군 사 과 학 연 구 지
제15권 제1호(2022년 6월)

군사목적의 합성 데이터셋 제작을 통한 객체인식 효율성 제고 방안

A Study on Improvement of Object Recognition Efficiency by Generating Synthetic Datasets for Military Purpose

변재현[1)]
Jaehyun Byeon

ABSTRACT

Object recognition requires manual work to collect and annotate actual data, which requires lots of time and money. In the case of artificial intelligence in the defense field, it is difficult to secure high-quality learning data, and there is a problem of the absence of robustness. Synthetic data can be configured by machine learning experts and enables fast, data-intensive development. Synthetic data also helps solve privacy and security problems, and plays an important role in reducing bias by securing data diversity. This paper examines the problems of object recognition in the defense field and proposes a plan to improve the efficiency of object recognition through the generation of synthetic data as an alternative. To this end, a commercial open-source unity3d simulator was used to generate simple military-purpose synthetic data, and similar research cases are examined.

Keywords : synthetic data, object recognition, unity3d

논문접수일 : 2022년 5월 2일, 심사일 : 2022년 6월 6일, 게재확정일 : 2022년 6월 24일
1) 국방대학교 군사운영분석 석사과정

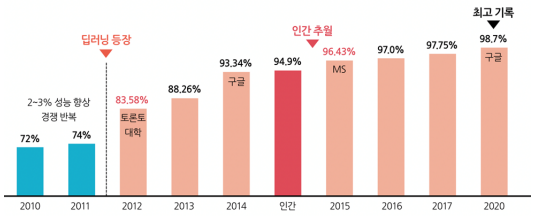
1. 서 론

객체인식은 실제 데이터를 수집 후 주석을 첨부하는 수작업이 요구되며, 이는 많은 시간과 비용이 필요하다. 국방 분야 인공 지능의 경우 고품질의 학습용 데이터의 확보가 어려우며, 강건성(Robust) 부재하는 문제가 상존한다. 성공적인 인공지능 객체인식을 위해서는 인공지능의 하위 분야인 기계학습(ML)을 통한 전문적인 컴퓨터 활용이 필수적이다. 합성데이터(Synthetic Data)는 머신러닝 전문가가 구성할 수 있으며, 빠르고 데이터에 집중한 개발을 가능하게 한다. 실제 데이터가 아닌 인공적으로 만들어진 자료를 사용하는 것은 혁신적 전환이기에 합성데이터는 4차 산업혁명의 기름과 같다. 또한 합성 데이터는 개인정보와 보안 문제 해결에 도움을 주며, 데이터의 다양성을 확보해 주어 편향(bias)을 줄이는데 중요 한 역할을 한다. 본 연구에서는 국방 분야 객체인식의 문제점을 살피고 이를 해결하기 위한 대안으로 합성데이터 생성을 통한 객체인식을 효율성 제고 방안을 제안한다. 이를 위해 상용 오픈소스 유니티 (Unity) 시뮬레이터 활용하여 간단한 군사 목적 합성데이터를 생성해보았으며, 유사 연구 사례 등을 살핀다.

2. 현 상황 및 문제점

2.1 객체인식 기술

객체 인식(Object Recognition)은 이미지나 비디오의 객체를 식별하는 컴퓨터 비전 기술이다. 객체 인식은 딥 러닝 및 머신 러닝 알고리듬을 통해 계산된 핵심 기술이다. 사람은 사진이나 비디오를 볼 때 사람, 물체, 장면, 시각적 세부 사항을 쉽게 인식할 수 있다. 컴퓨터를 이용한 객체 인식의 목표는 컴퓨터가 이미지에 포함된 것을 이해하는 능력과 같이 인간이 자연스럽게 할 수 있는 것을 하도록 가르치는 것이다. 특히 딥러닝을 활용한 인공지능 객체인식 기술은 대규모 이미지 인식 경진대회에서 (ILSVRC) 2015년 인간의 인식률 (94.90%)를 추월하여 2020년에는 98.7%가 넘는 정확도를 선보이고 있다.



<그림 1> 이미지인식 대회 정확도 향상>[1]

객체인식은 4차 산업혁명을 이끄는 핵심 기술 중 하나로 다양한 분야에서 활용되고 있다. 예를 들어 무인자동차가 정지신호를 인식해 보행자와 가로등을 구분할 수 있도록 하고, 바이오이미징에 질병 식별, 산업점검, 로봇비전 등 다양한 분야에 활용된다.

국방 분야에서도 인공지능 물체 인식이 결합된 최신 기술을 도입하기 위한 다양한 노력이 진행되고 있다. 최근 육군 교육사령부는 군사용 AI 이미지 데이터를 수집·처리할 수 있는 '군사 이미지넷' 플랫폼을 구축해 이날부터 지휘·군사학교용으로 운영한다고 밝혔다. 밀리터리 이미지넷은 이미지 세트를 만드는 프로젝트로 구축된 데이터 세트는 인트라넷을 중심으로 공유되며 보안규정에 저촉되지 않는 범위 내에서 산학관들과 공유된다. 육군이 별도 예산으로 이들 사업을 시행한 배경에는 인터넷상에 수많은 이미지가 있지만, 이대로는 군에 적용하기 어렵기 때문이다. 군 내부 인트라넷 내에 인터넷의 이미지넷과 유사한 시스템을 구축해 이미지를 수집하려는 프로젝트가 밀리터리 이미지넷 사업이다.

