## Задание 1

Листинг программы в файле task1_generate_png.py. Результат работы в папке generated_images.

Из недочетов могу выделить следующие:

1. Не получилось реализовать отрисовку ромба. Вместо него квадрат. (не знаю в чем проблема, не получалось реализовать поворот квадрата)
2. Генерируемые треугольники всегда имеют одинаковые углы.

## Задание 2

Пытался сначала реализовать с использованием tensorflow (файл task2_nn.py) – что-то получилось, но не думаю, что корректные результаты (но я их приложил в качестве скринов).

Второй, итоговый вариант, с применением pytorch и предобученной модели resnet18 (Resnet'ы в целом подходят для такого задания). Так как, на данный момент dataloader и dataset в pytorch не работает самостоятельно (albumentations), необходимо самостоятельно написать класс для этого.

Для аугментации применял следующие методы:

1. Преобразование изображений в тензоры (Tensor).
2. Нормализация изображений с параметрами среднего (0.5, 0.5, 0.5) и стандартного отклонения (0.5, 0.5, 0.5).

Пытался применять и другие методы (поворот, градации серого, изменение яркости и подобные методы), но почему-то ошибка только увеличивалась.

Архитектура модели:

Модель использует предобученную ResNet18 в качестве основной части, а в верхних слоях имеет два полносвязных слоя:

1. Один для классификации объектов.
2. Один для определения границ объекта (bounding box).

Используется комбинированная функция потерь, состоящая из потерь для классификации (CrossEntropyLoss) и регрессии (SmoothL1Loss).

Полносвязные слои:

1. fc1: преобразует данные из 64x64x64 в 512.
2. fc2_class: получает на вход 512 признаков и выдает 4 выхода (один для каждого класса).

3. fc2_bbox: получает на вход 512 признаков и выдает 4 выхода для координат рамки.

## Результаты tensorflow

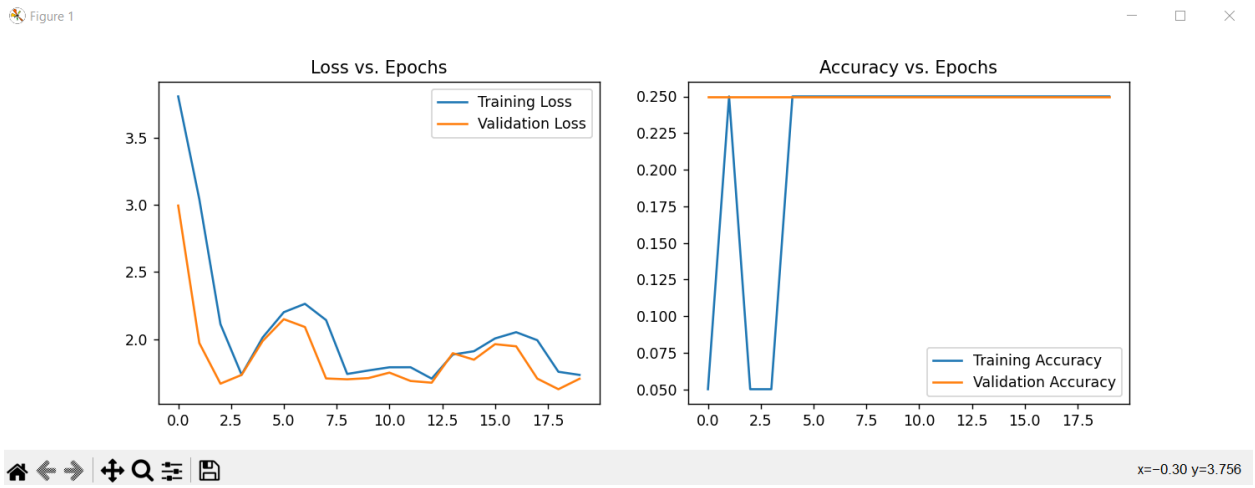


Рисунок 1 – обучение модели tensorflow (batch_size=32, lr=0.001, epoch=10).

