|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | Информатика, искусственный интеллект и системы управления |
| КАФЕДРА | Системы обработки информации и управления |

**Лабораторная работа №2:**

**“Сверточная нейросеть”**

ИСПОЛНИТЕЛИ: Якубов А. Р.

ФИО

группа ИУ5-24М \_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_" \_\_\_\_\_\_\_ 2023 г.

ПРЕПОДАВАТЕЛЬ: Канев А.И.

ФИО

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

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2023 г.

Москва - 2023

**Задание ЛР2:**

По заданию выбрать свои классы и обучить сверточную нейронную сеть до 70% точности на тестовой выборке используя GPU. Провести три обучения для 3 разных тактик пуллинга: пуллинг с помощью шага свёртки stride, макс пуллинг, усредняющий пуллинг. Сравнить достигнутое качество, время обучения и степень переобучения. Выбрать лучшую конфигурацию. Сохранить модель. Перезапустить среду выполнения - теряются все текующие переменные.

Загрузить в colab готовую уже обученную на cifar100 модель. Преобразовать в onnx и сохранить локально.

Скачать каталог с html-файлом и встроить в него два файла моделей - обученную на ЛР1 и на ЛР2.

Скачать картинки из интернета согласно варианту и открыть их в html по кнопке. Автоматически в скрипте масштабируется изображение.

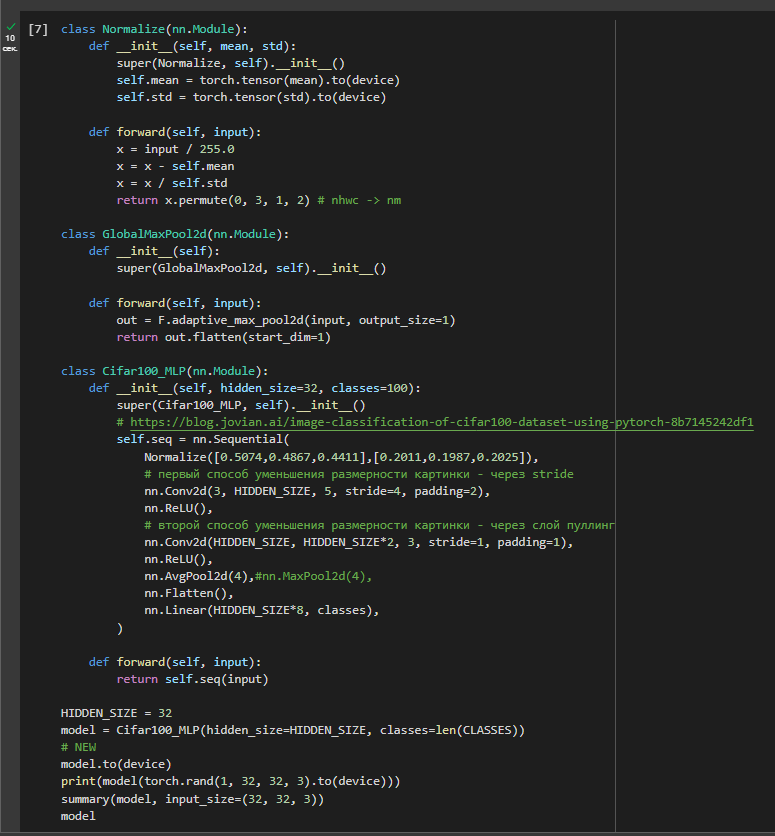
Выбрать в js нужные классы для готовой модели. Проверить на устойчивость обе модели, полносвязную и свёрточную, двигая картинку, убедиться в наличии свойства инвариантности сверточного слоя.

