Лабораторная работа 1.

Вариант 1

Москалев Владислав Александрович.

**Github репозиторий с лабораторными -** https://github.com/worlddeleteRin/neural\_sets

**Задание:** Задано двух байтное целое число в битовом формате. Определить, делится ли оно на 3

**Выборка:** 10000 случайных чисел из диапазона от 0 до 65535

В первую очередь, необходимо создать виртуальное окружение Python и установить все необходимые пакеты.

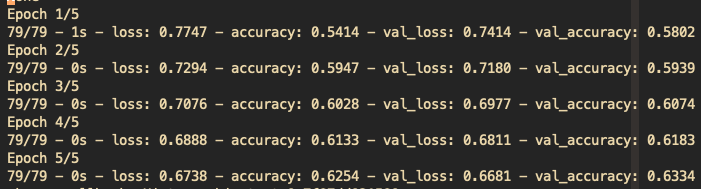
В моем проекте на github есть файл requirements.txt, в котором будут хранится все необходимые пакеты, которые необходимо установить в виртуальном окружении для достижения работоспособности всего проекта и приложенного кода.

Итак, для начала создаем выборку в побитовом представлении, после чего определяем массив зависимой переменной, для данного задания необходимо проверить, будет ли число делится на 3, для этого я создал вспомогательную функцию, которая будет принимать на вход делитель, и, число, которое мы хотим поделить. Функция будет возвращать 0, если число делится на 3 без остатка, и, 1, если число не делится нацело на 3. Таким образом, мы можем составить массив зависимой переменной, вызывая для каждого значения из константы total\_count функцию check\_divide с аргументами (3, текущее\_число).

Далее, разбиваем наши данные на тренировочные и тестовые сеты соответственно. Для тренировочного сета изначально задаем значение в 10000 ед. согласно заданию в лабораторной работе. Затем, определяем необходимые классы для нашего задания и приводим переменные к необходимым категориальным признакам. В моем случае будет 2 класса – ‘делится на 3’ и ‘не делится на 3’.

Создаем обучающую модель с функцией активации softmax, и, пока что с одним слоем, проверяем с помощью функции summary() статистику нашей модели и производим ее компиляцию с типом градиентного спуска ‘adam’ с функцией штрафа ‘categorical\_crossentropy’, в качестве параметра metrics передаем в массиве значение ‘accuracy’, чтобы вычислять в нашей модели точность (то есть случаи, когда модель будет правильно угадывать значение)

Обучаем нашу сеть, на данном этапе с параметрами epochs = 5, то есть, осуществляем 5 прогонов по нашим данным. Смотрим на результаты



На основе полученного вывода при обучении модели можно сделать следующие заключения:

Функция ошибки на обучающих данных – 67%

Точность на обучающих данных – 62%

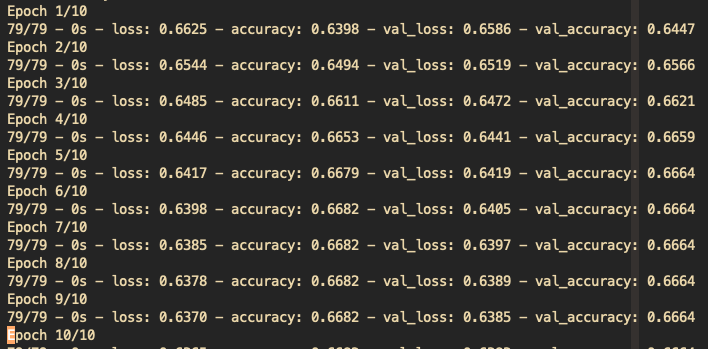
Функция ошибки на тестовых данных – 66%

Точность на тестовых данных – 63%

Такие результаты нас не устраивают, соответственно, будем пробовать всячески модифицировать модель.

