창직종합설계 프로젝트

홍익대학교 컴퓨터 공학과 4학년
B211218 정해윤

1. 프로젝트의 주제
2. 개발 환경
3. 프로젝트 진행 과정
4. 프로젝트 결과

1. 프로젝트의 주제

1.1 주제에 대한 설명

- 특정한 객체를 탐지하는 알고리즘의 학습에서 실제 데이터뿐만 아니라 Unity 등으로 만들어진 객체의 '가상데이터를 데이터셋의 일부로 활용하여 어느정도의 가능성 및 효율성을 가질 수 있는가' 에 대한 연구를 화재라는 객체를 탐지하는 모델을 만들면서 확인하였습니다.

1.2 주제를 선택한 이유

- 주제를 선택한 첫 번째 이유는 실제 데이터 학습을 시킴에 있어서 가장 큰 어려움 중 하나인 데이터셋의 확보에 대한 새로운 접근 방법의 모색입니다. 저 또한 객체 탐지를 하는 과정에서 데이터셋을 확보하는 것에 많은 어려움을 겪었으며, 그러한 방법을 해결하기 위하여 고민하던 중 가상의 데이터를 실제와 아주 비슷하게 만들어서 학습시켜 보는 방향을 생각해 보았고 실제로 가능성을 보기 위하여 테스트하였을 때 나쁘지 않은 결과를 보였기 때문에 위와 같은 주제를 선택하여 진행하게 되었습니다.

모델	100 data (real)	400data(real)	100 data (unity)
정답률	85%	87%	70%
오답률	39%	19%	6%
정확도	73%	84%	82%

<표1> 실제 데이터로 학습시킨 모델과 가상의 데이터로 학습시킨 모델 간 성능 차이

표1의 결과를 토대로 unity를 이용하여 학습시킨 데이터가 정답률이 조금 떨어지지만 오답률의 성능이 더욱 좋은 것에 착안하여 두 데이터를 적절하게 조합하여 만드는 데이터셋을 구성하기로 하였습니다.

실제 데이터를 더 추가하여 학습시킨 400data 결과에서 정답 오답률이 이전의 데이터에 비하여 많이 향상되었지만 100 data (unity) 의 학습결과와 정확도 측면에서 크게 차이가 나지 않는 것을 관찰할 수 있었습니다.

* 정확도 : 정확하게 예측한 데이터 / 전체 데이터
* 정답률 : 화재로 정확하게 인식한 데이터 / 전체 화재 데이터
* 오답률 : 화재가 아닌 것을 화재로 인식한 데이터 / 화재가 아닌 데이터

* 인식률 : 감지한 객체가 특정한 객체일 확률 (0.5 -> 50%)
* 전체 샘플 데이터는 화재 사진 100장 화재가 아닌 사진 100장 총 200장으로 구성 되어 있으며 모델의 학습에 사용되지 않은 데이터를 샘플 모델으로 사용하였습니다.
* 기본적으로 표시 되어있지 않은 모델들은 모두 인식률을 0.5 로 측정하였습니다.

-주제 선택의 두 번째 이유로는 실제 데이터수집에 대한 비용적인 측면에서 생각하였습니다.
실제 데이터 수집의 경우 특정 객체에 대한 데이터 수집 시
https://www.shutterstock.com 사이트의 경우 750여장의 사진을 200$에 구매 할 수 있습니다.

경제적 측면에서 생각하였을 때 기계학습을 통하여 unity 등의 가상이미지를 만드는 프로그램을 사용하여 만들어진 데이터셋이 어느정도의 정확도를 보장 할 수 있다면 더욱 효과적 이라고 생각 하였기 때문에 위의 주제를 선택하게 되었습니다.

1. Description of the project

This project is a study of "how much potential and efficiency can be achieved by using virtual data as part of a dataset"

the first reason for studying this topic is finding new ways to collect data, and the second reason is cost savings.

for this reason, I aimed to create a good data set with virtual data using unity.

2. 개발 환경

- Ubuntu 16.04
- GTX 1060
- Python , C , C++ , C# , Unity
- OpenCV 3.3
- Cuda 8.0
- Yolo v2

2.1 Yolo v2

-객체 탐지를 위하여 Yolo v2를 사용하였습니다.
준비시간 및 실행시간에 있어 v1 버전보다 좋은 성능을 나타내며 동시에 객체의 인식률에

성능 차이가 있었습니다. v3 버전에 비하여 시간은 v2가 우수하지만 객체 인식률의 경우 v3모델이 조금 더 개선되었습니다. 하지만 v3 모델은 개발된지 얼마 지나지 않았기 때문에 자료들이 부족하며 신뢰성이 떨어 질 수 있다고 판단되어 v2 버전을 사용하였습니다.

