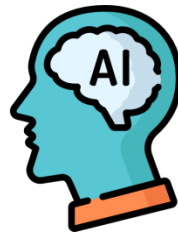


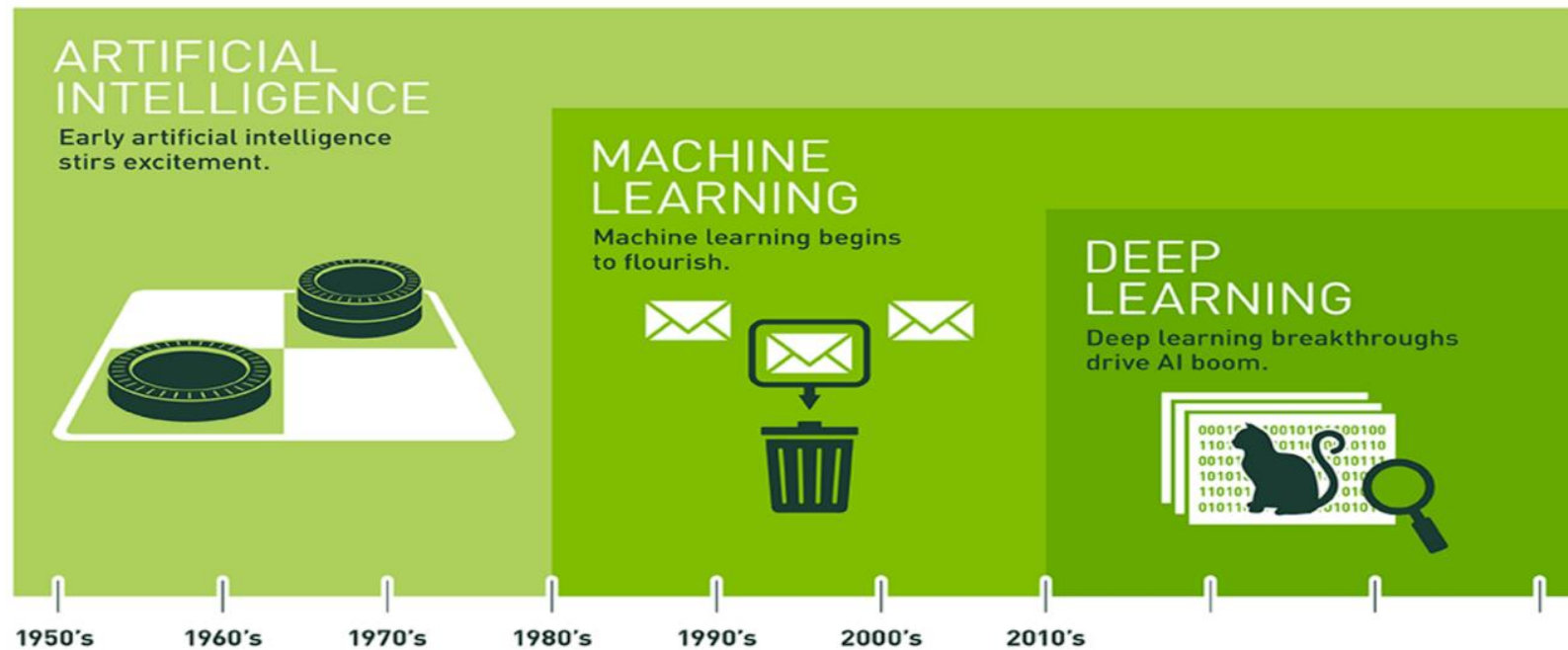
