# 0. 강사 및 강의 소개

## 강사 소개

### 기본 사항

김민준 | 1993년 11월 22일 | 대구 출생 | 이대역 근처 거주

서울대학교 자유전공학부 오랜 기간 동안 휴학 중 | 사회복무요원 복무 중

010-5511-4898 문자 전화 카톡 환영 | lakiu@naver.com

### 이력 및 수상

대구 대륜고등학교 졸업 | 수리 가형 + 사회탐구 응시

EBS 공부의왕도 출연 | 청소년을 위한 만만한 경제학 저술

TESAT 최우수상 | 매경TEST 대상 | 경제학원론(조순 外) 10판 개정작업 참여

삼성전자 C-Lab 인턴십 - VR 관련 기술 프로젝트 참여

인디게임 Bytes of Hexagon 기획 & 개발

색 조합 추천 딥러닝 "Daltonism" 개발

영어교육 웹 솔루션 스타트업 "Flowenglish" CTO

용산구 전공연구반 (2013년~) | 성심여고 방과후학교 (2012년~)

세종시 전공연구반 (2017년~)

### 관심과 취향

딥러닝, 데이터과학, 게임제작, 경제학, 창업, 특이점, 인공지능, VR, 3D 프린터

책 모으기, 게임하기, 오늘 할 일 미루기, 저축 안 하고 이거저거 막 지르기

호: 고래, 모찌, 부대찌개 | 불호: 카카오프렌즈

## 강의 목표

python을 통해 프로그래밍을 익히고 기초적인 데이터 처리를 수행할 수 있다.

딥러닝의 이론적 배경과 개념에 대해 익히고 각 딥러닝 모형의 특성을 이해한다.

Keras, Tensorflow 등을 이용해 딥러닝 모형을 설계하고 학습시킬 수 있다.

## 강의 구성

### 컴퓨터 구조와 python 프로그래밍

딥러닝을 활용하기에 앞서 가장 기본이 되는 프로그래밍 능력을 갖추기 위해 컴퓨터 구조에 대한 이해와 python 프로그래밍 언어에 대한 기초 지식을 갖춘다.

컴퓨터 구조는 각 부품의 쓰임과 역할, 딥러닝용 컴퓨터를 위한 조립 견적 작성 등 하드웨어 부분에 대한 지식과 데이터의 이동과 연산 과정, 특히 딥러닝에서 주로 쓰이는 병렬처리, GPGPU (General Purpose computing on Graphics Processing Units)의 세밀한 부분에 대해서 탐구할 것이다.

프로그래밍 언어는 본 강의의 목적상 모든 것을 배우고 직접 활용해보는 대신, 주어진 예제 코드의 흐름을 이해하고 필요한 부분에 대해 수정하는 등의 제한적인 목적을 달성할 정도의 수준을 학습할 것이다.

### 딥러닝

딥러닝 관련 이론 및 개념을 강의자료 및 동영상 자료 등을 통해 학습한다. 이후 Keras 혹은 Tensorflow 등의 딥러닝 프레임워크로 구현한 예제 프로젝트를 실습하며 배운 것을 활용하고 딥러닝 모형의 성능을 개선하는 활동을 한다.

GPU 연산을 위해 Amazon 등의 클라우드 VM을 사용할 것이며, 웹 브라우저를 통해 접속해 개인 노트북으로 수업 및 실습을 진행할 것이다.

# 일정

강사 소개 및 강의 OT

4차 산업혁명 시대, 왜 딥러닝인가?

머신러닝과 딥러닝

컴퓨터 구조와 프로그래밍

---

기초 python 프로그래밍

딥러닝 개발환경 구축 및 활용

---

응용 python 프로그래밍

내 생애 첫 딥러닝

---

CNN (Convolutional Neural Network)

이미지 데이터 학습

---

RNN (Recurrent Neural Network)

순서가 있는 데이터 학습

---

GAN (Generative Adversarial Network)

딥러닝을 이용한 이미지 생성

# 유의사항

## 강의의 목적 및 한계

본 강의는 고등학생을 대상으로 한 6차시 강의라는 점에서 딥러닝을 본격적으로 다루기에는 많은 한계를 가지고 있다. 따라서 강의의 목적은 수강생 스스로가 강의내용을 바탕으로 직접 데이터 수집 및 가공, 목표 설정 및 모형 구축, 학습 및 활용에 이르는 딥러닝 활용의 전 과정을 완벽하게 수행하는 것을 목표로 하지 않는다. 대신 주어진 프로젝트 혹은 참고로 할 수 있는 코드가 주어진 상황에서, 필요에 따라 기능을 수정하고 보강을 하며 원하는 목표를 달성할 수 있는 능력을 갖출 수 있는 것을 목표로 하였다.

