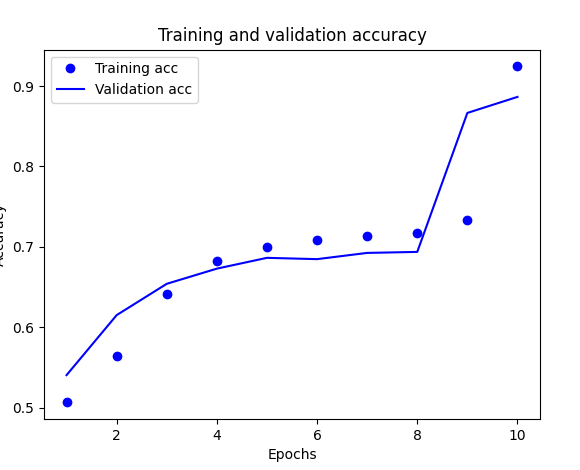
5 проход:





6 проход:





Источники:

https://disk.yandex.ru/d/\_lf7vTNlcpW5Xw