

## 3장 케라스와 텐서플로우

## 주요 내용

- 딥러닝 필수 요소
- 케라스와 텐서플로우 간략 소개
- 텐서플로우, 케라스, GPU를 활용한 딥러닝 작업환경
- 케라스와 텐서플로우를 이용한 신경망의 핵심 구성요소 구현

## 3.1 텐서플로우 소개

## 텐서플로우

- 구글을 중심으로 개발된 머신러닝 플랫폼(platform)
  - TF-Agents: 강화학습 연구 지원
  - TFX: 머신러닝 프로젝트 진행과정(workflow) 운영 지원
  - TF-Hub: 훈련된 모델 제공
- 파이썬 기반
- 텐서 연산 지원

## 넘파이(Numpy)와의 차이점

- 미분 가능한 함수들의 그래디언트 자동 계산
- GPU, TPU 등 고성능 병렬 하드웨어 가속기 활용 가능
  - 높은 확장성: 일기예보, 바둑 프로그램 등 매우 많은 데이터와 계산이 요구되는 실전 상황에 활용됨.
- C++(게임), 자바스크립트(웹브라우저), TFLite(모바일 장치) 등 다른 언어가 선호되는 도메인 특화 프로그램에 쉽게 이식 가능

## 3.2 케라스

# 케라스와 텐서플로우

- 딥러닝 모델 훈련에 최적화된 인터페이스 제공.
- 원래 텐서플로우와 독립적으로 시작됨.
- 텐서플로우 2.0부터 텐서플로우 라이브러리의 최상위 프레임워크(framework)로 포함됨.
- 다양한 워크플로우 제공: 모델 구축과 훈련 방식에 있어서 고수준/저수준 방식 모두 제공

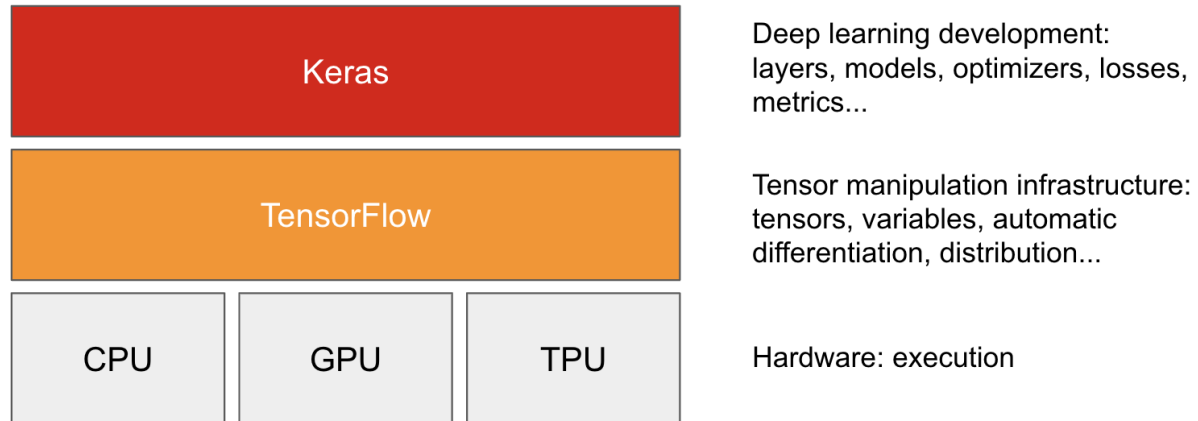


그림 출처: [Deep Learning with Python\(Manning MEAP\)](#)

### 3.3 케라스와 텐서플로우의 약력



- 2007년: 씨아노(Theano) 공개. 캐나다 몬트리올 대학교 연구팀.
  - 계산 그래프, 미분 자동화 등을 최초로 활용
- 2015년 3월: 케라스 라이브러리 공개
  - 씨아노(Theano)를 백엔드로 사용하는 고수준 패키지
- 2015년 11월: 텐서플로우 라이브러리 공개
- 2016년: 텐서플로우가 케라스의 기본 백엔드로 지정됨
- 2017년: 씨아노, 텐서플로우, CNTK(마이크로소프트), MXNet(아마존)이 케라스의 백엔드로 지원됨.
- 2019년 9월: 텐서플로우 2.0부터 케라스가 텐서플로우의 최상위 프레임워크로 지정됨.

