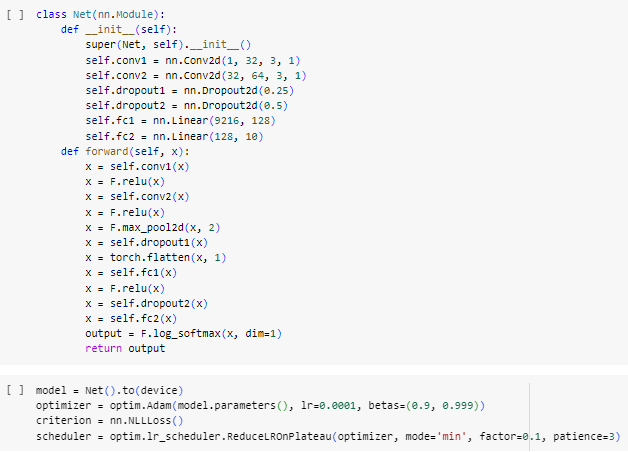
1. Выполним импорт библиотек и загрузим набор данных и настроим использование графического устройства.



2. Создадим класс НС на основе фреймворка torch и проверим работоспособность.



3. Определим функцию обучения и обучим модель. Построим график потерь.



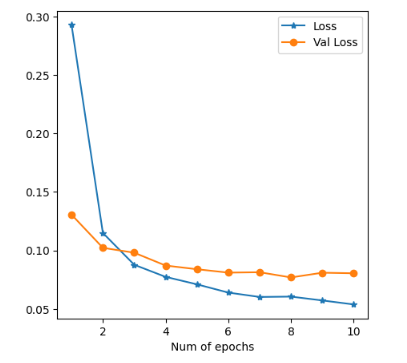
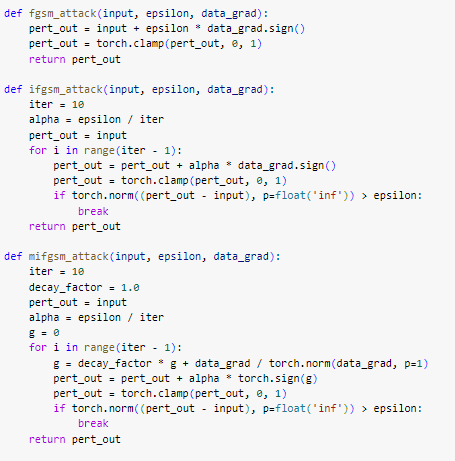


Рисунок 1 – График потерь при обучении и валидации в

зависимости от эпохи

4. Создадим функции атак (FGSM, I-FGSM, MI-FGSM)



5. Создадим Функцию проверки



6. Воспроизведем график успешности атак.