인공지능



인공지능

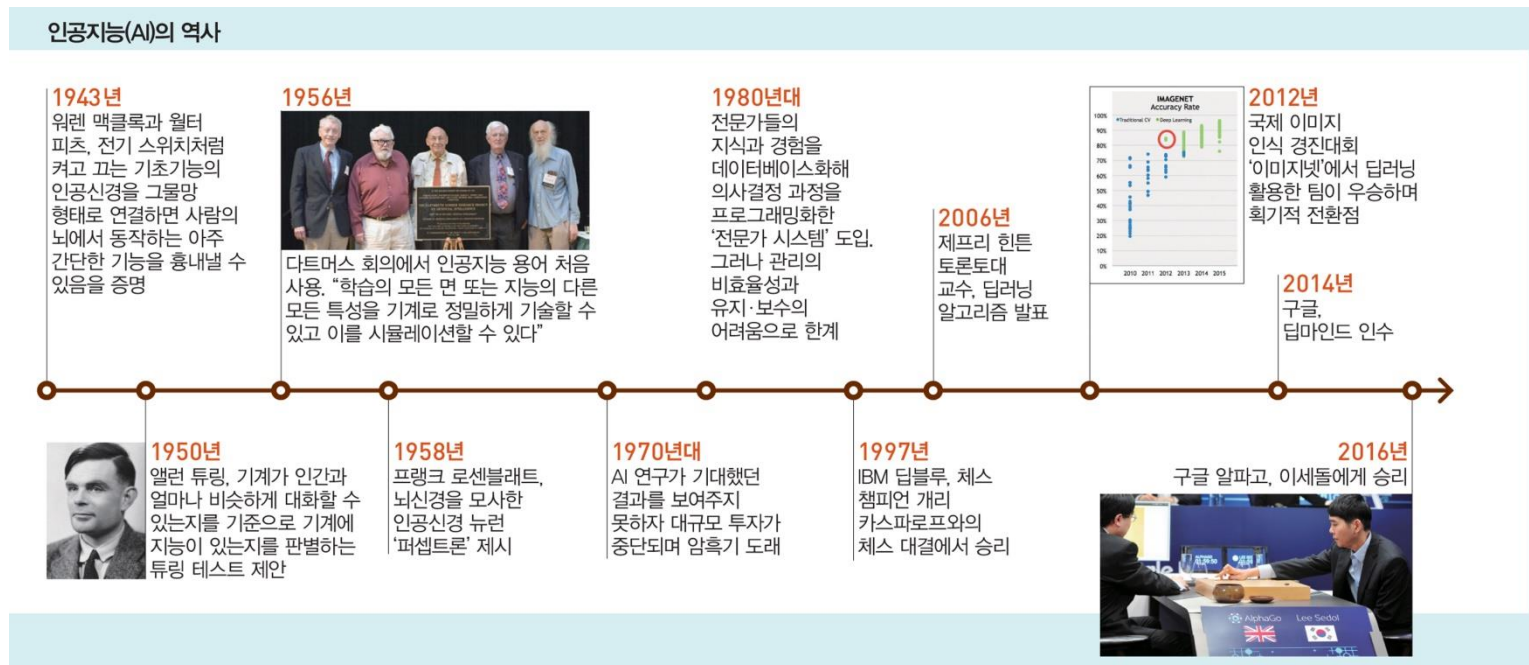
• 인공지능(AI)

