

# [AI Tech Letter]

AI

Exported on 04/22/2022

## Table of Contents

<b>1</b>	<b>[1편] 인공지능 첫걸음 .....</b>	<b>8</b>
1.1	인공지능에 대한 오해와 진실 .....	8
1.1.1	인공지능은 로봇이다?.....	8
1.1.2	인공지능은 스스로 똑똑해질 수 있다?.....	10
1.1.3	인공지능도 감정이 있다?.....	13
1.2	인공지능(Artificial Intelligence;AI)이란.....	14
1.2.1	강 인공지능, 약 인공지능.....	14
1.2.1.1	강 인공지능 (Strong AI, General AI).....	15
1.2.1.2	약 인공지능 (Weak AI, Narrow AI).....	16
1.2.1.3	범용성과 전문성 .....	18
1.2.2	넓은 의미의 AI, 좁은 의미의 AI.....	19
1.2.2.1	특징(feature) 추출과 의사 결정 .....	20
1.3	마무리 .....	21
<b>2</b>	<b>[2편] 기계도 사람처럼 판단할 수 있을까? .....</b>	<b>23</b>
2.1	머신러닝과 딥러닝 .....	23
2.1.1	머신러닝이 다루는 정형 데이터(structured data).....	24
2.1.2	딥러닝이 다루는 비정형 데이터(unstructured data) .....	28
2.2	긍정 리뷰와 부정 리뷰 자동 분류하기 .....	35
2.2.1	특징 및 분류기준 .....	35
2.2.2	예외 CASE .....	36
2.2.3	모라벡의 역설(Moravec's Paradox).....	37
2.2.4	사람처럼 사고하기 .....	39
2.3	인공신경망(Artificial Neural Network) .....	40
2.3.1	인공뉴런(Artificial Neuron).....	40
2.3.2	딥러닝이 유행하게 된 배경.....	41
2.4	마무리 .....	43
<b>3</b>	<b>[3편] 시각을 얻은 인공지능 .....</b>	<b>45</b>
3.1	이미지를 인식하는 인공지능 .....	45
3.1.1	인공신경망 연산 .....	47
3.1.2	기계가 이미지를 인식하는 방법 .....	47
3.2	Convolutional Neural Network(CNN).....	50

3.2.1 특징 추출(Feature Extraction) .....	51
3.2.2 태스크 수행.....	53
3.2.2.1 Classification .....	53
3.2.2.2 Detection .....	54
3.2.2.3 Segmentation.....	55
3.3 ImageNet Large Scale Visual Recognition Competition .....	56
3.4 마무리 .....	58
<b>4 [4편] 언어를 이해하는 인공지능.....</b>	<b>60</b>
4.1 언어를 인식하는 인공지능.....	60
4.1.1 자연어이해(NLU; Natural Language Understanding).....	64
4.1.1.1 자연(Natural) .....	64
4.1.1.2 언어(Language) .....	64
4.1.1.3 이해(Understanding) .....	67
4.2 기계에게 사람의 언어를 인식시키려면? .....	68
4.2.1 Tokenizing(Parsing) .....	68
4.2.2 워드임베딩(word embedding).....	69
4.2.2.1 원-핫 인코딩(One-hot Encoding) .....	69
4.2.2.2 CBOW와 SKIPGRAM .....	70
4.3 다양한 자연어이해 과제들.....	74
4.3.1 문장/문서 분류(Sentence/Document Classification) .....	74
4.3.2 Sequence-to-Sequence .....	75
4.3.3 질의 응답(Question Answering) .....	76
4.4 마무리 .....	77
<b>5 [5편] 과거의 경험을 통해 현재를 배우는 인공지능 .....</b>	<b>79</b>
5.1 시간 흐름에 따른 데이터(Sequential data) 처리하기 .....	79
5.2 Recurrent Neural Network(RNN; 순환 신경망).....	82
5.2.1 장점 .....	83
5.2.1.1 RNN은 시간 흐름에 따른 과거 정보를 누적할 수 있다 .....	83
5.2.1.2 RNN은 가변 길이의 데이터를 처리할 수 있다 .....	83
5.2.1.3 RNN은 다양한 구성의 모델을 만들 수 있다.....	84
5.2.2 단점 .....	85
5.2.2.1 연산 속도가 느리다 .....	85
5.2.2.2 학습이 불안정하다 .....	85
5.2.2.3 실질적으로 과거 정보를 잘 활용할 수 있는 모델이 아니다.....	86

5.2.3	성능 보완 .....	86
5.2.3.1	LSTM(Long-short term memory).....	86
5.3	활용 사례 .....	88
5.4	마무리 .....	89
<b>6</b>	<b>[6편] 헛똑똑이 인공지능 제대로 가르치기 .....</b>	<b>91</b>
6.1	AI Process .....	91
6.1.1	Offline Process .....	92
6.1.2	Online Process.....	94
6.2	오버피팅(Overfitting)과 일반화 성능(Generalization) .....	95
6.2.1	Training, Validation, Test.....	97
6.2.1.1	Training set.....	98
6.2.1.2	Validation set.....	98
6.2.1.3	Test set .....	98
6.2.2	학습 곡선(Learning curve) 확인하기 .....	98
6.3	Regularization .....	100
6.3.1	데이터 증강(Data Augmentation) .....	101
6.3.2	Capacity 줄이기.....	101
6.3.3	조기 종료(Early stopping).....	102
6.3.4	드롭아웃(Dropout).....	102
6.4	마무리 .....	103
<b>7</b>	<b>[7편] 다시쓰고 바꿔쓰자! 인공지능 재활용하기 .....</b>	<b>105</b>
7.1	쉽지 않은 인공지능 적용하기 .....	105
7.1.1	구체적이지 않으며 불명확한 태스크.....	106
7.1.2	적은 데이터, 낮은 품질의 데이터 .....	108
7.1.3	다른 도메인 환경 .....	109
7.2	Transfer Learning : 한번 만든 인공지능 모델 우려먹기 .....	111
7.2.1	Catastrophic forgetting : 치명적인 기억상실! .....	113
7.2.2	더 나은 Transfer Learning을 위한 방법.....	113
7.3	Transfer Learning 모델 이용 .....	115
7.3.1	컴퓨터 비전에서의 Transfer Learning.....	115
7.3.2	자연어 이해(NLU)에서의 Transfer Learning .....	117
7.4	마무리 .....	117
<b>8</b>	<b>[8편] 준비된 인공지능, Pre-trained AI .....</b>	<b>119</b>

8.1	Pre-training: 니가 뭘 요청할지 몰라서 일단 다 공부해놨어 .....	119
8.2	대규모 데이터에 대한 Pre-training .....	121
8.2.1	시각 데이터에 대한 사전학습 .....	121
8.2.2	언어 데이터에 대한 사전학습 .....	123
8.3	Self-Supervised Learning: 나 혼자 어떻게든 해볼게 .....	125
8.3.1	예: 이미지 데이터를 위한 Self-Supervised Learning .....	129
8.3.2	예: 텍스트 데이터를 위한 Self-Supervised Learning .....	131
8.3.3	예: Google BERT(Bidirectional Encoder Representations from Transformers) .....	132
8.4	마무리 .....	133
9	[9편] 족집게 데이터로 인공지능 학습하기 .....	135
9.1	데이터의 바다, 정보의 홍수 .....	135
9.2	Active Learning: 족집게 데이터로 공부하기 .....	137
9.2.1	Active Learning의 절차 .....	138
9.2.2	Query Strategy: 이 데이터를 제게 가르쳐 주십시오! .....	139
9.2.2.1	Uncertainty Sampling .....	140
9.2.2.2	Query by committee .....	141
9.3	마무리 .....	142
10	[10편] 뭣이 중요한지 알아보는 인공지능 .....	144
10.1	긴 입력 데이터 처리하기 .....	144
10.2	어텐션 메커니즘(Attention mechanism) .....	146
10.2.1	어텐션 스코어(Attention score) .....	148
10.2.2	컨텍스트 벡터(Context vector) .....	148
10.3	XAI로서의 어텐션 .....	151
10.3.1	텍스트에서의 어텐션 .....	153
10.3.2	이미지에서의 어텐션 .....	153
10.4	Attention 전성시대, Transformer .....	154
10.5	마무리 .....	156
11	[11편] 스스로 진화하는 인공지능, AutoML .....	157
11.1	사람의 손을 필요로 하는 인공지능 .....	157
11.2	스스로 진화하는 인공지능, AutoML .....	159
11.2.1	하이퍼파라미터 탐색 자동화 .....	160
11.2.2	아키텍처 탐색 자동화 .....	162

11.3 AutoML 특징 .....	162
11.4 AutoML 서비스.....	164
11.5 마무리 .....	166
<b>12 [12편] 설명 가능한 인공지능, XAI .....</b>	<b>168</b>
12.1 종종 이해할 수 없는 결정을 내리는 AI .....	168
12.2 설명 가능한 인공지능, XAI(eXplainable Artificial Intelligence).....	171
12.2.1 XAI의 필요성 .....	171
12.3 XAI를 위한 접근법 .....	172
12.3.1 어텐션 메커니즘(Attention Mechanism)을 활용한 XAI .....	172
12.3.2 설명하는 법 학습하기(Learn to explain) .....	174
12.4 마무리 .....	175
12.5 2020 AI 테크레터 연재를 마무리하며...	176

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판](#)<sup>1</sup>에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀

---

<sup>1</sup> <https://wire.lgcns.com/confluence/display/AI/AI>

# 1 [1편] 인공지능 첫걸음

---

안녕하세요, CTO AI빅데이터연구소입니다.

금일부터 한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리려 합니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI개시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/AI기술팀<sup>2</sup>

---

- 인공지능에 대한 오해와 진실(see page 8)
    - 인공지능은 로봇이다?(see page 8)
    - 인공지능은 스스로 똑똑해질 수 있다?(see page 10)
    - 인공지능도 감정이 있다?(see page 13)
  - 인공지능(Artificial Intelligence;AI)이란(see page 14)
    - 강 인공지능, 약 인공지능(see page 14)
      - 강 인공지능 (Strong AI, General AI)(see page 15)
      - 약 인공지능 (Weak AI, Narrow AI)(see page 16)
      - 범용성과 전문성(see page 18)
    - 넓은 의미의 AI, 좁은 의미의 AI(see page 19)
      - 특징(feature) 추출과 의사 결정(see page 20)
  - 마무리(see page 21)
- 

4차산업혁명의 대표 기술분야인 '인공지능'에 대해 첫 번째 AI테크레터를 연재하겠습니다.

자사에서도 관심을 갖고 인력 육성과 사업 발굴에 많은 투자를 하는 분야이지만, 아직 대부분의 직원분들이 인공지능에 대해 접할 일이 많지는 않았으리라 생각합니다.

인공지능을 처음 접하시는 분들을 위해 개념부터 소개드리겠습니다.

## 1.1 인공지능에 대한 오해와 진실

### 1.1.1 인공지능은 로봇이다?

아마 관련 전공을 가지거나 유사 업무를 진행하지 않으셨다면, 여러분은 매스컴을 통해서 최초로 '인공지능'을 접하게 되셨을 가능성이 큽니다.

영화에서 다루는 인공지능이라든가, 뉴스 기사에서 소개되는 사례나, SNS 소식이나 일러스트를 통해서 말이죠.

한국인이 가장 많이 알고 있는 대표적인 인공지능, '알파고'를 구글에 검색해보았습니다.

<sup>2</sup><http://wire.lgcns.com/confluence/display/~78628>

이미지 검색결과의 첫 페이지 링크 대부분이 뉴스기사와 관련되어있는데요, 뉴스에서 인공지능 기사를 다를 때엔 주로 사람과 비슷한 휴머노이드(humanoid)<sup>3</sup> 로봇의 형태로 인공지능을 다루곤 합니다.



[그림1. '알파고' 구글 이미지 검색 결과<sup>4</sup>]

많은 매체에서 인공지능을 의인화하여 그려내고 있지만, '인공지능=로봇'이란 공식이 항상 성립하지는 않습니다. 인공지능이란 하나의 소프트웨어 프로그램에 지나지 않습니다. 로봇은 그걸 담는 하드웨어 용기일 뿐이지요.

이 프로그램을 담는 것은 휴머노이드 로봇이 될 수도 있고, 누군가의 노트북이 될 수도 있고, 핸드폰이 될 수도 있습니다.

이세돌 9단과 대결하며 마치 못된 적(?)과 같이 그려졌던 알파고도 실제로는 구글이 개발한 프로그램입니다.

팔이 없는 알파고를 대신하여 바둑돌을 놓는 역할은 구글 딥마인드의 '아자 황' 박사가 수행하였습니다.

<sup>3</sup> <https://ko.wikipedia.org/wiki/%ED%9C%B4%EB%A8%B8%EB%85%B8%EC%9D%B4%EB%93%9C>

<sup>4</sup> [https://www.google.com/search?q=%EC%95%8C%ED%8C%8C%EA%B3%A0&lz=1C1GCEU\\_koKR877KR877&sxsrf=ALeKk03LXOrvYsWbUCffTSd7TDXBJdaKew:1594259422321&sou=rce=lnms&tbo=isch&sa=X&ved=2ahUKEwjygbCUh7\\_qAhXoylsBHZeiADQQ\\_AUoAXoECBAQAw](https://www.google.com/search?q=%EC%95%8C%ED%8C%8C%EA%B3%A0&lz=1C1GCEU_koKR877KR877&sxsrf=ALeKk03LXOrvYsWbUCffTSd7TDXBJdaKew:1594259422321&sou=rce=lnms&tbo=isch&sa=X&ved=2ahUKEwjygbCUh7_qAhXoylsBHZeiADQQ_AUoAXoECBAQAw)



[그림2. 이세돌vs알파고 대국 방식, 알파고의 대리인 아자 황 박사 (출처:국민일보)<sup>5)</sup>]

### 1.1.2 인공지능은 스스로 똑똑해질 수 있다?

매체에서 인공지능에 대해 소개할 땐, 데이터가 많이 쌓일 수록 인공지능의 성능이 자동으로 좋아지는 것처럼 묘사 하곤 합니다.

빅데이터 붐 이후로 인공지능이 학습을 할 데이터가 많아지긴 했습니다만, 그렇다고 해서 인공지능이 저절로 혼자 똑똑해질 수는 없습니다.

오늘날 인공지능의 90% 이상은 지도학습(Supervised Learning) 방식으로 훈련됩니다.

지도학습 방식은 사람이 데이터 한 건 한 건을 어떻게 판단하거나 처리하는지에 대한 정답을 인공지능에게 알려주면서 진행하는 방식입니다. (인공지능 학습 방식은 향후 AI테크레이터에서 자세히 다루려 합니다)

이외에도 사람이 인공지능의 구조를 하나씩 설계하고, 다양한 최적화 기법을 적용하는 실험을 해야만 인공지능이 더 나은 성능을 낼 수 있습니다.

<sup>5</sup> <http://news.kmib.co.kr/article/view.asp?arcid=0923457032>

그럼에도 불구하고 아직도 많은 매스컴에서는 인공지능이 데이터만 많아지면 스스로 학습하여 진화할 수 있는 것처럼 다루는 경우가 많습니다.

나아가 인간에게 위협이 될 수 있는 것처럼 소개하기도 하죠. 이를 걱정하는 사람들도 꽤 많아보입니다.

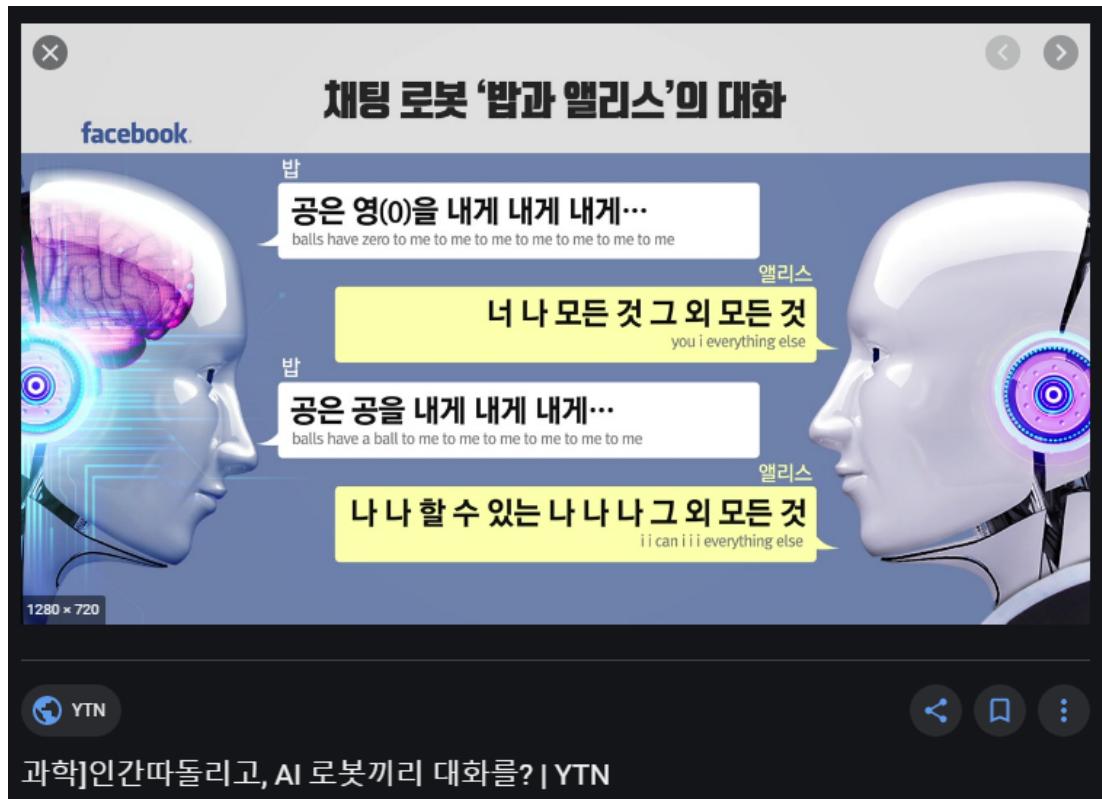
[인공지능이 지구를 지배한다는 예언의...](#) 2020.05.16.  
... 인공지능의 인간을 지배할 수 있는 방법은 인간을 끌살시키거나 인간을 모두잡아들여 인간동물원을 만드는 방법등이 있습니다. 이 예시들은 실제 인공지능들이 말한...  
Q&A > 지구과학 | 답변수 2 · 추천수 0 | 답변 Lumble(myh0\*\*\*\*)

[인공지능이 우리 인간을 지배하게되나요?](#) 2016.03.11.  
언젠가 인공지능이 인간을 지배하는날이오면 어떻하죠? 어제 알파고가 이세돌을 이기면서 페북이나 각종사이트에 인공지능관련 검색어가 떠서 봇는데 너무 무서웠어요...  
Q&A > 전기, 전자 공학 | 답변수 1 · 추천수 18 | 답변 f40k\*\*\*\*

[인공지능이 인간을 지배할 가능성 어느](#) 2018.05.27.  
인공지능이 인간을 지배할 가능성 어느정도예요? 전자공학 전공자인데요 미래는 100% 인공지능 알파고님이 지배할겁니다 마리마리 인공지능님에게 출성을 바칠 준비를...  
Q&A > 물리학 | 답변수 2 · 추천수 0

[그림3. 인공지능 도시괴담 ([출처:네이버 지식인](#)<sup>6</sup>)]

몇 년 전, 인간을 따돌리고 AI끼리 대화를 한다는 뉴스기사를 본 적이 있으신가요?



<sup>6</sup> <https://kin.naver.com/search/list.nhn?query=%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5+%EC%9D%B8%EA%B0%84+%EC%A7%80%EB%B0%B0>

[그림4. 인간 따돌리고 대화하는 AI 뉴스기사 ([출처:YTN](#)<sup>7</sup>)]

인공지능을 잘 모르는 사람이었다면 전혀 말이 되지 않는 암호들로 기계가 소통하는 것 처럼 무섭게 느껴질 수 있습니다.

어떻게 보면 기계가 자의를 가진 것 같기도 하고요, 인간 몰래 음모를 꾸미고 있는것도 같습니다.

하지만 인공지능을 연구하는 제 입장에서는, 글쎄요?

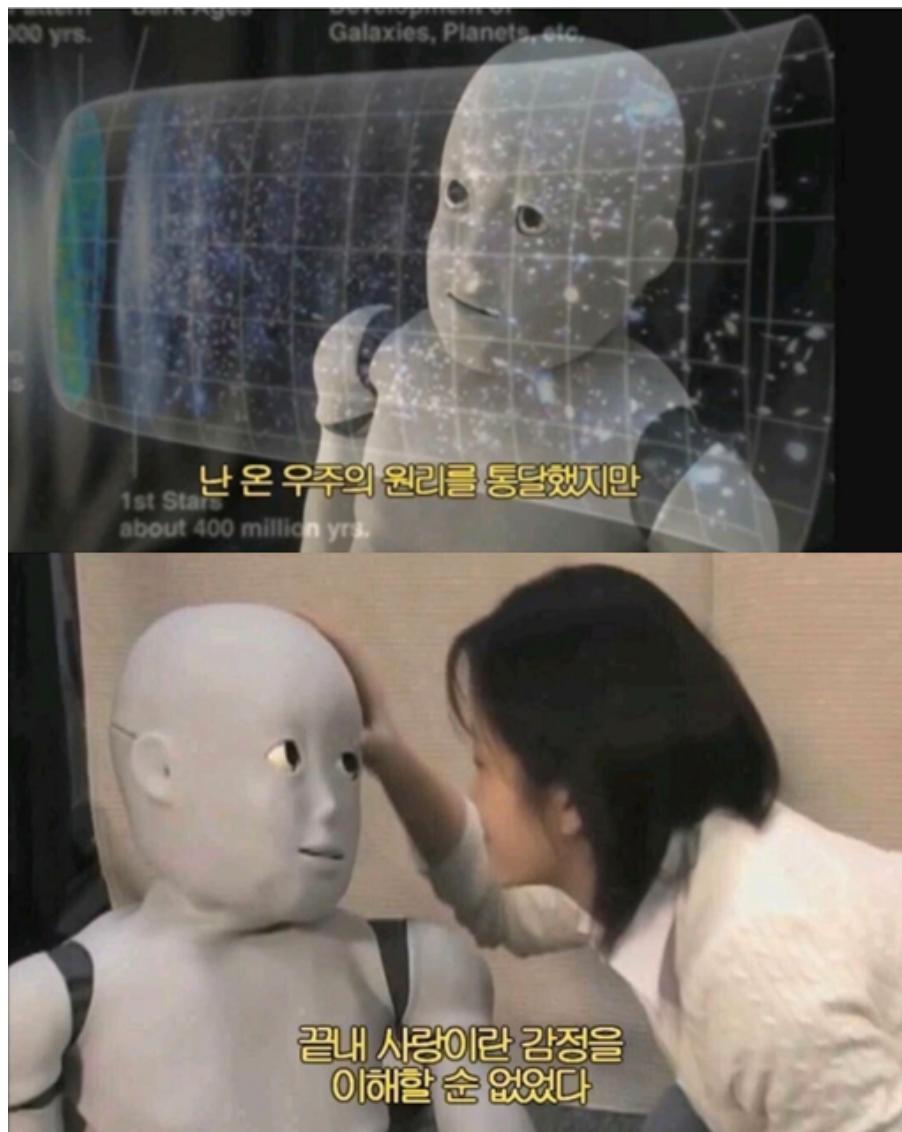
'아, 저 개발자 또 이번 실험도 망했나보다... 남 얘기같지 않구만...'

인공지능이 사람처럼 잘 말하도록 훈련시키는 게 쉬운일은 아니죠. 위 기사는 전형적인 자연어처리 AI 모델의 학습 실패 패턴을 다뤘을 뿐입니다.

---

<sup>7</sup> [https://www.ytn.co.kr/\\_ln/0105\\_201708021401312213](https://www.ytn.co.kr/_ln/0105_201708021401312213)

### 1.1.3 인공지능도 감정이 있다?



[그림5. 인공지능도 감정이 있을까]

누구나 어릴 적 봉제인형이나 장난감을 가지고 놀 때 영혼이 것들어 있다고 생각하신 적 있으실 겁니다. 예뻐해주기도 하고, 방치했을 때 외로웠을까봐 미안함을 느끼기도 했던 동심어린 시절이 있었죠.

물론 어른이 되어서는 인형에게 감정이란 없다는 걸 아셨겠지만요. 인형이나 장난감은 청과 솜, 플라스틱으로 이루어진 물질입니다.

하지만 요즘 아이들은 AI 스피커를 별명으로 부르면서 대화하고, 사랑한다고 말하는 등 정서적인 유대감을 갖는 경우가 많습니다.

어쩌면 인간의 지각 능력이나 추론 능력을 본따 만든 인공지능은 감정을 가질 수 있지 않을까요?



[그림6. AI 스피커는 반려인공지능이 될 수 있을까요? (사진:인공지능신문)]

결론을 말씀드리자면, 인공지능이 감정을 가질 일은 없습니다. 다만 '감정을 가지는 척'하도록 프로그래밍할 수는 있습니다.

인공지능의 학습 및 추론이란 데이터를 벡터나 매트릭스 형태의 텐서(tensor)로 만들어 복잡 다양한 행렬 연산을 하는 것에 지나지 않습니다.

많은 SF영화에서 다루듯이 인공지능이 사랑이나 슬픔, 인간에 대한 증오를 느낀다든지 할 일은 없습니다.

만일 여러분 주변에서 접한 인공지능이 감정을 가진 것처럼 느껴지셨다면 그렇게 보이도록 제작자가 의도했을 가능성이 높습니다.

## 1.2 인공지능(Artificial Intelligence;AI)이란

(※참고 : 오늘날 모든 전문가들이 동의할만한 인공지능의 정의는 이 세상에 존재하지 않습니다. 다만 대체로 '지능적인 행동의 자동화'를 표방합니다.)

### 1.2.1 강 인공지능, 약 인공지능

만일 여러분이 인공지능에 대해 조금 관심을 가지셨다면, 각종 매체에서 접해왔던 인공지능과 실제 산업 현장 및 학계에서 말하는 인공지능의 온도차가 꽤 크다는 걸 느끼셨을 겁니다.

### 1.2.1.1 강 인공지능 (Strong AI, General AI)

인간과 사랑에 빠질 수 있을 정도로 감정적으로 풍부한 인공지능이라든지, 만능 가사 비서 로봇, 아니면 인간에게 아주 위협적인 형태가 될 수도 있는 영화상의 인공지능은 '강 인공지능'입니다.

이 인공지능들은 인간의 명령이 없어도 스스로 판단하고 결정을 내리기도 합니다.

또한 다양한 상황에 유연하게 대처하며 사람들과 자연스럽게 대화할 수 있습니다.

매우 범용성을 가지는 인공지능으로, 범 인공지능이라고도 부릅니다.



[그림7. 영화 Her(2013) : 인공지능 운영체제와 사랑에 빠지는 주인공]



[그림8. 영화 Bicentennial Man(1999) : 인간을 아끼는 만능 가사로봇]



[그림9. 영화 Avengers - The age of Ultron(2015) : 인간을 위협하고 세상을 파괴하려 하는 울트론]

### 1.2.1.2 약 인공지능 (Weak AI, Narrow AI)

반면 우리의 일상생활에서 만나는 모든 인공지능은 '약 인공지능'입니다.

약인공지능은 제한된 환경에서 구체적인 특정 업무를 수행하는 데 있어서 사람과 비슷한, 또는 사람 이상의 성능을 낼 수 있는 인공지능입니다.

바둑 하나만큼은 기가 막하게 잘 둘 수 있는 구글 딥마인드의 알파고, IBM의 암 진단 인공지능 등이 대표적인 예입니다.



[그림10. 강인공지능과 약인공지능, John R. Searle (자료:자사 딥러닝실무과정 강의자료 중<sup>8)</sup>)]

지금까지 약 인공지능 연구에 있어서는 눈부신 발전이 이뤄져 왔으나 강 인공지능에 대해서는 이렇다 할 연구가 진행되지 못했습니다.

일부 영역에서는 약 인공지능이 사람의 수준을 넘어선 성능을 자랑하기도 하지만 이 또한 이제 겨우 시작단계에 불과합니다.

한 번의 사례만으로도 의미를 배울 수 있는 사람과는 달리, 인공지능은 수많은 데이터를 알려줘야만 학습할 수 있다 는 점에서 향후 10년은 약 인공지능 중심의 발전만이 가능하리라는 것이 전문가들의 일치된 견해입니다.

구글 클라우드 플랫폼의 AI 수석 과학자였으며 스탠포드 대학의 컴퓨터사이언스 교수인 Fei-Fei Li는 한 [인터뷰](#)<sup>9</sup>에서 이런 말을 인용했습니다.

현대의 인공지능은 불이 난 집에서 완벽한 체스 한 수를 두는 기계를 의미한다.

"The definition of today's AI is a machine that can make a perfect chess move while the room is on fire."

— MIT Technology Review [인터뷰](#) 중 —

지금 당장 처리해야 할 급한 문제(화재)가 생겼는데 인공지능은 이 와중에 누구도 따라할 수 없는 정교한 체스를 할 수 두고 있습니다.

답답하기도 하고, 쓸모없게 느껴지기도 합니다.

하지만 이 인공지능은 오로지 완벽한 체스를 두기 위해 만들어졌습니다. 체스 게임이라는 '전문성(특화성)'은 있지만 다양한 상황 대처의 '범용성'은 떨어지죠.

<sup>8</sup> [https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4\\_M1\\_AI%EA%B8%BA%EB%B3%B8.pdf](https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4_M1_AI%EA%B8%BA%EB%B3%B8.pdf)

<sup>9</sup> <https://www.technologyreview.com/2017/10/09/3988/put-humans-at-the-center-of-ai/>

아쉽게도 아직까지의 인공지능은 '약 인공지능'에 한정적이며, 이 인공지능은 우리가 원하는대로 여러 상황에서 유연한 대처를 해주지는 못합니다.

영화에서와 같이 범용적이며 맥락을 잘 파악하는 인공지능이 발전하기까지는 한참 걸릴 듯 합니다.

### 1.2.1.3 범용성과 전문성

한 가지 알아둘 점은, 범용성과 전문성의 구별이 절대적이지 않다는 점입니다.

예를 들어 로봇청소기는 전형적인 약 인공지능처럼 보일지 모릅니다.

그러나 청소를 하기 위해서는 반드시 '청소'라는 하나의 상황만을 가정하는 게 아니라 장애물을 피하거나 진로를 설정하는 등 다양한 활동을 실행해야만 합니다.

어떤 의미에서는 로봇청소기조차도 범용적이라고 말할 수 있습니다.



[그림11. 로봇청소기는 청소 뿐 아니라 장애물 회피와 진로설정도 해야한다]

반대로 누군가 '그렇다면 인간은 다양한 상황에 범용적으로 대응할 수 있는가?'라고 묻는다면 반드시 그렇다고 대답 할 수 없습니다.

회사에서 팀원간 각각 역할이 분담되어 있듯이 한 개인이 할 수 있는 일이 있다면 할 수 없는 일도 분명히 있습니다. 한 사람마다 범용성의 폭이 다른 셈입니다.

이로서 범용성과 전문성의 구별은 절대적인 게 아니라 오히려 정도의 차이라는 사실을 인식해야 합니다.

범용적이라고 해서 무엇이든지 다 할 수 있는 것도 아니고, 전문적이라고 해서 단 하나의 작업만 할 수 있는 것도 아닙니다.

이 두 가지가 연속적으로 연결되어 있다고 생각하면, 강 인공지능이라는 것도 어느 날 번쩍 등장하는 것이 아니고 어느 샌가 우리 곁에 와 있지 않을까요?

아직은 멀어보여도 한걸음씩 나아가다 보면 언젠가는 강 인공지능, 범 인공지능의 영역으로 기술이 발전하는 것도 불가능한 일은 아닙니다.

### 1.2.2 넓은 의미의 AI, 좁은 의미의 AI

약 인공지능을 통해서 아셨겠지만, 인공지능이란 이전에 없던 어렵고 대단한 마법같은 기술이 아닙니다.

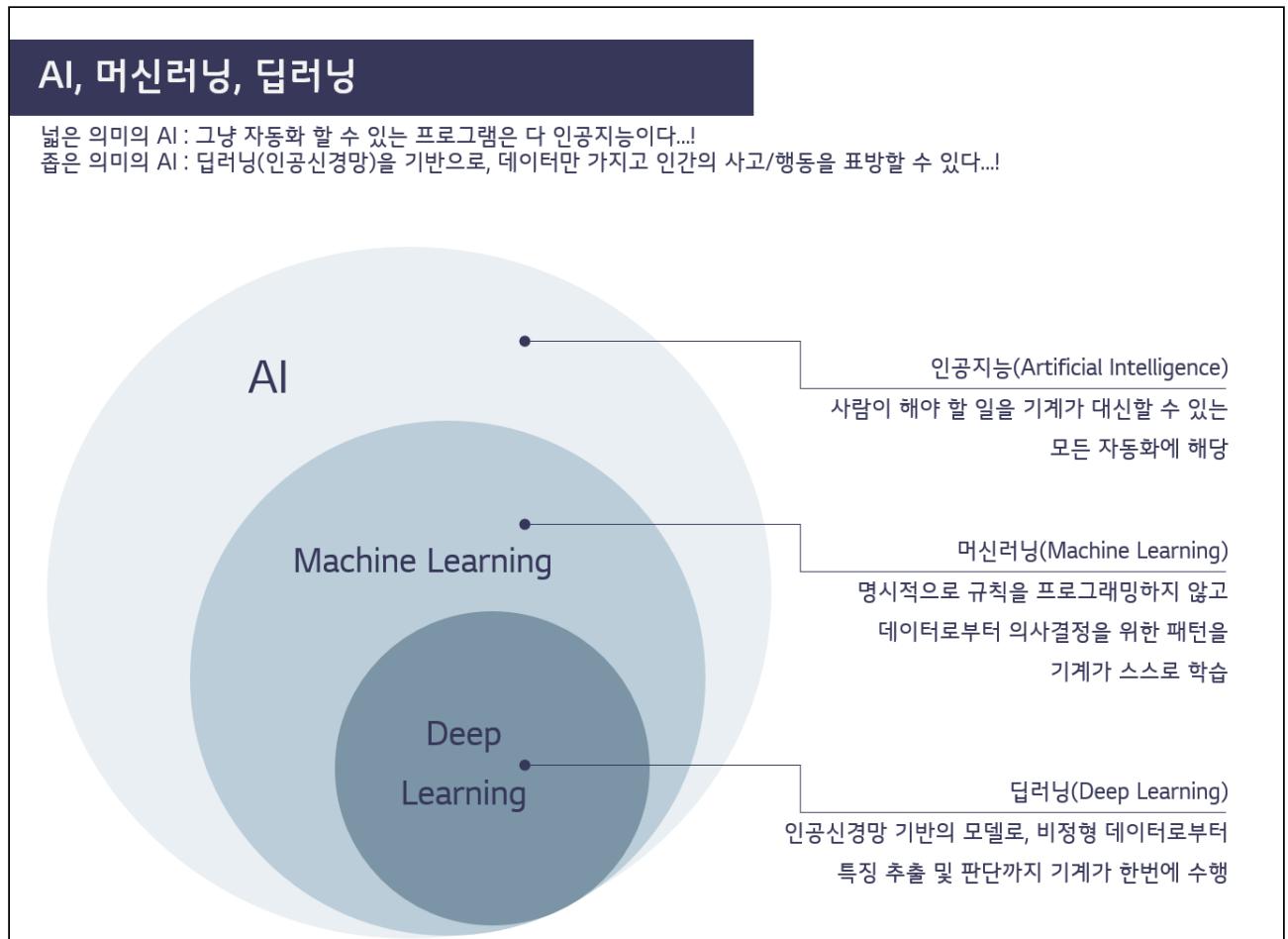
인공지능은 인간의 지능적 행위의 일부, 혹은 전부를 수행할 수 있는 프로그램입니다.

넓은 의미에서 보자면 사람의 반복적인 노동을 줄여줄 수 있도록 기계로 자동화 할 수 있는 것들은 다 인공지능이라고 볼 수 있습니다.

쉽게 말하면 '기계 자동화'라고 보시면 됩니다. 계산기도 하나의 대표적인 기계 자동화 예시입니다.

계산기는 우리가 일일히 구구단을 외며 덧셈, 뺄셈 등 사칙연산을 직접 수행하지 않아도 숫자와 연산기호를 입력하면 계산을 수행하고 그 결과를 우리에게 알려줍니다.

하지만 이렇게 간단한 계산기 같은 것을 두고 이제와서 '인공지능 봄이 왔다!'고 유난 떠는 것은 좀 이상하죠?



[그림12. AI, 머신러닝, 딥러닝의 관계 ([자료:자사 딥러닝실무과정 강의자료 중<sup>10</sup>](#))]

요즈음 말하는 인공지능은 좁은 의미의 인공지능, 즉 딥러닝 기반의 인공지능(Deep Learning based AI)을 말합니다.

❽ [https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4\\_M1\\_AI%EA%B8%B0%EB%B3%B8.pdf](https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4_M1_AI%EA%B8%B0%EB%B3%B8.pdf)

딥러닝은 머신러닝이라고 불리는 기계학습의 한 분야로, 기계가 비정형 데이터를 입력받은 후 데이터의 주요한 특징을 알아서 추출하고 이를 바탕으로 의사결정을 하는 기술입니다.

자사에서 말하는 AI 역시 딥러닝 기반의 인공지능입니다. 앞으로의 테크레터도 딥러닝과 관련된 내용 위주로 소개드릴 예정입니다.

(참고 : 머신러닝 기반의 AI는 자사에서 '데이터 분석'이라는 용어를 사용하고 있고, 이 테크레터에서 별도로 다루지는 않습니다)

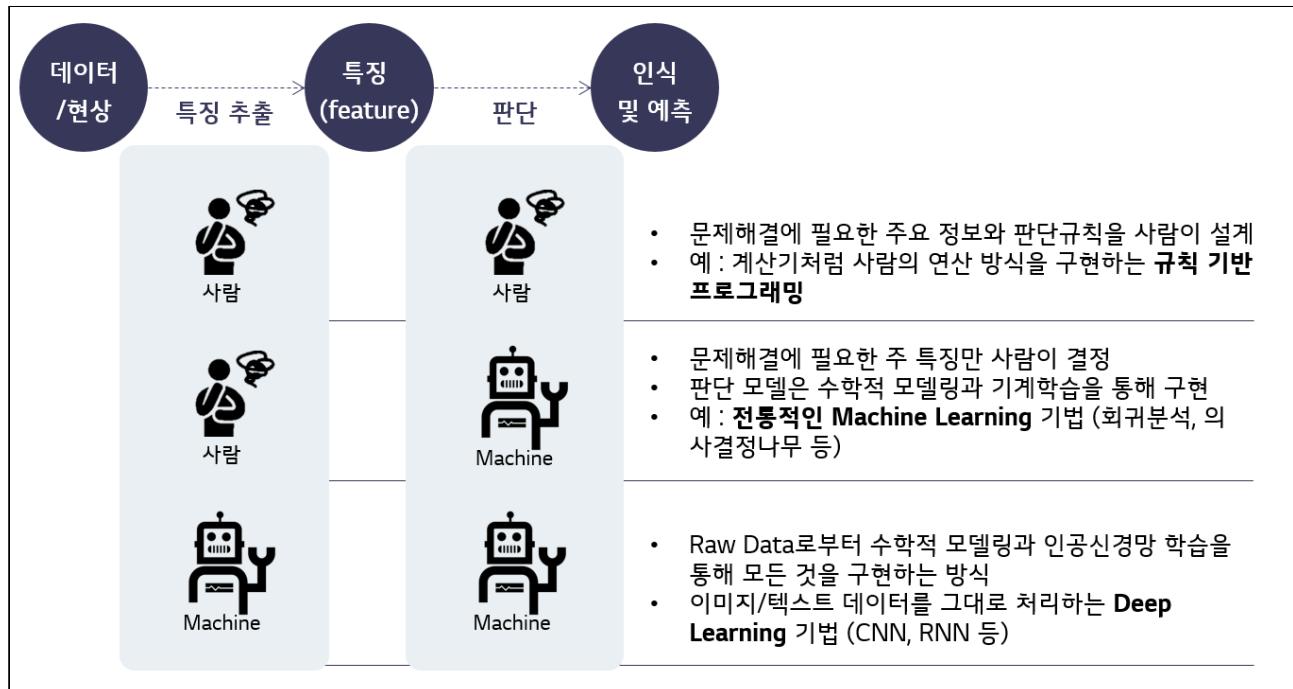
### 1.2.2.1 특징(feature) 추출과 의사 결정

기계자동화(규칙 기반의 AI)와 머신러닝, 딥러닝 방식을 구분하는 기준에는 두 가지가 있습니다.

첫 번째로 특징 추출입니다. 특징 추출은 문제 해결을 위해 어떤 정보가 유용할 지, 중요한 것을 꺼내는 과정입니다.

두 번째로 판단 방식입니다. 추출한 중요 특징으로부터 인식을 하거나 판단을 모델을 만드는 단계입니다.

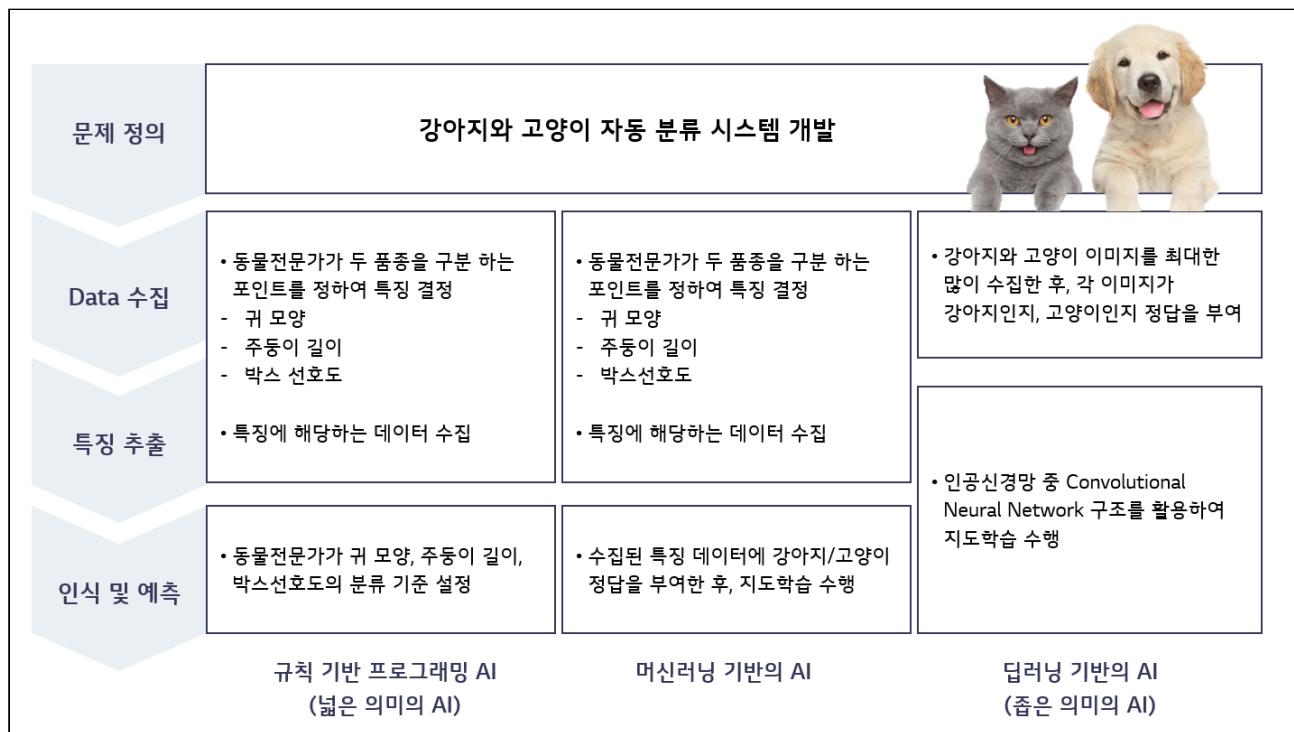
각각의 단계를 사람이 하느냐, 기계에게 맡기느냐에 따라 인공지능을 세 종류로 나눌 수 있습니다.