만약 이러한 제한적인 수준의 학습을 넘어 전문적인 딥러닝 활용을 원할 경우, 우선 대부분의 딥러닝 프레임워크가 사용하는 python 언어에 대한 중급 수준의 이해가 선행되어야 한다. 또한 이와 관련해 python 가상환경 관리 프로그램인 Anaconda, 데이터 및 그래프 표시를 위한 matplotlib, 행렬 및 벡터 연산 등을 위한 numpy 등을 다루는 법을 익혀야 한다. 이후 개인의 선택에 따라 Tensorflow, PyTorch, CNTK 등의 딥러닝 프레임워크를 택한 뒤 각각에 관한 튜토리얼과 문서 등을 보며 기본적인 딥러닝 모형을 구축하고 학습하는 과정을 거쳐 익힌다. 이 과정에서 쉬운 학습과 빠른 개발을 위해 Keras를 사용할 수 있다.

한편, 딥러닝에는 많은 데이터가 필요하므로, 이러한 데이터를 얻고 가공할 방법에 대해 배우고 자신의 분야에 대한 지식 (Domain Knowledge)를 쌓을 필요가 있다. 또한 높은 컴퓨팅 파워를 활용하는 것이 요구되므로, 가능하면 직접 쓸 수 있는 컴퓨터에 NVIDIA 사의 그래픽카드가 장착되어 있어야 하며 여의치 않은 경우 Amazon Web Service나 Azure 혹은 딥러닝 전용 클라우드 서비스를 이용해야 한다.

## 강의자료

본 강의의 목적상 딥러닝의 수학적 배경과 같은 깊은 이론적 지식보다는, 실습을 위한 실용적 지식에 초점을 두므로 강의자료는 긴 글과 수식을 첨부하는 대신 간략한 개괄 및 중요내용으로 채울 것이며, 필요할 경우 코드나 동영상, 참고 자료 등의 링크를 첨부할 수 있다.

## 실습

강의의 대부분은 짧은 이론적 설명 후, 강사의 시범과 함께 수강생이 따라하는 실습으로 진행될 것이다. 실습 환경은 클라우드를 활용할 것이므로 웹 브라우저를 무리없이 쓸 수 있는 정도 사양의 개인 노트북이면 충분하다. 만약 추가 실습을 원할 경우 강사와 협의 후 클라우드 인스턴스를 할당 받아 이용할 수 있다.

# 1. 4차 산업혁명 시대, 왜 딥러닝인가?

#4차산업혁명 #자동화 #인공지능 #딥러닝
#문송합니다 #코딩과영어 #뉴칼라 #새로운기득권

## 4차 산업혁명

### 1차 - 증기, 산업화, 대량생산

인간과 가축의 노동이 아닌 증기력을 사용한 생산

연료를 쉽게 구할 수 있는 입지와 대량생산과 판매를 위한 시장

노동력의 이동, 자본가와 노동자의 분리

### 2차 - 전기, 모터, 전구, 가전제품

거대하고 다루기 어려운 증기기관을 전기가 대체

이전에 없던 새로운 제품과 시장의 형성

컨베이어 벨트와 제조공정 혁신 – 높은 생산성

### 3차 - 컴퓨터, 자동화, IT

컴퓨터의 발명과 보급

IT 산업의 등장

관리와 생산의 자동화

### 4차 - 데이터, 초연결성, 초지능성

데이터의 수집과 처리, 활용

인공지능과 빅데이터

IoT, 예측, 학습, 추천

인간과 기계의 협동, 가상현실, 데이터화

가상화폐, 로봇, 드론, 3D 프린터, 무인/전기 자동차 등

## 딥러닝에 주목하는 이유

### 생물학적 지능과 같은 원리

딥러닝은 여러 층이 쌓인 네트워크에 수 많은 노드가 결합되고, 각 노드가 연결된 노드들에서 입력받은 값을 가중치에 따라 더하고, 이를 특정한 규칙에 따라 (Activation Function) 출력으로 변환해 내보내는 방식으로 연산을 한다. 학습도 이들 노드의 가중치를 에러를 줄이는 방향으로 점차 변화시키며 진행된다.

인간의 뇌와 같은 생물학적 지능도 마찬가지 방식으로 작동한다. 컴퓨터 속, 프로그램 안의 코드가 아닌 뉴런과 시냅스로 구성된 생물학적 기관의 일부라는 점을 제외하면 그 작동 방식은 같다. 물론 현재 딥러닝의 수준은 아직 인간의 뇌 활동에 비할 것은 아니나, 쥐의 수준은 뛰어넘은 지 오래 됐다. 기하급수적인 발전 속도에 비추어 볼 때, 레이 커즈와일을 비롯한 특이점주의자들과, 일론 머스크 같은 인공지능의 급진적인 발전을 경계하는 이들은 2045년 내외로 생물학적 지능의 총합보다 인공지능의 총합이 더 큰 기술적 특이점이 도래할 것으로 예측하고 있다.