- 인간의 학습능력과 추론능력, 지각능력, 자연언어의 이해능력 등을 컴퓨터 프로그램으로 실현한 기술



인공지능

• 인공지능의 역사



- 1950년대 : 인공지능의 시작
- 1970년대 : AI의 겨울
- 1980년대 : 전문가 시스템

- 1990년대 : 자연의 모방
- 2010년대 : 이질적인 테크닉들을 공통의 수학적 프레임으로 설명가능

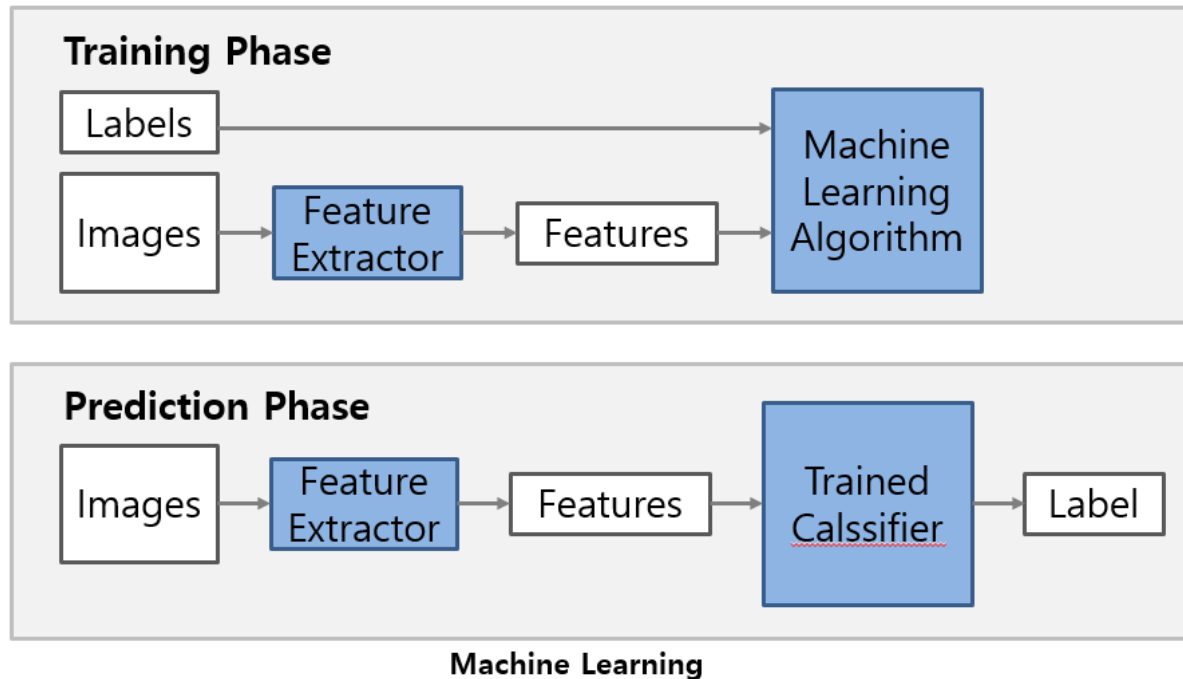
머신러닝

- 머신러닝(Machine Learning)

- 인공지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야

- 머신러닝의 핵심

- 표현(representation) : 데이터의 평가
- 일반화(generalization) : 아직 알 수 없는 데이터에 대한 처리



머신러닝 분류

• 학습 종류에 따른 분류

A. 지도 학습(Supervised Learning)

- 사람이 교사처럼 각각의 입력(x)에 대해 *레이블(y)을 달아놓은 데이터를 컴퓨터에 주면 컴퓨터가 그것을 학습하는 것
- 정확도가 높은 데이터를 사용할 수 있음
- ❖ 분류(Classification) : 레이블 y가 이산적(Discrete)인 경우 즉, y가 가질 수 있는 값이 유한한 경우 분류 혹은 인식 문제라고 함
- ❖ 회귀(Regression) : 레이블 y가 실수 인 경우

B. 비지도 학습(Unsupervised Learning)

- 사람 없이 컴퓨터가 스스로 레이블 되어 있지 않은 데이터에 대해 학습하는 것
- 정확도가 높은 데이터를 사용할 수 있음
- ❖ 군집화(Clustering) : x만 사용하여 데이터 간 거리에 따라 군집을 학습하는 것
- ❖ 분포 추정 : 군집화에서 더 나아가서, 데이터들이 쪽 뿌려져 있을 때 애네들이 어떤 확률 분포에서 나온 샘플들인지 추정하는 것

*레이블(label) : 학습 데이터의 속성을 무엇을 분석할 지에 따라 정의되는 데이터

머신러닝 종류

C. 반지도 학습(Semi-Supervised Learning)

- 레이블이 있는 데이터와 없는 데이터 모두를 활용해서 학습하는 것

D. 강화 학습(Semi-Supervised Learning)

- 현재의 State에서 어떤 Action을 취하는 것이 최적인지 학습하는 것
- 행동을 취할 때마다 외부 환경에서 Reward가 주어지며, 이러한 보상을 최대화하는 방향으로 학습 진행




















딥러닝

- 딥러닝(Deep Learning)
 - 컴퓨터들이 인간의 두뇌와 비슷한 모양의 대형 인공 신경망을 형성하는 일종의 기계 학습
- 딥러닝 VS 머신러닝 차이