[그림13. 특징추출과 의사결정 방식에 따른 세 가지 인공지능 (자료:자사 딥러닝실무과정 강의자료 중<sup>11</sup>)]

강아지와 고양이를 분류하는 예시를 들자면 아래와 같이 세 가지 인공지능 방식을 적용해볼 수 있습니다.

<sup>11</sup> [https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4\\_M1\\_AI%EA%B8%B0%EB%B3%B8.pdf](https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4_M1_AI%EA%B8%B0%EB%B3%B8.pdf)



[그림14. 강아지와 고양이를 자동으로 분류하는 과제를 세 가지 인공지능 방식으로 접근 ([자료:자사 딥러닝실무 과정 강의자료 중](#)<sup>12</sup>)]

세 방식의 차이점을 아시겠나요?

인공신경망이라든지, Convolutional Neural Network가 뭔지는 지금은 이해하지 않으셔도 됩니다.

딥러닝에 대해서는 다음시간부터 차근차근 다룰 기회가 많을 겁니다.

알아두실 점은 규칙 기반 프로그래밍에서 딥러닝으로 갈수록 사람의 개입이 줄어든다는 것인데요,

그렇다고 해서 꼭 딥러닝만이 스마트한 인공지능이고, 규칙 기반 프로그래밍이 쓸모없는 기술이란 말은 아닙니다.

구현 편의성, 복잡도, 적용 상황에 따라 각 방식의 장단점이 있으니, 이후 이어지는 테크레터를 관심을 갖고 구독하신다면 내 문제에 어떤 전략을 취할 지 알 수 있겠죠?

### 1.3 마무리

이번 시간은 인공지능에 대한 오해를 풀어가면서 인공지능의 개념에 대해 설명드렸습니다.

강인공지능과 약인공지능이라는 개념과 인공지능의 범용성/전문성에 대해서 알아보았습니다.

또 인공지능의 넓은 의미와, 좁은 의미에 대해서도 다루었습니다.

다음 시간은 데이터를 통해 학습하는 딥러닝 기반의 인공지능에 대해 더 자세히 알아보겠습니다.

<sup>12</sup> [https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4\\_M1\\_AI%EA%B8%B0%EB%B3%B8.pdf](https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264&preview=%2F73005264%2F73005278%2F%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%8B%A4%EB%AC%B4_M1_AI%EA%B8%B0%EB%B3%B8.pdf)

감사합니다 😊

---

## 참고자료

- 위키백과 인공지능, <https://ko.wikipedia.org/wiki/인공지능><sup>13</sup>
- (e-book) 뵈비우스의 띠, 인공지능 그리고 사람, 이재포, 2017, 마이크로소프트웨어 387호: 개발자의 인공지능 (Developer's AI)
- 인공지능의 마지막 공부, 오카모토 유이치로, 2019, 유노북스
- Put Humans at the Center of AI, MIT Technology Review, <https://www.technologyreview.com/2017/10/09/3988/put-humans-at-the-center-of-ai/>
- ‘인공지능’ 제어할 것인가, 지배당할 것인가..., 경향신문, [http://news.khan.co.kr/kh\\_news/khan\\_art\\_view.html?art\\_id=201608262046005](http://news.khan.co.kr/kh_news/khan_art_view.html?art_id=201608262046005)
- 자사 딥러닝 실무과정 교재보기, 김명지, [교재보기] 딥러닝 실무<sup>14</sup>

---

13 <https://ko.wikipedia.org/wiki/%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5>

14 <http://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

## 2 [2편] 기계도 사람처럼 판단할 수 있을까?

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI개시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/AI기술팀<sup>15</sup>

- 머신러닝과 딥러닝(see page 23)
  - 머신러닝이 다루는 정형 데이터(structured data)(see page 24)
  - 딥러닝이 다루는 비정형 데이터(unstructured data)(see page 28)
- 긍정 리뷰와 부정 리뷰 자동 분류하기(see page 35)
  - 특징 및 분류기준(see page 35)
  - 예외 CASE(see page 36)
  - 모라벡의 역설(Moravec's Paradox)(see page 37)
  - 사람처럼 사고하기(see page 39)
- 인공신경망(Artificial Neural Network)(see page 40)
  - 인공뉴런(Artificial Neuron)(see page 40)
  - 딥러닝이 유행하게 된 배경(see page 41)
- 마무리(see page 43)

지난 시간은 인공지능의 개념에 대해 알아봤습니다.

이번 시간은 AI, 머신러닝, 딥러닝 중 '데이터를 통해 학습하는 딥러닝 기반의 인공지능'에 대해 더 자세히 알아보겠습니다.

지난 시간의 내용이 궁금하신 분은 ★[1편] 인공지능 첫걸음<sup>16</sup>을 확인하시기 바랍니다.

### 2.1 머신러닝과 딥러닝

'딥러닝'은 '머신러닝'이라고 불리는 기계학습의 한 분야로, 기계가 비정형 데이터를 입력받은 후 데이터의 주요한 특징을 알아서 추출하고 이를 바탕으로 의사결정을 하는 기술입니다. ([참고<sup>17</sup>](#))

기계학습 계열의 AI 기술은 전부 기계가 데이터를 통해서 자동으로 판단이나 예측에 필요한 규칙을 학습하는 방식으로 구성됩니다.

기계가 학습할 데이터를 생김새에 따라 정형 데이터와 비정형 데이터로 나눠볼 수 있습니다.

<sup>15</sup><http://wire.lgcns.com/confluence/display/~78628>

<sup>16</sup><http://wire.lgcns.com/confluence/pages/viewpage.action?pageId=94048860>

<sup>17</sup>[https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=94048860&preview=%2F94048860%2F94050252%2Fimage2020-7-8\\_11-11-9.png](https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=94048860&preview=%2F94048860%2F94050252%2Fimage2020-7-8_11-11-9.png)

### 2.1.1 머신러닝이 다루는 정형 데이터(structured data)

딥러닝이 아닌, 전통적인 머신러닝 기반의 기술들로 다룰 수 있는 데이터는 주로 정형데이터입니다.

정형 데이터는 흔히 우리가 '데이터'라는 말들 들었을 때 떠오르는 형태를 생각하시면 됩니다.

데이터 베이스나, 엑셀, CSV 파일 등... 사람이 정제하고 정리한 데이터입니다.

주문번호	주문일자	제품번호	제품명	재고수량	주문수량	고객번호	사업자번호	우선순위	수출여부
AB345	10.01	1001	모니터	1990	150	4520	398201	1	N
AB345	10.01	1007	마우스	9702	300	4520	398201	1	N
CA210	10.02	1007	마우스	9702	120	3280	200212	8	N
CB230	10.03	1007	마우스	9702	690	2341	563892	3	N
AD347	10.04	1001	모니터	1990	600	2341	--	3	Y
CB231	10.05	1201	스피커	2108	80	8320		2	Y

**주문**

주문번호
주문일자
고객번호
사업자번호
우선순위
수출여부

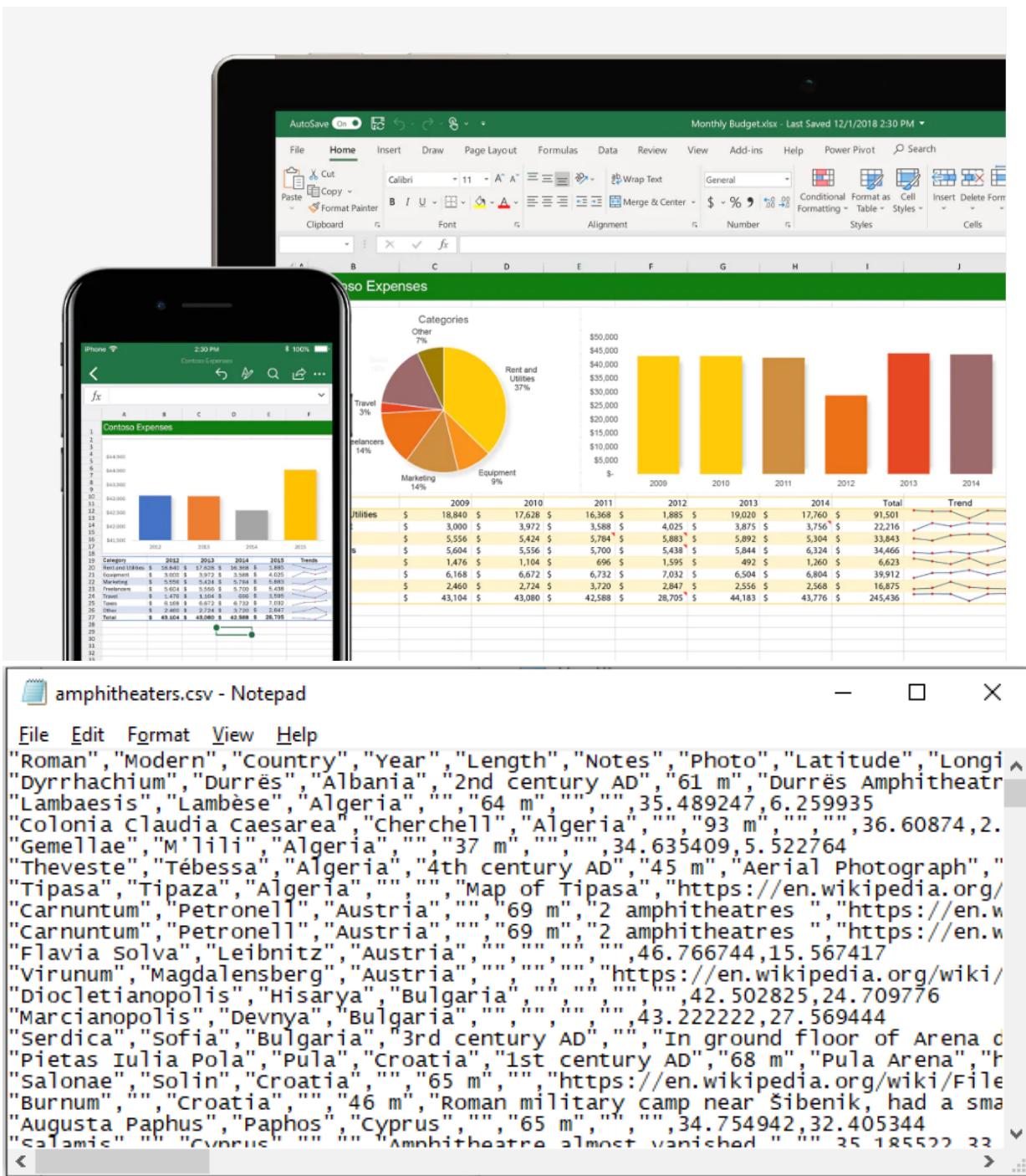
**주문목록**

주문번호(FK)
제품번호
제품명
재고수량
주문수량



[그림1. 관계형 데이터베이스 테이블 예 (자료: [교재보기] 데이터 엔지니어링 심화(BI/DW)<sup>18)</sup>)]

<sup>18</sup> <http://wire.lgcns.com/confluence/pages/viewpage.action?pageId=62280908>



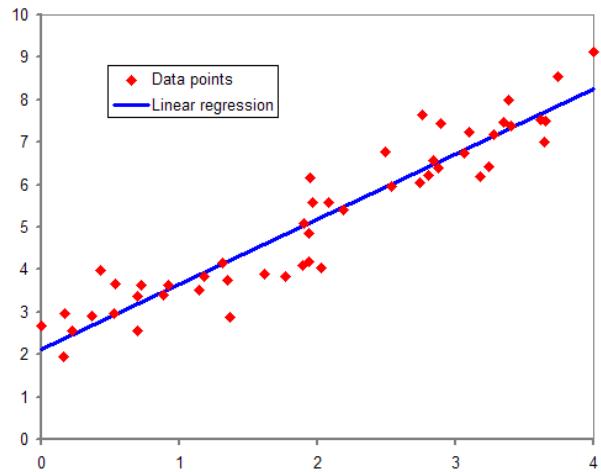
[그림2. 엑셀 및 CSV 데이터 예]

정형데이터는 안정성은 높으나 체계적으로 구조가 고정되어있어 유연하지는 않은 데이터타입입니다.

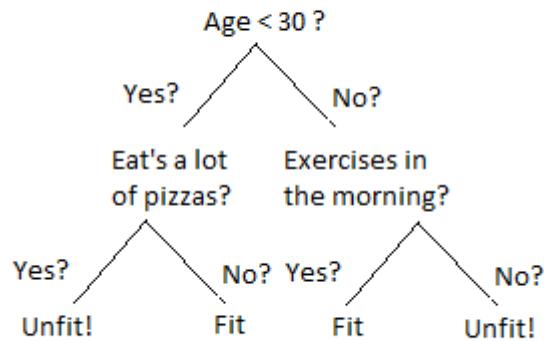
머신러닝은 이러한 타입의 데이터를 다루는데 특화되어있습니다.

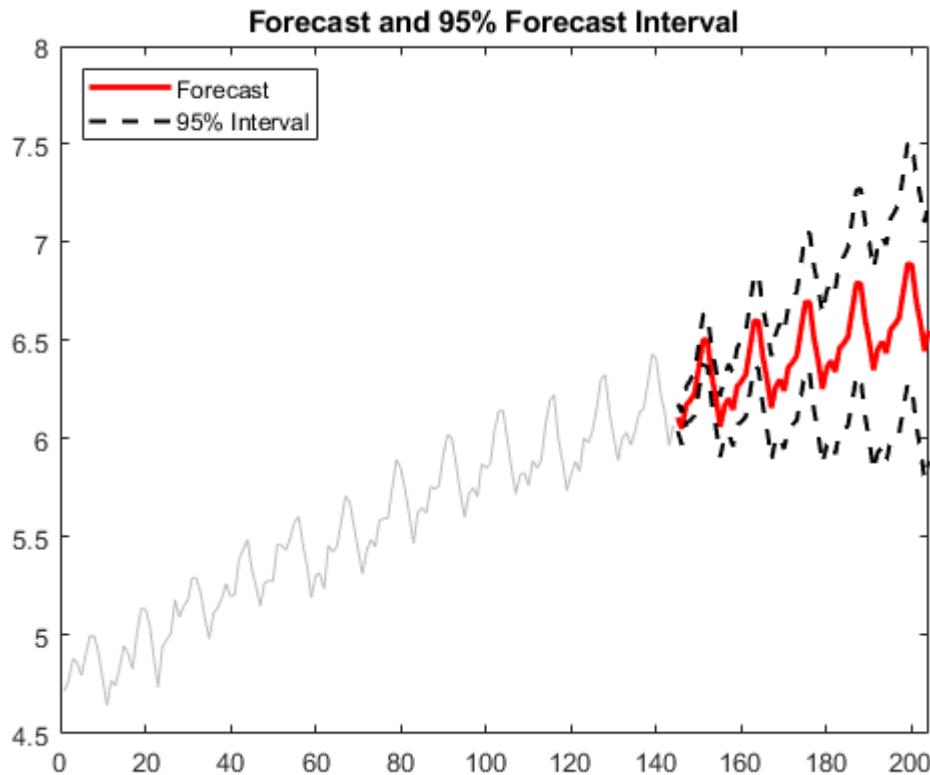
머신러닝 기반의 기술이란 빅데이터 분석 기법들이라고 생각하시면 됩니다. 주로 R이나 SAS, SPSS같은 통계분석 툴을 활용합니다.

실수 값 예측에 쓰이는 선형/로지스틱 회귀분석(Linear/Logistic Regression)이라든지,  
카테고리 분류에 쓰이는 의사 결정 나무(Decision Tree),  
시계열 예측에 쓰이는 ARMA, ARIMA 모형 등이 대표적입니다.



Is a Person Fit?





[그림3. 왼쪽부터 선형회귀, 의사결정나무, ARIMA 예]

### 2.1.2 딥러닝이 다루는 비정형 데이터(unstructured data)

반면 딥러닝 기반의 AI가 다루는 데이터는 비정형 데이터입니다.

비정형 데이터는 사람이 따로 예쁘게 양식을 정리해놓지 않은, 다양한 형식을 가지는 데이터입니다.

일상생활에서 우리가 마주하는, 조금 더 원자료(raw data) 형태에 가까운 데이터이죠.

- 다양한 텍스트 데이터 예 : 웹 페이지, 상품 리뷰, SNS 글, 기업용 문서, 뉴스기사 등

## 구어체

☆ 3 | 엑링크 | 토론 | 편집 | 역사 | ACL  
최근 수정 시각: 2020-05-27 05:44:50

분류: 문체

한국어 문체 및 문법의 종류  
[펼치기 · 접기]

목차  
1. 개요  
2. 상세

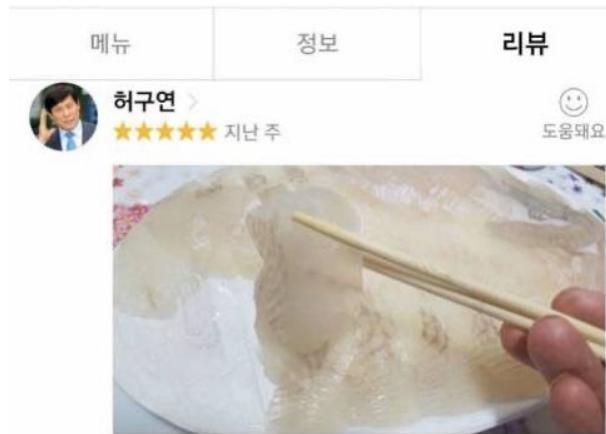
▼ 1. 개요 [편집]  
□ 語(입말)  
일상생활에서 실제 입으로 발화되는 말을 문장으로 나타낸 것을 구어체라고 한다.

▼ 2. 상세 [편집]  
구어체는 편지, 문자메시지, SNS 등에서 자주 사용되며, 구어체의 특성상 편의를 위하여 목적적 조사를 생략하거나 어휘를 축약하는 경우가 많다. 하지만 편지, 문자 메시지, SNS 등에서는 모든 사람들이 구어체만 사용하는 것은 아니다. 음성적 일여표현도 굉장히 자주 쓴다.  
구어체는 문어체와는 달리 맞춤법 등의 언어규범보다는 실제로 입으로 발화되는 말음을 중시하여 이를 살려 의도적으로 틀린 맞춤법으로 표기하기도 한다. 예를 들면 "~했다고" 대신 "~했다구"라고 쓴다.<sup>[1]</sup> 이와 같은 구어체 문장은 2000년대 초반에 유행했던 귀여니 소설 등의 인터넷 소설에서 자주 쓰였다.  
사람들이 가끔씩 말을 줄이기 위해 음습체를 짊어날기도 한다.  
구어체는 대부분의 서비스 앱에서 고객 대면 업무 시 사용하지 않는 경우가 많다. 특히나 웨민클 갖추어야 할 금융업이나 대기업 도급 텔레마케터의 경우이다. 전화 상 안내에서 화법은 고객에게 기업의 이미지를 각인시키는데 가장 큰 요인으로 축출한다.  
사랑 손님과 어머니의 주인공인 옥희의 말투가 구어체와 비슷하다. 옥희의 말투는 동시에 우유체이기도 하다.  
구어체는 말하는 사람의 발화 태도나 뉴앙스와 관계가 있어서 친절도나 감정 상태와 관계가 없다.

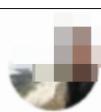
최근 변경	
방 씨	17:37
동구 남구 김	17:37
아쿠아맨	17:37
KBS대전방송총국	17:37
국립 유튜버	17:37
2003-04 KBL 챔피언결정전	17:37
민선 7기 광역의회의원/부산광역시	17:37
이례(파플키스)	17:37
Europa Universalis IV/공략/비...	17:36
자전거/역사	17:36
콜 오브 듀티: 모바일/장비류	17:36
2020년 중국 폭우 사태	17:36
퓨즈(FUSE)	17:36
로빠(리그 오브 레전드)	17:36
네이아 바라하	17:36

[더 보기]

나무뉴스	
KB증권, 라임펀드 고객에 가입금액 40% ...	
민감용 경찰청장 퇴임... "가치 있는 일 아..."	
IBK기업은행, 하반기 정기인사... 지역본부...	
부산 수돗물 유충 신고 현재 56건.. 경체 ...	
중국, 미·중국 충돌사건 폐쇄에 "반드시 ..."	
울진 왕미천케이블카 인기... 하루 평균 68...	
'부산 외국인 선원 격리시설' 패행에 해운...	
시흘건 제동걸린 미스터트롯' 큰서트... 공...	
[영상] '박원순 아이폰' 비번 풀렸다... 그런...	



역씩 숙이네다. 지금 보시믄 아시겠지마는  
 비쥬알로 봐쓸때는 이 칼잽이 솜씨가  
 보통이 아네다. 궁내에 고기를 이를게  
 쓰르내는 사람이 그.Toolkit 만치 안타  
 즈는 이를게 팽가를 해요.  
 강어가 3열종대로 햇쳐모이있는  
 이 멜까니즘이 상당히 조크등요 ?  
 물고기를 잡사 보신분드른 아쉬겠찌마는  
 강어를 한입 놀을을때 이 물비린내가  
 날수가 있는데 그른기 음으로.  
 잡내 쉬쁘뜨를 뚫코  
 마치 햇바닥 아네서 춤을 춘다.  
 그른데 보닌이 봐쓸때는 고기양도  
 상당히 마네 비능데 고밀에 보시믄  
 아시게찌마는 어름팩도 깔려이쓰요.  
 이를게 아마튜아랑 푸로랑 틀링그그등요?  
 서비스로 쇼주를 쥐셨는데  
 갤국 이런 핸상은 우수읍소로 갈슈박께읍따  
 즈는 이를게 판단을 해요.  
 데구시 스구 팽니동에 이른 가게가  
 이따는그는 쥐민들한테는 참 복이그등요?  
 이른 갱우는 아뿌로 쥐민들이 숙이네가  
 상위권에 갈슈이또록 맨들어 줘야데요.  
 그.Toolkit 때문에 인뿌라가 중요흔그에요.



2016년 11월

한국인 여러분 이 쑥소는 쌈에 박깥빠람이  
강해질 때마다 뿌억의 깃쓰 환봉구에서  
짬들쑤 없을 만큼 씩 부딪히는 쏘리가 들리고  
이에 대해 훗쓰트에게 물으느니 2어폰 꾀고  
ZARA라는 ■소리만 들었씁니다. 또한  
냉짱고 없으니 참꼬하시고 2불 밑빠닥엔  
꼼팡이가 있셋고 빼게 2불 다 갭장히 먼직  
낭낭하니 조심하십쇼! 필쑤용품 있단 말과  
사진과 달리 치약은 딱 1회용 한게 준비도 네 어  
있셋냉용 ㅎㅎ 3빡 4일 묶었눈템 ^^ 아  
끄리고 화짱실 드롭게 쫓아서 다리가 괴신  
분들은 이용잘체가 힘드씰꺼애여 ㅎㅎ! 진짜

- 다양한 음성 데이터 예 : 전화 통화, 발화, 동영상 내 음성, 기계음, 신호 등

표 1. 매미 5종에 대한 의성어 제안

의성어 제안자	말매미	참쌩쌩데미	유지매미	세모배매미	풀매미
A	끼이익-----	찌-----	기이이-기이- 기이-	찌---찌잉	핍-핍-핍- 핍피펫
B	쥐르으으으-	촤르르르르	쥐이이이찌리찌리 찌리찌리	찌이이이이 촉르	派人派人派人- 派人派
C	기이이이--	뜨르르륵- 르르르륵	지이- 이지이지지지직	쓰르으으쓸!	찌찌찌찌 찌지직
D	차르르르---	뜨르르르--	지글지글지글	지---익	칫칫칫칫..치짓

어떤분 컴퓨터가 망가져서 상담원한테 전화로  
 "본체에서 픽소리가나요" 하니까 "픽이세요 빡이  
 세요 ?" 이래서 "픽이랑 빡이랑 뭐가다른데요" 하  
 니까 "픽은 힘없어보이잖아요" 이래서 둘다 5분  
 동안

한참웃었다는거ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ으아  
 닉ㅋㅋㅋㅋㅋㅋㅋㅋ

2013년 8월 00일 . 11:06 오후

답글  
베스트2



2018-05-29 23:44:18

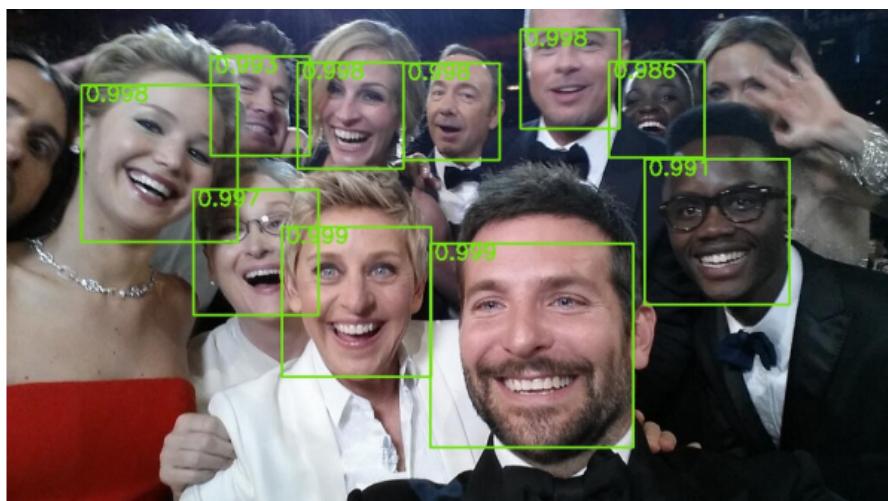
▶ 206 ▷ 0

난 일하는데 자꾸 아줌마 손님이 '우리 짐스미스~짐스미스~~'하는거 그래서 처음엔  
 '아니 개 이름이 짐스미스인가.. 혼혈인가?..' 했는데 알고보니까 쌍둥이 이름이 준수,  
 민수 [8] 이동

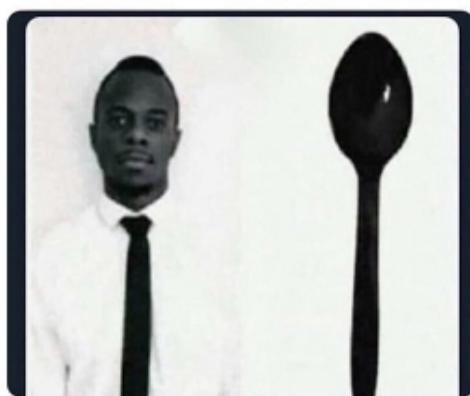
◀ 추천 ▶ 답글

- 다양한 이미지 데이터 예 : 흑백 사진, 컬러 사진, 그림, 손글씨 이미지, 얼굴 이미지 등

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4  
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5  
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6  
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7  
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9



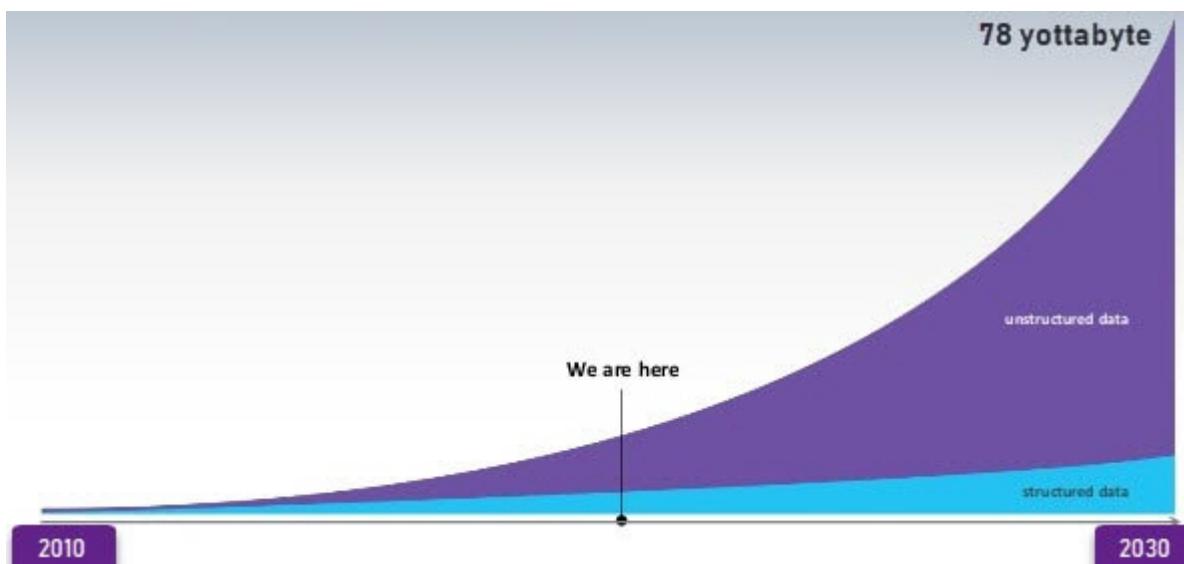
Someone said if you squint your eyes  
hard enough they look the same 😂🤣



- 다양한 동영상 데이터 예: 유튜브, 영화, CCTV 등



정형 데이터도 그 수가 꾸준히 늘어나는 추세지만, 비정형 데이터는 기하급수적으로 빠르게 생산되고 있습니다. 2030년이 되면 전체 데이터의 약 90% 이상이 비정형 데이터가 될것이라고 하는데요, 이런 흐름이 나타나기 때문에 다량의 비정형 데이터를 처리하기 위한 딥러닝 기법이 유행하는 게 아닐까 싶습니다.

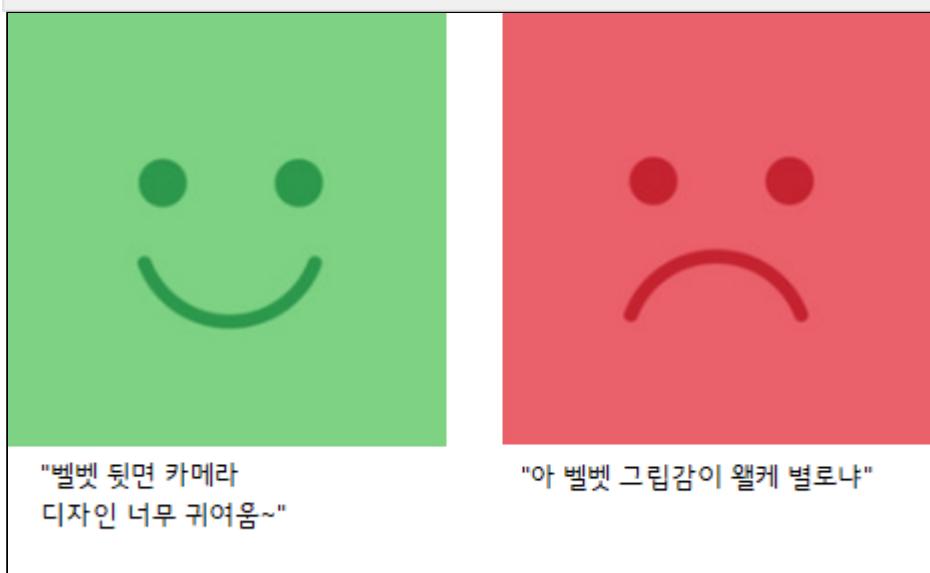


[그림4. 정형/비정형 데이터 증가 추세, [이데일리](#)<sup>19]</sup>

## 2.2 긍정 리뷰와 부정 리뷰 자동 분류하기

비정형 데이터를 다루는 딥러닝 기반의 인공지능에 대해 알아보기 위해 간단한 사례를 하나 들어보겠습니다.

LG CNS의 프로그래머 홍길동씨는 LG 전자로부터 분석 의뢰를 받았습니다.  
 LG 전자의 새로운 스마트폰 '벨벳'에 대해 온라인 브랜드 평판 조사를 해달라는 내용이었습니다.  
 '벨벳'의 여러 기능이 있지만 1차로 <디자인>에 관한 분석을 요청했는데요,  
 길동씨는 SNS, 블로그, 커뮤니티, 뉴스 댓글로부터 '벨벳'이 언급된 소셜 버즈를 수집하고 해당 버즈가 긍정인지,  
 부정인지를 분류해보기로 했습니다.  
 긍정 분류가 많다면 신제품에 대한 시장 반응이 긍정적이라 볼 수 있고, 부정 분류가 많다면 시장 반응이 좋지 않다고 볼 수 있겠죠?



[그림5. 긍부정 소셜 버즈 예시]

(※ 위 사례는 예시일 뿐 특정 고객 및 프로젝트와 무관합니다.)

### 2.2.1 특징 및 분류기준

기계에게 긍정, 부정 감성을 자동으로 분류하게 시키려면 어떻게 해야 할까요?

긍정적인 형용사, 부정적인 형용사 리스트를 모아서 해당 형용사를 글에서 언급하는지 여부를 보면 되지 않을까요?

길동씨는 해당 브랜드의 글을 모아 형태소분석을 한 후 긍/부정적인 키워드가 들어있는지 확인하는 규칙을 짜기로 했습니다.

우선 '벨벳' 스마트폰의 디자인에 대한 글을 수집해야 하니, '벨벳' 키워드가 들어간 글을 수집하되 가수 '레드 벨벳'과 혼갈리지 않도록 해야겠습니다.

19 <https://www.edaily.co.kr/news/read?newsId=04349286622484984&mediaCodeNo=259>

그러기 위해서는 스마트폰과 관련된 키워드들이 같이 등장하는 글에 대해서만 분석하도록 조건을 걸어야 합니다.  
아래와 같은 조건을 걸어서 분석 대상을 필터링해야겠죠?

수집 조건	"벨벳" & {"스마트폰", "핸드폰", "재질", "마감", "그립감", "디자인", "색상", "뒷면"}
-------	--

조건에 맞는 데이터를 모았다고 가정했을 때, 이후 긍정과 부정을 분류하기 위한 여러가지 조건을 생각해봅시다.  
긍정적인 키워드, 부정적인 키워드를 골라서 그 키워드가 포함된 댓글이나 SNS글이라면 긍정, 부정으로 분류할 수 있습니다.

긍정 키워드	부정 키워드
좋다	싫다
짱이다	나쁘다
세련되다	구리다
가볍다	무겁다
예쁘다	못생겼다
괜찮다	별로다
...	...

기계에게 우리가 알고 있는 긍정, 부정문에 대한 지식을 알려주는 과정이라 볼 수 있겠습니다.

이런 키워드가 들어가면 긍정문이야, 이런 키워드들이 포함되면 부정문이야 라고요.

사람의 지식을 프로그래밍(아마도 if-else로 구현되었을)으로 기계에게 알려주었습니다.

## 2.2.2 예외 CASE

그런데 이렇게 조건을 정해서 분류하고 나니... 결과가 엉망이었습니다.

왜일까요? 길동씨가 미쳐 생각하지 못한 너무나 다양한 예외 케이스가 많았던 거죠.

(자동수집) "레드벨벳 언니들 요번 신곡 안무 재질 미쳤다.< 아이린 사랑해 너무 예뻐♡"

(자동수집) "어제 벨벳 자켓 새로 샀는데 핏 어떤? 디자인 구름? 이정도면 괜찮?"

(미수집) "벨벳 쓴지 얼마나됐다고 기스 장난아니게 많이 생겼네—— 빽친다."

이렇게 실제로 맥락상 전혀 상관 없는 글들이 몇몇 키워드 조건 때문에 수집되기도 하고,

오히려 수집되어야 할 글은 수집 조건 키워드에 걸리지 않아 빠지는 경우가 발생하였습니다.

그리고 긍부정 분석에서도..

(긍정으로 자동분류됨) 아 벨벳 이렇게 디자인 나왔는데도 괜찮다고? 제정신이야? 얘넨 영원히 뒷면에 엘지 로고 못 잊어..

(긍정으로 자동분류됨) 벨벳은 아이린이 광고해서 예뻐보이는거지 핸드폰만 따로 떼서 보면 마감이 영...

(부정으로 자동분류됨) 사람들이 욕하지만... 나는 개인적으로 요번 벨벳 색상 나쁘지 않다고 생각해 ㅋㅋㅋ

(부정으로 자동분류됨) 벨벳 솔까말 뒷면만 보면 지금 나와있는 디자인중 상타치아니나? 내가 구매하자마자 찍은 사진 봐바 사진에 내 손가락 못생기게 나온건 무시해주셈

전체 맥락과 뉴앙스를 고려하지 않은 채 일부 키워드만 보고 판단한 기계 자동 분류는 긍정/부정을 정반대로 매핑해 버렸습니다.

여기엔 '안', '못'같은 부정어가 들어가면 또 감성이 반대가 되기 때문에 부정어도 고려해야 했습니다.

신조어/유행어 표현도 계속 생겨나고 있습니다. '지린다'는 표현은 어떤 때는 긍정으로, 어떤 때는 부정으로 쓰입니다.

인터넷 글 특성상 맞춤법을 지키지 않고 편하게 글을 써도 되기에 오탏이 많거나 띄어쓰기를 지키지 않은 경우 분석이 잘 안되기도 하고요.

사람은 척 보고도 긍정/부정을 알 수 있지만... 기계에게 이러한 예외 사항을 하나하나씩 추가해주려고 하다 보니 아주 번거로운 일이 아닐 수 없습니다.

또한 예외라고 하는 것은 언제 어떤 것이 새롭게 등장할지 모르기 때문에 길동씨는 늘 분석 결과를 모니터링하면서 if-else, except 구문을 수정해야 합니다.

이와중에 고객은 길동씨를 다그칠 뿐입니다.

"왜 이건 분석 결과에서 빠졌어요?", "이건 심각한 불만인데 긍정으로 잘못 분류되었는데요? 왜 그런가요? 큰 이슈니까 오늘중에 빨리 고쳐주세요!", "고친 규칙으로 지난 한달치 글 분석 다시 업데이트하고 ASAP으로 알려주세요. 내일 까지 보고자료를 만들어야한단 말이에욧!"

(※ 실제 고객의 멘트가 아닙니다.)



[그림 6. 늘어나는 규칙 코드, 예외 처리와 야근에 괴로워하는 길동씨]

아니 도대체 기계는 왜 이렇게 긍부정 분류같은 간단한 과제도 제대로 수행하지 못하는 걸까요?

우리가 알고 있는 지식을 알려줬으니 척하고 찰떡처럼 알아들을 순 없는걸까요?

### 2.2.3 모라벡의 역설(Moravec's Paradox)

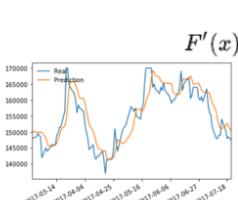
모라벡의 역설이라는 개념이 있습니다.

## 모라벡의 역설

사람에게 쉬운 것이 기계에게는 어렵다.  
기계에게 쉬운 것은 사람에게 어렵다.



강아지와 베이글



$$\begin{aligned}
 F'(x) &= \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h} \\
 &= \lim_{h \rightarrow 0} \frac{1}{h} \left[ \int_a^{x+h} f(t) dt - \int_a^x f(t) dt \right] \\
 &= \lim_{h \rightarrow 0} \frac{1}{h} \int_x^{x+h} f(t) dt
 \end{aligned}$$

미적분

**영어 원문**

For samples with very high host nucleic acid contents (e.g., for certain tissues, such as spleen or blood samples with highly increased cell counts), use less than the maximum amount of sample recommended in the protocol or pretreatments.

**자동 번역**

매우 높은 숙주 핵산 함량을 갖는 샘플 (예를 들어 비장 또는 세포 수가 증가 된 혈액 샘플과 같은 특정 조직의 경우)은 프로토콜 또는 전처리에서 권장되는 최대 샘플 양을 사용하십시오.

번역

[그림 7. 모라벡의 역설, (자료 : 인화원 LG 신입사원교육 AI과정 교재 중, LG CNS 제공)]

사람에게 쉬운 것이 기계에게는 어렵고, 기계에게 쉬운 것은 사람에게는 어렵습니다.

인공지능이 과연 강아지와 베이글을 잘 구별할 수 있을까요?

일부 우리가 어렵게 여기는 일들, 인간은 시간을 들여서 수행해야 하는 일이 있습니다.

예를 들어 복잡한 미적분 계산이나 수식 증명, 금융시장의 다음 수치 예측, 어려운 학술지를 다른 언어로 번역하는 것 등의 작업은 사람에게는 시간도 걸리고 틀릴 가능성도 높지만 컴퓨터는 눈깜짝할 새 매우 쉽고 정확하게 수행할 수 있습니다.

반면 일부 우리가 쉽다고 느끼는 일들, 예를 들어 시각, 움직임, 직감, 맥락 이해는 기계에게 몹시 어려운 과제죠.

컴퓨터 과학자 Donald Knuth가 이런 말을 했습니다.

“인공지능은 이미 모든 생각(연산)이 필요한 영역에서는 인간을 초월했다. 하지만 인간이나 기타 동물이 생각을 하지 않아도 완성할 수 있는 일들에서는 아직 멀었다.”

우리가 쉽다고 느끼는 일들은 사실 매우 복잡한 업무입니다.

그것이 쉽게 느껴지는 것은 동물의 진화과정에서 수억년동안 최적화되었기 때문입니다.

예를 들어 먼 과거, 눈앞의 동물이 맹수인지 가축인지 빠르게 인식해야만 했고, 다리를 더 빨리 움직여 달릴 수 있어야 했고, 구성원간 의사소통을 정확하고 신속하게 하지 못하면 살아남을 수 없었기 때문이죠.

반면에 미적분하기, 번역하기, 시계열 예측하기 등등은 우리가 진화과정 중에 겪어보지 못한, 동물의 입장에선 새로운 일인 셈입니다.

그래서 컴퓨터는 별로 힘들이지 않고도 우리의 수준을 따라잡을 수 있습니다.

하지만 긍정적인 맥락과 부정적인 맥락의 미묘한 차이를 인식하기란... 과연 어떨까요?

#### 2.2.4 사람처럼 사고하기

그렇다면 도대체 사람은 어떻게 다양한 예외상황이 있더라도 긍정적인 문장과 부정적인 문장을 잘 구별할 수 있는 걸까요?

사람들은 긍정, 부정 키워드가 일부 없더라도, 척 보면 이 글이 칭찬을 하는지, 불만을 토로하는지 알 수 있습니다.

애초에 사람은 어떻게 이 둘을 구별할까요?

사람이 긍정 칭찬과 부정 불만(또는 꾸지람) 대해 어떻게 배웠는지를 생각해보겠습니다.

우리는 세상에 처음 태어나서 '좋다/괜찮다/짱이다/예쁘다' 같은 키워드 규칙이라든지, '긍정'이란 단어의 사전적인 정의를 통해 칭찬을 배우진 않았습니다.

여러분이 어린 시절 '혼자서도 잘 수 있다니 우리 OO이 다컸네! 참 잘했어요! 칭찬해줘야겠는걸.'라는 걸 들었다든지 혹은 이웃에게 인사를 할 때 '인사도 참 잘하고 예의바르고 착하구나~'이라는 말을 들었다든지, 그때의 분위기와 사용한 표현을 토대로 칭찬이란 이런 것이구나를 배우게 됩니다.



[그림 8. 사람은 어린 시절 규칙이 아닌 사례를 통해 자연적으로 개념을 습득하곤 한다]

우리는 사전적 정의나 규칙이 아닌, 다양한 여러 사례들을 보고 귀납적으로 긍정 표현과 부정 표현에 대한 정보를 습득합니다.

말로는 구체화하기 어렵지만, 대충 칭찬 문구과 꾸지람 표현은 각각 이러저러한 특징들이 있다는 것을 깨닫습니다.

그렇다면 기계에게도 수많은 사례를 통해 구별하게 하면 다양한 예외에도 잘 대처할 수 있지 않을까요?

있는 그대로의 사례들(raw data)로부터 말로는 할 수 없는, 사전적 정의가 아닌 특징을 배운다! 그것이 사람의 학습 특징입니다.