### 데이터는 있지만 규칙이 없는 학습

기존의 인공지능은 대개 개발자가 직접 규칙을 집어넣고, 해당 규칙과 관련된 수치 등을 학습을 통해 조정하는 방식으로 구현되었다. 따라서 학습을 통해 초기보다 개선될 수는 있으나, 개발자의 구상안을 뛰어넘는 학습은 사실상 불가능하였다. 이로 인해 성능이 제한적이었고 개발 비용도 매우 높았다.

그러나 딥러닝은 충분한 데이터와 컴퓨팅 파워가 있으면 전혀 규칙이 없는 상태에서도 직관적으로 데이터의 특징을 파악하고 학습을 진행한다. 인간이 전혀 생각 못 한 수를 알파고가 두는 것도 이러한 딥러닝의 특성에 기인한다.

특히 IoT, 빅데이터 등의 발전과 함께 이용할 수 있는 데이터의 양이 폭발적으로 늘어난 것은 딥러닝에게 날개를 달아준 격이 된다.

# “문송합니다”

## 문과라서 죄송합니다

인문학적, 사회과학적 지식과 소양은 분명 인생에서도 사업에서도 정치에서도 중요하다. 그러나 많은 문과생들은 자신이 그러한 지식과 소양을 갖췄거나 갖출 것이라는 잘못된 예측을 범한다.

갖췄다고 한들 그런 요소들이 21세기를 살아가는 이들에게 그 자체로 도움을 주지는 못 한다. 돈이 있든지, 기술이 있어 돈을 벌 수 있는 방법이 있든지, 그렇지 않으면 그것은 죽은 지식이다.

사업가 입장이 되어서 고용을 한다고 하자. 내 사업에 어떤 사람이 필요할까? 어떤 능력을 가진 사람이 있으면 경쟁에서 우위를 점할 수 있을까? 경우에 따라 다르겠지만 가장 일반적이고 공통적인 답은 우리 회사가 필요로 하는 업무에 대한 전문적인 능력을 갖춘 사람이 필요하다는 것이다. 이제 그러한 전문적인 능력의 종류에 대해 열거해보자.

회계, 법률을 제외하고 문과의 전문지식이 필요하다고 할 능력이 있는가? 심리학을 배운다고 소비자의 심리를 잘 파악할 수 있는 것이 아닌 것처럼, 얼핏 보기에는 문과 전공에 해당하는 듯한 업무도 사실은 문과가 배우는 것과는 큰 연관이 없고 어차피 일을 하며 다시 배워야 하는 것들이 대부분이다.

반면 프로그래밍, IT, 공학, 과학 등의 능력은 회사에서 배워야 할 부분도 있지만 그 전에 기본적인 공학적, 과학적 소양을 대학 등에서 갖춘 사람만이 배울 수 있는 것이다. 그리고 이런 사람들이 회계나 법률, 심리 등을 비롯한 문과적 전문지식을 갖출 수는 있지만 그 반대는 현실적으로 불가능하다.

문과와 이과라는 것보다는 기술이 없는 사람과 기술이 있는 사람으로 구분하는 것이 실정에 맞다. 문과 출신이어도 기술이 있다면, 그는 더 이상 문송하지 않다. 스티브 잡스도 인문학적 소양과 함께 기술에 대한 이해가 있었고 워즈니악이란 공돌이 친구가 있었기에 문송하지 않다. 사실 문과라는 간판이 기술이 없는 사람이라는 적나라한 처지를 가려주는 것이 아닐까?

# New Collar

## Ginni Rometty (IBM CEO)

"It is a partnership between man and machine, if you want to put it that way,"

"Think more about activities changing with the technologies. When you do your job, there will be things that take you a lot of time to research and do,"

"Yes, they'll be done faster. Then you have the time to do what I think we all humans do best."

"We've seen it in the past, whether when people come off of doing farming, they had to learn to read. The industrial area, it was mechanical skills,"

"If we would change the basis and align what is taught in school with what is needed with business ... that's where I came up with this idea of 'new collar.' Not blue collar or white collar,"

## Data Literacy

글을 못 읽는 사람을 문맹이라고 하며, 글을 읽고 쓰는 능력을 문해능력(Literacy)라고 한다. 21세기에는 모든 가치 창출 과정에서 데이터가 중심이 된다. 그리고 인간의 일도 데이터를 다루고 처리하는 것이 된다. 이러한 시대에 데이터를 보고, 이해하고 판단하는 능력 즉 Data Literacy가 없다는 것은 문맹과 다를 바가 없다.

# 간단한 3단 논법

1. 모든 기업은 IT 기업이다 ( )
2. 모든 IT 기업은 인공지능 기업이다. ( )
3. 모든 기업은 인공지능 기업이다. ( )

# 2. 머신러닝과 딥러닝

#머신러닝 #데이터마이닝 #딥러닝
#ANN #CNN #RNN #강화학습 #GPGPU

## 머신러닝이란?

기존의 프로그램과 달리, 컴퓨터가 스스로 주어진 문제로부터 학습하고 주어진 데이터에 대한 평가와 새롭게 들어올 데이터를 처리할 수 있는 능력을 갖추게 하는 것을 의미한다.