	Yolo v1	Yolo v2	Yolo v3
준비시간	4.963	3.719	4.645
실행시간(detection)	0.189	0.020	0.046

<표2> Yolo v1, v2 , v3를 각각 비교

2. Development environment

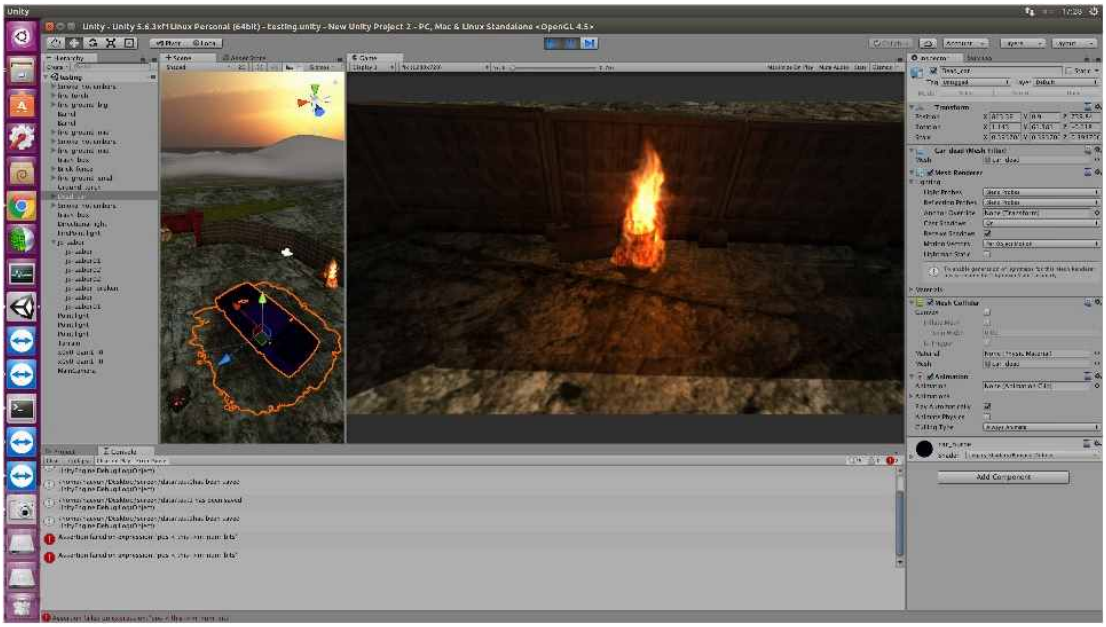
the development environment is as above.

I used Yolo v2 for object detection.
It showed better performance than the Yolo v1 version in preparation time and execution time, and at the same time there was a performance difference in the recognition rate of objects. Yolo v3 model lacked a lot of data than v2 so I decided to use Yolo v2.

3. 프로젝트 진행 과정

처음 시작한 작업은 unity 프로그램을 이용한 가상 데이터셋을 만드는 작업입니다. unity가 가상데이터를 만들기에 적합한 프로그램라고 생각하였기에 unity 프로그램을 사용하였으며, unity 프로그램에서 main camera를 이용하여 가상으로 만든 불 객체를 인식하여 1프레임당 1장의 사진 (저의 컴퓨터 기준 약 0.3초당 1장)을 추출하여 자동으로 저장하였습니다.

동시에 unity 내에서 설정한 불 객체의 위치를 파악하여 수집과 동시에 학습에 필요한 정보를 추출하여 수작업으로 하여야 하는 데이터셋 학습 전의 marking 과정을 최소한으로 줄였습니다.

