МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Лабораторная работа №\_\_1\_\_**

по дисциплине«Разработка нейросетевых систем»

Тема: «Введение в DL»

ИСПОЛНИТЕЛЬ: \_\_Журавлев Н.В.\_\_\_

ФИО

группа ИУ5-24М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"16"\_\_02\_\_\_\_\_2024 г.

ПРЕПОДАВАТЕЛЬ: \_\_\_Канев А.И.\_\_\_\_\_\_

ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_202\_ г.

Москва - 2024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Задание**

Необходимо познакомиться с фреймворком машинного обучения PyTorch и выполнить три задания:

1. Регрессия по теореме универсальной аппроксимации, ручное дифференцирование
2. Бинарная классификация с помощью автодиффиренцирования PyTorch
3. Обучить полносвязную нейронную сеть классификации 3 классов изображений из набора данных CIFAR100 по варианту из примера и затем повысить точность на тестовой выборке.

Использовались следующие классы из CIFAR100: 19, 61, 26.

Задания для самостоятельной работы:

1. Проанализируйте результаты обучения вашей модели. Что говорит о ней точность на обучающей и тестовой выборке? С какими классами модель справляется лучше и почему?
2. Проанализируйте результаты обучения. Возникает ли переобучение вашей модели? Что необходимо сделать, чтобы нивелировать это (не используя регуляризацию)?
3. Измените размер батча, но сохраните общее количество итераций. Проанализируйте результаты обучения с новыми гиперпараметрами. Что изменилось и почему?
4. Уменьшите скорость обучения и увеличьте общее количество итераций, чтобы повысить точность модели.
5. Измените вашу модель - поменяйте количество нейронов и слоев. Проанализируйте результаты обучения новой модели. Найдите лучшие гиперпараметры для этой модели.
6. Укажите, какие действия помогли повысить точность вашей модели и объясните почему.

**1. Регрессия по теореме универсальной аппроксимации, ручное дифференцирование**

Теорема универсальной аппроксимации утверждает, что нейронные сети с одним скрытым слоем достаточно большого размера (с достаточным количеством нейронов) могут аппроксимировать любую непрерывную функцию на компактном множестве с любой желаемой точностью.

В рамках лабораторной работы за исходную функцию был взят сигмоид. Были заданы 500 точек по оси X, принимающие значения от -0,5 до 0,5. Для значений по оси Y был задан разброс от изначальных значений функции сигмоида, так же к значению y добавляется шум и в итоге, получаем график представленный на рисунок 1. Необходимо восстановить чёрную линию.

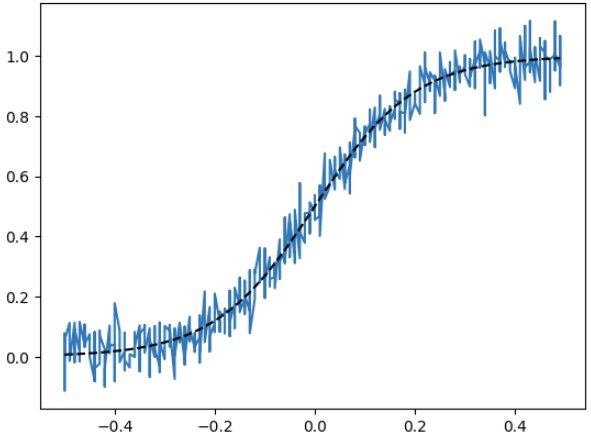


Рисунок 1 - Изначальный сигмоид и сигмоид с зашумленными значениями

Используем нейронную сеть с 3мя слоями: входной, скрытый и выходной слой. Скрытый слой состоит из 64 нейронов. Так как нейронная сеть – полносвязная, то у скрытого слоя 1 вес на нейрон, а для выходного – 64 веса на нейрон.

Обучение нейронной сети задачи регрессии состоит из шагов (итераций градиентного спуска). Во время обучения выводим результат обученный данный момент сети каждые 10000 итерация. После проведения обучения выбираем оптимальный вариант обучения и получаем, что это на 40000 итераций. Результат сравнения представлен на рисунке 2.

