**인공지능의 이해,** 서주영**,** 2022 1st semester

**Google Teachable Machine을 이용한 AI 웹서비스 기획, 구현, 배포**

201823869

E-Business

조성우



**INDEX**

**1. 프로젝트 개요**

**2. 서비스 핵심 기능 및 사용 설명**

**3. 구현 및 배포 과정**

3.1 Training a ML model

3.2 Web page

3.3 배포 : Netlify

서비스 url : <https://finger-gunner-horizd.netlify.app/>

1. **프로젝트 개요**

해당 인공지능의 이해 강의 중 실습했던 Google의 오픈소스 머신러닝 도구인 **Teachable Machine**로 새로운 아이디어의 모델을 만들고 **Html + javascript + css** 기반의 웹페이지에 탑재하여 AI를 활용한 새로운 웹 서비스를 기획, 구현, 배포하는 일련의 과정을 진행했습니다.

처음엔 아주대 건물 인식 및 소개, 정수리 이미지 인식을 통한 탈모 판별 및 대응 소개 등의 실용적인 웹사이트를 구상하였지만 이미지 데이터 확보의 어려움이 예상되었고, 당장 혼자서 생성할 수 있는 데이터로부터 서비스를 구상하다가 웹 캠의 손동작 이미지 인식을 통한 총 발사 시뮬레이터 - Finger Gunner 라는 AI 웹 서비스 프로젝트를 생각하여 시작하게 되었습니다.

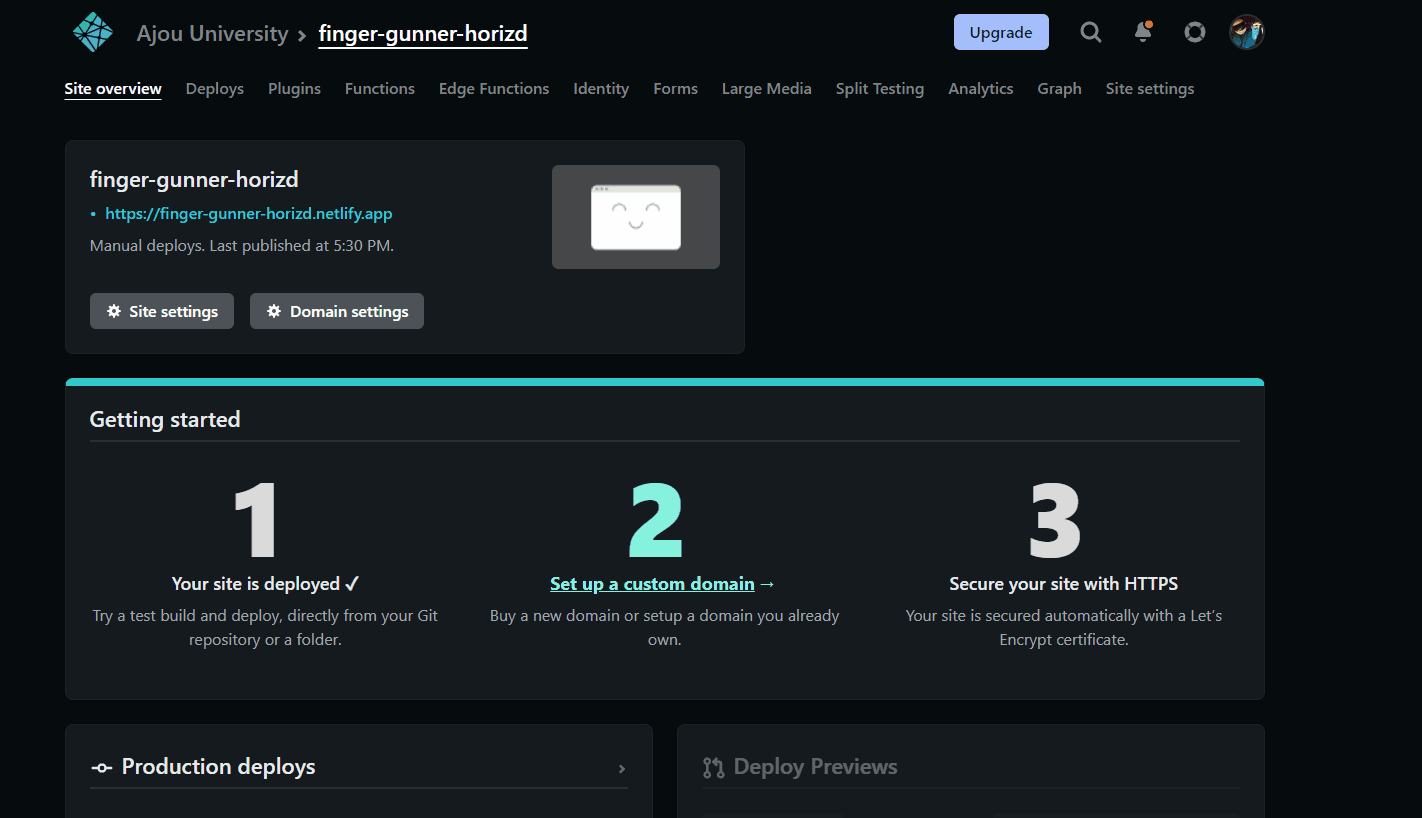
여러가지 게임의 요소를 구현하여 실질적 재미를 줄 수 있는 서비스로 발전시키고자 하는 욕심이 있으나, 과제의 시간제약을 고려하여 우선 서비스의 핵심 요구사항인 (1) 사용자 손동작 인식 (2) 상호작용 구현 두가지의 완성도에 집중한 프로토타입을 기획, 구현, 배포했습니다.

