서버 자원 기반 사전훈련모델(Pre-trained Model)을 이용한 웨이퍼 결함패턴 분류 알고리즘 개발

2020.01~2020.02

github: https://github.com/joyoon1110/teachable_machine

Table of Contents	Page No
웹 브라우저에서 동작하는 Tensorflow.js에 대하여 조사하고, 선형회귀 문제를 구현한다.	3
Teachable Machine을 구현하기 위한 JavaScript 환경을 구성하고, MNIST 데이터를 로드 과정을 구현한다.	5
신경망을 학습하기 위한 모델 아키텍처를 정의하고, 로드한 MNIST 데이터를 사용하여 훈련 및 평가를 진행한다.	6
Google Teachable Machine을 분석하고, UI/UX를 고려하여 클라우드 서비스 개발을 검토한다.	8
Google Teachable Machine을 분석 결과를 바탕으로 중간결과를 도출하고, 클라우드 서비스를 위한 UI/UX 및 화면 설계한다.	11
클라우드 서비스를 위한 UI/UX 및 화면을 추가적으로 설계하고, 각 화면 진행과정에 스크립트 동작에 따른 디렉토리 구조를 설계한다.	13
설계한 디렉토리 구조를 바탕으로 데이터 생성 스크립트를 작성 및 테스트를 진행한다.	15
로컬에 GPU 환경을 구성하고, 데이터 생성 스크립트 성능평가를 진행한다.	16

연구 노트

연구 목표

웹 브라우저에서 동작하는 Tensorflow.js에 대하여 조사하고, 선형회귀 문제를 구현한다.

개념

1. Tensorflow.js

Tensorflow.js는 웹 버전의 머신러닝 플랫폼으로 기존에 deeplearn.js가 이름을 바꾼 것임. TensorFlow.js는 JavaScript에서 텐서를 사용하여 계산을 정의하고 실행하는 프레임 워크임. 학습된 모델을 불러와서 쓰거나 불러온 모델을 학습 시킬 수 있는 기능을 제공함. 또한, 직접 모델을 작성할 수 있음.

2. 선형회귀 문제

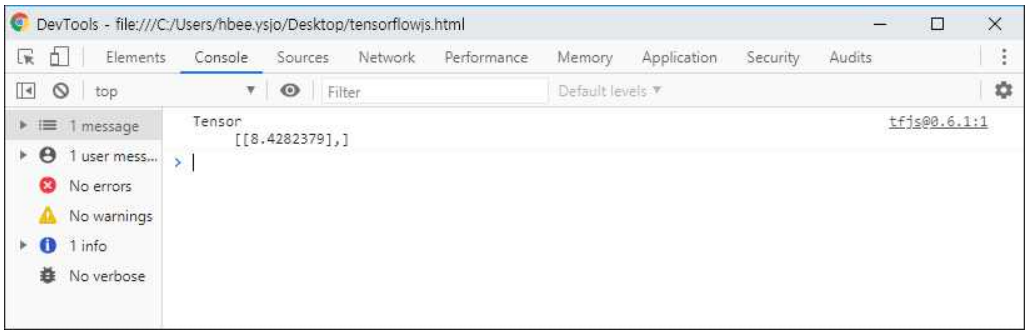

그림 1. Tensorflow.js 선형회귀 실행결과 화면

TensorFlow.js는 브라우저와 Node.js에서 작동하며 두 플랫폼 모두에서 사용 가능한 구성이 다양함. TensorFlow.js 프로그램이 실행될 때 특정 구성을 환경이라고 함. 환경은 단일 글로벌 백엔드와 TensorFlow.js의 세분화 된 기능을 제어하는 플래그 세트로 구성됨. TensorFlow.js는 텐서 스토리지 및 수학 연산을 구현하는 여러 가지 다른 백엔드를 지원함. 대부분의 경우, TensorFlow.js는 현재 환경에 따라 가장 적합한 백엔드를 자동으로 선택함.

Tensorflow.js를 설치하기 위해서 Node.js가 사전에 설치되어있어야 하며 설치 방법은 아래 참고문헌의 2번째 url을 참고하여 설치를 진행함. 그림 1은 그림 2의 선형회귀 구현 결과를 나타냄.

*참고문헌: https://neurowhai.tistory.com/156
*참고문헌: 윈도우 npm 설치(https://blog.danggun.net/4147)
*참고문헌: tensorflow.js 설치(https://www.tensorflow.org/js/tutorials/setup)

```
<!doctype html>
<html>
<head>
    <title>TF.js Test</title>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.6.1"></script>
    <script type="text/javascript">
        // 선형회귀 모델 생성
        const model = tf.sequential();
        model.add(tf.layers.dense({units: 1, inputShape: [1]}));

        // 학습을 위한 준비 : 손실 함수와 최적화 함수를 설정
        model.compile({loss: 'meanSquaredError', optimizer: 'sgd'});

        // 학습 데이터 생성
        const xs = tf.tensor2d([1, 2, 3, 4], [4, 1]);
        const ys = tf.tensor2d([1, 3, 5, 7], [4, 1]);

        // 데이터를 사용해서 학습
        model.fit(xs, ys).then(() => {
            // 학습된 모델을 가지고 추론
            model.predict(tf.tensor2d([5], [1, 1])).print();
        });
    </script>
</head>
<body>
    콘솔을 확인하세요.
</body>
</html>
```

그림 2. Tensorflow.js 선형회귀 구현

연구 노트

연구 목표

Teachable Machine을 구현하기 위한 JavaScript 환경을 구성하고, MNIST 데이터를 로드 과정을 구현한다.