이 사업을 통해 지능형 식별모델을 개발, 컴

<그림 2> 밀리터리 이미지넷 사업

퓨터로 처리·분석할 수 있는 관련 정보 모음인 데이터 세트와 군사용 AI 개발 환경이 조성됐다. 이미지 분석을 통해 아군과 적의 무기 시스템을 인식할 수 있으며 적의 침투(침략), 로밍 탐지, 테러/재난 및 이상 탐지, 동료 정체성및 움직임 탐지 등 탐지부터 타격에 이르는 전체 군사 의사 결정 과정을 지원한다. 육군은 앞으로 인터페이스 수집 범위를 적의 영상·위성·음성데이터·해상신호 등으로 확대할 계획이다.

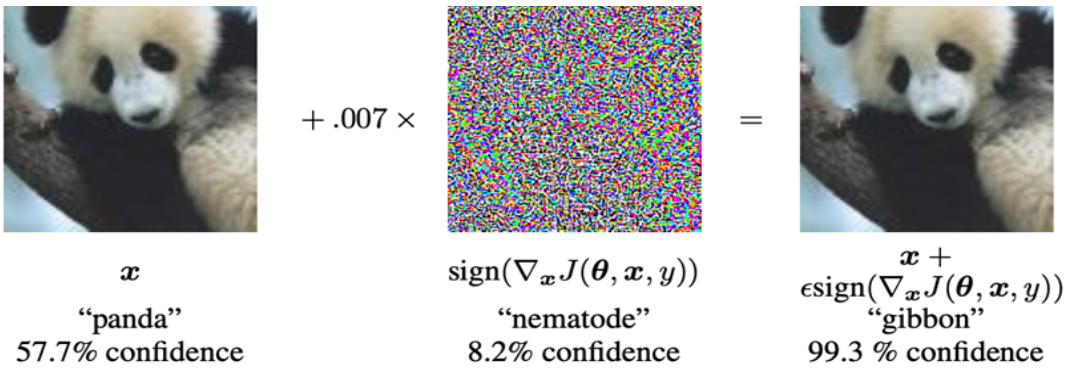
하지만, 밀리터리 이미지넷 사업의 경우 합성데이터가 아닌 실물을 촬영한 데이터로 프로그램을 구축하는 데 상당한 시간이 걸렸고, 현재보유중인 군장비 이미지 자료도 부족하다. 현재 우리 군은 국방개혁 추진의 기본방향으로는 무인화·지능화를 우선시하고 있다. 이는 카메라 센서를 통해 입력되는 데이터를 인식하고 결정하는 것으로 시작한다. 그만큼 인공지능 물체 인식은 한국군의 선진화를 이끌 핵심 기술임을 알 수 있다. 다만 인공지능을 이용해 인공지능 사물을 인식하는 데는 분명한 한계가 있다.

2.2 문제점

2.2.1 고품질 훈련자료 확보 어려움

딥러닝 모델은 주어진 입력에 대해 기대하는 결과를 출력하는 방법을 학습한다. 지도학습에서 레이블링이란 주어진 데이터에 정답지를 만들어주는 작업이고, 이때 정답지를 레이블이라고 한다. 레이블은 딥 러닝이 입력 공간의 연쇄 미분 변환으로 대상공간으로 사상하기 위한 가장 중요한 조건이다. 다만 실제 머신러닝 환경에서는 이 정도 수준의 학습 데이터를 준비하는데 많은 비용과 시간이 소요되고, 더 큰 문제는 라벨을 붙일 수 없는 AI 문제가 증가하고 있다는 점이다. 또한 지도 학습 기반 딥 러닝 기술을 사용하는 이미지 인식은 수 백장에서 수 백

<그림 3> 이미지 레이블링 예시



<그림 4> Advesarial Noise 사례 [3]

만장까지의 많은 학습 데이터를 필요로 한다.

전문가들은 국방 분야 AI 기술 활용 증가로 인한 '데이터 확보에 따른 보안 문제'를 가장 유력한 관심사로 꼽으며 향후 국방 AI 전력 향상에 필수적인 데이터 확보를 위해 까다로운 보안 절차에 대한 규제가 필요할 것으로 내다봤다.[2]

2.2.2 인공지능의 강건성(Robust) 부재

2018년 자율주행중인 테슬라 차량의 오작동으로 사망사고가 발생했다. 보행자 인식과 차선 인식에 각각 오류가 발생하면서 촉발된 사고였다. 인공지능의 활용도가 높아지면서 기술의 안정성과 신뢰성 확보가 선택이 아닌 필수다. 특히 딥러닝을 활용한 인공지능 객체 인식은 적대적인 예제(Adversarial Examples)라 불리는 이미지를 판단하는데 기술적인 한계가 존재한다. 누구나 이미지에 악의적인 적대적 노이즈를 임의로 주입하여 적대적 사례를 생성할 수 있다. 이러한 적대적 사례는 인간이 판단하는 데 문제가 없지만 인공 신경망의 판단을 방해한다. 이를 악용하는 것도 문제가 될 수 있지만, 이미 인공 신경망이 적대적 사례를 악용해 교통 표지판이나 사람을 정확하게 인식하지 못하는 사례를 찾을 수 있다.

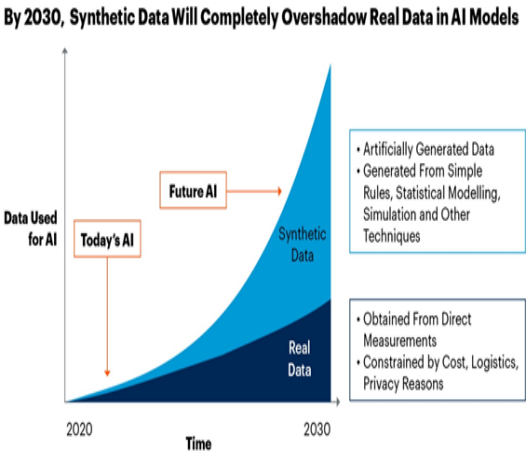
<그림 4>의 경우에서 볼 수 있듯이 처음에는 보통 57.7% 이상의 판다로 인식되던 인공신경망에 적대적 소음을 임의로 할당한 결과 긴팔원숭이로 판정된다. 이와 같은 환경외란에 의한 변동요인(noise)에 취약한 문제를 개선하기 위해 앞으로 많은 강건한 인공지능(Robust AI) 연구들이 필요하다. 그리고 이러한 문제를 해결하기 위한 방법으로 합성 데이터를 생성하고 학습 데이터로 활용하는 대안이 있다.