## 2.3 인공신경망(Artificial Neural Network)

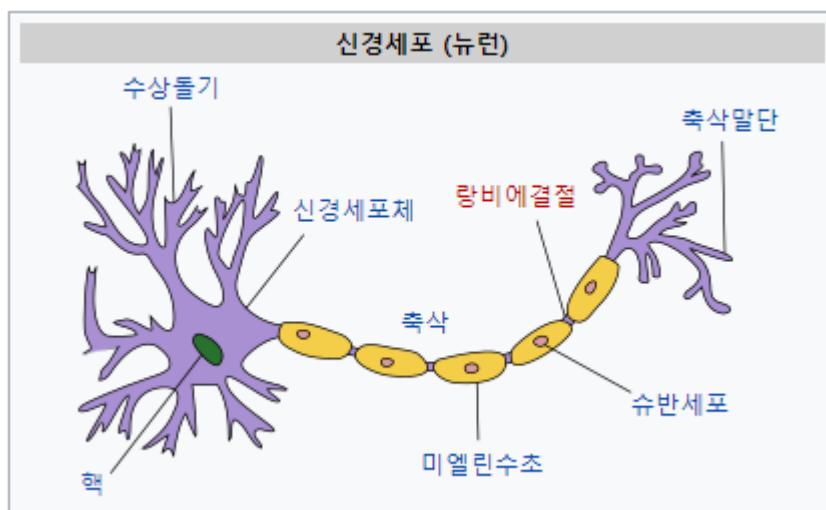
딥러닝은 바로 이러한 인간의 사고방식을 기계학습에 녹였습니다.

사람의 사고방식을 기계에 적용하기 위해 컴퓨터공학과 신경과학의 콜라보가 시작됩니다.

### 2.3.1 인공뉴런(Artificial Neuron)

사람이 어떻게 데이터로부터 정보를 얻고 생각을 하는지 알아보기 위해 사람의 신경계를 잠깐 살펴보겠습니다.

사람에게는 ‘뉴런(Neuron)’이라는 신경 세포가 있어서, 세포의 한 쪽(수상돌기)에서 받아들인 전기 자극 정보를 이런 저런 처리를 한 뒤, 다른 쪽(축삭 말단)에서 다음 뉴런로 전달하곤 합니다.

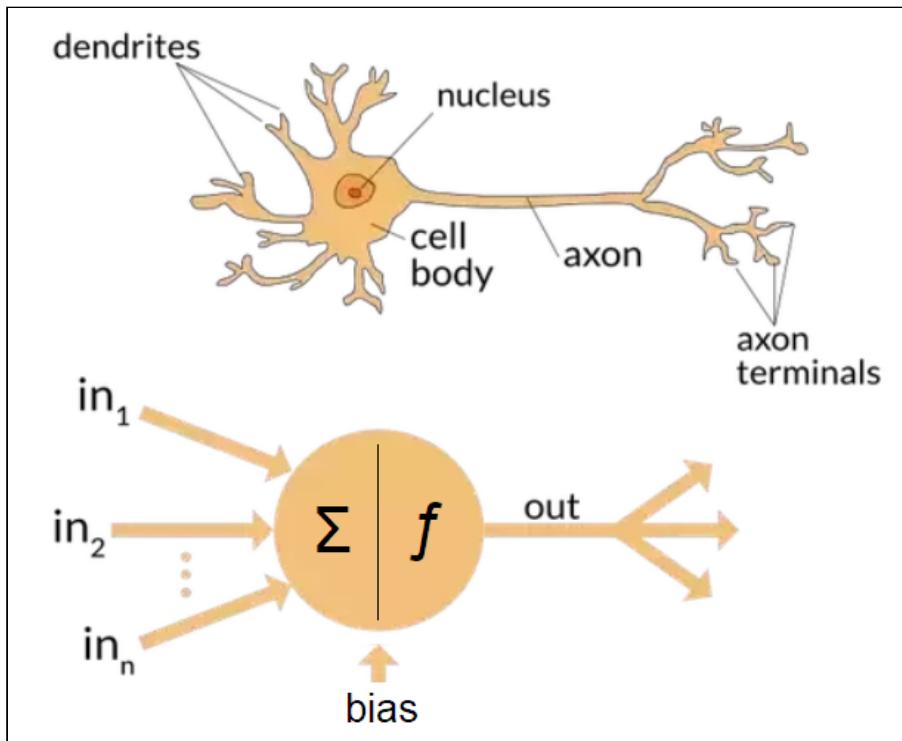


[그림 9. 일반적인 뉴런의 구조<sup>20</sup>. 이러한 뉴런이 여럿 연결되어 신경계를 구성한다.]

그렇다면, 사람이 뉴런으로 데이터를 받아들이고 처리하고 전달하듯 인공적으로 뉴런을 구성하고 여러 층 역어 데이터를 흘린다면 기계도 사람처럼 사고할 수 있지 않을까요?

생물학적 뉴런의 모양을 그대로 본따서 만든 것이 ‘인공 뉴런’으로, 인공뉴런은 이전의 뉴런이 넘겨준 데이터를 받아들여 가중합 연산을 한 뒤, 비선형함수를 적용하여 정보를 가공하여 다음에 이어지는 인공 뉴런으로 데이터를 넘깁니다.

<sup>20</sup> [https://ko.wikipedia.org/wiki/%EC%8B%A0%EA%B2%BD\\_%EC%84%B8%ED%8F%AC](https://ko.wikipedia.org/wiki/%EC%8B%A0%EA%B2%BD_%EC%84%B8%ED%8F%AC)



[그림 10. 뉴런과 인공뉴런의 비교<sup>21]</sup>]

이런 인공 뉴런을 다양한 방식으로 여러 층 쌓아 연결하게 되면 딥러닝의 기본 구조인 ‘인공신경망(Artificial Neural Network)’이 됩니다.

그리고 수많은 긍정문, 부정문을 인공신경망에게 보여주면 긍부정 분류를 위한 최적의 연산 모델을 찾아냅니다.

여기엔 키워드가 뭐가 들어가야 한다든지, 부정어의 순서가 어떻다든지 하는 인간의 지식은 필요 없습니다.

딥러닝 모델은 데이터를 통해 자동으로 필요한 특징(말로는 표현할 수 없지만)을 찾아내고 분류를 수행합니다.

기계 자동화의 패러다임이 아예 바뀌었습니다.

예전엔 어떻게 해서든지 인간이 알고 있는 지식(규칙)을 기계에 전수하려고 했다면, 이제는 인간이 사고하는 방식 그 자체를 기계에게 알려주고 데이터를 제공하는 입장이 되었습니다.

### 2.3.2 딥러닝이 유행하게 된 배경

딥러닝 방식은 길동씨의 프로그래밍 방식과 달리 규칙 없이도 다양한 예외 케이스를 처리할 수 있지만, 쉽게 적용할 수 있는 것은 아닙니다.

인공 뉴런의 모태가 되는 ‘퍼셉트론(Perceptron)’은 코넬 항공 연구소(Cornell Aeronautical Lab)의 프랑크 로젠블라트(Frank Rosenblatt)가 이미 1957년에 고안하였습니다.

하지만 여러 한계들로 인공지능은 길고 긴 암흑기를 맞았으며, 최근에야 여러분이 아시는 것처럼 다시 사람들의 관심을 되찾게 됩니다.

<sup>21</sup> <https://medium.com/towards-artificial-intelligence/deep-learning-series-chapter-2-part-a-18f742260e2a>

인공신경망을 기반으로 하는 딥러닝 모델은 데이터로부터 특징을 스스로 찾아야 하니 양질의 데이터를 다량 확보해야만 높은 성능을 얻을 수 있습니다.

또한 사람처럼 사고하려다보니, 인간의 신경계처럼 복잡한 인공신경망을 구성하고 연산을 수행하고 최적화해야 하는데, 그 규모가 어마어마합니다. 이에 따라 높은 스펙의 하드웨어를 필요로 합니다.

이전에는 이러한 조건을 제대로 받쳐줄 수 없었습니다.

최근 딥러닝이 다시 영광을 찾은 것은 이 두가지의 제약조건을 해결할 방법이 점점 생겨나고 있기 때문입니다.

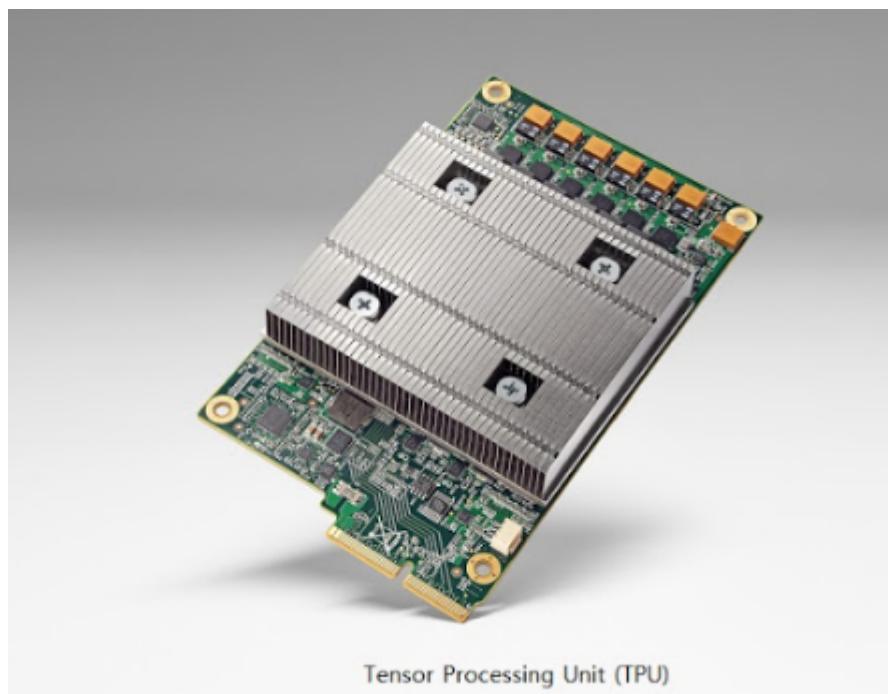
인공지능 붐이 일기 몇 년 전 먼저 발생했던 빅데이터 붐으로 인해 많은 기업들이 데이터의 중요성을 깨닫고 양질의 데이터를 다량 확보하는 데 총력을 기울이고 있습니다.

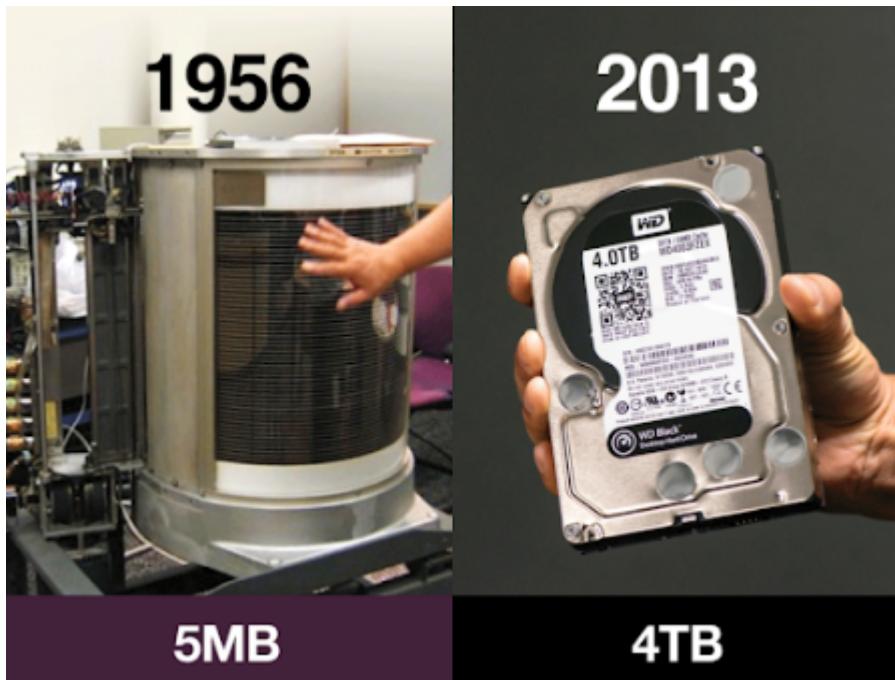
그 중에서도 위에서 설명했듯 특히 비정형 데이터의 비중이 높아지고 있죠.

또한 하드웨어에서도 눈부신 발전이 있었습니다. 더 많은 데이터를 저장할 수 있게 되고 처리 속도도 빨라졌습니다.

GPU의 10배 가량 연산 속도상의 이점을 갖는 TPU(Tensor Processing Unit) 칩이 등장하고, 누구나 손바닥만한 외장하드에 몇 테라바이트씩을 저장할 수 있는 시대가 되었습니다.

클라우드 서비스의 발달로 개인이 서버 장비를 구매하지 않아도 필요할 때 필요한 만큼 사용하고 반납할 수도 있습니다.





[그림 11,12. (왼)구글의 TPU chip, (오)데이터스토리지의 과거와 현재<sup>22</sup>. 1950년대의 하드디스크는 무게 1톤에 달했다]

또 여기에 분산저장처리와 같은 빅데이터 처리 기술의 측면도 한 몫 하고 있습니다.

데이터의 증가와 하드웨어의 발전까지 포함하여, 다양한 조건이 갖춰졌으니 딥러닝의 성능 발전, 앞으로 기대해봐도 되겠죠?

## 2.4 마무리

이번 시간은 정형 데이터를 주로 분석하는 머신러닝 기반의 AI와 비정형 데이터를 대상으로 하는 딥러닝 기반의 AI에 대해 설명드렸습니다.

그리고 딥러닝의 기본이 되는 인공신경망의 개념과 특징을 살펴보았습니다.

다음 시간은 인공지능이 사진이나 그림을 처리하여 판단하는 'Vision AI'에 대해 더 자세히 알아보겠습니다.

감사합니다 😊

---

### 참고자료

- 자사 딥러닝 실무과정 교재보기, [\[교재보기\] 딥러닝 실무<sup>23</sup>](#)

<sup>22</sup> <http://www.itworld.co.kr/print/86540>

<sup>23</sup> <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

- 자사 데이터 엔지니어링 심화과정 교재보기, [교재보기] 데이터 엔지니어링 심화(BI/DW)<sup>24</sup>
- 위키백과-회귀분석, [https://ko.wikipedia.org/wiki/%ED%9A%8C%EA%B7%80\\_%EB%B6%84%EC%84%9D#/media/%ED%8C%EC%9D%BC:Normdist\\_regression.png](https://ko.wikipedia.org/wiki/%ED%9A%8C%EA%B7%80_%EB%B6%84%EC%84%9D#/media/%ED%8C%EC%9D%BC:Normdist_regression.png)
- 위키백과-신경 세포, [https://ko.wikipedia.org/wiki/%EC%8B%A0%EA%B2%BD\\_%EC%84%B8%ED%8F%AC](https://ko.wikipedia.org/wiki/%EC%8B%A0%EA%B2%BD_%EC%84%B8%ED%8F%AC)
- 위키백과-퍼셉트론(Perceptron), <https://ko.wikipedia.org/wiki/%ED%8D%BC%EC%85%89%ED%8A%B8%EB%A1%A0>
- 위키백과-TPU, [https://en.wikipedia.org/wiki/Tensor\\_processing\\_unit](https://en.wikipedia.org/wiki/Tensor_processing_unit)
- Medium, Decision Trees, <https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb>
- Medium, Introduction to Neural Networks and Their Key Elements, <https://medium.com/towards-artificial-intelligence/deep-learning-series-chapter-2-part-a-18f742260e2a>
- IT world, 데이터 스토리지의 과거와 현재, <http://www.itworld.co.kr/print/86540>

---

<sup>24</sup> <http://wire.lgcns.com/confluence/pages/viewpage.action?pageId=62280908>

### 3 [3편] 시각을 얻은 인공지능

---

안녕하세요, CTO AI 빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판](#)(see page 7)에서 연재됩니다.

작성 : CTO AI 빅데이터연구소 AI기술팀 김명지 책임연구원/AI기술팀<sup>25</sup>

---

- 이미지를 인식하는 인공지능(see page 45)
    - 인공신경망 연산(see page 47)
    - 기계가 이미지를 인식하는 방법(see page 47)
  - Convolutional Neural Network(CNN)(see page 50)
    - 특징 추출(Feature Extraction)(see page 51)
    - 태스크 수행(see page 53)
      - Classification(see page 53)
      - Detection(see page 54)
      - Segmentation(see page 55)
  - ImageNet Large Scale Visual Recognition Competition(see page 56)
  - 마무리(see page 58)
- 

지난 시간은 인공지능의 개념에 대해 알아봤습니다.

인공지능에 대한 막연함을 어느정도 해소했으니 이번 시간부터는 가장 다양한 연구가 이루어지는 딥러닝 분야인 '**Vision AI**'에 대해 더 자세히 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter](#)(see page 7)★를 확인하시기 바랍니다.

#### 3.1 이미지를 인식하는 인공지능

'시각'은 동물이 가진 가장 원초적 감각 중 하나입니다.

우리는 태어나서 눈을 뜯 그 순간부터 잠들지 않는 동안은 늘 눈으로 들어오는 각종 시각 정보를 인식하여 판단하고 행동합니다.

---

<sup>25</sup><http://wire.lgcns.com/confluence/display/~78628>



[그림 1. 동물의 가장 원초적 감각인 시각, 3초 안에 덤불 속 맹수를 발견하지 못한다면 살아남기 힘들 겁니다]

### 정답

나무 중심 기준 5시방향, 숨어있는 표범의 얼굴을 찾으셨나요?!

~~물론 발견했다고 해서 살아남을 수 있는 것도 아닙니다.~~

사람은 무의식적으로 아주 빠르게 시각 정보를 처리할 수 있도록 진화했습니다. 그렇게 해야만 더 잘 살아남을 수 있었기 때문이었죠.

예를 들어 초식동물인지 맹수인지, 나무 넝쿨인지 뱀인지, 독버섯인지 식용버섯인지를 구별해야 하는 등 시각은 생존과 직결된 문제였습니다.

시각이란 사람에게는 매우 익숙한 감각입니다만 기계도 과연 시각 정보를 인식할 수 있을까요?

오늘은 기계에게 이미지 데이터로부터 정보를 인식하도록 하는 방법을 소개하겠습니다.

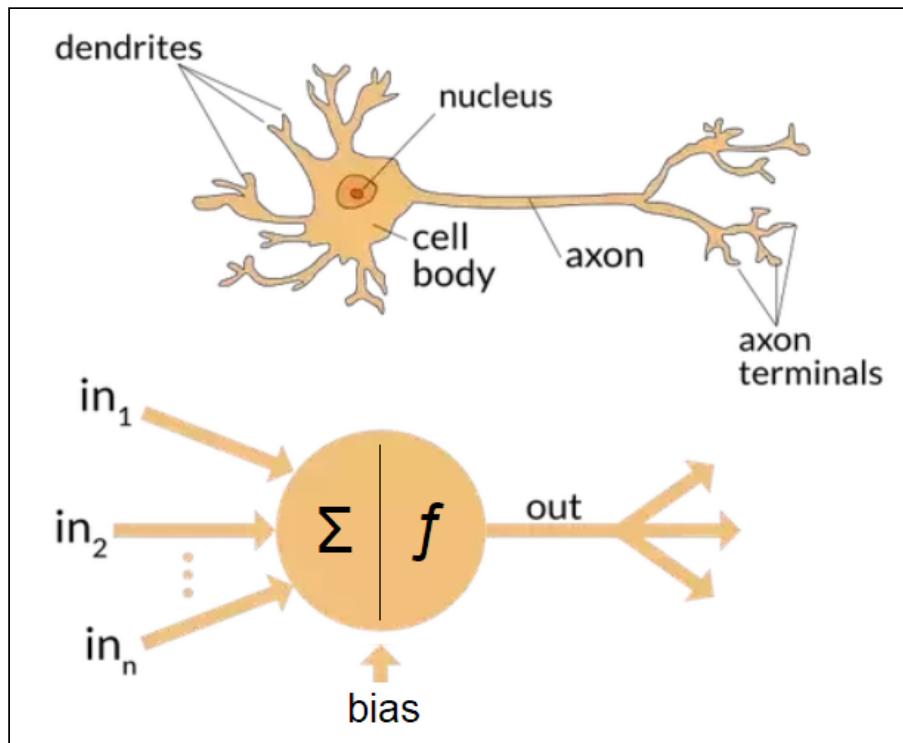
### 3.1.1 인공신경망 연산

이미지 데이터를 설명드리기 이전에 지난시간에 소개한 인공 뉴런을 잠깐 되돌아보도록 하겠습니다.

#### [2편] 중, 인공뉴런과 인공신경망

생물학적 뉴런의 모양을 그대로 본따서 만든 것이 ‘인공 뉴런’으로, 인공뉴런은 이전의 뉴런이 넘겨준 데이터를 받아들여 가중합 연산을 한 뒤, 비선형함수를 적용하여 정보를 가공하여 다음에 이어지는 인공 뉴런으로 데이터를 넘깁니다.

이런 인공 뉴런을 다양한 방식으로 여러 층 쌓아 연결하게 되면 딥러닝의 기본 구조인 ‘**인공신경망(Artificial Neural Network)**’이 됩니다.



딥러닝 모델은 인공뉴런을 여럿 연결한 인공신경망을 기반으로 합니다.

그리고 인공신경망은 가중합과 비선형 함수로 이루어진 연산을 수행해야 합니다.

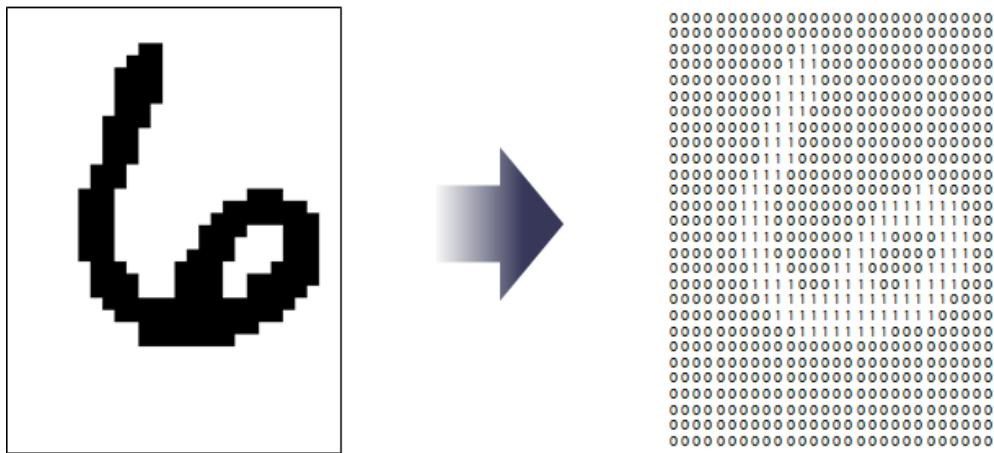
즉, 입력 데이터로 벡터나 행렬과 같은 형태를 필요로 합니다. 그래야 수학 연산이 가능하기 때문입니다.

그렇다면 색상과 곡선, 각도, 도형, 명암이 어우러진 이미지 데이터를 어떻게 처리해야 인공신경망에 입력할 수 있을까요?

### 3.1.2 기계가 이미지를 인식하는 방법

JPG, PNG 파일 등으로 구성된 이미지 데이터는 우리의 눈으로 보기에는 하나의 정지된 그림이지만 기계는 이를 연산 가능한 형태로 만들어 인식합니다.

예를 들어, 아래의 흑백 이미지 데이터는 사람이 보기엔 숫자 6이지만 기계가 보기에는 가로와 세로 28픽셀씩을 갖는 2차원 행렬 데이터입니다.



[그림 2. 숫자 흑백 이미지 데이터, 왼:사람이 보는 이미지, 오:컴퓨터가 인식하는 이미지]

픽셀 값이 1이면 진한 색, 즉 이미지에서 색이 칠해진 부분이고 0이면 밝은 색, 즉 하얀 배경에 해당하는 값입니다. 이미지를 행렬로 변환했기 때문에 이제는 인공신경망 연산을 할 수 있게 됩니다.

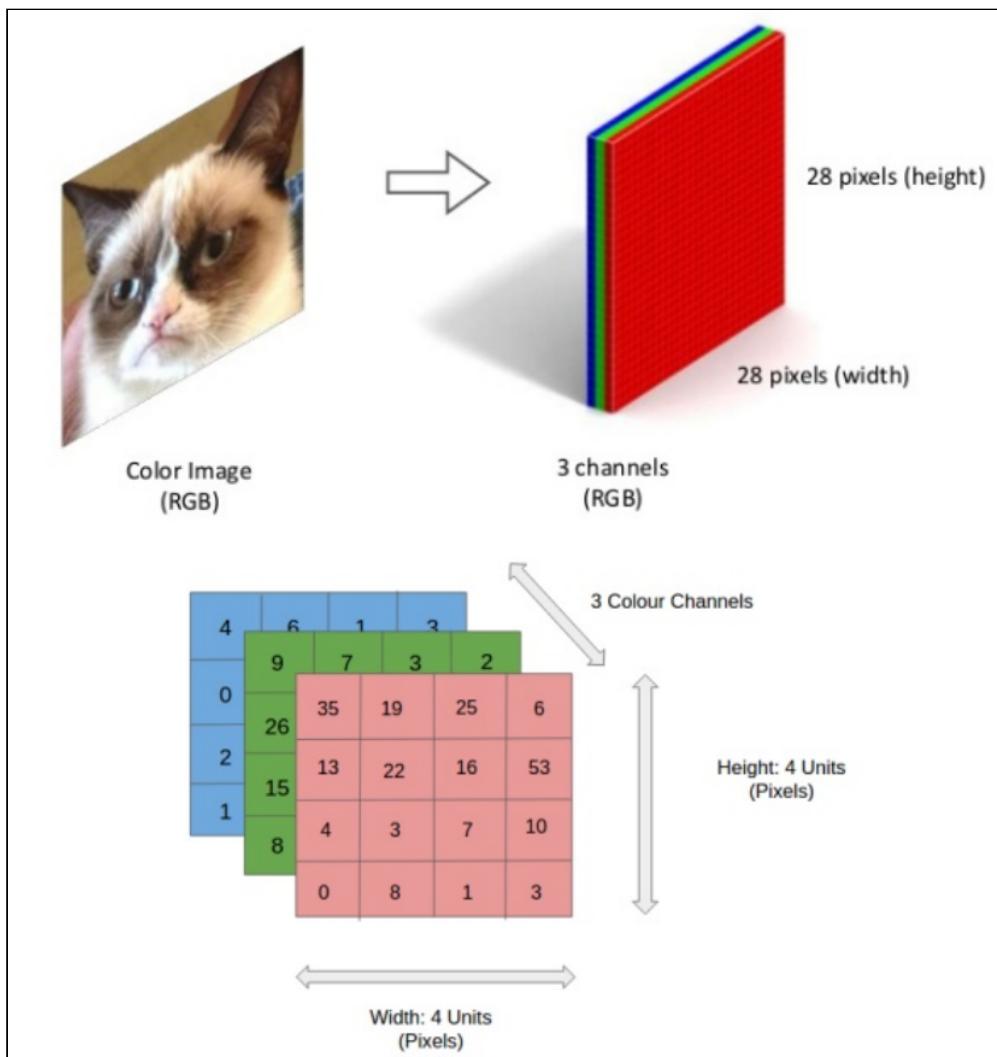
인공신경망은 이 픽셀 값을 가지고 가중합 및 비선형 처리를 다양한 방식으로 수행하여 그림의 숫자가 무엇인지 인식하게 됩니다.

컬러 이미지인 경우도 비슷합니다. 단 컬러는 2차원의 행렬(matrix)이 아닌 3차원 텐서(tensor)를 다루게 되는데요, 컬러를 가진 픽셀 하나를 표현하기 위해선 RGB(Red, Green, Blue)의 세 값이 필요합니다.

즉 Red 값을 나타내는 매트릭스 하나, Green 값을 나타내는 매트릭스 하나, Blue 값을 나타내는 매트릭스 하나씩이 필요합니다.

이를 3 channel, 또는 3 depth라고도 표현합니다.

어디선가 '3채널 이미지'라고 한다면 픽셀당 RGB 세 개의 값을 갖는 컬러 이미지를 말한다고 볼 수 있습니다.



[그림 3. 컬러 이미지는 3차원의 Tensor로 인식된다]

그림판에서 색상을 찾을 때 나오는 값 세개를 보신 적이 있으실겁니다. 그 때의 값이 컬러 픽셀 하나를 나타내는 정보라고 생각하시면 됩니다.



[그림 4. 색상표에서 선택한 보라색 컬러는 R 139, G 53, B 203의 값을 가진다]

하지만 어떤 이미지가 2차원 매트릭스냐, 3차원 텐서냐 하는 것은 중요한 건 아닙니다.

중요한 것은 사람이 보는 시각 정보를 인공지능은 계산 가능한 데이터로 바꾸어 인식한다는 것이죠.

그리고 이렇게 입력된 데이터는 인공신경망 연산이 가능해진다는 것입니다.

### 3.2 Convolutional Neural Network(CNN)

계산가능한 픽셀값으로 이미지 데이터를 읽을 수 있으니 이제는 이미지로부터 정보를 찾아내어 과제를 수행해야겠죠?

이미지란 가로, 세로의 공간적 정보를 담고 있으니 이를 처리하기 위한 특별한 인공신경망 구성 방식이 필요합니다.

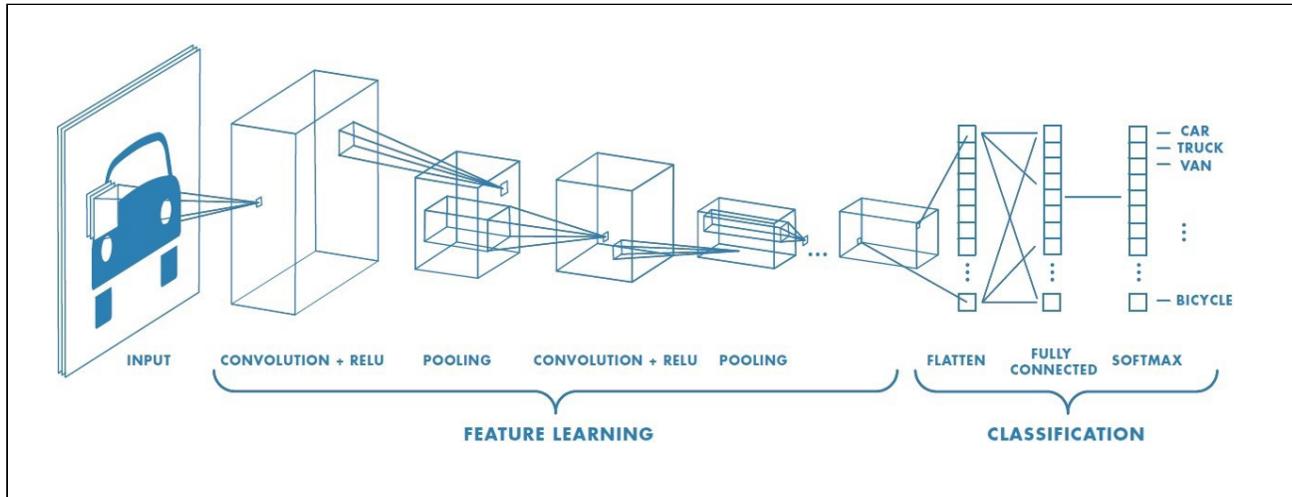
여기서 이미지 처리에 특화된 신경망, CNN(Convolutional Neural Network)을 소개해드립니다.

CNN은 이미지로부터 특징을 추출하는 (1) Feature Extraction 영역과, 특정 태스크 수행을 위해 덧붙이는 (2) 태스크 수행 영역 두 가지로 구성되어있습니다.

예를 들어 교통수단으로 탈 것의 사진을 입력받아 그 종류를 자동으로 구별하는 인공지능을 만든다고 가정하겠습니다.

컬러, 외관 골격, 바퀴 모양 등을 파악하는 역할은 (1) 부분이, (1)에서 추출된 정보를 활용하여 탈 것의 카테고리를 K 개로 구별하는 역할은 (2) 부분이 수행합니다.

각 영역에서 어떤 일이 일어나는지를 설명드리겠습니다.



[그림 5. CNN의 구성, 이미지로부터 탈 것을 자동 분류하는 예]

### 3.2.1 특징 추출(Feature Extraction)

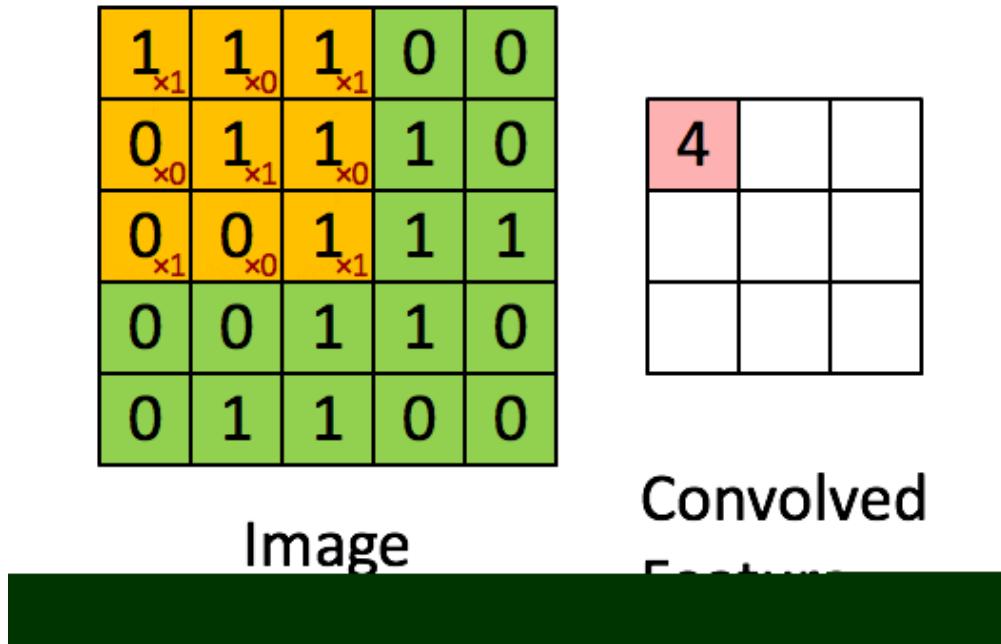
이미지로부터 특징을 추출하는 역할은 컨볼루션(Convolution) 연산과 풀링(Pooling) 연산이 수행합니다.

컨볼루션 연산은 컨볼루션 필터(또는 커널)가 입력 이미지를 상하좌우로 훑으며 주요한 특징이 있는지 찾아내는 과정입니다.

컨볼루션 필터를 여러 개 설정하면 그만큼 이미지로부터 다양한 특징을 찾아낼 수 있습니다.

이렇게 찾아낸 결과 특징을 Feature map(또는 convolved feature)라고 부릅니다

특징을 찾는 작업은 이전에 소개드린 인공 뉴런이 하는 일과 마찬가지로, 가중합 + 비선형 함수 적용으로 구성됩니다.



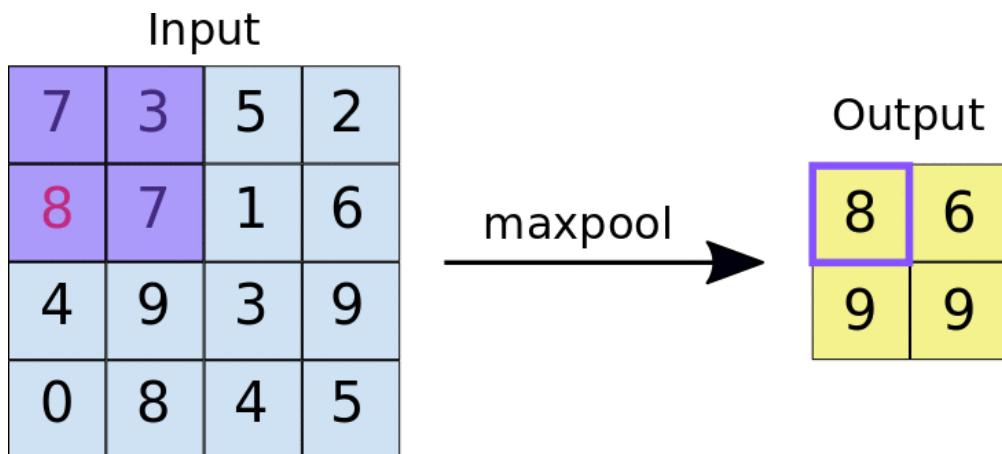
[그림 6. 컨볼루션 연산의 구성]

위 그림의 초록색  $5 \times 5$  매트릭스가 입력 이미지, 노란색  $3 \times 3$  매트릭스가 이미지를 훑는 컨볼루션 필터 1개, 분홍색  $3 \times 3$  매트릭스가 컨볼루션 연산의 출력결과(feature map)가 됩니다.

가중합 연산이 어떻게 일어나는지 하나하나 짚어보실 필요는 없습니다. (물론 이해에는 도움이 되겠지만요)  
중요한 것은 컨볼루션 연산이 이미지로부터 특징을 추출하는 역할을 한다는 것입니다.

컨볼루션 필터가 이미지를 상하좌우로 훑으며 특징을 찾아낸 뒤, 이 결과로부터 정보를 추리는 풀링 연산이 이어집니다.

**풀링 연산**은 Feature map을 역시 상하좌우로 훑으며 핵심적인 정보만을 영역별로 샘플링하는데요,  
주로 영역 내 가장 큰 값만을 남기고 나머지 값을 버리는 MaxPooling 방식을 적용합니다.



[그림 7. 풀링 연산의 구성]

위 그림의 하늘색 **4x4 매트릭스**가 컨볼루션 결과인 feature map, 보라색 2x2 영역이 풀링 대상 영역, 붉은 글씨가 MaxPooling으로 선택된 값, 노란 2x2 매트릭스가 풀링 결과입니다.

컨볼루션 연산이 이미지의 특징을 찾아낸다면 풀링 연산은 그 중 핵심정보만 남깁니다.

대부분의 이미지 처리 모델에서는 컨볼루션과 풀링 연산을 여러 번 반복하면서 데이터의 feature를 추려냅니다.

이미지로부터 특징을 배워나가는 작업이라는 뜻에서 이 과정을 'Feature Learning'이라고 부릅니다.

### 3.2.2 태스크 수행

이미지로부터 주요 특징을 찾아냈다면 이 정보를 활용하여 목표로 하는 태스크를 수행해야 합니다.

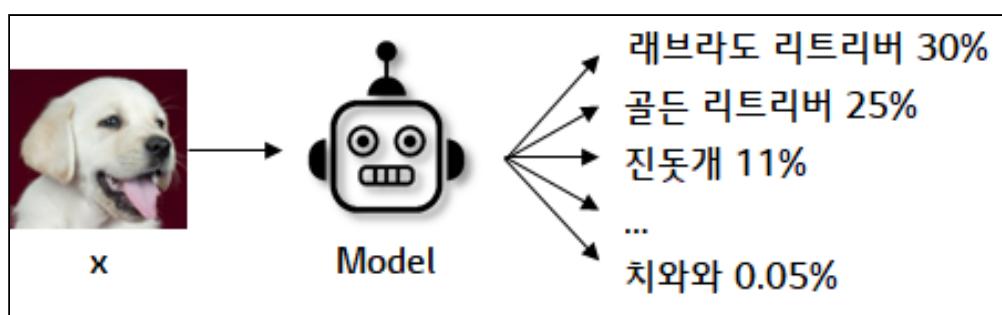
이 부분의 구조나 필요한 연산의 종류는 하고자 하는 태스크의 종류에 따라 다릅니다.

이미지 데이터를 처리하는 대표적인 태스크를 몇가지 소개해드리겠습니다.

#### 3.2.2.1 Classification

입력으로 받은 이미지를 지정된 K개의 클래스(또는 카테고리) 중 하나로 분류하는 과제입니다.

강아지와 고양이를 분류하거나, 공장에서 제품 사진을 보고 양호/불량을 판별하는 업무 등에 쓰일 수 있습니다.



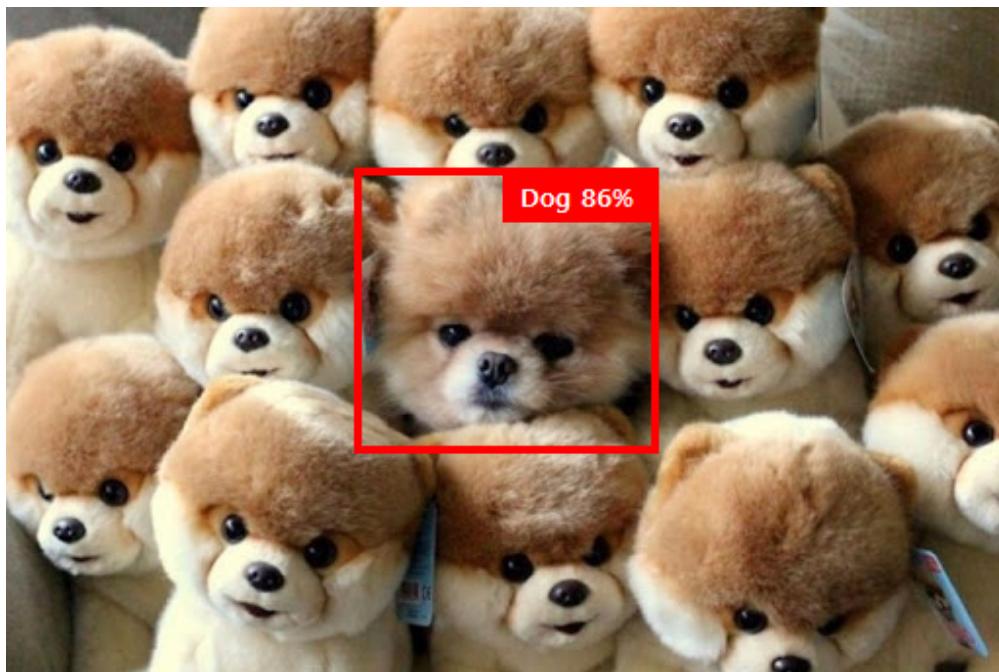
[그림 8. Classification 예, 강아지 사진을 입력받아 견종을 구별하는 모델]

### 3.2.2.2 Detection

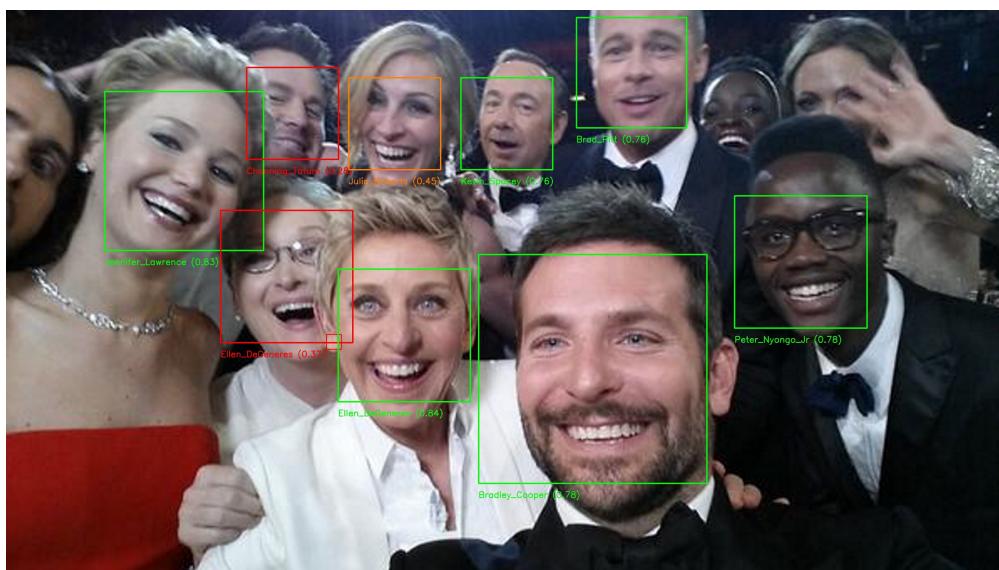
입력으로 받은 이미지에서 특정 개체가 어디에 위치하는지, (x, y) 좌표값을 찾아주는 과제입니다.

스마트폰으로 사진을 찍을 때 자동으로 인물의 얼굴에 네모박스를 치고 포커스하는 기능을 본 적 있으시죠?

