

18기 정규세션

ToBig's 17기 강의자

김상윤

Framework

Contents

Unit 01 | Intro

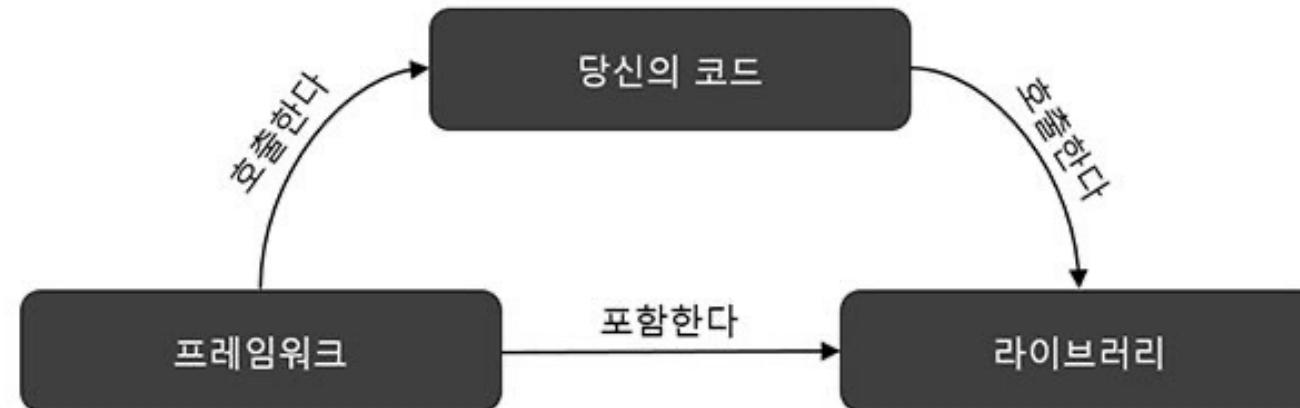
Unit 02 | 객체지향프로그래밍

Unit 03 | Framework

Unit 04 | TensorFlow, Keras, PyTorch

Unit 01 | Intro

API? Library? Framework?



Unit 01 | Intro

라이브러리

- 소프트웨어를 개발할 때, 컴퓨터 프로그램이 사용하는 비휘발성 자원의 모임
- 자주 사용될 수 있는 기능들을 구현하여 모아 놓은 것
- 프로그램이 능동적으로 라이브러리를 사용
 - : 필요할 때 호출하여 해당 기능 사용
- 모듈과 패키지로 구성
 - 모듈 : 프로그램 구성 소스코드(변수, 함수, 클래스)들을 모아놓은 것
 - 패키지 : 어떤 기능과 관련된 모듈을 모아 놓은 것



Unit 01 | Intro

import

- 모듈 가져오기
- import
- import as
- from import
- from import *

```
-MacBookAir ~ % pip3 install numpy
```

```
import numpy

arr = [1, 9, 25, 49]
arr_sqrt = numpy.sqrt(arr)
print(arr_sqrt)

[1. 3. 5. 7.]
```

```
import numpy as np

arr_sqrt = np.sqrt(arr)
print(arr_sqrt)

[1. 3. 5. 7.]
```

```
from torch import nn, optim
from torch.nn import functional as F
```

```
import progressbar
import time

bar = progressbar.ProgressBar()
for i in bar(range(100)):
    time.sleep(0.02)

100% (100 of 100) |#####| Elapsed Time:
```

```
from progressbar import *

bar = ProgressBar()
for i in bar(range(100)):
    time.sleep(0.02)
```

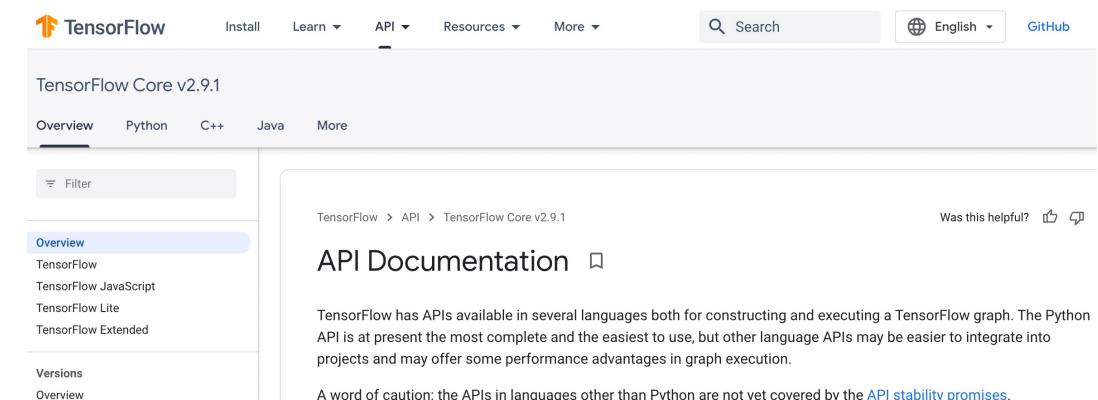
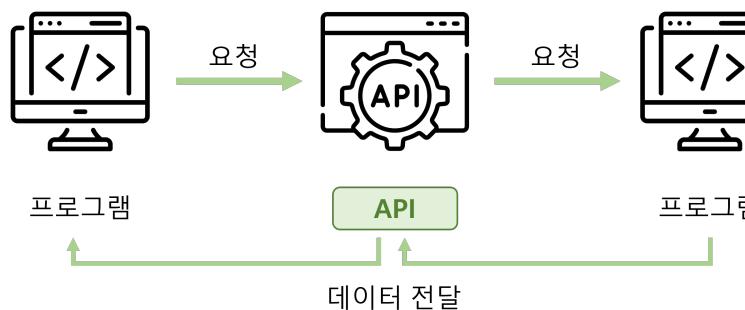
```
100% (100 of 100) |#####| Elapsed Time:
```

Unit 01 | Intro

API

(Application Programming Interface)

- 소프트웨어 컴포넌트(프로그램)간의 연결을 위한 매개체
- 컴포넌트를 활용하기 위한 규약
 - : 내 코드에서 동작하도록 하는 방법에 대한 정보

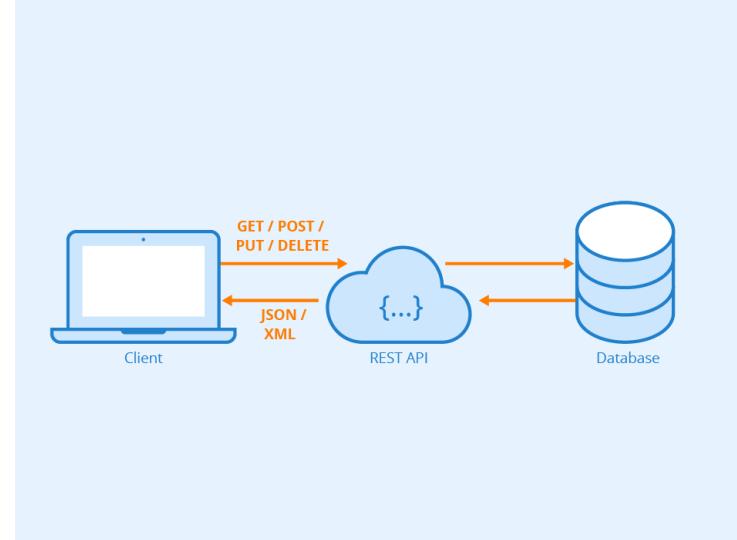


Unit 01 | Intro

REST API

REST

- **HTTP(:웹표준) 기반 클라이언트-서버 아키텍처**
- **HTTP URI로 resource를 지정하고,**
HTTP Method를 통해 해당 resource의 데이터 처리 동작 수행
- **Server-Client(서버-클라이언트 구조)**
: resource가 있는 쪽이 Server, 자원을 요청하는 쪽이 Client가 된다.



