WALL-E

- 이미지 분석을 이용한 마케팅 서비스 프로젝트 -

에이콘 아카데미 2조 파이널 프로젝트 (체리블러썸조)

차례

01. 임무분담 및 프로젝트 소개

임무 분담

알고리즘 연구	김호*, 오해*, 이승*, 정동*
학습 model 연구	김호*, 이준*
데이터 탐색	최정*
데이터 가공 및 분류	이준*, 정동*, 최정*
model 리서치	이민*
model 생성	오해*, 이승*
model 및 코드 리뷰	팀원 전원
데이터 시각화	정동*
모델 방향성 제시	이준*

프로젝트 소개 - 기획의도

'ESG' 란?

사회책임투자란 사회적·윤리적 가치를 반영하는 기업에 투자하는 방식, 장기적 관점에서 기업 가치와 지속가능성에 영향을 주는 ESG(환경·사회·지배구조) 등의 비재무적 요소를 충분히 반영해 평가한다.

환경(Environmental)을 선택한 이유?

최근 전기차, 수소차, 친환경, 재활용 등의 자연환경을 오염하지 않고 이전의 환경으로 돌아가기 위한 가장 친숙한 방식으로 접근해보자는 마음으로 프로젝트를 시도하게 되었습니다.

기대효과!

- 외국인들에게도 거부감 없이 쉽게 접근 !
- 어린이들의 친숙하고 즐거운 놀이 방식으로 접근 !

단순 어플 접근한다면 친환경에 한걸음 다가설 수 있을것이라 예측됩니다.

프로젝트 소개 – 작업과정 및 사용 모듈,언어

01
프로젝트 기획
- 프로젝트 방향성
- 구조 계획

02
아이디어 회의
- 토론 및 주제선정

03
자료수집
공용데이터를 활용한
이미지 수집

04
이미지 변환
라벨링 되어있는 이미지를
프로젝트 환경에 맞춰 변환

05
모델생성
모델 학습
데이터 시각화

Keras

06
최종 프로젝트 구현

02. 이미지 변환 (Image Crop)

이미지 변환 (Image Crop)

초기 이미지

변환된 이미지

- 초기 이미지의 크기가 1920 x 1080 여서 모델 학습에 시간이 크게 소비될것으로 예측
- 모델학습시간 단축 및 정확도를 위해 28,203장의 **이미지**의 배경을 **짜르는(crop)과정 진행**
- 초기에 **학습(Train)**, **검증(Validation)**, **학습과검증이 완료된(Test)** 으로 나누어 폴더에 저장하고 모델링 작업 시작

-이미지 출처 : aihub.or.kr

03. 이미지 분류 모델
생성, 학습, 시각화

데이터 전처리

```python
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow as tf

from tensorflow.keras.preprocessing import image_dataset_from_directory

from google.colab import drive
```

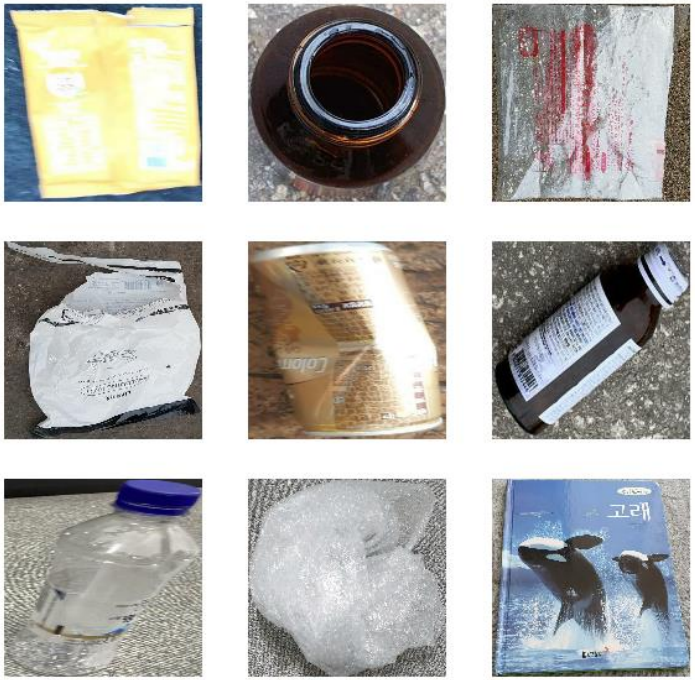
이미지 경로 불러오기

```python
PATH = '/content/drive/MyDrive/gabi'
train_dir = os.path.join(PATH,'train')
validation_dir = os.path.join(PATH, 'test')

LABELS = ['can', 'glass', 'paper', 'pet', 'plastic', 'vinyl']
BATCH_SIZE = 32 # 몇 개의 샘플로 가중치를 갱신할 것인지 설정
EPOCHS = 30
IMG_SIZE = (160, 160)

# train dataset 설정 - 학습
train_dataset = image_dataset_from_directory(train_dir,
                                             shuffle=True,
                                             batch_size=BATCH_SIZE,
                                             image_size=IMG_SIZE,
                                             label_mode='categorical')
# validation dataset 설정 - 검증
validation_dataset = image_dataset_from_directory(validation_dir,
                                                  shuffle=True,
                                                  batch_size=BATCH_SIZE,
                                                  image_size=IMG_SIZE,
                                                  label_mode='categorical')
```