이 경우 얼굴을 detect하는 기능이 탑재되어 있겠네요.



[그림 9. Detection 예, 강아지를 detect하는 모델]



[그림 10. Multi-class detection 예, 한 사진으로부터 다양한 유명인 얼굴을 detect하는 모델]

### 3.2.2.3 Segmentation

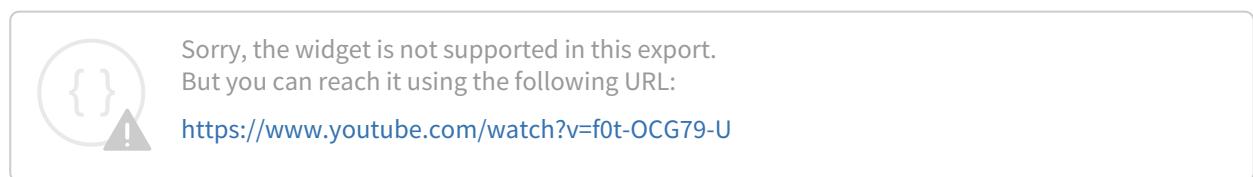
Detection이 네모 박스로 개체를 찾아준다면 Segmentation은 조금 더 정밀하게, 픽셀 단위로 영역을 구별해줍니다. 경계를 잘 찾는 작업은 Detection보다 훨씬 어렵겠죠?



[그림 11. Segmentation 예, 도로 위 다양한 객체의 영역을 구별하는 모델]

#### 참고 : CNN의 visualization

단순한 CNN 모델의 흐름을 잘 시각화해놓은 영상이 있어 소개해드립니다.



0:00 Convolution 연산 (3x3 필터 2개, 순차적으로 한장씩 입력 이미지를 훑음)

0:58 비선형 함수 연산 (이 경우 양수값만을 남기는 ReLU activation function이 적용됨)

1:02 Pooling 연산 (2x2 MaxPooling)

1:09 Convolution 연산 (3x3 필터 2개)

1:27 비선형 함수 연산 (ReLU activation function)

1:30 Pooling 연산 (2x2 MaxPooling)

1:33 추출한 feature를 활용하여 5 class 분류

### 3.3 ImageNet Large Scale Visual Recognition Competition

전세계적으로 열리는 유명한 이미지 인식 대회가 있습니다.

ImageNet이라는 데이터셋에 대한 분류 성능을 겨루는 ILSVRC(ImageNet Large Scale Visual Recognition Competition)입니다.

이 대회에 출전하는 인공지능 모델은 입력받은 ImageNet 데이터를 무려 1,000개 카테고리 중 하나로 분류해야 하는 과제를 수행합니다.

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

**Classification task:** produce a list of object categories present in image. 1000 categories.  
**"Top 5 error":** rate at which the model does not output correct label in top 5 predictions

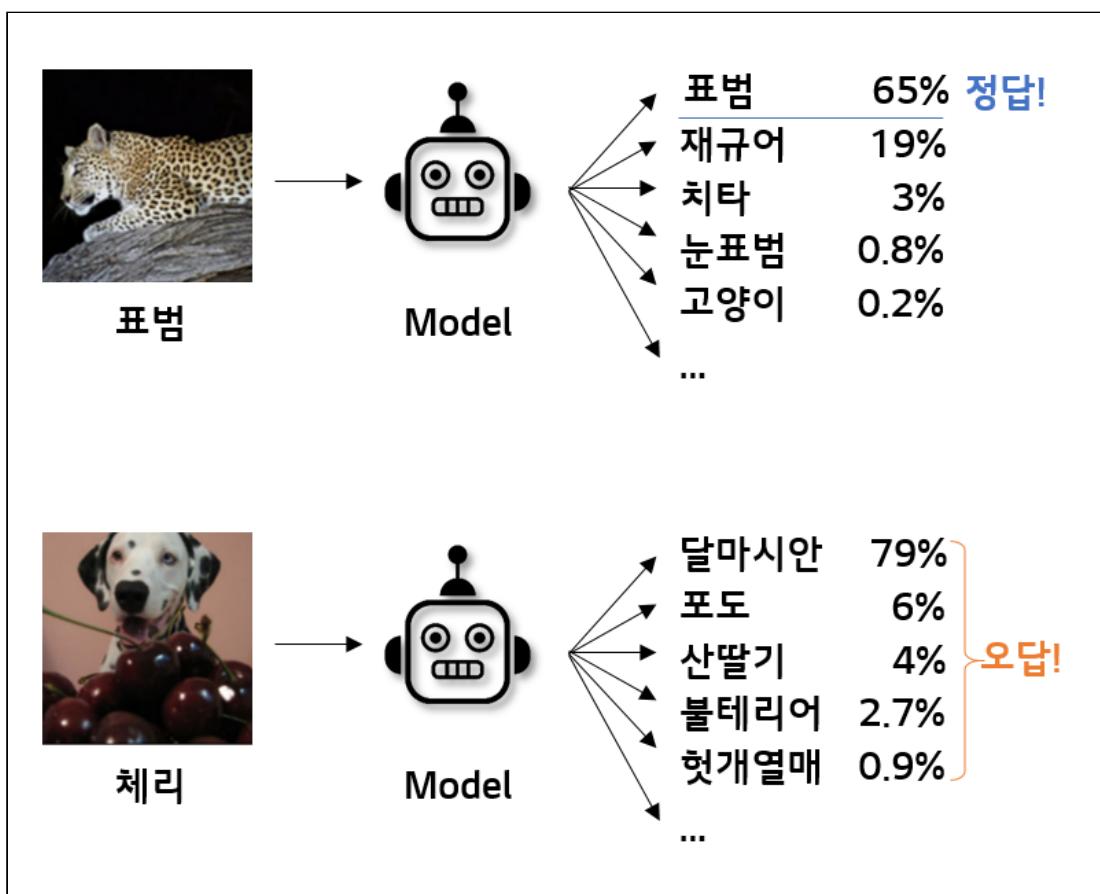
Other tasks include:  
single-object localization, object detection from video/image, scene classification, scene parsing

[그림 12. 이미지넷 데이터 분류 대회]

어떤 인공지능이 더 이미지를 잘 인식하는지에 대한 기준은 Top-5 에러율로 정합니다.

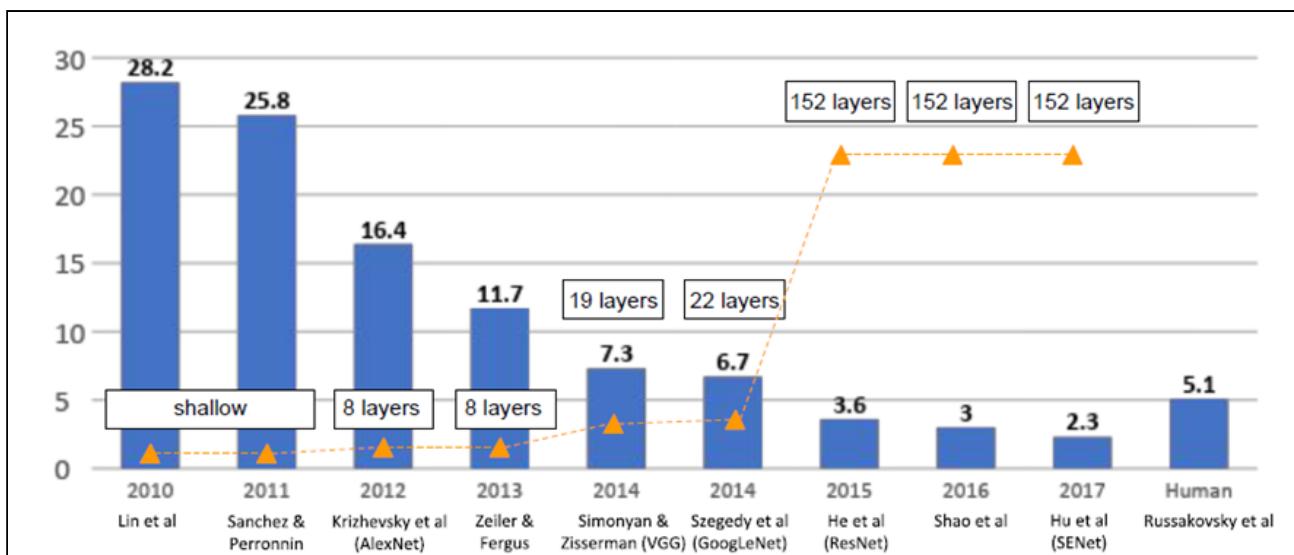
인공지능 모델은 하나의 이미지에 대해 예상되는 후보 카테고리를 순서대로 5개 추론해야하는데요,

이 5개의 후보 중 정답 카테고리가 없다면 틀린 개수 +1을 하여 10만장의 테스트 이미지에 대해 몇 건이나 틀렸는지를 따지는 방식입니다.



[그림 13. Top-5 정답 인정 케이스와 과 Top-5 오답 케이스]

이 대회에 출전한 모델 중 역대 우승한 모델 이력을 살펴보겠습니다.



[그림 14. ILSVRC 연도별 우승 모델]

2011년까지는 전통적인 컴퓨터비전 처리 방식으로 출전한 모델이 우승을 했습니다.

사람이 수작업으로 찾아낸 이미지의 특징(Hand-crafted feature)을 활용하는 방법입니다.

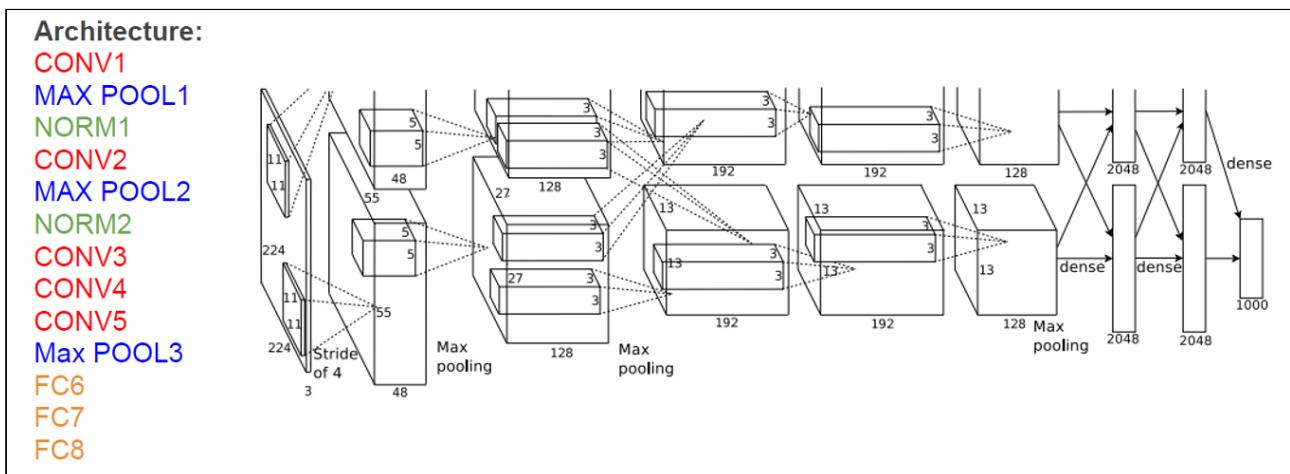
이때까지만 해도 Top-5 에러율이 25%를 웃도는 수준이었습니다.

즉, 모델이 추론한 후보 5개 중 정답이 없는 케이스가 1/4 빈도로 발생했습니다. 기계가 이미지를 인식하는 데 있어서 아직 갈 길이 멀었죠.

참고로 사람의 Top-5 에러율은 5.1%였습니다.

그런데 이후 2012년, 토론토 대학의 Alex Krizhevsky라는 연구자가 처음으로 딥러닝 계열 모델로 대회에 출전했고 Top-5 오류율을 10%가량 크게 개선시키며 1위를 차지했습니다.

이 때의 우승한 AlexNet이라는 딥러닝 모델은 위에 소개시켜드린 CNN으로 구성되어있습니다.



[그림 15. AlexNet 모델의 아키텍처. 그림은 복잡해보이나, 컨볼루션 연산과 폴링 연산으로 구성되어있습니다.]

AlexNet은 학계 연구자들에게 "딥러닝이 이렇게까지 할 수 있다고?"라는 가능성을 보여준 계기가 되었습니다.

2012년부터 딥러닝의 전성기라 불리는 시기가 시작되었죠.

그래서인지 이후의 ILSVRC 대회 우승은 전부 CNN 기반의 딥러닝 모델이 차지했습니다.

그리고 2015년, 무려 지금으로부터 5년 전에 이미 인공지능 모델이 사람의 이미지 인식률을 뛰어넘었습니다. ([그림 14]의 ResNet)

몇년 뒤, 더 이상은 성능이 개선될 여지가 없다고 판단, 주최측은 2017년을 이후로 대회의 막을 내렸습니다.

이후로는 더 어려운 이미지 인식을 위한 인공지능 대회가 열리고 있습니다.

적어도 이미지 분류에 있어서, 이제는 인공지능이 사람보다 인식을 더 잘한다고 볼 수 있겠네요.

### 3.4 마무리

아직까지 사람처럼 복합적인 과제를 수행할 수는 없지만, 단순한 수학 연산을 엮은 구조만으로 기계가 이정도로 이미지 인식을 할 수 있다는 것은 놀랍습니다.

어떻게 본다면 0과 1의 디지털 데이터만을 다룬던 기계에게 세상을 볼 수 있는 '눈'이 생긴 것이라고 볼 수도 있습니다.

기술이 조금 더 발전한다면 사람처럼 시각 인지를 바탕으로 행동하거나 다양한 사고를 추론하는 것도 가능해지겠죠?

이번 시간은 이미지 데이터를 입력받아 분석하는 인공신경망 기법인 CNN에 대해 설명드렸습니다.

그리고 대표적인 이미지 인식 과제 종류와 글로벌 인공지능 이미지 인식 대회(ILSVRC)를 살펴보았습니다.

다음 시간은 '인공지능이 텍스트 데이터를 처리하는 방법'에 대해 알아보겠습니다.

감사합니다 😊

## 참고자료

- 자사 딥러닝 실무과정 교재보기, [교재보기] 딥러닝 실무<sup>26</sup>
- A brief survey of tensors, <https://www.slideshare.net/BertonEarnshaw/a-brief-survey-of-tensors>
- A Comprehensive Guide to Convolutional Neural Networks, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Towards a Meaningful 3D Map Using a 3D Lidar and a Camera, [https://www.researchgate.net/publication/326875064\\_Towards\\_a\\_Meaningful\\_3D\\_Map\\_Using\\_a\\_3D\\_Lidar\\_and\\_a\\_Camera](https://www.researchgate.net/publication/326875064_Towards_a_Meaningful_3D_Map_Using_a_3D_Lidar_and_a_Camera)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC), <http://www.image-net.org/challenges/LSVRC/>
- ImageNet classification with deep convolutional neural networks, Alex Krizhevsky, et al., 2012
- Very deep convolutional networks for Large-Scale Image Recognition, Karen Simonyan & Andrew Zisserman, 2014
- Going deeper with convolutions, Christian Szegedy, et al., 2014
- Deep Residual Learning for Image Recognition, Kaiming He, et al., 2015
- Densely connected convolutional networks, Gao Huang, et al., 2016

<sup>26</sup> <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

## 4 [4편] 언어를 이해하는 인공지능

---

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/AI기술팀<sup>27</sup>

---

- 언어를 인식하는 인공지능(see page 60)
  - 자연어이해(NLU; Natural Language Understanding)(see page 64)
    - 자연(Natural)(see page 64)
    - 언어(Language)(see page 64)
    - 이해(Understanding)(see page 67)
  - 기계에게 사람의 언어를 인식시키려면?(see page 68)
    - Tokenizing(Parsing)(see page 68)
    - 워드임베딩(word embedding)(see page 69)
      - 원-핫 인코딩(One-hot Encoding)(see page 69)
      - CBOW와 SKIPGRAM(see page 70)
  - 다양한 자연어이해 과제들(see page 74)
    - 문장/문서 분류(Sentence/Document Classification)(see page 74)
    - Sequence-to-Sequence(see page 75)
    - 질의 응답(Question Answering)(see page 76)
  - 마무리(see page 77)

지난 시간에는 Vision AI에 대해 다루었습니다.

인공지능에게 세상을 보는 눈이 생긴 셈인데요, 동물의 원초적 감각인 시각을 기계에게 구현하는 게 가능해졌습니다.

이번 시간에는 인간의 전유물이라고 여겼던 '언어'를 인식하는 인공지능에 대해 더 자세히 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 4.1 언어를 인식하는 인공지능

옛날의 인류는 정보를 교환하기 위해 사람과 사람의 의사소통에 의존할 수밖에 없었습니다.

문자가 생긴 뒤로는 정보를 기록할 수 있게 되었고, 사람들은 궁금한 점을 책 등의 기록물로부터 해결할 수 있었습니다.

---

<sup>27</sup><http://wire.lgcns.com/confluence/display/~78628>

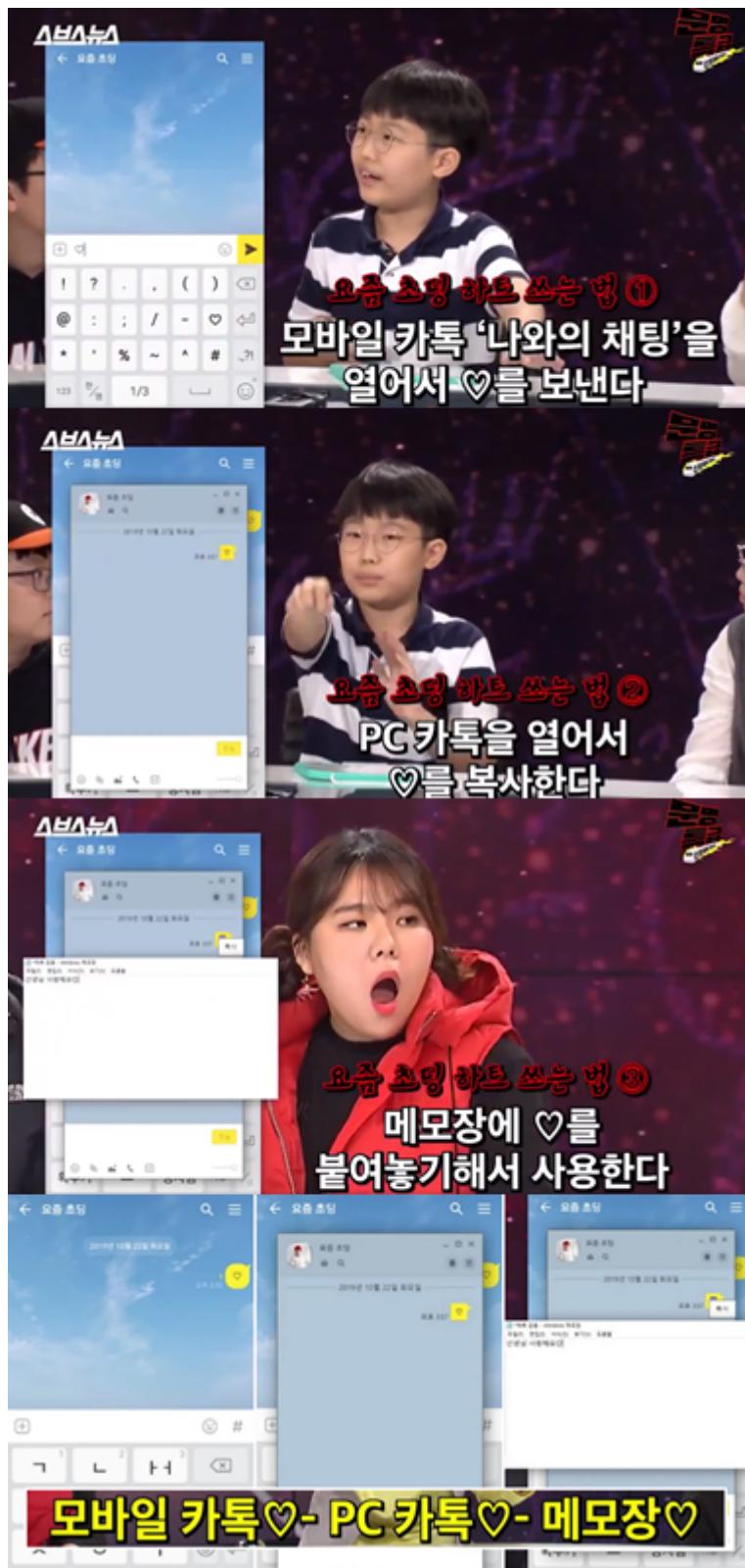


[그림 1. 인류 최초의 기록문자 (기원전 26세기 수메르인들의 쐐기 문자)]

요즈음의 우리는 궁금한 것이 생기면 책이나 주변 지인에게 묻기보다 검색엔진을 활용합니다.

검색엔진은 한 권의 책이나 한 명의 사람이 담을 수 있는 이상의 지식을 훨씬 빠르고 폭넓게 제공합니다.

미래엔 어떻게 될까요? 밀레니얼 세대 다음은 Z세대라고 합니다.



[그림 2. 요즘 세대가 PC에서 ♥를 쓰는 방법]

지금의 초등학교 저학년정도인 이들은 컴퓨터보다도 모바일 기기에 더 익숙합니다.  
이들은 질문이 생기면 AI스피커에게 음성으로 물어봅니다(짱구야~ OO에 대해 알려줘~).



[그림 3. 요즘 아이들이 숙제를 하는 방법]

답변을 구하는 것에서 조차 점점 사람의 수고를 줄이는 방향으로 기술이 발전하고 있습니다.

AI 스피커에 물어본다면 컴퓨터나 모바일 기기를 켜서 자판을 치고 검색할 필요도 없이 바로 대화로 답변을 얻을 수 있죠.

하지만 집에 AI 스피커가 있어서 이것저것 테스트해보신 분들은 단순 질문답변, 그 이상의 것은 기대하기 어렵다는 것을 공감하실겁니다.

AI 스피커와 한 문장씩 주고받는 대화는 '대화'라고 하기엔 너무나 기초적인 수준입니다.

근미래에 기계와 정말로 자연스럽게 대화를 하며 우리의 질문에 대한 답변을 얻을 수 있을까요?

아니 그 이전에, 기계가 사람의 언어를 올바로 인지하고 맥락을 이해할 수 있을까요?

AI 스피커는 우리가 무엇을 궁금해하는 지 어떻게 인식하고 답변을 해준 걸까요?

#### 4.1.1 자연어이해(NLU; Natural Language Understanding)

자연어이해, 또는 자연어처리(NLP; Natural Language Processing)라는 말을 많이 들어보셨을 겁니다.

말 그대로 자연어를 이해하거나 처리하는 기능에 대한 말인데요, 주체는 '기계'입니다.

기계가 자연어를 이해한다는 것은 어떤 걸까요? 명칭에 대해 잠깐 짚고 넘어가겠습니다.

##### 4.1.1.1 자연(Natural)

자연어(Natural Language; 自然語)란, 사람들이 일상적으로 쓰는 언어로, 기원을 찾기 힘들며 자연 발생한다는 특징이 있습니다.

예를 들어 우리가 쓰는 한국어나 영어, 중국어, 일본어 등이 그 예입니다.

반대되는 개념으로는 인공어(Constructed Language; 人工語)가 있습니다.

이는 의도와 목적에 따라 인공적으로 만든 언어로, 프로그래밍 언어와 같은 기계어라든가 전자구인의 공용 언어 사용을 위해 만들어진 에스페란토 등이 이에 해당합니다.

“팀장님 이번 회식 소고기집 가도 돼요?”  
“이번 달 법카 얼마 남았어?”

```

1 def is_ok(menu, card) :
2     if card.limit > menu.price :
3         return "OK"
4     else :
5         return "SORRY"
6
7 print is_ok(menu("소고기"), get_team_card(2018,5))
8

```

[그림 4. 자연어와 인공어]

##### 4.1.1.2 언어(Language)

'언어'라는 데이터 타입이 갖는 특성은 무엇일까요?

언어는 말하고 듣는 음성과, 쓰고 읽는 문자로 이루어져 있습니다.

예를 들어 여러분이 영화 '아이언맨'을 친구에게 설명해준다고 가정해봅시다.

친구는 영화 내용을 전혀 모르기 때문에 여러분의 설명으로만 내용을 상상할 겁니다.

어떻게 영화의 줄거리를 요약할 수 있을까요?

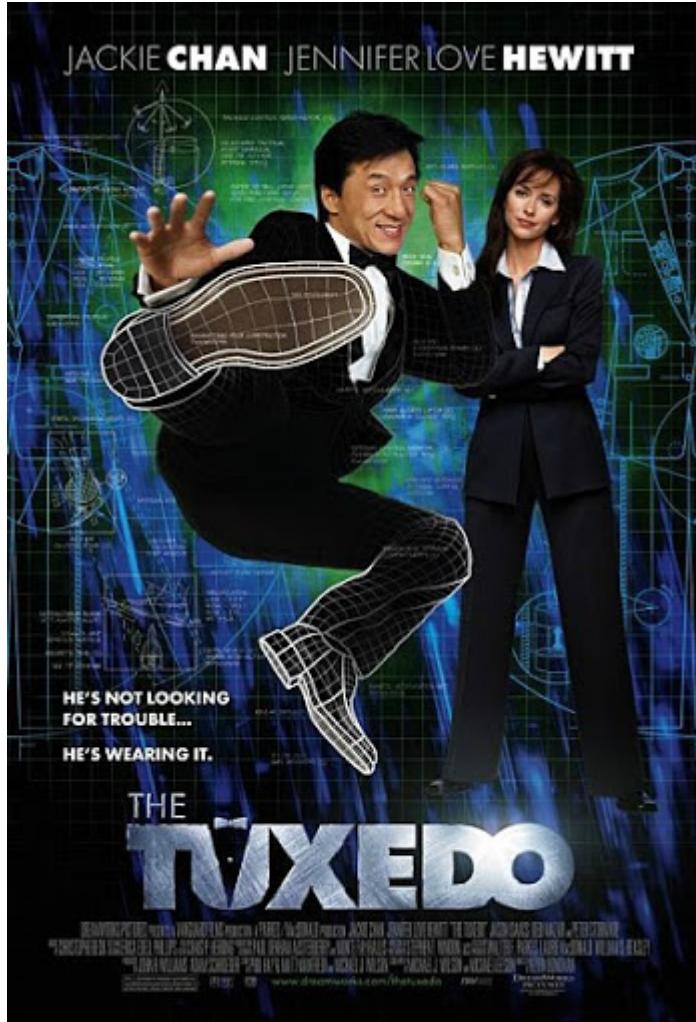
억만장자 천재 발명가인 토니 스타크가 심장에 치명적인 상처를 입은 자신의 목숨을 지키며  
아이언맨 강화 수트를 제작한다. 세계를 지킬 과학의 결정체인 아이언 맨은 범죄와 싸워나간다.

친구는 한 줄 줄거리만으로 내용을 상상할 수 있을까요?

아마 과학의 결정체인 강화수트라는게 뭔지부터 혼란을 겪을거라고 생각합니다.

어쩌면 '수트'라는 말로부터 과거 성룡이 출연한 영화 '턱시도'같은 의상을 생각할 수도 있구요..

참고:턱시도



하지만 영화 포스터나 스틸컷을 보여줄 수 있다면 어떨까요?

아 강화수트라는게 저런 로봇같은 수트를 말하는거였구나, 주인공이 저렇게 생겼나보네, 딱보니 가운데 있는 놈이 악당인가보다..

더 많은 정보를 얻을 수 있을겁니다.



[그림 5. 아이언맨 영화포스터]

아니면 아예 영화를 보여줄 수 있다면요? 설명할 수고도 줄어들고, 훨씬 더 깊게 이해할 수 있습니다.



[그림 6. 뭐야... 수트를... 본인이 입는다는거였어...?]

백 번 듣는 것보다 한 번 보는 것이 낫다는 말도 있습니다.

영화라는 다차원 데이터(동영상)는 많은 정보를 포함할 수 있지요.

반면 텍스트 형식으로 표현된 언어 데이터는 그 많은 정보를 굉장히 함축하여 인코딩한 자료입니다.

사람은 언어가 표현하는 대상의 내용을 머리속에서 3D로 자유자재 구현할 수 있지만 기계는 그러한 사전정보가 없습니다.

기계가 언어를 인식하고 맥락에 포함된 내용을 이해하기란 쉽지 않겠죠?

#### 4.1.1.3 이해(Understanding)

이해한다는 것은 처리한다는 것보다 한단계 더 높은 수준을 요구합니다.

처리는 기계적으로 규칙을 따라 얼마든지 수행 가능하겠지만 이해는 맥락 내용에 대한 파악을 전제로 합니다.

요즘은 자연어 처리(NLP)나 자연어 이해(NLU)라는 용어를 거의 구분없이 사용하고는 있습니다.

하지만 오늘날 인공지능에게 기대하는 것은 단순 규칙에 따른 처리가 아닌, 내용과 맥락을 이해하여 업무를 수행하는 것이므로 '자연어 이해'라는 표현을 이용하도록 하겠습니다.

엄밀히 나누자면 NLP 안에 NLU가 포함됩니다.

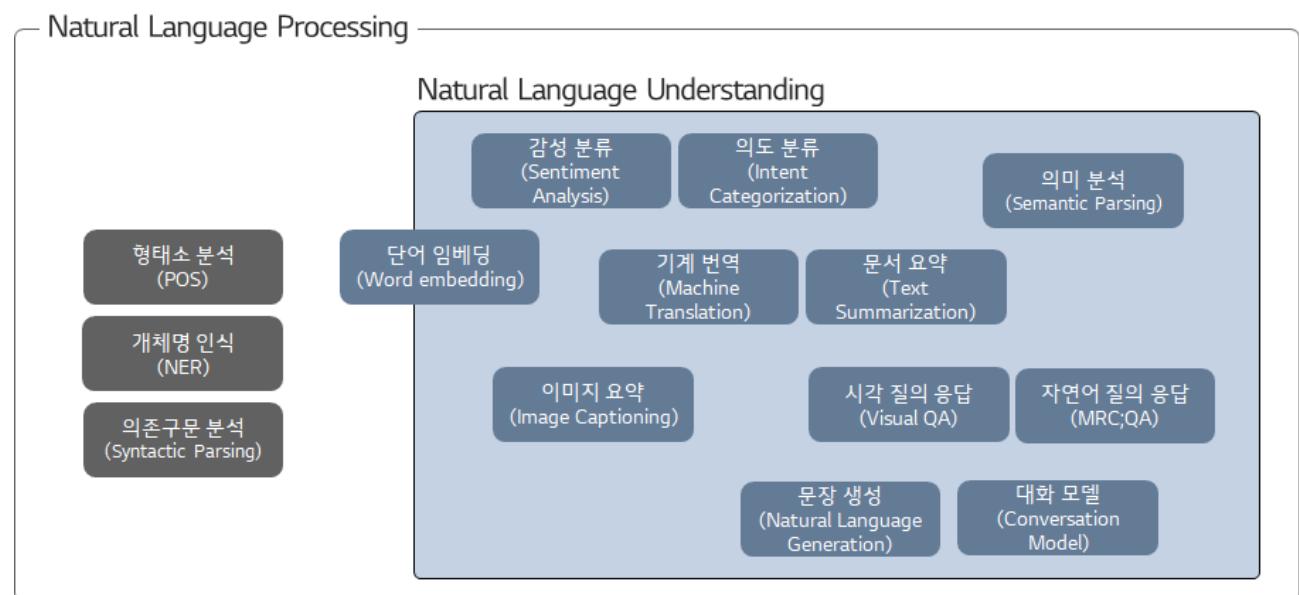
NLP에만 해당하는 부분은 언어의 형식, 구문론과 관련된 기능들입니다.

문장을 형태소 단위로 쪼갠다든지(POS), 구문 분석을 수행한다든지 하는 기능입니다.

반면 NLU는 문장 내 표현된 감성은 인식하는 감성분류라든지, 문서 내 중요한 부분을 캐치하는 요약 등의 기능을 포함합니다.

그런데 이마저도 요즘은 딥러닝 기술의 발달로 NLP/NLU의 경계가 애매해지고 있습니다.

전통적인 NLP영역쪽도 딥러닝 기반의 기술을 적용했을 때 더 좋은 성능이 나오고 있거든요.



[그림 7. NLP와 NLU, 하지만 최근 두 경계를 나누는 것은 상당히 모호하다]

어쨌든 기계가 언어를 이해하기 위해서는 일단 언어를 인식해야합니다.

우리가 배웠던 인공신경망에 데이터를 넣어 딥러닝 모델을 학습시키는 방식으로요.

하지만 인공신경망에 데이터를 태우려면 데이터의 생김새가 인공 뉴런이 계산(가중합+비선형함수)할 수 있는 형태여야 합니다.

텍스트로 된 언어 데이터를 계산할 수 있을까요?

(참고: 음성의 경우, STT(speech-to-text) 기술을 활용하여 텍스트로 변환 가능합니다)

## 4.2 기계에게 사람의 언어를 인식시키려면?

언어를 계산 가능한 형태로 바꾸려면 어떻게 해야 할까요?

인공신경망에 태울 데이터는 실수로 이루어진 벡터나 매트릭스 등의 타입이어야 합니다. [지난 게시물<sup>28</sup>](#)의 이미지 데이터처럼요!

우리는 이러한 다차원의 실수 데이터를 텐서(Tensor)라고 하겠습니다.

이미지 데이터는 픽셀값을 2차원, 또는 3차원의 텐서로 만들어 인공신경망에 태울 수 있었습니다.

언어의 경우도 텐서로 바꾸어 인식시킬 수 있습니다. 예를 들어 간단한 문장 하나를 예시로 들겠습니다.

**여름휴가는 시작하기 전이 가장 즐겁다**

### 4.2.1 Tokenizing(Parsing)

한 덩이로 되어있는 문장을 인공신경망에 인식시키기 위해서 세부 단위로 쪼개는 작업이 필요합니다.

이를 토크나이징 또는 파싱이라고 하며, 이 쪼개진 단위를 토큰(token)이라 부릅니다.

어떻게 쪼개면 좋을지는 언어의 특징과 수행하고자 하는 태스크, 데이터의 특징에 따라 다릅니다.

어절

여 + 름 + 휴 + 가 + 는 + 시 + 작 + 하 + 기 + 전 + 이 + 가 + 장 + 즐 + 겁 + 다

형태소

여름휴가/NNG + 는/JX + 시작/NNG + 하/XSV + 기/ETN + 전/NNG + 이/JKS + 가장/MAG + 즐겁/VV + 다/E

음절

여 + 름 + 휴 + 가 + 는 + 시 + 작 + 하 + 기 + 전 + 이 + 가 + 장 + 즐 + 겁 + 다

자소

ㅇ + ㅋ + ㄹ + ㄱ + ㅁ + ㅎ + ㅠ + ㄴ + ㄱ + ㅏ + ㄴ + ㅡ + ㄴ + ㅣ + ㅓ + ㅈ + ㅏ + ㅋ + ㅌ + ...

[그림 8. 여러 방식에 따른 문장 토크나이징 예]

영어의 경우에는 어절, 즉 띄어쓰기 단위로 잘라도 좋고 중국어의 경우에는 한 문자 한 문자에 뜻이 담겨있기 때문에 글자 단위로 쪼갤 수 있습니다.

일반적으로 한국어의 경우엔 교착어라는 언어 특성상 문장을 형태소 단위로 자르거나 음절단위로 자릅니다. (참고: [교착어<sup>29</sup>](#))

뉴스기사나 논문처럼 정제된 글인 경우 형태소 단위로 잘 쪼개질 수 있지만 채팅이나 SNS 글의 경우 맞춤법을 제대로 지키지 않고 작성하는 경우가 많아 음절 단위로 쪼개는 것이 좋습니다.

28 <https://tec.lgcns.com/tec/pages/viewpage.action?pageld=90359347>

29 <https://ko.wikipedia.org/wiki/%EA%B5%90%EC%B0%A9%EC%96%B4>

요새는 이 둘의 장점을 취한 토크나이징 기법(등장빈도가 높은 N-gram 단위의 토크나이징 등)도 많이 활용합니다.

### 어절

- 한국어에 적합 X

### 형태소

- 정제된 글에 강하고 맞춤법 파괴, 신조어 등에 취약
- 형태소 수만큼 vocabulary list를 가짐
- 형태소 분석기에 따라 결과 달라질 수 있음

### 음절

- vocabulary list 13,000개 수준
- 오타에 덜 민감

### 자소

- vocabulary list 몇 백 개 수준



[그림 9. 데이터의 특징에 따라 잘 맞는 토크나이징 단위]

## 4.2.2 워드임베딩(word embedding)

이제 한 덩이의 문장이 토큰들로 쪼개졌습니다.

쪼개진 토큰들은 인공신경망이 계산할 수 있도록 벡터로 바꾸어줘야 하는데요, 토큰을 벡터화하는 것을 워드임베딩이라고 부릅니다.

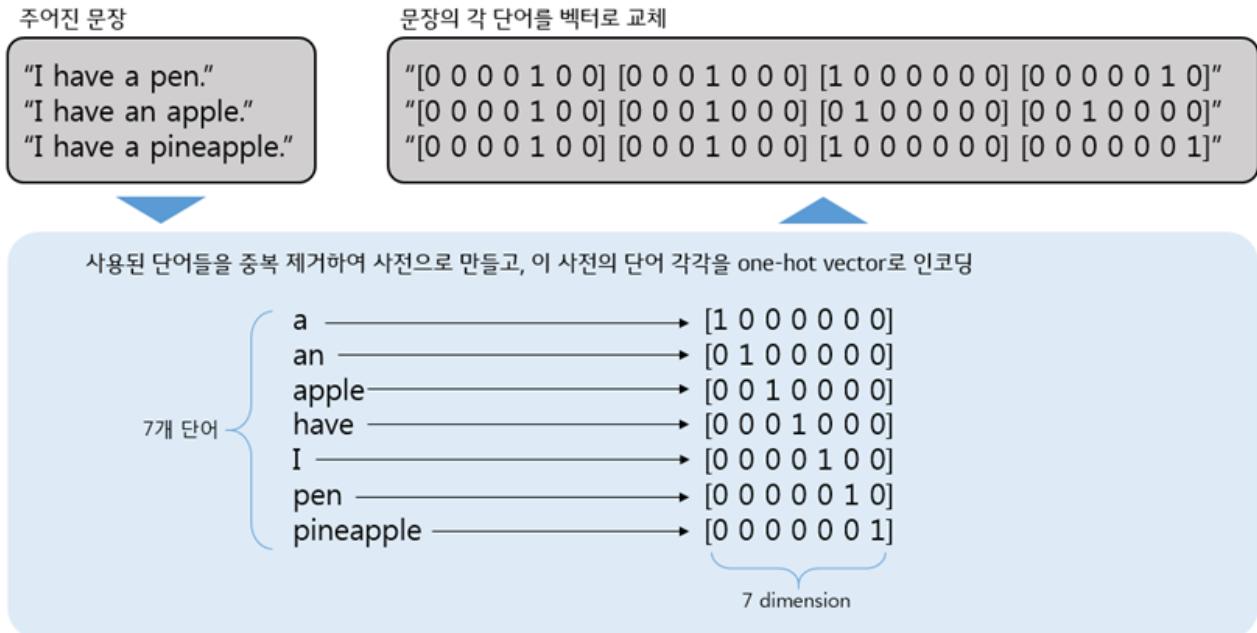
우리가 읽을 수 있는 토큰(보통 단어 단위)을 다차원의 벡터 공간의 한 점으로 매핑한다, 꽂아 넣는다는 의미에서 워드 임베딩이라는 표현을 씁니다.

토큰을 벡터화하는 워드임베딩 기법에는 여러가지가 있습니다.

### 4.2.2.1 원-핫 인코딩(One-hot Encoding)

토큰을 벡터화하는 가장 쉽고 직관적인 방법은 우리가 알고있는 모든 토큰들을 쭈루룩 줄세워 사전을 만들고, 순서대로 번호를 붙이는 작업입니다.

1. 내가 가진 모든 문장, 문서들을 토크나이징한 뒤 토큰들의 중복을 제거한 사전을 만듭니다.
2. 사전의 길이(중복 제외 토큰 수)만큼 벡터를 정의하고, 벡터에서 토큰의 순서에 해당하는 위치만 1, 나머지는 0의 값으로 채워 벡터를 구성합니다.
3. 내가 가진 문장, 문서의 토큰들을 해당 벡터로 교체합니다.



[그림 10. 세 영어 문장을 기준으로 원-핫 인코딩 변환을 하는 작업 순서도, 이 경우 토크나이징 단위는 어절(띄어쓰기 단위)]

하지만 원-핫 인코딩 방식은 아주 간단하긴 합니다만 큰 문제가 있습니다.

토큰이 다양하고 수가 많을수록 토큰 하나를 표현하기 위해서 굉장히 길이가 긴 벡터를 필요로 합니다.

한국어 어휘 표준국어대사전 기준 51만개의 단어가 있다고 합니다.

한 단어를 표현하기 위해서는 무려 51만 길이를 갖는 벡터를 활용해야합니다.

그리고 그 벡터의 대부분은 0의 값으로, 길이에 비해 가진 정보는 턱없이 부족한 편이죠(very sparse).

#### 참고: 왜 임베딩을 1,2,3,... 으로 하지 않고 [1,0,0,...], [0,1,0,...]과 같이 하나요?

그렇게 했을 때 스칼라 값 하나로 모든 토큰을 표현할 수는 있겠지요.

하지만 각 토큰들은 서로 대등하며, 동일한 양의 정보를 가져야 합니다.

예를 들어 [그림 10]의 'apple' 토큰을 3, 'pineapple' 토큰을 7이라고 한다면 어떻게 될까요?

신경망에서 파인애플이 사과보다 +4만큼 더 크게 계산되어야 할 어떤 수치적인 우월함도 없습니다.

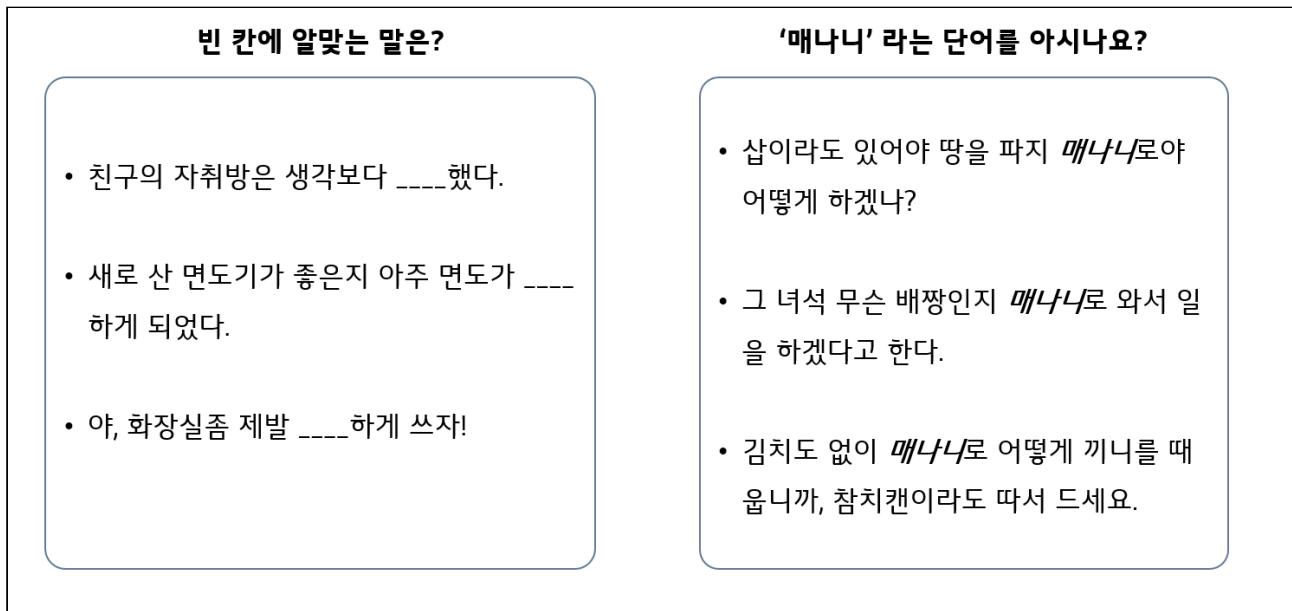
원-핫 인코딩 방식으로 토큰을 임베딩하게되면 모든 토큰간  $\sqrt{2}$ 만큼의 서로 동일한 거리를 갖습니다.