REST API

- REST를 기반으로 구현한 서비스 API

(Google Cloud에서 제공 하는 REST API)

The image shows two side-by-side screenshots of developer platforms:

- Google Cloud:** The "Speech-to-Text" page under the "Cloud Speech-to-Text" section. It features a "Speech-to-Text" input field, a "Transcript" output field, and a "Transcribe" button. Below the input field, there are buttons for "무료로 사용해 보기" (Try for free) and "영업팀에 문의" (Contact sales). A note at the bottom states: "최고의 Google AI 연구 및 기술로 지원되는 API를 사용하여 음성을 텍스트로 정확하게 변환할 수 있습니다." (You can accurately convert speech to text using the best Google AI research and technology). A list of pros follows: "✓ 정확한 자연으로 콘텐츠의 스크립트 작성", "✓ 음성을 활용해 더욱 우수한 사용자 경험 설계", and "✓ 고객 상호작용에서 얻은 유용한 정보로 서비스 개선".
- kakao developers:** The main landing page. It includes sections for "Cloud Speech-to-Text", "카카오 키워드 추출", "카카오 챗봇", and "카카오 API 특징". It also features a search bar and links for "제품 소개", "문서 보기", and "검색".

Unit 01 | Intro

Framework

- 소프트웨어 솔루션 개발을 위해 제공되는 '틀'(뼈대)
- 프로그램 흐름을 개발자가 아닌 프레임워크가 가지고 있다.
→ 개발자는 핵심 로직에만 집중할 수 있다.
- 미리 구현된 코드 조각들이 포함되어 있다.
: 라이브러리
- 프로그램이 수동적으로 프레임워크에 의해 사용된다.



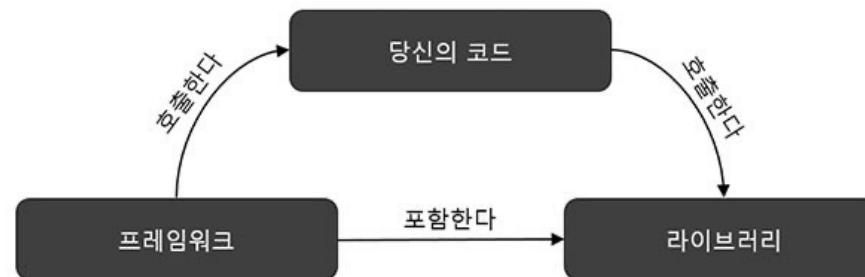
Unit 01 | Intro

라이브러리 vs API

- 라이브러리는 컴포넌트 자체를 뜻하고, API는 이 컴포넌트를 활용하는 규약

라이브러리 vs Framework

- 응용프로그램(애플리케이션)의 흐름 주도권을 누가 가지고 있는가
: main문을 바꿀 수 있으면 라이브러리고 아니면 프레임워크다 하는 식의 이야기도 나오곤 한다.



Unit 01 | Intro

코드의 재사용

모델 구현의 틀

Contents

Unit 01 | Intro

Unit 02 | 객체지향프로그래밍

Unit 03 | Framework

Unit 04 | TensorFlow, Keras, PyTorch

Unit 02 | 객체지향프로그래밍

객체지향프로그래밍

(Object Oriented Programming)

추상화

공통의 속성 묶어서 구현

캡슐화

정보 은닉 가능

상속

코드의 재사용

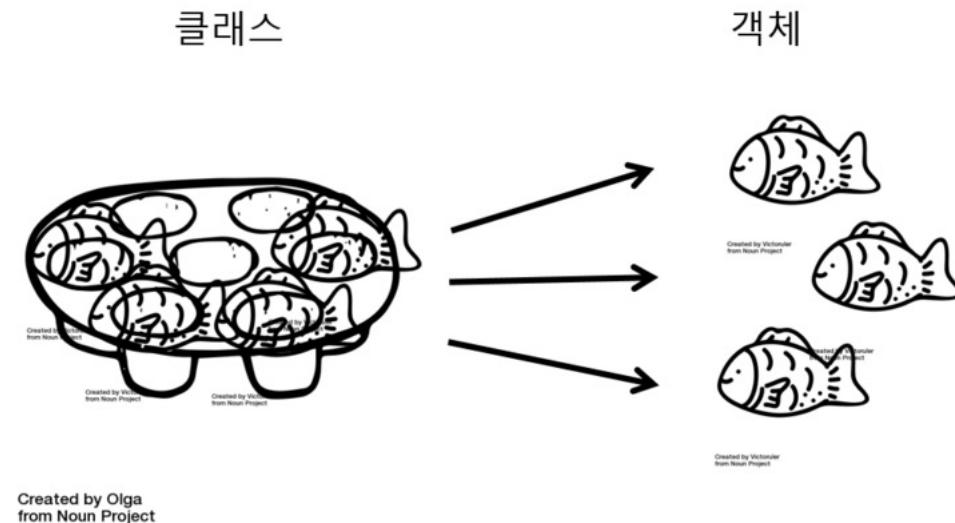
다형성

객체 변경 용이

Unit 02 | 객체지향프로그래밍

Class

: OOP의 모든 표준을 제공하는 방법



Unit 02 | 객체지향프로그래밍

Special Method

- 특정 상황에서 자동으로 호출 되는 메서드들(이름이 약속되어 있는 메소드)
- 메서드 이름이 “`__x__`” 형태를 가진다
 - : 앞, 뒤로 언더바 두 개 씩
- Python 표준에 맞게 클래스 설계하기 위해 필수
- 100개가 넘는 Special Method가 존재

Table 3.9 Special Methods for Object Creation, Destruction, and Representation

Method	Description
<code>__new__(cls [, *args , **kwargs])</code>	A static method called to create a new instance
<code>__init__(self [, *args , **kwargs])</code>	Called to initialize a new instance
<code>__del__(self)</code>	Called to destroy an instance
<code>__repr__(self)</code>	Creates a full string representation of an object
<code>__str__(self)</code>	Creates an informal string representation
<code>__cmp__(self, other)</code>	Compares two objects and returns negative, zero, or positive
<code>__hash__(self)</code>	Computes a 32-bit hash index
<code>__nonzero__(self)</code>	Returns 0 or 1 for truth-value testing
<code>__unicode__(self)</code>	Creates a Unicode string representation

Table 3.12 Methods for Sequences and Mappings

Method	Description
<code>__len__(self)</code>	Returns the length of self
<code>__getitem__(self, key)</code>	Returns <code>self[key]</code>
<code>__setitem__(self, key, value)</code>	Sets <code>self[key] = value</code>
<code>__delitem__(self, key)</code>	Deletes <code>self[key]</code>
<code>__getslice__(self, i, j)</code>	Returns <code>self[i:j]</code>
<code>__setslice__(self, i, j, s)</code>	Sets <code>self[i:j] = s</code>
<code>__delslice__(self, i, j)</code>	Deletes <code>self[i:j]</code>
<code>__contains__(self, obj)</code>	Returns <code>True</code> if <code>obj</code> is in self; otherwise, returns <code>False</code>

Unit 02 | 객체지향프로그래밍

생성자

- 객체가 생성될 때 자동으로 호출되는 메서드
- **special method “`__init__()`”**을 통해 선언
- 객체의 멤버 변수 선언

```
class Bbbsale:  
    def __init__(self, angkko):  
        self.angkko = angkko  
        self.min = 0  
  
    def set_angkko(self, angkko):  
        self.angkko = angkko  
  
Bbbsale_1st = Bbbsale("pat")  
Bbbsale_2nd = Bbbsale("chou")
```

```
class Bbbsale:  
    def __init__(self, angkko):  
        # self.angkko = angkko  
        self.min = 0  
  
    def set_angkko(self, angkko):  
        self.angkko = angkko
```

```
Traceback (most recent call last):  
  File "main.py", line 14, in <module>  
    first_day.intro()  
  File "main.py", line 10, in intro  
    print(self.angkko)  
AttributeError: 'Bbbsale' object has no attribute 'angkko'  
* []
```