개념

1. JavaScript 환경 구성

Teachable Machine을 JavaScript로 개발하기 위해 Chrome과 같은 브라우저가 필요하며, 브라우저 권한에 따라 CORS 제한을 피하기 위해 로컬 웹 서버(Python SimpleServer 또는 Node http 서버)를 사용하여 파일을 볼 수 있음.

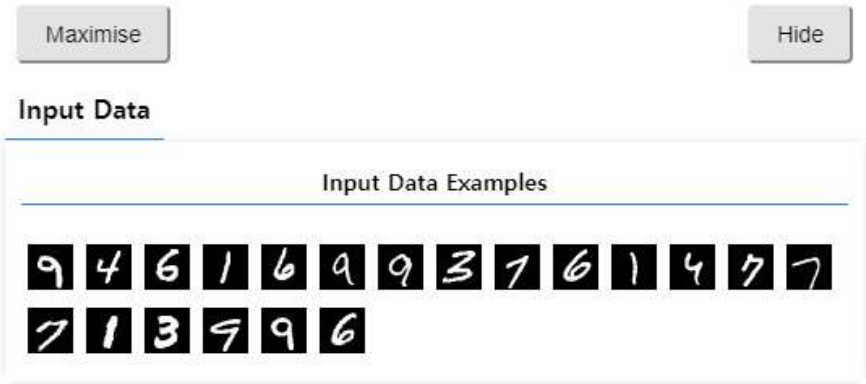

그림 1. MNIST 데이터 로드 화면

2. 데이터 로드

-CSS 이미지 스파라이트 파일(Image Sprite file): 스프라이트 파일(sprite file)이란 여러 개의 이미지를 하나의 이미지로 합쳐서 관리하는 이미지를 의미

-웹 브라우저에서 서버에 이미지를 요청할 경우의 문제점 및 효과

웹 페이지에 이미지가 사용될 경우 해당 이미지를 다운받기 위해 웹 브라우저는 서버에 이미지를 요청해야 하지만 사용된 이미지가 많을 경우, 웹 브라우저는 서버에 해당 이미지의 수만큼 요청해야만 하므로 웹 페이지의 로딩 시간이 오래 걸리게 됨. 스프라이트 파일을 사용하면 이미지를 다운받기 위한 서버 요청을 단 몇 번으로 줄일 수 있음.

*참고문헌: http://tcpschool.com/css/css_basic_imageSprites

*참고문헌: https://www.tensorflow.org/js/tutorials/training/handwritten_digit_cnn

연구 노트

연구 목표

신경망을 학습하기 위한 모델 아키텍처를 정의하고, 로드한 MNIST 데이터를 사용하여 훈련 및 평가를 진행한다.

개념

1. 모델 아키텍처 정의 및 훈련, 평가

Input Data | Visor | Evaluation

Model Architecture

Layer Name	Output Shape	# Of Params	Trainable
conv2d_Conv2D1	[batch,24,24,8]	208	true
max_pooling2d_MaxPooling2D1	[batch,12,12,8]	0	true
conv2d_Conv2D2	[batch,8,8,16]	3,216	true
max_pooling2d_MaxPooling2D2	[batch,4,4,16]	0	true
flatten_Flatten1	[batch,256]	0	true
dense_Dense1	[batch,10]	2,570	true

그림 1. 모델 아키텍처

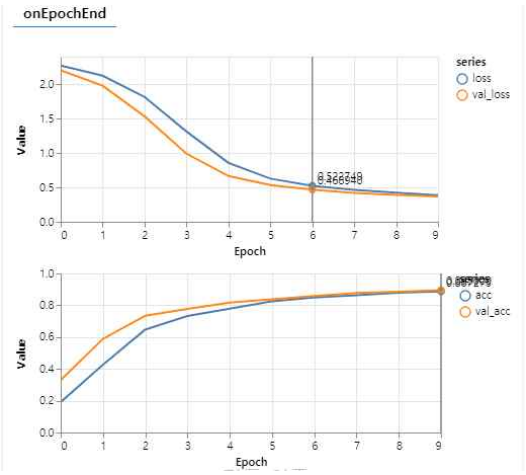

그림 2. 트레이닝 그래프

모델 아키텍처는 "실행할 때 어떤 함수가 모델을 실행할 것인가" 또는 대안으로 "모델이 응답을 계산하는 데 사용할 알고리즘 "을 말하는 방법으로 머신 러닝에서 아키텍처 (또는 알고리즘)를 정의하고 트레이닝이 해당 알고리즘의 매개 변수를 학습하게 함.

그림 1은 데이터 셋을 학습하기 위한 네트워크 구조이며 학습이 진행된 후 몇 초후에 트레이닝 진행 상황을 나타내는 그래프가 그림 2와 같이 표시됨.

```
model.add(tf.layers.conv2d({
  inputShape: [IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS],
  kernelSize: 5, filters: 8, strides: 1, activation: 'relu',
  kernelInitializer: 'varianceScaling' }));
```

표 1. 아키텍처 설계 예시

그림 3에서 유효성 검사 정확도는 데이터가 유효성 검사 세트와 유사하다면 이전에 보지 못한 데이터에 대해 모델이 얼마나 잘 수행하는지에 대한 추정치를 제공함. 혼동 행렬은 클래스 별 정확도와 유사하지만 잘못 분류 된 패턴을 보여주기 위해 세분화함. 모델이 특정 클래스 쌍에 대해 혼란스러워하는지 확인할 수 있음(그림 4).