데이터에서 의미 있는 규칙과 패턴을 찾는 데이터마이닝과 비슷해 보일 수도 있고, 실제로 기계학습의 일부 과정에 데이터마이닝이 쓰이기도 하나 기계학습은 데이터의 검증과 예측이 목표란 점이 다르다.

### 머신러닝이 각광받는 이유

데이터는 머신러닝을 위한 자원이다. 기계학습의 본질은 결국 데이터의 분류와 검증, 예측이기 때문이다. 좋은 데이터가 많을수록 심화된 학습이 가능하며, 이로부터 더 정확하고 세밀한 처리 능력이 개발된다.

한편, IT혁명 이후로 웹, 스마트폰, SNS, 사물인터넷의 발달과 확산은 무수한 데이터의 생산과 수집이 가능해졌다. 본래 이러한 데이터는 정리나 분류가 되어있지 않아 쓰레기에 가까웠으나 데이터마이닝, 빅데이터 기술의 발달로 데이터에서 의미 있는 정보를 빠르게 추출할 수 있게 됐다. 이용할 수 있는 데이터의 양과 질의 획기적 개선과 GPGPU를 비롯한 컴퓨터 성능의 발전은 머신러닝의 잠재력을 폭발시켰으며, 수많은 기업과 사회가 머신러닝에 주목하는 이유가 됐다.

## 딥러닝이란?

기계학습의 한 분야로, 사람의 뇌와 같은 뉴런 신경망을 인공적으로 구현한 ANN(Artificial Neural Network)를 기반으로 한 알고리즘들을 통칭한다. 다른 방법과 달리 높은 추상화가 특징이며, 사람이 직접 규칙을 지정하지 않아도 네트워크 스스로 문제에 대한 해결법을 구축하는 듯한 모습이 장점이자, 그 내부 구조를 개발자도 정확히 파악하기 어렵다는 점에서 단점이 되기도 한다.

### 딥러닝이 각광받는 이유

딥러닝의 원류인 인공신경망은 학계에서 사장됐던 분야이다. 학습이 매우 느리고 부정확했으며, 사전에 규칙을 넣어주지 않아도 되는 만큼 더 많은 데이터가 필요했기 때문이다. 그러나 학습 알고리즘의 개선과 함께 컴퓨터의 발달, 특히 GPU의 발달로 인해 학습에 걸리는 시간이 획기적으로 줄어들었으며, 빅데이터의 등장은 바로 이러한 딥러닝에 필요한 무지막지한 양의 데이터를 마련할 수 있게 하였다. 내외적으로 모든 조건이 딥러닝에게 최고의 환경을 조성한 것이다.

또한 딥러닝을 위한 여러 기법이 개발되고 연구되며 딥러닝의 잠재력 또한 더 커졌다. 초기 딥러닝은 숫자 몇 개를 입력으로 받아 역시 숫자로 된 출력을 내놓는 정도였으나, CNN(Convolution Neural Network)를 활용한 이미지 분석, RNN(Recurrent Neural Network)를 활용한 시계열, 자연어 분석 등을 통해 더 많은 데이터와 문제에 딥러닝을 적용할 수 있게 됐다. 나아가 알파고와 같은 강화학습(Reinforcement Learning)은 게임이나 운전과 같은 분야에 딥러닝을 사용할 수 있는 가능성을 열었다.

## GPGPU

General Purpose computing on Graphics Processing Units의 약자로, 일반적으로 2D, 3D 이미지를 처리하기 위한 장치인 GPU를 수치연산 등의 다른 프로그램의 계산에 활용하는 것을 말한다. CPU는 소수의 천재가 있는 방이라면, GPU는 수천의 평범이들이 있는 방이라 볼 수 있으며, 단순하고 병렬화하기 쉬운 연산에 대해선 CPU 대비 수 백, 수 천 배의 속도를 낼 수 있다.

간단한 머신러닝의 경우 일반적인 CPU로도 충분하나, 딥러닝의 경우 매우 간단한 경우를 제외하면 GPU의 사용이 필수적이다. 대부분 NVIDA의 그래픽카드를 사용하며, Tesla, Titan X(p), GTX 1080 (Ti) 등 최고사양의 제품이 딥러닝 연구, 개발을 위한 컴퓨터에 장착되고 있다. 또한 AWS, Azure 등 클라우드 서비스에서도 GPU가 장착된 서버를 합리적인 가격에 대여할 수 있다.

# 3. 컴퓨터 구조와 프로그래밍

#CPU #RAM #GPU #OS
#프로그래밍 #Python #HPC #CUDA

## 컴퓨터 하드웨어

### CPU (Central Processing Unit)

CPU는 컴퓨터에서 가장 핵심적인 연산과 논리처리를 담당한다. 저장장치에서 데이터와 명령어를 불러온 뒤, 주어진 명령어에 해당하는 처리를 수행한다. 현대의 CPU는 이 처리에 매우 작은 시간만이 소요되며, CPU 내부에 작업자가 하나가 있는 싱글 코어가 아닌 멀티 코어도 보급이 이루어졌다.