학습횟수 30회 설정

SampleData

```python
class_names = train_dataset.class_names

plt.figure(figsize=(10, 10))
for images, labels in train_dataset.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.axis("off")
```



데이터 전처리

```
val_batches = tf.data.experimental.cardinality(validation_dataset)
test_dataset = validation_dataset.take(val_batches // 5)
validation_dataset = validation_dataset.skip(val_batches // 5)
```

원본 DataSet에는 Test Set이 포함되어 있지 않으므로 Test Set를 생성
tf.data.experimental.cardinality를 사용하여 Validation Set에서 사용할 수 있는 데이터 배치 수를 확인한 다음, 그 중 20%를 Test Set으로 이동

```
print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation_dataset))
print('Number of test batches: %d' % tf.data.experimental.cardinality(test_dataset))
```

```
Number of validation batches: 142
Number of test batches: 35
```

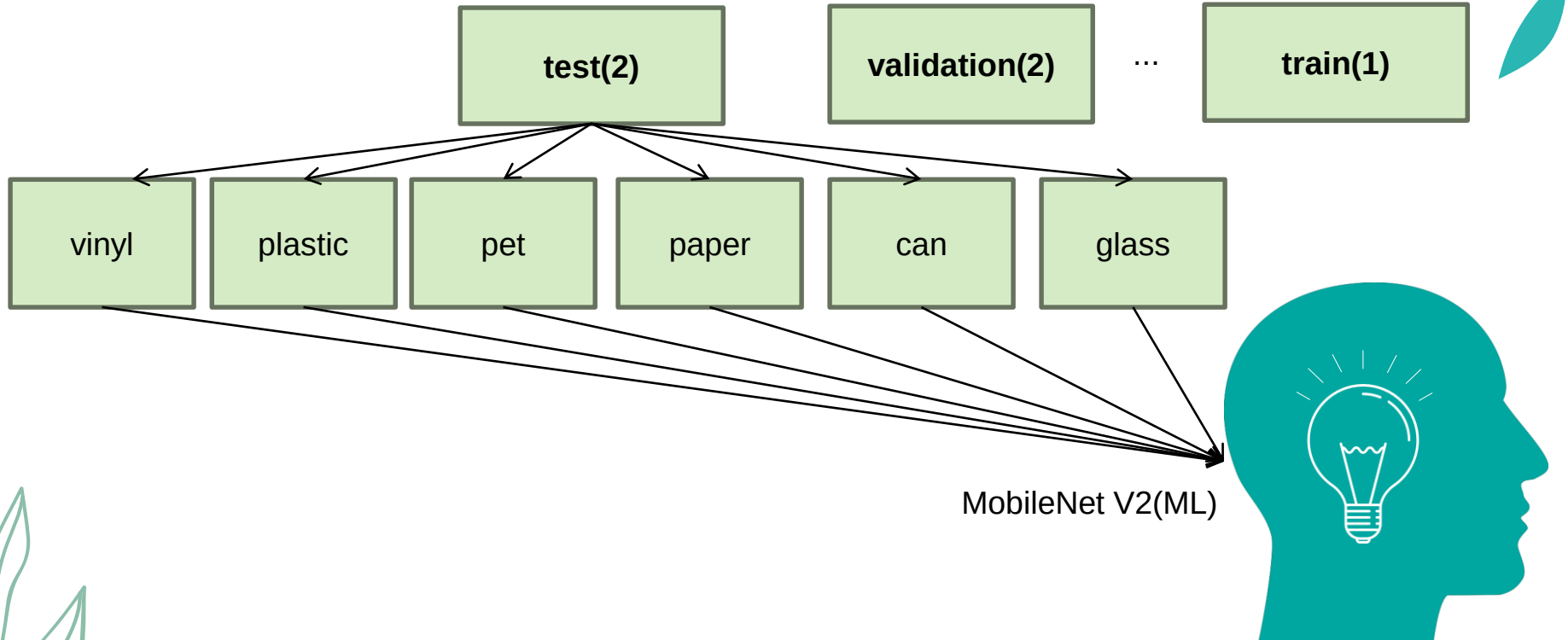
성능을 높이도록 Dataset 구성하기
버퍼링된 prefetch를 사용하여
I/O 차단 없이 디스크에서 이미지를 로드

```
AUTOTUNE = tf.data.experimental.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input

rescale = tf.keras.layers.experimental.preprocessing.Rescaling(1./127.5, offset= -1)
```