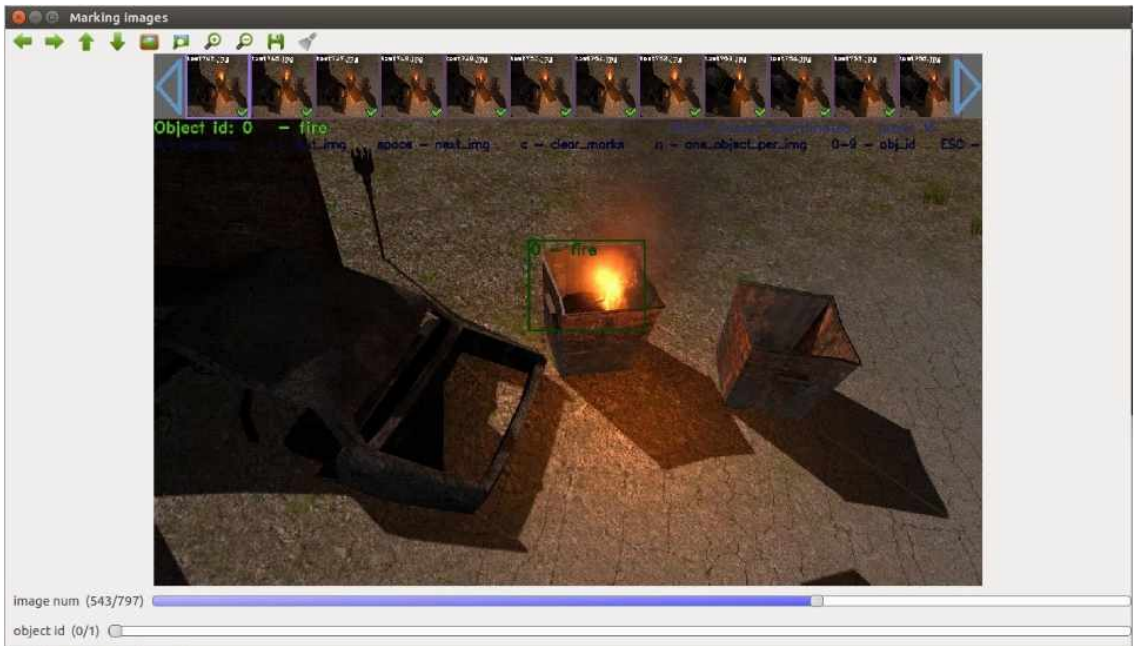


<그림 1> unity 프로그램을 이용하여 가상 데이터셋을 만드는 과정

maincamera가 화재 객체를 인식하고 인식 된 객체의 위치정보가 화면상의 위치정보로 전환된 후 yolo 에 맞는 형태로 변환되어 txt 파일으로 지정 경로에 저장되도록 만들었습니다. 데이터에 다양한 변화를 주기 위하여 주변 환경이나 객체의 색상, 위치, 카메라의 구도등을 임의로 바꾸면서 데이터를 모았습니다. (오버피팅 방지)

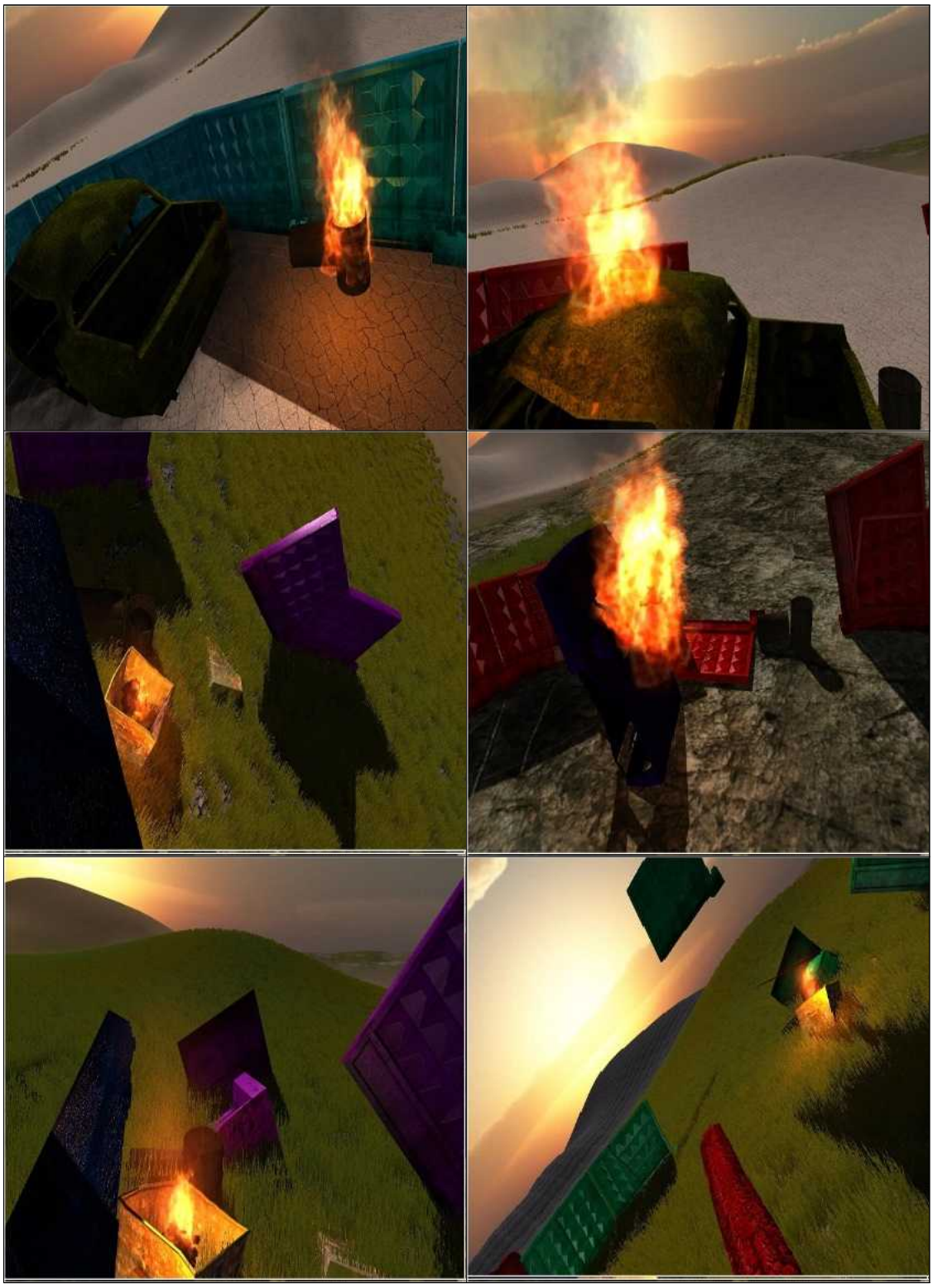
<그림2> 만들어진 가상 데이터들과 저장된 위치 정보

실제 데이터파일인 test#.jpg 와 위 사진에서 화재 객체가 있는 위치 좌표를 yolo에 맞게 바꾸어서 변경한 test#.txt 파일으로 각각 저장하였습니다.



