인공지능과 보안기술

AI 프로젝트 분석 최종 보고서

-Teachable Machine

3조

박규원 신준오 우현진 이상윤

서론

1. Teachable Machine의 정의
2. Tensorflow.js
3. Teachable Machine 학습 단계

본론

1. Teachable Machine 1.0
2. Teachable Machine 2.0
3. Teachble Machine 2.0 코드
   (1) Audio 코드
   (2) Image 코드
   (3) Pose 코드

결론

1. 실생활에서의 활용
2. 한계점 및 개선점
3. 주제 변경 이유 및 프로젝트 시 문제

참고자료

서론

1. Teachable Machine의 정의

Teachable Machine은 브라우저와 카메라만 있으면 가능한 머신러닝으로, 트레이닝 결과에 따른 output 설정이 가능하다. Teachable Machine은 누구나 기계 학습이 어떻게 작동하는지, 재미있고 실제적인 방식으로 탐구할 수 있게 하며, 코딩할 필요 없이 카메라와 브라우저만 있으면 서버에 이미지를 전송하지 않고 기기에서 로컬로 신경망을 교육할 수 있다. 또한 Google, Nikhil Thorat과 Daniel Smilkov가 TensorFlow.js를 이용하여 제작하였다. 즉 Teachable Machine은 웹 개발자들이 브라우저에서 로컬로 기계학습 모델을 교육하고 실행할 수 있도록 해주는 오픈소스 라이브러리이다.

2. Tensorflow.js

Teachable Machine의 근간이 되는 tensorflow.js(이하 tfjs)는 파이썬 모듈로 제공되던 tensorflow를 웹 상에서도 사용할 수 있게 만들어진 javascript 모듈이다. tfjs는 웹에서 사용하기 편리하도록, 제공되는 API를 사용하여 직접 ML 모델을 구축하는 것뿐만 아니라 기존에 존재하는 모델을 사용하거나, 다시 학습시키는 기능을 제공한다.

이때 기존에 존재하는 모델은 python에서 tf와 keras로 작성된 모델을 import할 수 있고, 이미지 분류, 포즈 추정, 사람 분류, 텍스트 인코딩 등의 tfjs에서 공식적으로 지원하는 모델을 사용할 수도 있다. 또한 ml5.js 라이브러리에서 제공되는 api를 이용하여 모델 설계 없이도 간단히 딥러닝을 사용할 수 있다.

3. Teachable Machine 학습 단계

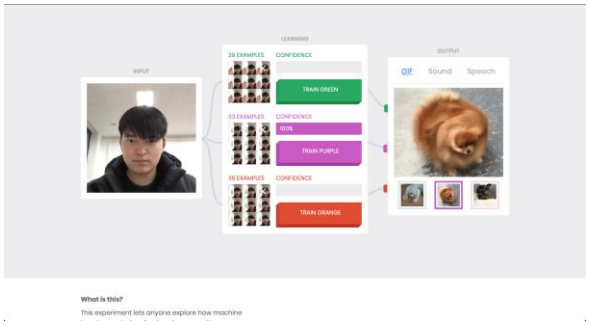
(1) Gather: 클래스나 카테고리안에 컴퓨터를 학습시키고자 하는 예시를 그룹 짓는다.

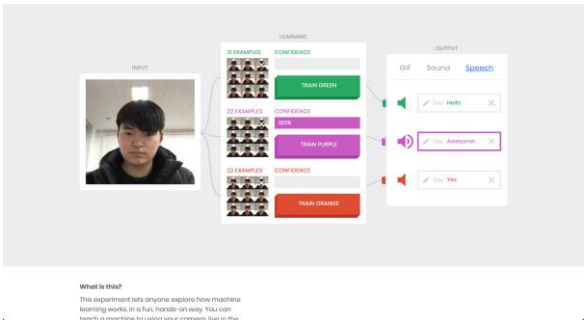
(2) Train: 모델을 훈련시키고 이것이 새 예시를 올바르게 분류하는지 확인한다.

(3) Export: 훈련시킨 모델을 사이트, 앱 그리고 더 많은 유형으로 내보낸다.