머신러닝	딥러닝
알고리즘에 입력하기 전 필요한 특징을 사람이 직접 선정 Ex) 사람의 나이 추정 시 얼굴의 주름 개수, 피부 색상의 균일도 등의 특징이 필요	'특징'을 선정하는 부분까지 한꺼번에 학습 Ex) 사람 얼굴을 찍은 사진을 입력으로 넣어주면, 스스로 특징을 선정

Tensorflow lite (실습)

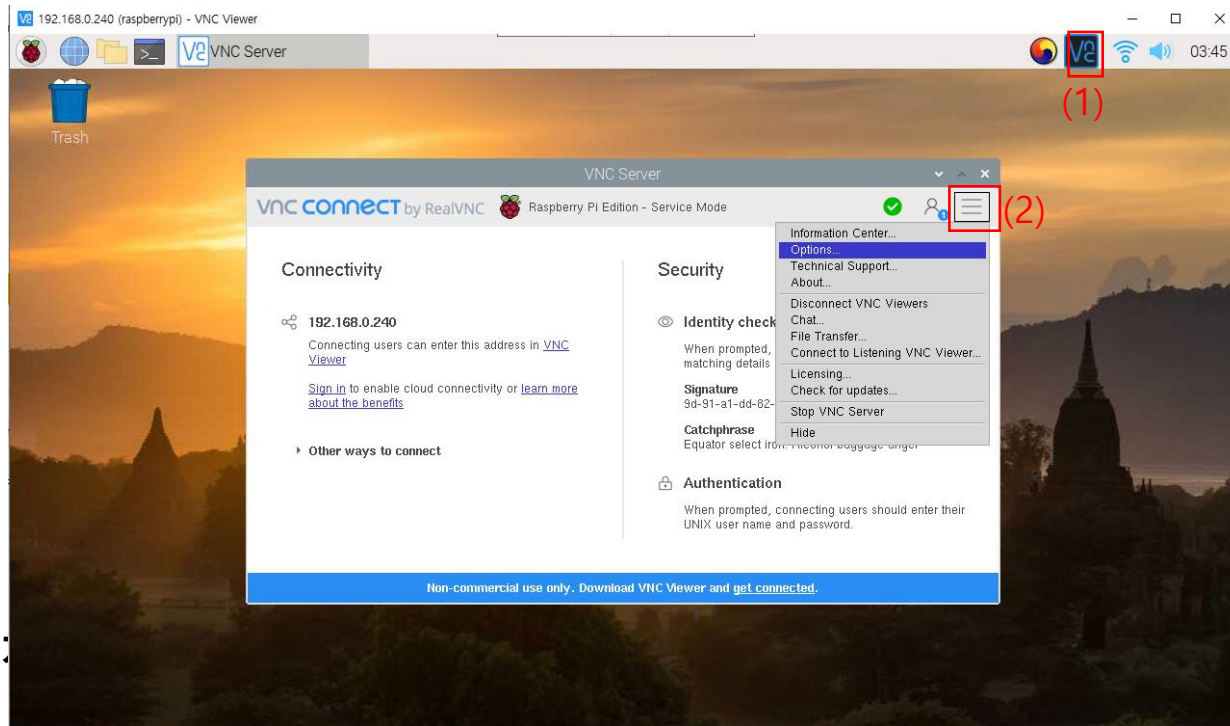
- Tensorflow lite
 - 기기 내 추론을 위한 오픈소스 딥 러닝 프레임워크
- <https://www.tensorflow.org/lite/examples?hl=ko>