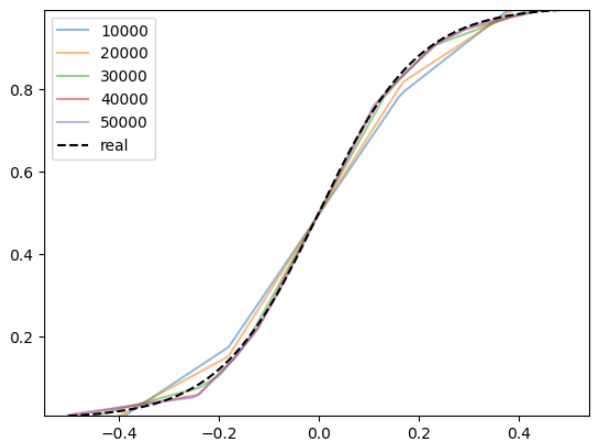


Рисунок 2 - Аппроксимированная функция на разных количествах итераций градиентного спуска

**2. Бинарная классификация с помощью автодиффиренцирования PyTorch**

Проведем бинарную классификацию. Создадим 4 группы точек, которые будут разбросаны вокруг значений: [0,0], [0,1], [1,0], [1,1]. Пусть искомые точки – [0,0], [1,1] (рисунок 3).

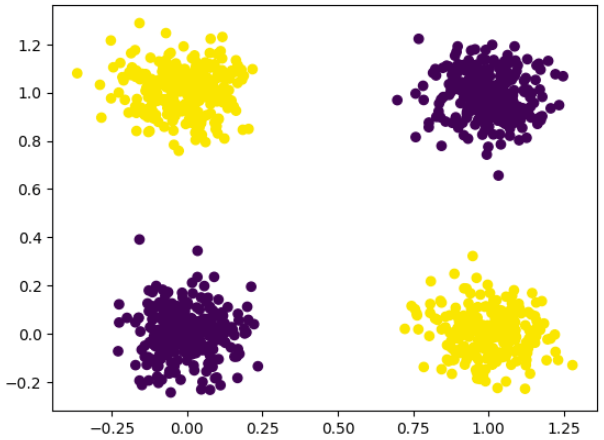


Рисунок 3 - Выборка для проведения бинарной классификации

Была сконфигурирована нейронная сеть с одним скрытым слоем с 16 нейронов. На вход подается два значения – координаты точки, а в результате получаем ответ относится ли она к фиолетовой группе или нет.

После обучения нейронной сети получили, что после 2100 итераций, дальнейшее обучение не принесло значительного улучшения (рис. 4).

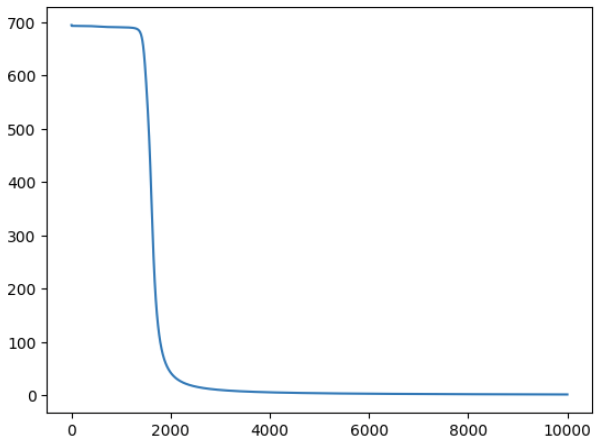


Рисунок 4 - Значение функции потерь после каждой итерации

Проверим нейронную сеть. Получаем график, где фиолетовая область означает “0” (не является классом), желтая область – “1” (является классом), а градиент между ними – промежуточные значения (неточное да или нет, которое формируется исходя из функции активации, в нашем случае это сигмоид) (рисунке 5).