본론

1. Teachable Machine 1.0





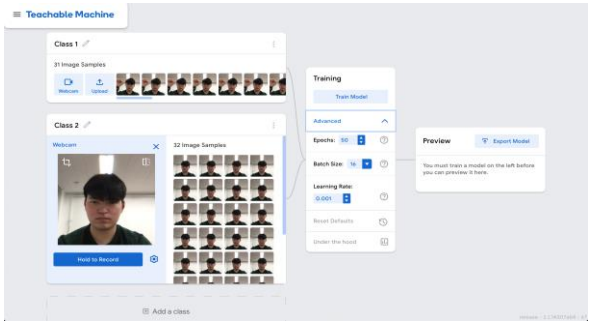
(1) 카메라를 통해 Input에서 어떻게 보이는지 확인한다.

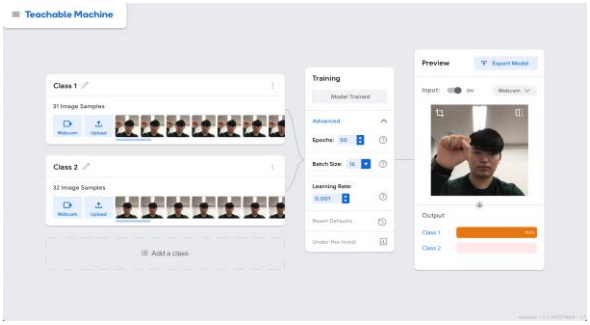
(2) 각 Learning 탭별로 다른 유형의 이미지를 30장 이상 저장한다.

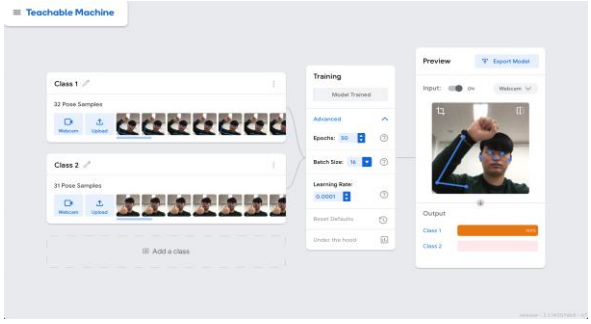
(3) Output에서 각 유형별로 인식이 되었을 때 출력될 형식 지정 가능하다. (GIF, Sound, Speech)
GIF: 유형에 연결된 GIF 이미지 실행
Sound: 유형에 연결된 임의의 음원 파일 재생
Speech: 유형에 연결된 임의의 텍스트 음성 재생

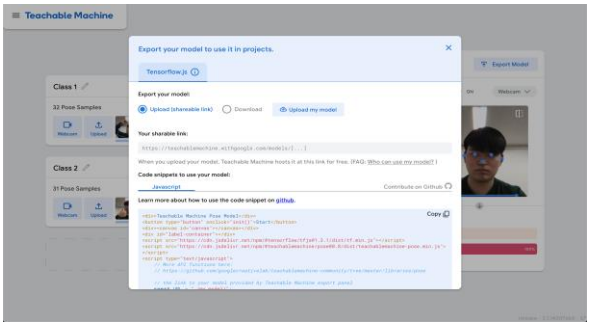
(4) 각각의 것은 다른 파일로 변경 가능하다.

2. Teachable Machine 2.0









(1) 카메라를 통해 Input에서 어떻게 보이는지 확인한다.

(2) Class별로 이미지 업로드 또는 카메라로 인식하여 샘플 저장한다.

(3) 1.0과는 달리 Input값에 이미지, 소리, 모션 인식이 가능하다.

(4) Input, Train에서 옵션값 부여 가능하다.

(5) Export로 모델을 Tensorflow에서 사용할 수 있게 다운로드 가능하다. (Input 종류에 따라 제한이 있다.)

(6) 모델을 사용하기 위한 코드 스니펫(Code Snippet)을 제공한다.