3. 합성데이터 및 관련연구 소개

3.1 합성 데이터

오늘날의 인공지능(AI) 시대에 데이터는 새로운 석유 자원과 같다. 그러나 데이터 소스는 최근 저작권, 지적재산권 보호 및 보안과 같은 소수의 영역에서만 자유롭게 사용할 수 있다. 그래서 최근 인공지능분야에서는 합성 데이터(Synthetic Data)라 부르는 저비용으로 효과적인 연료를 자체 생산하려는 노력이 이루어지고 있다.

합성데이터는 컴퓨터 시뮬레이션이나 알고리즘에 의해 생성된 정보를 실제 데이터를 대체하는 것이다. 즉, 합성 데이터는 실제 환경에서 수집되거나 측정되지 않고 디지털 환경에서 생성된다. 실제 사물, 사건, 사람을 기반으로 얻은 데이터보다 합성 데이터가 AI 모델 훈련에더

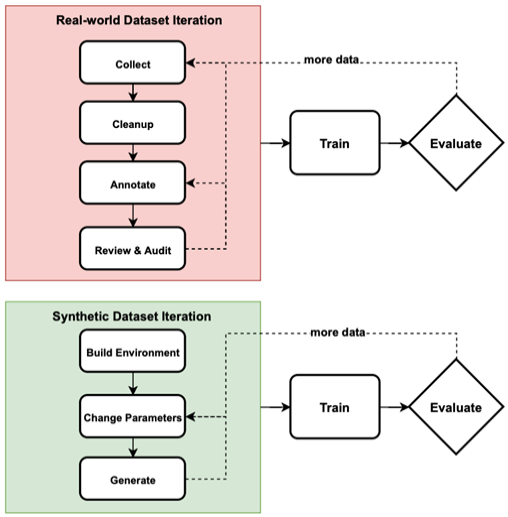


<그림 5> 합성데이터가 주도하는 AI시장 [5]

적합하다는 해외연구 결과가 나왔다.[4] 미국 정보기술(IT) 연구·자문업체 가트너는 2030년이면 AI에 사용되는 데이터 대부분이 시뮬레이션 등을 통해 인공적으로 만들어 합성데이터가 차지할 것으로 전망했다. 보고서는 "합성 데이터를 사용하지 않고서 고품질의 고부가가치 AI 모델을 만들 수는 없을 것"이라고 밝혔다.

합성데이터의 중요성은 이제 막 사람들의 인식에 각인되었고, 미 육군 역시 이러한 최신 추세를 쫓고있다. U.S Army SBIR|STTR은 미 육군이 운영하는 연구기관으로 육군을 현대화하고 생명을 구하는 기술을 병사들에게 이전하기 위해 중요한 우선 사항을 충족하기 위한 가장 새로운 기술 솔루션을 연구하고 있다. U.S Army SBIR|STTR는 최근 공개합성데이터 생성을 위한 연구 용역금액 20억원 규모의 모집공고를 인터넷에 공지하였다.[6]

<그림 6>의 상단에서 볼 수 있듯이 실제 데이터 집합에는 최소한 수집, 정리, 주석 달기, 마지막으로 검토 및 감사 단계가 필요하다. 이러한 단계에는 비용과 시간이 많이 소요되는 인적 노력이 수반된다. 반면 하단의 합성 데이터 세트를 사용하려면 3D 자산으로 구축된 환경, 랜덤화 매개 변수 설정 또는 변경 및 새로운 데이터 생성을 위한 환경 실행이 필요하다.



<그림 6> 데이터 수집방안 비교 [7]

데이터셋에는 정확한 주석이 포함되어 있으며 자동으로 검증되므로 시간이 많이 걸리는 대부분의 단계가 필요하지 않아 경제적이다.

3.2 연구사례

최근 몇 년간 합성데이터 (Synthetic-Data)를 국방분야에 적용하고자 하는 시도가 있었다. 양훈민(2019)은[8] 적대적 훈련 신경망에 대한 기술적 분석을 수행하고 경쟁 벤치마크 데이터 세트를 사용하여 탱크, 항공모함, 적외선 이미지 등의 방어 복합 이미지 데이터를 생성하는 실험을 수행했다. 해당 연구에서 생성된 합성 이미지는 머신러닝을 이용하여 국방 R&D 분야의 학습 자료로 활용될 수 있음을 확인하였다. 또한 아울러 초대용량의 학습용 데이터를 효율적으로 처리하기 위한 빅데이터 분산 저장 및 처리 기술을 적절히 활용하면, 대규모 머신러닝용 데이터 처리시스템 구축도 가능함을 확인하였다.