Input Data Visor Evaluation

Accuracy

Class	Accuracy	# Samples
Zero	1	39
One	1	55
Two	0.8163	49
Three	0.9184	49
Four	0.9811	53
Five	0.8043	46
Six	0.8696	46
Seven	0.8302	53
Eight	0.717	53
Nine	0.807	57

그림 3. 평가 정확도

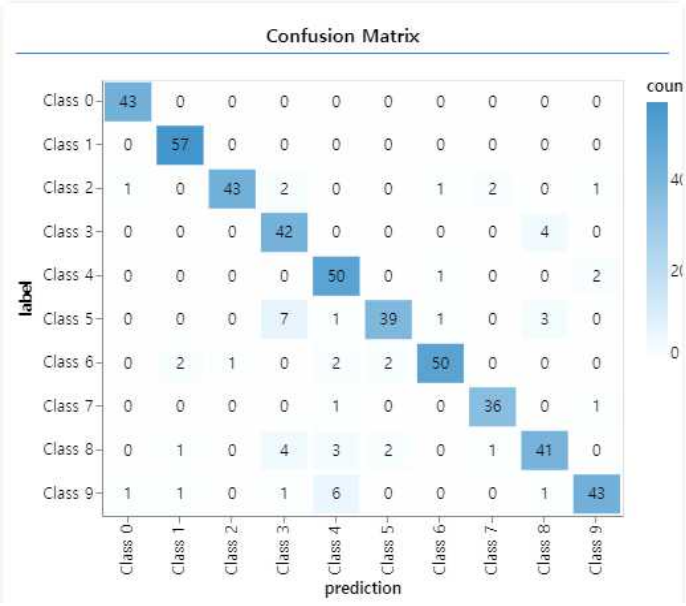

그림 4. 혼동 행렬

연구 노트

연구 목표

Google Teachable Machine을 분석하고, UI/UX를 고려하여 클라우드 서비스 개발을 검토한다.

개념

1. Google Teachable Machine UI/UX 분석

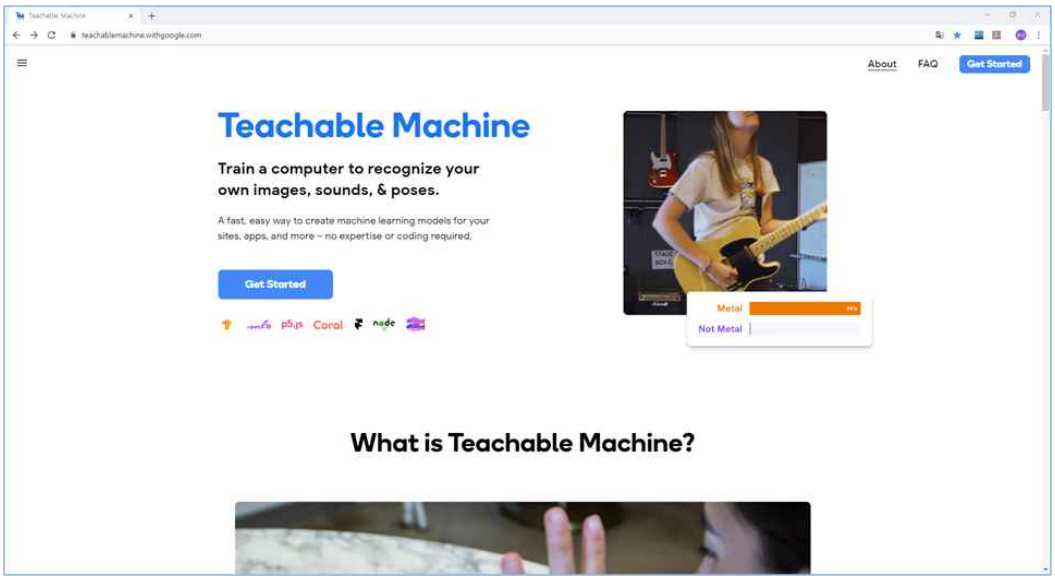

그림 1. 메인 화면

Teachable machine의 홈페이지를 접속하면 가장 처음 보이는 화면으로 홈페이지의 메인 화면에서 gif 그림을 통해 Teachable machine을 활용하여 이용 가능한 도메인 영역을 미리 볼 수 있음. 또한, Teachable machine은 웹 기반 도구로, 누구나 쉽고 빠르게 액세스 할 수 있는 기계 학습 모델을 만들 수 있다는 것을 보여주기 위해 메인 화면에 유튜브 영상을 삽입하였음.

화면 좌상단의 전체메뉴 아이콘이나 우상단의 'Get Started' 버튼 또는 메인 화면은 가운데 'Get Started'버튼을 클릭하여 프로젝트 생성 화면으로 이동할 수 있음.

메인 화면의 아래쪽에는 사용법, 사용하기 위한 데이터 종류, Teachable Machine으로 진행한 프로젝트들과 Teachable machine 프로젝트를 만들기 위해 사용한 도구들이 소개되었음. Teachable Machine으로 할 수 있는 프로젝트는 주로 분류를 수행할 수 있도록 구성되었음.