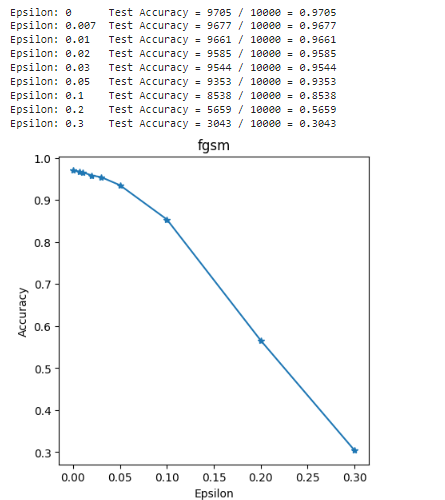


Рисунок 2 - Результат

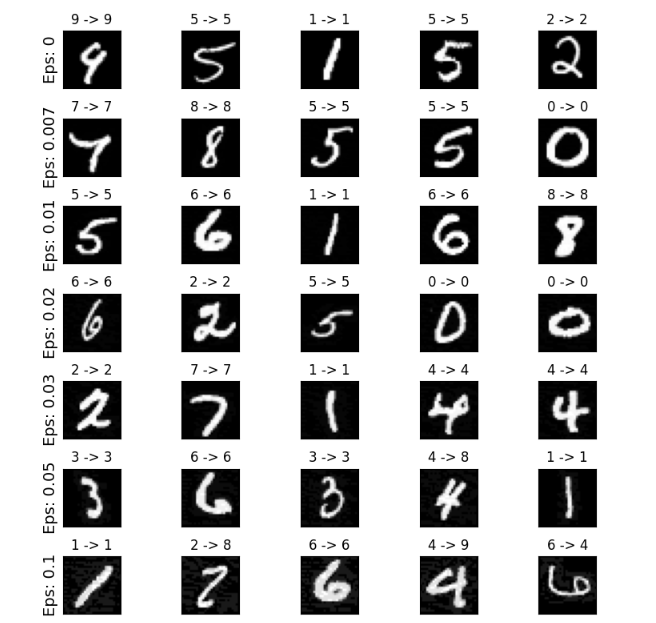


Рисунок 3 - Результат

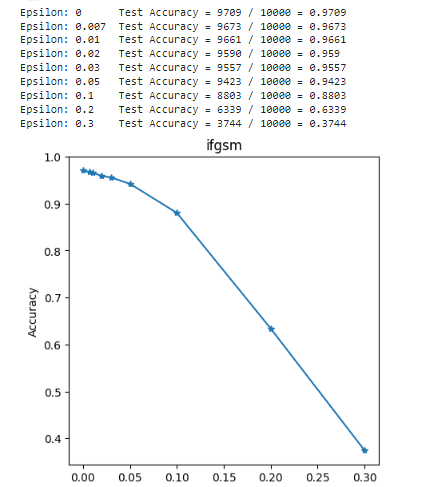


Рисунок 4 - Результат

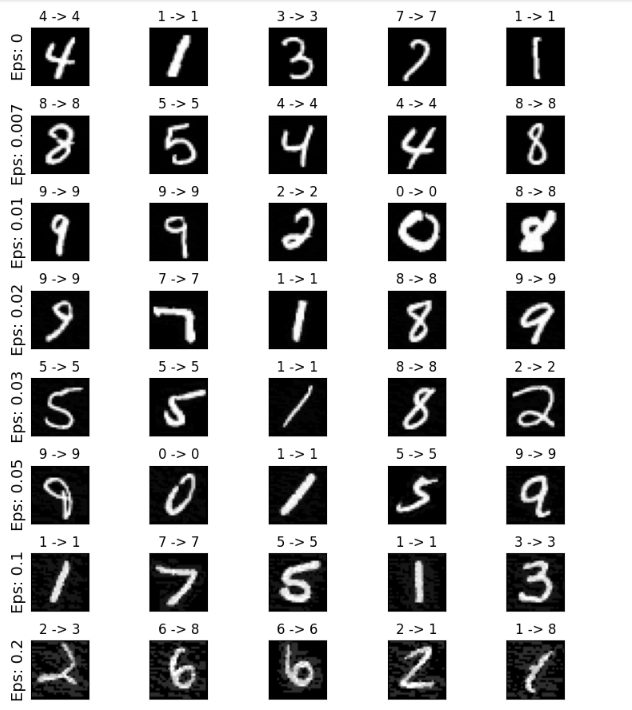


Рисунок 5 - Результат

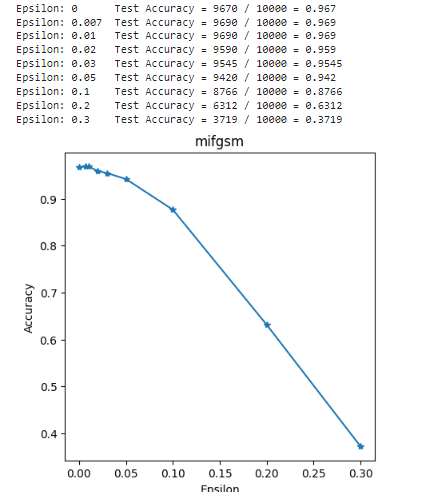


Рисунок 6 - Результат

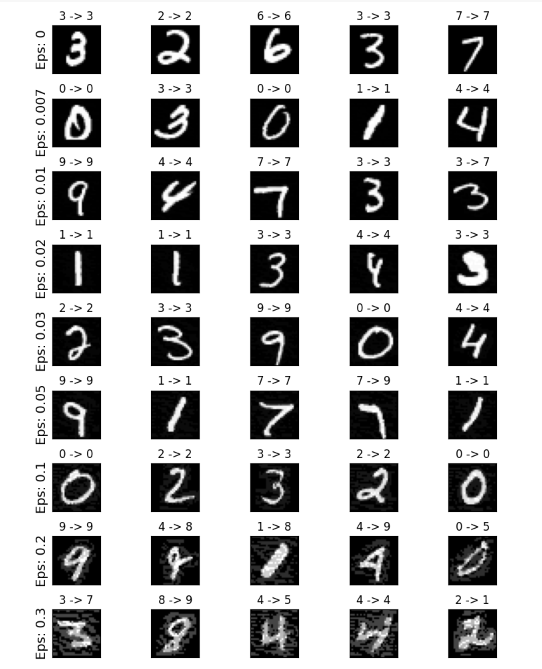