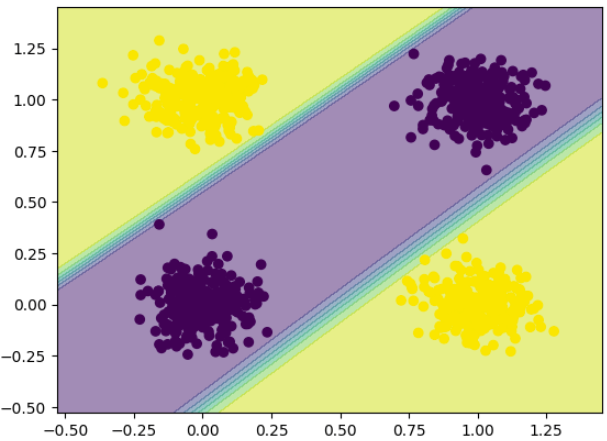


Рисунок 5 - Классификация выборки

**3. Обучить полносвязную нейронную сеть классификации 3 классов изображений из набора данных CIFAR100 по варианту из примера и затем повысить точность на тестовой выборке**

Для данной задачи были выбраны 3 класса в соответствии с вариантом, а именно классы: тарелка (19 класс), крупный рогатый скот (61 класс), краб (26 класс). Также был задан Dataloader, который выдает нейронной сети данные батчами размерностью 128.

Сконфигурированная нейронная сеть насчитывает один скрытый слой, который содержит 10 нейронов и выходной слой, содержащий 3 нейрона. Для обучения была использована функция кросс-энтропии и градиентный спуск с шагом 0.005. Количество эпох при обучении равно 250.

**Часть 3.1. Проанализируйте результаты обучения вашей модели. Что говорит о ней точность на обучающей и тестовой выборке? С какими классами модель справляется лучше и почему?**

После обучения получили, что на обучающей выборке f1-оценка для всех классов составила в районе 0.9833. Для тестовой выборки accuracy составил 0.7233. Проверка качества модели представлена в таблице 1 и таблице 2.

Таблица 1 - Результат обучения модели на тренировочных данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.9802 | 0.9900 | 0.9851 |
| 61 | 0.9819 | 0.9760 | 0.9789 |
| 26 | 0.9880 | 0.9840 | 0.9860 |

Таблица 2 - Результат обучения модели на тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.7312 | 0.6800 | 0.7047 |
| 61 | 0.6600 | 0.6600 | 0.6600 |
| 26 | 0.7757 | 0.8300 | 0.8019 |

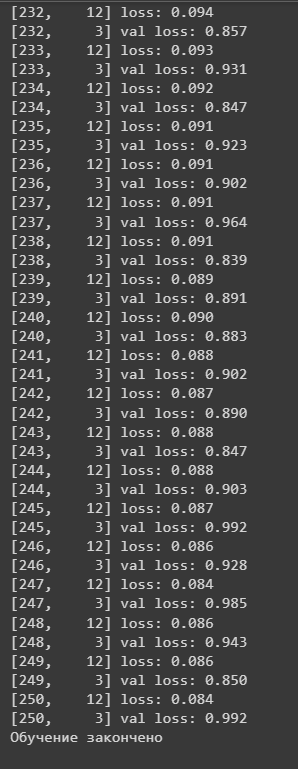


Рисунок 6 – Точность обучения

Точность на обучающей выборке говорит о том, насколько модель обучена – эта величина увеличивается с каждой эпохой. Тогда как, точность на тестовой показывает насколько модель эффективна на практике. Если точность на обучающей выборке высока, а на тестовой низкая, это может свидетельствовать о переобучении модели. Если точность низкая как на обучающей, так и на тестовой выборке, это может указывать на недообучение или несостоятельность модели.

Лучше всего модель справляется с определением класса “краб”, что может быть объяснено формой существа и отличный от него самого раковиной. Распознавание класса “тарелка” может быть затруднено относительно предыдущего, поскольку имеет круглую форму, но с рисунком. Худшие показатели у класса “крупный рогатый скот”, что может быть связано с тем, что в датасете находятся фото, на которых представлено животное не целиком, а только его часть.

**Часть 3.2. Проанализируйте результаты обучения. Возникает ли переобучение вашей модели? Что необходимо сделать, чтобы нивелировать это (не используя регуляризацию)?**

Поскольку точность на обучающей выборке высокая, а на тестовой – значительно ниже, то возможно имеет место переобучение. При помощи поиска определили, что функция потерь для тестовой выборки достигает своего минимума на 68 эпохе (рис.7).