해당 리포트 제출 이후, 실제 이용자를 확보하는 것을 목표로 추가 기능 개발, 보완을 통해 완성도 높은 웹 서비스로 발전시키고자 합니다.

1. **서비스 핵심 기능 및 사용 설명**

Finger Gunner는 현재 프로토타입 수준으로 최소 핵심 기능만 구현돼 있습니다.

서비스 이용을 위해 캠이 준비된 디바이스가 필요하며, 개발자의 학습데이터가 노트북 캠을 통해 제공되었으므로 현재로서는 노트북을 사용한 플레이가 가장 이상적인 사용 환경입니다. 또한 아래의 동작 예시처럼 상반신만 노출되도록 캠을 조절하여 플레이합니다.



[Finger Gunner 사용 영상.gif]

**[ Flow ]**

**반복 :** 웹 캠을 통한 사용자 포즈 입력 -> ML 포즈 클래스 분류 ->하단의 Web 상호작용 동작 실행

사람, 실내, 창문, 젊은이(가) 표시된 사진

자동 생성된 설명사람, 실내이(가) 표시된 사진

자동 생성된 설명사람이(가) 표시된 사진

자동 생성된 설명

[대기, 발사, 재장전 동작 예시: 반전된 이미지로 주의 요망]

* 대기: 오른손으로 총모양을 만들어 정면으로 겨누는 동작
* Web 동작:

웹 캠 하단에 ‘대기중’ 텍스트 렌더링

* 발사: 오른손 총을 발사하는 반동으로 팔을 드는 연출의 동작

(정확성을 높이기 위해서 손목만 움직이는 것이 아니라, 팔이 확실히 들어져야 합니다. )

* Web 동작:

웹 캠 하단에 ‘땅!’ 텍스트 렌더링

총 발사 효과음 실행

웹 캠 하단의 배경에 총알 적중 표시 렌더링 (+색상변화)

빨간색이(가) 표시된 사진

자동 생성된 설명

* 재장전: 왼손을 오른손 총의 아래에 얹어 양손이 몸의 중앙에 모이도록 한다.
* Web 동작:

웹 캠 하단에 ‘장전중..’ 텍스트 렌더링

재장전 효과음 실행

1. **구현 및 배포**
   1. **Training a ML model**

모델은 대기, 발사, 장전 세가지 모션으로 wait, shoot, reload 세가지 클래스를 기획하여 학습시켰습니다. 오른손으로만 학습시켰기 때문에 향후 왼손잡이를 위한 추가 학습이 요구됩니다.