픽셀 값 재조정
기본 모델로 사용할 tf.keras.applications.MobileNetV2를 다운
이 모델은 [-1, 1]의 픽셀 값을 예상하지만 이 시점에서 이미지의 픽셀 값은 [0, 255]이다.

모델 설명_MobileNetV2 (config/Acorn_2TProject.ipyb)



Model생성 (MobileNet V2)

```
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

Google에서 개발한 MobileNet V2 모델로부터 기본 모델을 생성

```
1 image_batch, label_batch = next(iter(train_dataset))
2 feature_batch = base_model(image_batch)
3 print(feature_batch.shape)
```

```
(32, 5, 5, 1280)
```

이 특징 추출기는 각 160x160x3 이미지를 5x5x1280 개의 특징 블록으로 변환
이미지 배치 예제에서 수행하는 작업을 확인

```
base_model.trainable = False
```

base model 내에 **학습해야 할 대상**
즉, 가중치를 '**학습 가능 상태**(업데이트 가능 상태)'로 설정
만약 True로 설정한다면, 가중치를 freeze 한 상태로
학습이 진행되지 않게끔 설정해 주는 것이라고 할 수 있다.

Model (MobileNet V2)

```
1 global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
2 feature_batch_average = global_average_layer(feature_batch)
3 print(feature_batch_average.shape)
```

```
(32, 1280)
```

분류 층을 맨 위에 추가하기
특성 블록에서 예측을 생성하기 위해
tf.keras.layers.GlobalAveragePooling2D 레이어를 사용하여
특성을 이미지당 하나의 1280-요소 벡터로 변환하여 5x5
공간 위치에 대한 평균을 구한다.

```
1 prediction_layer = tf.keras.layers.Dense(6, activation='softmax')
2 prediction_batch = prediction_layer(feature_batch_average)
3 print(prediction_batch.shape)
```

```
(32, 6)
```

tf.keras.layers.Dense 레이어를 사용하여
특성을 이미지당 단일 예측으로 변환
이 예측은 logit또는 원시 예측 값으로 취급되므로 활성화
함수가 필요하지 않는다.
양수는 클래스 1을 예측하고 음수는 클래스 0을 예측한다.

```
1 inputs = tf.keras.Input(shape=(160, 160, 3))
2 x = preprocess_input(inputs)
3 x = base_model(x, training=False)
4 x = global_average_layer(x)
5 x = tf.keras.layers.Dropout(0.2)(x)
6 outputs = prediction_layer(x)
7 model = tf.keras.Model(inputs, outputs)
```

모델의 입력은 160x160 크기의 3채널(RGB) 이미지.
그리고 base model에 입력을 넣어 feature을 extraction 한다.
마지막 결과 feature map에 대해 global average pooling 을 진행하고,
최종적으로 prediction layer를 거쳐 출력(ouput)을 정의

Model (MobileNet V2)

```
base_learning_rate = 0.0005
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate),
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])
```

- 학습률 = 0.0005로 부여

학습방식에 대한 환경 설정 하기

- 정규화 = Adam
- 손실값 = 범주형 교차 엔트로피
- 평가지표(척도) = 정확도