3. Teachable Machine 2.0 코드

(1) Audio 코드

```
<div>Teachable Machine Audio Model</div>
<button type='button' onclick='init()'>Start</button>
<div id='label-container'></div>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/speech-commands@0.4.0/dist/speech-commands.min.js"></script>

<script type="text/javascript">

        // teachable machine 패널로 제공되는 모델에 대한 url 을 삽입한다.
    const URL = '{{URL}}';

    async function createModel() {
        const checkpointURL = URL + 'model.json'; // 모델 토폴로지
        const metadataURL = URL + 'metadata.json'; // 모델 메타데이터

        const recognizer = speechCommands.create(
            'BROWSER_FFT', // 퓨리에 변환 유형으로, 변경될 시에는 유용하지 않음
            undefined, // 음성 명령 어휘 기능으로, 모델에 유용하지 않음
            checkpointURL,
            metadataURL);

        // 해당 모델을 확인하고 메타데이터는 https 요청을 통해 업로드 된다.
```

```
        await recognizer.ensureModelLoaded();

        return recognizer;
    }

    async function init() {
        const recognizer = await createModel();
        const classLabels = recognizer.wordLabels(); // 클래스 레이블을 얻을 수 있음
        const labelContainer = document.getElementById('label-container');
        for (let i = 0; i < classLabels.length; i++) {
            labelContainer.appendChild(document.createElement('div'));
        }

        // listen()은 두 개의 인수를 사용한다.
        // 1. 단어가 인식될 때마다 호출되는 콜백 함수
        // 2. 조정 가능한 필드의 configuration object
        recognizer.listen(result => {
            const scores = result.scores; // 예측 확률 각 클래스
            // 클래스당 확률 스코어 렌더링
            for (let i = 0; i < classLabels.length; i++) {
                const classPrediction = classLabels[i] + ': ' + result.scores[i].toFixed(2);
                labelContainer.childNodes[i].innerHTML = classPrediction;
            }
        }, {
            includeSpectrogram: true, // result.spectrogram 을 반환해야하는 경우
            probabilityThreshold: 0.75,
            invokeCallbackOnNoiseAndUnknown: true,
            overlapFactor: 0.50 // 0.5 and 0.75. 사이
        });
}
</script>
```

(2) Image 코드

```
<div>Teachable Machine Image Model</div>
<button type='button' onclick='init()'>Start</button>
<div id='webcam-container'></div>
```

```
<div id='label-container'></div>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@0.8.3/dist/teachablemachine-image.min.js"></script>
<script type="text/javascript">
// More API functions here:
// https://github.com/googlecreativelab/teachablemachine-community/tree/master/libraries/image

// Teachable Machine export를 통해 제공되는 모델 링크
const URL ='{{URL}}';

let model, webcam, labelContainer, maxPredictions;

// 이미지 모델 불러오기 및 웹캠 설정
async function init() {
const modelURL =URL +'model.json'; // 모델 변수 설정
const metadataURL =URL +'metadata.json'; // 메타데이터 변수 설정

// 모델과 메타데이터 불러오기
        // file picker의 support files API 안 tmImage.loadFromFiles() 참조
// 또는 로컬 하드디스크에 있는 파일 참조
model =await tmImage.load(modelURL, metadataURL);
maxPredictions =model.getTotalClasses();

// 웹캠을 설정하기 위한 함수
const flip =true; // 웹캠 반전여부 -> true(활성화)
webcam =new tmImage.Webcam(200, 200, flip); // 폭, 높이, 반전
await webcam.setup(); // 웹캠 액세스 요청
webcam.play();
window.requestAnimationFrame(loop);

// DOM에 요소 추가
document.getElementById('webcam-container').appendChild(webcam.canvas);
```

```
labelContainer =document.getElementById('label-container');
for (let i =0; i < maxPredictions; i++) { // and class labels
labelContainer.appendChild(document.createElement('div'));
}
}

async function loop() {
webcam.update(); // 웹캠 프레임 업데이트
await predict();
window.requestAnimationFrame(loop);
}

// 이미지 모델을 통해 웹캠 이미지 실행
async function predict() {
// 예측 후 이미지, 동영상, html 요소를 가져올 수 있음
const prediction =await model.predict(webcam.canvas);
for (let i =0; i < maxPredictions; i++) {
const classPrediction =
prediction[i].className +': '+ prediction[i].probability.toFixed(2);
labelContainer.childNodes[i].innerHTML = classPrediction;
}
}
</script>
```

Image API

1. Loading the model – url checkpoints

tmImage는 script src를 사용할 때 자동으로 추가되는 모듈이다.