1. **Увеличение числа прогонов**

Увеличим число прогонов нашей модели по данным, то есть увеличим переменную epochs с 5 до 10. Снова прогоняем, обучаем нашу модель, смотрим на результаты:



На основе полученного вывода при обучении модели можно сделать следующие заключения:

Функция ошибки на обучающих данных – 63%

Точность на обучающих данных – 66%

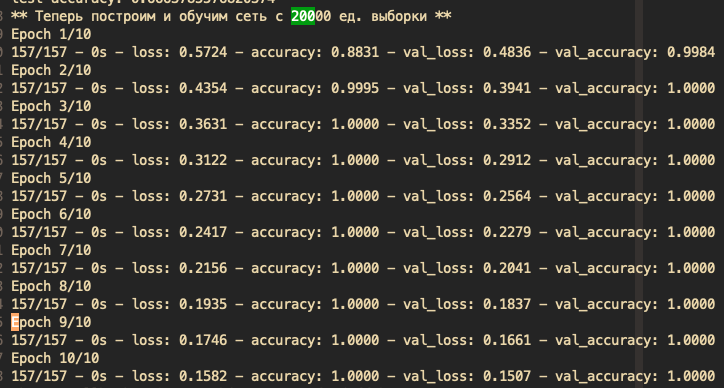
Функция ошибки на тестовых данных – 63%

Точность на тестовых данных – 66%

Похоже, что увеличение кол-во прогонов по данным не помогло нашей модели обучиться лучше. Хорошо, попробуем другие методы

1. **Увеличение размера обучающей выборки**

Увеличим размер данных для обучающей выборки, тоесть перменную learning\_sample\_size с 10000 до 20000. Снова обучаем нашу модель, смотрим на результаты



На основе полученного вывода при обучении модели можно сделать следующие заключения:

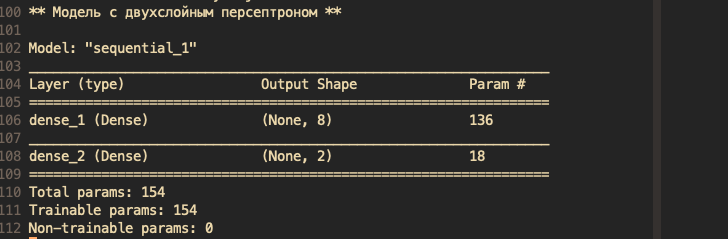
Функция ошибки на обучающих данных – 15%

Точность на обучающих данных – 100%

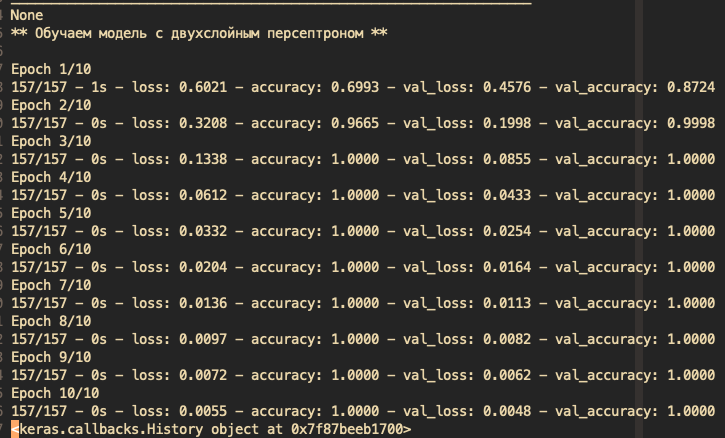
Функция ошибки на тестовых данных – 15%

Точность на тестовых данных – 100%

Уже неплохо! В качестве последнего метода по улучшению модели попробуем добавить второй слой для персептрона. Модель с двухслойным персептроном будет иметь следующие параметры:



Также создаем, компилируем и обучаем модель. Смотрим на полученные данные обучения:



На основе полученного вывода при обучении модели можно сделать следующие заключения:

Функция ошибки на обучающих данных – 0,5%

Точность на обучающих данных – 100%

Функция ошибки на тестовых данных – 0,4%

Точность на тестовых данных – 100%

Отлично. С помощью увеличения кол-ва ед. данных для обучения модели и добавления второго слоя для модели были достигнуты максимально приемлемые и эффективные результаты для работы сети. Файлы h5 с данными **для каждой** модели были также сохранены и содержатся в git репозитории