```
model.summary()
```

모델 구조 확인하기

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 160, 160, 3)]     0

tf.math.truediv (TFOpLambda  (None, 160, 160, 3)       0
)

tf.math.subtract (TFOpLambd  (None, 160, 160, 3)       0
a)

mobilenetv2_1.00_160 (Funct  (None, 5, 5, 1280)        2257984
ional)

global_average_pooling2d (G  (None, 1280)              0
lobalAveragePooling2D)

dropout (Dropout)            (None, 1280)              0

dense (Dense)                (None, 6)                 7686

=================================================================
Total params: 2,265,670
Trainable params: 7,686
Non-trainable params: 2,257,984
_________________________________________________________________
```

Model 학습

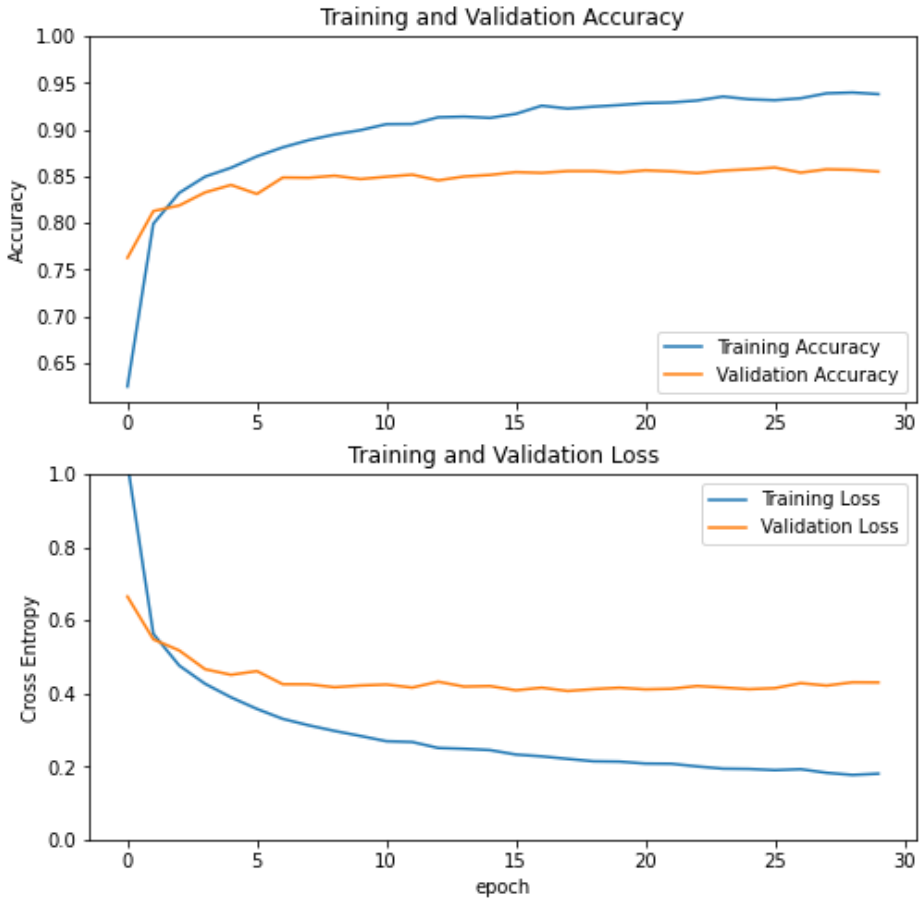
```
history = model.fit(train_dataset,
                    epochs=EPOCHS,
                    validation_data=validation_dataset)
```

```
Epoch 1/30
182/182 [==============================] - 589s 3s/step - loss: 1.0297 - accuracy: 0.6252 - val_loss: 0.6652 - val_accuracy: 0.7628
Epoch 2/30
182/182 [==============================] - 35s 187ms/step - loss: 0.5631 - accuracy: 0.7992 - val_loss: 0.5481 - val_accuracy: 0.8127
Epoch 3/30
182/182 [==============================] - 35s 188ms/step - loss: 0.4763 - accuracy: 0.8322 - val_loss: 0.5176 - val_accuracy: 0.8187
Epoch 4/30
182/182 [==============================] - 35s 187ms/step - loss: 0.4264 - accuracy: 0.8496 - val_loss: 0.4663 - val_accuracy: 0.8327
Epoch 5/30
182/182 [==============================] - 35s 187ms/step - loss: 0.3893 - accuracy: 0.8590 - val_loss: 0.4510 - val_accuracy: 0.8406
Epoch 6/30
182/182 [==============================] - 35s 190ms/step - loss: 0.3576 - accuracy: 0.8712 - val_loss: 0.4611 - val_accuracy: 0.8310
Epoch 7/30
182/182 [==============================] - 37s 199ms/step - loss: 0.3303 - accuracy: 0.8808 - val_loss: 0.4252 - val_accuracy: 0.8486
Epoch 8/30
182/182 [==============================] - 35s 188ms/step - loss: 0.3127 - accuracy: 0.8886 - val_loss: 0.4250 - val_accuracy: 0.8483
Epoch 9/30
182/182 [==============================] - 35s 190ms/step - loss: 0.2973 - accuracy: 0.8946 - val_loss: 0.4170 - val_accuracy: 0.8505
Epoch 10/30
182/182 [==============================] - 41s 222ms/step - loss: 0.2836 - accuracy: 0.8992 - val_loss: 0.4218 - val_accuracy: 0.8470
Epoch 11/30
182/182 [==============================] - 35s 189ms/step - loss: 0.2692 - accuracy: 0.9056 - val_loss: 0.4243 - val_accuracy: 0.8494
Epoch 12/30
182/182 [==============================] - 35s 191ms/step - loss: 0.2670 - accuracy: 0.9058 - val_loss: 0.4161 - val_accuracy: 0.8516