Unit 02 | 객체지향프로그래밍

Default Parameter

: argument 전달하지 않아도 default value로 수행

```
class Bbbsale:  
    def __init__(self, angkkos, cooktime_chou = 1, cooktime_pizza = None, customers = None):  
        self.angkkos = angkkos  
        self.cooktime_pat = 2  
        self.cooktime_chou = cooktime_chou  
  
        if cooktime_pizza:  
            self.cooktime_pizza = cooktime_pizza  
        else:  
            self.cooktime_pizza = 3  
  
        if customers != None:  
            self.customers = customers  
  
    def intro(self): ...  
    def set_angkkos(self, angkkos): ...  
  
Bbbsale_1st = Bbbsale({"pat":3, "chou":3}, cooktime_chou = 3)  
Bbbsale_2nd = Bbbsale({"pat":1, "pizza":2}, cooktime_pizza = 7)  
Bbbsale_3rd = Bbbsale({"pat":1, "chou":1, "pizza":2}, customers = ["david", "tom"])  
  
print(Bbbsale_2nd.cooktime_pizza)  
print(Bbbsale_3rd.cooktime_pizza)
```

```
class Dense(tf.Module):  
    def __init__(self, in_features, out_features, name=None):  
        super().__init__(name=name)  
        self.w = tf.Variable(  
            tf.random.normal([in_features, out_features]), name='w')  
        self.b = tf.Variable(tf.zeros([out_features]), name='b')
```

Console Shell
7
3

Unit 02 | 객체지향프로그래밍

self

- 객체 자기자신을 참조
- 모든 멤버함수는 호출 시 항상 첫 번째 인자로 인스턴스가 전달
 - : [Python] 생성자와 멤버 함수 선언 할 때 첫 번째 인자로 “self”를 명시하지 않으면, error 발생
- 생성자, 객체의 메서드를 호출 할 때 self에 해당하는 인자는 전달하지 않는다.

```
class Bbsale:  
    def __init__(self, angkko):  
        self.angkko = angkko  
        self.min = 0  
  
    def intro():  
        print(self.angkko)  
        # undefined name 'self'  
        # pyflakes
```

```
class MyCell(torch.nn.Module):  
    def __init__(self):  
        super(MyCell, self).__init__()  
        self.linear = torch.nn.Linear(4, 4)  
  
    def forward(self, x, h):
```

```
object1 = Class_X()  
object1.setdata(4, 2)  
Class_X.setdata(a, 4, 2)  
# 객체의 멤버함수를 호출하는 다른 형태 : self에 해당하는 객체 인자 전달  
* 실제로 잘 사용하지 않지만, self 키워드의 이해에 도움이 된다.
```

* self 명시X : error

Unit 02 | 객체지향프로그래밍

__call__

- **Callable Object**
 - : 말 그대로 호출 가능 한 object (ex.메서드)
- class에 special method “__call__”을 내장하여, 그 class의 인스턴스를 callable object로 만들 수 있다.
 - Callable Object : 1)메서드 2)__call__함수를 내장한 객체
- 객체에 call을 내장하여 함수처럼 사용하는 이유?
 - : 일반 함수와 달리 OOP의 이점 적용 가능
 - (ex. 함수 자체의 상태 정보를 담은 멤버변수를 관리 할 수 있다.)

```

1 count = 0          # 함수 호출 회수를 기록할 전역 변수
2 def add(x, y) :
3     global count
4     count += 1    # 함수 호출 회수를 기록 한다
5     return x + y
6
7 n = add(1, 2)
8 print(count)      # 1

```

* 코드에서 변수 count와 함수 add() 사이에 연관성X

```

1 class Plus :
2
3     def __init__(self) :
4         self.count = 0
5
6     def __call__(self, x, y) :
7         self.count += 1
8         return x + y
9
10 plus = Plus()
11
12 n = plus(1, 2)
13
14 print(plus.count)

```

Unit 02 | 객체지향프로그래밍

상속

- 부모 클래스의 멤버 변수와 메서드를 (코드를)
자식 클래스가 재사용하는 개념
- 추상화
 - : Country 속성을 공유하는 class들 선언 가능
(Korea, USA, Japan, ...)
- 캡슐화
 - : 부모 클래스의 은닉(보호)하고 싶은 변수를
자식클래스들에서는 접근하지 못하도록 할 수 있다

```
class Country:  
    """Super Class"""  
  
    name = '국가명'  
    population = '인구'  
    capital = '수도'  
  
    def show(self):  
        print('국가 클래스의 메소드입니다.')  
  
    * 선언 때 소괄호로 부모 클래스 지정  
class Korea(Country):  
    """Sub Class"""  
  
    def __init__(self, name):  
        self.name = name  
  
    def show_name(self):  
        print('국가 이름은 : ', self.name)  
  
>>> a = Korea('대한민국')  
>>> a.show()  
국가 클래스의 메소드입니다.  
>>> a.show_name()  
국가 이름은 : 대한민국  
>>> a.capital  
'수도'  
>>> a.name  
'대한민국'
```

Unit 02 | 객체지향프로그래밍

Super()

- 부모 클래스를 호출하는 키워드
- 부모 클래스의 코드를 재사용 할 때 활용
→ 새로운 기능 덧붙일 때
- 오버라이딩(Overriding)
 - : 부모 클래스의 메서드를 자식 클래스에서 재정의 하는 것

```

class Family:
    def __init__(self):
        self.lastname = "홍"
    def lname(self):
        print("성은 %s입니다." %self.lastname)

class Person(Family):
    def __init__(self):
        self.firstname = "길동"
        super().__init__()
    def fname(self):
        print("이름은 %s입니다." %self.firstname)

a = Family()
b = Person()

print(a.lastname) # 홍
print(b.lastname) # 홍

class Family:
    def introduce(self):
        print("저희는 가족입니다.")

class Person(Family):
    def introduce(self):
        super().introduce()
        print("저는 가족의 구성원입니다.")

a.introduce() # 저희는 가족입니다.
b.introduce() # 저희는 가족입니다.
# 저는 가족의 구성원입니다.

```

Unit 02 | 객체지향프로그래밍

- ***args**

: Variable length argument (가변 인수)

: 여러 개의 argument가 들어올 때, 해당 변수들을 tuple로 처리

- ****kwargs**

: Variable length keyword argument (가변 키워드 인수)

: key가 있는 argument들을 dict로 처리

- **파라미터 순서**

1) 위치 인수

2) 기본값 인수

3) 가변 인수

4) 키워드 인수

5) 가변 키워드 인수

```
def mixed_params(age, name="아이유", *args, address, **kwargs):
    print("name=", end=""), print(name)
    print("args=", end=""), print(args)
    print("age=", end=""), print(age)
    print("kwargs=", end=""), print(kwargs)
    print("address=", end=""), print(address)

mixed_params(20, "정우성", "01012341234", "male", mobile="01012341234", address="seoul")
```

```
class addition:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def add(self):
        result = self.a + self.b
        return result

class calculator(addition):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.cal = 'I am a Calculator'

    >>> tool1=calculator(1,2)
    >>> tool1.add()      *args로 받는다
    3

    >>> tool2=calculator(a=1,b=2)
    >>> tool2.add()      **kwargs로 받는다
    3
```