객체로 추가되기에 이용자가 window.tmImage나 단순 tmImage를 통해 접근할 수 있다.

```
tmImage.load(
        checkpoint: string,
        metadata?: string | Metadata
)
```

checkpoint: 모델의 topology와 bin 파일에 대한 참조가 포함된 json 파일의 URL(모델 가중치)

metadata: 모델의 텍스트 레이블과 추가 정보가 들어 있는 json 파일의 URL

사용법: await tmImage.load(checkpointURL, metadataURL)

1. Loading the model – browser files

이용자는 file picker와 file 인터페이스를 사용하여 로컬 하드드라이브의 모델 파일을 업로드 가능

```
tmImage.loadFromFiles(
        model: File,
        weight: File,
        metadata: File
)
```

model: 모델 topology를 토함하는 파일 객체(.json)

weights: 모델 가중치가 있는 파일 객체(.bin)

metadata: 이용자 모델의 텍스트 라벨과 추가정보를 담고 있는 파일 객체(.json)

사용법: const uploadModel = document.getElementById('upload-model');

const uploadWeights = document.getElementById('upload-weight');

const uploadMetadata = document.getElementById('upload-metadata');

model = await tmImage.loadFromFiles(uploadModel.files[0], uploadWeights.files[0], uploadMetadata.files[0])

2. Model – get total classes

모델을 불러오면 모델 내의 클래스 총 수를 획득 가능

이 메소드는 tmImage.load에서 불러와짐

```
model.getTotalClasses()
```

3. Model – get class labels

모델을 불러오면 모델 내의 클래스 제목 획득 가능

이 메소드는 tmImage.getClassLabels에서 불러와짐

```
model.getClassLabels()
```

4. Model – predict

모델을 불러오면 몇 개의 다른 인풋 옵션을 가진 분류를 만들 수 있음

이 메소드는 tmImage.load에서 불러와짐

```
model.predict(
        image: HTMLImageElement | HTMLCanvasElement | HTMLVideoElement | ImageBitmap,
        flipped = false
)
```

image: 분류를 하기 위한 이미지, 캔버스 또는 비디오 요소

flipped: 입력하는 이미지를 반전시킬지 여부를 결정하는 불리언 요소

사용법:

```
const flip = true;
const allPredictions = await model.predict(webcamElement, flip);
```

5. Model – predictTopK

모든 클래스에 대한 확률을 반환하는 predict()를 대체하는 함수

이 메소드는 tmImage.load에서 불러와짐

```
model.predictTopK(
        image: HTMLImageElement | HTMLCanvasElement | HTMLVideoElement | ImageBitmap,
        maxPredictions = 10,
        flipped = false
)
```

image: 분류를 하기 위한 이미지, 캔버스 또는 비디오 요소

flipped: 입력하는 이미지를 반전시킬지 여부를 결정하는 불리언 요소

maxPredictions: 반환할 예측의 총 값

사용법:

```
const flip = true;
const maxPredictions = model.getTotalClasses();
const prediction = await model.predictTopK(webcamElement, maxPredictions, flip);
```

6. Webcam

선택적으로 라이브러리와 함께 제공되는 웹캠 클래스를 사용하거나 웹캠을 회전시킬 수 있다.
이 클래스는 tmImage 모듈에 존재함

teachable machine에서는 flipped가 x에 기본값이 있기 때문에 이용자 자신의 웹캠을 만들기 위해서는 filp값을 true로 설정해야한다.

```
new tmImage.Webcam(
        width = 400,
        height = 400,
        flip = false,
)
```

width: 웹캠의 폭. Teachable Machine으로 훈련된 모델이기 때문에 이상적으로는 사각형
height: 웹캠의 높이. Teachable Machine으로 훈련된 모델이기 때문에 이상적으로는 사각형
flip: 웹캠이 반전되었는지 아닌지에 대한 불리언 신호. 이것인 오직 CSS에서만 적용된다는 점 유의

사용법:

```
const webcam = new tmImage.Webcam(200, 200, true);
await webcam.setup();
webcam.play();
document.body.appendChild(webcam.canvas);
```

7. Webcam – setup

웹캠 객체를 생성한 후 이용자는 설정하기 위해 셋업을 한 번 콜해야함

```
webcam.setup(
        options: MediaTrackConstraints = {}
)
```

options: 웹캠에 대한 선택적 미디어 트랙 제약사항

사용법:

```
await webcam.setup();
```

8. Webcam – play, pause, stop

```
webcam.play();
webcam.pause();
webcam.stop();
```

웹캠은 loads를 재생하고 미디어 리소스의 재생을 시작

9. Webcam – update

웹캠 프레임의 업데이트를 위한 콜

```
webcam.update();
```

10. 기타 코드

1) Teachable 모델이 훈련되었는지 확인

```
public get isTrained() {
        return !!this.model && this.model.layers && this.model.layers.length > 2;
}
```