Epoch 13/30
182/182 [==============================] - 35s 191ms/step - loss: 0.2507 - accuracy: 0.9130 - val_loss: 0.4320 - val_accuracy: 0.8456
Epoch 14/30
182/182 [==============================] - 35s 191ms/step - loss: 0.2485 - accuracy: 0.9137 - val_loss: 0.4188 - val_accuracy: 0.8497
Epoch 15/30
182/182 [==============================] - 37s 199ms/step - loss: 0.2449 - accuracy: 0.9123 - val_loss: 0.4202 - val_accuracy: 0.8513
Epoch 16/30
182/182 [==============================] - 41s 220ms/step - loss: 0.2327 - accuracy: 0.9166 - val_loss: 0.4087 - val_accuracy: 0.8544
Epoch 17/30
182/182 [==============================] - 36s 192ms/step - loss: 0.2277 - accuracy: 0.9254 - val_loss: 0.4158 - val_accuracy: 0.8535
Epoch 18/30
182/182 [==============================] - 35s 187ms/step - loss: 0.2208 - accuracy: 0.9223 - val_loss: 0.4067 - val_accuracy: 0.8555
Epoch 19/30
182/182 [==============================] - 31s 166ms/step - loss: 0.2143 - accuracy: 0.9243 - val_loss: 0.4119 - val_accuracy: 0.8555
Epoch 20/30
182/182 [==============================] - 31s 166ms/step - loss: 0.2134 - accuracy: 0.9261 - val_loss: 0.4156 - val_accuracy: 0.8538
Epoch 21/30
182/182 [==============================] - 31s 165ms/step - loss: 0.2081 - accuracy: 0.9281 - val_loss: 0.4112 - val_accuracy: 0.8563
Epoch 22/30
182/182 [==============================] - 31s 166ms/step - loss: 0.2075 - accuracy: 0.9288 - val_loss: 0.4127 - val_accuracy: 0.8552
Epoch 23/30
182/182 [==============================] - 31s 165ms/step - loss: 0.2002 - accuracy: 0.9309 - val_loss: 0.4203 - val_accuracy: 0.8533
Epoch 24/30
182/182 [==============================] - 32s 173ms/step - loss: 0.1940 - accuracy: 0.9352 - val_loss: 0.4161 - val_accuracy: 0.8560
Epoch 25/30
182/182 [==============================] - 31s 166ms/step - loss: 0.1932 - accuracy: 0.9323 - val_loss: 0.4117 - val_accuracy: 0.8574
Epoch 26/30
182/182 [==============================] - 31s 165ms/step - loss: 0.1900 - accuracy: 0.9312 - val_loss: 0.4144 - val_accuracy: 0.8593
Epoch 27/30
182/182 [==============================] - 31s 166ms/step - loss: 0.1926 - accuracy: 0.9333 - val_loss: 0.4282 - val_accuracy: 0.8538
Epoch 28/30
182/182 [==============================] - 31s 165ms/step - loss: 0.1827 - accuracy: 0.9386 - val_loss: 0.4214 - val_accuracy: 0.8574
Epoch 29/30
182/182 [==============================] - 31s 166ms/step - loss: 0.1769 - accuracy: 0.9395 - val_loss: 0.4305 - val_accuracy: 0.8568
Epoch 30/30
182/182 [==============================] - 31s 165ms/step - loss: 0.1803 - accuracy: 0.9378 - val_loss: 0.4301 - val_accuracy: 0.8549
```