Unit 02 | 객체지향프로그래밍

추상 클래스

- 미구현 추상 메서드를 한 개 이상 가진다
- 추상 클래스를 상속받은 자식 클래스에서 추상 메서드를 구현하지 않으면 error 발생
- 1) abc 모듈을 import
- 2) 선언 할 때 소괄호에 (metaclass=ABCMeta) 지정
- 3) 추상 메서드는 함수 위에 @abstractmethod 키워드로 다른 메서드들과 구별
- 공통된 속성(: 구현해야 할 코드에 대한 틀) 상속 받을 클래스에 제공하는 역할
→ Framework

```
from abc import *

class Family(metaclass=ABCMeta):
    @abstractmethod
    def introduce(self):
        pass

class Person(Family):
    def introduce(self):
        print("저는 사람입니다.")

a = Person()
a.introduce() # 저는 사람입니다.
```

Unit 02 | 객체지향프로그래밍

```
from keras.layers import Layer

class MyLayer(Layer):

    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        super(MyLayer, self).__init__(**kwargs)
```

```
from torch import nn

class Mnist_Logistic(nn.Module):
    def __init__(self):
        super().__init__()
        self.weights = nn.Parameter(torch.randn(784, 10) / math.sqrt(784))
        self.bias = nn.Parameter(torch.zeros(10))
```

Contents

Unit 01 | Intro

Unit 02 | 객체지향프로그래밍

Unit 03 | Framework

Unit 04 | TensorFlow, Keras, PyTorch

Unit 03 | Framework

프레임워크의 장점은?

1. 복잡한 모델의 구조를 몰라도 정해진 틀에 따라 사용해볼 수 있습니다.
2. GPU 활용하여 빠르게 처리할 수 있습니다.

Unit 03 | Framework



TensorFlow



PyTorch



Keras



DL4J



Caffe

Unit 03 | Framework



TensorFlow



PyTorch



Keras



DL4J



Caffe

Unit 03 | Framework

- 이 프레임워크들의 기본적인 data structure Tensor 란?

7	$\begin{bmatrix} 15 \\ -3 \\ \vdots \\ 7 \end{bmatrix}$	$\begin{bmatrix} 15 & 1 \dots 11 \\ -3 & 8 \dots 9 \\ \vdots & \vdots \ddots \vdots \\ 7 & 9 \dots 5 \end{bmatrix}$	$\left[\begin{array}{cccc} 15 & 1 & \dots & 11 \\ -3 & 8 & \dots & 9 \\ \vdots & \vdots & \ddots & \vdots \\ 7 & 9 & \dots & 5 \end{array} \right]$	
스칼라	벡터	행렬	3차원 텐서	N 차원 텐서
$x[1] = 5$	$x[1,0] = -3$		$x[0,1,1] = 8$	$x[\underbrace{1,3,\dots,1}_{N\text{개 인덱스}}] = 7$
0차원	1차원	2차원	3차원	N 차원

- Numpy Array와 같은 다차원 배열(multidimensional array)
- GPU등 HW 가속기에서 실행 할 수 있는 자료구조 → 연산 속도 가속

Unit 03 | Framework

TensorFlow 란?

TensorFlow를 사용해야 하는 이유

TensorFlow는 머신러닝을 위한 앤드 투 앤드 오픈소스 플랫폼입니다. 도구, 라이브러리, 커뮤니티 리소스로 구성된 포괄적이고 유연한 생태계를 통해 연구원들은 ML에서 첨단 기술을 구현할 수 있고 개발자들은 ML이 접목된 애플리케이션을 손쉽게 빌드 및 배포할 수 있습니다.

정보 →



손쉬운 모델 빌드

즉각적인 모델 반복 및 손쉬운 디버깅을 가능하게 하는 즉시 실행 기능이 포함된 Keras와 같은 높은 수준의 직관적인 API를 사용하여 ML 모델을 쉽게 빌드하고 학습시키세요.



어디서든 강력한 ML 제작

사용하는 언어에 상관없이 클라우드, 온프레, 브라우저 또는 기기에서 모델을 손쉽게 학습시키고 배포하세요.



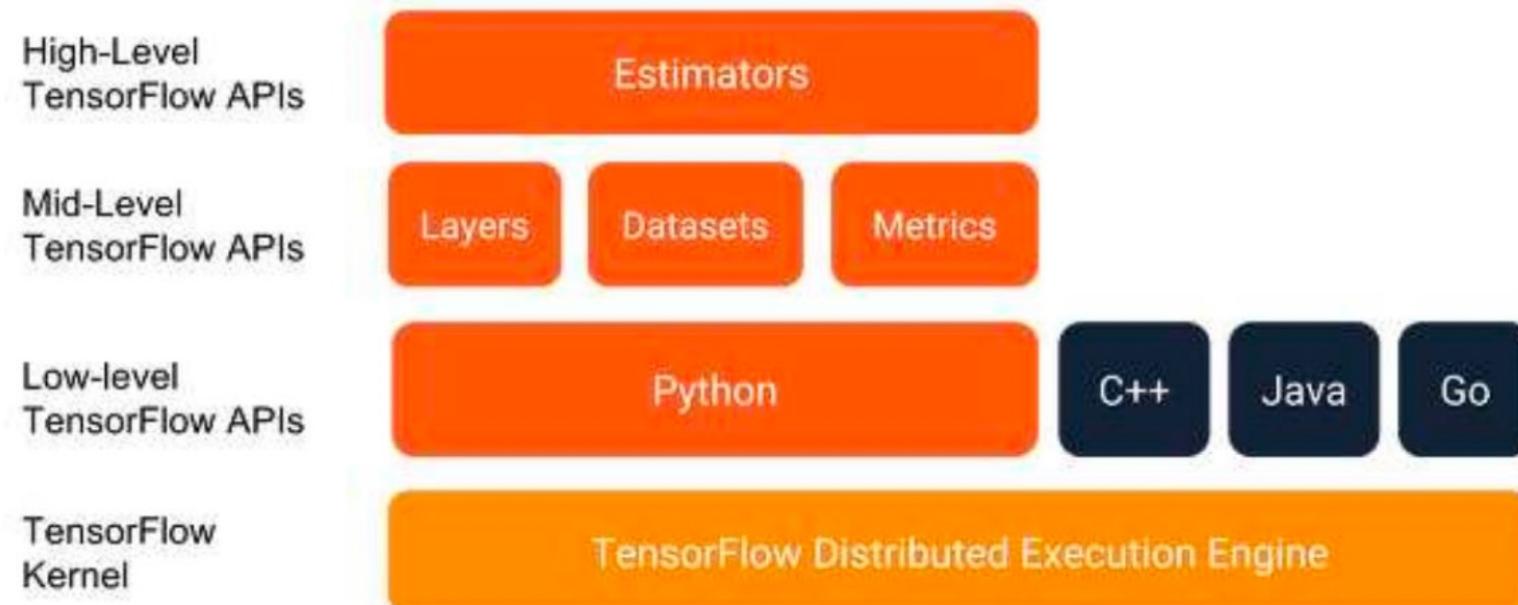
연구를 위한 강력한 실험

새로운 아이디어를 개념부터 코드, 최첨단 모델, 개시에 이르기까지 더 빠르게 발전시킬 수 있는 간단하고 유연한 아키텍처

딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 머신러닝 라이브러리로, 구글에서 만들었음.

Unit 03 | Framework

TensorFlow 란?