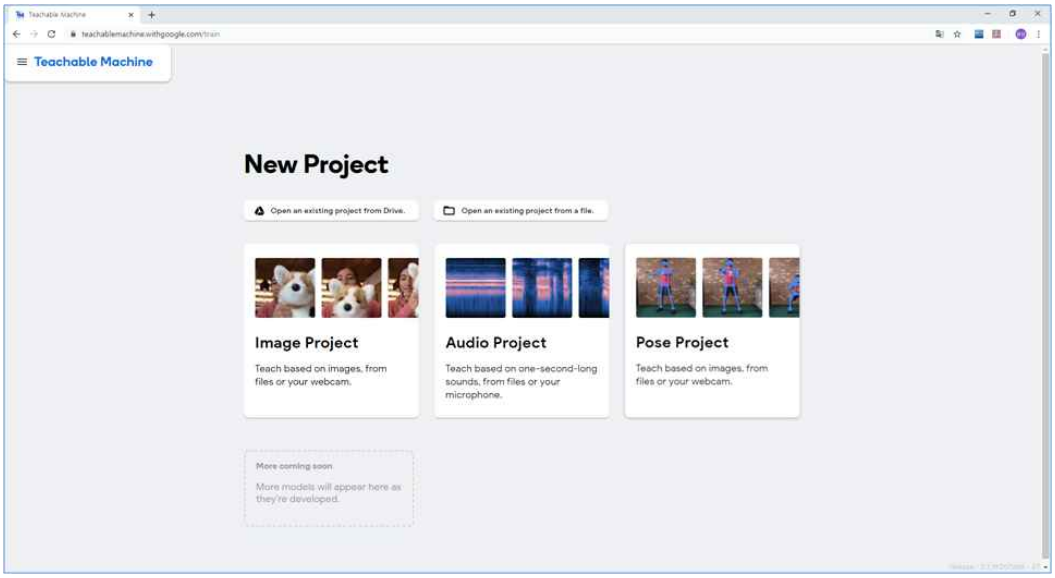

그림 2. 프로젝트 생성화면

프로젝트 생성화면으로 사용자의 목적에 따라 Image, Audio, Pose 프로젝트를 생성할 수 있으며 기존에 로컬 또는 클라우드에 저장된 기존의 프로젝트를 가져올 수 있음.

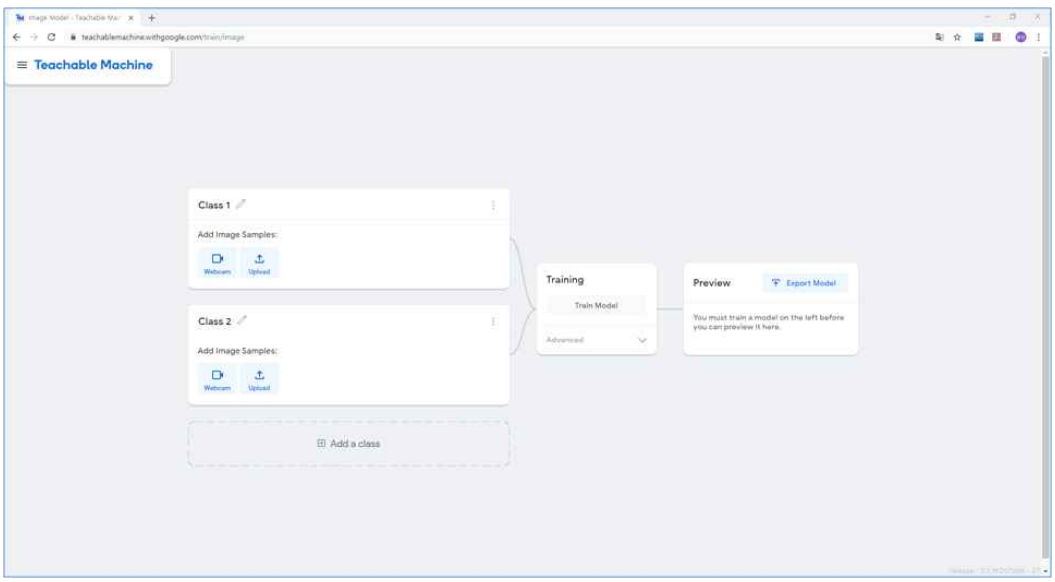

그림 3. 트레이닝 화면

프로젝트를 생성한 뒤에 학습 데이터 트레이닝을 위한 화면으로 클래스 추가, 훈련 파라미터 설정, 테스트 이미지 예측 및 모델 배포 화면으로 구성됨.

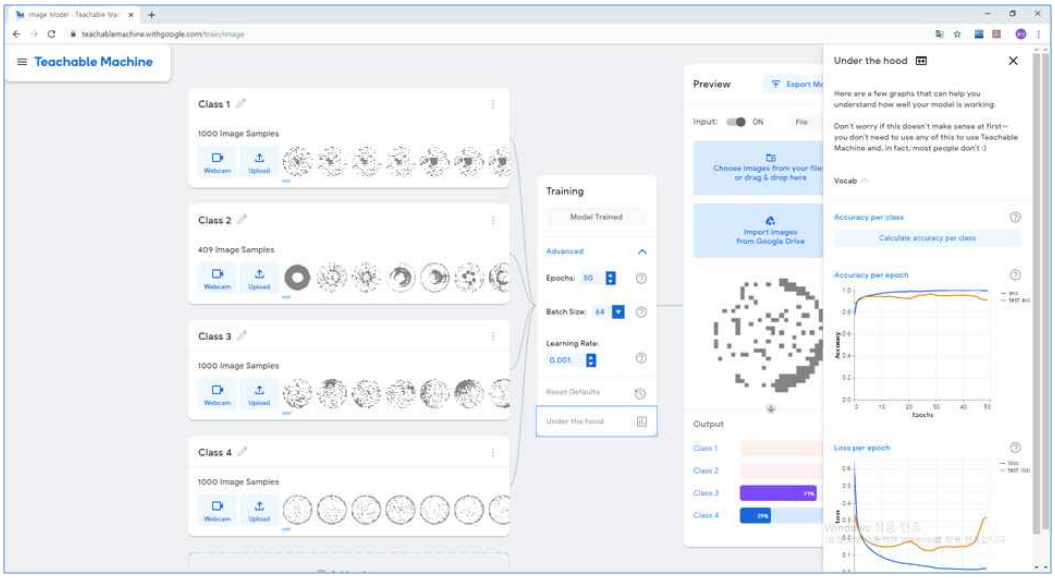

그림 4. 트레이닝 완료 예시

학습 데이터를 훈련하기 위해 제일 먼저 해야 하는 과정은 ‘클래스 추가’ 과정임. 이미지 프로젝트에서는 ‘이미지’를 업로드 하거나 ‘웹캠’ 영상으로부터 이미지를 입력받도록 할 수 있음. 각 클래스별 데이터의 개수가 표시되며, 각 클래스 내에 샘플을 개별적으로 지울 우 있음.