총 **30회**의 **학습**을 시켰으며
-훈련 손실값(Loss) : 0.1803
-훈련 정확도(Accuracy) : 0.9378
-검증 손실값(Val_loss) : 0.4301
-검증 정확도(Val_accuracy) : 0.8549
값이 나온것을 확인할수가있다.

Model 시각화

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



Model 평가 및 예측

```
[ ]  1 loss_final, accuracy_final = model.evaluate(test_dataset)
```

```
28/28 [==============================] - 3s 98ms/step - loss: 0.4813 - accuracy: 0.8504
```

```
1 print("Final loss: {:.2f}".format(loss_final))
2 print("Final accuracy: {:.2f}".format(accuracy_final))
```

```
Final loss: 0.48
Final accuracy: 0.85
```

마지막으로 테스트 세트를 사용하여 새 데이터에 대한 모델의 성능을 확인할 수 있습니다.

04. 프로젝트 구현

프로젝트 구현 (Visual Studio)

```
<title>WALL-E</title>
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<link rel="stylesheet" href="static/button.css">

<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap" rel="stylesheet">
<style>
  body{
    height: 88vh;
    background: url(static/댕댕이.jpg);
    background-repeat : no-repeat;
    background-size : cover;
  }
  #preview{
    width:40%;
    height:40%;
    object-fit: contain;
  }
  #mainDiv{
    width:50%;
    height:90%;
    margin:auto;
    margin-top:100px;
    background-color: rgba( 255, 255, 255, 0.5 );
    text-align: center;
    z-index:0;
  }
  @font-face {
    font-family: 'Arita-dotum-Medium';
    src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_one@1.0/Arita-dotum-Medium.woff') format('woff');
    font-weight: normal;
    font-style: normal;
  }
  p {
    font-family:'Arita-dotum-Medium';
    font-size:25px;
    font-weight: bold;
  }
  #title{
    font-family:'Arita-dotum-Medium';
    padding-top:4vh;
    font-weight: 1000;
    color: #025402;
    font-size: 52px;
    margin-bottom:6vh;
  }
</style>
</head>
```

main.html의 css 부분

- background image
- 버튼에 대한 CSS 적용
- 동적 웹 사이트 적용

프로젝트 구현 (Visual Studio)

```
<body>
  <div id="mainDiv">
    <p id="title">어디에 버려야할까요?</p>
    <button id="selectImg" class="btn-two green small">select picture</button>
    <br>
    <img id="preview" style="display:none">
    <br>
    <button id="submit_btn" class="btn-two green small" style="display:none">go</button>
    <br>
    <div>
      <p id="result"></p>
    </div>
    <form id="imgForm" enctype="multipart/form-data" method="post" style="display: none">
      {% csrf_token %}
      <input type="file" accept=".jpg, .jpeg, .png, .JPG, .JPEG, .gif" name="img" id="img" onchange="readURL(this);">
    </form>
```

main.html의 body 부분

- select 버튼을 통한 이미지 형상화
- go 버튼을 통한 쓰레기의 종류 및 직관적인 퍼센트 분류

프로젝트 구현 (Visual Studio)

```
<script>
  $.ajaxSetup({
    headers: { "X-CSRFToken": '{{csrf_token}}' }
  });
  $("#submit_btn").click(function (e) {
    e.preventDefault();
    var form = $("#imgForm")[0];
    var imgData = new FormData(form);
    imgData.append("img", $("#imgForm")[0][0]);
    $.ajax({
      url: "/classify/",
      type: "POST",
      contentType: false,
      processData: false,
      enctype: 'multipart/form-data',
      data: imgData,
      success: function (data) {
        $("#result").text("이 쓰레기는 "+data.percentage+"%의 확률로 "+data.label+" 쓰레기입니다")
      }
    })
  })
```