## 3.4 딥러닝 작업환경

## GPU 활용 옵션

- 개인 NVIDIA 그래픽카드가 장착된 PC 또는 노트북 사용
  - 딥러닝을 많이 활용하는 경우
  - Ubuntu 설치 또는 WSL(Windows Subsystem for Linux) 활용 추천
- 구글 클라우드 플랫폼 또는 아마존 웹서비스(AWS EC2) 활용
  - 단기간동안 고성능 컴퓨터를 활용하고자 하는 경우
- **구글 코랩 활용**
  - 강좌 이수 용도로 추천

## 구글 코랩 사용

- 기본 사용법은 인터넷 검색 참조
- 코드 실행에 필요한 추가 패키지 설치: pip(파이썬 패키지 관리자) 활용

```
!pip install package_name
```

- 참고: 느낌표(!)는 주피터 노트북 코드셀에서 터미널 명령어를 실행하는 경우 사용
- GPU 활용: 런타임 유형을 GPU로 지정만 하면 됨.
- TPU 활용: 좀 더 복잡한 세팅 필요. 13장 참조.

## 3.5 텐서플로우 기본 사용법

# 신경망 모델 훈련 핵심 1

## 1. 상수 텐서와 변수 텐서

- 상수 텐서(constant tensor): 입출력 데이터 등 변하지 않는 텐서
- 변수 텐서(variable): 모델 가중치, 편향 등 업데이트 되는 텐서

## 2. 텐서 연산: 덧셈, relu, 점곱 등

## 3. 역전파(backpropagation):

- 손실함수의 그래디언트 계산 후 모델 가중치 업데이트
- 그래디언트 테이프(GradientTape) 이용

## 3.6 케이스의 핵심 API 이해

## 신경망 모델 훈련 핵심 2

1. 층(layer)과 모델: 층을 적절하게 쌓아 모델 구성
2. 손실 함수(loss function): 학습 방향을 유도하는 피드백 역할 수행
3. 옵티마이저(optimizer): 학습 방향을 정하는 기능 수행
4. 메트릭(metric): 정확도 등 모델 성능 평가 용도
5. 훈련 반복(training loop): 미니 배치 경사하강법 실행



## 층(layer)의 역할

- 모델의 상태(지식)로 사용되는 가중치(weight)와 편향(bias) 저장
- 데이터 표현 변환(forward pass)
- 케라스 활용 딥러닝 모델: 호환 가능한 층들의 적절한 연결

## 층의 종류와 처리 가능 텐서

- Dense 클래스를 사용하는 밀집층(dense layer): (샘플수, 특성수) 모양의 2D 텐서로 제공된 데이터셋
- LSTM 클래스, Conv1D 클래스 등을 사용하는 순환층(recurrent layer): (샘플수, 타임스텝수, 특성수) 모양의 3D 텐서로 제공된 순차 데이터셋
- Conv2D 클래스 등을 사용하는 층: (샘플수, 가로, 세로, 채널수) 모양의 4D 텐서로 제공된 이미지 데이터셋

## Layer 클래스와 `__call__()` 메서드

- 케라스에서 사용되는 모든 층에 대한 부모 클래스
- `__call__()` 메서드의 역할
  - 가중치와 편향 벡터 생성 및 초기화
  - 입력 데이터를 출력 데이터로 변환

## `__call__()` 메서드의 대략적 정의

```
def __call__(self, inputs):  
    if not self.built:  
        self.build(inputs.shape)  
        self.built = True  
    return self.call(inputs)
```

- `self.built`: 가중치와 편향 벡터가 초기화가 되어 있는지 여부 기억
- `self.build(inputs.shape)`: 입력 배치 데이터셋(`inputs`)의 모양(shape) 정보 이용
  - 가중치 텐서 생성 및 무작위적으로 초기화
  - 편향 텐서 생성 및 0벡터로 초기화
- `self.call(inputs)`: 출력값 계산(forward pass)
  - 아핀 변환 및 활성화 함수 적용