<그림3> yolo에 맞게 marking 되어있는 가상 데이터

실제 만들어진 가상 데이터를 학습시키기 전 과정으로 보면 화재 객체에 대한 위치값이 이미 저장되어 있는 것을 확인할 수 있습니다.



<그림4> 실제 만들어진 가상 데이터들의 예시

최대한 다양성을 주기 위해서 여러 가지 구도, 색채, 객체의 크기 등 여러 가지 요인들을 바꾸어가면서 데이터를 만들었습니다.

위의 실험으로 가상데이터를 어느정도 안정적이고 확정적으로 만들게 된 후
가상 데이터를 사용하는 것에 대한 효용성을 검증함과 동시에 가상 데이터만으로 이루어진 데이터셋의 실용 가능성을 보기 위하여 표 1의 실험을 진행 하였습니다.

모델	100 data (unity)	1000 data (unity)	1500 data (unity)
정답률	70%	58%	23%
오답률	6%	31%	12%
정확도	82%	63.5%	55.5%

<표3> Unity로 만든 가상데이터로만 학습시킨 모델들의 결과

위의 표의 결과를 토대로 가상 데이터로만 학습시킨 모델은 데이터셋의 크기가 커질수록 정확도가 감소한다고 생각하였고 가상데이터만으로 학습시키는 것에 무리가 있다고 판단하였습니다.

위의 실험 후 만들어진 약 1500여장의 가상데이터를 화재 객체의 크기와 전체 데이터의 밝기에 따라 4개의 분류군으로 나눈 후 각각의 데이터 분류군에 대한 결과를 알아보았습니다.
이 실험은 갑작스럽게 데이터들의 정답률 오답률과 정확도가 낮아진 이유를 알아보기 위한 실험이었습니다.

모델	class0(S/W)	class1(S/B)	class1(B/W)	class1(B/B)
정답률	68%	62%	33%	35%
오답률	24%	31%	12%	4%
정확도	72%	65.5%	60.5%	65.5%