**[Pose:** Wait, Shoot, reload 순**]**

사람, 실내, 창문, 젊은이(가) 표시된 사진

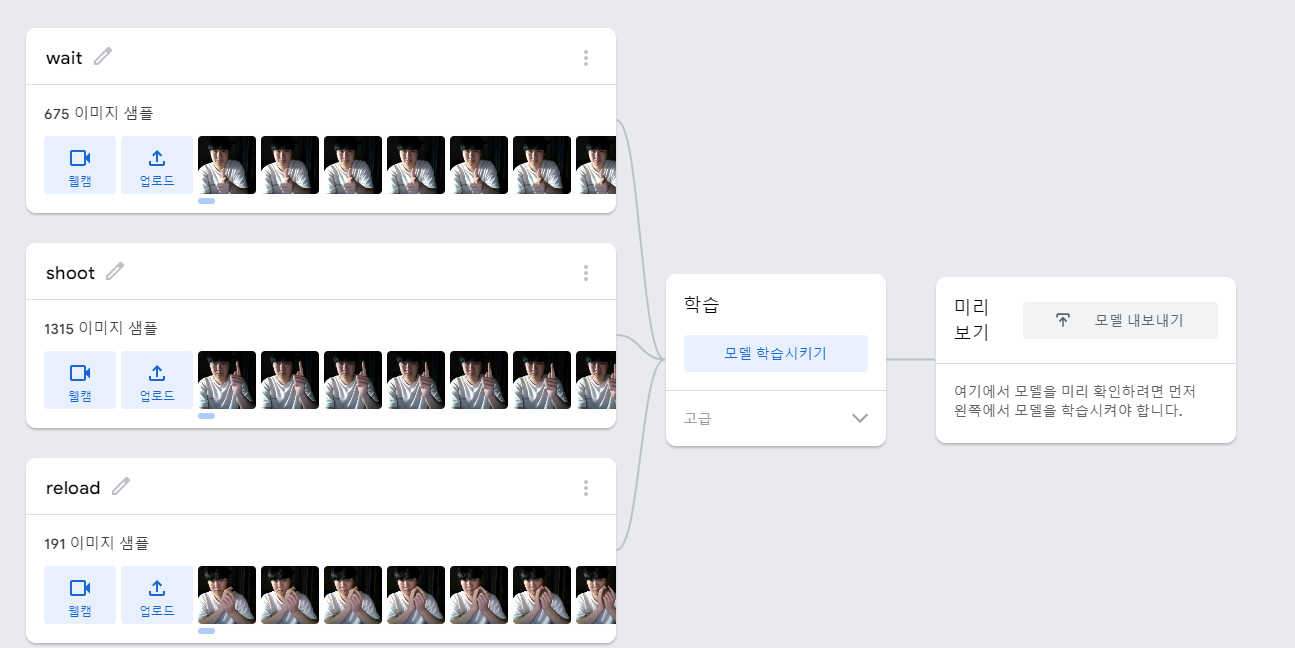
자동 생성된 설명사람, 실내이(가) 표시된 사진

자동 생성된 설명

* **Image 기반 모델의 문제점 (Pose training 도입)**

첫 기획 및 학습 시 이미지 기반 학습모델을 사용하였습니다. 처음에도 조명 수준, 입은 옷, 웹 캠의 거리 변화 등에 따라 예측 정확성이 크게 떨어지는 것을 확인하였으나, 데이터 학습을 위한 이미지 생성 시 옷을 다양하게 바꿔 입고 조명을 바꾸는 시도를 통해 총 3500개에 이르는 적지 않은 학습데이터를 만들었고, 해당 방법으로 예측 적중률을 높여 큰 문제의식 없이 막바지까지 진행하였습니다.

**텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

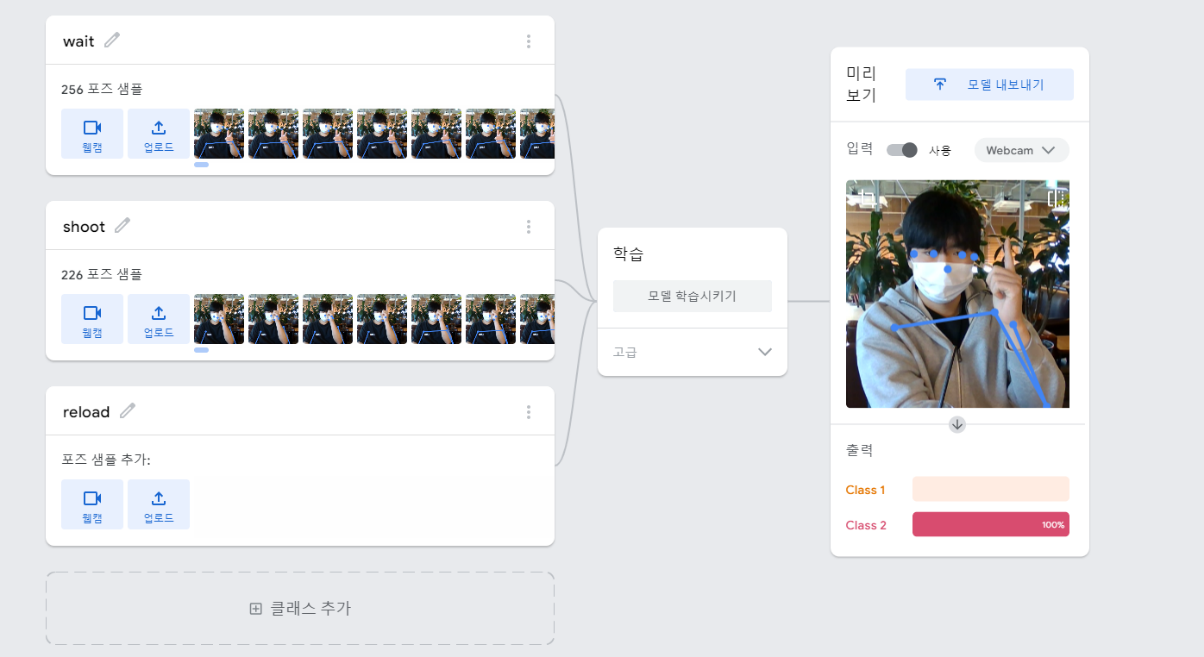
[이미지 데이터 학습을 이용한 모델]

하지만 집이 아닌 다른 배경에서 테스트를 진행해보고 몇몇 지인에게 테스트를 부탁한 결과 적중률이 크게 떨어지는 것을 확인하여 포즈 데이터를 통한 학습으로 급하게 모델을 변경했고 사용자의 실행 환경에 따른 적중률 하락을 크게 감소시킬 수 있었습니다.

하지만 포즈 기반 학습은 일반 이미지 학습에 비해 시간 및 메모리 성능이 현저히 떨어진다는 사실을 확인했습니다(+pc 깜박임 등 문제 발생), 이에 따라 데이터가 일정 수준 이상 커지면 개발자 개인 PC에 큰 무리를 줄 수 있다는 한계점을 발견했습니다. 또한 손가락의 위치를 인식하는 것이 아닌, 팔과 손목의 노드를 기반으로 이미지를 인식하기 때문에 팔의 움직임을 정확히 수행하지 않을 경우 이미지 인식보다 적중률이 떨어질 수 있는 문제점이 여전히 존재합니다.