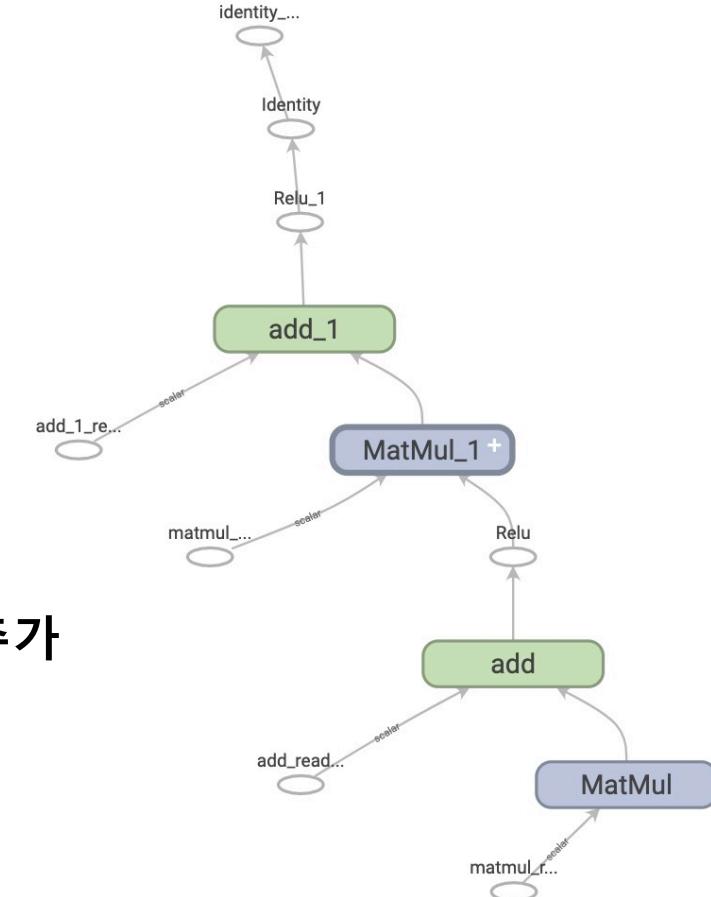


파이썬을 최우선으로 지원하여 대다수 편의 기능이 **파이썬 라이브로리**로만 구현되어있음.

Unit 03 | Framework

그래프(Static Computational Graph)

- TensorFlow에서 Tensor의 계산이 진행되는 방식
- 구성
 - tf.Operation : 계산(작업)의 단위 (graph의 node)
 - tf.Tensor : 데이터의 단위 (graph의 edge)
- TensorFlow API 함수 호출
 - 생성된 tf.Tensor 객체와 tf.Operation 객체를 tf.Graph 인스턴스에 추가
- 디버거에 단일 명령문 [tf.Session.run()]으로 그래프를 실행
 - 그래프의 내부 구조와 상태가 노출되지 않는다.
 - 디버깅 어렵다.



Unit 03 | Framework

그래프(Static Computational Graph)

- **병행.** 명시적인 경계를 사용하여 연산 간의 종속성을 나타냄으로써 시스템이 병렬로 실행할 수 있는 연산을 쉽게 식별 할 수 있습니다.
- **분산 실행.** 명시적인 가장자리를 사용하여 작업간에 흐르는 값을 나타내면 TensorFlow가 다른 컴퓨터에 연결된 여러 장치 (CPU, GPU 및 TPUs)에 프로그램을 분할 할 수 있습니다. TensorFlow는 장치간에 필요한 통신 및 조정을 삽입합니다.
- **편집.** TensorFlow의 XLA 컴파일러는 데이터 흐름 그래프의 정보를 사용하여 인접한 작업을 융합하는 등보다 빠른 코드를 생성 할 수 있습니다.
- **이식성.** 데이터 흐름 그래프는 모델에있는 코드의 언어 독립적 인 표현입니다. Python으로 데이터 흐름 그래프를 작성하고, SavedModel에 저장하고, C ++ 프로그램에서 이를 복원하여 짧은 시간이 짧은 추론을 할 수 있습니다.

Unit 03 | Framework

**그래프의 사용으로 TensorFlow는 빠르게, 병렬로,
효율적으로 여러 기기에서 실행될 수 있습니다.**

Unit 03 | Framework

Keras 란?

케라스: 파이썬 딥러닝 라이브러리



케라스에 오신걸 환영합니다.

케라스는 파이썬으로 작성된 고수준 신경망 API로 **TensorFlow**, **CNTK**, 혹은 **Theano**와 함께 사용하실 수 있습니다. 빠른 실험에 특히 중점을 두고 있습니다. 아이디어를 결과물로 최대한 빠르게 구현하는 것은 훌륭한 연구의 핵심입니다.

케라스를 사용하면:

- (사용자 친화성, 모듈성, 확장성을 통해) 빠르고 간편한 프로토타이핑을 할 수 있습니다.
- 컨볼루션 신경망, 순환 신경망, 그리고 둘의 조합까지 모두 지원됩니다.
- CPU와 GPU에서 매끄럽게 실행됩니다.

Unit 03 | Framework

Keras 란?

- 파이썬으로 작성된, TensorFlow 위에서 동작하는 오픈 소스 신경망 라이브러리
- 빠른 실험을 가능케 하도록 만들어졌으며 최소한의 모듈 방식의 확장 가능성에 초점
- TensorFlow의 문제를 해결하기 위해 개발된, 더욱 단순화된 인터페이스를 제공하는 프레임워크

Unit 03 | Framework

PyTorch 란?



Deep Learning with PyTorch

PyTorch 소개

PyTorch(파이토치)는 연구용 프로토타입부터 상용 제품까지 빠르게 만들 수 있는 오픈 소스 머신러닝 프레임워크입니다.

PyTorch는 사용자 친화적인 프론트엔드(front-end)와 분산 학습, 다양한 도구와 라이브러리를 통해 빠르고 유연한 실험 및 효과적인 상용화를 가능하게 합니다.

PyTorch에 대한 더 자세한 소개는 [공식 홈페이지](#) 또는 [공식 저장소](#)에서 확인하실 수 있습니다.

2016년에 발표된 딥러닝 구현을 위한 파이썬 기반의 오픈소스 머신러닝 라이브러리로 facebook 인공지능 연구팀에 의해 개발되었음

Unit 03 | Framework

Static Computational Graph : TensorFlow

- Define and Run (Static) : 컴파일 타임에 graph 정의 후 수행 시 data feed
- 디버깅 어려움

Dynamic Computational Graph : PyTorch

- Define by Run (Dynamic) : forward propagation 수행 시 런타임에 graph가 동적으로 정의
- 중간 중간 값을 확인 할 수 있다.
: 디버깅 쉽게 가능

Unit 03 | Framework

TensorFlow vs PyTorch - Computational Graph

TensorFlow: Build graph once, then run many times (**static**)

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.Variable(tf.random_normal((D, H)))
w2 = tf.Variable(tf.random_normal((H, D)))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

learning_rate = 1e-5
new_w1 = w1.assign(w1 - learning_rate * grad_w1)
new_w2 = w2.assign(w2 - learning_rate * grad_w2)
updates = tf.group(new_w1, new_w2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    values = {x: np.random.randn(N, D),
              y: np.random.randn(N, D),}
    losses = []
    for t in range(50):
        loss_val, _ = sess.run([loss, updates],
                             feed_dict=values)
```

Build graph

Run each iteration

PyTorch: Each forward pass defines a new graph (**dynamic**)