<표4> 만들어진 가상 데이터셋을 정해놓은 분류에 따라 분류 후 각각의 데이터를 학습시킨 모델들의 결과

*각각의 데이터의 분류는 화재 객체의 크기와 전체 데이터의 밝기에 따라 결정

객체의 크기 - S / B , 데이터의 밝기 - B / W

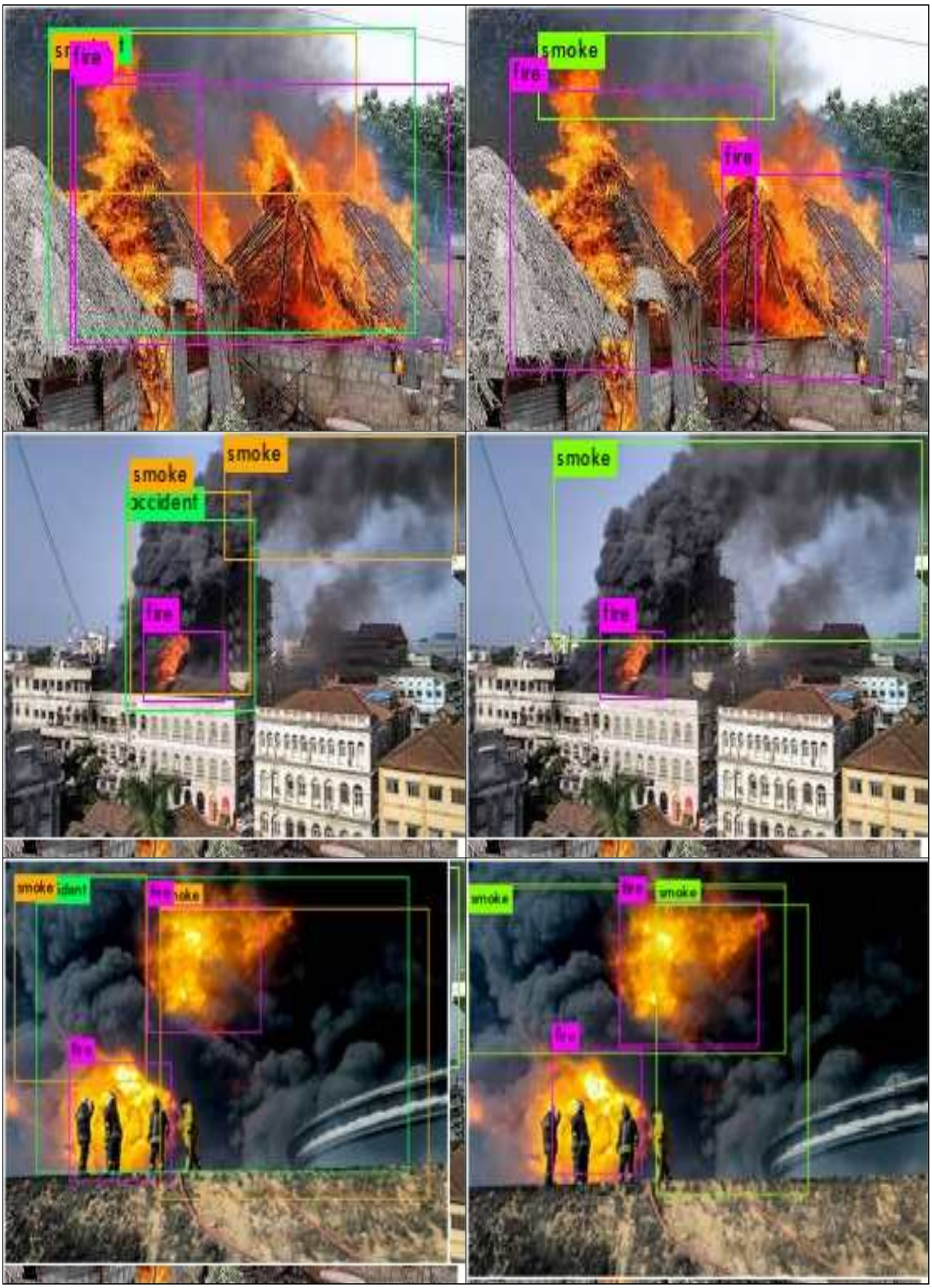
위의 실험을 토대로 정답률이 높은 군집들을 묶어서 학습시켯으나 모델의 결과가 좋지 못하였습니다. 저는 위의 데이터들을 분류하는것에 있어 컴퓨터가 학습하는 기준을 임의로 찾아내서 분류하기가 어렵다고 판단하여 역으로 현재 만들어진 모델 중 가장 성능이 뛰어낫던 1500개의 실제 데이터사진을 학습시킨 모델에 약 1500여장의 가상 데이터를 모두 탐지시켜보았고 그 중 데이터의 인식률이 낮게 나오는 (50% 미만의 인식률) 약 500여장의 정제된 데이터를 구할 수 있었습니다.

다음 실험으로는 기존에 탐지하던 fire 및 smoke 객체에 이은 fire 객체와 smoke 객체를 합친 accident 객체를 추가했습니다. accident 객체의 도입 이유는 두가지 입니다.

첫 번째로는 기존 fire 객체만으로 인식하지 못하던 화재 상황의 인식률을 높이기 위해서 입니다. 실제로 약 1500장의 실제 화재 사진을 토대로 인식하였을 때 이전의 fire 와 smoke 두 가지 객체를 사용하던 때보다 accident 객체를 추가 할때 인식하는 비율이 높게 나왔습니다. 이는 화재 상황을 fire 객체로 인식하지 못하더라도 accident 로 인식하는 경우가 있고 그것 또한 바르게 인식 한 것으로 포함하였기 때문입니다.

두 번째로는 실제 사람이 생각하는 화재 및 연기 탐색과 더불어 컴퓨터에게 자체적인 화재상황에 대한 인식 및 학습을 시킨 후 컴퓨터가 인식하는 화재의 상황을 관찰하기 위하여입니다.

실험 당시 accident 객체의 조건은 fire 객체와 smoke 객체가 함께 연관되어서 화면에 분명하게 나타나는 것을 조건으로 하였습니다.
실제 실험에서 사람이 인식하기에는 fire 객체 밖에 없는 상황에서 accident 객체로 인식하는 상황 등의 자율적으로 학습하여 성능이 개선 된 상황이 있었습니다.



<그림5> accident 객체 추가 후 인식한 사진(왼쪽)과 기존의 사진(오른쪽)의 비교

이후의 실험에서는 실제 화재 사진과 만들어 놓은 가상 데이터를 단순히 통합만 시킨 후 기존의 학습시킨 모델과의 차이를 확인하고 이후 실험으로 미리 만들어진 가상 데이터중 정제된 가상 데이터들을 추가로 데이터에 넣어 학습시키는 실험을 진행하였습니다.

모델	1500 data (real)	3000 data (complex)	2000 data (complex)
정답률	98%	95%	99%
오답률	16%	12%	11%
정확도	91%	91.5%	94%

<표5> 실제 데이터로 이루어진 모델과 기존 데이터셋에 정제되지않은 가상 데이터셋을 추가해 학습시킨 모델의 결과

위의 실험에서 약 1500여장의 가상 데이터를 추가 했음에도 불구하고 정답률은 소폭 감소하고 오답률은 소폭 증가하여 결과적으로 정확도는 약 0.5% 증가 했습니다. 하지만 저는 약 2배에 달하는 데이터를 추가하였음에도 불구하고 너무 미비한 발전이였다고 생각하였고 이전에 분류 했었던 정제된 가상데이터 500여장을 추가하여 학습을 다시 한번 시켰습니다.