클래스 추가 과정이 끝나면 훈련파라미터를 설정하는 단계로 'Epoch', 'Batch size', 'Learning rate'를 변경하여 학습을 조절할 수 있도록 하였음. 또한 머신러닝에 익숙하지 않은 사용자를 배려하여 각 옵션에 도움말 기능을 넣어 좋은 학습 모델을 만들기 위해서 훈련 파라미터를 어떻게 설정하면 좋은지 팁을 제공함. ‘Training' 박스의 제일 아래 'Under the hood' 버튼을 클릭하면 확장된 도움말 기능 및 그래프 보기 기능을 제공함. 그래프 보기 기능은 학습 횟수에 따른 손실 그래프와 학습 횟수에 따른 정확도 그래프를 보여줌.

사용자 업로드 클래스에 대한 데이터 학습이 끝나면 훈련에 사용하지 않은 테스트 이미지를 업로드하거나 웹캠으로부터 영상을 입력받아 예측작업을 수행할 수 있음. 예측작업의 결과물로 각 클래스에 대한 확률값을 리턴함.

마지막으로 프로젝트에서 생성된 모델 배포을 배포할 수 있음. Teachable Machine에서 제공하는 모델 배포 확장자는 케라스에서 학습 모델을 저장할 때 쓰이는 '.h5' 확장자 또는 텐서플로에서 사용하는 ‘pb’, ‘variable’이 저장되어 있는 파일을 배포함으로써 실제 서비스에 응용하기 위한 환경을 제공함.

연구 노트

연구 목표

Google Teachable Machine을 분석 결과를 바탕으로 중간결과를 도출하고, 클라우드 서비스를 위한 UI/UX 및 화면 설계한다.

개념

1. Teachable Machine UI/UX 화면 설계

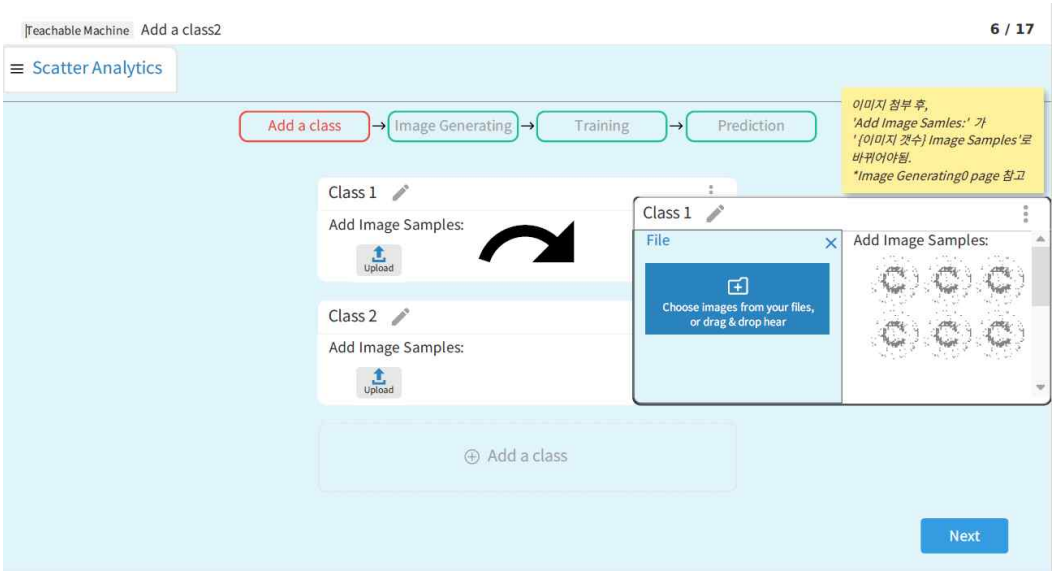

그림 1. 클래스 관리 화면

1) 분석 및 중간 도출 결과

- Google Teachable Machine은 Tensorflow.js로 개발였으며 로컬 사용자의 컴퓨터 리소스를 사용하여 동시 사용자에 대한 제한이 없음.
- 본 클라우드 서비스는 분류 문제뿐만 아니라 데이터 생성과정을 포함하므로 GPU 사용이 필수적으로 요구됨.