```
import torch
from torch.autograd import Variable

N, D_in, H, D_out = 64, 1000, 100, 10
x = Variable(torch.randn(N, D_in), requires_grad=False)
y = Variable(torch.randn(N, D_out), requires_grad=False)
w1 = Variable(torch.randn(D_in, H), requires_grad=True)
w2 = Variable(torch.randn(H, D_out), requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    if w1.grad: w1.grad.data.zero_()
    if w2.grad: w2.grad.data.zero_()
    loss.backward()

    w1.data -= learning_rate * w1.grad.data
    w2.data -= learning_rate * w2.grad.data
```

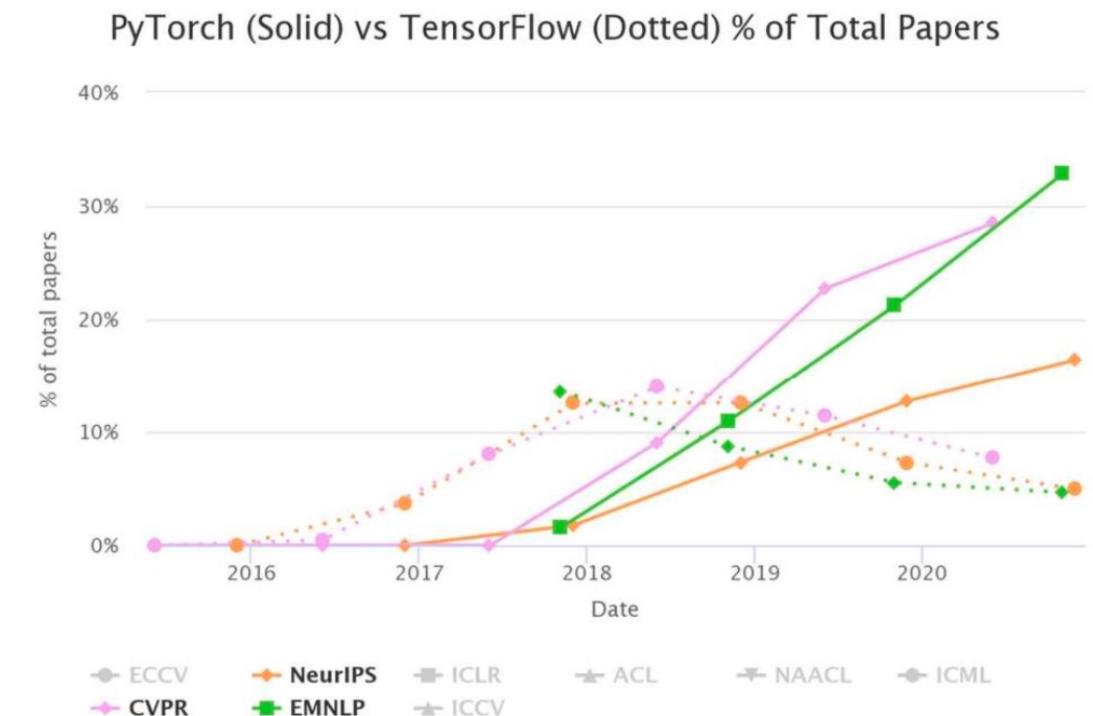
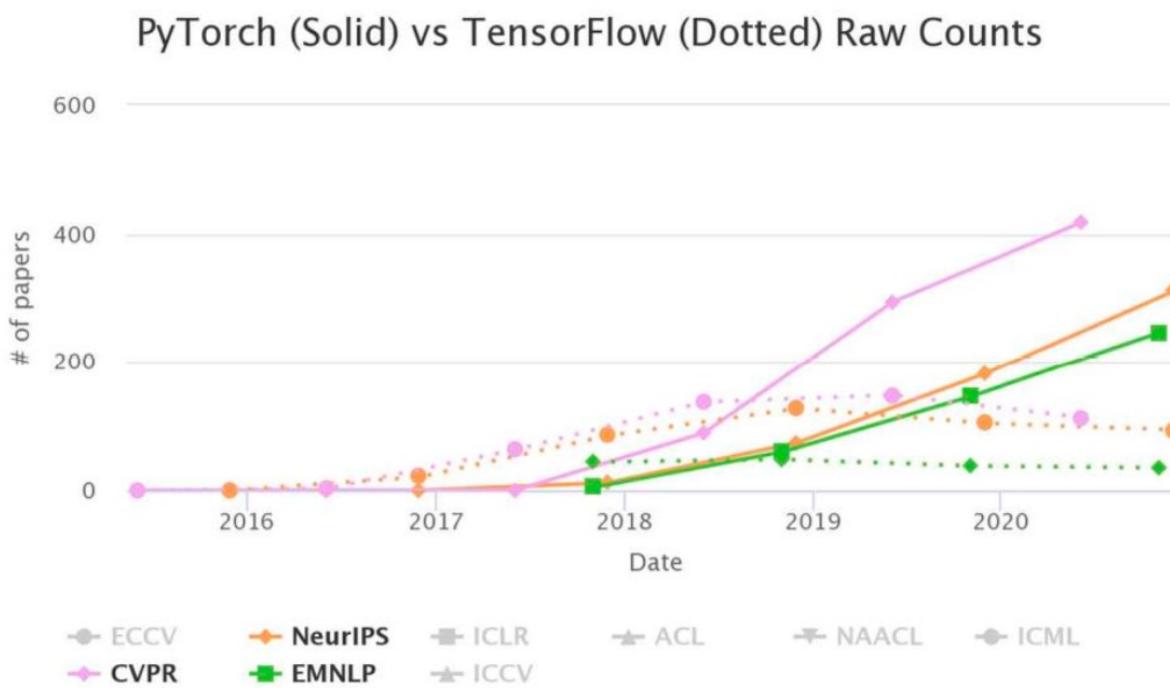
New graph each iteration

Unit 03 | Framework

TensorFlow vs PyTorch – Management & Production

- **TensorFlow**
 - CPU와, GPU 모두 사용 가능하며 각각 따로 구현되어 있다.
GPU가속의 자동화로 매모리 사용량 제어가 불가능하다.
 - Serving의 관점에서 매우 유연하다. 다양한 Serving tool이 잘 마련되어 있다.
- **PyTorch**
 - CUDA가 반드시 존재해야 하고, Tensorflow에 비해서 사용자가 리소스 사용을 효과적으로 제어할 수 있어 훈련 과정에서 더 유용하다.
 - 아직 초기 실험적 단계의 Serving tool들이 나오고 있지만 빠른 발전을 보이고 있다.

Unit 03 | Framework



Pytorch로 쓰이는 논문 많아지고 있는 경향

Content

Unit 01 | Intro

Unit 02 | 객체지향프로그래밍

Unit 03 | Framework

Unit 04 | TensorFlow, Keras, PyTorch

Unit 04 | TensorFlow, Keras, PyTorch

TensorFlow

The screenshot shows the TensorFlow API documentation for the `tf.Module` class. The page title is "tf.Module". It includes a "TensorFlow 1 version" button and a "View source on GitHub" button. A brief description states: "Base neural network module class.". Below this, there is a "View aliases" section and a code snippet showing the constructor:

```
tf.Module(  
    name=None  
)
```

```
import tensorflow as tf

class SimpleModule(tf.Module):
    def __init__(self, name=None):
        super().__init__(name=name)
        self.a_variable = tf.Variable(5.0, name="train_me")
        self.non_trainable_variable = tf.Variable(5.0, trainable=False,
                                                name="do_not_train_me")

    def __call__(self, x):
        return self.a_variable * x + self.non_trainable_variable

simple_module = SimpleModule(name="simple")

simple_module(tf.constant(5.0))

