**Dataset generation**

0. dataset generation 내 코드는 steam에 등록되어 있는 게임인 ARMA3가 있어야 실행시킬 수 있습니다.

1. 폴더 내 main을 실행하면 sqf\_generation이라는 서브 프로세스를 실행시킵니다.

2. 서브 프로세스는 ARMA3에서 제작한 가상 공간을 구현하고 랜덤하게 사람을 배치하고 시점을 변환한 후 각 사람의 headpoint를 리턴시키는 코드를 생성합니다. 코드는 ARMA3의 언어인 sqf로 생성됩니다.

2. main함수는 sqf 코드로 ARMA3를 실행시키고 opencv를 활용하여 화면을 캡처하는 방식으로 데이터 셋을 생성합니다.

3. 생성이 완료되면 프로그램을 종료하고 정해진 epoch만큼 반복하며 데이터셋을 자동으로 생성합니다.

**Model Architecture**

0. 좌측과 우측 RGB 이미지를 합쳐 6채널, 256x256 크기로 입력 데이터를 받습니다.

1. Residual Block 3개, downsampling 및 채널 확장(6->32->64->128), 최종 출력은 32x32 feature map, 128채널의 데이터를 내보냅니다.

2. (1)의 feature map을 flatten(B, 128x32x32) 후 Linear 레이어로 512차원으로 축소한 뒤, ReLU 활성화를 적용합니다.

3. 이후 2개의 분기를 나눠, 좌표와 신뢰도를 계산하게 합니다. coord\_head(B,max\_people\*2)와 conf\_head(B,max\_people)로 이루어지며, 두 head 모두 Sigmoid 활성화한 뒤 concat하여 (x,y,confidence) 형태의 벡터를 출력하게 합니다.

**Loss Function**

0. 좌표 손실(Coordination\_loss) : 예측 좌표와 GT 좌표 간 유클리드 거리를 계산합니다. 이후 헝가리안 알고리즘으로 최소 거리 매칭, 매칭된 좌표는 L1 Loss를 사용합니다. 최종적으로 평균 계산 후 최종 손실을 도출합니다.

1. 신뢰도 손실(Confidence\_loss) : max\_people의 좌표 일치 여부(1/0) 기준 BCE 손실을 계산합니다.

2, Total\_Loss = a \* coord\_loss + b \* conf\_loss (a, b is weight) 로 총 손실을 계산해 학습에 사용합니다.

**Output**

0. 예측한 사람 수와 좌표 기반 Top-view Density map을 생성합니다.

1. Threshold 값 변화에 따른 각 좌표의 confidence를 계산해 density map을 생성할 수 있고, 이번 프로젝트에서는 Threshold = 0.75로 설정해 진행했습니다.

**Demo Implementation**

**데모는 딥러닝 모델을 간단히 GUI로 depoly할 수 있는 Gradio라는 툴을 이용해 개발되었습니다.**

0. 모델을 불러옵니다.

1. Frontend UI에서 사용자는 좌/우측 이미지 각 1장과 confidence threshold를 입력합니다.

2. Backend에서는 이를 파라미터로 predict() 함수를 실행합니다.

3. predict() 함수는 load\_checkpoint() 함수로 학습된 가중치를 불러온 모델에 이미지를 forward하고, cords\_to\_density\_map() 함수가 model output을 사용해 density map과 count를 생성하여 frontend로 넘겨줍니다.

4. Frontend에서는 threshold에 따른 model inference 결과를 확인할 수 있습니다.