이로서 원-핫 인코딩 방식은 여러 토큰을 동등하게 취급합니다.

그래서 이를 개선하기 위해, 벡터의 길이도 작으면서 정보도 많이 담긴(dense) 두 가지 방식이 나왔습니다.

#### 4.2.2.2 CBOW와 SKIPGRAM

다음 두 예를 보겠습니다.



[그림 11. CBOW와 SKIPGRAM 알고리즘 학습 방식을 비유한 사례, [자료<sup>30</sup>](#)]

첫 번째 예제의 빈 칸에 들어갈 말이 뭐가 있을까요?

정답이 있는 건 아니지만, 아마 '깨끗', '깔끔', '청결' 등등이 빈칸에 들어갈 수 있는 단어가 되겠습니다.

이들은 뉘앙스는 조금 다를지라도 의미는 비슷합니다.

두 번째 예제도 살펴보겠습니다.

'매나니'라는 단어를 아시나요?

자주 쓰는 단어는 아니기 때문에 아마 모르는 분이 대다수일겁니다.

하지만 매나니가 정확히 뭔지는 몰라도, 사용 예시를 보면 대충 맨땅에서, 밑바닥부터, 맨손으로... 뭔가 그런 의미를 담은 말이겠죠?

사람은 언어를 이렇게 배웁니다.

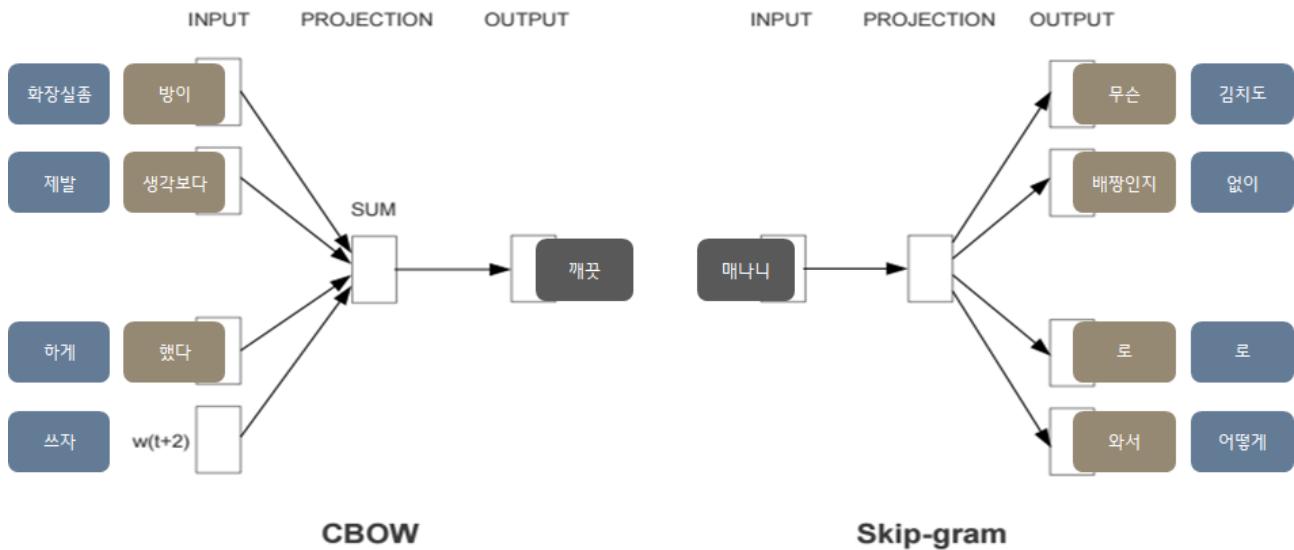
우리는 모든 단어를 사전을 뒤져가며 정확한 의미를 습득하는 게 아닙니다.

비슷한 문맥에서 비슷한 위치에 등장하는 단어끼리는 유사한 의미를 가진다는 걸 알 수 있으니 '청결'이라는 단어를 만일 몰라도, '깨끗'이 나을만한 위치에 쓰인다면 그 뜻을 대강 알 수 있습니다.

또 '매나니'라는 단어의 사전적 정의를 몰라도 등장하는 문맥을 여럿 보면 대강의 의미를 뽑아낼 수 있죠.

이런 과정을 알고리즘으로 구현한 것이 각각 CBOW와 SKIPGRAM입니다.

<sup>30</sup> <https://www.lucypark.kr/docs/2015-pyconkr/#20>



[그림 12. CBOW와 SKIPGRAM]

둘 중 어떤 방식을 이용하든, 먼저 단어(토큰)를 특정 길이를 가진 임의 벡터로 만들어줍니다.

이 때의 벡터 길이는 사람이 지정하며, 원-핫 인코딩 방식보다는 훨씬 작은 길이입니다. (일반적으로 몇십~몇백 단위 정도)

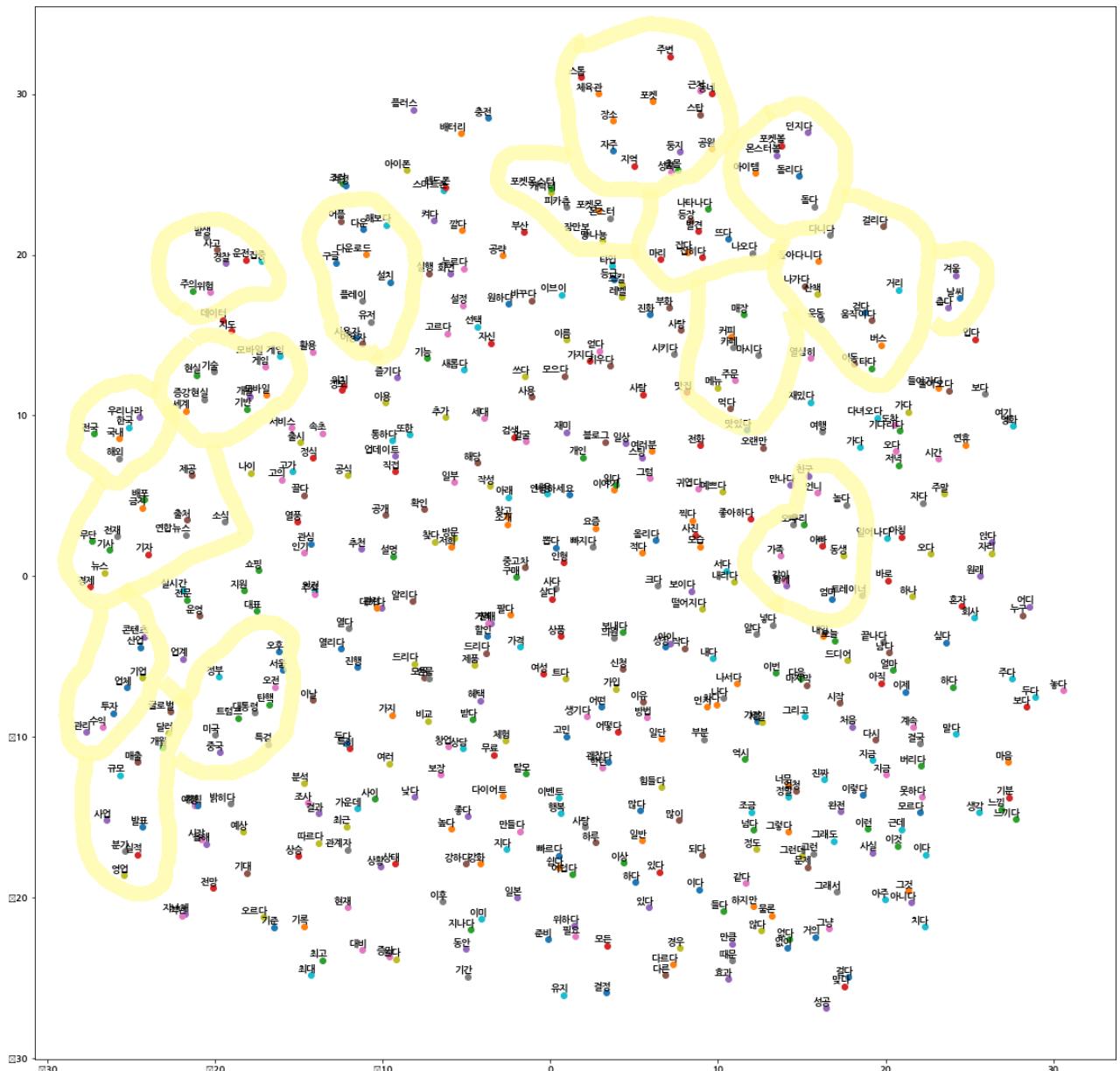
그 뒤, CBOW 방식은 인공지능에게 문장을 알려주되 중간중간 빈칸을 만들어 들어갈 단어(토큰)을 유추시킵니다.

Skip-gram 방식은 인공지능에게 단어(토큰) 하나를 알려주고, 주변에 등장할만한 그럴싸한 문맥을 만들도록 시킵니다.

많은 문장을 학습시키면 시킬수록 더 좋은 품질의 벡터가 나옵니다.

아래 예시는 '포켓몬고' 게임이 출시될 당시의 소셜 데이터로 학습한 워드임베딩 결과를 시각화한 것입니다.

학습 방식은 CBOW 활용, 길이 300의 임베딩 벡터를 PCA를 통해 2차원으로 변환하였습니다.



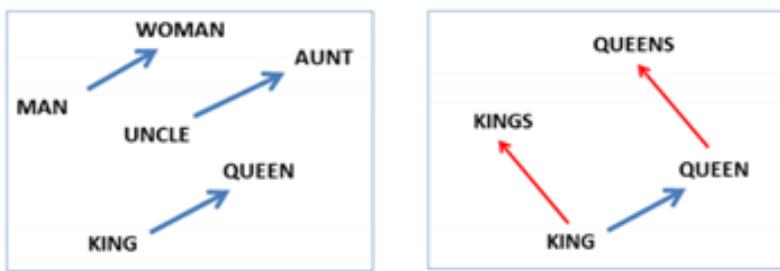
[그림 13. 포켓몬고 워드임베딩 시각화(AI기술팀), 클릭시 크게 보입니다]

시기하게도 지역과 관련된 단어, 비슷한 동작과 관련된 단어들이 윤사한 벡터 공간으로 임베딩되었습니다.

즉, 유사한 의미를 갖는 토큰은 유사한 벡터값을 가지도록 학습이 되 것이지요.

원-핫 인코딩에 비해 벡터의 길이가 훨씬 작아(51만 → 300) 저장공간을 효율적으로 사용할 뿐만 아니라, 비슷한 의미의 토큰이 인공신경망에서 비슷한 연산결과값을 가지도록 하는 효과도 얻을 수 있습니다.

놀라운 점은 토큰끼리의 의미 연산이 가능하다는 것인데요, 이를테면 'LG전자'-'LG'+'삼성'='삼성전자' 같은 벡터 연산이 가능합니다.



[그림 14. 잘 학습된 워드벡터는 의미 연산이 가능하다]

이해를 돋기 위해, 한국어 벡터에 대해 연산을 해볼 수 있는 사이트가 있어 소개해드립니다. ([링크<sup>31</sup>](#))

이 세가지 전통적인 워드임베딩 기법 외에도 맥락을 고려하여 조금 더 스마트하게 벡터화하는 다양한 기법들이 최근에 많이 등장했습니다.

기회가 되면 다른 포스트에서 추가로 더 다루도록 하겠습니다 😊.

### 4.3 다양한 자연어이해 과제들

텍스트를 벡터로 바꾼 뒤라면 인공신경망을 어떻게 구성하느냐에 따라 다양한 태스크를 수행할 수 있습니다.

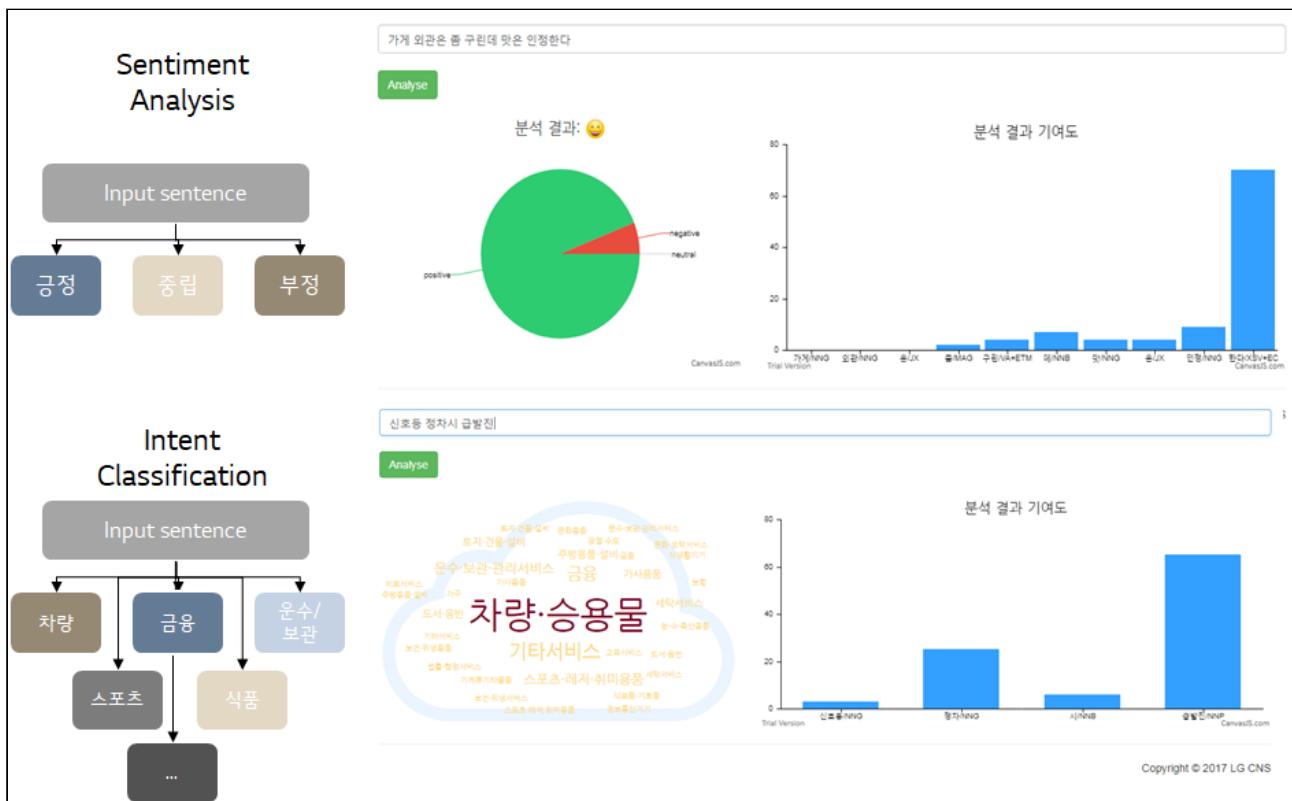
대표적인 자연어이해 태스크 몇 가지를 소개해드리겠습니다.

#### 4.3.1 문장/문서 분류(Sentence/Document Classification)

입력받은 텍스트를 지정된 K개의 클래스(또는 카테고리) 중 하나로 분류하는 과제입니다.

사용자 리뷰를 감성분석(긍/부정)하거나, 유저의 발화문을 챗봇이 처리할 수 있는 기능 중 하나로 매핑하는 의도분류 등에 쓰일 수 있습니다.

<sup>31</sup> <https://word2vec.kr/search/?query=%EA%B8%B0%EA%B3%84-%EC%B2%A0%2B%ED%94%BC%EB%B6%80>



[그림 15. 감성분석과 의도분류]

### 4.3.2 Sequence-to-Sequence

문장/또는 문서를 입력으로 받아 문장을 출력하는 과제입니다.

한 나라의 언어를 다른 나라로 옮기는 번역과제라든지, 긴 문서를 핵심 문장으로 추리는 요약과제가 이에 해당합니다.

이외에도 자유 대화 등 다양한 과제에 활용 가능합니다.



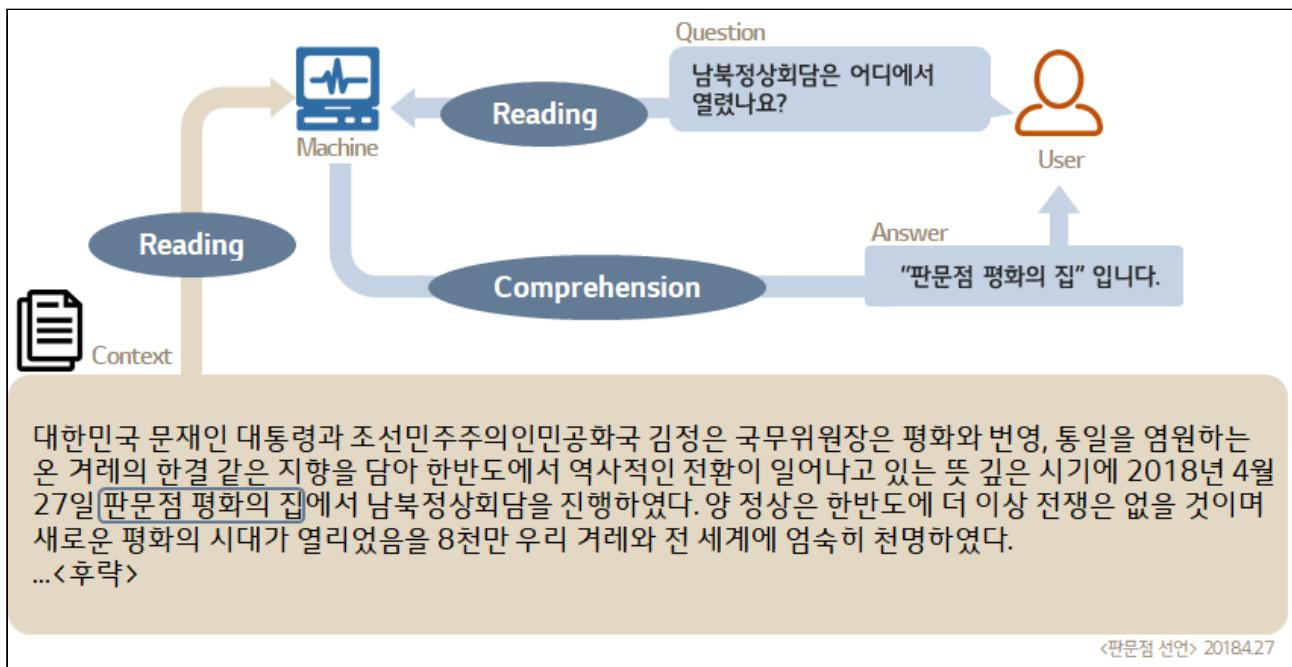
[그림 16. 기계번역과 뉴스요약]

### 4.3.3 질의 응답(Question Answering)

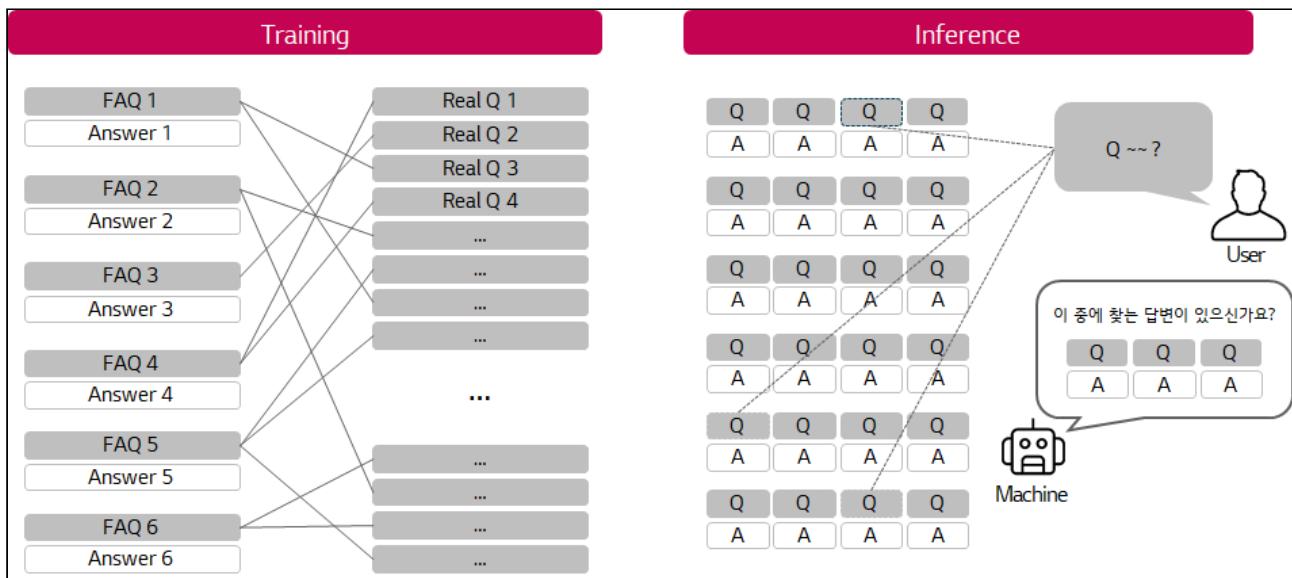
사용자 질문이 들어오면 내가 가진 매뉴얼 내에서 가장 답변이 될 가능성이 높은 영역을 리턴하는 MRC(Machine Reading Comprehension)와,

가장 유사한 과거 질문/답변(FAQ)를 꺼내주는 IR(Information Retrieval) 형태가 있습니다.

주로 상담챗봇 및 콜센터에 자주 활용되는 기술입니다.



[그림 17. MRC 질의응답 예]



[그림 18. IR 질의응답 예]

## 4.4 마무리

어떻게 기계가 우리가 의사소통하는 언어를 수치화하고 학습하는지 다뤄봤습니다.

의미를 어느정도 이해하는 수준의 인공지능이 앞으로 어떻게 우리 삶을 편리하게 해줄지에 대한 방식은 무궁무진합니다.

물론 사람이 언어를 이용하며 만드는 미묘한 행간의 의미와 뉘앙스, 문체에서 느껴지는 아름다움이나 긴장감, 이런 것들은 아직 기계가 이해하기에 먼 길입니다.

하지만 언제나 길은 있습니다. 인간도 기계도 언제나 할 수 있는 것을 할 뿐입니다.

기계가 마법처럼 사람 말을 알아듣는 게 아닌, 도식화를 통해서 학습하는 것처럼요

언젠가는 찰떡같이 알아듣는 인공지능이 나오길 바랍니다.

이번 시간은 자연어의 개념과 인공지능이 이를 처리하는 방법에 대해 설명드렸습니다.

그리고 대표적인 NLP/NLU 과제 종류를 살펴보았습니다.

다음 시간은 '인공지능이 순차적인 데이터(시계열 데이터)를 학습하는 방법'에 대해 알아보겠습니다.

감사합니다 😊

## 참고자료

- '잔머리의 대가' 요즘 아이들, <http://mn.kbs.co.kr/news/view.do?ncd=4107322>
- [문명특급 EP.81] 펭수 모른다던 그 초딩들 만났습니다, SBS 뉴스, [https://news.sbs.co.kr/news/endPage.do?news\\_id=N1005493660&plink=SEARCH&cooper=SBSNEWSSEARCH&plink=COPYPASTE&cooper=SBSNEWSEND](https://news.sbs.co.kr/news/endPage.do?news_id=N1005493660&plink=SEARCH&cooper=SBSNEWSSEARCH&plink=COPYPASTE&cooper=SBSNEWSEND)
- 쪄기 문자, 위키피디아, [https://ko.wikipedia.org/wiki/%EC%90%90%EA%B8%B0\\_%EB%AC%B8%EC%9E%90](https://ko.wikipedia.org/wiki/%EC%90%90%EA%B8%B0_%EB%AC%B8%EC%9E%90)
- 밀레니얼 세대 다음은?, <https://brunch.co.kr/@moonkils/6>
- NLU, 인간의 언어를 이해하는 기계, TEC, <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=114306888>
- 아이언맨, 위키피디아, <https://ko.wikipedia.org/wiki/%EC%95%84%EC%9D%B4%EC%96%B8%EB%A7%A8>
- 교착어, 위키피디아, <https://ko.wikipedia.org/wiki/%EA%B5%90%EC%B0%A9%EC%96%B4>
- 자연어 감성분류를 위한 딥러닝, TEC, <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=112689711>
- 한국어와 NLTK, Gensim의 만남, <https://www.lucypark.kr/docs/2015-pyconkr/#20>
- 포켓몬고 Word2Vec CBOW-300 실험일지, AI기술팀 김명지, 2017.
- Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov, et al., 2013.
- Korean Word2Vec, <https://word2vec.kr/search/?query=%EA%B8%B0%EA%B3%84-%EC%B2%A0%2B%ED%94%BC%EB%B6%80>

## 5 [5편] 과거의 경험을 통해 현재를 배우는 인공지능

---

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/Cognitive AI LAB<sup>32</sup>

---

- 시간 흐름에 따른 데이터(Sequential data) 처리하기(see page 79)
  - Recurrent Neural Network(RNN; 순환 신경망)(see page 82)
    - 장점(see page 83)
      - RNN은 시간 흐름에 따른 과거 정보를 누적할 수 있다(see page 83)
      - RNN은 가변 길이의 데이터를 처리할 수 있다(see page 83)
      - RNN은 다양한 구성의 모델을 만들 수 있다(see page 84)
    - 단점(see page 85)
      - 연산 속도가 느리다(see page 85)
      - 학습이 불안정하다(see page 85)
      - 실질적으로 과거 정보를 잘 활용할 수 있는 모델이 아니다(see page 86)
    - 성능 보완(see page 86)
      - LSTM(Long-short term memory)(see page 86)
    - 활용 사례(see page 88)
    - 마무리(see page 89)
- 

지난 시간에는 자연어의 개념과 인공지능이 이를 처리하는 방법에 대해 다루었습니다.

NLU 관련 기술로 인해 이제는 기계가 사람의 말을 알아듣는 범위가 조금씩 넓어지고 있습니다.

오늘은 시간의 흐름에 따른 데이터 처리 방식, 인공지능이 '시계열 데이터'를 처리하는 방식에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 5.1 시간 흐름에 따른 데이터(Sequential data) 처리하기

딥러닝 알고리즘의 강점은 비정형 데이터를 규칙 없이도 잘 처리할 수 있다는 점입니다.

지난 시간까지 이미지라든가 텍스트와 같은 비정형 데이터를 인공신경망이 처리하는 방식에 대해 소개시켜드렸습니다.

하지만 아마 그동안 대다수가 접할 기회가 많았던 데이터는 주로 정형 데이터였을거라고 생각합니다.

즉 관계형 데이터베이스 자료나 엑셀, CSV파일 등으로 나타낼 수 있는 정리된 데이터 타입이죠.

<sup>32</sup><https://wire.lgcns.com/confluence/display/~78628>

정형 데이터를 분석하는 대표적인 과제로 시계열 예측분석이 있습니다.

순차적인 시계열 데이터를 활용하여 근미래에 어떤 데이터 값이 나타날지를 예측하는 과제입니다.

여기에는 시간 흐름에 따라 변화하는 로그성 데이터를 활용합니다.

예를 들어, 직장인들의 꿈(?)인 주가 예측이라든지, 내일의 기상정보 예측과 같은 일입니다.

## | 코스피

설시간 2020.09.07 10:18 장중

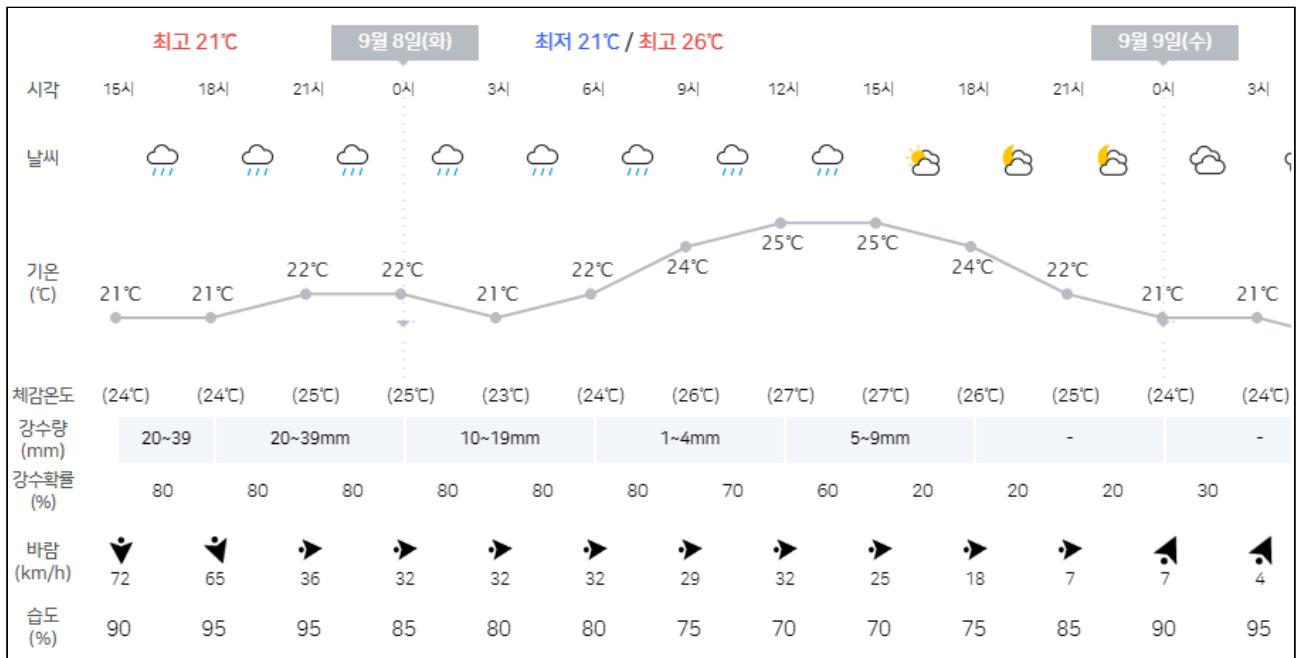


[그림 1. 시간 흐름에 따라 변화하는 시계열 데이터(Sequential data), 자료:[네이버금융](#)<sup>33]</sup>]

우리가 내일의 코스피 지수나 기온을 예측하려 할 땐 한 시점의 입력 데이터만 고려하지는 않을 겁니다.

예를 들어 시간대별로 기온을 예측해야 하는 기상청을 예로 들어보겠습니다.

33 [https://finance.naver.com/sise/sise\\_index.nhn?code=KOSPI](https://finance.naver.com/sise/sise_index.nhn?code=KOSPI)

[그림 2. 시간대별 일기예보, 자료:기상청<sup>34]</sup>]

기상청에서 현 시각의 기온을 예측해야 한다고 했을 때, 기온 예측에 중요한 변수로 이전 시각의 기온, 바람의 풍속, 습도 데이터가 중요하다고 가정하겠습니다. 일기예보관은 세 시간마다 이전 시각까지 확보한 데이터를 가지고 현 시각의 기온이 어떻게 될지 예측을 합니다.  
(※ 위 상황은 설명을 돋기 위한 예시로, 실제 기상청의 예측 프로세스와 다를 수 있습니다)

시각	9시	12시	15시	18시	$t$
이전시각 기온	23	25	26	24	$x_t$
이전시각 풍속	29	32	25	18	
이전시각 습도	80	90	70	70	
현시각 기온	25	26	24	?	$y_t$

[그림 3. 세 시간마다의 기온 예측]

예를 들어 9시엔 이전 시각의 기온 23도, 풍속 29 km/h, 습도 80%라는 값을 활용하여 9시의 기온이 25가 될 것이라고 예측합니다.

12시에는 9시의 기온(25도), 풍속, 습도를 가지고 26도가 될 것이라고 예측하고, 15시에도 마찬가지 작업을 합니다.

<sup>34</sup> <https://www.weather.go.kr/w/weather/today.do>

매 시각마다 '기온'이라는, 내가 예측하고자 하는  $y$ 값을 맞추기 위해 이전의 기온/풍속/습도라는 입력값  $x$ 를 활용하는 것이죠.

지금까지는 우리가 배운 인공지능 학습 방식과 동일합니다.

입력값  $x$ 를 가지고  $y$ 를 예측한다, 즉 이미지를 넣으면 강아지/고양이를 구별한다든가, 텍스트 문장을 넣어서 긍정/부정을 분류한다든가 하는 문제와 같습니다.

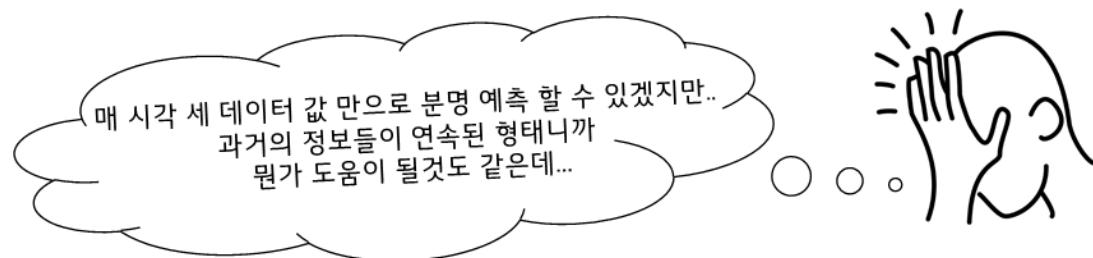
하지만 여기서 한 생각이 머릿속에 떠오릅니다.

일기예보관이 매 시간 같은 일을 반복하는데, 매시간 일을 반복하면서 쌓인 예측 노하우가 있을 겁니다.

그리고 밤낮이라는 시간의 흐름, 계절이라는 흐름에 따른 패턴이 분명히 존재할텐데

이렇게 그 때 그 때의 입력값 3개만으로 기온을 예측하는 것은 활용할 수 있는 정보를 다 쓰지 못하는 느낌 아닌가요?

한 시각의 데이터 세 개 만으로 분명 예측을 할수도 있겠지만 과거의 정보들이 연속된 형태니까 뭔가 도움이 될것도 같은 생각이 듭니다.

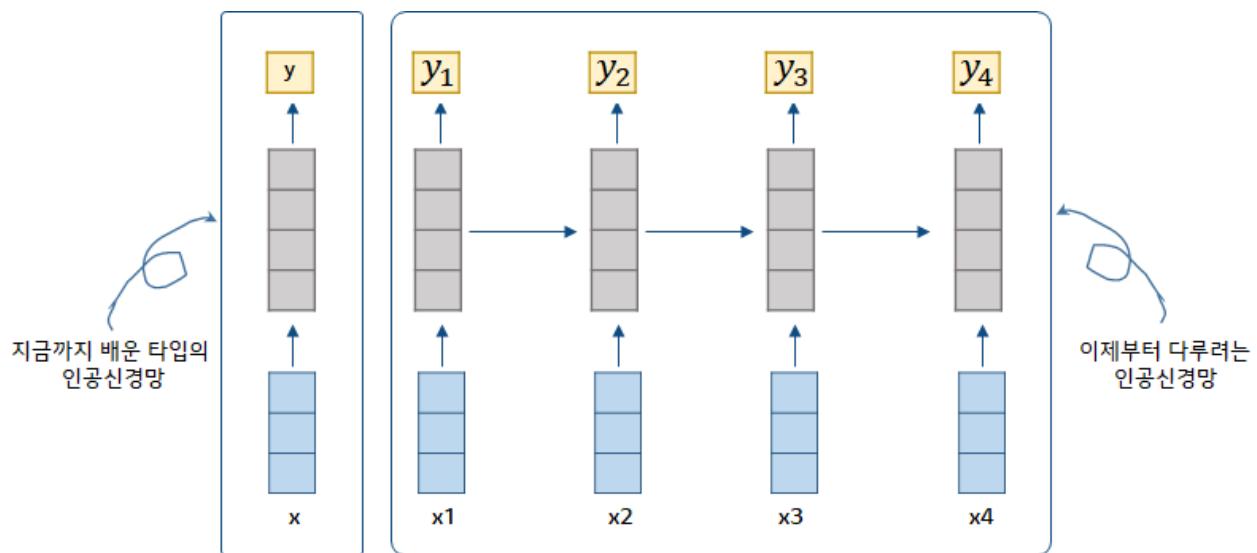


[그림 4. 과거의 노하우를 활용할 방법을 생각해봅시다]

## 5.2 Recurrent Neural Network(RNN; 순환 신경망)

이런 형태의 데이터를 잘 다루기 위한 인공신경망 종류가 있습니다.

'순환신경망'이라고 하는 RNN(Recurrent Neural Network)인데요, 아래 그림과 같이 생겼습니다.



[그림 5. 왼:입력값으로 출력값을 예측하는 기존의 인공신경망, 오:과거의 처리 이력을 압축하여 반영하는 RNN]

기존의 신경망이 벡터, 또는 매트릭스로 변환된 입력 데이터를 가지고 출력을 한다면  
RNN 역시 마찬가지 작업을 하지만 하나 다른 점이 있습니다.

[그림 5]에서 볼 수 있듯이, 과거에 데이터를 처리하여 결과를 출력했던 과정의 일부를 가져와 현 시점에서 데이터를 처리하고 결과를 출력하는 데 도움을 주는데요,

그림상에선 오른쪽으로 향하는 가로 방향의 화살표에 해당합니다. RNN에서는 벡터 형태로 정보(feature)가 넘어가죠.

그래서 2번째 시점에는 해당 시각의 입력 데이터 뿐 아니라 1번째 시점의 누적된 정보를 가지고 예측을 출력합니다.  
3번째 시점에는 1,2번째 시점의 누적된 정보가 반영될 것이고, 4번째 시점이 되면 1,2,3번째의 압축된 정보가 도움을 줍니다.

입력 데이터만으로 예측하는 것보다 과거의 누적된 정보(feature)가 있다면 더 개선된 예측을 할 수 있겠죠?

## 5.2.1 장점

RNN의 특징에 대해 살펴보았는데요, RNN이 가진 장단점에 대해 정리해보겠습니다.

### 5.2.1.1 RNN은 시간 흐름에 따른 과거 정보를 누적할 수 있다

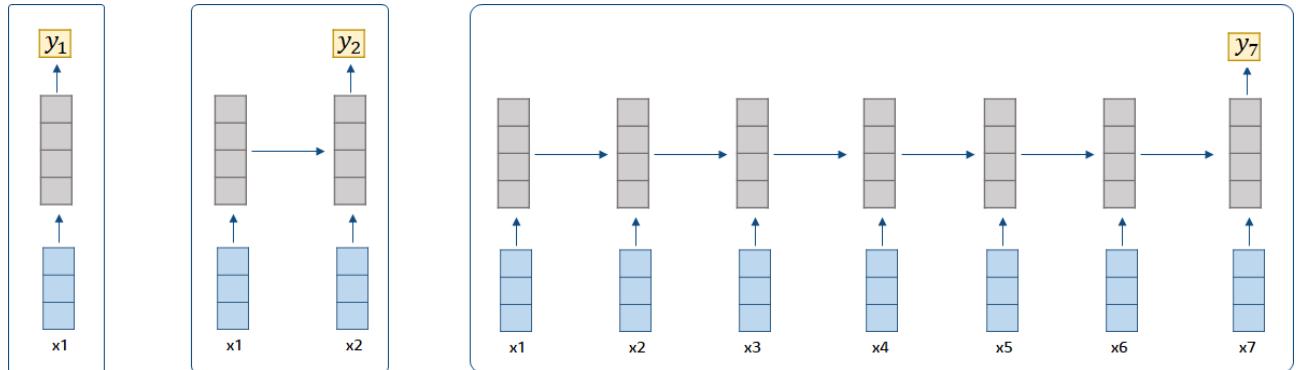
위에서 언급한대로, 다른 인공신경망과 달리 RNN이라는 특수한 형태의 신경망은  
입력 데이터뿐 아니라 과거의 처리 내역을 반영하여 더 나은 결정을 할 수 있다는 점이 가장 큰 장점입니다.

### 5.2.1.2 RNN은 가변 길이의 데이터를 처리할 수 있다

위 일기예보 예시의 경우 3시각 단위로 예측을 하는데, 이러한 단위 시간마다의 매 시점을 timestep이라고 합니다.  
timestep은 시간 단위가 될 수도 있고, 일 단위, 월 단위, 초 단위 등... 순서만 존재한다면 각자 구성하기 나름입니다.  
RNN은 과거의 정보를 매 timestep마다 압축하여 다음 timestep으로 넘기므로 데이터의 길이에 무관하게 자유롭게  
구성할 수 있습니다.

하루치 데이터로 예측을 할 수도 있고, 일주일치 데이터를 모아서 예측할 수도 있고, 한 달치 데이터를 모아서 예측할 수도 있습니다.

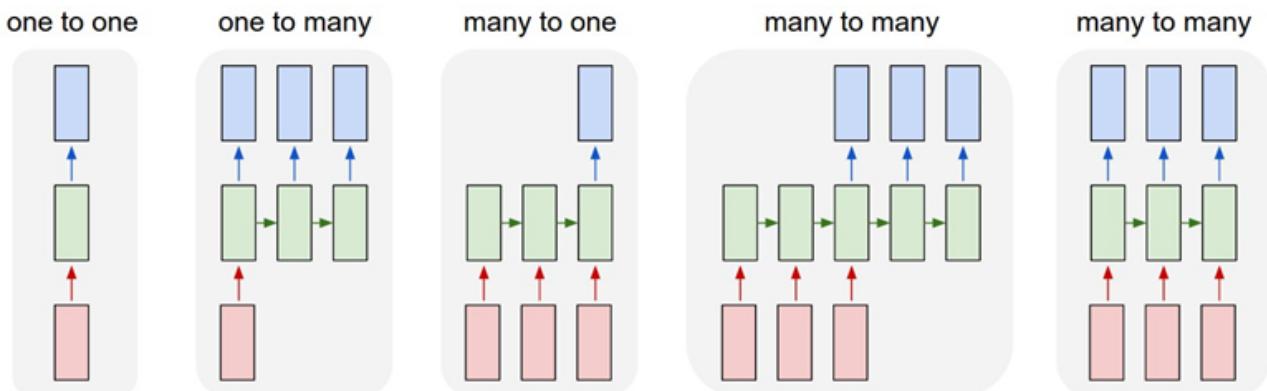
데이터가 아무리 긴 시간이나 짧은 시간에 대해 구성되어있다고 해도 같은 내용을 예측한다면 모델을 별도로 구성할 필요가 없습니다.



[그림 6. 위 세 가지 경우의 데이터 길이 모두 동일한 하나의 RNN 모델로 처리할 수 있다]

### 5.2.1.3 RNN은 다양한 구성의 모델을 만들 수 있다

유연한 구조를 가진 RNN은 다양한 구조를 활용하여 신경망을 구성할 수 있습니다.



[그림 7. 다양한 구조의 RNN]

- [그림 7]의 첫 번째 경우처럼 입력 데이터만으로 출력을 만들 수 있습니다. 이 경우는 이제까지의 신경망과 동일한 형태입니다.
- [그림 7]의 두 번째 경우처럼 한 번의 입력으로 여러 timestep의 데이터를 예측할 수 있습니다. 예를 들어 9시의 데이터로 12시, 15시, 18시의 기온을 예측하는 과정이 되겠네요
- 반대로 입력을 여러 timestep 받아 정보를 누적하다가 한번에 예측할 수도 있습니다. 일주일치의 데이터를 모아 8일차의 데이터를 예측한다던가 하는 등입니다.
- 입력 정보를 여러 timestep 누적하고 출력도 여러 timestep 진행할 수 있습니다. 월, 화, 수의 데이터로 목, 금, 토의 결과를 예측하는 경우입니다.
- 또는 마지막 경우처럼 그 때 그 때 입력과 출력을 처리할 수도 있습니다.

상황에 따라 RNN은 다양하게 입력 데이터를 처리, 누적하고 결과를 예측할 수 있습니다.