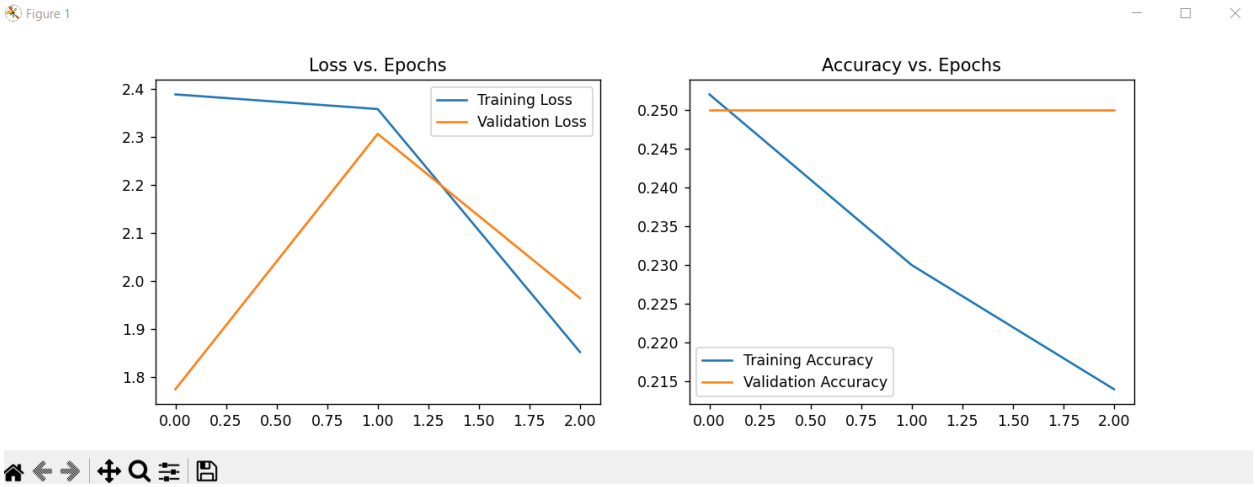


Рисунок 2 – обучение модели tensorflow (batch_size=32, lr=0.001, epoch=10).

На данном этапе увеличил размер выборки до 3 тысяч.

## Результаты PyTorch

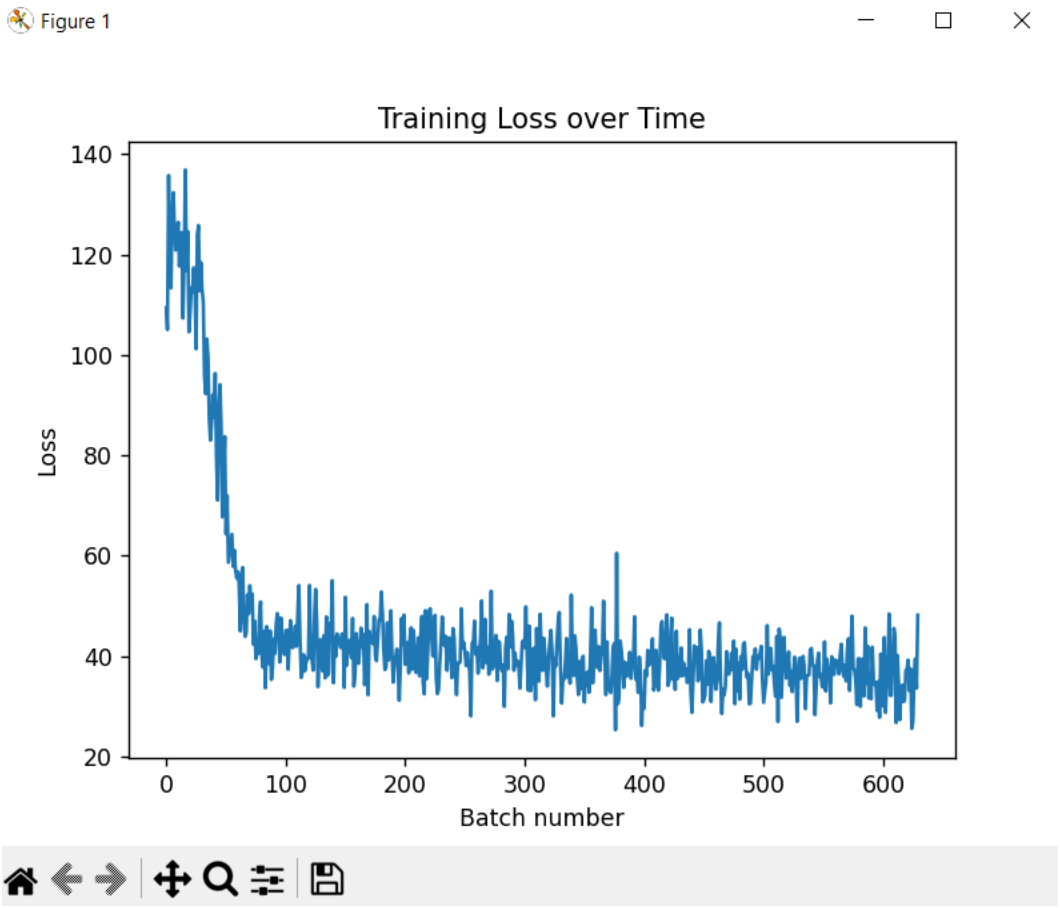


Рисунок 3 – Обучение tesnet18 (batch_size=16, lr=0.0005, epoch=10).

Данное обучение производилось без IoU и без промежуточных этапов, но с различными методами аугментации (RandomHorizontalFlip, RandomVerticalFlip, ColorJitter, RandomRotation).

```
Epoch 9/10, Batch 60/63, Loss: 39.5269660949707
Predicted class: circle
Predicted bbox: [89.7724609375, 95.4580078125, 203.3898468017578, 206.46920776367188]
```

Рисунок 4 – Результаты предсказания.

```
Intermediate Checkpoint:
Max IoU: 0.8692774772644043, Min IoU: 0.0, Avg IoU: 0.2683829069137573, Precision: 0.7597765363128491, Recall: 0.14211076280041798
Final Checkpoint:
Max IoU: 0.8692774772644043, Min IoU: 0.0, Avg IoU: 0.2683827877044678, Precision: 0.7597765363128491, Recall: 0.14211076280041798
```

Рисунок 5 – Результаты предсказания.