2) 데이터 셋이 라벨이 다 있고 샘플이 진행된 채로 준비되었는지 확인

```
public get isPrepared() {
        return !!this.trainDataset;
}
```

3) 데이터 셋에 클래스가 얼마나 있는지 확인

```
public get numClasses(): number {
        return this._metadata.labels.length;
}
```

(3) Pose 코드

1.Script tag를 통해 자동으로 포함된 tmpose 모듈을 "window.tmPose"나 "tmPose"로 접속한다.

-checkpoint : 모델 토폴로지와 bin파일 레퍼런스를 포함하는 json file URL

-metadata : 모델의 text label과 추가 정보를 포함하는 json file URL

```
await tmPose.load(checkpointURL, metadataURL);
```

2. 로컬 하드드라이브를 통해 모델 파일들을 불러온다.

```
// you need to create File objects, like with file input elements (<input type="file" ...>)
const uploadModel = document.getElementById('upload-model');
const uploadWeights = document.getElementById('upload-weights');
const uploadMetadata = document.getElementById('upload-metadata');
model = await tmPose.loadFromFiles(uploadModel.files[0], uploadWeights.files[0], uploadMetadata.files[0])
```

-model : 모델 토폴로지를 포함하는 파일(.json) **-weights** : 모델 weight(.bin)

-metadata : 모델의 text label과 추가 정보를 포함하는 파일(.json)

3.모델들의 class 총 개수를 부여해준다.

```
model.getTotalClasses()
```

4.posenet을 통해, 그리고 teachable machine의 분류 모델을 통해 두 가지 방법으로 input 을 측정한다.

posenet의 output값이 나오면 훈련시킨 teachable machine 모델로 분류할 수 있다.

predict()함수를 통해 모든 class에 확률치를 부여해준다.

```
// predictTopK can take in an image, video or canvas html element
// if using the webcam utility, we set flip to true since the webcam was only
// flipped in CSS
const maxPredictions = model.getTotalClasses();
const flipHorizontal = false;

const { pose, posenetOutput } = await model.estimatePose(webcamElement, flipHorizontal);
const prediction = await model.predictTopK(posenetOutput, maxPredictions);
```

-sample : posenet을 통과하는 이미지, 캔버스나 비디오

-flipHorizontal : 포즈의 키포인트 flip = x 설정 여부에 대한 불리언

-poseOutput : mode.estimatePose함수를 통해 나온 posenet의 output 정렬

-maxPredictions : return받을 총 prediction의 개수

5. Webcam 라이브러리를 불러오기 위해 webcam class를 사용할 수 있다.

```
// webcam has a square ratio and is flipped by default to match training
const webcam = new tmPose.Webcam(200, 200, true);
await webcam.setup();
webcam.play();
document.body.appendChild(webcam.canvas);
```

(여기서 모델이 teachable machine에서 훈련받은 대로 정사각형이어야 하므로 width = height)

-width : webcam의 너비 **-height** : webcam의 높이 **-flip** : webcam의 flip=x 설정에 대한 불리언

6. Webcam 설정값 설정

```
await webcam.setup();
```

7. webcam의 play,pause,stop 설정

```
webcam.play();
webcam.pause();
webcam.stop();
```

8. webcam frame 업데이트

```
webcam.update();
```

9. 기능함수를 통해 pose의 keypoints를 그리고 skeleton까지 완성한다.

```
const flipHorizontal = false;
const { pose, posenetOutput } = await model.estimatePose(webcamEl, flipHorizontal);

const minPartConfidence = 0.5;
tmPose.drawKeypoints(pose.keypoints, minPartConfidence, canvasContext);
tmPose.drawSkeleton(pose.keypoints, minPartConfidence, canvasContext);
```

-keypoints : keypoints 정렬

-minConfidence : minConfidence 기준 이하로는 confidence 값이 없다.

