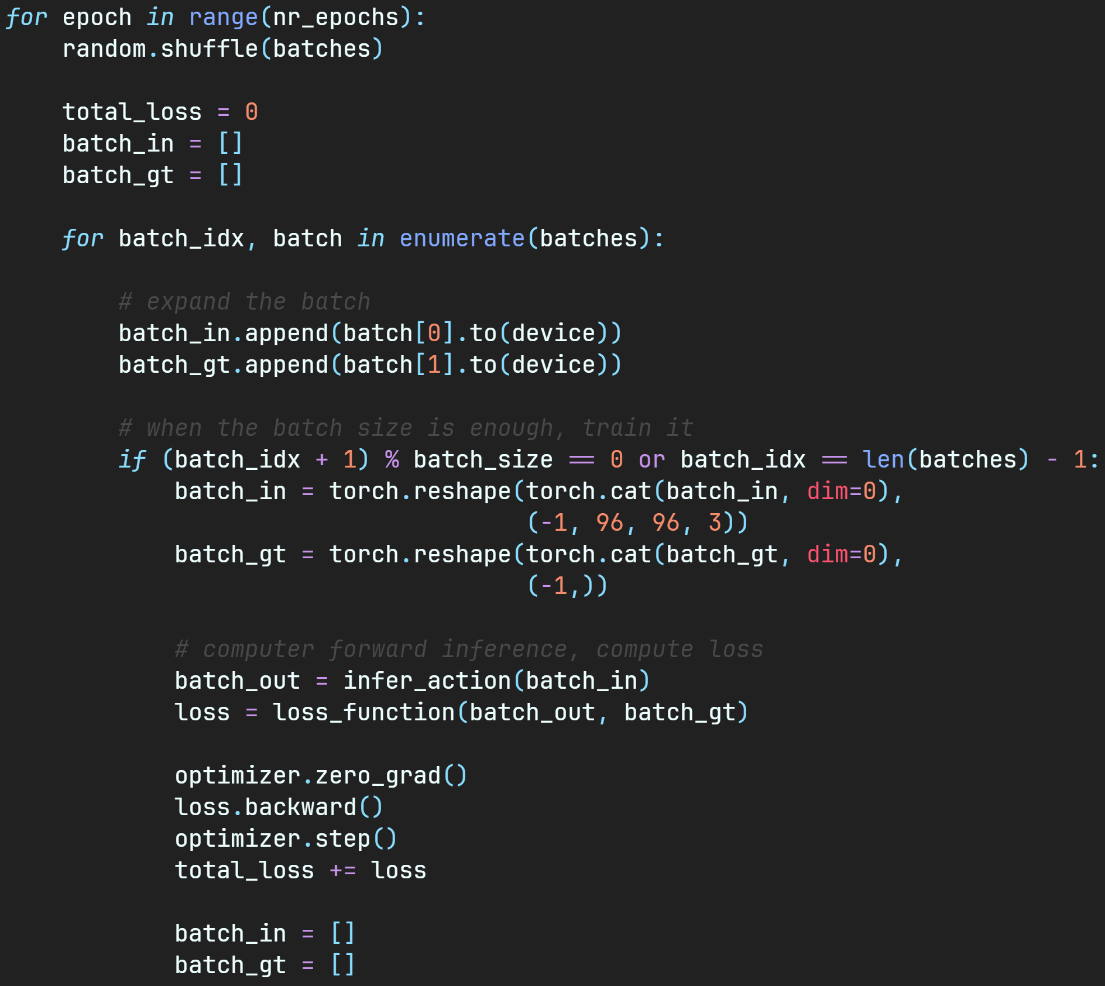
**고급소프트웨어실습I HW6 보고서**

**20211584 장준영**

**1. 데이터를 배치로 나누는 이유**

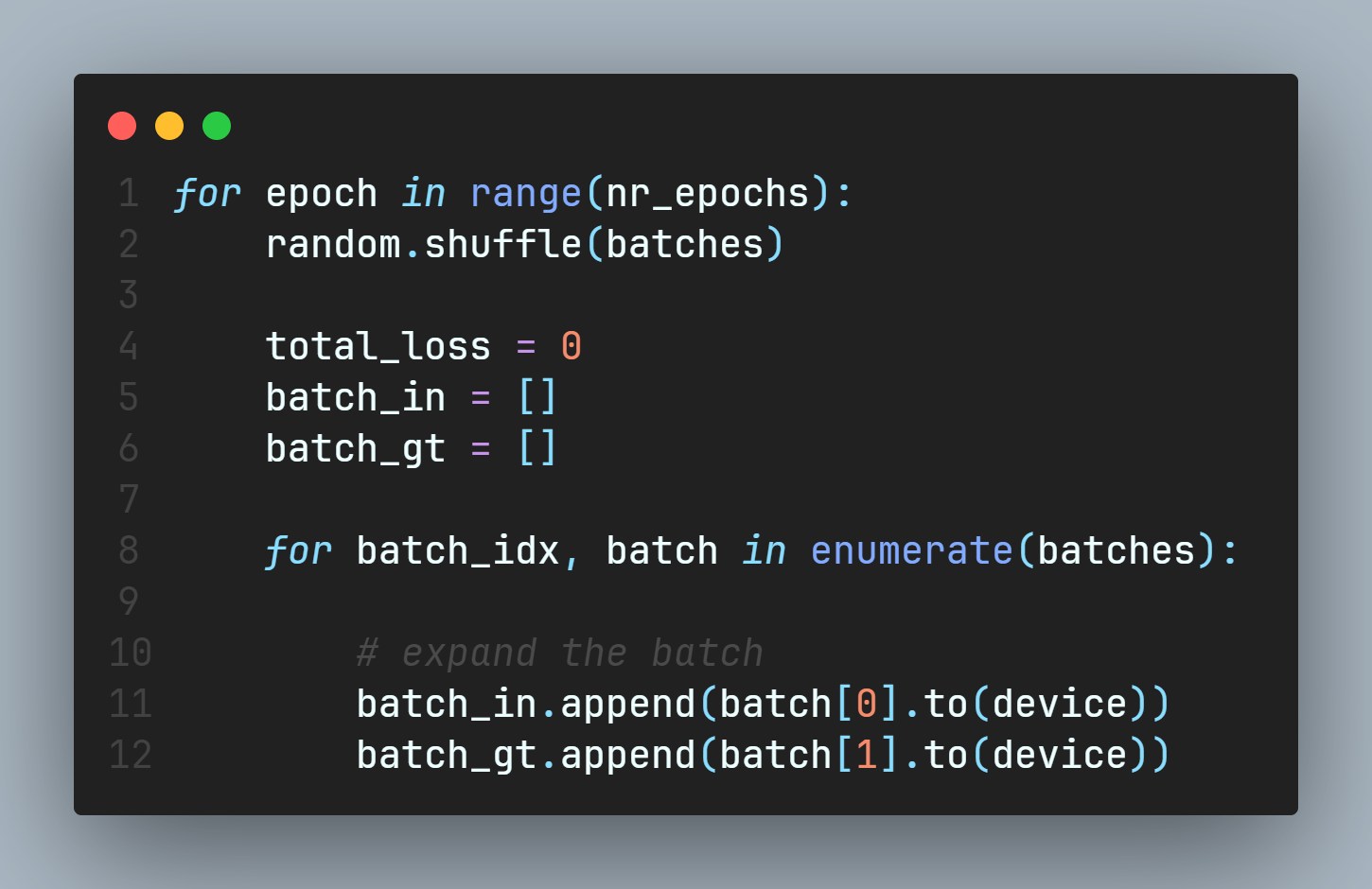
학습을 위해선 아주 큰 데이터셋이 필요하다. 대규모 데이터셋을 한 번에 메모리에 로드하고 전체 처리 하는 것은 컴퓨터의 자원을 매우 많이 소모하고 따라서 시간이 상당히 오래 걸릴 수 있다. 따라서 데이터를 작은 배치로 나누어 각 배치를 순차적으로 모델에 입력으로 제공하면 효율적인 학습을 할 수 있다. 메모리 사용이 최적화되고, 파라미터 업데이트가 배치 단위로 이루어져 안정적이다.



train.py에서 배치로 데이터를 나누는 부분이다. 먼저 batch\_in과 batch\_gt를 빈 리스트로 초기화하고, 앞서 zip을 통해 만들어둔 batches 리스트에서 각 배치를 순회하며 batch\_in과 batch\_gt에 해당 배치의 입력 데이터와 레이블을 추가한다. 만들어진 배치 단위의 데이터는 학습을 하는 프로세서 유닛으로 전달되고, 충분한 사이즈가 되면 batch\_in과 batch\_gt를 하나의 텐서로 결합하고(torch.cat) 크기를 조정한다(torch.reshape). 구성이 완료된 배치 텐서를 통해 학습 과정을 진행하고, 마치면 다시 batch\_in과 batch\_gt를 빈 리스트로 초기화한다. 이 과정을 반복하면, 큰 데이터셋을 배치 단위로 나누어 사용할 수 있다.