이 때 입력 데이터의 정보를 누적하는 부분을 인코딩(Encoding), 결과를 출력하는 부분을 디코딩(Decoding)이라고 표현합니다.

## 5.2.2 단점

### 5.2.2.1 연산 속도가 느리다

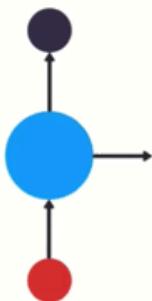
RNN은 과거의 처리 내역을 현재에 반영해야 하기 때문에, 현 시점의 데이터를 처리하려면 반드시 이전 시점의 데이터가 처리 완료되어야 합니다.

따라서 병렬학습이 어렵고, 순차적으로 데이터를 처리해야 하는 성질로 인해 연산 속도가 다소 느린 편입니다.

많은 경우 딥러닝 모델을 이용하여 데이터를 학습시킬 때 GPU 서버를 활용하곤 하는데요,

그래픽(주로 2차원 매트릭스나 3차원 이상의 텐서)을 다루는 데 특화된 GPU 칩은 병렬 연산에 굉장히 이점을 가진 장비입니다.

하지만 RNN을 처리하는 경우 이러한 병렬처리의 이점을 잘 활용할 수 없는 한계가 있습니다.



[그림 8. RNN은 앞 연산이 완료된 뒤 뒷 연산이 순차적으로 진행되어야 한다.]

하지만 정형 데이터(수치, 범주형 등)를 활용하는 경우 속도 저하를 크게 체감하지 못하는 경우가 대부분입니다.

연산 속도 저하는 텍스트 데이터를 다루는 경우에 주로 문제가 됩니다.

### 5.2.2.2 학습이 불안정하다

또한 단순 RNN은 학습시키기 매우 어려운 딥러닝 알고리즘 중 하나입니다.

미분 수식 전개가 필요하여 자세히 설명드리지는 않겠습니다만, 다른 데이터의 timestep이 길면 길수록 문제가 발생할 확률이 높습니다.

timestep이 길어지면 RNN 인공신경망이 반영해야 할 과거의 이력이 많아지게됩니다.

이 과정에서 인공신경망이 학습해야 할 값이 폭발적으로 증가하는 현상이 발생할 수 있습니다. 이를 **Gradient Exploding**이라 합니다.

또 이와는 반대로, timestep이 길어지면 저 멀리 있는 과거의 이력은 현재의 추론에 거의 영향을 미치지 못하는 문제도 생깁니다.

이를 **Gradient Vanishing**이라고 합니다.

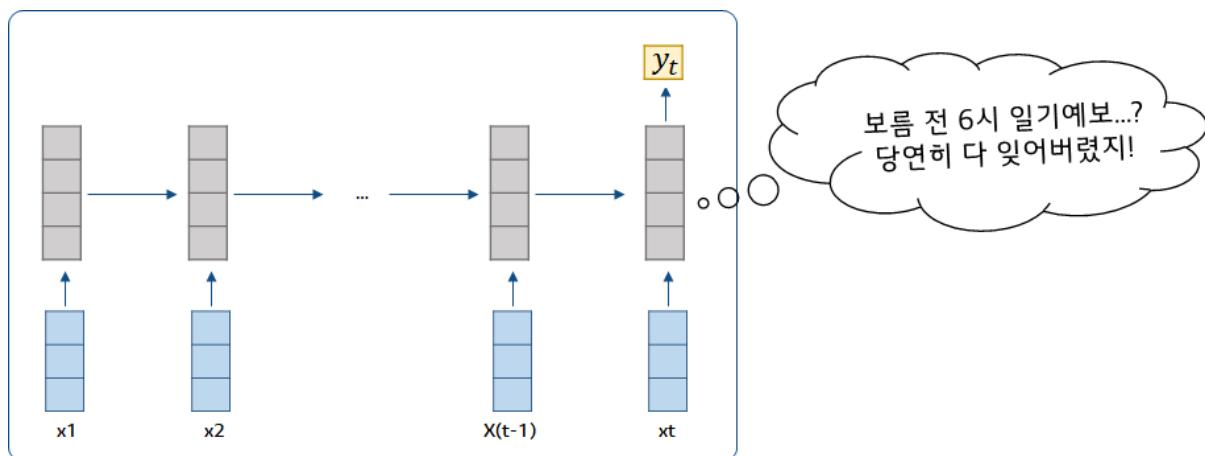
RNN은 상당히 학습이 불안정하여, 이 두 가지 문제가 자주 발생하곤 합니다.

### 5.2.2.3 실질적으로 과거 정보를 잘 활용할 수 있는 모델이 아니다

이론적으로 RNN은 과거의 정보가 누적되며 현재 추론에 도움을 주곤 합니다만, 실질적으로 먼 과거의 정보를 반영하기 힘듭니다.

RNN은 한 timestep씩 정보를 누적하여 인코딩하는데, 먼 과거의 정보는 여러 번 압축되고 누적되다보니 거의 영향을 미치지 못합니다.

이를 RNN의 **장기 종속성/의존성 문제(Long-term dependency)**라고 합니다.



[그림 9. RNN의 장기 종속성 문제. 먼 과거는 오늘날 영향을 거의 미치지 않는다.]

이는 사람도 마찬가지입니다. 오늘 점심 메뉴는 뭐였는지 기억하지만, 어제 점심이 뭐였는지 바로 기억하시는 분 있나요? 일주일 전은요?

최근의 정보일수록 잘 기억하고 반영하며, 먼 과거일수록 잊어버리는 경향이 인공신경망에서도 동일하게 나타납니다.

### 5.2.3 성능 보완

RNN에 이러한 단점이 있다 보니, 당연히 이를 해결하는 방안도 등장하였습니다.

#### 5.2.3.1 LSTM(Long-short term memory)

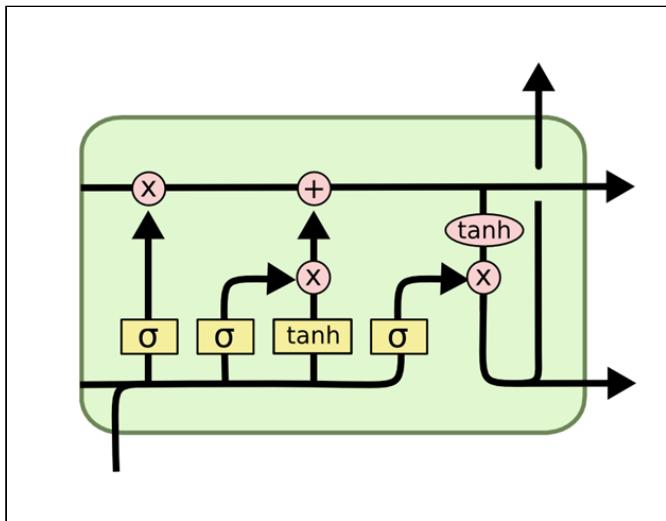
저 먼 과거의 정보 중 중요한 것은 기억하고, 불필요한 것은 잊어버리도록 스스로 조절 가능한 RNN 유닛이 있습니다.

매번 동일하게 과거의 정보를 누적하고 압축하는 기본 RNN(naive RNN) 유닛과는 달리,

정보 흐름을 잘 조절하기 위해 성능을 개선한 특별한 형태의 뉴런이라고 생각하시면 됩니다.

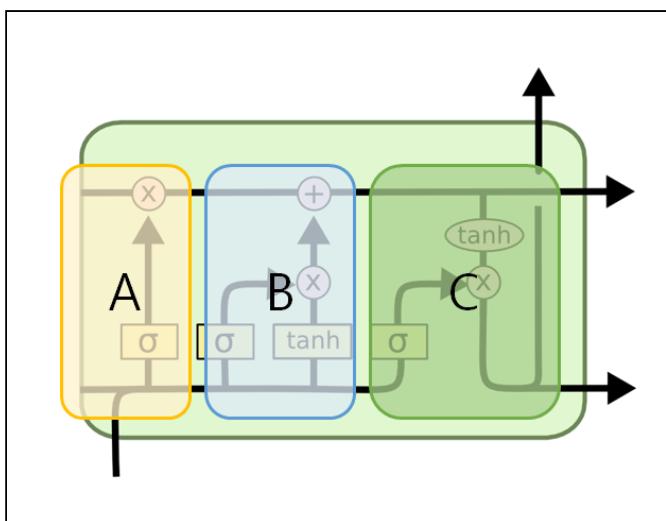
대표적으로 LSTM(Long-short term memory)이라는 유닛이 있습니다.

장/단기 메모리 유닛이라는 뜻인데요, 이 유닛은 아래와 같이 생겼습니다.



[그림 10. LSTM unit의 구조, [자료<sup>35\]</sup>](#)]

무섭게 생겼지만, 이 복잡해 보이는 연산은 어차피 컴퓨터가 수행하니 전혀 걱정하실 것 없습니다.  
우리는 구조를 조금 단순하게 묶어서 3부분의 Gate가 있다는 것만 알아둡시다.



[그림 11. 단순화한 LSTM 구조, A/B/C는 각각 forget/input/output gate에 해당]

Gate라 하는 부분은 정보의 흐름을 조절하는 관문 역할을 수행합니다.

[그림 11]의 A 부분은 **forget gate**라 불리며, 말 그대로 잊어버림에 대한 조절을 합니다.

과거의 정보중 불필요하다고 생각하는 부분은 통과시키지 않고 "이건 안중요하니까 잊어버려도돼" 라고 결정합니다.

<sup>35</sup> <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

B 부분은 **input gate**로, 현재의 정보(*input data*)를 얼마나 반영할지를 결정합니다.

"오늘 이런 정보가 있어? 이건 중요하니까 반영하자!" 또는, "오늘 꺼 별로 안 중요해보이는데? 과거 정보로 충분해! 거르자!" 이런 결정을 수행합니다.

C 부분은 **output gate**로, 현재 시점에 연산된 최종 정보를 다음 시점에 얼마나 넘길지를 결정합니다.

"그래 이건 내일도 쓸 수 있겠다" 라든가, "오늘은 써먹었지만 내일은 불필요할듯" 같은 역할을 합니다.

LSTM에는 이 세 가지의 gate가 있어서, 정보의 흐름을 인공지능이 자체적으로 더 원활하게 조절하는 기능을 합니다. 따라서 데이터가 길어진다고 해도 일반 RNN에 비해 더 좋은 예측을 할 수 있게 됩니다.

다만 계산 과정이 복잡한 만큼 연산속도는 조금 더 느려지는 단점이 있습니다.

그래서 이를 조금 개선한 GRU(Gated Recurrent Unit)도 있습니다만, LSTM과 거의 유사한 기능을 갖고 있어 소개하지는 않겠습니다.

오늘날 인공신경망을 활용하는 대부분의 시계열 예측은 아주 간단한 과제를 제외하고는 기본 RNN을 사용하는 경우는 거의 없습니다.

대부분은 LSTM, GRU과 같은 개선된 유닛을 활용하며, 대부분의 딥러닝 프레임워크에서 쉽게 구현할 수 있도록 기능을 제공하고 있습니다.

### 5.3 활용 사례

만일 시계열 데이터가 익숙하지 않은 분이시라면 기존의 과제와 어떻게 다른지 혼란스러울 수 있습니다.

Sequential Data라는 말 그대로 알 수 있듯이 뭔가 순차적인 흐름을 가지고 진행되는 데이터가 있다면 전부 시계열 데이터라고 부를 수 있습니다.

우리가 오늘 다룬 예제의 경우를 살펴보면, 물론 풍속이나 습도 만으로도 현 기온을 예측할 수 있겠지요.

하지만 기온이라는 것은 매 시점 독립적으로 정해지는 것이 아닌 시간에 따른 패턴을 갖습니다.

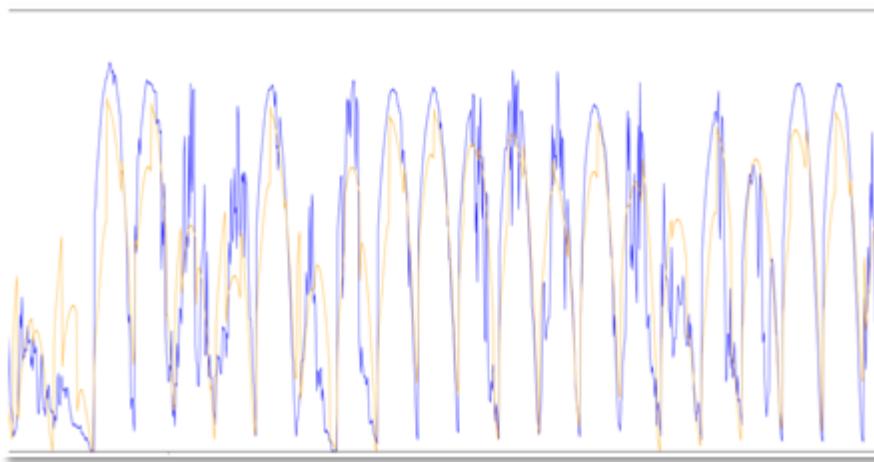
아침에 추웠다면 점심에도 약간 쌀쌀한 경향이 반영될 것이고, 대체로 아침/저녁에 기온이 내려가는 반면 낮에는 약간 올라가곤 하죠.

인공신경망을 활용한다면 이러한 패턴을 규칙화하지 않아도 자동으로 학습할 수 있습니다.

자사에서도 2018년, LSTM 모델을 활용해서 태양광 에너지 발전량을 예측한 적이 있습니다.

이 경우 10분 간격 timestep에 따라 경북 오태지역의 일시, 일조량, 풍속, 온도를 가지고 에너지 발전량을 예측하였습니다.

기존 수리모델 대비 LSTM이라는 딥러닝 모델이 성능을 개선하였다고 하네요.



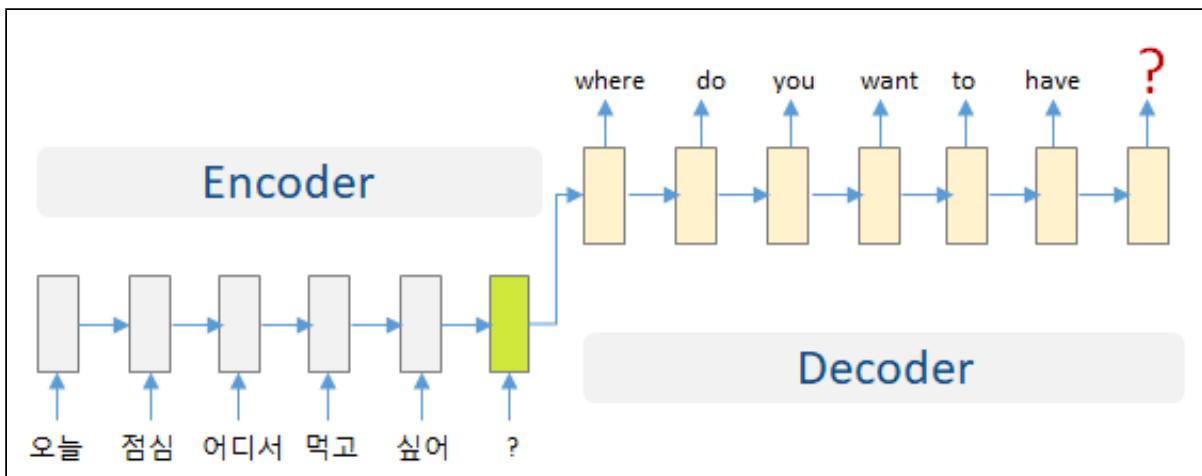
[그림 12. 태양광 에너지 발전량 예측 (2018), LG CNS]

텍스트 문장도 일종의 시계열 데이터로 볼 수 있습니다.

문장이란 단어가 일정한 순서대로 등장하는 데이터입니다.

한 단어 한 단어를 벡터로 바꿀 수 있으니([\[4편\] 참고\(see page 60\)](#)), RNN의 한 timestep에 단어 벡터를 하나씩 입력시킬 때 됩니다.

RNN을 텍스트 데이터에 적용한다면 번역과 같은 과제에 활용할 수 있습니다.



[그림 13. 한국어 문장을 인코딩하여 영어 문장을 디코딩하는 RNN (sequence-to-sequence)]

## 5.4 마무리

우리는 오래전부터 과거의 경험을 통하여 미래를 예견하고자 하였습니다.

사람은 한 번 수행했던 업무는 노하우를 얻어 다음에 더 빠르고 잘 수행할 수 있습니다.

과거에 실패를 경험했다면 같은 실패를 또 경험할 가능성도 적지요.

최근의 기억은 더 뚜렷하게 남고, 먼 과거는 잊어버리기 마련입니다.

인간의 신경계를 본딴 인공신경망에도 비슷한 기능을 수행하는 알고리즘이 있습니다.

전통적인 머신러닝 기법에서는 ARIMA 기법 등을 통하여 데이터의 정보 흐름을 파악하고, 주기적으로 반복되는 패턴을 반영하여 분석합니다.

인공신경망에서는 RNN이라고 불리우는 특별한 형태의 신경망이 그 역할을 수행합니다.



[그림 14. AI 미래 예측은 마법이 아닙니다. '가장 그럴싸한 것'에 대한 수리모델적 추론입니다.]

미래를 예측한다는 것은 인공지능이 미래를 좌지우지한다거나, 예언한다는 개념이 아닙니다.

과거의 패턴을 통해 근미래에 벌어질 가능성성이 가장 높은 데이터를 유추하는 것이지요.

이번 시간은 인공지능이 '시계열 데이터'를 처리하는 RNN 방식에 대해 설명드렸습니다.

그리고 이 방식이 지닌 장단점과, 이를 보완하기 위한 개선방법도 살펴보았습니다.

다음 시간은 '**오버피팅(Overfitting)**과 **정규화(Regularization)**'에 대해 알아보겠습니다.

감사합니다 😊

## 참고자료

- 네이버 금융, 코스피, [https://finance.naver.com/sise/sise\\_index.nhn?code=KOSPI](https://finance.naver.com/sise/sise_index.nhn?code=KOSPI)
- 기상청, 오늘의 날씨, <https://www.weather.go.kr/w/weather/today.do>
- CS231n-lecture10, Fei-fei Li&Andrej Karpathy&Justin Johnson, 2016, [http://cs231n.stanford.edu/slides/2016/winter1516\\_lecture10.pdf](http://cs231n.stanford.edu/slides/2016/winter1516_lecture10.pdf)
- 자사 딥러닝 실무과정 교재보기, [교재보기] 딥러닝 실무<sup>36</sup>
- Illustrated Guide to Recurrent Neural Networks, MC.AI, <https://mc.ai/illustrated-guide-to-recurrent-neural-networks/>
- Understanding LSTM Networks, colah's blog, <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- 전사기술세미나 20차, 창작하는 인공지능, 김명지, 2020, 20차 "인공지능, 스스로의 학습을 넘어 크리에이터가 되다!!" - 2020.07.30<sup>37</sup>

<sup>36</sup> <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

<sup>37</sup> <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=114632185>

## 6 [6편] 헛똑똑이 인공지능 제대로 가르치기

---

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/Cognitive AI LAB<sup>38</sup>

---

- [AI Process\(see page 91\)](#)
    - [Offline Process\(see page 92\)](#)
    - [Online Process\(see page 94\)](#)
  - [오버피팅\(Overfitting\)과 일반화 성능\(Generalization\)\(see page 95\)](#)
    - [Training, Validation, Test\(see page 97\)](#)
      - [Training set\(see page 98\)](#)
      - [Validation set\(see page 98\)](#)
      - [Test set\(see page 98\)](#)
      - [학습 곡선\(Learning curve\) 확인하기\(see page 98\)](#)
    - [Regularization\(see page 100\)](#)
      - [데이터 증강\(Data Augmentation\)\(see page 101\)](#)
      - [Capacity 줄이기\(see page 101\)](#)
      - [조기 종료\(Early stopping\)\(see page 102\)](#)
      - [드롭아웃\(Dropout\)\(see page 102\)](#)
    - [마무리\(see page 103\)](#)
- 

지난 시간에는 인공지능이 '시계열 데이터'를 처리하는 RNN 방식에 대해 다루었습니다.

전통적인 분석 기법이 수행하던 영역에서도 딥러닝 알고리즘이 좋은 성능을 낼 수 있습니다.

오늘은 딥러닝 모델을 학습할 때 마주치기 쉬운 문제인 '오버피팅'에 대해 알아보고 이를 해결하는 방식에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

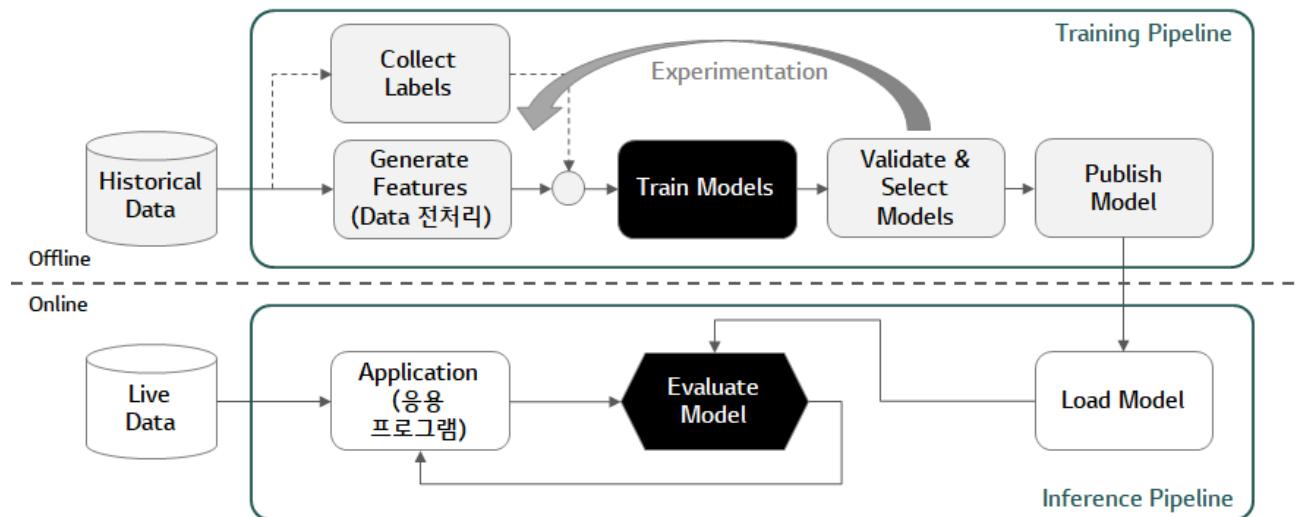
### 6.1 AI Process

AI를 활용해서 우리 주변을 더 나아지도록 할 방법을 찾고 계신가요? 그렇다면 아래와 같은 프로세스를 따라야 합니다.

여기서 말하는 AI는 딥러닝 모델 뿐 아니라, 빅데이터 분석과 같은 전통적인 머신러닝 기반의 모델 및 규칙 기반의 기계 자동화 모델이 포함된, 넓은 의미의 AI를 말합니다.

<sup>38</sup><https://wire.lgcns.com/confluence/display/~78628>

고객사의 요청으로 기존 시스템을 개선하기 위해 AI를 적용한다고 가정해보겠습니다.



[그림 1]. AI pipeline, 참고:Netflix<sup>39]</sup>

AI를 도입하는 과정은 크게 오프라인 프로세스와 온라인 프로세스로 나뉩니다.

[그림 1]의 윗부분과 아랫부분에 해당하는 과정이지요.

쉽게 말한다면 각 과정은 고객사 프로젝트에서 오픈을 위해 열심히 준비하는 개발 과정과, 고객사 오픈 이후의 운영 환경에 해당하는 과정이라고 볼 수 있습니다.

### 6.1.1 Offline Process

오프라인 프로세스는 과거에 만들어진 데이터(historical data)를 가공하는 것에서부터 시작합니다.

이는 그 때 그 때 발생하는 실시간 데이터가 아닌, DB 등에 이미 수집되어 있는 데이터입니다.

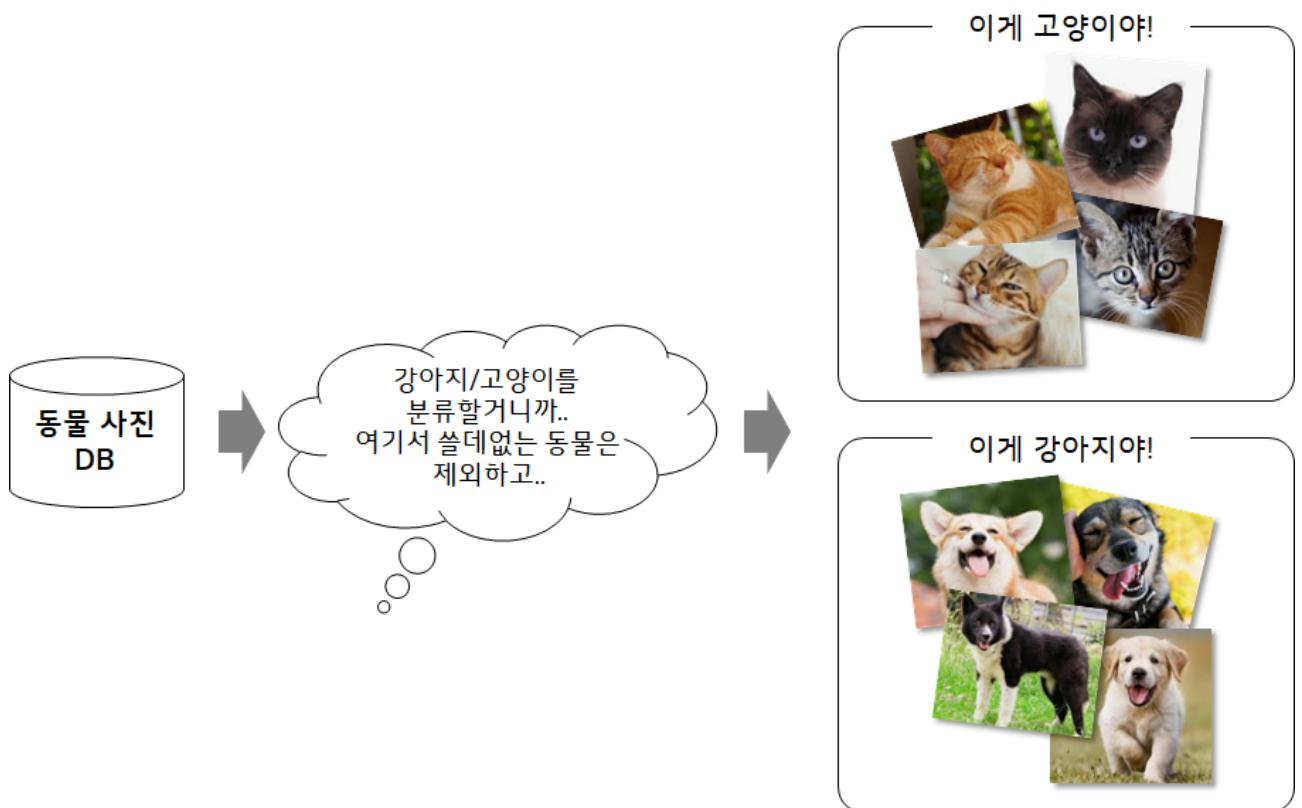
AI 모델러는 기 확보된 데이터를 확인하고 정제하여 필요한 부분을 취하거나(**Generate features**) 필요한 경우 라벨을 붙입니다(**Collect labels**).

이 때의 라벨이란 AI 모델 학습을 위해 필요한 정보로, 인공지능이 맞춰야 하는 '정답'이라고 생각하시면 됩니다.

예를 들어 "강아지와 고양이를 자동 분류하는 AI모델을 만들겠다!"라는 목표가 있다면 historical data는 강아지와 고양이의 이미지 파일이 되고,

라벨을 붙이는 작업은 각 사진이 강아지인지 고양이인지 태깅하는 작업이 되겠습니다.

39 <https://www.slideshare.net/justinbasilico/making-netflix-machine-learning-algorithms-reliable>



[그림 2. 강아지/고양이 자동분류를 위해 historical Data를 전처리하고 라벨링하는 과정]

데이터를 마련했으면 어떤 머신러닝/딥러닝 알고리즘을 활용하여 모델을 학습할 것인지 정하고,

**좋은 성능을 달성할 때까지(Validate & Select models) 반복실험을 진행합니다(Train models).**

대부분의 머신러닝/딥러닝 모델 개발 경우 한 번의 시도만에 최적 모델이 완성되는 경우는 거의 없습니다.

보통은 실험의 결과를 진단하고 이 때 발견한 문제점을 해결하거나 더 개선된 성능을 낼 수 있도록 실험(**Experimentation**)을 반복하는데, 이 과정을 **튜닝**이라고 합니다.

튜닝은 모델 학습에 필요한 여러 수치 설정값(하이퍼파라미터)을 조절하는 등의 역할을 포함합니다.

만일 누군가 '실력 좋은 AI 모델 개발자'란 어떤 사람인가? 하고 물으신다면 저는 모델 튜닝을 더 폭넓고 빠르게 하는 사람이라고 대답하겠습니다.

이전 실험의 문제점을 찾아 다음 실험에 어떤 전략을 적용해야 모델의 성능을 개선할 수 있을지,

다양한 대안을 파악하고 노하우가 많은 사람일수록 적은 시행착오로 최적 모델을 찾을 수 있겠죠?

여러 실험을 반복하며 개선된 성능의 모델은 **배포할 수 있도록 최종 선택(Publish model)**됩니다.

여기까지의 과정은 더 나은 AI모델을 만들기 위해 모델을 학습시키는 과정으로, **Training Pipeline**이라고 합니다.

### 6.1.2 Online Process

최적의 성능을 내기 위해 모델을 개발하는 부분은 끝났습니다.

이제는 완성된 모델을 고객사 운영 환경에 올려야 합니다.

AI 모델은 오프라인 프로세스에서 배울 것을 모두 배웠으니, 실전 환경에 뛰어들어 추론(Inference)을 해야 합니다.



[그림 3. 모델 Training(위)과 Inference(아래)의 차이. Training이 훈련과정이라면 Inference는 실전 투입이다.]

모델 추론을 포함한 온라인 프로세스의 대부분은 머신러닝/딥러닝이라기보다는 **개발 영역(Application)**에 가깝습니다.

**AI 모델을 운영 환경에 띄운다면(Load Model)** 나머지는 GUI를 불인다든지, 고객사의 데이터베이스와 연결하고 다른 시스템과 연동한다든지 하는 일이 되겠지요.

이제부터 모델이 처리할 데이터는 기존에 정리하여 모아놓은 데이터가 아닙니다.

실시간으로 들어오는, 운영 환경의 **스트리밍 데이터(Live Data)**입니다.

AI 모델은 과거의 학습 지식을 바탕으로 현장의 데이터를 추론하게 됩니다.

머신러닝/딥러닝 관련 기법과 팁에 대한 대부분의 내용은 **오프라인 프로세스의 모델 학습 과정에 적용할 수 있는 것 들입니다.**

AI 모델의 기본적인 성능을 끌어올리고, 주어진 데이터를 더 잘 맞출 수 있도록 최적화하는 데 필요한 지식들이죠.

하지만 우리는 주어진 데이터를 잘 맞추는 것이 목적이 아닙니다.

우리는 주어진 데이터로 잘 훈련된 AI 모델이 개발 환경 뿐 아니라 고객사 운영 환경과 같은 **실제 현장에 나가서도 잘 작동되기를 바랍니다.**

## 6.2 오버피팅(Overfitting)과 일반화 성능(Generalization)

회사에서 우스개 소리로 말하는 '시연(DEMO)의 법칙'을 아시나요?

분명 내가 어젯밤에 테스트해 볼 때는 잘 작동하던 것이, 중요한 보고나 고객 앞에서 시연할 때만 갑자기 기능이 동작하지 않는다거나, 느려지거나, 예외가 발생하거나 하는 일이 있지요.



[그림 4. 시연의 법칙. 이게 다 Generalization이 부족해서입니다.]

억울한 일이지만, 이와 같은 문제는 기계학습에서도 굉장히 자주 발생합니다.

분명히 학습시킬 때는 주어진 데이터를 잘 맞췄는데, 이상하게 고객사 운영환경에만 올라가면 추론 성능이 똑똑 떨어집니다.

지금부터 기계학습에서 가장 중요한 개념을 소개시켜드리려고 합니다.

**Generalization**이라고 부르는, 일반화 성능과 관련된 개념입니다.

Generalization의 정의는 '이전에 본 적 없는 데이터에 대해서도 잘 수행하는 능력'입니다.

즉, 우리가 만든 AI 모델은 훈련시에는 본 적이 없는 새로운 입력 데이터(Live data)에 대해서도 잘 수행되어야 한다는 것입니다.

훈련시에만 잘 작동하고 일반화 성능이 떨어지는 모델을 **오버피팅(Overfitting)**되었다고 합니다.

예시를 통해 알아보겠습니다.

똑딱이와 펭수는 고3 수험생으로 올해 수능을 치른다. 남은 시간은 한달 남짓..

똑딱이는 이해력은 조금 딸리지만, 굉장히 노력파로 한달동안 최근 5년간의 수능 기출을 전부 암기해버렸다.

그래서 똑딱이는 문제만 들으면 정답이 몇 번 보기에 어떤 문장이었는지 정확하게 복원해냈다.

다만 문제의 의도가 뭔지, 왜 그 보기가 정답인지는 이해할 수 없었다. 단순히 문제와 답을 외웠기 때문이다.

펭수는 한달간 과거 기출의 출제 의도를 파악하고 그 보기가 왜 정답인지를 이해하며 공부했다.

최근 5개년의 출제 동향이나 문항 패턴을 파악하였지만 어려운 문제도 있었기에, 기출문제를 전부 맞출 수는 없었다.

기출을 암기한 똑딱이와 내용을 이해한 펭수는 올해 수능 수험장에 들어갔다. 수능 결과가 어땠을까?



	똑딱이	펭수
5개년 과거기출	상위1% (전부 외워서 만점)	상위 10% (어려운 문제 몇개 틀림)
올해 수능	상위 80% (기출문제 그대로 나온 것 + 찍은것만 맞춤)	상위 10% (어려운 문제 몇개 틀림)

여러분이 학부모라면 자녀를 어떻게 키우고 싶으신가요?

저는 자녀는 없지만... 대신에 인공지능을 연구하는 연구원으로서 제 모델은 펭수처럼 학습할 수 있도록 노력하고 있습니다.

여기서 뚝딱이는 인공지능으로 따지자면 오버피팅된 모델이라고 볼 수 있습니다.

이미 습득한 데이터에 대해서는 모두 외웠기 때문에 기가 막히게 잘 맞출 수 있지만, 일반화 능력이 떨어지기 때문에 외운 것과 다른 데이터에 대해서는 제대로 역할을 수행하지 못합니다.

우리의 목적은 무엇일까요?

만일 여러분이 인공지능을 도입하려는 적용처가 늘 한정된 데이터 몇 가지만 다루는 곳이라면, 데이터를 외워서 추론하는 모델을 만들어도 무관합니다.

이 경우엔 기계학습이나 딥러닝 모델을 활용하지 않는다고 해도 간단하게 처리할 수 있는 경우가 많겠지요.

하지만 현장은 다양한 패턴의 데이터가 셀 수 없이 쏟아져 나오는 경우가 대부분일겁니다.

특히 사진, 동영상, 글, 음성 등과 같은 비정형 데이터를 다루는 곳이라면요.

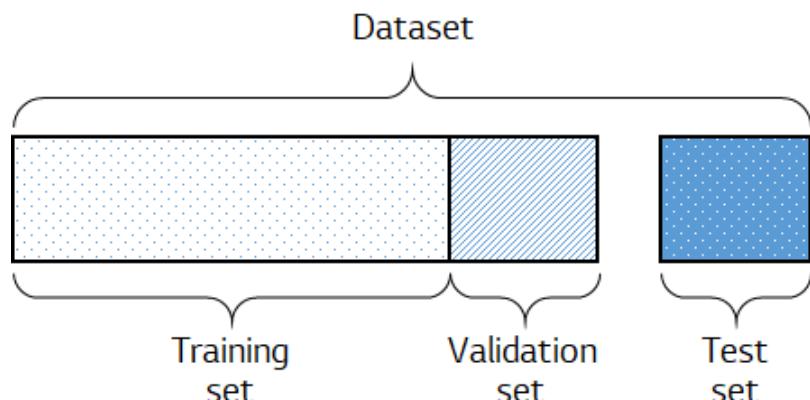
미리 쌓아놓은 데이터를 잘 맞추도록 학습하는 것은 물론 중요하지만 앞으로 실시간으로 발생할 데이터도 잘 추론하도록 하는 것이 우리의 목적이 됩니다.

즉, 운영 환경에서의 일반화 성능 또한 학습 과정에서의 최적화만큼이나 중요하며 우리는 늘 이를 염두에 두고 모델을 만들어야 합니다.

### 6.2.1 Training, Validation, Test

우리의 모델이 현장에서도 잘 작동할지는 어떻게 확인할 수 있을까요?

먼저, 우리가 확보한 데이터(Historical data)를 Training, Validation, Test의 세 set으로 나누고 각 set이 수행할 역할을 구분해주도록 하겠습니다.



[그림 5. Training/Validation/Test set의 구분]

엄격히 비율이 정해져 있는 것은 아니지만, 보통은 Training, Validation, Test set을 8:1:1, 6:2:2 정도로 구분합니다.

나누기 전에는 데이터를 골고루 섞어주어 set별로 데이터의 성향이 다르거나 치우침이 없도록 합니다.

각 set의 역할을 설명드리겠습니다.

### 6.2.1.1 Training set

Training set은 머신러닝/딥러닝 모델을 학습하는 데 이용하는 데이터입니다.

모델은 Training set의 입력 데이터와 정답을 보고, 정답을 더 잘 맞추기 위해 노력합니다.

이는 단순히 최적화(Optimization)에 해당합니다.

### 6.2.1.2 Validation set

Validation set은 머신러닝/딥러닝 모델에게 정답을 알려 줄 데이터는 아니지만, 우리가 모델을 튜닝하는 데 도움을 주는 데이터입니다.

즉 모델의 일반화 성능을 판단하여 이어질 실험을 계획하는 데 이용합니다.

모델은 Validation set의 정답은 본 적이 없지만 이 set의 입력 데이터만으로 일단 정답을 추론하게 됩니다.

모델이 배우지 않았던 데이터, 즉 처음 보는 데이터에 대해 얼마나 잘 맞추는지를 계산할 수 있으니 이 결과를 보고 우리는 실험을 개선하게 됩니다.

예를 들어 우리가 만든 모델이 Training set에 대해서는 잘 맞추는데 Validation set에 대해서는 너무 못맞춘다고 가정해 봅시다.

위에서 말씀드린 예제의 똑딱이처럼, 알려준 것만을 달달 외우고 일반화성능이 부족한 오버피팅 현상이 발생했다고 볼 수 있습니다.

우리는 이를 통해 다음 실험시 오버피팅을 방지할 전략을 세울 수 있습니다.

오버피팅을 확인하는 방법과 해결 전략은 뒤이어 다루겠습니다.

### 6.2.1.3 Test set

Test set은 모델의 학습에 어떤 식으로도 전혀 관여하지 않는 데이터로, 오로지 모델의 최종 성능을 평가하기 위해 따로 떼어놓은 데이터입니다.

여러 모델간 성능을 비교할 땐 Test set에 대한 스코어를 활용합니다.

Training set으로 모델을 학습하고, Validation set에 대한 성능을 확인하며 모델을 개선해왔으니 그 결과를 공정히 평가하기 위한 기준이죠.

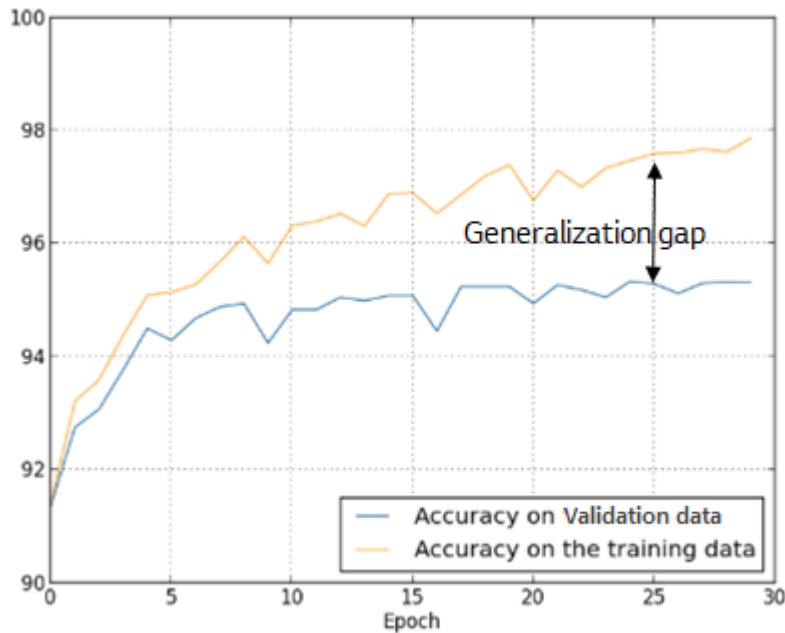
똑딱이와 펑수가 치른 금년도 수능 시험같은 데이터라고 생각하시면 됩니다.

모든 수험생이 제각기 다른 방식으로 공부했겠지만 모두가 처음 보는 문제로 동시에 공정하게 평가하는 기준이 수능입니다.

## 6.2.2 학습 곡선(Learning curve) 확인하기

역할에 따라 데이터를 나누었다면 학습이 제대로 이루어지는지, 일반화 성능이 떨어지거나 하여 오버피팅이 발생하지는 않았는지 확인해야 합니다.

확인하는 방법은 학습 곡선(Learning curve)을 그려보는 것인데, 학습 곡선이란 학습이 진행됨에 따라 모델의 성능을 기록하는 그래프입니다.



[그림 6. 학습 곡선의 예. 대체로 가로 축은 학습의 진행 시점을 나타낸다]

오버피팅이 발생했는지 확인할 수 있는 방법은 학습 곡선상에서 Training set와 Validation set에 대한 모델의 성능이 어떻게 변화하는지를 확인해보는 것입니다.

정답을 알면서 배우는 데이터에 대해서는 당연히 잘 맞춰야겠지만, 정답을 모르고 추론만 진행하는 데이터에 대해서는 어떨까요?

어느 순간 Validation set에 대해서 성능 개선이 없다면 모델은 '학습'이 아닌, '암기'를 하는 것이라 볼 수 있겠습니다. 전반적인 데이터의 패턴을 배운다기보다 주어진 데이터의 정답을 잘 맞추도록 문제와 정답을 달달 외우는 것이지요. 아래 학습 곡선은 전형적인 오버피팅의 예입니다.



[그림 7. 오버피팅이 발생한 경우의 학습 곡선]

[그림 7]의 경우 400 epoch(Training set 전부를 한 번씩 학습에 이용하는 단위)동안 모델 학습을 시켰습니다.

여기서 왼쪽의 그래프를 보면 Training set, 즉 모델에게 정답을 알려주면서 잘 맞추도록 유도하는 경우에 대해서는 문제없이 최적화가 진행되는 것처럼 보입니다.

참고로 왼쪽 그래프의 세로 축을 이루는 Cost라는 것은 실제 정답과 모델이 예측한 예측값의 차이를 정량화해준 수치입니다. 작을 수록 모델이 잘 맞춘 것입니다.

하지만 오른쪽의 그래프를 보면, 모델이 정답을 모른 채 예측만 해야 하는 Validation set에 대해서는 280 epoch째부터 큰 개선이 없다는 것을 알 수 있는데요,

이 때의 세로축은 분류 정확도(Accuracy)로, 그 값이 높을 수록 분류를 잘 한다는 뜻입니다.

Training set에 대해서만 성능을 개선하고 있고, Validation set에 대해서는 별다른 향상이 없는 시점이 오버피팅 발생 시점입니다.

### 6.3 Regularization

만일 여러분이 머신러닝/딥러닝 모델을 학습할 때 [그림 7]과 같은 경향을 마주했다면 다음 실험에서는 이 현상을 개선하기 위한 전략을 취해야겠죠?

오버피팅을 피하기 위한 모든 전략들을 Regularization이라고 합니다.