2)스크립트 실행 예시

Generate 단계: python generate_wafer.py
--generating_dir="C:/User/hbee_ysjo/wafer/generated_image"
--status="[{"name": "class1", "ai_use": True}, {"name": "class2", "ai_use": False}]"

Training 단계: python training.py
--training_dir="C:/Users/hbee.ysjo/wafer_class2"
--epochs=100 --batch_size=64 --learning_rate=0.001

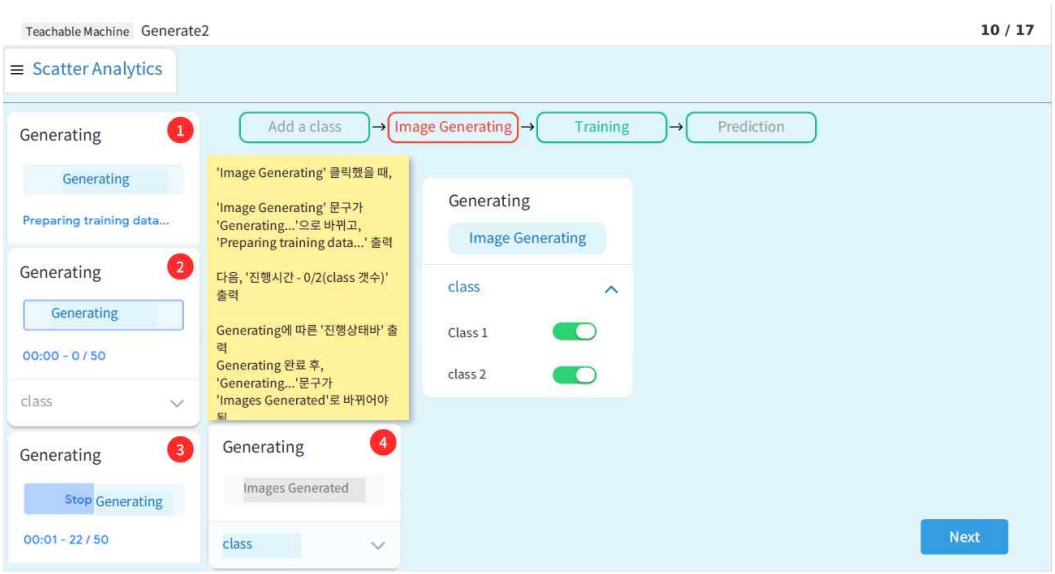

그림 2. 이미지 데이터 생성 화면

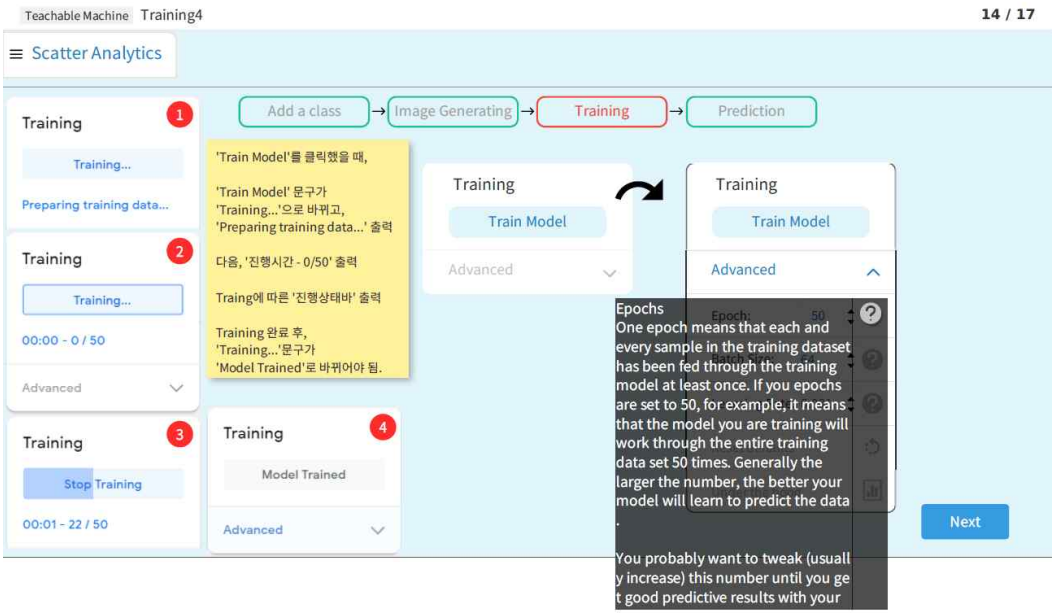

그림 3. 데이터 training 화면

연구 노트

연구 목표

클라우드 서비스를 위한 UI/UX 및 화면을 추가적으로 설계하고, 각 화면 진행과정에 스크립트 동작에 따른 디렉토리 구조를 설계한다.