 <p>이미지 분류</p> <p>모바일 카메라의 입력 프레임에서 1,000개의 서로 다른 항목 유형을 인식할 수 있는 선행 학습된 모델로 이미지 분류 솔루션을 테스트하세요.</p> <p>Android에서 시도해 보기 </p> <p>iOS에서 시도해 보기 </p> <p>Raspberry Pi에서 시도해 보기 </p>	 <p>객체 감지</p> <p>모바일 카메라의 입력 프레임에서 1,000개의 서로 다른 인식 가능한 객체 주위에 경계 상자를 그린 다음 라벨을 지정하는 선행 학습된 모델이 적용된 앱을 확인해 보세요.</p> <p>Android에서 시도해 보기 </p> <p>iOS에서 시도해 보기 </p> <p>Raspberry Pi에서 시도해 보기 </p>	 <p>자세 추정</p> <p>이미지에 있는 인물의 자세를 추정하는 앱을 살펴보세요.</p> <p>Android에서 시도해 보기 </p> <p>iOS에서 시도해 보기 </p>
 <p>음성 인식</p> <p>마이크를 사용하여 키워드를 발견하고 말하는 단어에 대한 확률 점수를 반환하는 앱을 살펴보세요.</p> <p>Android에서 시도해 보기 </p> <p>iOS에서 시도해 보기 </p>	 <p>동작 인식</p> <p>TensorFlow.js를 사용하여 웹캠에 포착된 동작을 인식하도록 신경망을 학습시킨 다음, TensorFlow Lite를 사용하여 기기에서 추론을 실행하도록 모델을 변환하세요.</p> <p>Android에서 시도해 보기 </p> <p>iOS에서 시도해 보기 </p>	 <p>스마트 답장</p> <p>대화식 채팅 메시지를 입력하기 위한 답장 추천을 생성하세요.</p> <p>Android에서 시도해 보기 </p>

Tensorflow lite (실습)

• 사전 준비

- 원격 접속 프로그램 사용 시 카메라 화면이 안 뜨는 문제점이 있음
- (1) → (2) → [Options] → [Troubleshooting] → “Enable direct capture mode” 체크 후 OK
- 해상도는 낮아지나 VNC로 카메라를 켤 수 있음



- 모니터 및 터:

Tensorflow lite (실습)

• 실습 1. 이미지 분류

- TensorFlow Lite 런타임 설치

```
pi@raspberrypi:~ $ pip3 install https://dl.google.com/coral/python/tflite\_runtime-2.1.0.post1-cp37-cp37m-linux\_armv7l.whl
```

- 사용하고 있는 Python version 확인하여 맞는 버전으로 설치

```
pi@raspberrypi:~ $ python3 -V
```

```
pi@raspberrypi:~ $ python3 -V  
Python 3.7.3
```

Platform	Python	URL
Linux (ARM 32)	3.5	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp35-cp35m-linux_armv7l.whl
	3.6	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-linux_armv7l.whl
	3.7	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-linux_armv7l.whl

Tensorflow lite (실습)

• 실습 1. 이미지 분류

- 예제 파일 다운로드

```
pi@raspberrypi:~ $git clone https://github.com/tensorflow/examples --depth 1
```

```
pi@raspberrypi:~/New $ git clone https://github.com/tensorflow/examples --depth 1
Cloning into 'examples'...
remote: Enumerating objects: 1698, done.
remote: Counting objects: 100% (1698/1698), done.
remote: Compressing objects: 100% (1139/1139), done.
remote: Total 1698 (delta 474), reused 1294 (delta 326), pack-reused 0
Receiving objects: 100% (1698/1698), 12.26 MiB | 1.57 MiB/s, done.
Resolving deltas: 100% (474/474), done.
Checking out files: 100% (1845/1845), done.
```

```
pi@raspberrypi:~/New/examples/lite/examples/image_classification/raspberry_pi
```

```
pi@raspberrypi:~ $bash download.sh /tmp
```

```
pi@raspberrypi:~/New/examples/lite/examples/image_classification/raspberry_pi $ bash download.sh /tmp

Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: numpy in /usr/lib/python3/dist-packages (from -r requirements.txt (line 2)) (1.16.2)
Requirement already satisfied: picamera in /usr/lib/python3/dist-packages (from -r requirements.txt (line 3)) (1.13)
Requirement already satisfied: Pillow in /usr/lib/python3/dist-packages (from -r requirements.txt (line 4)) (5.4.1)
% Total      % Received % Xferd  Average Speed   Time    Time     Time  Current
             Dload  Upload   Total   Spent    Left   Speed
100 2997k  100 2997k    0     0  1583k      0  0:00:01  0:00:01 --:--:-- 1583k
```

Tensorflow lite (실습)