### RAM (Random Access Memory)

램은 주기억장치로, 필요한 명령어와 데이터를 빠르게 넣고 뺄 수 있는 일종의 작업공간이다. CPU 내부에도 매우 빠른 저장공간이 있으나, 용량의 한계가 있어 램이 필요하다. 램이 커야만 대용량의 데이터를 다루고 무거운 프로그램을 구동할 수 있다. 일반 용도는 8기가면 충분하나, 16기가 이상도 필요한 경우가 있다.

### GPU (Graphic Processing Unit)

본래 게임이나 영상, 3D 모델링 등에서 사용되는 부품이었으나, 현대에는 다양한 용도로 쓰이고 있다. 특히, CPU가 코어 1개, 2개에서 많으면 8개, 서버 용으로도 32개 정도임에 비해 GPU는 그에 비해 작고 간단한 코어지만 그러한 코어가 1,000개, 2,000개 정도로 많아 병렬처리에 매우 큰 효율을 보인다.

## 운영체제 (OS, Operating System)

컴퓨터는 하드웨어만으로 작동하지 않으며, 반드시 운영체제가 설치되어야 제 기능을 할 수 있다. 운영체제는 우리가 흔히 말하는 프로그램과, 그걸 이용하는 사용자를 하드웨어와 이어주는 역할을 한다. 사람의 요구에 맞게 하드웨어의 성능과 처리 우선순위를 조정하는 것이 운영체제의 역할인 것이다. 주로 Windows를 많이 쓰지만, 서버나 고속 연산 등에서는 주로 Linux를 사용하는 경우가 많다. 최근에는 윈도에서도 딥러닝 환경이 잘 갖춰져 큰 불편함이 없다.

## 프로그래밍

내 의도와 목적, 그리고 그것을 이루는 방법을 컴퓨터가 이해할 수 있는 방식으로 전달하는 것이 프로그래밍의 본질이다. 따라서 논리적이고 기계에 적합하게 문제의 해결과정을 체계화하여 이를 프로그래밍 언어로 표현하는 것이다.

엑셀을 잘 쓰고, 인터넷을 잘 찾고, 포토샵 등을 잘 다루는 것은 외국어로 따지면 글을 잘 읽고 잘 듣는 것에 해당한다. 그러나 말하기, 쓰기와 같이 내 의도를 컴퓨터에게 자유롭게 전달하고 업무의 효율을 극대화하기 위해서는 단순한 컴퓨터 활용능력이 아닌 프로그래밍 능력이 필요하다.

### Python

1989년 귀도 판 로썸(Guido van Rossum)이 크리스마스에 연구실에서 다른 사람도 없고 심심해서 만들었다. 현대에는 간단한 실용 프로그래밍은 물론 웹사이트, 서버, 게임 등을 비롯해 연구용 코드, 머신러닝, 딥러닝에 이르기까지 광범위하게 사용되며 특히 딥러닝 프레임워크는 대다수가 python에 기반한다.

Python은 배우기 쉽다는 것이 강점이다. 직관적인 명령어와 문법으로 인해 마치 영어로 설명문을 적는 것과 비슷한 느낌으로 기초적인 코딩을 할 수 있으며, 이를 통해 빠른 학습과 빠른 개발이 가능해 효율적이다. 또한 고급 기능의 경우 다양한 라이브러리를 가져와 활용할 수 있어 강력하며, 다양한 분야에서 쉽게 응용하여 사용할 수 있다.

### 성능 극대화

Python은 원본 자체는 프로그래밍 언어 중에서 상당히 느린 편에 속한다. 현대 컴퓨터가 충분히 빠르므로 일반적인 용도에서는 체감할 수 없지만, 딥러닝 등의 연산에서는 치명적이다. 그러나 이를 극복하기 위해 내부적으로는 C언어 등의 고속 언어를 사용한 라이브러리를 쓰고, 여기서 벡터 연산, 병렬 처리 등을 활용하고 CUDA를 통해 GPU도 활용할 수 있어 성능을 극대화하는 것이 가능하다.

# 4. 딥러닝의 구조와 설계

## 딥러닝의 설계와 학습 과정

### 목표 설정

딥러닝을 통해 성취할 목표를 설정해야 한다. 목표를 달성했는 지의 여부를 판단할 기준지표도 이 단계에서 같이 설정한다.

### 입력과 출력 정의

딥러닝에 주어질 입력과 출력의 종류와 형태를 정의한다. 딥러닝의 전체적인 구조와 성능에 큰 영향을 미치기 때문에, 사용가능한 데이터와 목표를 함께 고려하며 신중하게 선택할 필요가 있다.