조선영(2021)은 [9] 불명확한 데이터를 합성·생성하여 부분적으로 가려진 물체를 감지하는 모델을 학습하는 방법을 제안했다. 다양한 가려짐의 상황을 고려하기 위해, 일정부분 식별이

제한된 특이점을 가진 데이터를 합성하고 생성하고, 이를 이용한 모델 학습을 통해 부분적으로 가려진 물체의 감지 성능을 향상시키는 방법을 제안하였다. 다양한 애매한 상황을 고려하기 위해 수준과 유형에 따라 합성 데이터가 생성되었다. 성능을 평가하기 위해 실제 군용 차량에 대한 데이터 세트를 수집하고, 합성 데이터를 생성하여 모델 학습에 사용했다. <그림 7>에서 보는 바와 같이, 다양한 실험을 통해 합성 데이터를 사용하여 학습한 모델은 부분적인 가려진 물체 감지성능을 향상시키는 것으로 나타났다.

<그림 7> 합성데이터를 통한 강건성 확보

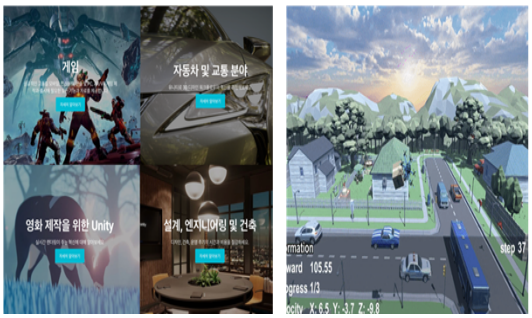
인공지능(AI) 연구가 발전하려면 현재의 AI 모델 학습용 벤치마크를 사용하여 기존 환경에 존재하는 어려운 문제를 해결해야 한다. 하지만 이러한 문제가 '해결'되고 나면 새로운 환경의 필요성이 대두되곤 한다. 하지만 그러한 환경을 조성하려면 많은 시간과 전문적이고 특별한 지식이 필요한 경우가 많다. 인공지능 연구의 핵심이 되는 '환경구축'의 문제를 해결 할 수 있는 하나의 대안이 유니티(Unity)다.

4. 합성데이터 구현

4.1 상용 프로그램 '유니티'

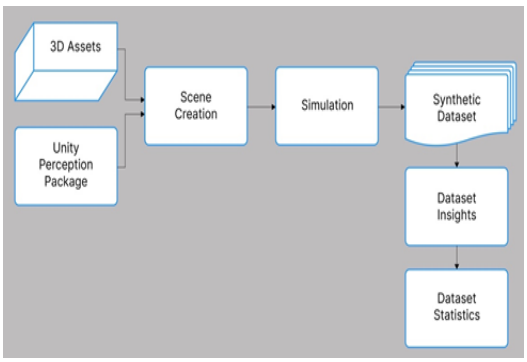
인공지능 분야를 크게 발전시킨 머신러닝(ML)을 통해 수많은 혁신 제품을 상용화되었다. 그러나 여전히 지도학습에서 머신러닝은 수집 비용이 많이 드는 크고 복잡한 데이터 세트를 필요로 한다. 라벨링 품질을 보장하고, 훈련 데이터가 프로덕션 데이터를 대표 하도록 해야 하는 등 여러 가지 문제도 상존한다. 이 문제는 대규모 합성 데이터 세트를 생성하여 해결할 수 있으며 특히 객체 감지 작업에서 효과적이다. 유니티 리얼엔진 시뮬레이션 툴을 활용하면 이러한 장점을 가진 합성데이터 세트를 생성하기 위한 좋은 환경을 구축할 수 있다.

유니티는 3D와 2D 비디오 게임의 개발 환경을 제공하는 게임 엔진이다. 주로 3D 애니메이션, 건축 시각화, 가상현실(vr) 등 대화형 콘텐츠 제작에 활용되고 있으며, 4차 산업혁명 시대 화두인 디지털-트윈을 구현하기 좋은 프로그램이다. 게임업계 뿐만 아니라 업계와 학계에서도 큰 인기를 끌고 있다. 게임 엔진 시장의 45% 이상을 차지하고 있으며, 등록 개발자 수만 500만명이 넘는다. 유니티는 물리 법칙을 구현 하고 다양한 3D 모델과 기능을 구매하며 비교적 단순하게 환경을 조성할 수 있다는 장점이 있다.



<그림 8> 프로그램 소개(좌), 예시(좌)

그리고 유니티는 기계학습을 개발하기 위한 훌륭한 패키지를 무료로 제공하고 있으며, 이미 산업현장 및 학계에서 뜨거운 관심을 받고 있다. 유니티 머신 러닝 에이전트를 활용하여 구



<그림 9> 유니티 Perception 패키지

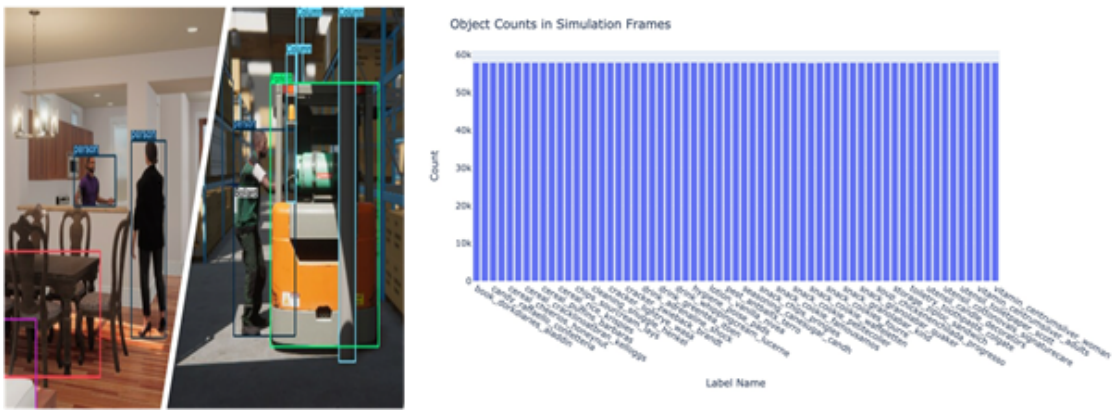
현된 환경에 기계학습의 다양한 알고리즘을 적용할 수 있다. Unity와 ML-Agents 툴킷을 사용하면 풍부한 물리적, 시각적, 인지적 요소를 갖춘 AI 환경을 조성할 수 있다. 새로운 알고리즘과 메서드의 연구는 물론이고 벤치마킹에도 이러한 환경을 사용할 수 있다.