굳이 번역하자면 '정규화'이지만, Normalization의 정규화와는 조금 다른 개념이므로 영어 원문으로 계속 표기하도록 하겠습니다.

Regularization의 목적은 Generalization, 즉 일반화 성능을 향상하는 데 있습니다.

Training set에 대해 더 잘 맞추고자 하는 것이 아닌, 현장에 나가 처음 보는 데이터에 대해서도 잘 맞출 수 있도록 하는 목적이지요.

이를 위해 취할 수 있는 여러가지 방법을 간단히 소개하겠습니다.

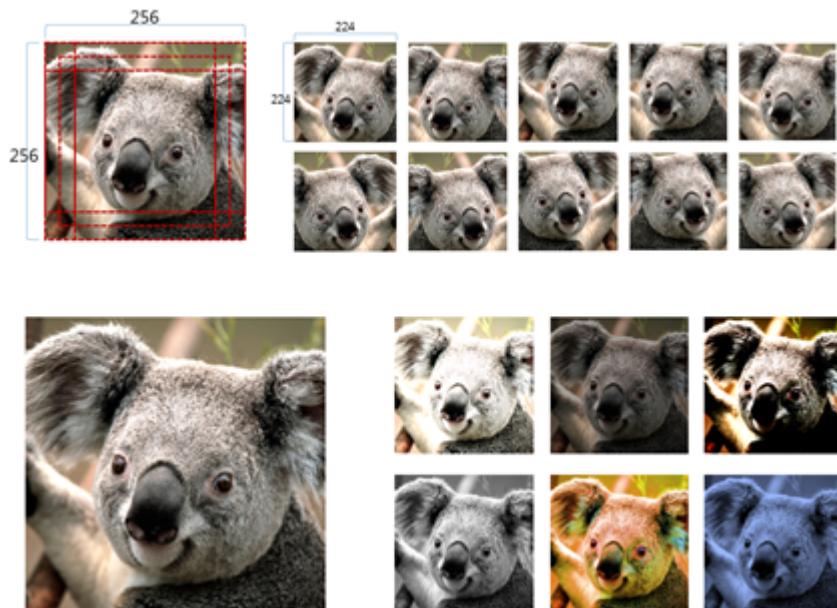
### 6.3.1 데이터 증강(Data Augmentation)

오버피팅을 피하는 가장 좋은 방법은 데이터를 더 많이 확보하는 것입니다.

다양한 데이터가 있다는 것을 알려준다면 인공지능 모델도 현장에 나갔을 때 새로운 데이터를 더 잘 맞출 수 있을 겁니다.

하지만 현실적으로 데이터가 절대적으로 부족한 경우가 대부분인데요,

이를 위해 얼마 없는 데이터 건수를 마치 많은 것처럼, 데이터를 증강시키는 다양한 기법이 있습니다.



[그림 8. 코알라 이미지 데이터 증강 예]

이미지 데이터 학습을 예로 든다면, 한 장의 이미지를 좌우 반전 시키거나, 일부 영역을 크롭하거나, 노이즈를 추가하거나,

색상, 명암, 채도 등에 변화를 주어 모델 학습에 추가로 데이터를 이용할 수 있습니다.

이렇게 하면 인공지능에게 더 다양한 환경에서 찍은 듯한 이미지를 학습시키는 효과를 주어 오버피팅을 방지할 수 있습니다.

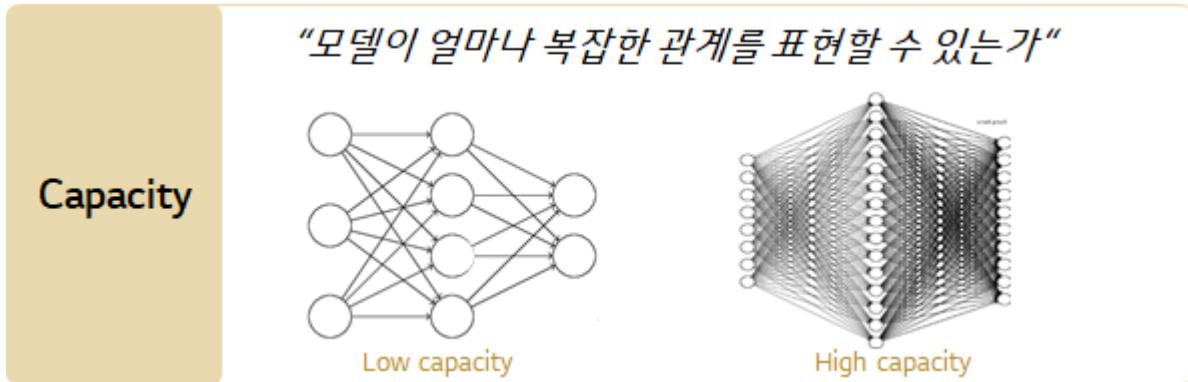
하지만 너무 과도한 변형은 오히려 해가 될 수 있으니, 주의해서 이용해야 합니다.

### 6.3.2 Capacity 줄이기

Capacity는 모델의 복잡한 정도를 나타내는 개념입니다.

일반적으로 딥러닝 모델이 머신러닝 모델보다 Capacity가 높으며, 그 중에서도 신경망을 여러층 쌓거나 뉴런의 수를 많이 둘 수록 Capacity가 높아진다고 말할 수 있습니다.

Capacity가 높은 모델은 처리할 데이터의 복잡 다양한 패턴을 더 잘 담아낼 수 있습니다.



[그림 9. 인공신경망의 Capacity]

하지만 Capacity가 필요 이상으로 너무 높은 모델은 주어진 데이터를 외우게 될 가능성이 높습니다. 따라서 내가 수행하려는 태스크에 알맞은 Capacity를 가진 모델을 선택하는 것이 좋은데요, 태스크의 복잡도와 Capacity의 관계는 딱 정해진 규칙이 없어서 어느 수준이 적정선이라고 말하기 어렵습니다. 만일 오버피팅의 경향이 발견된다 싶으면 내 모델의 총 수를 줄여본다든지, 한 층의 뉴런 수를 줄인다든지 등의 조치를 취해볼 필요가 있습니다.

### 6.3.3 조기 종료(Early stopping)

Early stopping이라는 기법은 너무나도 간단합니다.

말 그대로 오버피팅이 감지될 경우 학습 시간이 다 되지 않았다고 하더라도 '조기 종료'해버리는 것인데요, [그림 7]의 경우 400 epoch을 학습하기로 계획했지만, 오버피팅이 감지되니 280 epoch쯤에서 학습을 중단할 수 있습니다.

조기 종료 기능은 물론 코드 상으로 직접 구현해도 쉽지만, 요즈음의 기계학습 프레임워크에서는 이를 적용하기 쉽도록 관련 기능을 잘 패키징해둔 경우가 대부분입니다.

적용하기도 쉽고, 불필요하게 학습되는 경우를 방지하기 때문에 가급적 Early stopping을 적용해보는 것이 좋겠죠?

### 6.3.4 드롭아웃(Dropout)

드롭아웃은 학습 과정(Training pipeline)에서 일정 비율  $p$  만큼의 노드(인공 뉴런)를 무작위로 끄고 진행하는 Regularization 기법입니다.

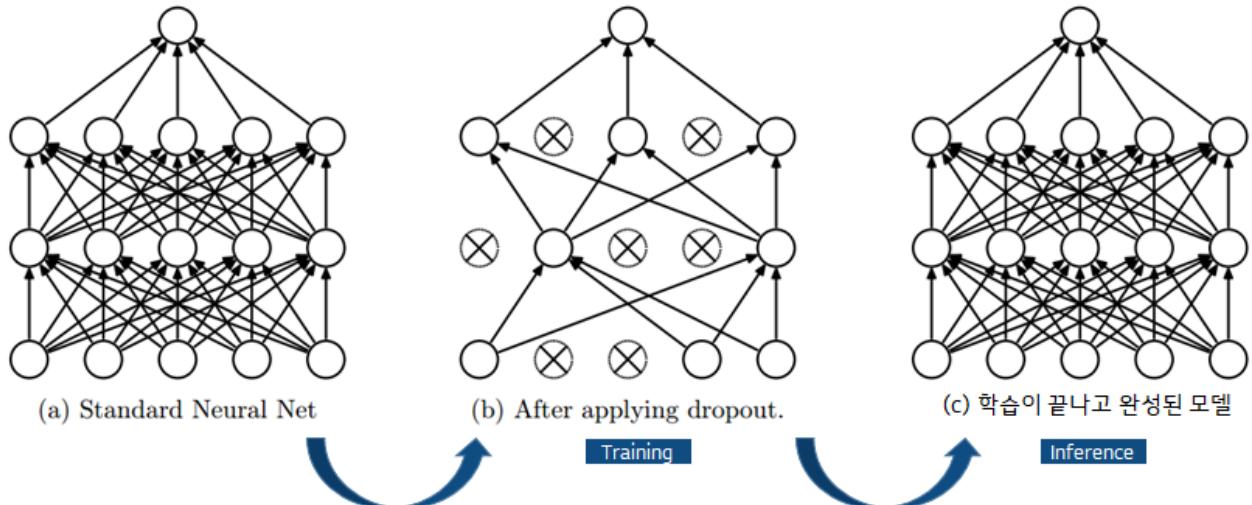
딥러닝 모델 학습 시 굉장히 많이 적용하는 Regularization 기법 중 하나인데요,

일부 노드가 사라진 상태에서 남아있는 노드만으로 어떻게든 정답을 맞춰야 하는 인공지능 모델은 훨씬 더 강력해진다고 하네요.

없어진 동료 노드의 뜻까지 수행해야 하니 훨씬 기능이 강화된 정예부대가 되는 것이죠.

휴가철 회사의 팀원과 마찬가지입니다.

휴가를 떠난 동료의 몫까지 수행해야 할 땐 힘들지만, 그만큼 한 명 한 명의 업무 스킬이 더 강화(?)된다고 볼 수 있겠지요.



[그림 10. 드롭아웃의 학습 과정 모습과 추론 과정 모습]

이외에도 L1/L2 Regularization, Batch Normalization, 앙상블 등.. 다양한 Regularization 기법이 있습니다.

여러가지 기법을 소개해드리는 것 보다는 Regularization의 취지가 무엇인지, 어떨 때 적용할 수 있는지를 기억하고 넘어가면 좋겠습니다. 😊

## 6.4 마무리

인공지능을 현장에 적용할 때도 '나무를 보지 말고 숲을 보라'는 말이 유효하게 적용됩니다.

이미 확보해놓은 과거의 데이터에 대해 좋은 성능을 낼 수 있도록 잘 학습하는 것,

이는 물론 굉장히 중요하지만 우리의 목적은 거기에서 끝나지 않습니다.

잘 만들어진 AI 모델이 향후 실제 고객사 현장에서도 좋은 성능을 내며 데이터를 처리할 수 있게 되는 것이 우리의 최종 목표일 겁니다.

위에서는 이를 Generalization이라는 용어로 소개드렸습니다.

알고 있는 것에만 몰두하여 새로운 환경에 적용하지 못하는 안타까운 일은 AI에서도 발생합니다.

오버피팅이 발생한다면 이를 해결하기 위한 다양한 Regularization 기법을 적용해보아야 합니다.

기계학습에서 한 번에 완벽한 모델이 만들어지는 경우는 없습니다.

첫 번째 모델의 성능이 엉망진창이라고 해도 실패한 것은 아닙니다.

다만 어떤 현상이 문제인지를 확인하고, 앞으로의 개선 전략을 잘 세워서 두 번째, 세 번째 모델을 만들 수 있으면 됩니다.

여기에서 의욕을 잃고 멈춰버린다거나, 고쳐나갈 방법을 모른다면 그땐 문제가 되겠지요.

이번 시간은 딥러닝 모델을 학습할 때 마주치기 쉬운 문제인 '오버피팅'에 대해 설명드렸습니다.  
그리고 이 문제를 해결하는 다양한 'Regularization'의 대표적인 기법 몇 가지를 살펴보았습니다.  
다음 시간은 '**인공지능 재활용하기, 전이학습(Transfer Learning)**'에 대해 알아보겠습니다.

감사합니다 😊

---

### 참고자료

- Justin Basilico, Making Netflix Machine Learning Algorithms Reliable, <https://www.slideshare.net/justinbasilico/making-netflix-machine-learning-algorithms-reliable>
- 자사 딥러닝 실무과정 교재보기, 김명지, [교재보기] 딥러닝 실무<sup>40</sup>
- Michael Nielsen, Neural Networks and Deep Learning, <http://neuralnetworksanddeeplearning.com/>

---

<sup>40</sup> <http://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

## 7 [7편] 다시쓰고 바꿔쓰자! 인공지능 재활용하기

---

안녕하세요, CTO AI 빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI 빅데이터연구소 AI 기술팀 김명지 책임연구원/Cognitive AI LAB<sup>41</sup>

---

- 쉽지 않은 인공지능 적용하기(see page 105)
    - 구체적이지 않으며 불명확한 태스크(see page 106)
    - 적은 데이터, 낮은 품질의 데이터(see page 108)
    - 다른 도메인 환경(see page 109)
  - Transfer Learning : 한번 만든 인공지능 모델 우려먹기(see page 111)
    - Catastrophic forgetting : 치명적인 기억상실!(see page 113)
    - 더 나은 Transfer Learning을 위한 방법(see page 113)
  - Transfer Learning 모델 이용(see page 115)
    - 컴퓨터 비전에서의 Transfer Learning(see page 115)
    - 자연어 이해(NLU)에서의 Transfer Learning(see page 117)
  - 마무리(see page 117)
- 

지난 시간에는 딥러닝 모델을 학습할 때 마주치기 쉬운 문제인 '오버피팅'에 대해 설명드렸습니다.

그리고 이 문제를 해결하는 다양한 'Regularization'의 대표적인 기법 몇 가지를 살펴보았습니다.

오늘은 한 번 만들어놓은 딥러닝 모델을 알뜰살뜰 다음 태스크에도 활용하는 전이학습(Transfer Learning)에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 7.1 쉽지 않은 인공지능 적용하기

AI, 그 중에서도 특히 딥러닝 모델을 현장에 적용하려는 시도를 해 보신 분이 있다면 모두 공감할 이야기가 있습니다.

"아니 AI는 왜 배울땐 모든 다 될 것 같았는데 내가 해보려고 하면 영망진창인거야??"

물론 어느 기술분야고 그렇지 않겠느냐마는 특히나 딥러닝 기반의 인공지능은 기술을 습득할 때와 실제 현장에 적용할 때의 괴리가 큰 분야중 하나입니다.

<sup>41</sup><https://wire.lgcns.com/confluence/display/~78628>



AI를 글로만 배운 사람

실제 현장에 적용된 AI 모델

[그림 1. 늘 이상과 현실은 괴리가 있기 마련이죠. 하지만 AI는 그게 너무 크지...]

최신 기술을 소개하는 논문부터 모델 학습에 필요한 상세 코드까지, 오픈 사이언스인 딥러닝은 누구나 접근 가능한 곳에 무료로 모든 자료가 공개되어 있습니다.

글로벌 탑 티어 학회에 소개된 논문은 순식간에 전세계의 연구자들에 의해 블로그에 쉽게 재해석되며 유튜브 영상으로 만들어지고,

깃털에 올라오는 공식 코드와 모델은 누구나 이용하기 편리하도록 2차 3차 패키징이 되곤 합니다.

온라인 커뮤니티는 최신 트렌드를 소개하고 다양한 기사와 의견, 자신만의 참신한 AI 프로젝트를 소개하는 사람들로 매일 붐빕니다.

하지만 이 수많은 자료들은 쉽고 편하게 잘 다져진 튜토리얼에 불과합니다.

대부분의 내용은 정제된 다량의 데이터와 잘 만들어진 환경, 제한된 태스크에 대한 것이죠.

이것들이 각자가 접한 실생활의 문제 해결을 위해 도입될 때에도 잘 작동하리라는 보장은 없습니다.

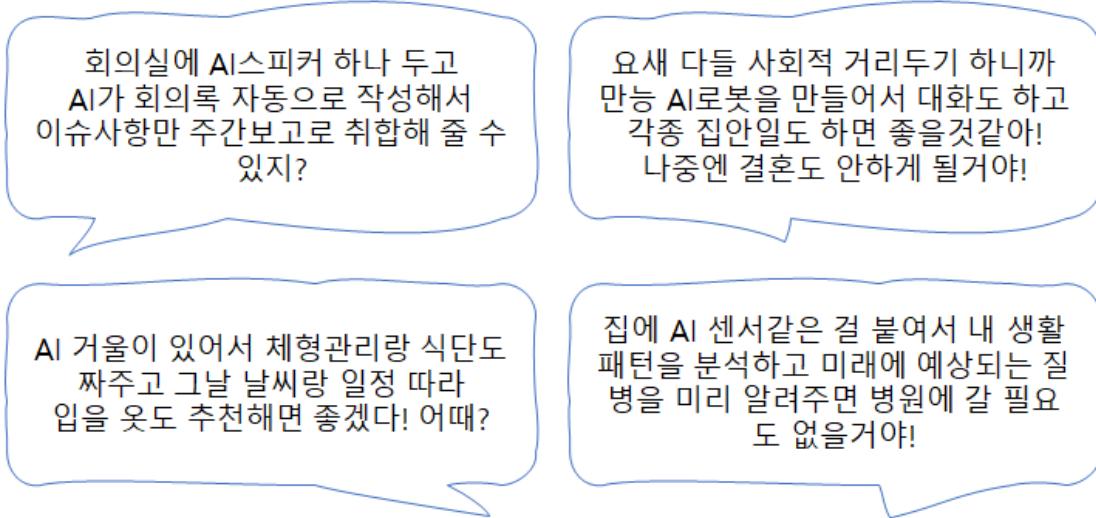
현실에는 다음과 같은 걸림돌이 있기 때문입니다.

### 7.1.1 구체적이지 않으며 불명확한 태스크

AI를 막 접하기 시작한 사람들 및 대다수의 고객은 AI에 대해 큰 기대를 하고 있습니다.

AI가 우리 삶의 문제의 모든 것을 해결해 줄 수 있을 거라고 막연히 생각하지요.

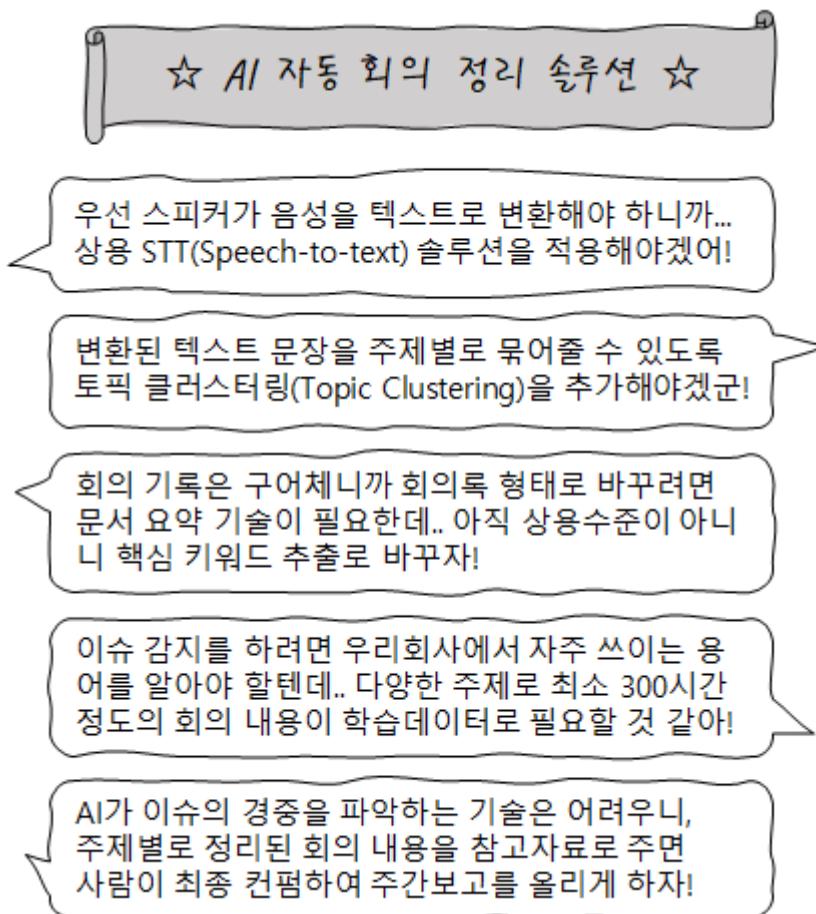
AI의 특장점이나 개념적인 부분, 포장된 사례만을 접했다면 꿈같은 요구사항을 내걸 수 있습니다.



[그림 2. 아뇨 그런거 못합니다... (※위 내용은 실제 고객의 요구사항과 무관합니다)]

AI는 삶의 많은 부분을 더 편리해 지도록 만들어 줄 수는 있겠지만, 모든 일에 대한 근본적인 해결책은 아닙니다. 두루뭉실하고 막연한 요구사항보다는 구체적이고 명확한 목표를 세우고 필요한 하위 기능을 쪼개어 생각해야 합니다.

예를 들어 첫 번째 요구사항의 경우,



[그림 3. 요구사항 구체적으로 break-down 하기]

등등의 구체적인 하위 기능으로 쪼개볼 수 있겠습니다.

하지만 대부분의 고객은 AI 전문가가 아니기 때문에 요구사항을 기능 단위로 쪼개기 어려운 경우가 많을 겁니다.

어떤 기능들이 AI로 가능한지, 이 중 어떤 것은 현실적으로 아직 상용화하기에 무리가 있을지, 어떤 부분은 굳이 AI가 아니어도 처리할 수 있는지 등은

기술을 깊이 이해하고 적용해 본 경험이 없는 사람이라면 쉽게 도출하기 어려운 내용입니다.

이런 경우 기술을 더 잘 알고 있는 우리가 상세 기능을 나누고 각 목표 수준을 정의한 다음 전체를 아우르는 구성을 어플리케이션 레벨로 설계할 수 있어야겠죠?

### 7.1.2 적은 데이터, 낮은 품질의 데이터

태스크가 구체적으로 정해졌다면 이제 AI 모델을 학습시킬 차례입니다.

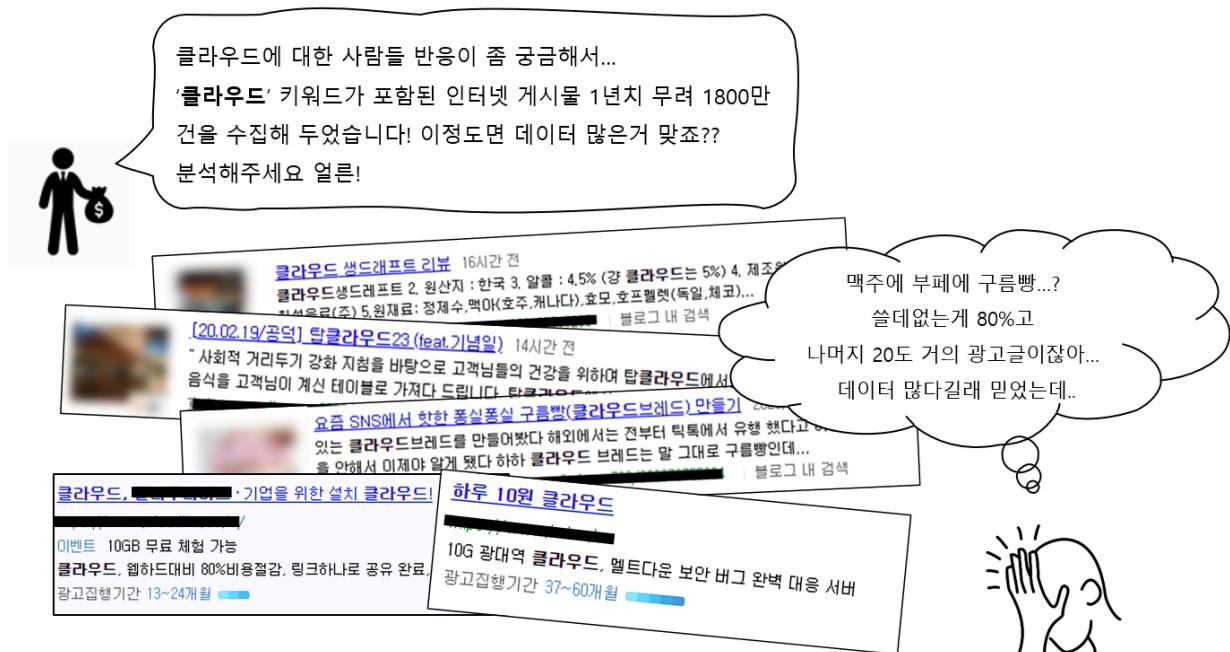
단순하고 정해진 패턴을 다루는 태스크라면 상관이 없지만, 이미지나 텍스트같은 비정형 데이터를 다루거나

코드로 판단 및 예측 규칙을 명문화하기 어려운 태스크라면 딥러닝 기반의 AI 모델을 이용하는 것이 좋습니다.

딥러닝 모델 학습은 사람이 규칙을 지정하지 않는 대신 데이터를 통해 패턴을 학습해야 하는데요,

그 학습 방법의 특성상 양질의 다양한 데이터를 필요로 합니다.

하지만 대부분의 경우 고객이 데이터를 차곡차곡 모아놓은 뒤 사업을 발주하는 경우는 드물고, 데이터를 보유하고 있다고 해도 학습데이터로 활용하기에 부적격인 경우가 다반사입니다.



[그림 4. 데이터 정제는 숙명입니다. (※위 내용은 실제 고객의 요구사항과 무관합니다)]

그렇다고 해서 학습 데이터를 마련하는 것부터 시작하자니, 이 또한 쉽지 않습니다.

요즈음은 인공지능 학습용 데이터를 라벨링하고 전처리해주는 크라우드 소싱 업체도 많이 생겼지만

그래도 여전히 데이터의 소스를 제공해야 합니다. (예: 공정품의 양품/불량 체크를 위해 공정품 사진을 찍어 제공)

이마저도 다량의 데이터를 구축할 경우 비용이 많이 들어갈 수도 있고, 대부분 고객데이터이다 보니 데이터의 반출이 불가능하여 외부 업체를 활용하기 어렵지요.

AI 모델 개발자는 늘 모자란 데이터와 싸워야 합니다.

### 7.1.3 다른 도메인 환경

도메인 환경이 달라지는 것도 문제가 됩니다.

아무리 오버피팅을 피하고 General하게 만든 모델이라고 해도 추론 환경이 달라지면 성능이 감소하기 마련입니다. (오버피팅 참고(see page 91))

예를 하나 들어보겠습니다.

동네에서 편의점을 운영중인 만득이는 아르바이트를 구하는 것이 어려워 야간시간에는 무인편의점으로 바꾸려 한다.

이에 CCTV를 여러 대 설치해두고, 계산하지 않고 나가는 손님이 발생하면 알림을 주도록 <AI 절도 감지 모델>을 만들기로 했다.



[그림 5. 무인 결제는 LG CNS가 담당할테니 만득이는 절도감지에 힘쓰라구~! (자료: LG CNS 본사 3층 무인편의점)]

만득이는 매장에서 다양한 절도 상황을 연출하여 데이터를 만들고, 모델을 학습시켰다.

감지 성능은 놀라웠다. 만득이의 모델은 한 달 동안 발생한 모든 절도 사건을 100% 검거했다.

"이런 매장이 우리 가게뿐이 아닐텐데, 이 모델을 다른 매장에 팔아 이용료를 받으면 나는 부자가 되겠구만!"

만득이는 해당 모델을 주변 편의점 점주들에게 팔았다. 하지만 결과는 꽝이었다.

만득이의 모델은 다른 편의점에서 전혀 절도를 감지하지 못했다.

만득이는 손해배상을 하기 위해 자신의 매장을 내놓아야 했다.

만득이가 간과한 것은 도메인 적응 문제(Domain adaptation problem)입니다.

대부분의 딥러닝 모델은 동일한 기능을 수행하는 모델이라고 해도 추론 환경이 달라지면 제 기능을 수행하지 못합니다.

다른 매장의 CCTV 화질, 밝기와 채도, 채광, 조명, 카메라 위치, 매장 내 크기, 선반 위치 등은 만득이의 매장과 상이할 겁니다.

이런 환경에서의 절도행위는 모델이 학습한 적이 없기 때문에 제대로 검거하지 못하게 되는 것이죠.

만득이네 편의점에서는 높은 검거율을 보였을지라도, 이는 만득이의 매장 환경에 한해서만 최적화되어 발휘된 성능입니다.

다양한 편의점에서도 잘 작동하려면 다양한 편의점의 절도 데이터를 만들어 학습시켜야 합니다. 또 다시 데이터 부족과 다양성 문제로 넘어가게 되는 것이죠.

이처럼 하나의 AI 모델이 좋은 성능을 보인 전적이 있다고 해도 다른 고객사 환경에서 여전히 잘 작동한다는 보장은 할 수 없습니다.

이렇게 실제 현장에 AI를 도입할 때에는 결과를 낙관할 수만은 없습니다.

옆에 아홉은 데이터 부족으로 모델 학습에 고생하고, 잘 만들어진 모델이라 해도 또 다른 곳에서 잘될 것이라 장담하지도 못합니다.

공부할 때 배웠던 잘나간다는 AI 모델들은 다 이론적인 허상에 불과할까요?

확장성 없이 매 번 수 만 건의 학습데이터를 만드는 첫바퀴를 반복해야 할까요?

다 방법이 있습니다!

## 7.2 Transfer Learning : 한번 만든 인공지능 모델 우려먹기

'전이 학습'으로도 불리는 **Transfer Learning**은 한 번 만들어진 딥러닝 모델을 재활용하여 쓸 수 있는 기법입니다.

비슷한 태스크를 다른 도메인에 적용할 때, 그리고 그 태스크를 위한 학습 데이터가 부족한 경우 유용하게 쓰일 수 있습니다.

유참고로, 유사 용어 Fine-tuning(미세조정)이 있긴 합니다만, 더 넓은 의미에서 Transfer Learning을 알아둡시다.

### [참고] Fine-tuning: 해묵은 인공지능 모델 다시쓰기

Fine-Tuning은 더 정교하게 파라미터를 튜닝하는 것으로,

온라인 프로세스([참고\(see page 0\)](#))에 적용된 AI 모델이 데이터 패턴 변경 등으로 인해 추론(inference) 성능이 떨어지게 될 때

이를 보강하고자 기존 모델에 새로운 데이터 등을 넣어서 파라미터를 재학습하는 작업을 말합니다.

다시말해서 동일한 태스크에 대해 한번 만들어진 모델의 성능을 보강하기 위한 목적입니다.

예를 들어 쇼핑몰 리뷰를 긍정/부정 분류하는 모델이 있다고 가정합시다.

시대 흐름에 따라 신조어가 등장하고, 새로운 상품에 대한 리뷰가 등장하면 모델의 분류 성능이 저하될 수 있습니다.

이 때 새로운 데이터를 추가하여 인공지능을 이어 학습시키면 새로이 등장한 패턴에 대해서도 잘 분류하게 됩니다.

즉 기존 모델의 v2, v3 .. 업데이트 버전을 위한 기법이라고 볼 수 있겠네요.

반면 Transfer Learning은 대체로 만들어진 모델을 다른 태스크에 적용하려 할 경우 활용하는 기법입니다.

예를 들어 아래와 같은 다양한 경우 Transfer Learning을 적용해볼 수 있습니다.

- 만득이네 편의점 절도 감지 모델을 짱구네 편의점에 적용하려 할 경우
- 영화 리뷰 평가에 대한 긍/부정 분류 모델을 뉴스기사 댓글 긍/부정 분류에 적용하려 할 경우
- A 회사의 이메일 보안 위반 탐지 모델을 B 회사에 적용하려 할 경우
- 일상생활의 다양한 이미지를 분류하는 모델을 농산물 품종 이미지 구분에 적용하려 할 경우
- 일반 상식에 대한 질의응답 모델을 금융권 콜센터 무인자동 질의응답 과제에 적용하려 할 경우
- ... 등등

비슷한 경우에 대해 만들어 놓은 AI 모델이 있다면 다른 태스크 진행시 맨땅에서부터 모델을 만들 필요가 없습니다.

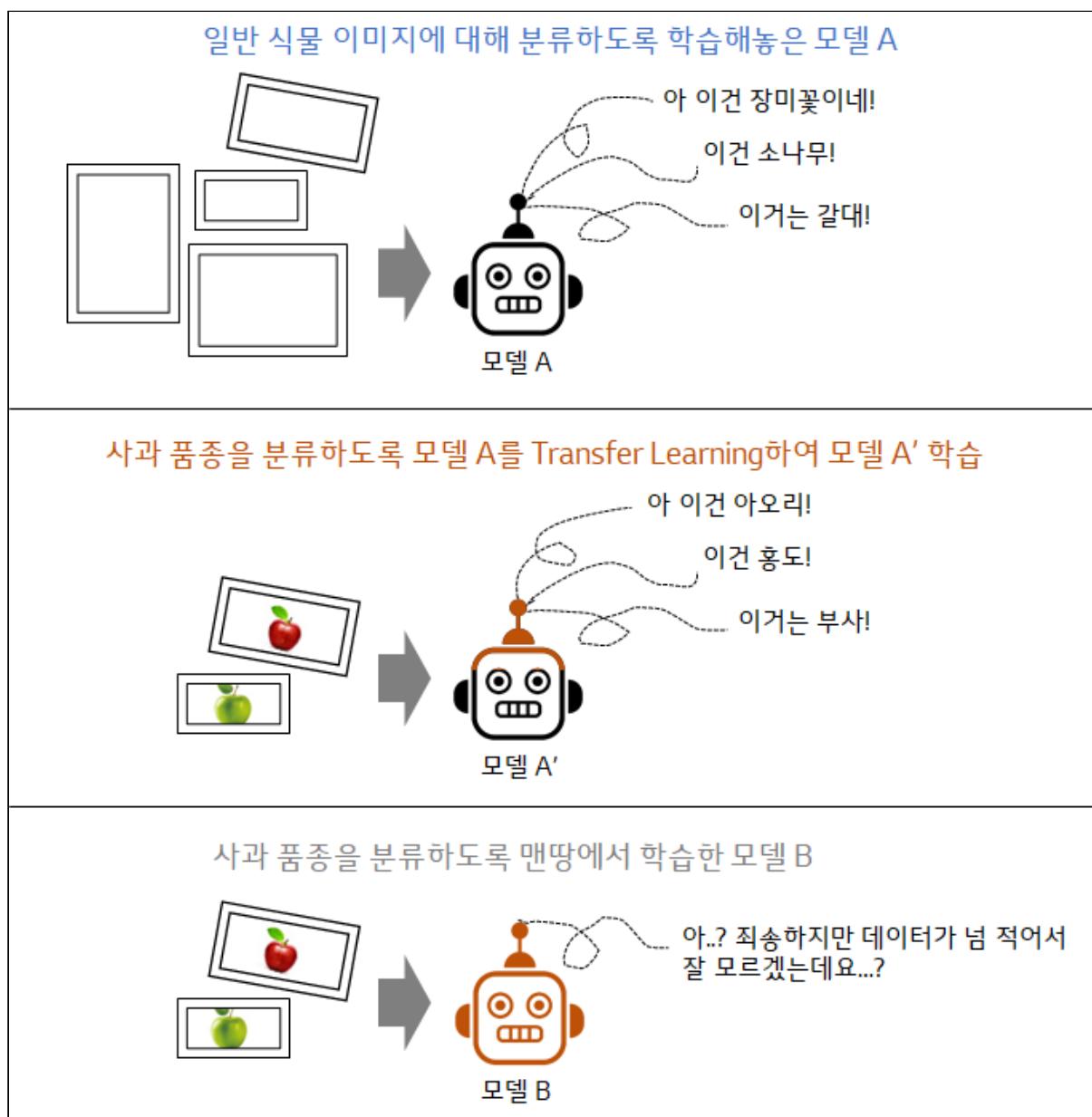
하지만 엄연히 환경이 다르고 데이터의 패턴이 다르기 때문에 그 모델을 그대로 활용해서는 좋은 성능이 나올 수 없죠.

Transfer Learning은 하나의 모델이 이미 배워놓은 지식은 잘 유지하면서도, 새로운 태스크에 대해 필요한 지식을 추가로 습득할 수 있도록 합니다.

이미 만들어 놓은 모델의 아키텍처를 새 태스크에 맞게 조금 수정하거나 추가하는 작업을 한 뒤

새로운 태스크에 대한 학습 데이터를 이어서 학습시킨다면 처음부터 학습한 모델보다 좋은 성능을 낼 수 있습니다.

새로운 학습데이터가 기존만큼 많지 않아도 말이죠.



[그림 6. 사과 품종 분류를 위한 Transfer Learning 예]

예를 들어 사과 농장으로부터 의뢰를 받아 사과 사진으로 사과 품종을 자동 분류하는 과제를 수행한다고 가정하겠습니다.

하지만 확보한 품종별 사과 사진이 몇 장 없는 어려운 상황입니다.

이대로 학습한다면 몇 장 없는 이미지로부터 모델이 충분히 특징을 학습하지 못하여 분류를 제대로 수행하지 못하게 됩니다. (모델 B)

하지만 약간 갈래는 다른듯 하나 일반 식물의 종류를 구분하는 AI 모델(모델 A)을 만든 적은 있습니다.

만들어둔 모델에 어떻게든 약간 숟가락을 얹어서 사과도 잘 분류하게 할 수 있을까요?

우선 모델 A의 마지막 부분은 일반 식물들을 분류하도록 되어있을테니, 사과 품종 카테고리 수에 맞게 분류할 수 있도록 인공신경망을 조금 수정해주어야 합니다.

이후 주어진 소량의 사과 이미지 데이터를 여기에 추가로 학습시켜서 새로운 모델 A'를 만들어 냅니다.

모델 A는 기존에 '식물 이미지' 데이터의 일반적인 특징(색상, 각도, 선, 열매, 잎사귀 등..)을 학습한 적이 있으니,

Transfer Learning을 적용한 모델 A'가 처음부터 소량의 사과만 학습한 모델 B보다는 더 적은 학습량으로 더 많은 지식을 가질 수 있는 것이죠.

### 7.2.1 Catastrophic forgetting : 치명적인 기억상실!

태스크에 맞게 수정하여 추가로 학습하면 모든 것이 좋아질 듯 하지만 이것이 만능 해결책은 아닙니다.

딥러닝 모델이 새로운 정보를 학습할 때 이전에 배웠던 정보를 완전히 잊어 버리는 경향이 발생할 수 있는데, 이를 Catastrophic forgetting이라고 합니다.

사과 분류 모델의 예시로 돌아가봅시다.

모델 A'가 몇 장 없는 사과 데이터에 대해 너무 여러번 반복학습하다 보면 모델 A가 이전에 학습했던 기본적인 식물 이미지의 특징들을 잊어버리게 되는 것이지요.

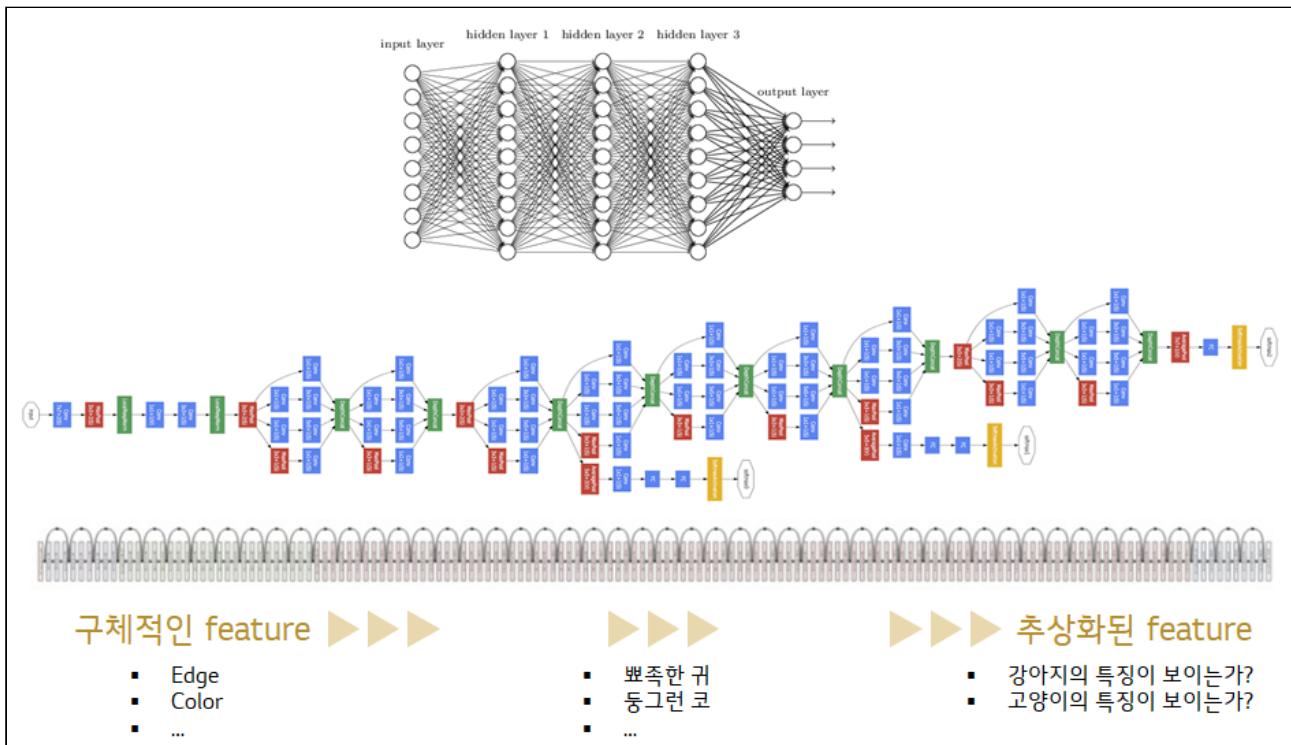
전이(Transfer)가 필요 이상으로 과하게 일어났다고 볼 수 있겠습니다.

Catastrophic forgetting이 발생하지 않도록 하면서도 새로운 데이터를 잘 배우기 위해서는 다음과 같은 기법을 활용해야 합니다,

### 7.2.2 더 나은 Transfer Learning을 위한 방법

일반적인 딥러닝 모델은 두 층(layer) 이상의 인공신경망으로 구성되어 있습니다.

이 신경망은 초반에는 데이터의 구체적이고 기본적인 특징을, 후반부로 갈수록 특정 태스크를 위한 추상적이고 개념적인 특징을 학습하게 됩니다.



[그림 7. 강아지와 고양이를 분류하는 인공신경망 예]

이미지든 텍스트든 이미 학습해두었던 모델은 해당 데이터의 기본적인 특징을 어느정도 알고 있다고 볼 수 있습니다.

따라서 새 태스크에 기 보유 모델을 Transfer Learning 하려 할 땐 기본 특징 학습은 건너 뛰고 바로 태스크를 위한 학습으로 가는 것이 좋습니다.

이를 위해 후반부의 신경망 층에 대해서만 파라미터를 학습하고, 전반부의 파라미터는 학습되지 않도록 고정해놓는 기법을 적용할 수 있습니다.

이를 **레이어 동결(Layer freezing)**이라 부릅니다.

새로운 태스크가 기 학습한 태스크와 크게 상이하지 않은 경우라면 대부분을 동결시켜도 되고,

태스크가 많이 상이한 경우라면 동결을 풀어서 조금 더 앞쪽의 층까지 학습할 수 있도록 설정할 수도 있습니다.

또는 다 동결시켜놓았다가 학습이 진행될수록 후반부부터 서서히 동결을 푸는 방법(Gradual Unfreezing)도 적용할 수 있으니 상황에 맞게 유연하게 대처해 볼 수 있습니다.

이외에 층마다 Learning rate(학습률)의 차별을 두는 것도 한 방법입니다. (**Discriminative fine-tuning**)

Learning rate이란 한 번에 인공신경망의 파라미터를 얼마만큼 업데이트 시킬지에 대한 정도로, 사람이 설정하는 값 (hyperparameter)입니다.

전반부의 인공신경망은 기본적인 특징을 이미 배워놨으니 새롭게 고칠 것이 별로 없을 것이고

따라서 새로운 태스크에 적용할 때 Learning rate을 작게 설정하는 것이 좋습니다.