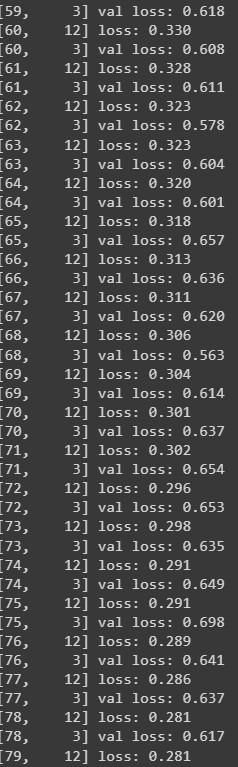


Рисунок 7 – Точность обучения

Снизим количество эпох и переучим модель. Результат представлен на таблице 3 и таблице 4.

Получаем общую точность равную для обучающей выборки – 0.9093, для тестовой выборки – 0.7267 (значение улучшились).

Таблица 3 - Результат обучения модели на тренировочных данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.9156 | 0.8900 | 0.9026 |
| 61 | 0.8809 | 0.9020 | 0.8913 |
| 26 | 0.9323 | 0.9360 | 0.9341 |

Таблица 4 - Результат обучения модели на тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.7625 | 0.6100 | 0.6778 |
| 61 | 0.6356 | 0.7500 | 0.6881 |
| 26 | 0.8039 | 0.8200 | 0.8119 |

**Часть 3.3. Измените размер батча, но сохраните общее количество итераций. Проанализируйте результаты обучения с новыми гиперпараметрами. Что изменилось и почему?**

Уменьшим размер батча с 128, до 64, так же уменьшим количество эпох в 2 раза. Получим точность на обучающей выборке – 0.8960, тестовой выборке – 0.73. Получаем, что результаты ухудшились. Это значит, что для нашей нейронной сети данный размер батча является более оптимальным.

Таблица 5 - Результат обучения модели на тренировочных данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.8898 | 0.9040 | 0.8968 |
| 61 | 0.8778 | 0.8620 | 0.8698 |
| 26 | 0.9202 | 0.9220 | 0.9211 |

Таблица 6 - Результат обучения модели на тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.7071 | 0.7000 | 0.7035 |
| 61 | 0.6542 | 0.7000 | 0.6763 |
| 26 | 0.8404 | 0.7900 | 0.8144 |

Попробуем увеличить размер батча до 256. Получаем, что значение точности для обучающей выборки составило 0.9013, для тестовой – 0.7367. Из чего можно сделать вывод, что нейросеть переобучается при данном значении.

Таблица 7 - Результат обучения модели на тренировочных данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.8609 | 0.8540 | 0.8574 |
| 61 | 0.8307 | 0.8440 | 0.8373 |
| 26 | 0.9052 | 0.8980 | 0.9016 |

Таблица 8 - Результат обучения модели на тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.7590 | 0.6300 | 0.6885 |
| 61 | 0.6522 | 0.7500 | 0.6977 |
| 26 | 0.8137 | 0.8300 | 0.8218 |

Таким образом, можно сделать вывод, что для нашей нейронной сети следует уменьшить размер батча, т.к. нейросеть переобучается.

**Часть 3.4. Уменьшите скорость обучения и увеличьте общее количество итераций, чтобы повысить точность модели**

Уменьшили скорость обучения и увеличили общее количество итераций в 5 раз. Получили следующие результаты относительно модели из 2 пункта: точность на обучающей выборке – 0.9187, на тестовой – 0.76. Нейронная сеть стала работать лучше и данный метод позволил добиться большей точности. Это связано с тем, что модель обучалась дольше, но градиентный спуск получился более плавный, что позволило достичь наименьшей точки функции потерь.