## 층에서 모델로

- 입력값을 보고 바로 입력값의 모양 확인
- MNIST 모델 사용된 `Dense` 클래스처럼 입력 데이터에 정보 미리 요구하지 않음

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.SimpleDense(512, activation="relu"),
    layers.SimpleDense(10, activation="softmax")
])
```

# 딥러닝 모델

- 층으로 구성된 그래프
- 예제: Sequential 모델
  - 층을 일렬로 쌓은 신경망 제공
  - 아래 층에서 전달한 값을 받아 변환한 후 위 층으로 전달
- 예제: 트랜스포머(Transformer)

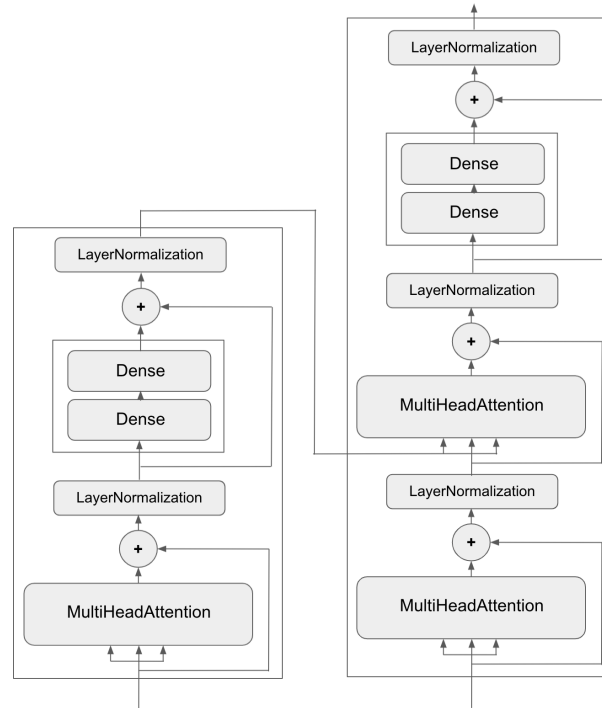


그림 출처: [Deep Learning with Python\(Manning MEAP\)](#)

## 망 구성방식과 가설 공간

- 모델의 학습과정은 층을 어떻게 구성하였는가에 전적으로 의존함.
- 여러 개의 Dense 층을 이용한 Sequential 모델
  - 아핀 변환, `relu()` 등의 활성화 함수를 연속적으로 적용한 데이터 표현 변환
- 다른 방식으로 구성된 모델: 다른 방식으로 텐서 표현 변환
- 이렇듯 층을 구성하는 방식에 따라 텐서들이 가질 수 있는 표현들의 공간이 정해짐.
- '망 구성방식(network topology)에 따른 표현 가설 공간(hypothesis space)'이 지정 됨.
- 신경망의 구성
  - 주어진 데이터셋과 모델의 목적에 따라 결정됨.
  - 특별한 규칙 또는 이론은 없음.
  - 이론 보다는 많은 실습을 통한 경험에 의존

## 모델 컴파일

모델의 구조를 정의한 후에 아래 세 가지 설정을 추가로 지정해야 함.

- 옵티마이저(optimizer): 모델의 성능을 향상시키는 방향으로 가중치를 업데이트하는 알고리즘
- 손실함수(loss function): 훈련 중 모델의 성능 얼마 나쁜가를 측정하는 기준. 미분가능 이어야 하며 옵티마이저가 경사하강법을 활용하여 손실함숫값을 줄이는 방향으로 작동함.
- 평가지표(metrics):: 훈련과 테스트 과정을 모니터링 할 때 사용되는 모델 평가 지표. 옵티마이저 또는 손실함수와 일반적으로 상관 없음.



## fit() 메서드 작동법

모델을 훈련시키려면 `fit()` 메서드를 적절한 인자들과 함께 호출해야 함.

- 훈련 세트: 보통 넘파이 어레이 또는 텐서플로우의 `Dataset` 객체 사용
- 에포크(`epochs`): 전체 훈련 세트를 몇 번 훈련할 지 지정
- 배치 크기(`batch_size`): 배치 경사하강법에 적용될 배치(묶음) 크기 지정

아래 코드는 앞서 넘파이 어레이로 생성한 (2000, 2) 모양의 양성, 음성 데이터셋을 대상으로 훈련한다.

## 검증 세트 활용

훈련된 모델이 완전히 새로운 데이터에 대해 예측을 잘하는지 여부를 판단하려면 전체 데이터셋을 훈련 세트와 **검증 세트**로 구분해야 함.

- 훈련 세트: 모델 훈련에 사용되는 데이터셋
- 검증 세트: 훈련된 모델 평가에 사용되는 데이터셋

```
model.fit(  
    training_inputs,  
    training_targets,  
    epochs=5,  
    batch_size=16,  
    validation_data=(val_inputs, val_targets)  
)
```