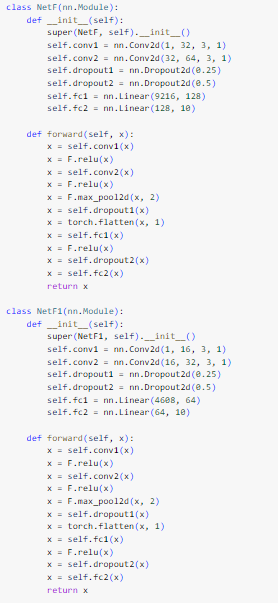
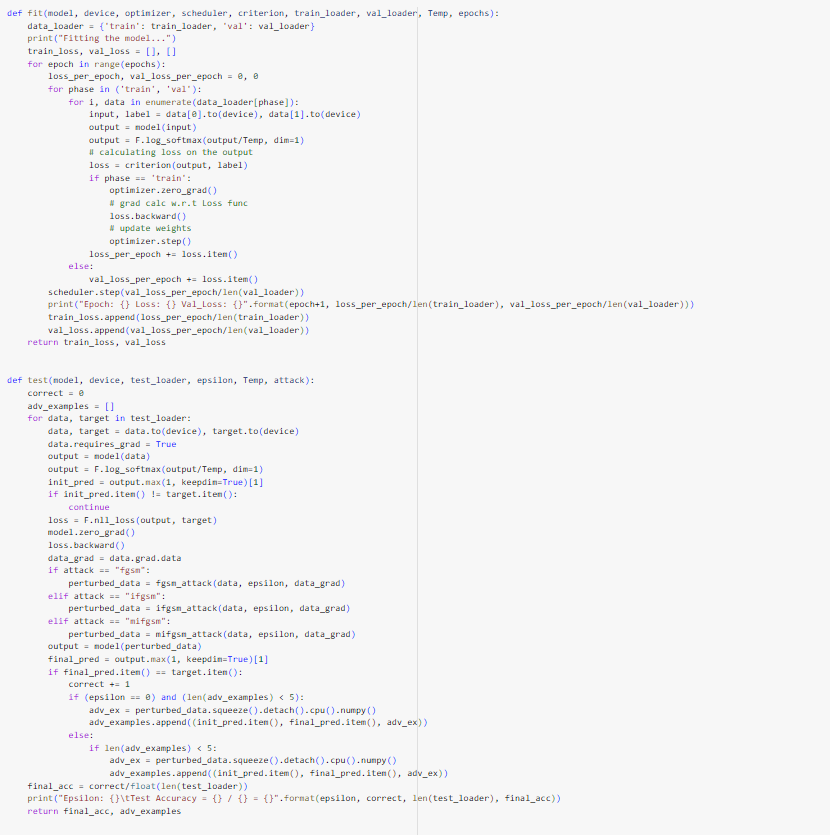


Рисунок 7 - Результат

7. Создадим 2 класса.



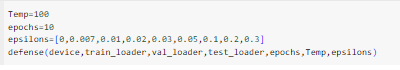
8. Преобразуем функцию обучения и тестирования.

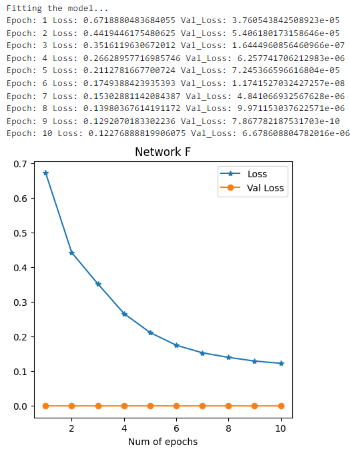


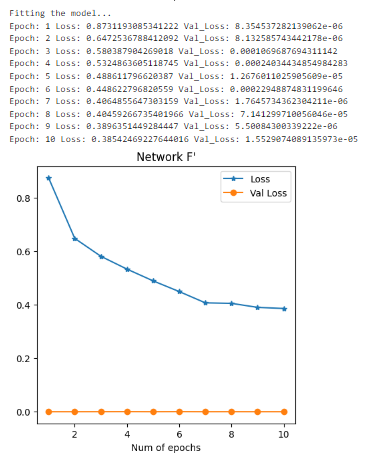
9. Сделаем функцию защиты.