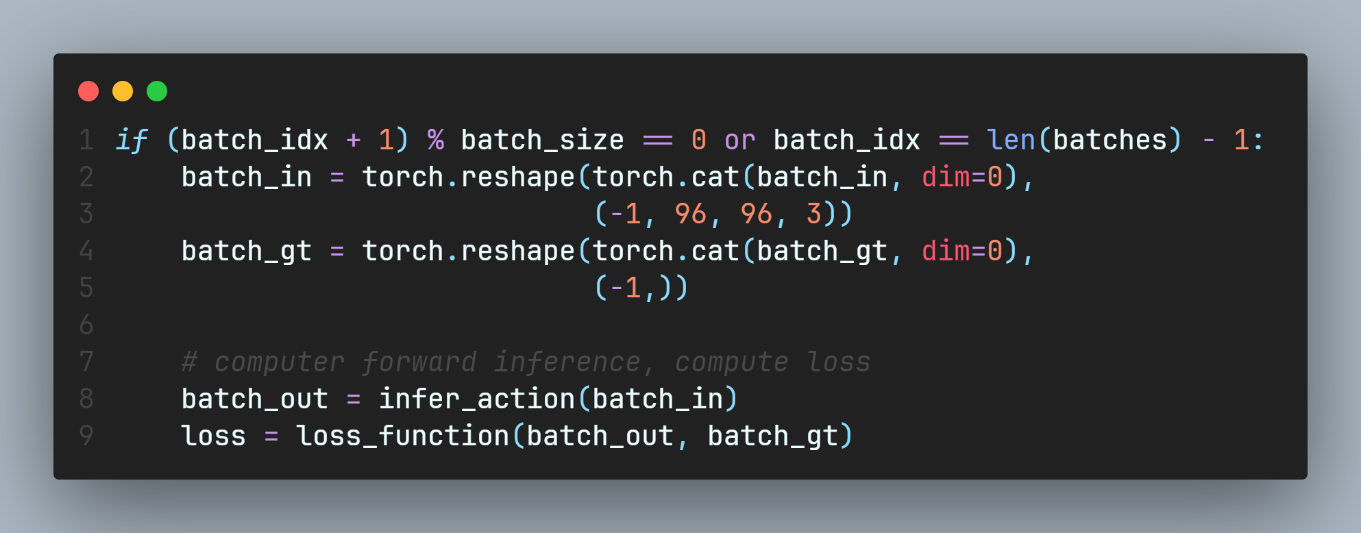
**2. Epoch(에포크)**

에포크는 전체 데이터셋이 구성된 신경망을 한 번 전체 통과하는 횟수를 말한다. 학습 과정에서 전체 데이터셋을 여러 번 반복하여 사용함으로써 모델이 데이터셋과 패턴 간의 관계를 학습할 수 있도록 한다. 일반적으로 에포크 수가 증가할수록 모델의 학습이 견고해지고 성능이 향상될 수 있지만, 특정 수치 이상을 넘기면 데이터셋의 노이즈까지 학습하여 과적합 문제가 발생할 수 있다.



앞서 확인한 batch loop 바깥에 epoch loop가 있다. nr\_epochs는 전체 학습 과정에서의 에포크 수를 나타내고, 해당 수만큼 반복한다. epoch loop 안에서는 데이터셋을 무작위로 섞은 후 각 배치에 대해 batch loop를 실행한다.

**3. Line 69~77**



앞서 **1.**에서 전부 설명한 내용이다. line by line으로 설명하면 다음과 같다.

- 조건문을 통해 정해둔 배치 크기에 도달하거나 batches 리스트의 마지막 배치인지를 확인한다. 이 경우 segmentation을 멈추고 학습 과정에 들어선다.

- batch\_in 리스트에 있는 입력 데이터 텐서들을 torch.cat으로 연결해 하나로 만든다. dim=0이므로 첫 번째 배치 차원을 따라 연결된다. 이후 torch.reshape로 텐서의 크기를 (batch\_size, 96, 96, 3)으로 조정한다.

- 바로 직전 라인과 동일하다.

- infer\_action 함수를 이용해 입력 데이터인 batch\_in에 대한 forward inference를 수행하고 출력 데이터인 batch\_out을 계산한다.

- loss\_function 함수를 이용해 출력 데이터인 batch\_out과 레이블 데이터 batch\_gt 간의 loss를 계산한다.

이후 optimizer의 그라디언트를 초기화하고 loss 값에 대한 backprop을 수행한다. 이후 업데이트된 그라디언트로 모델의 파라미터를 업데이트한다. 이러한 과정으로 모델의 학습이 진행된다.