앞서 합성 데이터의 장점이 충분히 열거하였지만, 이를 도입하여 기계 학습 모델을 훈련시킬 수 있는 데이터 세트를 생산하기 위해서는 몇 가지 중간 단계가 필요하다. 이 과정에서 프로그래머들은 공통문제에 직면하여, 기계학습 모델을 훈련시키기 위해 대체로 품질이 떨어지는 일회성 맞춤형 솔루션을 활용을 강요받는 상황에 봉착한다. Unity는 불필요한 단계를 거치지 않고도 고품질 합성 데이터 세트를 쉽게 생성하고 분석할 수 있는 통합 인식 패키지 및 데이터 세트 분석의 인사이트를 제공한다.

4.2 시뮬레이션 활용의 장점

유니티의 컴퓨터 비전 데이터 세트는 도메인-랜덤화 기술을 사용하여 애플리케이션의 품질을 개선하고 편견을 제어하는 다양한 데이터 세트를 생성한다. 이 프로세스를 통해 대상 물체의 위치 및 방향, 조명 및 카메라 각도 등 조합할 수 있는 수많은 구성 순열을 유니티 환경으로 출력할수 있다. 이를 통해 인터넷에서 촬영한 실제 인물·장소 이미지를 직접 촬영하거나 포함하는 과정에서 발생할 수 있는 개인정보 침해와 예방이 어려운 편향 문제도 피할 수 있다.

유니티 Perception 패키지를 이용하면 가상 데이터 세트를 생성하는 워크플로우를 구현할 수 있다. 2D 바운딩 박스나 시멘틱 세그멘테이션 마스크와 같은 실측데이터를 생성한다. 생성된 실측 데이터는 관련 메트릭스와 함께 JSON 파일로 캡처된다. 또한 데이터 분석 툴도 함께 제공하는 것은 또 다른 장점이다.



<그림 10> 합성데이터 생성 (좌), 분석 (우)

성공적인 이미지 객체 인식을 위해 기계학습 전문가의 주요 과제는 레이블이 지정된 데이터를 연구하고 분석하는 동시에 데이터 세트를 구축하는 것이다. 다만 합성 데이터로 작업할 경우 클라우드기반 시뮬레이션 실행을 통해 수만 개의 이미지가 생성되어 컴퓨터 자원의 소모가 빠르게 커질 수 있다. 유니티는 Python 패키지인 Dataset Insights를 제공하여 통계를 단순하고 효율적으로 계산하고, 대규모합성 데이터 세트에서 필요한 정보를 추출하고, 전체 데이터 세트에 걸쳐 집계된 통계를 시각화하는데 용이하다.

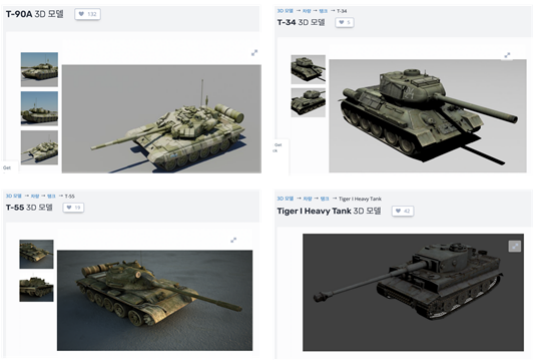
4.3 군사목적 합성데이터 생성

유니티의 asset store에서 프로그래밍의 목적에 맞게 획득하여 사용함으로써 직접 에셋을 개발하는데 들어가는 시간을 줄일 수 있다. 이를 활용하여 <그림 11>과 같이 비교적 간단한 합성데이터 생성을 실시하였다.

<그림 11> 합성데이터 생성의 예시

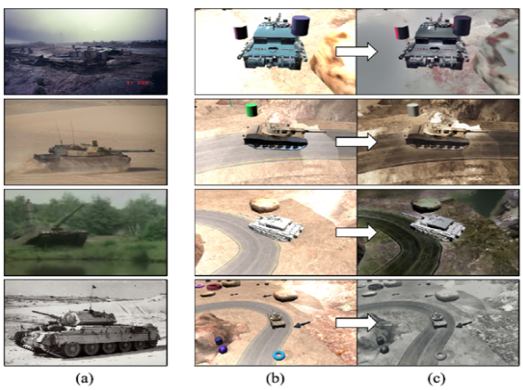
<그림 11> 상단과 같이 3개를 생성한다. 그리고 3개의 에셋에 대해서 작성한 스크립트의 알고리즘으로 다양한 색깔 및 카메라 각도에서 생성한다. 이는 유니티에서 합성데이터 생성의 기본적인 이해를 돕기 위해서 준비한 자료로 물론 더 복잡한 환경에서 더 많은 객체들을 생성할 수 있다.[2)]

이 방법 외에도 무료로 다양한 이미지를 제공하는 사이트를 활용하여 기초 데이터를 생성할 수 있다. 다운받은 자료로 위에서 제시한 방법과 동일하게 합성데이터를 생성하면 된다.