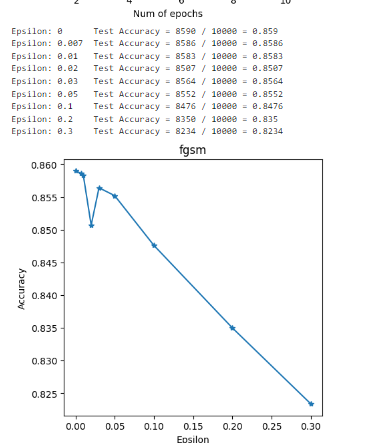


10. Получим результат

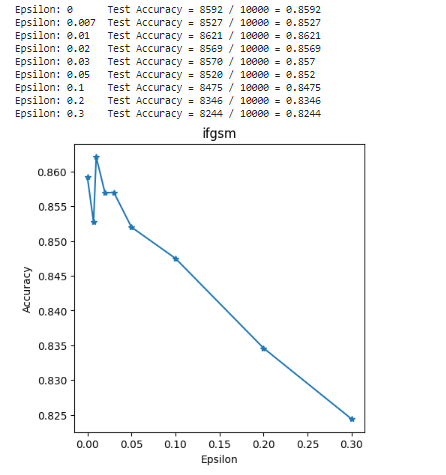


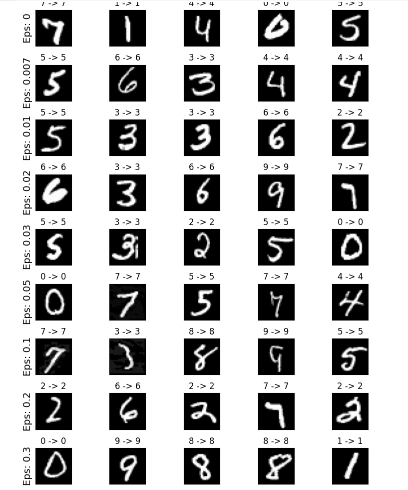


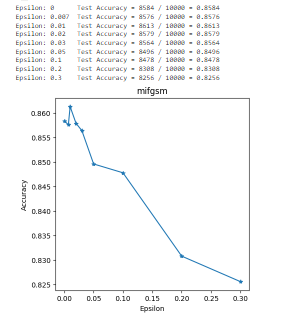


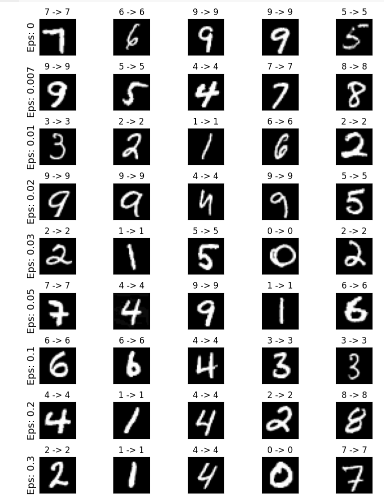












**Вывод**

На основе выполнения лабораторной работы можно сделать следующие выводы:

Обучение с двумя моделями: Использование двух моделей (NetF и NetF1), где NetF1 получает мягкие метки от NetF, является стратегией дистилляции знаний. Это может помочь в передаче "опыта" более устойчивой модели NetF менее устойчивой модели NetF1.

Графики потерь и потерь на валидационном наборе данных для обеих моделей позволяют оценить их обучение. Отслеживание изменений во времени помогает определить, происходит ли переобучение. Оценка точности модели NetF1 после обучения с использованием предложенного механизма защиты под воздействием различных атак (например, fgsm, ifgsm, mifgsm) при разных значениях эпсилон предоставляет информацию о ее устойчивости к атакам.