<tf.Tensor: shape=(), dtype=float32, numpy=30.0>
```

- **[tf.Module]을 상속**
- **tf.Module**
 - 기본 신경망 모듈 클래스.
 - `tf.keras.layers.Layer`, `tf.keras.Model`도 `tf.Module`을 상속합니다.

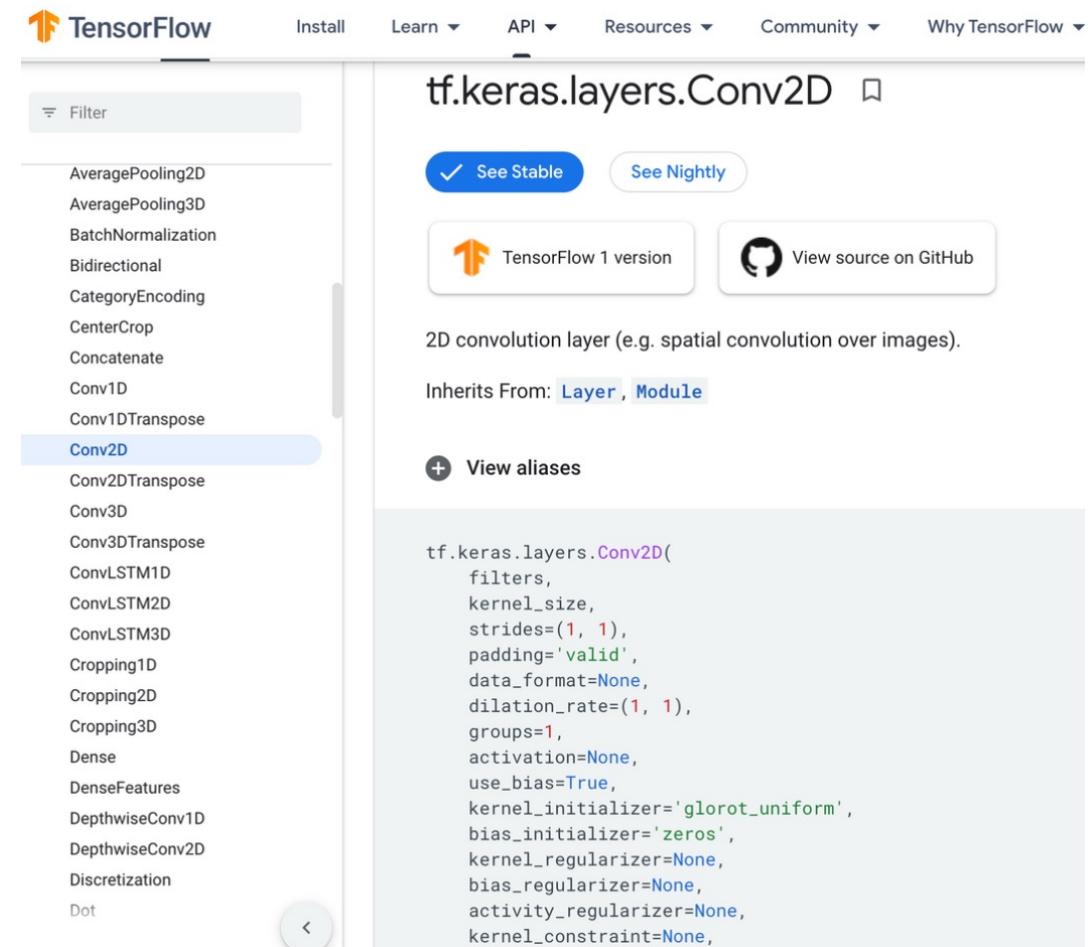
Unit 04 | TensorFlow, Keras, PyTorch

Docs

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

- Framework, 라이브러리를 사용함에 있어서
공식 문서를 찾아보는 것은 필수적이고,
습관화 해야한다!
- **tf.keras.layers.Conv2D**

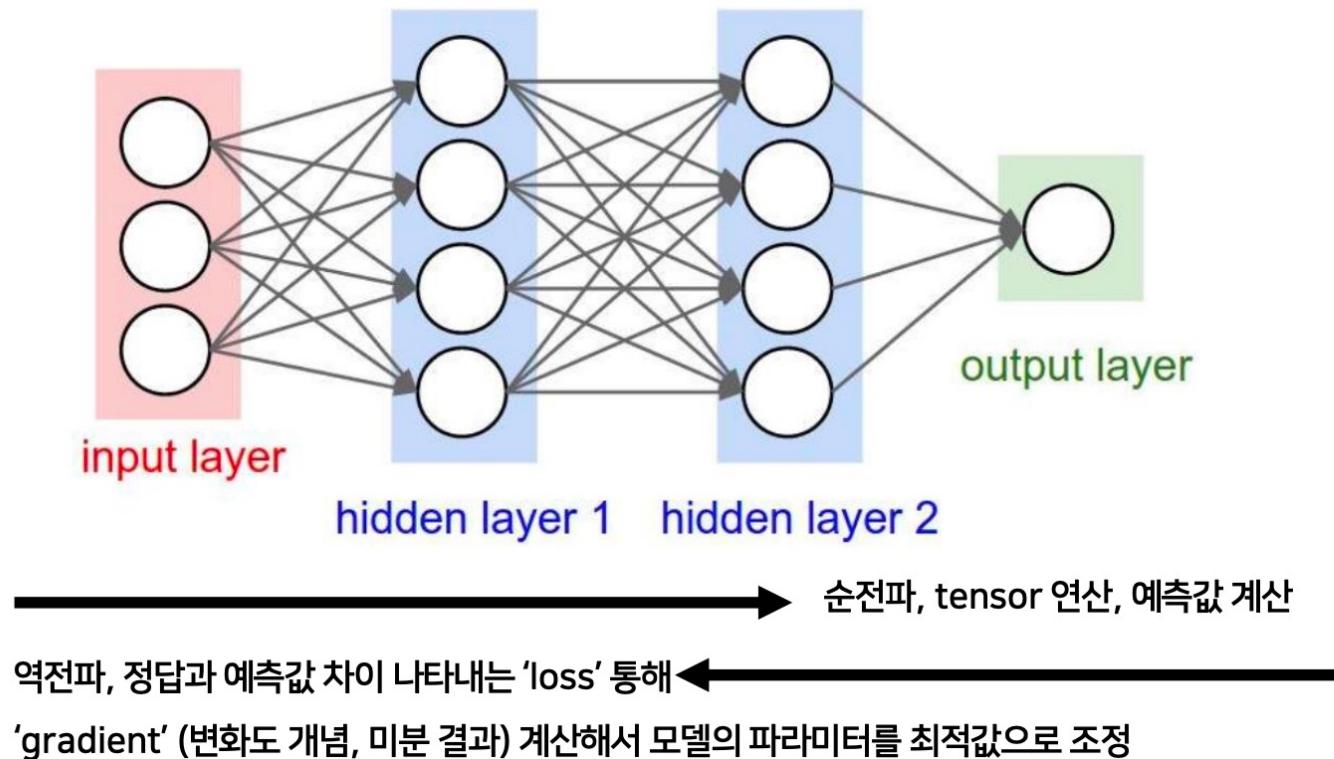


* https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

Unit 04 | TensorFlow, Keras, PyTorch

- MNIST 샘플 데이터로 Deep Neural Network 구현하는 예제 _ 프레임워크별로 살펴보기

OVERVIEW



- 데이터셋 불러오고 전처리
- 데이터 로더 설정
- 모델 선언
- 손실함수, 옵티마이저 선언
- 모델 학습
- 모델 평가

Unit 04 | TensorFlow, Keras, PyTorch

- 데이터 로더

- 데이터셋을 한 번에 모두 램에 올리지 않고,
- 각 미니배치 만큼만 램에 올리도록 하여 효율적인 연산 가능하게 함

- 모델 선언

- 텐서플로, 파이토치에서 모델 선언 시 모델 관련 추상 클래스를 상속받아
- __init__ 메소드로 각 레이어를 생성하고
- 각각 call, forward 메소드를 이용해 데이터가 각 레이어를 통과하도록 함

+ 추상클래스는 메서드의 구현은 없이 이름만 존재하도록 정의해 둔 클래스를 의미한다.

프레임워크의 추상클래스를 모델 선언시에 상속 받아, 구현이 요구되는 특정 메서드들을 구현하면 되는 상황

- 손실함수와 옵티마이저

- 태스크에 맞는 손실함수를 선언 손실함수 정리 <https://bskyvision.com/822>
- 손실함수를 통해 계산된 값을 이용해 가중치를 최적화하는 옵티마이저 선언

Unit 04-1 | TensorFlow, Keras

TensorFlow (Keras)

Unit 04-1 | TensorFlow, Keras

NN Implementation Flow in TensorFlow

1. Set hyper parameters ; learning rate, training epochs, batch size, ect.
2. Data Agument ; rotate & shift
3. Make a data pipelining ; use tf.data
4. Build a neural network model ; use tf.keras
5. Define a loss function ; cross entropy
6. Calculate a gradient ; use tf.GradientTape
7. Select an optimizer ; Adam optimizer
8. Define a metric for model's performance ; accuracy
9. (optional) Make a checkpoint for saving
10. Train and Validate a neural network model

Unit 04-1 | TensorFlow, Keras

Keras - Model 클래스

- **tf.keras.Model**
- **keras.layers.Layer** 클래스에
내장 훈련, 평가 및 예측 루프[**model.fit()** ,
model.evaluate(), **model.predict()**]를 추가 제공
- **call()**
 - **tf.keras**가 모델이나 계층을 호출할 때 내부 구조를 유지하는 데 필수적인 자체 내부 작업
호출
 - “**__call__**” Special Method의 재정의 함수