main.html의 script 부분

- AJAX을 활용한 페이지 전환 없이 사진 첨부 기능

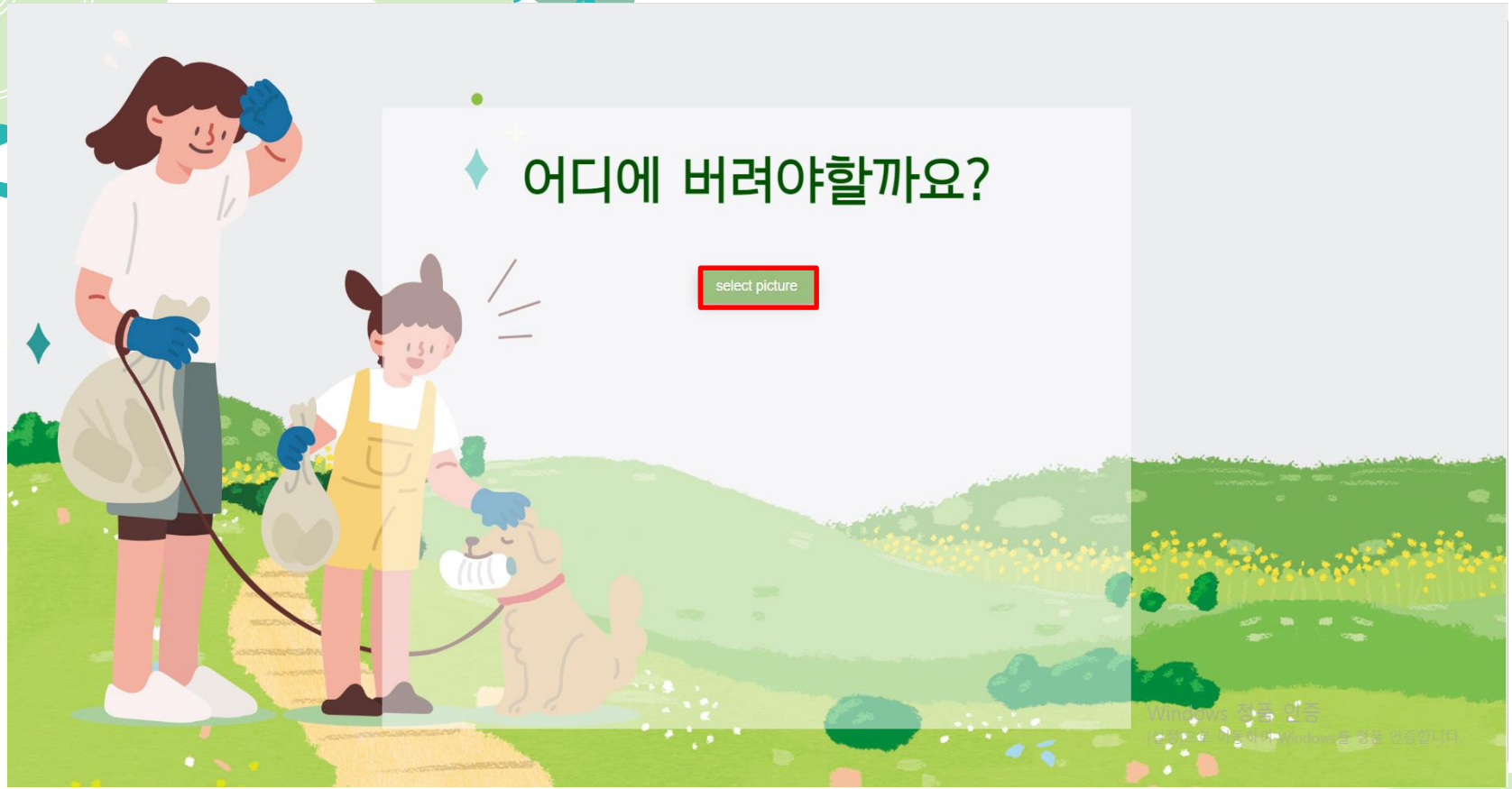
프로젝트 구현 (Visual Studio)

```
  function readURL(input) {
    if (input.files && input.files[0]) {
      var reader = new FileReader();
      reader.onload = function (e) {
        document.getElementById('preview').src = e.target.result;
      };
      reader.readAsDataURL(input.files[0]);
    } else {
      document.getElementById('preview').src = "";
    }
    $("#preview").removeAttr("style");
    $("#submit_btn").removeAttr("style");
  }
  $("#selectImg").click(function(e){
    $("#img").click();
  })
</script>
</body>

</html>
```

main.html의 script 부분

- AJAX을 활용한 페이지 전환 없이 사진 첨부 기능

05. 프로젝트 실행


어디에 버려야할까요?
select picture

어디에 버려야할까요?

select picture

go

어디에 버려야할까요?