### 데이터 준비

사용할 데이터를 준비한다. 앞서 정의한 입력 형태에 맞게 데이터를 가공할 필요가 있다. 너무 큰 이미지는 막대한 성능을 요구하므로 리사이징을 하거나, 컬러를 흑백으로 바꾸는 등 딥러닝에 더 적합하도록 데이터의 형식을 바꾸어야 할 때도 있다. 또한 언어 데이터의 경우 단어장을 만들어 각 단어가 하나의 숫자에 대응하도록 하거나 Word2Vec 등의 기존 솔루션을 활용한다.

### 학습 데이터와 검증 데이터의 분리

딥러닝의 특성상, 학습이 오랜 기간 진행되면 실제 딥러닝의 성능과 상관없이 학습에 사용된 데이터에 한해서는 높은 성취도를 보일 수 있다. 따라서 학습에 사용할 데이터와 검증에 사용할 데이터를 분리해야 만 정확하고 객관적인 검증이 가능하다. 편향을 막기 위해 전체 데이터를 임의로 추출해 검증용으로 쓸 수도 있으며, 목적에 따라 학습과 검증 데이터가 아예 다른 종류가 될 수도 있다.

### 딥러닝 레이어 설계

입력부터 출력 레이어에 이르기까지 딥러닝의 구조를 설계한다. 사용하는 데이터의 종류와, 목적에 따라서 적합한 구조가 달라지기 때문에 이들의 특성을 잘 알고 조합할 필요가 있다. 이론적으로 더 깊고 넓은 네트워크일수록 더 복잡한 일을 수행할 수 있으나, 반대급부로 학습에 더 오랜 시간이 걸리고, 더 많은 데이터와 더 빠른 하드웨어를 요구한다. 따라서 목적을 달성할 수 있는 네트워크의 적절한 규모가 어느 정도인지를 생각하며 설계해야 한다.

### 학습

딥러닝 라이브러리를 통해 설계한 네트워크를 학습시킨다. 학습 방법과 학습 횟수 등의 변수에 따라 학습 시간과 효율이 크게 달라지기 때문에 이 또한 데이터와 네트워크에 맞게 조정할 필요가 있다. 가능한 경우 GPGPU를 지원하는 하드웨어와 라이브러리를 사용해야 하며, AWS 등 Cloud 서비스를 이용할 수도 있다.

### 재설계

학습 결과를 토대로 딥러닝 네트워크를 평가하고, 부족한 부분을 보완할 수 있도록 재설계한다. 특히 딥러닝 네트워크의 성능은 대개의 경우 실제 학습을 거치기 전엔 예측하기 어려운 경우가 많기 때문에 많은 시행착오가 필요하다. 시간과 비용을 절약하기 위해 이러한 시행착오 중에는 전체 데이터 중 일부만을 사용하고, 의미 있는 성과가 나오면 전체 데이터를 이용한 학습을 진행할 수도 있다.

### 검증

학습 데이터로 학습한 결과를 검증 데이터를 통해 검증한다. 목표 설정 단계에서 정의한 기준 지표를 통해 딥러닝의 성취도를 평가한다. 또한 오류나 미흡한 특성을 보이는 데이터의 특징을 파악해 딥러닝의 한계를 파악한다.

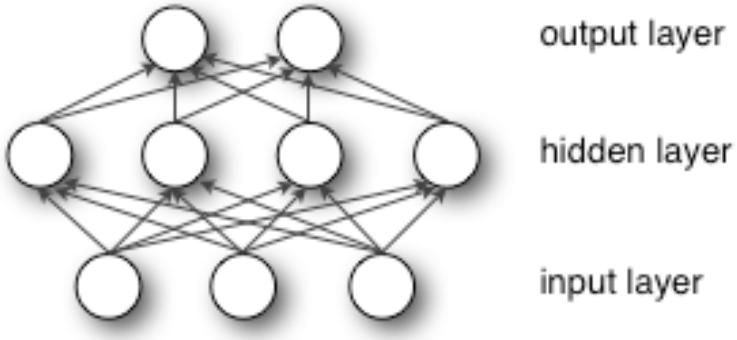
### 실사

학습한 딥러닝 네트워크를 통해 의미 있는 정보를 추출하거나, 별도의 서버나 하드웨어 등에 배치한다. 딥러닝은 학습은 높은 성능을 요구하지만 비해 사용에는 비교적 낮은 성능의 하드웨어로도 충분하므로, 서비스의 용도에 따라 다양한 하드웨어를 이용할 수 있다.

# 딥러닝 레이어

딥러닝의 구조는 딥러닝의 입력과 출력 사이에 어떤 레이어들이 쌓여 있는지를 의미한다. 각 레이어마다 다양한 특징을 가지고 있으며. 이를 잘 조합해야 데이터와 목적에 맞는 딥러닝을 설계할 수 있다.