-ctx : keypoints들을 그릴 캔버스 **-keypointsSize** : drawing을 위한 keypoints의 사이즈

-fillColor: css의 fillColor **-strokeColor** : css의 stroke colo

결론

1. 실생활에서의 적용

Teachable Machine 은 가르치는 용도로만 사용되는 것이 아니다. Teachable Machine 은 머신러닝 학습 도구지만, 실생활에서 사용될 수 있다. 종이컵과 플라스틱 페트병의 구별, 반려동물을 포함한 가족들의 얼굴 인식, 다양한 품종의 장미 이미지에 대한 학습, 장애인을 위한 의사소통 도구 등으로 활용할 수 있다. 프로젝트로 만들어진 모델을 물리적인 컴퓨팅 도구와 연동하면, 색상에 따라 사탕을 분류하거나, 집에서 기르는 반려동물을 인식해 자동으로 문을 열어주는 용도로 활용하는 것이 가능하다.

실제로 접근성 기술 전문가인 스티브 살링은 언어 장애가 있는 사람들의 의사소통을 향상시키는 데 티처블 머신을 사용했다. Teachable Machine 은 PC 용 웹캠과 브라우저만 있으면 프로그램에 대한 지식이 없어도 누구나 머신러닝을 체험해볼 수 있다. 그렇기 때문에 인공지능 전문가나 관련 지식을 가진 사람뿐만 아니라 청소년, 심지어 어린이까지 쉽게 머신러닝을 체험해볼 수 있다.

2. 한계점 및 개선점

[한계점]

(1) 소리의 경우 브라우저에 따라 지원 여부가 갈린다. (크롬, 파이어폭스 권장)

(2) 모션의 경우 웹캡으로 인식되는 것이 한계가 있다. Ex) 주먹은 인식하지 못해 Class 2와 다른 모션임에도 2에 100%를 보여준다.

(3) 저장된 오디오 모델이 'Error: Unsupported URL scheme in metadata URL' 문제를 일으키는 현상이 나타난다.

➔ Google Creative Lab 측의 답: Tensorflow.js 는 모델을 로드할 때 프로토콜을 지정해야 하므로 해당 경로를 http/https 를 포함하여 모델 파일을 호스팅할 전체 경로로 확장해야 하고, 위의 문제에 대해 스니펫에 안내글을 추가할 예정이다.

[개선점]

(1) NodeJS 지원문제
-> Google Creative Lab 측의 답: 추후 업데이트 계획 시 고려하겠다.

(2) MP3를 포함하는 데이터 셋 사용 여부
-> Google Creative Lab 측의 답: 포함 일자는 불명확하지만 추가하고 싶은 기능이다.

(3) 오디오 모델의 파이썬 Exporting
-> Google Creative Lab 측의 답: 희망사항이지만, 네트워크가 예측하는 형식으로 오디오 데이터를 전처리한 것이 파이썬용으로 작성되지 않았다. 오디오의 처리는 자바스크립트와 파이썬에서 기본적으로 다르게 처리되어 까다롭다.

3. 주제 변경 이유 및 프로젝트 시 문제

Fast.ai 프로젝트를 분석하고 코드를 실행하는 과정에서 오류 발생으로 인해 코드 실행을 완료할 수 없었다. 라이브러리 실행 과정에서 course 부분이 ver 1.0 이후부터는 사용할 수가 없어 example 내부 파일을 바탕으로 실행을 해야 하는데, 이에 대한 사용 방안이 제공되지 않아 실행을 이어 나갈 수 없었다. 또한 course ver 3.0 기준으로 작성된 코드에서도 오류를 해결할 수 없었다. 즉, 데이터 파일 다운로드 후에 라이브러리 로드 문제가 있었고 1.0 이후 버전에 대해 사용법이 제시되어 있지 않았다. 그렇기 때문에 Fast.ai와 유사한 프로젝트인 Teachable Machine로 분석 대상을 변경하였다.

참고 자료

1. A.I. Experiments: Teachable Machine . (2017).

2. Teachable Machine 2.0: Making AI easier for everyone . (2019).

3. https://youtu.be/T2qQGqZxkD0

4. https://youtu.be/3BhkeY974Rg.

5. Teachable Machine . (n.d.). https://teachablemachine.withgoogle.com/v1/

6. Teachable Machine . (n.d.). https://teachablemachine.withgoogle.com

7. "누구나 활용하는 웹기반 머신러닝 툴"… 구글, 티처블 머신 2.0 공개,

http://www.ciokorea.com/news/136039

8. https://github.com/googlecreativelab/teachablemachine-community