<그림 12> 무료 T-계열 전차 Asset [10]

합성데이터를 생성하여 경제성을 제고시키고자 하는 관련 연구들이 진행중에 있으며 <그림 13>은 전차 객체를 3D로 생성하여서 다양한 환경에서 접목시킨 예시이다. 해당연구는 합성 데이터에 FastPhotoStyle을 적용하였으며, (a) 스타일 이미지(실제 데이터), (b) 콘텐츠 이미지(원래 합성 데이터), (c) 스타일화된 이미지(FastPhotoStyle)를 나타낸다.



<그림 13> 군사목적 합성데이터 해외 논문 [11]

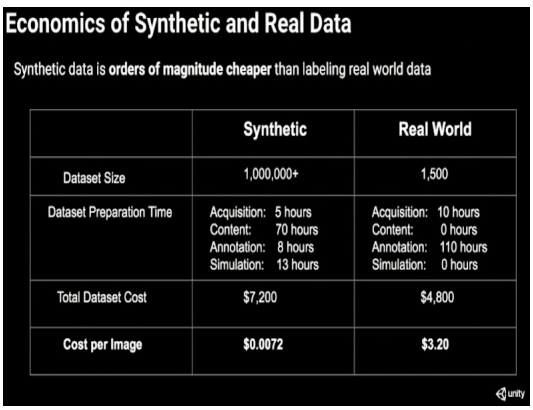
2) 보다 자세한 내용은 붙임1,2에 첨부한 C#스크립트를 참고

합성 데이터가 보다 사실적으로 스타일화된 방식으로 변환되면 데이터 분포가 충분히 실제 데이터와 가까운지 여부를 조사할 수 있다. 이처럼 합성 탱크 이미지의 분산범위를 넓히기 위하여 다양한 산만기(distractor)를 활용할 수 있다. 각 장면은 카메라의 여러 거리, 각도 및 높이에서 촬영된다.

4.4 기대효과

이미지 객체 인식을 위한 신경망을 훈련하려면 신중하게 레이블링된 대규모 데이터가 필요하다. 데이터를 다양하게 훈련할수록 더 정확한 AI 모델을 만들 수 있다. 하지만 수천에서 수만개의 요소를 포함하는 데이터 세트를 수집하고 레이블을 지정하는 데 시간이 오래 걸리며, 많은 비용이 소요된다. 반면에, 추정치에 따르면, 라벨링 서비스가 3달러의 비용이 드는 단일 이미지를 인위적으로생성하는 경우, 복합 데이터는 $0.0072의 비용으로 충분하다.

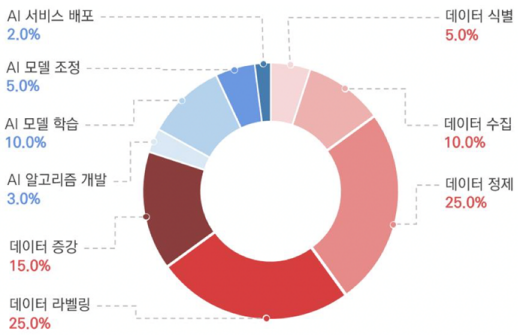
또한 합성 데이터는 개인정보보호 문제를 해결하고, 현실을 잘 대변할 수 있는 데이터의 다양성을 확보해 편향(bias)을 줄이는 데 중요한 역할을 할 수 있다. 더욱이 합성데이터는 자동으로 레이블링 되어 인간의 수작업보다 훨씬 더 빠르게 데이터셋을 조작할 수 있다.



<그림 14> 합성데이터의 경제적 이익성 [12]

이처럼 실제 데이터가 아닌 합성데이터를 이용하여 컴퓨터 비전 시스템을 구축하는 시도는 혁신적인 방법이다. 실제 데이터는 수집하고 주석을 다는 등 번거로운 작업을 해야 확보할 수 있는 반면, 합성 데이터 세트는 머신러닝 엔지니어가 (군 전문특기 장교 등) 홀로 몇 분 이내에 구성할 수 있으며, 더 빠르고 데이터에 집중한 개발 주기를 가능한다.

AI기술이 접목된 프로젝트에 소요되는 대부분의 시간은 수집, 정리, 분석, 시각화,데이터 구축과 정제까지가 주요 병목 현상을 만들고 있다. <그림 15>에서 확인할 수 있듯이, AI·데이터 기업에서는 데이터 구축 시 데이터의 종류에 따라 다르지만, 수집-정제 및 라벨링 작업에 30%~80%의 비용이 소요됨을 알 수 있다. 평균 75%의 비용이 데이터 수집·정제·라벨링에 소요되고 있다. 이 문제는 합성데이터의 장점으로 인공지능(AI) 프로젝트 작업시간을 단축하여 해결할 수 있다.

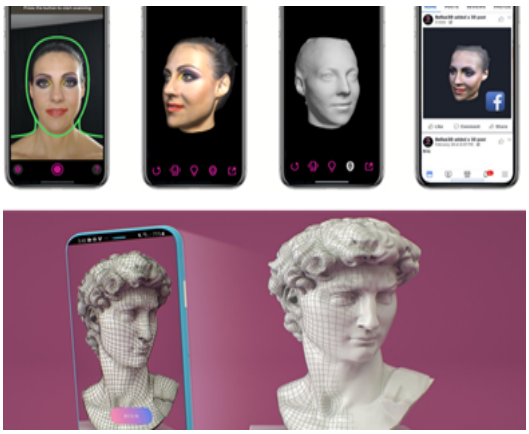


<그림 15> AI프로젝트에 소요되는 시간 [13]

4.5 제한사항 및 극복방안

하지만 여전히 데이터 수집은 어려운 과제로 남겨져 있다. 수천 개의 이미지와 주석 대신, 수십 또는 수백 개의 3D 에셋(Mesh, Texture, Animation 등)을 확보하는 노력이 요구된다. 즉 데이터 생성 최초 단계에 학습 대상물의

<그림 16> 3D이미지 생성의 예시



<그림 17> 상용 APP을 활용 3D에셋 획득

3D데이터 원본이 필요하다. 물론 인터넷상에서 이미 구축된 자료를 사용할 수 있지만 우리 군에서 바로 적용하기 위한 북한군 전투장비 및 시설 등에 대한 자료는 별도로 제작해야 할 것이다. 합성데이터를 구축을 위한 3D 에셋을 확보에 창의적인 수집이 필요한 이유다.