결과적으로 정제된 데이터를 추가 한 결과 정답률의 경우 아주 작은 증가가 있었으나 이미 이전 데이터에서 매우 높은 정답률을 기록하였기에 큰 차이는 없었습니다. 오답률의 경우 16%에서 11%로 5%가량 하락하는 성능 개선의 효과를 볼 수 있었습니다.

다음으로는 실제 데이터에 가상데이터를 넣었을때에 성능의 차이가 얼마정도가 나는지에 대한 실험이었습니다. 이 실험에서는 실제 데이터 1500장을 기준으로 하여 1500장의 실제 데이터에 가상 데이터 500장을 넣은 혼합 데이터셋 2000장을 학습시킨 모델과 실제 사진 1500장에 새로운 실제 사진 500장을 학습시켜 만든 실제사진으로만 구성된 데이터셋으로 학습시킨 모델간의 비교를 하였습니다.

모델	1500 data (real)	2000 data (real)	2000 data (complex)
정답률	98%	97%	99%
오답률	16%	14%	11%
정확도	91%	91.5%	94%

<표6> 기준모델과 각각 실제데이터 및 가상 데이터를 추가로 넣어 학습시킨 모델들의 결과 (단 실제 데이터 500장은 완전 새로운 데이터를 추가하였으며 가상데이터의 경우 이전 실험에서의 결과(표5)를 토대로 비교하였습니다)

위의 표의 결과를 토대로 실제 사진 데이터의 경우 학습 데이터 500장이 증가하였을 때에 정답률의 미미한 감소와 오답률이 약간 개선되는 결과를 보여주었습니다.

하지만 같은 상황에서 실제 데이터에 가상데이터를 혼합하여 학습시킨 모델은 정답률의 조금의 증가와 오답률의 감소 및 정확도 개선의 확실하게 개선된 모델의 모습을 보여 주었습니다.

위 실험결과를 토대로 실제 데이터를 추가하는것과 가상 데이터를 추가하는 것에 큰 차이점이 없으며 오히려 잘 정제된 가상 데이터를 혼합시킬 경우 성능의 개선을 도모할 수 있다고 생각하였습니다.

이후의 실험에서 두자리대였던 오답률을 감소 시키기 위하여 추가적인 실험을 하였습니다. 실험은 이전에 학습시킨 후 가장 높은 정확도를 보여주는 2000장의 실제 데이터와 가상데이터로 만들어진 모델으로 하였습니다.

이 실험에서 저는 3개의 객체에 추가적으로 other 라는 새로운 객체를 추가 하였습니다. 이 객체는 화재 상황으로 감지하지 않는 모든 객체를 총칭하는 객체 이지만 other 객체의 정확도나 인식률을 크게 높일 필요는 없고 다만 현재까지의 모델이 자주 오인하는 연기 등의 객체에 대한 오인률을 조금 낮추기 위하여 추가 하기로 하였습니다.

other 객체의 경우 이전 실제 데이터 중 약 30% 가량에서 추가로 객체를 표시 하였고 2000장의 데이터에 추가로 화재 객체가 없는 순수한 other 의 데이터를 약 200장 추가 하여 총 2200장의 데이터셋을 학습시켰습니다.

모델	1500 data (real)	2000 data (complex -3obj)	2200 data (complex - 4obj)
정답률	98%	99%	97%
오답률	16%	11%	3%
정확도	91%	94%	97%

<표7> 기존 데이터 와의 비교 및 객체 추가에 따른 결과 비교

위의 실험 결과 처음 1500여장의 실제 데이터를 학습시킨 모델에 추가적으로 정제된 500여장의 가상 데이터를 더하여 학습시킨 모델이 정확도가 더욱 높아진 것을 볼 수 있었으며 이렇게 개선된 데이터에 새로운 객체인 other 라는 객체를 추가 함으로서 정답률은 조금의 손실이 생겼지만 오답률의 경우 눈에띄는 개선을 보였으며 더불어 정확도 또한 매우 높아졌음을 확인 할 수 있었습니다.

위의 결과들을 토대로 저는 실제 데이터셋 뿐만 아니라 가상 데이터 셋 또한 얼마든지 좋은 데이터셋으로 사용 가능하며 경우에 따라서는 더욱 효과적인 데이터셋으로서 사용이 가능하다고 결론지었습니다.

3. Progress of project

Through various experiments, I tried to find a way to increase the accuracy of the model as much as possible and to lower the error rate.
In the first experiment I compared the results of the model with increasing virtual data data. I have hypothesized that it would be better to use a mixture of data rather than a model using only virtual data sets.

In the second experiment, it was an experiment to find out how to create a cluster of virtual data to make better data. The second experiment was based on the size of the fire object and the brightness of the screen

Because the result of the second experiment was not good, this time, the virtual data was recognized in the existing well-made model(1000) and the data with good recognition rate and the data which was not (500)

In subsequent experiments, I was experimenting with detecting fire / smoke two objects and adding new accident objects to detect fire.