Таблица 9 - Результат обучения модели на тренировочных данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.9181 | 0.9420 | 0.9299 |
| 61 | 0.9115 | 0.9060 | 0.9087 |
| 26 | 0.9469 | 0.9280 | 0.9374 |

Таблица 10 - Результат обучения модели на тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.7374 | 0.7300 | 0.7337 |
| 61 | 0.6822 | 0.7300 | 0.7053 |
| 26 | 0.8617 | 0.8100 | 0.8351 |

**Часть 3.5. Измените вашу модель - поменяйте количество нейронов и слоев. Проанализируйте результаты обучения новой модели. Найдите лучшие гиперпараметры для этой модели**

Изменим исходную конфигурацию сети, добавив дополнительный скрытый слой. Зададим малый шаг градиентного спуска – 0.001. Опытным путём было определено, что оптимальным количеством эпох для данной нейронной сети является 57, а размер батча - 64. Получили общую точность определения равную для тренировочной выборки 0.833, для тестовой – 0.73.

Таблица 11 - Результат обучения модели на тренировочных данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.9107 | 0.9380 | 0.9241 |
| 61 | 0.8929 | 0.9000 | 0.8964 |
| 26 | 0.9543 | 0.9180 | 0.9358 |

Таблица 12 - Результат обучения модели на тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** |
| 19 | 0.7732 | 0.7500 | 0.7614 |
| 61 | 0.6607 | 0.7400 | 0.6981 |
| 26 | 0.8681 | 0.7900 | 0.8272 |

**Часть 3.6. Укажите, какие действия помогли повысить точность вашей модели и объясните почему**

* Более сложная модель лучше работает, но может переобучаться. Поэтому в случае усложнении модели следует определить оптимальное количество эпох, дабы не допустить переобучения
* Можно изменить размер батча. Для данной модели следовало уменьшить размер батча, что позволило увеличить случайность обновлений весов модели и уменьшить вероятность застревания в локальных минимумах функции потерь
* Пройти с большей точностью. Уменьшение размера шага градиентного спуска позволяет более точно спускаться, тем самым достигая лучшей минимизации функции потерь
* Добавление нового нейронного слоя. Добавление позволяет повысить точность, но необходимо перевыбрать гиперпараметры и не допускать большее количество нейронов, т.к. это может повлиять на результат

**Итоговая таблица с результатами для всех вариантов обучения**

На табл. 13 представлены результаты обучения.

Таблица 13 - Итоговая таблица с результатами для всех вариантов обучения

|  |  |  |  |
| --- | --- | --- | --- |
| **Конфигурация нейросети** | **Гиперпараметры** | **Точность** | **Комментарий** |
| FC(10), FC(3) | Lr = 0.005  batch\_size = 128  epoch = 250 | train = 98,33%  test= 72,33% | Базовый вариант |
| FC(10), FC(3) | Lr = 0.005  batch\_size = 128  epoch = 68 | train = 90,93%  test= 72,67% | Уменьшение количества эпох |
| FC(10), FC(3) | Lr = 0.005  batch\_size = 64  epoch = 34 | train = 89,60%  test= 73,00% | Уменьшение размер батча и уменьшение количество эпох в 2 раза |
| FC(10), FC(3) | Lr = 0.005  batch\_size = 256  epoch = 136 | train = 90,13%  test= 73,67% | Увеличение размер батча и увеличение количество эпох в 2 раза |
| FC(10), FC(3) | Lr = 0.001  batch\_size = 256  epoch = 780 | train = 91,87%  test= 76,00% | Увеличение количество эпох и уменьшение шаг в 5 раз |
| FC(20), FC(3) | Lr = 0.001  batch\_size = 256  epoch = 400 | train = 86,60%  test= 76,00% | Добавление скрытого слоя и подбор лучших гиперпараметров |
| FC(30), FC(10), FC(3) | Lr = 0.001  batch\_size = 256  epoch = 600 | train = 82,47%  test= 75% | Добавление скрытого слоя и подбор лучших гиперпараметров |

**Вывод**

При обучении нейронных сетей очень важны гиперпараметры. Важно определить количество эпох, при которых не будет появляться переобучение, но при этом достаточное количество для обучения с высокой точностью. Необходимо использовать batch\_size и изменение скорости обучения нужно размера, иначе точность на тестовых данных может быть понижена. Так же точность может повысить добавление ещё одного слоя, но необходим подбор гиперпараметров.