이를 해결하기 위한 몇 가지 방법이 있는데, 한 가지 예는 만약 현실세계에 존재하는 사물을 직접 볼 수 있다면 훨씬 정밀한 3D객체를 만들 수 있다. 군사목적의 합성데이터 생성을 위해서는 이처럼 해결해야 하는 과제가 남아있다. 가상의 환경에서 바로 3D객체를 생성할 수 있다면 좋겠지만, 조금더 정밀하며 효율적인 이미지 객체 인식을 위해서는 위 방법처럼 직접 실물객체의 사진을 촬영하여 이를 3D화 시키는 방법을 사용하면 효과적이다.

또는 상용화되어 있는 3D스캐너 앱을 통해 사진을 촬영하면 <그림 17>에서 보듯이 입체적인 3D에셋을 얻을 수 있다. 이렇게 획득한 최초의 데이터를 앞서 기술한 합성데이터 생성 방법을 통해서 다양한 환경에서 다양한 각도, 채광, 음영 조건을 변경하여 생성할 수 있다.

5. 결 론

합성데이터는 인공지능 시대의 '새로운 석유'라는 찬사처럼 그 활용분야는 무궁무진하다. 그리고 이를 자유롭게 생성하고 변형 및 응용할 수 있는 원천기술은 곧 향후 방위력개선사업의 진행에 반드시 필요한 핵심기술이 될 것이다. 밀리터리 이미지넷 사업의 경우 합성데이터가 아닌 실제 현실의 객체를 촬영한 데이터로 프로그램을 구축하느라 전력화까지 상당한 시간이 소요되었고, 현재 보유중인 군사장비 이미지의 데이터도 현격히 부족한 실정이다.

한국군의 무기체계개발은 [모방생산- 역설계 - 추격형 - 독자개발]의 과정을 거치면서 발전하였다. 핵심기술을 자체적으로 개발한 것은 2000년대 이후부터이다. 향후 펼쳐질 전장환경에서는 새로운 형태의 핵심기술들이 적용되고 무기체계로 전개될 것이다. 이러한 전장환경에서는 남의 것을 뒤따라 하는 무기체계의 개발 방식으로는 기대하는 방위력을 확보하기란 지극히 어렵다.[14]

새롭고 혁신적인 무기체계를 개발하기 위해서는 기존의 무기체계가 기반하고 있는 핵심기술보다 더 새롭고 그래서 기존 기술을 훨씬 뛰

어넘거나 와해시키는 수준의 핵심기술 확보가 필수적이다. 그런 의미에서 합성데이터를 활용한 인공지능 객체인식 기술은 향후 전장의 판도를 좌우할 게임체인저(Game-Changer) 기술이다.

본 연구에서는 국방 분야 객체인식의 문제점을 살피고 이를 해결 하기 위한 대안으로 합성데이터 생성을 통한 객체인식을 효율성 제고 방안을 제안하였다. 합성데이터의 장점을 극대화하려면 이를 잘 생성해야 한다. 그리고 연구에서 제시한 상용 오픈소스 유니티 (Unity) 시뮬레이터 활용하면 충분히 의미 있는 군사목적 합성데이터를 생성할 수 있을 것이다.

합성데이터(Synthetic-Data) 활용에 대한 본격적으로 산・학・연의 관심을 받기 시작한 것은 10여년의 짧은 기간이다. 그 이후 해당 기술은 눈부신 속도로 발전하고 있으며, 앞으로도 그러할 것이다. 이런 최신기술을 접목한 무기체계의 경우 소요제기 후 구매/양산까지의 긴 시간이 지나면 곧 시대에 뒤떨어진 골동품으로 전락할 수 있다. 하지만 밀리터리 이미지넷과 같이 소프트웨어를 기반으로 하는 경우 '진화적 ROC' 의 적용이 가능하다.

이처럼 합성데이터를 접목한 무기체계가 개발된다면, 현 작전운용성능의 경직성 문제를 탈피하고 신기술을 무기체계에 용이하게 적용 가능하며, 소요군이 장비를 운용하면서 경험을 환류시켜 최종 요구능력을 점진적으로 달성할 수 있을 것이다.

참 고 문 헌

[1] 이주열, 인공지능 이미지 인식 기술동향, TTA 저널 187호 ('20.2)

[2] 윤정현, 국방 분야의 인공지능 활용성 제고 방안과 시사점 과학기술정책연구원 FUTURE HORIZON :2020 제48호('20.12.)R. Bellman. Introduction to Matrix Analysis, 2nd Ed., pp. 234. McGraw-Hill, New York, 1979.

[3] Ian J. Goodfellow et al, Explaining and Harnessing Adversarial Examples (Published as a conference paper at ICLR 2015)

[4] Jonathan Tremblay, Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects (Conference on Robot Learning (CoRL),2018)

[5] Leinar Ramos, Forget About Your Real Data - Synthetic Data Is the Future of AI (Gartner Research, '21.6.24.)

[6] 출처 : U.S Army SBIR|STTR (https://www.armysbir.army.mil/topics/sensor-synthetic-data-generation/)

[7] Steve Borkman et al. Unity Perception: Generate Synthetic Data for Computer Vision (Unity Technologies, '21.7.)