## 일반적인 NN(Neural Network) 레이어 / Dense




바로 위의 네트워크의 노드(그림에서 동그라미)를 입력으로 받아 출력으로 지정된 개수만큼의 노드를 내는 레이어. 입력과 출력 노드 모두가 연결되어 있다. 이러한 NN을 많이 쌓아 DNN(Deep Neural Network)를 만들 수 있으며, 다른 레이어들과 조합하기도 용이하다.

가장 기본적인 레이어로, 분류, 변환, 회귀 등 이론적으로 충분한 수가 있으면 모든 함수를 따라할 수 있다. 그러나 계산의 효율성과 학습 측면에서의 한계도 비교적 명확하다. 따라서 다른 레이어들과 결합하여 사용되는 경우가 많다.

## Activation

딥러닝 네트워크의 각 노드는 자기에게 들어오는 화살표, 즉 입력값을 각 입력에 대한 가중치를 곱한 뒤 모두 더해 계산한다. 이후 출력값을 어떻게 내보낼지를 정하는 것이 Activation 함수이다. 수치를 예측하기 위한 회귀분석이 목적이면 Linear를 쓰지만 대개의 경우 0근방에서 값이 변화가 큰 비선형적인 함수를 많이 쓰며, 최근에는 Linear에서 음수의 경우 0으로 만들어버리는 ReLU (Rectified Linear Unit)이 각광을 받고 있다.

## Recurrent Neural Network (RNN)

일반적인 NN은 주어진 입력을 판단할 때 그 입력만을 고려해서 출력을 낸다. 다시 말해 맥락에 따른 판단이 불가능한 것이다. RNN은 이를 해결하기 위해 네트워크가 메모리를 가지고 이용하며, 입력한 내용이 한 번 쓰이고 끝나는 대신 다시 되새김질 되듯 네트워크에 재투입된다. 가령 1, 2, 3, 4, 3, 2, 1, 2, 3, … 으로 가는 수열을 예측할 때, 직전 숫자만 입력 받아 다음 숫자를 예측하는 것은 불가능하다. 2 다음에 3이 올 수도 있고 1이 올 수도 있기 때문이다. RNN은 2 이전의 4, 3를 같이 입력 받아 출력을 결정하기 때문에, 입력이 현재 하락하는 중이란 것을 인식할 수 있고, 정확한 예측을 할 수 있다.

따라서 이러한 맥락이 중요한 텍스트 분석이나 주가 분석, 필기체 인식 등에서 높은 성능을 보인다. 그러나 같은 크기의 다른 네트워크보다 더 많은 매개변수를 가지고 있어 학습에 시간이 오래 걸리고 난이도가 높다.

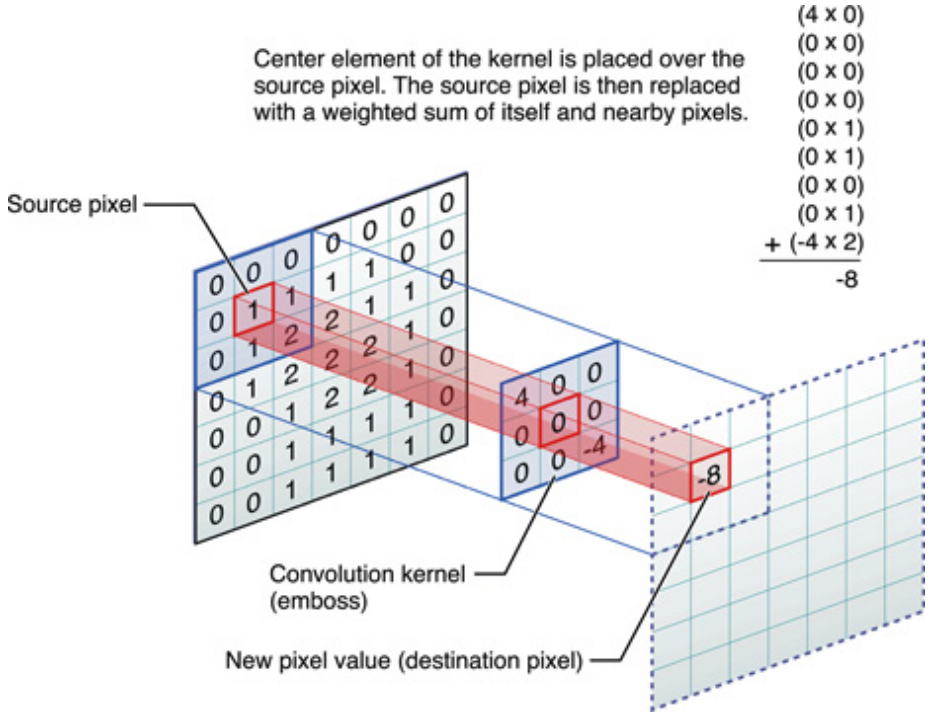
## LSTM (Long Short Term Memory)

모든 딥러닝은 네트워크의 출력과 정답(목표값)을 비교하여, 정답에 가까워질 수 있는 방향으로 각 레이어의 계수를 조정하는 것을 통해 학습을 한다. 이 때 이 방향은 미분을 통해 기울기로 계산되며, 이 기울기에 일정한 학습률 (Learning Rate)를 곱하고 입력의 크기만큼 가중치를 넣어 조금씩 계수를 변화시킨다.