```
import tensorflow as tf
from tensorflow import keras

class MNISTModel(tf.keras.Model):
    def __init__(self):
        super(MNISTModel, self).__init__()
        self.conv1 = ConvBNRelu(filters=32, kernel_size=[3, 3], padding='SAME')
        self.pool1 = keras.layers.MaxPool2D(padding='SAME')
        self.conv2 = ConvBNRelu(filters=64, kernel_size=[3, 3], padding='SAME')
        self.pool2 = keras.layers.MaxPool2D(padding='SAME')
        self.conv3 = ConvBNRelu(filters=128, kernel_size=[3, 3], padding='SAME')
        self.pool3 = keras.layers.MaxPool2D(padding='SAME')
        self.pool3_flat = keras.layers.Flatten()
        self.dense4 = DenseBNRelu(units=256)
        self.drop4 = keras.layers.Dropout(rate=0.4)
        self.dense5 = keras.layers.Dense(units=10, kernel_initializer='glorot_normal')

    def call(self, inputs, training=False):
        net = self.conv1(inputs)
        net = self.pool1(net)
        net = self.conv2(net)
        net = self.pool2(net)
        net = self.conv3(net)
        net = self.pool3(net)
        net = self.pool3_flat(net)
        net = self.dense4(net)
        net = self.drop4(net)
        net = self.dense5(net)
        return net
```

Unit 04-1 | TensorFlow, Keras

Loss Function

```
def loss_fn(model, images, labels):
    logits = model(images, training=True)
    loss = tf.reduce_mean(tf.keras.losses.categorical_crossentropy(
        y_pred=logits, y_true=labels, from_logits=True))
    return loss
```

Calculating Gradient

```
def grad(model, images, labels):
    with tf.GradientTape() as tape:
        loss = loss_fn(model, images, labels)
    return tape.gradient(loss, model.trainable_variables)
```

Caculating Model's Accuracy

```
def evaluate(models, images, labels):
    predictions = np.zeros_like(labels)
    for model in models:
        logits = model(images, training=False)
        predictions += logits
    correct_prediction = tf.equal(tf.argmax(predictions, 1), tf.argmax(labels, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    return accuracy
```

Optimizer

```
lr_decay = tf.keras.optimizers.schedules.ExponentialDecay(learning_rate,
                                                       train_images.shape[0]/batch_size*num_models*5,
                                                       0.5, staircase=True)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_decay)
```

Training

```
# train my model
print('Learning started. It takes sometime.')
for epoch in range(training_epochs):
    avg_loss = 0.
    avg_train_acc = 0.
    avg_test_acc = 0.
    train_step = 0
    test_step = 0
    for images, labels in train_dataset:
        for model in models:
            grads = grad(model, images, labels)
            optimizer.apply_gradients(zip(grads, model.trainable_variables))
            loss = loss_fn(model, images, labels)
            avg_loss += loss / num_models
            acc = evaluate(models, images, labels)
            avg_train_acc += acc
            train_step += 1
        avg_loss = avg_loss / train_step
        avg_train_acc = avg_train_acc / train_step

    for images, labels in test_dataset:
        acc = evaluate(models, images, labels)
        avg_test_acc += acc
        test_step += 1
    avg_test_acc = avg_test_acc / test_step
```

Unit 04-2 | PyTorch

PyTorch

Unit 04-2 | PyTorch

PyTorch - DataLoader

- 생성한 dataset을 설정한 옵션에 맞게 batchsize로 슬라이싱하여 mini-batches로 만들어준다.
- 모델 학습을 위해 전달 할 data sample의 minibatch의 batch_size를 설정,
매 epoch 마다 데이터를 섞어 overfitting을 방지,
multiprocessing을 통해 데이터 검색 속도를 높이는 과정들을 추상화하여 제공하는 객체

```
from torch.utils.data import DataLoader

train_dataloader = DataLoader(training_data, batch_size=64, shuffle=True)
test_dataloader = DataLoader(test_data, batch_size=64, shuffle=True)
```

Unit 04-2 | PyTorch

PyTorch - Parameter Optimize

- loss function으로 loss 계산
- Backpropagation
 - **optimizer.zero_grad()** : backward()함수는 새로운 gradient를 기존 gradient에 (+=) 누적시키므로 0으로 초기화해줘야 한다.
 - **loss.backwards()** : loss function의 미분값, gradients 저장
 - **optimizer.step()** : gradients를 바탕으로 parameter 조정

```
def train_loop(dataloader, model, loss_fn, optimizer):  
    size = len(dataloader.dataset)  
    for batch, (X, y) in enumerate(dataloader):  
        # 예측(prediction)과 손실(loss) 계산  
        pred = model(X)  
        loss = loss_fn(pred, y)  
  
        # 역전파  
        optimizer.zero_grad()  
        loss.backward()  
        optimizer.step()
```

과제

과제

- Framework_PyTorch.ipynb 파일

1) 구현 흐름 6-단계 이상으로 구분하여 정리하기

(텍스트 셀에 (markdown 활용) 각 단계의 구현 기능 묘사)

2) API(함수)가 사용된 코드에 대한 주석 10개 이상 작성

(해당 line에서 구현한 기능 묘사)

Reference

[강의안]

- 투빅스 16기 김송민님 강의안

[참고자료]

<https://velog.io/@somday/RESTful-API-이란>

<https://eine.tistory.com/entry/라이브러리-API-ABI-뜻-비교-정리>

<https://kukuta.tistory.com https://velog.io/@0sunset0/OOP의-네가지-특징추상화캡슐화상속다형성>

<https://www.tensorflow.org/guide?hl=ko>

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=ji_sung31&logNo=221174179409

<https://developers-kr.googleblog.com/2017/03/debug-tensorflow-models-with-tfdbg.html>

http://wiki.hash.kr/index.php/파일:붕어빵틀과_붕어빵.png

<https://keras.io/ko/layers/writing-your-own-keras-layers/>

<https://hwan-hobby.tistory.com/93>

<https://wikidocs.net/16073>

<https://velog.io/@byoungju1012/TIL-10.-Python-Function-Parameter>

<https://hyoseok-personality.tistory.com/m/6>

<https://engineer-mole.tistory.com/21>

<https://yong0810.tistory.com/17>

Q & A

들어주셔서 감사합니다.