- 실습 1. 이미지 분류

- 예제 실행

```
pi@raspberrypi:~ $python3 classify_picamera.py \  
--model /tmp/mobilenet_v1_1.0_224_quant.tflite \  
--labels /tmp/labels_mobilenet_quant_v1_224.txt
```

```
pi@raspberrypi:~/New/examples/lite/examples/image_classification/raspberry_pi $ python3 classify_  
picamera.py \  
> --model /tmp/mobilenet_v1_1.0_224_quant.tflite \  
> --labels /tmp/labels_mobilenet_quant_v1_224.txt  
█
```

Tensorflow lite (실습)

- 실습 1. 이미지 분류
 - 예제 결과



Tensorflow lite (실습)

- **classify_picamera.py**

```
(1) from __future__ import absolute_import
    from __future__ import division
    from __future__ import print_function

    import argparse
    import io
    import time
    import numpy as np
    import picamera

    from PIL import Image
    from tf.lite_runtime.interpreter import Interpreter

(2) def load_labels(path):
    with open(path, 'r') as f:
        return {i: line.strip() for i, line in enumerate(f.readlines())}

(3) def set_input_tensor(interpreter, image):
    tensor_index = interpreter.get_input_details()[0]['index']
    input_tensor = interpreter.tensor(tensor_index)[0]
    input_tensor[:, :] = image
```

(1) 파이썬 2와 3의 버전 차이로 인해 생기는 문제를 방지하고 호환이 되도록 하기 위해 사용

(2) Label 파일을 한 줄씩 읽음

(3) Input tensor를 set
- interpreter.get_input_details() : 인덱스를 알아야 데이터를 전달 할 수 있음
- tensor : 데이터 배열의 집합

Tensorflow lite (실습)

- **classify_picamera.py**

```
def classify_image(interpreter, image, top_k=1):  
  
    """Returns a sorted array of classification results."""  
  
    (4) set_input_tensor(interpreter, image)  
    interpreter.invoke()  
    output_details = interpreter.get_output_details()[0]  
    output =  
    (5) np.squeeze(interpreter.get_tensor(output_details['index']))  
  
    # If the model is quantized (uint8 data), then dequantize the  
    results  
  
    (6) if output_details['dtype'] == np.uint8:  
        scale, zero_point = output_details['quantization']  
        output = scale * (output - zero_point)  
  
    ordered = np.argsort(-output, top_k)  
    return [(i, output[i]) for i in ordered[:top_k]]
```

(4)

- Input tensor 정보
- get_output_details() : 인덱스를 알아야 결과를 받아올 수 있음

(5)

- squeeze : 1차원 배열로 축소
- Get_tensor로 출력 데이터 읽기

(6) 모형이 정량화된 경우(uint8 데이터) 결과의 정량화를 해제

Tensorflow lite (실습)

- **classify_picamera.py**

```
def main():  
(7)  parser = argparse.ArgumentParser(  
      formatter_class=argparse.ArgumentDefaultsHelpFormatter)  
  
      parser.add_argument(  
          '--model', help='File path of .tflite file.', required=True)  
  
      parser.add_argument(  
          '--labels', help='File path of labels file.', required=True)  
  
(8)  args = parser.parse_args()  
  
(9)  labels = load_labels(args.labels)  
  
      interpreter = Interpreter(args.model)  
      interpreter.allocate_tensors()  
      _, height, width, _ = interpreter.get_input_details()[0]['shape']  
  
(10) with picamera.PiCamera(resolution=(640, 480), framerate=30)  
      as camera:  
          camera.start_preview()  
          try:
```

(7)

- ArgumentParser에 parser 객체를 생성
- add_argument()로 원하는 인자 추가

(8) parse_args()로 명령창에서 주어진 인자를 파싱

(9) Args라는 이름으로 파싱을 성공하면 args.parameter형태로 주어진 인자 값을 받아 사용할 수 있음

(10) 해상도 640 x 480 , framerate = 30인 파이 카메라 미리보기 켜기

Tensorflow lite (실습)