[8] 양훈민, 국방용 합성이미지 데이터셋 생성을 위한 대립훈련신경망 기술 적용 연구, Journal of the KIMST, Vol. 22, No. 1, pp. 49-59, 2019.

[9] 조선영, 합성 데이터를 통한 부분 가려짐에 강인한 군용 차량 검출, KIISE Transactions on Computing Practices, Vol. 27, No. 11, pp. 519-530, 2021. 11

[10] 출처 : https://free3d.com

[11] Agency for Defense Development, Applying FastPhotoStyle to Synthetic Data for Military Vehicle Detection (ICCAS 2020)

[12] 출처 : Nvidia 공식홈페이지 (https://blogs.nvidia.co.kr/2021/08/30/what-is-synthetic-data/)

[13] Chanju Park, Dongsu Kang, "A DOM-Based Fuzzing Method for Analyzing Seogwang

Document Processing System in North Korea," KIPS Trans. Comp. and Comm. Sys. Vol.8, No.5 pp.119~126, 2019.
[14] 양희승, 국방R&D정책 (피앤씨미디어 | 2020년 08월 20일)

저 자 소 개

변재현 (E-mail: uce03211@gmail.com)
2014 육군사관학교 지역연구 졸업(학사)
현 재 국방대학교 군사운영분석 석사과정
관심분야: 강화학습, 객체인식, 시뮬레이션

붙임#1 : 씬컨트롤러 C#스크립트

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SceneController : MonoBehaviour {
    public ImageSynthesis synth;
    public GameObject[] prefabs;
    public int minObjects = 10;
    public int maxObjects = 50;
    public int trainingImages;
    public int valImages;
    public bool grayscale = false;
    public bool save = false;

    private ShapePool pool;
    private int frameCount = 0;

    // Start is called before the first frame update
    void Start() {
        pool = ShapePool.Create(prefabs);
    }

    // Update is called once per frame
    void Update() {
        if (frameCount < trainingImages + valImages) {
            if (frameCount % 30 == 0) {
                GenerateRandom();
                Debug.Log($"FrameCount: {frameCount}");
            }
            frameCount++;
            if (save) {
                if (frameCount < trainingImages) {
                    string filename = $"image_{frameCount.ToString().PadLeft(5, '0')}";
                    synth.Save(filename, 512, 512, "captures/train", 2);
                }
                else if (frameCount < trainingImages + valImages) {
                    int valFrameCount = frameCount - trainingImages;
                    string filename = $"image_{valFrameCount.ToString().PadLeft(5, '0')}";
                    synth.Save(filename, 512, 512, "captures/val", 2);
                }

            }
        }
    }
```

```
    void GenerateRandom() {
        pool.ReclaimAll();
        int objectsThisTime = Random.Range(minObjects, maxObjects);
        for (int i = 0; i < objectsThisTime; i++) {
            // Pick out a prefab
            int prefabIndx = Random.Range(0, prefabs.Length);
            GameObject prefab = prefabs[prefabIndx];

            // Position
            float newX, newY, newZ;
            newX = Random.Range(-10.0f, 10.0f);
            newY = Random.Range(2.0f, 10.0f);
            newZ = Random.Range(-10.0f, 10.0f);
            Vector3 newPos = new Vector3(newX, newY, newZ);

            // Rotation
            var newRot = Random.rotation;

            var shape = pool.Get((ShapeLabel)prefabIndx);
            var newObj = shape.obj;
            newObj.transform.position = newPos;
            newObj.transform.rotation = newRot;

            // Scale
            float sx = Random.Range(0.5f, 4.0f);
            Vector3 newScale = new Vector3(sx, sx, sx);
            newObj.transform.localScale = newScale;

            // Color
            float newR, newG, newB;
            newR = Random.Range(0.0f, 1.0f);
            newG = Random.Range(0.0f, 1.0f);
            newB = Random.Range(0.0f, 1.0f);
            var newColor = new Color(newR, newG, newB);
            newObj.GetComponent<Renderer>().material.color = newColor;

        }
        synth.OnSceneChange(grayscale);
    }
}
```

붙임#2 : ShapePool C#스크립트

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public enum ShapeLabel { Cube, Sphere, Cylinder };

public class Shape {
    public ShapeLabel label;
    public GameObject obj;
}

public class ShapePool : ScriptableObject
{
    private GameObject[] prefabs;
    private Dictionary<ShapeLabel, List<Shape>> pools;
    private List<Shape> active;

    public static ShapePool Create(GameObject[] prefabs) {
        var p = ScriptableObject.CreateInstance<ShapePool>();
        p.prefabs = prefabs;
        p.pools = new Dictionary<ShapeLabel, List<Shape>>();
        for (int i = 0; i < prefabs.Length; i++) {
            p.pools[(ShapeLabel)i] = new List<Shape>();
        }
        p.active = new List<Shape>();
        return p;
    }

    public Shape Get(ShapeLabel label) {
        var pool = pools[label];
        int lastIndex = pool.Count - 1;
        Shape retShape;
        if (lastIndex <= 0) {
            var obj = Instantiate(prefabs[(int)label]);
            retShape = new Shape() { label = label, obj = obj };
        } else {
            retShape = pool[lastIndex];
            retShape.obj.SetActive(true);
            pool.RemoveAt(lastIndex);
        }
        active.Add(retShape);
        return retShape;
    }

    public void ReclaimAll() {
```

```
            foreach (var shape in active) {
                shape.obj.SetActive(false);
                pools[shape.label].Add(shape);
            }
            active.Clear();
        }
}
```