# **Поэтапное выполнение работы:**

## Часть 1 - Классификация изображений CIFAR100

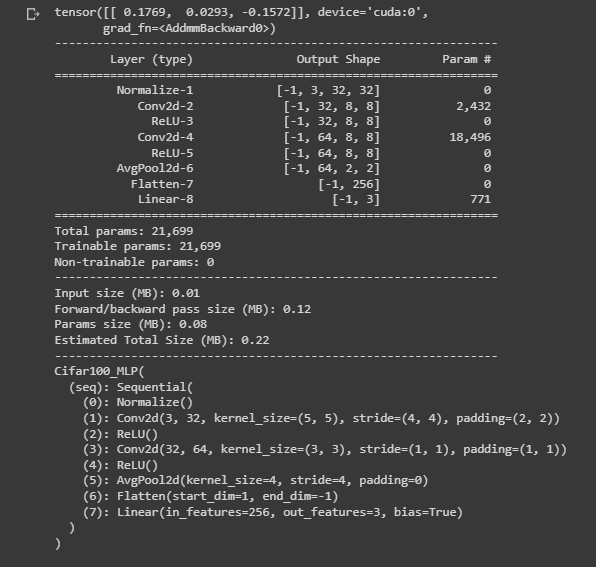
device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

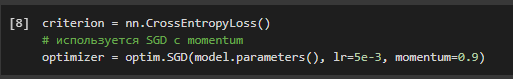
В этой строке мы указываем, что будем работать с GPU на нашем ноутбуке. Затем следует код, повторяющий начало Части 3 ЛР1, где мы загружаем датасет и разбиваем данные на тренировочную и тестовую выборки.



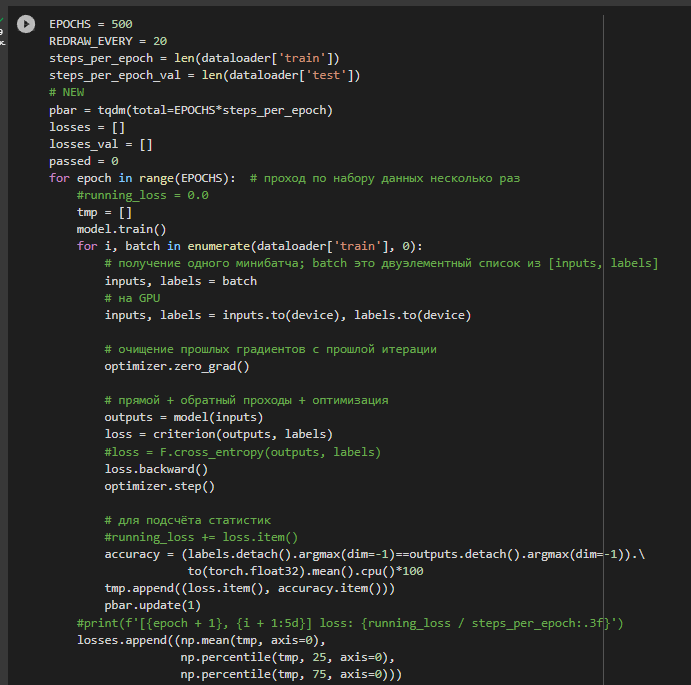
Краткое описание коды выше:

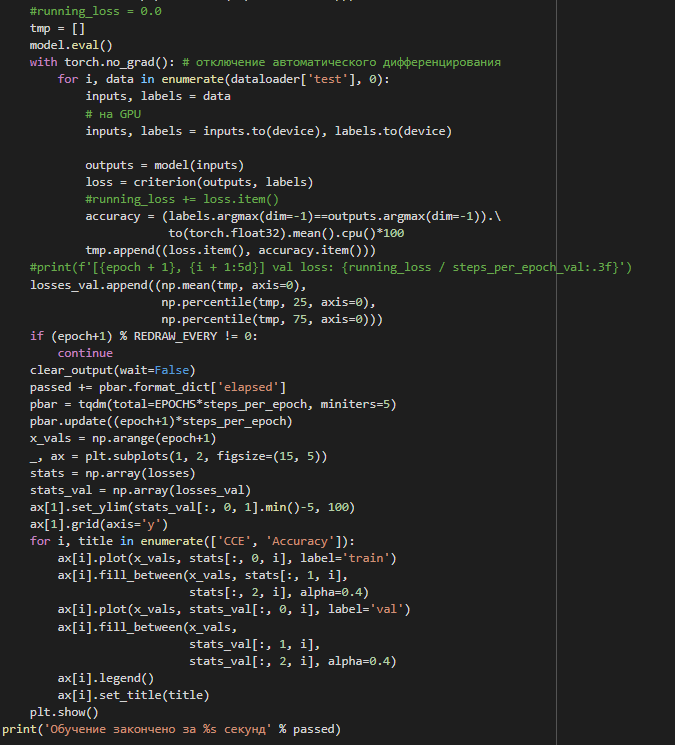
* Мы определяем класс Normalize, который нормализует входные изображения по среднему и стандартному отклонению значений пикселей.
* Мы определяем класс GlobalMaxPool2d, который применяет операцию максимального пуллинга ко всему входному тензору и возвращает его как одномерный тензор.
* Далее определяется класс Cifar100\_MLP, который представляет модель нейронной сети с несколькими сверточными и линейными слоями.
* Создается экземпляр модели Cifar100\_MLP с заданным размером скрытого слоя и количеством классов.
* Мы выводим результат работы модели на случайном тензоре размером 1x32x32x3 (одно изображение RGB размером 32x32 пикселя).
* Затем мы используем функцию summary из библиотеки torchsummary для получения информации о структуре и параметрах модели.
* В конце мы выводим модель на экран:





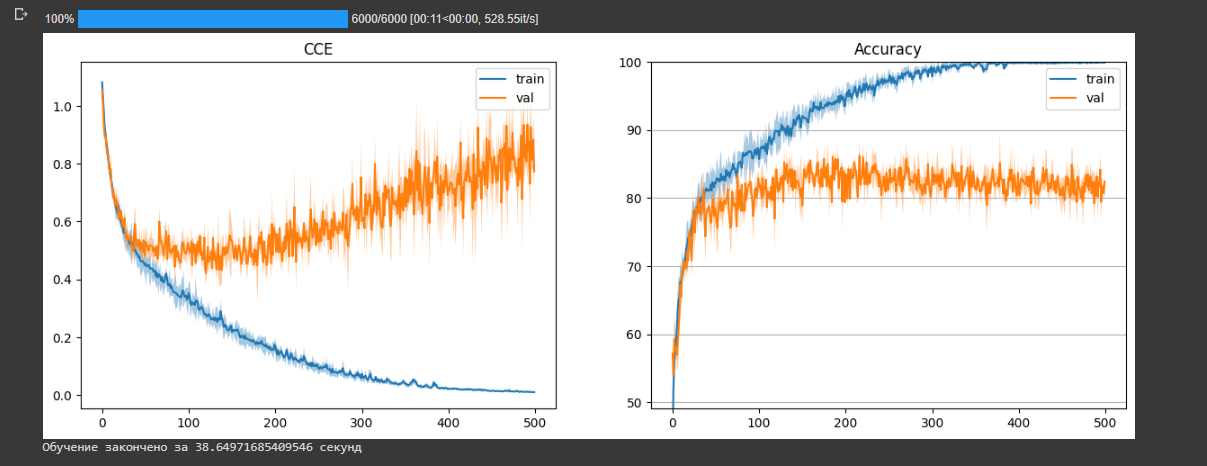
Для того, чтобы научить нашу модель, мы задаем функцию потерь и оптимизатор. Мы используем класс SGD с параметром импульс. Импульс - это гиперпараметр, который помогает обучаться быстрее и не попадать в локальные минимумы. Он делает это, добавляя часть предыдущего шага к текущему шагу



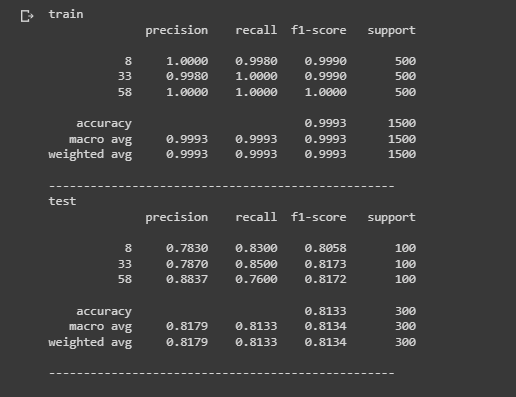


Этот код делает следующее:

* Задает количество эпох (проходов по набору данных) и частоту обновления графика
* Вычисляет количество шагов в эпохе для обучающего и тестового наборов данных
* Создает индикатор прогресса и списки для хранения потерь и точности
* Для каждой эпохи:
  + Переводит модель в режим обучения
  + Для каждого минибатча из обучающего набора данных:
    - Получает входы и метки
    - Переносит их на GPU (если есть)
    - Обнуляет градиенты оптимизатора
    - Вычисляет выходы модели и функцию потерь
    - Выполняет обратное распространение ошибки и шаг оптимизации
    - Вычисляет точность модели на минибатче
    - Добавляет потерю и точность во временный список
    - Обновляет индикатор прогресса
  + Вычисляет среднее, 25-й и 75-й перцентили потери и точности по всему обучающему набору данных и добавляет их в список потерь
  + Переводит модель в режим оценки (валидации)
  + Отключает автоматическое дифференцирование (для экономии памяти)
  + Для каждого минибатча из тестового набора данных:
    - Получает входы и метки
    - Переносит их на GPU (если есть)
    - Вычисляет выходы модели и функцию потерь
    - Вычисляет точность модели на минибатче
* Для каждой эпохи валидации:
  + Добавляет потерю и точность во временный список
  + Вычисляет среднее, 25-й и 75-й перцентили потери и точности по всему тестовому набору данных и добавляет их в список потерь валидации
  + Если номер эпохи не кратен частоте обновления графика, то пропускает остальные шаги
  + Очищает вывод ячейки
  + Обновляет индикатор прогресса
  + Создает два графика: один для функции потерь (CCE), другой для точности (Accuracy)
  + Наносит на графики значения потерь и точности для обучающего и тестового наборов данных по каждой эпохе
  + Заполняет область между 25-м и 75-м перцентилями потерь и точности полупрозрачным цветом
  + Добавляет легенду и заголовок к каждому графику
  + Показывает графики на экране
* После завершения всех эпох печатает время обучения в секундах

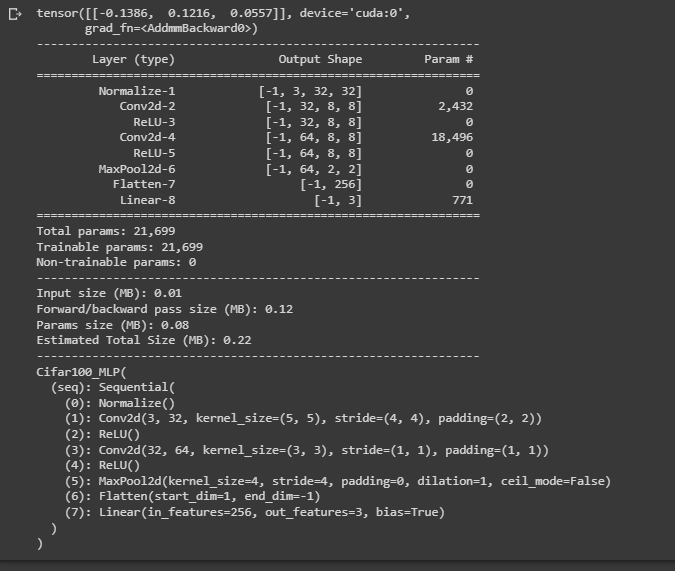


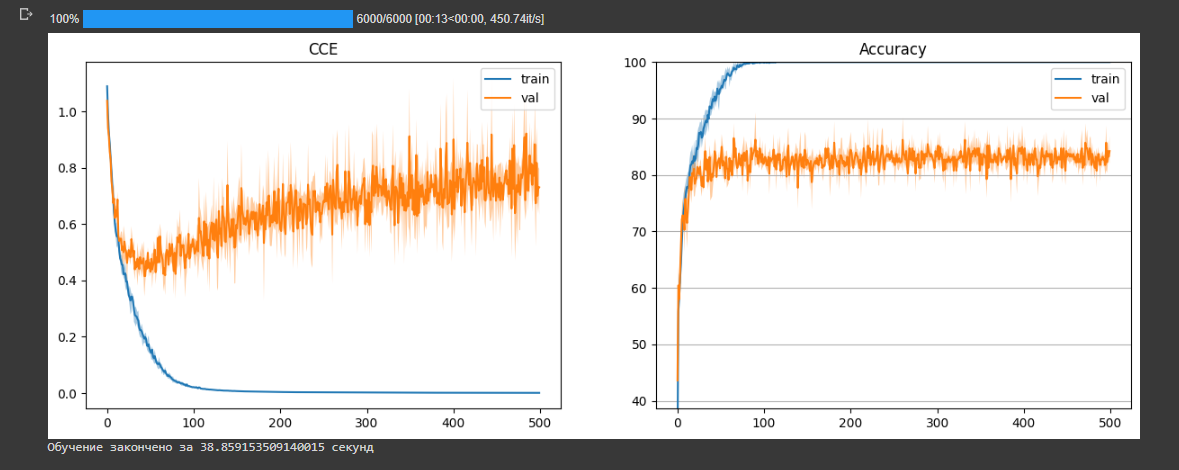
Далее идет код повторяющий первую лабораторную работу, который выводит результаты обучения нашей модели:

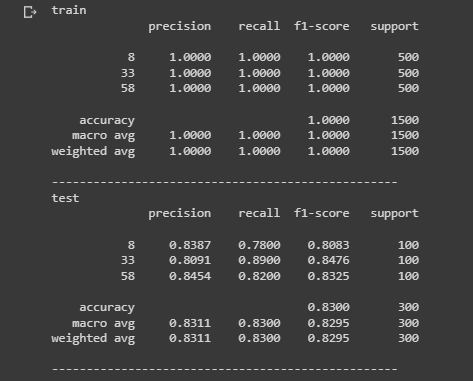


Это были результаты для усредняющего пуллинга.

Результаты для макс пуллинга:

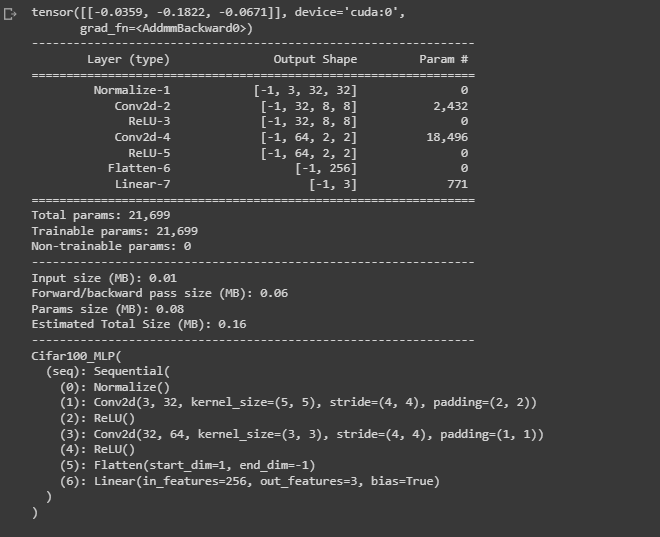


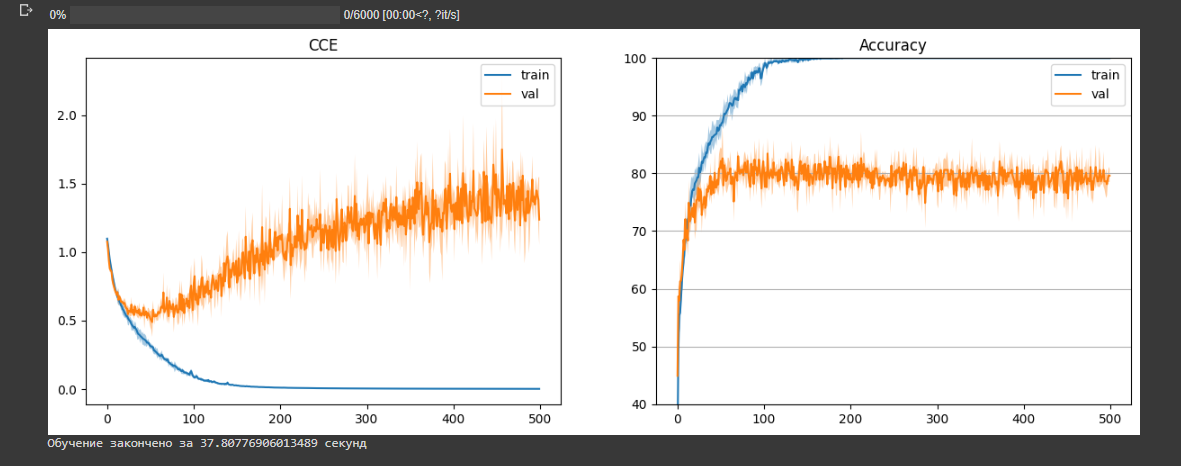
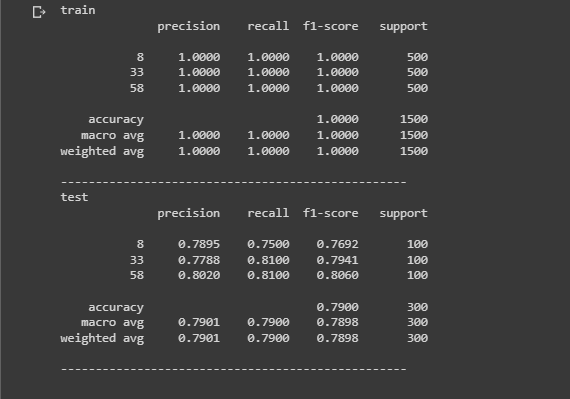