- **classify_picamera.py**

```
(11) stream = io.BytesIO()
    for _ in camera.capture_continuous(
        stream, format='jpeg', use_video_port=True):
        stream.seek(0)
        image = Image.open(stream).convert('RGB').resize((width,
height),
                                                    Image.ANTIALIAS)

        start_time = time.time()
        results = classify_image(interpreter, image)
(12) elapsed_ms = (time.time() - start_time) * 1000

(13) label_id, prob = results[0]
(14) stream.seek(0)
        stream.truncate()
(15) camera.annotate_text = '%s %.2f\n%.1fms' %
(labels[label_id], prob,
                                elapsed_ms)

    finally:
        camera.stop_preview()

if __name__ == '__main__':
    main()
```

(11) Capture 기능 계속 수행

(12) 분류 결과를 알아내기까지의 시간

(13) 분류 결과

(14)

- seek(n) : 파일의 n번째 바이트로 이동
- truncate([size]) : 파일 크기를 지정된 크기로 자름. 인수를 정해주지 않으면 현재 위치에서 자름

(15) 카메라 미리보기에 text 입력

Tensorflow lite (실습)

- 실습 2. 객체 감지

- 예제 파일 다운로드

```
pi@raspberrypi:~ $git clone https://github.com/tensorflow/examples --depth 1
```

```
pi@raspberrypi:~/New $ git clone https://github.com/tensorflow/examples --depth 1
Cloning into 'examples'...
remote: Enumerating objects: 1698, done.
remote: Counting objects: 100% (1698/1698), done.
remote: Compressing objects: 100% (1139/1139), done.
remote: Total 1698 (delta 474), reused 1294 (delta 326), pack-reused 0
Receiving objects: 100% (1698/1698), 12.26 MiB | 1.57 MiB/s, done.
Resolving deltas: 100% (474/474), done.
Checking out files: 100% (1845/1845), done.
```

```
pi@raspberrypi:~/New/examples/object_detection/raspberry_pi
```

```
pi@raspberrypi:~ $bash download.sh /tmp
```

```
pi@raspberrypi:~/New/examples/lite/examples/object_detection/raspberry_pi $ bash
download.sh /tmp

Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: numpy in /usr/lib/python3/dist-packages (from -r
requirements.txt (line 2)) (1.16.2)
Requirement already satisfied: picamera in /usr/lib/python3/dist-packages (from
-r requirements.txt (line 3)) (1.13)
Requirement already satisfied: Pillow in /usr/lib/python3/dist-packages (from -r
requirements.txt (line 4)) (5.4.1)
```

Tensorflow lite (실습)

- 실습 2. 객체 감지

- 예제 실행

```
pi@raspberrypi:~ $python3 detect_picamera.py \
```

```
--model /tmp/detect.tflite \
```

```
--labels /tmp/coco_labels.txt
```

```
pi@raspberrypi:~/New/examples/lite/examples/object_detection/raspberry_pi $ python3 detect_picamera.py \  
> --model /tmp/detect.tflite \  
> --labels /tmp/coco_labels.txt
```

Tensorflow lite (실습)

- 실습 2. 객체 감지
 - 예제 결과



Tensorflow lite (실습)

- **detect_picamera.py**

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import argparse
import io
import re
import time
(1) from annotation import Annotator

import numpy as np
import picamera
from PIL import Image
from tf-lite-runtime.interpreter import Interpreter
CAMERA_WIDTH = 640
CAMERA_HEIGHT = 480

def load_labels(path):
    with open(path, 'r', encoding='utf-8') as f:
        lines = f.readlines()
        labels = {}
    for row_number, content in enumerate(lines):
(2)         pair = re.split(r'[:\s]+', content.strip(), maxsplit=1)
        if len(pair) == 2 and pair[0].strip().isdigit():
            labels[int(pair[0])] = pair[1].strip()
        else:
            labels[row_number] = pair[0].strip()
    return labels
```

(1) 외부 파이썬 파일 Annotator라는 이름으로 사용

(2) re.split() : '[:\s]+'을 기준으로 나누는 함수

Tensorflow lite (실습)

- **detect_picamera.py**

(3) `def set_input_tensor(interpreter, image):`
 `tensor_index = interpreter.get_input_details()[0]['index']`
 `input_tensor = interpreter.tensor(tensor_index())[0]`
 `input_tensor[:, :] = image`

(3) Input tensor set

(4) `def get_output_tensor(interpreter, index):`
 `output_details = interpreter.get_output_details()[index]`
 `tensor =`
 `np.squeeze(interpreter.get_tensor(output_details['index']))`
 `return tensor`

(4) 지정된 인덱스에서 output tensor 반환

(5) `def detect_objects(interpreter, image, threshold):`
 `set_input_tensor(interpreter, image)`
 `interpreter.invoke()`
 `# Get all output details`
 `boxes = get_output_tensor(interpreter, 0)`
 `classes = get_output_tensor(interpreter, 1)`
 `scores = get_output_tensor(interpreter, 2)`
 `count = int(get_output_tensor(interpreter, 3))`
 `results = []` for i in range(count):
 if scores[i] >= threshold:
 result = {
 'bounding_box': boxes[i],
 'class_id': classes[i],
 'score': scores[i] }
 results.append(result)
 `return results`

(5) 각각의 물체 정보 목록 반환

Tensorflow lite (실습)

- **detect_picamera.py**