In addition, I have experimented with comparing unity data to learn and adding real data to the model to learn, and adding another object, another object, to improve performance.

As a result, I have created a learning model that adds virtual refined data from unity to existing fire data and classifies it into four object classes (fire, smoke, accident, other).

4. 프로젝트 결과

이전의 실험에서 얻은 결론들을 토대로 마지막으로 새로운 모델을 만들기로 하였습니다. 새로운 모델은 이전의 실험에서 사용되었던 추가 화재 데이터 500장을 추가 하였으며 , other 데이터의 양을 늘려서 어느정도의 효율을 더욱 높일 수 있지 않을까 라는 실험을 위하여 other 데이터를 2000장을 데이터에 추가하였습니다.

추가적으로 마지막 모델의 실험에 앞서 새로 만들어지는 최종 모델과 이전에 가장 좋은 효율을 보였었던 모델 (2200-complex) 의 세부적인 차이점을 비교 하기 위하여 모델의 인식률에 따른 성능을 따로 확인했습니다.

모델	2200 data(0.5)	2200 data(0.6)	2200 data(0.3)	2200 data(0.35)
정답률	97%	92%	99%	99%
오답률	3%	3%	9%	9%
정확도	97%	94.5%	95%	95%

<표8> 기존 최고성늘 모델의 인식률에 따른 성능 차이

위의 데이터를 토대로 인식률의 변화에 따라 최적의 성능이 나오는 구간이 따로 있다고 생각되었습니다. 위의 모델같은 경우에 0.5 의 인식률을 사용한 기존의 모델이 성능이 가장 높았습니다. 인식률을 매우 낮게 잡은 경우에 정답률은 매우 높아지지만 오답률 또한 같이 높아지는 부작용이 있음을 확인 하였습니다. 이 모델의 경우 대부분의 화재 사진에서 화재상황을 평균적으로 약 0.6 이상의 인식률을 보이며 인식 하는 것으로 나타납니다. 인식률을 0.5 이상으로 높인 실험에서 공통으로 틀리게 인식하는 3%의 예시에서의 사진들은 모두 65%이상의 인식률을 보였습니다.

새로 만들어진 최종 데이터의 갯수는 약 4900여장 이며 이중 unity로 만든 가상 데이터는 약 500여장, 화재 데이터와 other 데이터의 비율은 3:4입니다. (가상 화재 데이터 포함)

모델	2200 data (complex 4obj)	final(0.6)	final(0.5)	final(0.35)	final(0.3)
정답률	97%	78%	92%	96%	98%
오답률	3%	0%	0%	0%	0%
정확도	97%	89%	96%	98%	99%

<표9> 기존 최고 모델과 최종 데이터의 비교 및 인식률에따른 결과

최종모델의 경우 처음 학습 시킨 뒤 기존의 데이터들과 함께 0.5의 인식률을 기준으로 실험을 하였습니다. 그 결과 정답률 92% , 오답률 0% , 정확도 96% 의 결과가 나왔습니다. 이 후 기준이 되는 인식률을 조금씩 바꾸어 가며 정답률과 오답률 , 정확도의 변화를 조사 하였습니다. 인식률의 경우 너무 높은 기준을 삼는 경우 화재를 화재로 인식하지 못하는 문제점이 있었습니다. 적정한 수준의 인식률을 알아보기 위하여 여러 방면으로 인식률을 조사 하였습니다. 최종모델의 경우에서 불이 아닌 예시들에 대하여 불으로 인식을 하는 경우는 인식률이 약 15%(0.15) 이하인 경우에만 발생하였습니다.