개념

1. Teachable Machine UI/UX 추가 화면 설계

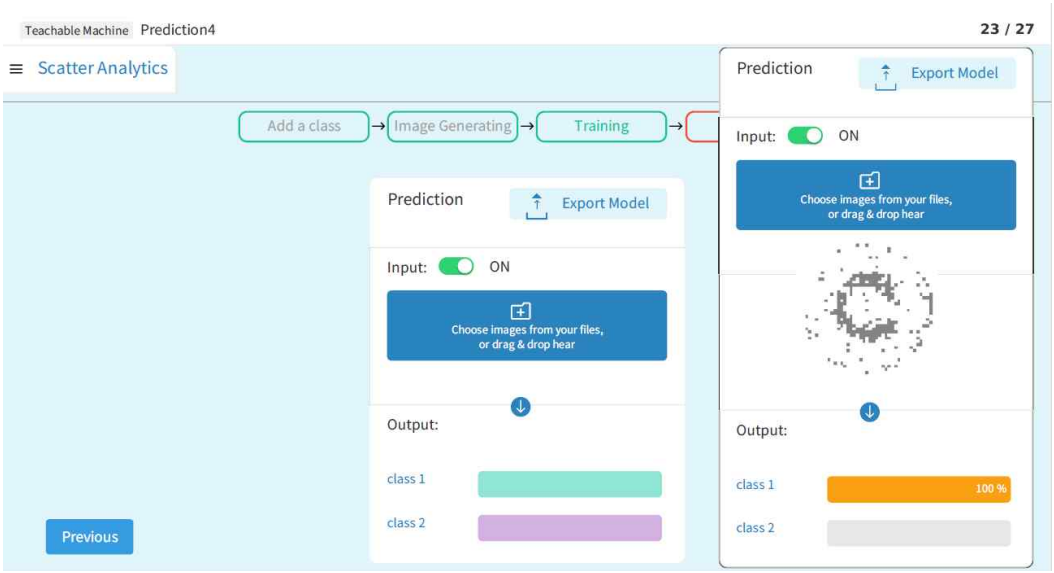

그림 1. 테스트 데이터 예측 화면

클라우드 서비스 제공 목적에 따라 메인 화면(그림 2) 및 테스트 데이터 예측 화면(그림 1)을 추가적으로 설계하였음. 메인 화면에서는 기존의 프로젝트 생성기능 외에 기존 프로젝트 불러오기, 프로젝트 다운받기, 메인화면으로 이동, FAQ와 feedback이 가능하도록 메시지 보내기 기능을 추가적으로 구성함. 예측 화면에서는 학습에 사용하지 않은 데이터를 업로드(드래그 앤 드랍)하여 학습 모델에서 리턴하는 확률값에 따라 해당 클래스의 확률을 막대바 형태로 표시하도록 구성함.

스크립트 동작에 따른 디렉토리 구조를 설계하였음(그림 3). 기본적으로 스크립트를 관리하는 bcs 폴더 안에 데이터 생성, 학습, 예측하는 스크립트가 있고, 하위에 프로젝트 관리 폴더를 구성함. 프로젝트 관리 폴더 하위에는 각 개별 프로젝트를 구성하여 사용자 업로드 데이터, 생성 데이터, 학습 후 만들어진 모델 파일을 관리하도록 구상 하였음. 현재 단계에서 스크립트의 동작 방식은 기존과 동일함.

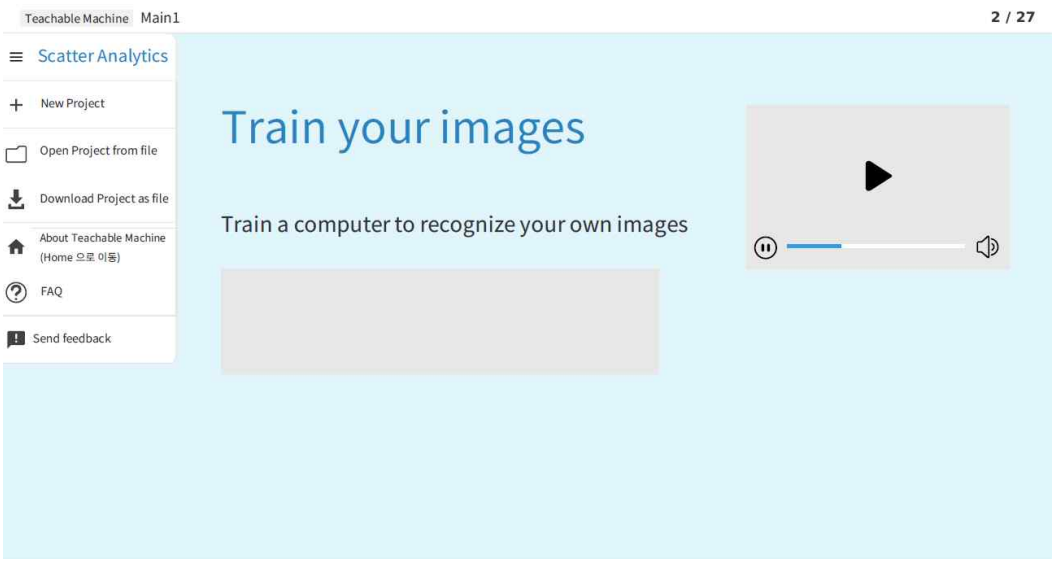

그림 2. 메인 화면

2. 디렉토리 구조

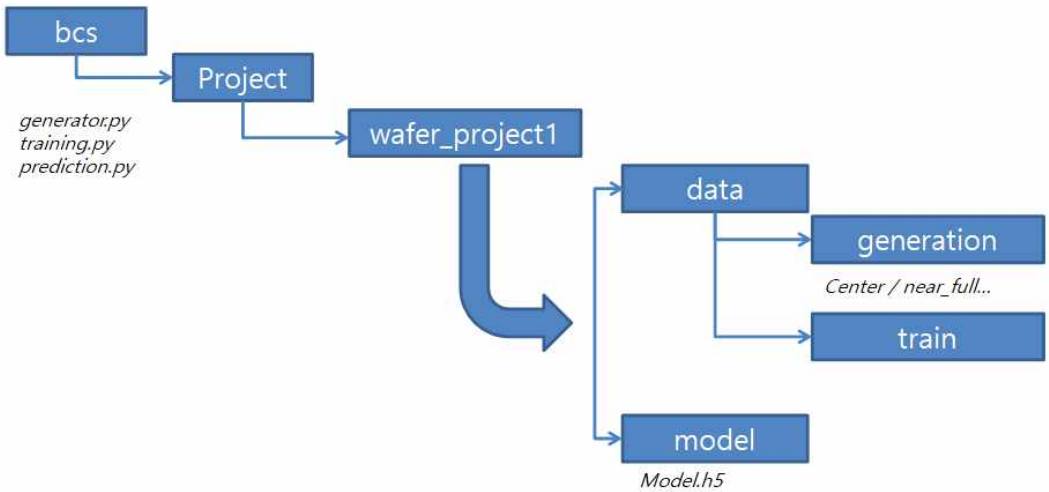

그림 3. 디렉토리 구조

연구 노트

연구 목표

설계한 디렉토리 구조를 바탕으로 데이터 생성 스크립트를 작성 및 테스트를 진행한다.