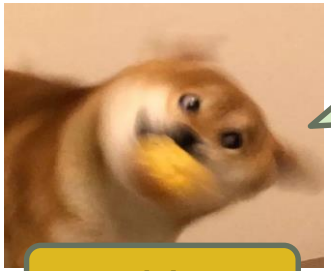
select picture

go

이 쓰레기는 88.0%의 확률로 paper 쓰레기입니다

06. 프로젝트 후기

프로젝트 후기



김호*

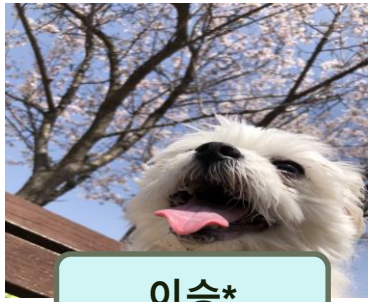
프로젝트를 시작하고 부족한 팀장으로써 다사다난했지만 믿고 따라와준 팀원들에게 고맙고 다들 프로젝트를 통해 성장해나가는 모습에서 기쁨을 느꼈습니다.
모델 부분에 있어서 많이 무언갈 해보진 못했지만,
전반적인 과정들을 통해 성장해나갈 수 있어서 좋았습니다!

Final Project를 마무리할 시간이 왔다는게 믿어지지 않을 정도로 6개월의 시간이 금방 지나간거 같습니다. 개인적으로 훌륭한 팀장 및 조원들을 만나 제가 부족했던 부분을 채울수 있어서 처음보다는 많이 성장할 수 있는 계기가 되었던거 같습니다.
저희 조원 뿐만 아니라 모든 분들이 원하시는 곳에
취업이 잘 되셔서 커리어를 쌓아가셨으면 좋겠습니다.

오해*

이민*

프로젝트에 능숙하고 좋은 팀원 분들과 함께 한 덕에 이것저것 배우며 좋은 자극을 많이 받을 수 있었습니다. 유능한 분들과 했기 때문에 개인적으로 프로젝트에 크게 기여할 기회는 적었던 것 같아 조금의 아쉬움은 있지만, 이번에 많이 배워가는 만큼 이제 앞으로는 반대로 더 나눠주는 사람이 될 수 있을 거라 믿습니다. 저희 팀원 모두 이대로 꽃길만 걷고 성공하는 모습 보면 좋겠습니다 :)

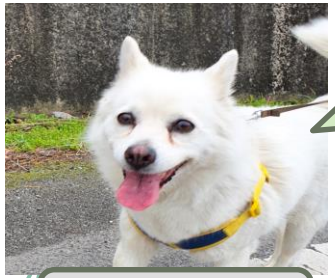


이승*

막연하게 수업으로 배우고 넘어갔었던 부분들도 프로젝트 기간 동안 다시 찾아보고 프로젝트 하면서 새로이 이해되었던 부분들 모두 즐거운 경험이었습니다. 모델을 만드는 과정에서 막히거나 오류가 발생하는 부분들의 경우 검색하고 이해하고 원활하게 돌아갈 수 있도록 만들 수 있어서 보람찬 시간들이었습니다. 함께한 모든 분들이 원하는 곳에서 지금 꿈꾸시거나 생각하는 일을 잘 이뤄가셨으면 좋겠습니다!

본 학원을 통해 IT에 첫 발걸음을 디딘게 되었는데, 좋은 사람들과 함께 공부하면서 많은 지식들을 쌓을 수 있었음에 감사의 말씀을 전합니다. 협업 프로젝트를 통해서 진로의 방향성을 정립할 수 있었고, IT 지식을 통해 세상을 더 크게 바라보는 관점을 터득하는 시간이었기에 큰 보람을 느낍니다.

이준*



정동*

다들 고생많으셨고 연이 있다면 언젠가 다시 만날 수 있길 바랍니다~

6개월간 고생많으셨습니다. 좋은 팀원분들을 만나 많이 배웠습니다. 늘 건강하시고 원하시는 일 다 이루시길 바라겠습니다.

최정*

06. 질의 및 응답

감사합니다!