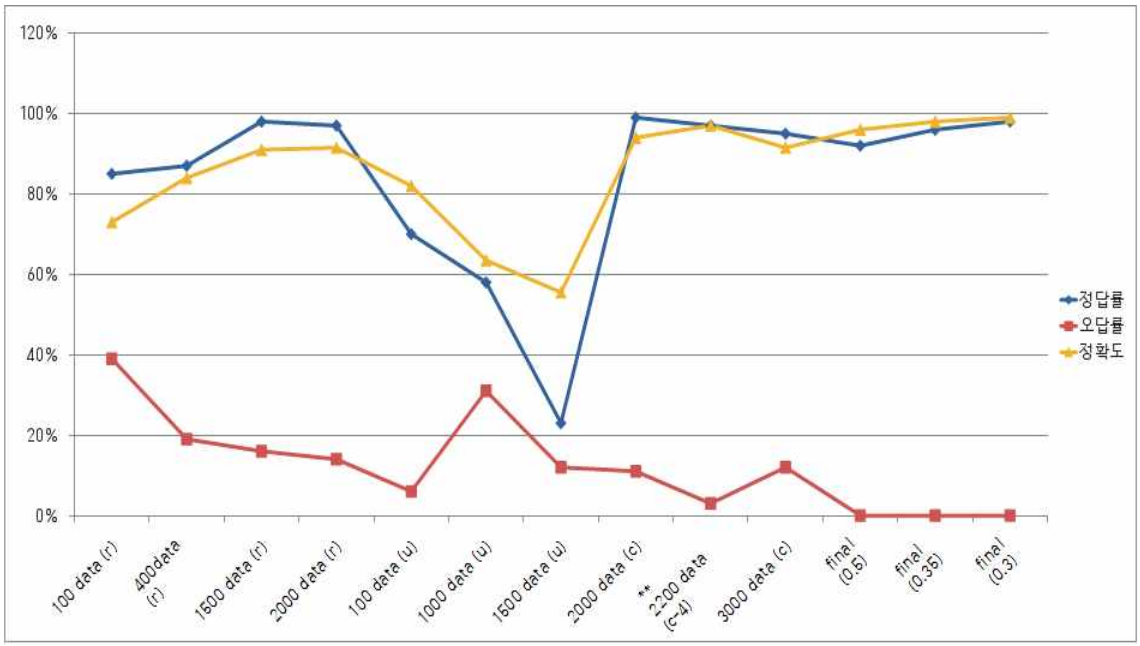
최종모델의 경우 0.5의 인식률에서는 기존의 모델과 비슷한 정확도를 나타내며 정답률이 조금 낮아졌지만 오답률이 매우 낮아지는 결과를 보았습니다. 이 후 인식률을 낮추면서 실험을 한 결과에도 오답률은 계속해서 0%를 유지하며 정답률이 조금씩 높아 졌습니다. other 객체의 수가 많아지면서 화재 상황과 그렇지 않은 상황을 인식하는 성능이 조금 더 개선된 것을 확인 할 수 있었습니다.

이전 최고 성능 모델과 비교를 하였을 때 other 객체를 더 많이 추가 함으로서 other 객체의 인식률이 매우 높아졌고 그로 인해서 오답률이 현저히 줄어 들었으며 또한 낮은 인식률에서 더욱 효과적인 성능을 나타 낼 수 있기 때문에 정답률 또한 개선 되는 것 같다고 생각됩니다.

마지막으로 실험했던 결과들의 도표 및 그래프 자료를 첨부합니다.

모델	100 data (r)	400 data (r)	1500 data (r)	2000 data (r)	100 data (u)	1000 data (u)	1500 data (u)	2000 data (c)	** 2200 data (c-4) (0.5)	3000 data (c)	final (0.5)	final (0.35)	final (0.3)
정답률	85%	87%	98%	97%	70%	58%	23%	99%	97%	95%	92%	96%	98%
오답률	39%	19%	16%	14%	6%	31%	12%	11%	3%	12%	0%	0%	0%
정확도	73%	84%	91%	91.50 %	82%	63.50 %	55.50 %	94%	97%	91.50 %	96%	98%	99%

<표9> 전체 모델들의 결과값



<그래프 1> 전체 데이터에 대한 결과값의 그래프 표시

* 데이터 뒤에 (r) 의 경우 실제 데이터로 학습시킨 모델 , (u)의 경우 가상데이터를 학습 시킨 모델 , (c) 의 경우 복합데이터를 학습 시킨 모델입니다.

* c-4의 경우 4개의 객체를 학습 시킨 모델입니다. 2200data와 final 데이터들이 4개의 객체를 학습시킨 모델입니다.

* 성공적인 결과값의 기준으로 잡았던 정답률 95% 이상, 오답률 5% 미만, 정확도 95% 이상의 모델은 밑줄 표시 하였습니다.

위의 실험들의 결과를 토대로 저는 실제 사진 데이터를 이용한 학습 모델과 가상 데이터셋을 이용하여 학습시킨 모델간의 차이 , 가상 데이터를 이용하여 학습 시켰을 때의 효율 , 가상 데이터 셋이 가지는 이점과 가상 데이터와 실제 데이터를 학습시킨 모델의 추가적인 성능 향상방법 등의 연구를 하였습니다.

이 프로젝트를 통하여 실제 데이터에 추가적으로 가상 데이터를 넣어서 학습 시키는 것이 경제적이고 효율적인 결과를 가져올 수 있으며 더 좋은 학습 모델을 만들 수 있다고 결론 지었습니다.

결과적으로 위의 다양한 실험들에서 나온 결과들을 통합하여 만족할만한 성능을 보여주는 새로운 화재 인식 모델을 만들어 보았습니다.

끝까지 읽어 주셔서 감사합니다.

4. Project results

Based on previous experiments, I decided to create a new fire recognition model. the new fire model has added an additional 500 actual fire data and 2000 other object data. In order to find the most appropriate recognition rate for the new model, fire recognition was performed based on various recognition rates.

In this project I have found many ways to improve the performance of the model while creating various models. I also found that by comparing with other models, I could improve the likelihood of using virtual datasets with unity and the performance of the model so created.

In conclusion, I think that I have succeeded in developing a model with satisfactory performance through various results mentioned above.