개념

1. Teachable Machine 데이터 생성 작성

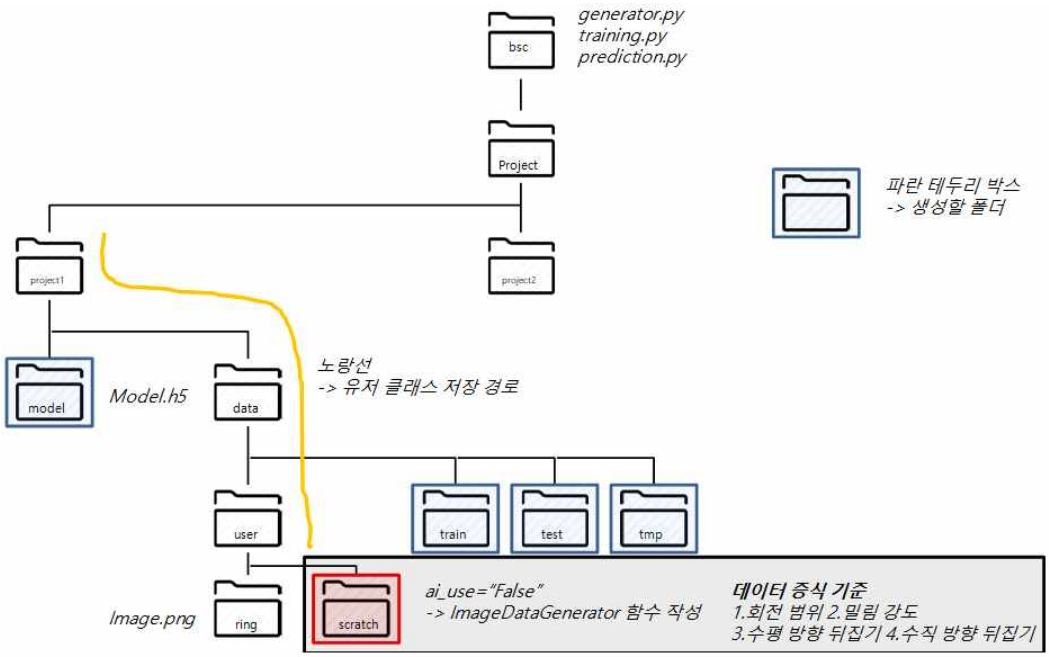

그림 1. 데이터 생성 스크립트 동작시 디렉토리 구조

데이터 생성을 위한 스크립인 'generator.py' 파일 작성을 진행하였음. 스크립트 실행시 클래스 명과 ai_use 옵션을 받아 스크립트를 실행할 수 있도록 작성함. (ex. {["name": "scratch", "ai_use": False}]

ai를 사용하지 않는 경우 케라스의 ImageDataGenetor를 이용하여 데이터 증식을 진행함. 이때 데이터 증식 옵션의 인자로 회전 범위, 밀림 강도, 수평 방향 뒤집기, 수직 방향 두집기 변수를 고정하여 데이터 증식을 수행함.

연구 노트

연구 목표

로컬에 GPU 환경을 구성하고, 데이터 생성 스크립트 성능평가를 진행한다.

개념

1. GPU 환경 구성

로컬 환경에서 GPU 환경을 구성하기 위해 CUDA Toolkit 및 cuDNN 설치를 진행하였음. 설치할 때 사용한 버전은 아래와 같음.

-CUDA Toolkit 9.0(Sept 2017)

-cuDNN v.7.4.1 (Nov 8, 2018) for CUDA 9.0

2. 데이터 생성 성능평가

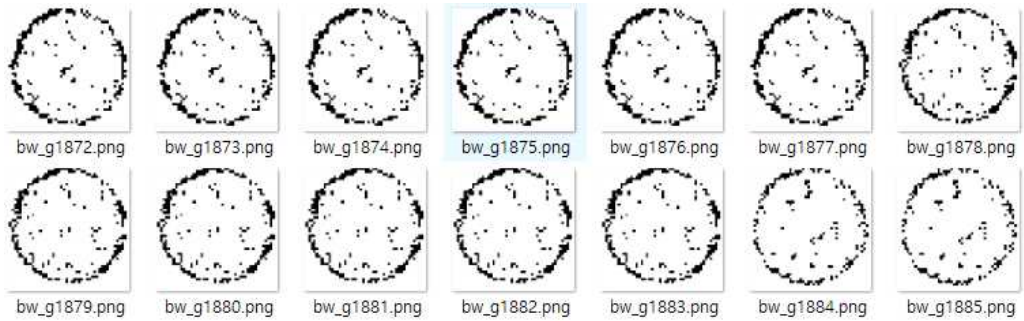

그림 1. ‘ai_use=True‘인 경우의 생성 데이터 예시

데이터 생성 스크립트의 동작 과정은 이미지 크기 다운사이징, 데이터 증식, 트레이닝, 이진화, 이미지 크기 업사이징의 순서로 진행됨. 그림 1은 웨이퍼 결함 패턴이 ‘edge ring'인 클래스에 대하여 데이터 생성 알고리즘을 사용하여 생성한 데이터는 나타냄.

-성능 측정

	시간	메모리
2000 장 / 1 class	4m 20s	1.5 G

* 벤치마크에 사용한 GPU는 GTX 1060 기준임
* 생성된 데이터 사이즈는 ai_use=True의 경우 1MB, ai_use=False의 경우 5MB