Это были результаты для макс пуллинга.

Результаты для пуллинга с помощью шага свёртки stride(4):

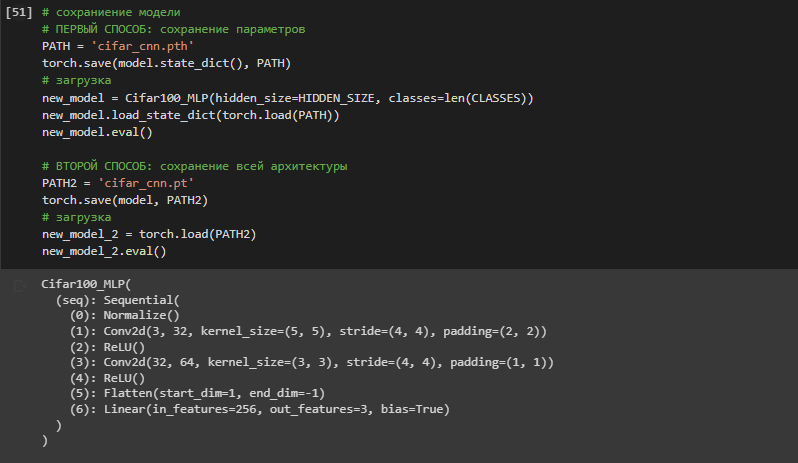


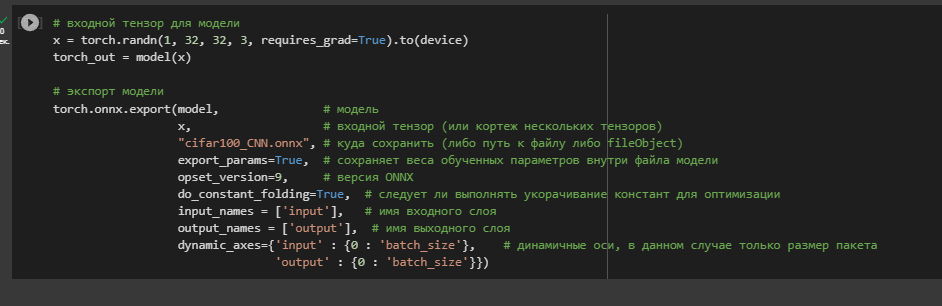
  


Это были результаты для пуллинга с помощью шага свёртки stride.

Из результатов видно, что все три вида пуллинга показали высокую точность на обучающих данных (100%), но различную точность на тестовых данных. Макс пуллинг показал самую высокую точность на тестовых данных (83%), затем усредняющий пуллинг (81%) и пуллинг с помощью шага свёртки stride (79%). Это может означать, что макс пуллинг лучше сохраняет важные признаки изображений и уменьшает переобучение модели. Усредняющий пуллинг также хорошо работает, но может сглаживать некоторые детали изображений. Пуллинг с помощью шага свёртки stride может быть менее эффективным, так как он не имеет отдельного слоя для регуляризации модели.

## **Часть 2 - Сохранение модели в ONNX**





Приведенный выше код позволяет нам скачать файл модели для ONNX в локальное хранилище google collab, откуда его можно скачать.

Результат работы нейронной сети.