문제는 이 변화의 방향 혹은 가야할 방향의 경사 (Gradient)를 계산할 때, RNN의 경우 네트워크가 여러 입력에 걸쳐 출력을 계산하기 때문에 출력에서 먼 부분의 입력에 대해서는 Gradient가 점점 감소하여 0이 되거나, 혹은 점점 커지면서 무한대로 발산하는 문제가 발생한다. 쉽게 말하면 글의 내용을 인식할 때, 글 앞부분의 내용에 대해서는 학습의 방향이 제대로 전달되지 않는 것이다.

이를 극복하기 위해 뉴런 내부에 이전 뉴런의 상태를 전달할 지 말지 정하는 관문을 설치하고 이 관문의 작동도 학습시키는 LSTM이 등장하였다. 사람의 기억을 딥러닝에 구현한 것이다. 특히 LSTM은 문장의 구조적인 요소를 학습시키는데 효과가 크다. (괄호를 열고 적절하게 다시 닫아주는 등 기존의 RNN으로는 힘들었던 것을 잘 할 수 있다)

## Convolution Neural Network (CNN)




컨볼루션 레이어는 특히 이미지를 인식하는 것에 큰 강점을 가지며, 사진 및 영상 데이터를 처리할 때 필수적으로 사용된다. 컨볼루션 레이어는 이미지의 각 좌표마다, 특정한 크기만큼 점과 주변 점들을 뽑아내고, 그 값에 필터를 적용한 값을 다시 내보내는 것을 통해 이미지의 다양한 특징을 추출한다.

가령 수직선을 강조하기 위해서는 다음과 같은 필터를 사용한다.

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

만약 주변 픽셀이 다 같은 색이면, 한 쪽은 1, 한 쪽은 −1이게 상쇄되어 0이 나타난다. 그러나 왼쪽은 검은색(값이 0), 오른쪽은 흰색(값이 1)이라고 하자. 그러면 컨볼루션한 값은 (−1) * 0 + (−1) * 0 + (−1) * 0 + 1 * 1 + 1 * 1 + 1 * 1 = 3이며, 반대로 왼쪽이 흰색, 오른쪽이 검은색일 경우 (−1) * 1 + … + 1 * 0 + … = −3으로 수직경계선이 아닌 곳에서는 0에 가까운 값이, 수직경계선에서는 절대값이 큰 값이 나타나 구분이 가능하다.

# 이미지 분류에서의 사례

## 입력

입력받을 데이터의 크기를 정해야 한다. 가령 (128, 128, 3)의 크기라면 가로 128, 세로 128 해상도의 RGB 이미지를 입력으로 받는 것이 된다. 만약 준비된 데이터가 크기가 다른 경우, 리사이징하거나 비율이 안 맞는 부분을 흰색이나 노이즈로 채울 필요가 있다.

## 컨볼루션

필요한 만큼 컨볼루션 레이어를 쌓는다. 컨볼루션 레이어에서 중요한 것은 필터의 크기다. 필터가 클수록 한 점의 값을 더 많은 점을 참고해 계산할 수 있어 더 많은 정보를 참고할 수 있으나, 그만큼 더 긴 계산 시간이 필요하고, 학습에도 오랜 시간이 걸린다.

이를 해결하기 위해 Pooling 레이어를 추가할 수도 있다. Pooling 레이어는 이미지를 특정한 크기의 칸으로 나누고, 칸 안에서 Max값을 뽑아내거나 Mean값을 뽑아낸다. 만약 4 * 4 이미지에 크기가 2*2인 Pooling을 하면 총 4개 칸으로 나눠지고, 원래 4 *4 = 16에서 4개로 다루는 숫자가 줄어들면서 원래 이미지의 중요한 특징을 유지할 수 있다.

## 출력

컨볼루션과 풀링 이후, 최종적으로 결과를 내기 위해 NN를 마지막에 추가한다. 이 때, NN의 출력 개수는 분류의 가지수와 같아야 한다. 즉 개와 고양이, 금붕어를 구분하는 딥러닝이라면 마지막 출력의 개수가 3이어야 하는 것이다. 가령 출력이 (0.2, 0.3, 0.5)로 나왔을 경우 가장 높은 값인 0.5가 이 딥러닝의 예측이며, 이것은 금붕어라는 것을 의미한다.

## 학습 및 검증

전체 이미지 셋에서 학습용 데이터와 검증용 데이터를 분리한 뒤, 충분한 시간을 들여 학습한 뒤 정확도를 평가한다.