하지만 여전히 포즈 기반 학습에 정확성의 강점이 눈에 띄게 존재하기 때문에 구현에 앞서서 제 서비스의 도메인을 진지하게 고려하고 모델학습 방법을 결정했다면 이와 같은 어려움을 겪지 않았을 것이라 반성합니다.

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

[포즈 데이터 학습을 이용한 모델]

* 1. **Web Page**

**3.2.1 ‘id : button\_start’ 요소 클릭 시, ML 코드가 담긴 init() 수행**

* Teachable Machine의 ML 알고리즘 수행

**3.2.2 init() 내부에 header 요소 제거 함수 삽입**

처음엔 init() 알고리즘 내부에 removeChild() 함수를 사용한 요소 제거 방식으로 구현하였으나, 갑자기 사라지는 부자연스러움을 극복하기 위해 방법을 찾아보았고 jQuery의 fadeOut() 함수를 사용하여 타이틀과 버튼이 서서히 사라지는 방식으로 수정했습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[remove child를 수행하는 delete\_startTag()함수]

텍스트이(가) 표시된 사진

자동 생성된 설명

[jQuery의 fadeout()을 사용한 Elements가 서서히 사라지는 효과 구현]

**3.2.3 포즈의 possibility 판별 조건 조정**

텍스트이(가) 표시된 사진

자동 생성된 설명

**3.2.4 ‘bullet’ 오브젝트 구현**

Teachable ML 코드가 사용자의 포즈를 shoot, reload 클래스로 예측할 시, 관련된 interactive 기능 수행을 위해 bullet object를 구현하고 내부에 shoot()메소드와 reload()메소드를 구현하였습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **shoot() 메소드**

* **총알 자국 그리기**

Javascript의 canvas 라이브러리를 이용하여 총알자국을 그렸습니다. 보다 사실적인 총알 자국 표현을 위하여 발사 후 짧은 시간의 흐름 동안 색상을 (white=>red=>black) 순으로 연속적으로 변경한 후 사라지도록 구현하였습니다.

**(+) 한계점 :**

기획 시엔 개별 발사 시도마다 특정 영역범위 내의 무작위 좌표값을 할당하여 총알 흔적을 여러 위치에 남기고자 하였으나, 해당 ML모델의 알고리즘의 ms 시간 단위 예측으로 인해 무분별한 탄환흔을 남기는 문제를 야기할 것으로 예상되어 사용자 경험에 좋지 않을 것이라 판단하여 보류하기로 하였습니다.

* **해결 시도:** 해당 ML알고리즘에서 bullet이 시간지연 함수 등을 통해 특정 주기마다 판별하도록 하여 shoot이 무차별적인 연속 실행되지 않도록 되도록 수정하여 문제를 해결해보고자 하였으나,

async function loop() {

  webcam.update(); // update the webcam frame

  await predict();

  window.requestAnimationFrame(loop);

}

위와 같이 구성된 알고리즘에서 predict() 함수의 동작을 시간지연 함수 등으로 지연시킬 경우 await을 통한 동기처리 방식이 webcam의 동작 또한 지연시켜 전체적으로 끊김 현상을 초래하였고, 결과적으로 사용자 경험을 크게 저해하였기 때문에 bullet의 무작위 좌표 할당 및 구현을 보류하였습니다.

* **발사 효과음 실행**

발사 효과음 구현을 위해 음원 오픈소스 사이트로부터 찾아 삽입하였습니다.

1. **reload() 메소드**

* **재장전 효과음 실행**

발사 효과음 구현을 위해 음원 오픈소스 사이트로부터 찾아 삽입하였습니다.

* 1. **배포: Netlify**

무료 웹 애플리케이션 및 정적 웹사이트의 호스팅 및 서버리스 백엔드 서비스를 제공하는 Netlify를 사용하여 Finger Gunner 서비스를 배포 및 호스팅하였습니다.

텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

자동 생성된 설명