```
def annotate_objects(annotator, results, labels):  
    """Draws the bounding box and label for each object in the  
    results."""  
  
    for obj in results:  
        # Convert the bounding box figures from relative coordinates  
        # to absolute coordinates based on the original resolution  
  
        ymin, xmin, ymax, xmax = obj['bounding_box']  
        xmin = int(xmin * CAMERA_WIDTH)  
        xmax = int(xmax * CAMERA_WIDTH)  
        ymin = int(ymin * CAMERA_HEIGHT)  
        ymax = int(ymax * CAMERA_HEIGHT)  
  
        # Overlay the box, label, and score on the camera preview  
        annotator.bounding_box([xmin, ymin, xmax, ymax])  
        annotator.text([xmin, ymin],  
                        '%s\n%.2f' % (labels[obj['class_id']], obj['score']))
```

(6)

(6) 외부 파이썬 파일의 함수 사용

Tensorflow lite (실습)

- **detect_picamera.py**

```
def main():  
(7) parser = argparse.ArgumentParser(  
    formatter_class=argparse.ArgumentDefaultsHelpFormatter)  
    parser.add_argument(  
        '--model', help='File path of .tflite file.', required=True)  
    parser.add_argument(  
        '--labels', help='File path of labels file.', required=True)  
    parser.add_argument(  
        '--threshold',  
        help='Score threshold for detected objects.',  
        required=False,  
        type=float,  
        default=0.4)  
  
(8) args = parser.parse_args()  
  
(9) labels = load_labels(args.labels)  
    interpreter = Interpreter(args.model)  
  
    interpreter.allocate_tensors()  
  
    _, input_height, input_width, _ =  
    interpreter.get_input_details()[0]['shape']
```

- (7)
- ArgumentParser에 parser 객체를 생성
 - add_argument()로 원하는 인자 추가

(8) parse_args()로 명령창에서 주어진 인자를 파싱

(9) Args라는 이름으로 파싱을 성공하면 args.parameter형태로 주어진 인자 값을 받아 사용할 수 있음

Tensorflow lite (실습)

- **detect_picamera.py**

```
(10) with picamera.PiCamera(
        resolution=(CAMERA_WIDTH, CAMERA_HEIGHT),
        framerate=30) as camera:
        camera.start_preview()
    try:
        stream = io.BytesIO()
        annotator = Annotator(camera)
        for _ in camera.capture_continuous(
            stream, format='jpeg', use_video_port=True):
            stream.seek(0)
            image = Image.open(stream).convert('RGB').resize(
                (input_width, input_height), Image.ANTIALIAS)
            start_time = time.monotonic()
            results = detect_objects(interpreter, image, args.threshold)
            elapsed_ms = (time.monotonic() - start_time) * 1000

            annotator.clear()
            annotate_objects(annotator, results, labels)
            annotator.text([5, 0], '%.1fms' % (elapsed_ms))
            annotator.update()
            stream.seek(0)
            stream.truncate()
        finally:
            camera.stop_preview()

    if __name__ == '__main__':
        main()
```

(10) 해상도 640 x 480 , framerate = 30인
파이 카메라 미리보기 켜기

(11)

- 물체 감지 함수 사용
- 인지하는데 걸리는 시간 계산