Рисунок 6 – График ошибки при обучении модели resnet18 (без дополнительных методов аугментации batch_size=32, lr=0.0005, epoch=10).

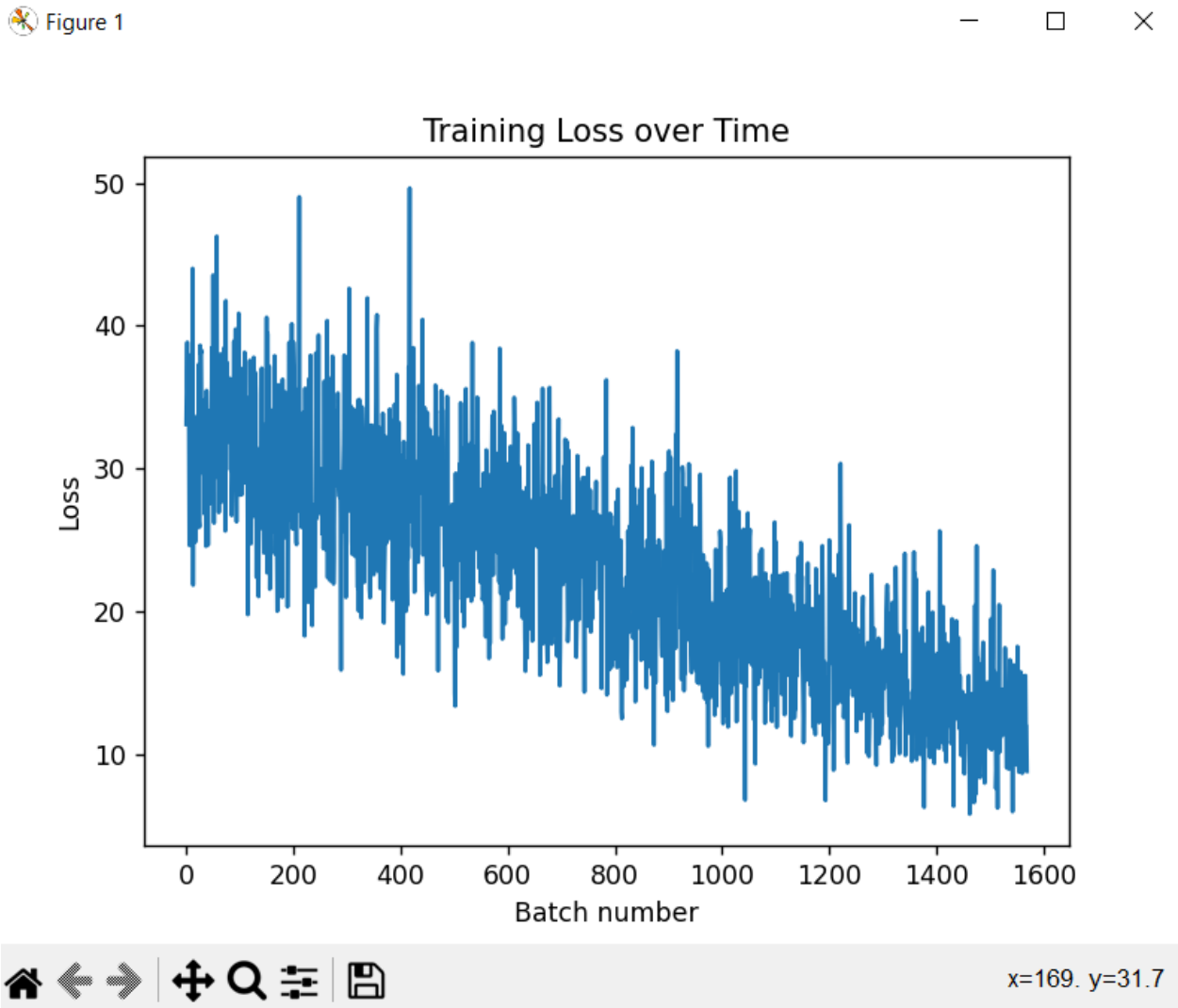


Рисунок 7 – График ошибки при обучении модели resnet18 (batch_size=32, lr=0.0005, epoch=10, 5000 фотографий).

```
Epoch 9/10, Batch 150/157, Loss: 8.894488334655762
Predicted class: circle
Predicted bbox: [88.06830596923828, 45.67429733276367, 227.14222717285156, 184.07455444335938]
```

Рисунок 8 – Результат тестирования модели resnet18.

P.S: Спасибо за предоставленную возможность.

Способы улучшения:

1. Дополнительная аугментация.
2. Изменение архитектуры.

3. Регуляризация (dropout или weight decay для предотвращения переобучения).

4. Обучение с ранней остановкой (валидационный набор данных, чтобы остановить обучение)

5. Подбор гиперпараметров.