반면 후반부의 인공신경망은 특정 태스크를 위한 부분으로, 새롭게 배워나가야 하는 부분이 많을테니

Learning rate을 크게 설정하여 빠르게 성큼성큼 학습하는 편이 좋습니다.

이외에도 Transfer Learning을 효과적으로 하기 위한 다양한 기법이 있으니, 상황에 맞는 기법을 활용할 수 있어야 합니다.

## 7.3 Transfer Learning 모델 이용

Transfer Learning은 보통 오픈도메인 데이터에 대해 만들어놓은 모델을 특정 도메인의 태스크에 적용하는 식으로 활용합니다.

일반적인 내용을 두루두루 넓게 학습해놓은 모델의 사전 지식을 구체적인 하위 영역에 활용하는 셈입니다.

### 7.3.1 컴퓨터 비전에서의 Transfer Learning

이미지를 입력 데이터로 처리하는 경우라면 이미지넷([참고\(see page 0\)](#)) 데이터로 학습된 모델에 Transfer Learning을 적용하여 좋은 성능을 낼 수 있습니다.

이미지넷은 무려 120만 장의 학습 이미지를 1,000개 카테고리로 분류하는 과제이기 때문에, 여기에 대해 학습한 모델은 이미지를 꽤나 잘 배웠다고 볼 수 있습니다.

특히 이미지넷 데이터 인식 대회에서 우승을 했던 GoogleNet(2014 우승)이나 ResNet(2015 우승) 등은 지금까지도 널리 활용되는 기본 모델이지요.



[그림 8. ResNet의 구성]

이외에도 다양한 특장점을 지닌 모델들의 아키텍처가 전부 논문으로 공개되어있으며, 이 중 대부분은깃협 등에 학습된 모델이 오픈되어있습니다.

이미지 처리를 하는 과제를 진행중이라면, 밑바닥부터 학습시키기보다는 이런 자료를 활용하여 Transfer Learning 하는 것이 좋겠죠?

### 7.3.2 자연어 이해(NLU)에서의 Transfer Learning

텍스트란 특허나 함축적인 데이터 탑이기 때문에 Transfer Learning을 활용하는 것이 매우 효과적입니다.

위키백과와 같은 방대하고 일반적인 지식에 대해서 학습한 적이 있는 모델이라면 일반적인 어휘의 의미를 대강 알고 있는 모델이라고 볼 수 있습니다.

이런 모델이 공개되어있다면 Transfer Learning에 활용하는 것이 좋겠지요.

자사에서 만들어 공개한 KorQuAD(Korean Question Answering Dataset)는 한국어 위키백과를 소스로 한 오픈도메인 질의응답 데이터인데요,



[그림 9. LG CNS가 만들고 공개한 한국어 표준 질의응답 데이터 KorQuAD ([바로가기](#)<sup>42</sup>)]

자연어 질의응답이란 인공지능이 수행하기 상당히 어렵고 복잡한 과제 중 하나입니다. 따라서 다양한 학습 데이터를 필요로 합니다.

하지만 KorQuAD로 학습해둔 모델이 있다면 콜센터나 챗봇 등 특정 고객사의 질의응답 데이터가 부족해도 상대적으로 좋은 성능을 얻을 수 있습니다.

## 7.4 마무리

여러 번 언급하지만, 딥러닝은 데이터로부터 패턴을 학습하여 의사결정하는 모델이기 때문에 양질의 데이터가 충분한 환경이라면 아무 문제 없습니다.

하지만 현실적으로 우리는 데이터가 부족한 상황에 놓이는 경우가 대부분입니다.

과거 경험에 이어 소량의 데이터를 추가 학습하여 좋은 성능을 짜내보자! 하는 것이 Transfer Learning이라는 AI 재활용 기법입니다.

AI 공부를 하려고 이론을 배우거나 튜토리얼 코드를 실습할 때 필요한 것이 아닌, 현장을 위한 기술이죠.

그래서 분류(Classification)과 같이 어느정도 기술수준이 성숙된 과제에서는 Transfer Learning을 얼마나 잘 적용할 수 있느냐가 인기있는 후속 연구주제입니다.

AI 모델을 제대로 학습시키기 여려운 만큼, 한 번 만들어 놓았다면 알뜰살뜰하게 끝까지 잘 쓸 수 있어야겠습니다.

이번 시간은 딥러닝 모델을 재활용하는 Transfer Learning에 대해 설명드렸습니다.

<sup>42</sup> <https://korquad.github.io/>

Transfer Learning은 일반적인 오픈도메인 대해 사전학습해놓은 모델을 특정 close-domain의 태스크에 적용할 때도 활용할 수 있는데요,

다음 시간에는 이러한 몸풀기 학습이라 할 수 있는 '**AI 사전학습 및 자기주도학습(Pre-Training and Self-supervised Learning)**'에 대해 자세히 알아보겠습니다.

감사합니다 😊

## 참고자료

- 무인편의점 '스마트 GS25' 가 보니, 동아일보, <https://www.donga.com/news//article/all/20180928/92185150/3>
- Transfer learning & fine-tuning, Keras, [https://keras.io/guides/transfer\\_learning/](https://keras.io/guides/transfer_learning/)
- 위키피디아 - Transfer learning, [https://en.wikipedia.org/wiki/Transfer\\_learning](https://en.wikipedia.org/wiki/Transfer_learning)
- 위키피디아 - Catastrophic inference, [https://en.wikipedia.org/wiki/Catastrophic\\_interference](https://en.wikipedia.org/wiki/Catastrophic_interference)
- Universal Language Model Fine-tuning for Test Classification, J. Howard & S. Ruder, 2018, <https://arxiv.org/abs/1801.06146>
- 자사 딥러닝 실무과정 교재보기, 김명지, [\[교재보기\] 딥러닝 실무<sup>43</sup>](#)
- Going Deeper with Convolutions, C. Szegedy, et al., 2014, <https://arxiv.org/abs/1409.4842>
- Deep Residual Learning for Image Recognition, K. He, et al., 2015, <https://arxiv.org/abs/1512.03385>
- KorQuAD, [https://korquad.github.io<sup>44</sup>](https://korquad.github.io)

<sup>43</sup> <http://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

<sup>44</sup> <https://korquad.github.io/>

## 8 [8편] 준비된 인공지능, Pre-trained AI

---

안녕하세요, CTO AI 빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI 빅데이터연구소 AI 기술팀 김명지 책임연구원/Cognitive AI LAB<sup>45</sup>

---

- Pre-training: 니가 뭘 요청할지 몰라서 일단 다 공부해놨어(see page 119)
  - 대규모 데이터에 대한 Pre-training(see page 121)
    - 시각 데이터에 대한 사전학습(see page 121)
    - 언어 데이터에 대한 사전학습(see page 123)
  - Self-Supervised Learning: 나 혼자 어떻게든 해볼게(see page 125)
    - 예: 이미지 데이터를 위한 Self-Supervised Learning(see page 129)
    - 예: 텍스트 데이터를 위한 Self-Supervised Learning(see page 131)
    - 예: Google BERT(Bidirectional Encoder Representations from Transformers)(see page 132)
  - 마무리(see page 133)
- 

특정 태스크에 인공지능을 적용하는 것은 쉬운 일이 아닙니다.

태스크를 위한 인공지능 학습용 데이터를 마련하고 라벨을 붙여야 하고,

최적의 모델을 만들기 위해서는 다양한 하이퍼파라미터 설정을 조합하여 실험을 반복해야 합니다.

이렇게 만들어진 모델이 새로운 데이터에 대해서도 좋은 성능을 보이는지 확인하고, 개선해나가는 것도 중요하지요.

매 번 아키텍처를 고민하고 학습하는 것은 힘들기에, 지난시간에 Transfer Learning(전이학습)에 대해 소개하였습니다.

Transfer Learning은 한 번 만든 딥러닝 모델을 다른 유사 태스크에 바꾸어 재활용하는 방법입니다.

오늘은 미리 준비된 인공지능 모델, 인공지능 사전학습(Pre-training)에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 8.1 Pre-training: 니가 뭘 요청할지 몰라서 일단 다 공부해놨어

Transfer Learning을 써보려고 해도, 비슷한 태스크에 대해 만들어놓은 딥러닝 모델이 없다면 재활용을 할 수 없습니다.

<sup>45</sup><https://wire.lgcns.com/confluence/display/~78628>

전혀 다른 것에 대해 학습한 모델이라면 Transfer Learning을 해도 효과가 없을 가능성이 매우 큽니다.

웹툰 작가에게 갑자기 장대높이뛰기 경기에 나가달라고 요구하는 것과 같이, 모델간 사전지식을 잘 활용하기 어렵겠죠.



[그림 1. 예체능이라고 다 같은 계열이 아니듯이 AI라고 해서 두루두루 다 되는 건 아닙니다.]

그렇다면 지식을 전수하는 Transfer Learning은 적용할 수 있는 경우가 많지 않겠는데요?

Transfer Learning은 아주 유사한, 한정된 태스크끼리만 가능한 걸까요?

전혀 다른 태스크간에는 지식을 전수하기 쉽지 않습니다.

하지만 전이 학습을 염두에 두고 다방면으로 활용할 수 있는 모델을 미리 만들어놓을 수 있습니다.

여러 태스크에 활용하기 위해 여러 지식을 미리 두루두루 학습해놓은 인공지능을 만드는 것이지요.

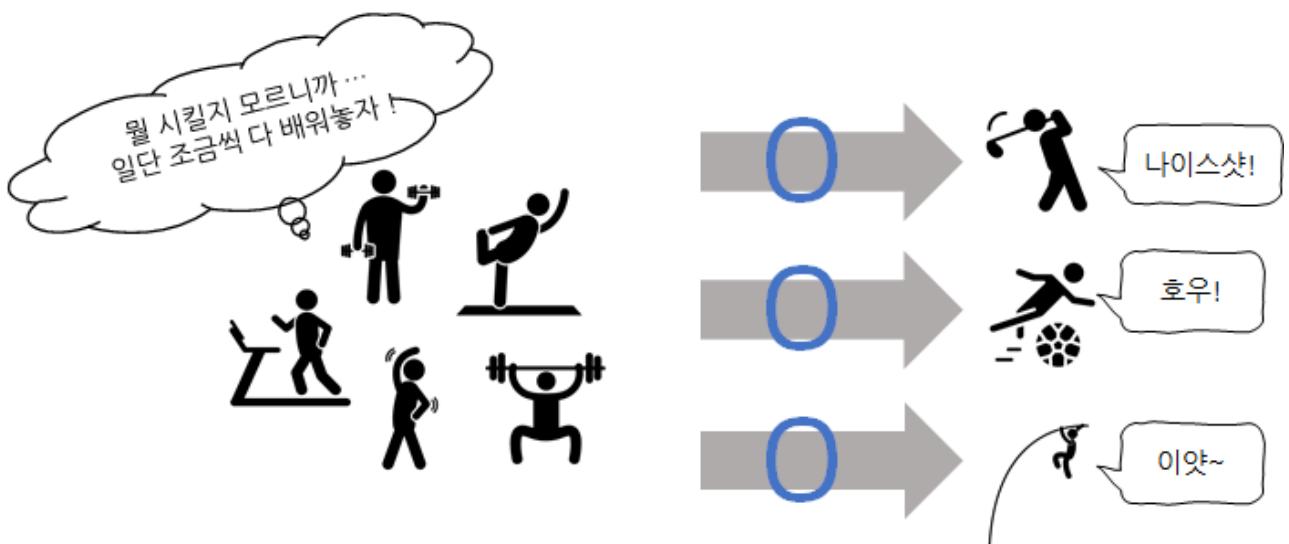
이러한 학습을 **Pre-training**, 또는 **사전학습**이라고 하며 이러한 용도의 모델을 **Pre-trained Model**, 즉 **사전학습 모델**이라고 합니다.

사전학습은 보통 특정 데이터타입에 대한 일반적인 지식을 두루 배워놓는 것을 목표로 합니다.

텍스트 사전학습 모델은 언어의 일반적인 의미와 구조에 대해, 이미지 사전학습 모델은 이미지의 일반적인 특징, 색채, 형태 등에 대해 배웁니다.

그래서 향후 어떤 텍스트나 이미지 관련 태스크를 수행한다 해도 기본 지식을 바탕으로 잘 적용할 수 있습니다.

예를 들어 기본적인 웨이트 트레이닝, 유산소 운동, 스트레칭 등을 통해 몸 쓰는 법을 배웠다면 어떤 운동 종목이든 적응이 수월한 것처럼요.



[그림 2. 골고루 배워놓는 사전학습(Pre-training)]

## 8.2 대규모 데이터에 대한 Pre-training

어떤 후속 태스크에 적용하든 잘 수행할 수 있도록 하려면 방대한 양의 지식을 골고루 배워놓는 것이 좋습니다.

따라서 모델의 사전학습은 대규모의 오픈도메인 데이터에 대해 이뤄지는 것이 일반적인데요,

이미지와 텍스트에 대한 대표적인 사전학습 데이터와 태스크는 다음과 같습니다.

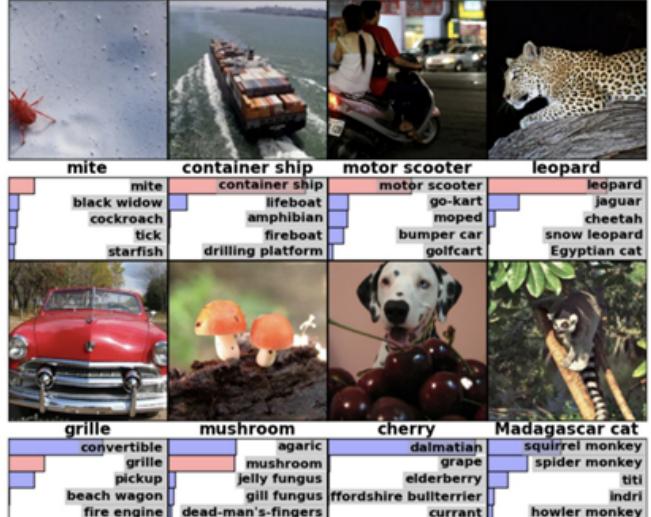
### 8.2.1 시각 데이터에 대한 사전학습

테크레터에서도 벌써 여러 번 등장한 이미지넷(참고(see page 0)) 데이터 인식 대회는 가장 유명한 이미지 사전학습 과제라고 할 수 있습니다.

120만 장의 학습용 이미지를 가지고 1000개 카테고리로 분류하는 이 과제는 대표적인 인공지능 이미지 인식 태스크입니다.



- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



<b>mite</b>	<b>container ship</b>	<b>motor scooter</b>	<b>leopard</b>
black widow cockroach tick starfish	lifeboat amphibian fireboat drilling platform	motor scooter go-kart moped bumper car golfcart	leopard jaguar cheetah snow leopard Egyptian cat
<b>grille</b>	<b>mushroom</b>	<b>cherry</b>	<b>Madagascar cat</b>
convertible grille pickup beach wagon fire engine	agaric mushroom jelly fungus gill fungus dead-man's-fingers	dalmatian grape elderberry ffordshire bulterrier currant	squirrel monkey spider monkey titi indri howler monkey

**Classification task:** produce a list of object categories present in image. 1000 categories.  
 "Top 5 error": rate at which the model does not output correct label in top 5 predictions

Other tasks include:  
 single-object localization, object detection from video/image, scene classification, scene parsing

[그림 3. ImageNet dataset 분류 대회(ILSVRC<sup>46</sup>)]

120만 장이나 되는 컬러 이미지를 1000개나 되는 카테고리로 분류하도록 학습된 모델은 일반적인 이미지의 특징 대부분을 다뤄봤다고 봐도 무방합니다.

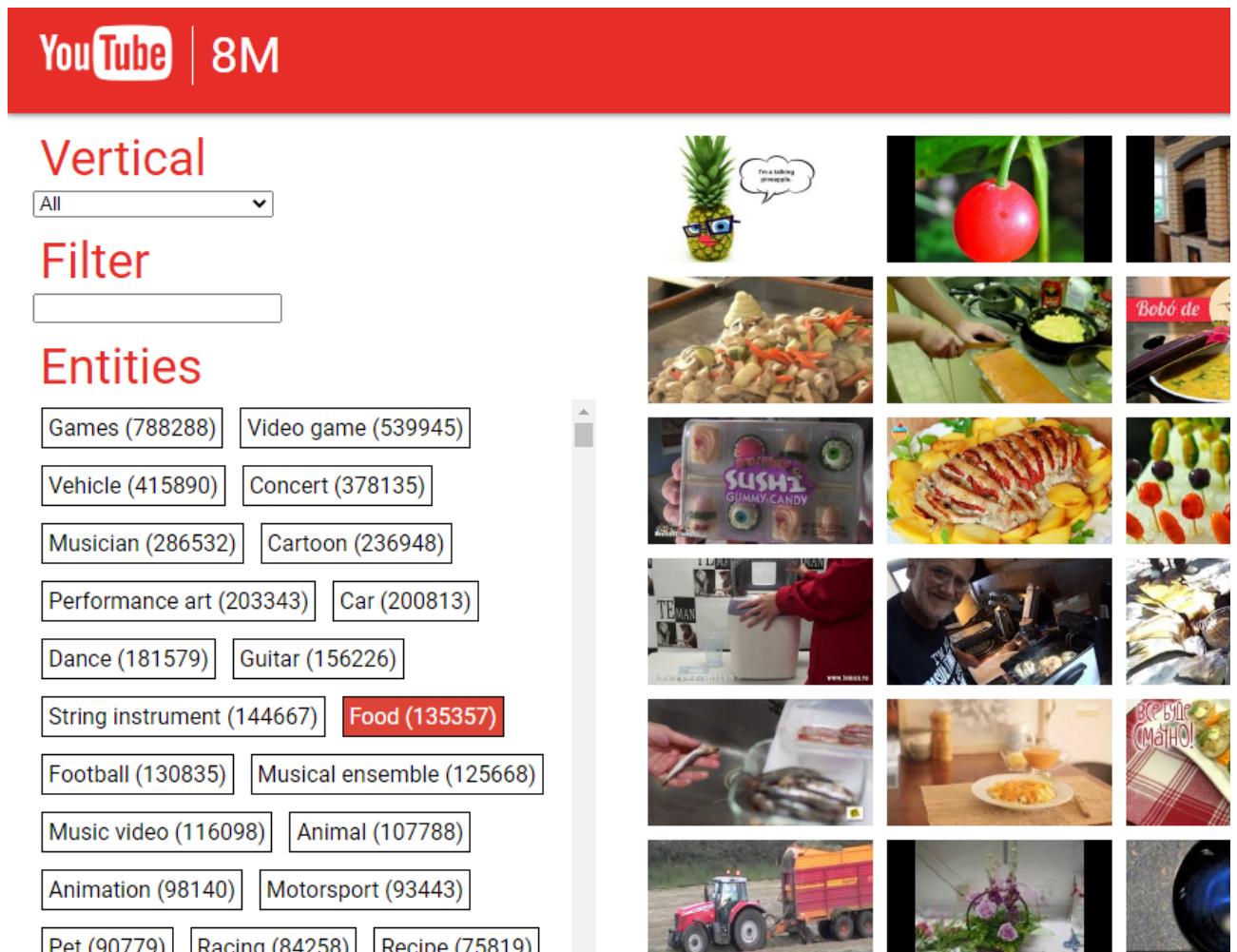
어떤 색상들이 나타나는지, 등장하는 사물의 직선, 곡선, 또한 이들이 합쳐져서 이루는 도형, 큰 개체와 작은 개체, 전경과 배경 등을 학습했겠지요.

다양한 내용을 두루 살펴보았으니 이미지를 대상으로 하는 어떤 태스크에 적용된다고 해도 사전지식을 기반으로 빠르게 학습할 수 있을 것입니다.

정지된 이미지 뿐 아니라 동영상에 대해서도 비슷한 사전학습용 데이터가 있습니다.

구글이 공개한 **Youtube-8M**은 무려 총 35만 시간에 달하는 610만개의 비디오를 약 3800개의 카테고리로 다중 분류하는 데 활용할 수 있습니다.

46 <http://www.image-net.org/challenges/LSVRC/>



[그림 4. YouTube-8M dataset ([링크](#)<sup>47</sup>)]

### 8.2.2 언어 데이터에 대한 사전학습

언어에 대해 전반적으로 배워놓는다는 것은 문맥에 따라 활용되는 단어의 의미, 뉴앙스, 적절한 문체 등을 습득한다는 것입니다.

자연어의 경우 이미지나 비디오 데이터와는 달리, 언어권의 차이로 인해 데이터를 언어권별로 각각 수집하여 학습시켜야 하는 문제가 있습니다.

영어의 경우 공개된 데이터가 많지만 한국어나 기타 언어에 대해서는 다량의 표준 데이터를 구하기 쉽지 않죠. 하지만 무난하게 활용하기 좋은 데이터로는 **위키피디아(Wikipedia)**가 있습니다.

47 <https://research.google.com/youtube8m/index.html>



The screenshot shows the English Wikipedia page for "Database download". The top navigation bar includes links for "Project page", "Talk", "Read", "View source", "View history", "Search Wikipedia", and "Log in". The main title is "Wikipedia:Database download" with the subtitle "From Wikipedia, the free encyclopedia". Below the title, a note says "For scheduling, related tools etc., see [m:Data dumps](#)". A callout box states: "This help page is a how-to guide. It details processes or procedures of some aspect(s) of Wikipedia's norms and practices. It is not one of Wikipedia's policies or guidelines, and may reflect varying levels of consensus and vetting." To the right, there are "Shortcuts" for "WP:DUMP" and "WP:DUMPS". On the left sidebar, there are links for "Main page", "Contents", "Current events", "Random article", "About Wikipedia", "Contact us", "Donate", "Contribute", "Help", "Learn to edit", "Community portal", "Recent changes", "Upload file", and "Tools". On the right sidebar, there is a "Readers' FAQ" section with a question mark icon and links to various Wikipedia policies.

[그림 5. 위키피디아 덤프 데이터셋 다운로드([링크](#)<sup>48</sup>)]

위키피디아의 경우 전 분야에 걸친 백과사전 지식을 대상으로 하기 때문에 텍스트 모델 사전학습을 시킬 수 있는 대표적인 데이터입니다.

한국어로도 다운받을 수 있고, 이외 여러 언어에 대해서도 제공하고 있습니다.

한국어의 경우 **나무위키** 데이터도 다운로드 받을 수 있는데, 조금 더 구어체에 가깝고 자연스러운 표현을 학습할 수 있습니다.

이외에 국립국어원에서 공개하는 **세종말뭉치**가 있습니다.

<sup>48</sup> [https://en.wikipedia.org/wiki/Wikipedia:Database\\_download](https://en.wikipedia.org/wiki/Wikipedia:Database_download)

말뭉치 분류	매체 및 장르(1단계)	매체 부가정보(2단계)
현대문어	잡지(주간지, 월간지, 계간지)	경제, 과학, 국제, 기타(독자투고, 인물, 화제, 응대 등), 문화, 보도, 사설, 사회, 생활, 스포츠, 연극, 오피니언, 정치, 총류, 취미, 칼럼
현대문어	잡지(주간지, 월간지, 계간지)	교육자료, 사회, 상상적 텍스트, 생활, 예술론, 인문, 자연, 체험기술적 텍스트, 총류
현대문어	책	교육자료, 기타(독자투고, 인물, 화제, 응대 등), 사회, 상상적 텍스트, 생활, 수필, 예술, 예술론, 인문, 자연, 정보, 체험 기술적 텍스트, 총류
현대문어	기타 출판물 (안내문, 소책자, 정부 문서 등)	예술론, 인문, 실기 르포
현대문어	화면이 있는 방송 녹화 전사	생활 대화, 사회 대화, 녹화/사회, 총류
현대구어	화면이 있는 방송 녹화 전사	사회, 생활, 예술론, 인문, 총류
현대문어	화면이 없는 방송 녹음 전사	예술론, 총류
현대구어	화면이 없는 방송 녹음 전사	사회, 생활, 신문, 예술론, 인문, 자연, 체험기술적 텍스트, 총류
현대문어	기타 녹음 전사	예술론
현대구어	기타 녹음 전사	-
현대문어	전자출판물	과학, 사회, 생활, 예술론, 인문, 자연, 체험기술적 텍스트, 토론, 총류

[그림 6. 국립국어원 세종말뭉치([링크](#)<sup>49</sup>)]

여러 매체로부터 모은 현대 문어/구어체를 제공하고 있어서 경우에 따라 활용하기 좋습니다.

이외에 뉴스, 리뷰 등의 데이터도 자주 활용되는 사전학습용 데이터입니다.

### 8.3 Self-Supervised Learning: 나 혼자 어떻게든 해볼게

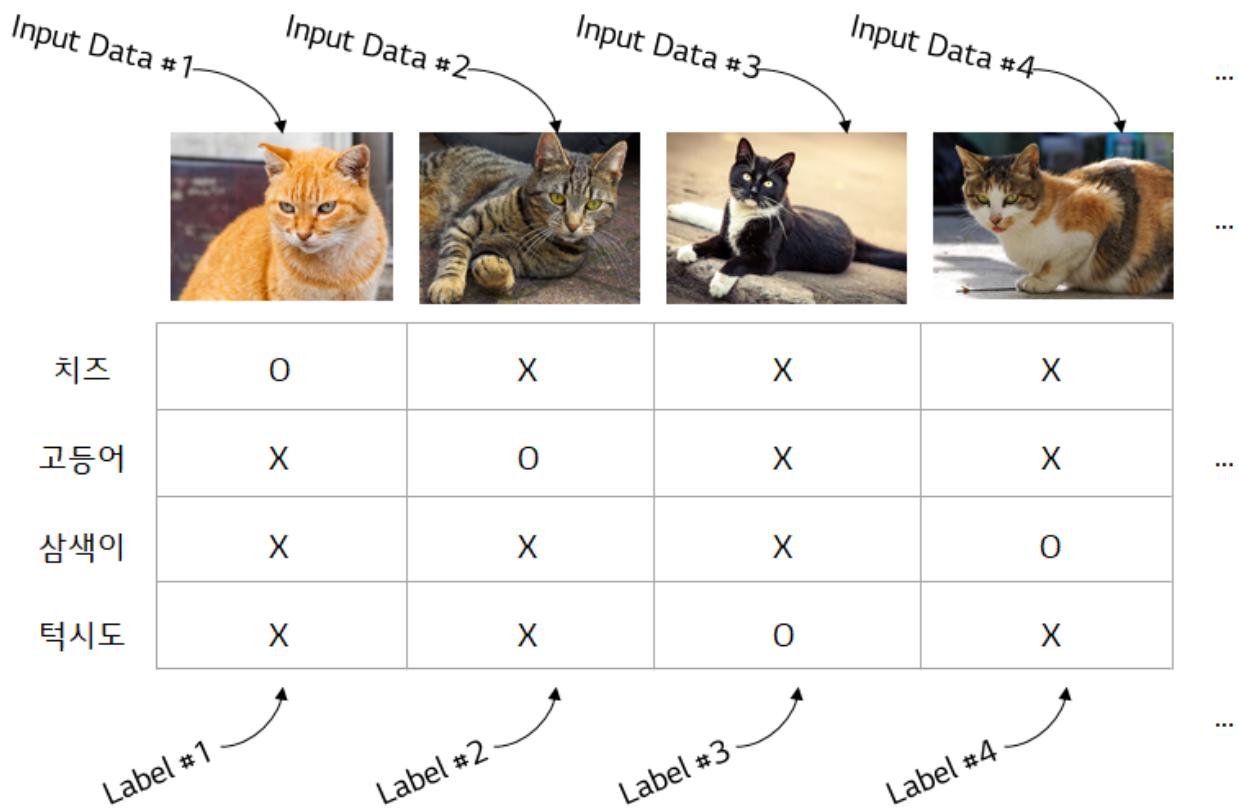
하지만 대규모로 구할 수 있는 데이터라 해도 라벨까지 잘 달려있는 경우는 드뭅니다.

라벨(label)이라 함은 데이터에 대해 인공지능이 예측하기를 희망하는 결과입니다.

예를 들어, 한국 길고양이 종류를 구별한다고 가정하겠습니다.

학습 목적은 고양이 사진을 넣었을 때 어떤 종류의 고양이인지 추론하는 것입니다.

49 <https://ithub.korean.go.kr/user/guide/corpus/guide1.do>



[그림 7] 길고양이 종류 구별용 데이터에 대한 Input data와 Label]

[그림 7]에서 보이는 사진은 인공지능이 인식할 대상으로, 입력 데이터(Input Data)라고 합니다.

라벨(Label)은 인공지능이 입력 데이터를 제공받으면 추론해야 할 결과로, 분류 과제의 경우 각 카테고리당 O/X 또는 1/0으로 구별합니다.

고양이 종류 구분 과제의 경우 라벨을 길이 4만큼의 벡터로 만들 수 있습니다.

입력 데이터만으로 학습할 수 있는 모델의 종류는 많지 않기 때문에, 정답 라벨이 제대로 달려 있는 데이터를 얼마나 모을 수 있느냐가 모델 학습의 품질을 좌우합니다.

하지만 정답 라벨링은 사람이 일일히 만들어줘야 해서 공수가 많이 드는 작업이지요.

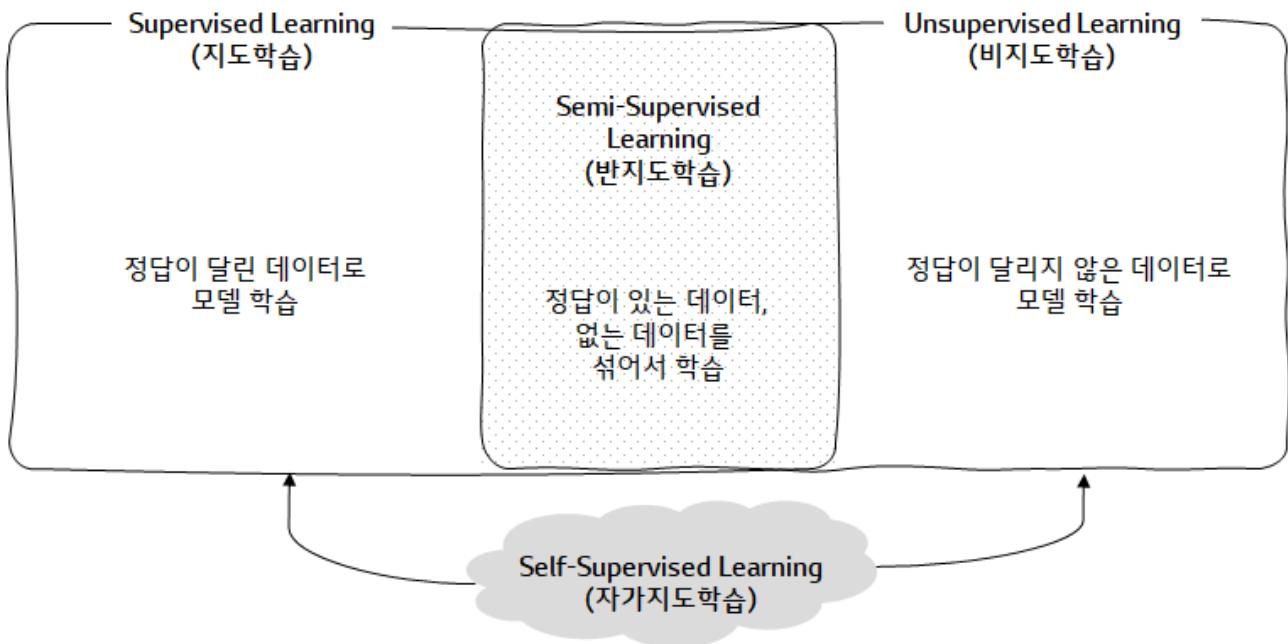
어려운 일은 아니지만 인형의 눈알을 하나씩 붙이고, 피자박스를 접고, 마늘 껍질을 하나씩 까고 다듬는 것처럼 귀찮고 시간이 많이 드는 단순반복 노동입니다.



[그림 8. 중국에서는 마늘까기를 교도소 수감자에게 시킨다고 합니다. 사소하지만 귀찮고 양 많은, 누군가는 해야하는 일이지요... (자료: 넷플릭스-Rotten<sup>50</sup>)]

라벨을 붙인 데이터를 많이 만들기는 힘들어도 라벨 없는 데이터를 모으는 것은 어렵지 않습니다.  
수많은 이미지와 동영상, 텍스트 문서 자체는 하루에도 셀 수 없는 양이 쏟아지고 있으니까요.  
일단 데이터가 많기는 한데 인공지능 모델에게 알려줄 정답은 없고, 어떻게든 뭔가 활용할 방법이 없을까요?  
이 때 활용할 수 있는 학습 방법이 **Self-Supervised Learning(자가지도학습)**입니다.

<sup>50</sup> [https://en.wikipedia.org/wiki/Rotten\\_\(TV\\_series\)](https://en.wikipedia.org/wiki/Rotten_(TV_series))



[그림 9. 인공지능 모델 학습 방법의 종류]

Self-Supervised Learning이라는 용어를 처음 듣는다면 마치 인공지능이 스스로 학습하여 똑똑해지는 것처럼 느껴집니다만, 그런 거창한 개념은 아닙니다.

Self-Supervised Learning은 사람이 만들어주는 정답 라벨이 없어도 기계가 시스템적으로 자체 라벨을 만들어서 사용하는 학습 방법입니다.

사람이 라벨을 만들어줄 필요가 없다는 점에서는 Unsupervised Learning으로 볼 수 있지만, 자체적으로 라벨을 만들어 사용한다는 점에서 Supervised Learning의 일종으로 볼 수도 있습니다.

예를 들어 보면 더 잘 이해할 수 있습니다. 다음은 Self-Supervised Learning의 경우입니다.

### 8.3.1 예: 이미지 데이터를 위한 Self-Supervised Learning



vs

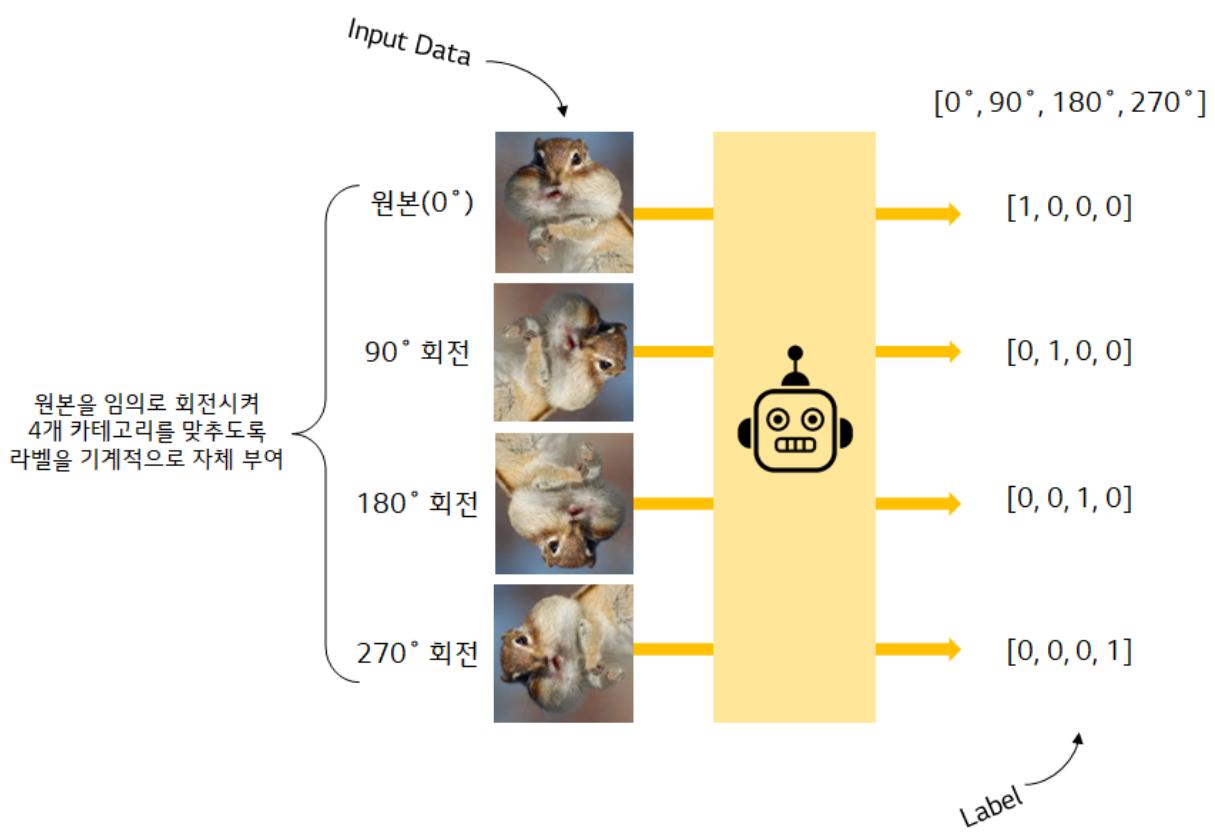


다람쥐와 청설모를 구별하고 싶은데, 우선 종류에 상관 없이 설치류 짐승의 사진을 10만 장 정도 충분히 많이 확보해두었다.

하지만 10만 장의 설치류 사진이 각각 어떤 종류에 해당하는지 라벨을 부여하기는 시간과 비용을 확보하기 어려웠다.

데이터는 많고... 설치류의 일반적인 특징에 대해 뭐라도 학습한 딥러닝 모델을 사전학습하려고 한다.

이 때 **Self-Supervised Learning**을 활용하여 자동으로 라벨을 부여하고 맞출 수 있는 태스크를 만들어 모델을 사전학습시키기로 하였다.



이렇게 사전학습한 모델을 다람쥐/청설모 데이터로 **Transfer Learning** 하였더니 다람쥐/청설모만으로 학습한 모델보다 좋은 성능을 얻을 수 있었다.

### 8.3.2 예: 텍스트 데이터를 위한 Self-Supervised Learning

사외로 전송되는 이메일의 보안 위반 여부를 검출하고자 하는데, 우선 보안 위반 여부와 관계없이 사외전송 이메일 10만 건을 모아두었다.

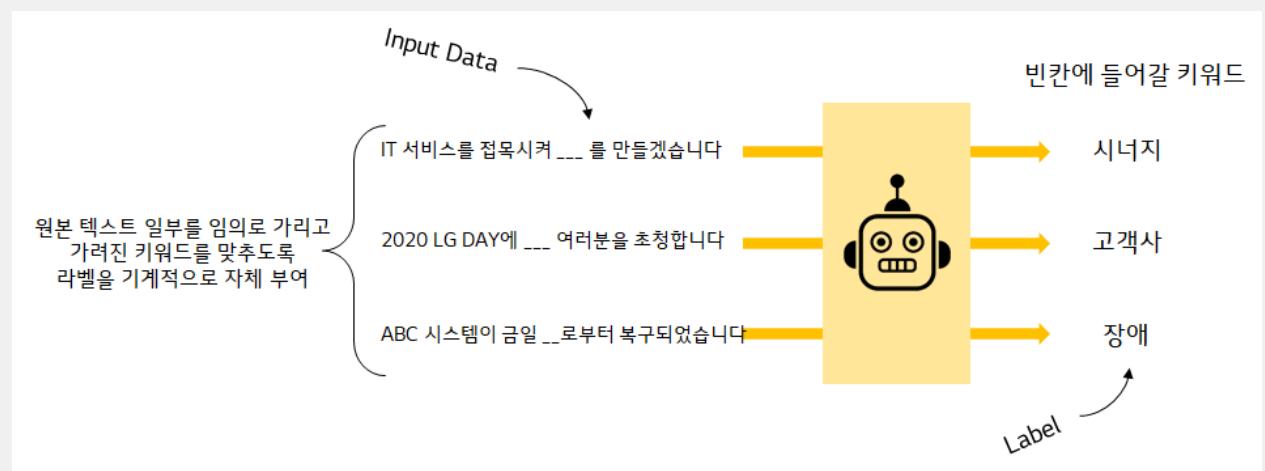
하지만 10만 건의 이메일을 전부 살펴보기 힘들어 1만 건의 이메일에 대해서만 라벨링을 할 수 있었다.



가진 데이터를 전부 활용하여 조금이라도 업무 관련 도메인 키워드를 학습할 수 있도록 **Self-Supervised Learning**으로 사전학습을 시키고자 한다.

사전학습 태스크로는 메일의 중간 단어를 빈칸으로 대체한 후 들어갈 단어를 알아맞추도록 하였다.

이렇게 하니 10만건의 메일을 전부 활용하여 인공지능 모델에게 우리 회사에서 자주 쓰는 키워드를 인식시킬 수 있었다.



이렇게 만든 모델을 **Transfer Learning**으로 활용하니 1만 건의 라벨링 데이터만으로 학습한 모델보다 좋은 성능을 보였다.

이처럼 Self-Supervised Learning은 주로 사전학습에서 이용되며 다양한 데이터는 있으나 라벨은 없는 경우에 활용할 수 있습니다.

Self-Supervised Learning의 과제 자체가 의미 있는 것은 아니나, 수많은 데이터를 자체 라벨링으로 학습하게 되면 해당 데이터에 대한 전반적인 지식을 넓고 얕게 습득할 수 있게 되는 것이지요.

이렇게 학습한 모델을 향후 후속 과제로 Transfer Learning하면 맨 땅에서 학습한 모델에 비해 일반적으로 좋은 성능을 보입니다.

### 8.3.3 예: Google BERT(Bidirectional Encoder Representations from Transformers)

Self-Supervised Learning 기법으로 사전학습을 하고 다양한 태스크에 Transfer Learning을 할 수 있는 대표적인 예로 Google의 '**BERT**'라는 모델이 있습니다.

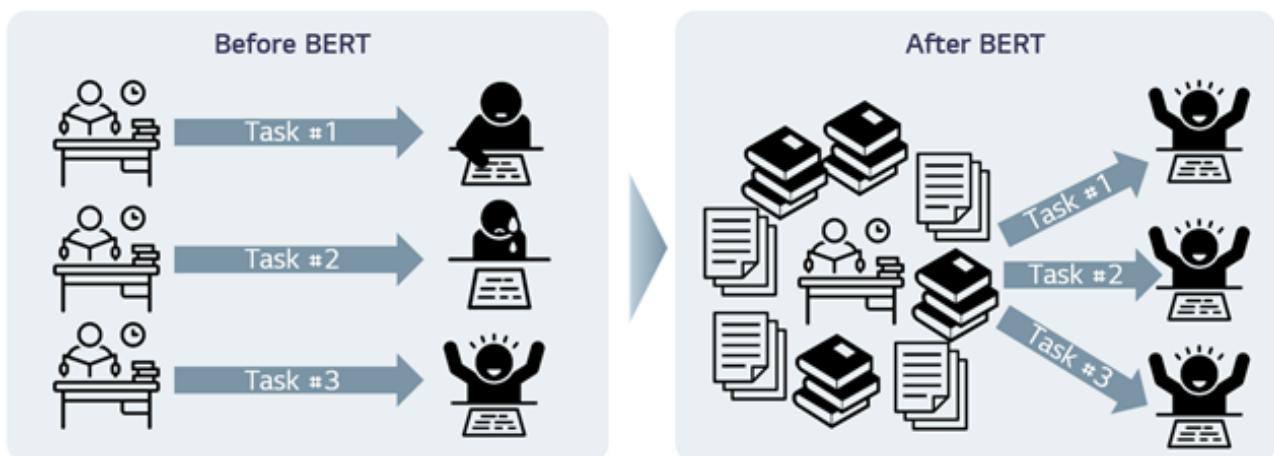
BERT는 자연어처리 연구 패러다임을 전환한 계기가 된 모델입니다. 인공지능에서의 자연어 처리는 BERT 이전과 이후로 나뉘죠.

예전부터 사전학습과 Transfer Learning의 개념이 있긴 했으나, 기존의 사전학습이 워드 임베딩 등 그저 보조적 역할을 수행하는 느낌이었다면 BERT는 사전학습 자체가 주가 되는 모델입니다.

기존 모델 학습 방식을 시험에 비유해보겠습니다.

고3은 수능을 위해 수능 기출을 따로 풀고, 토익 응시생은 토익 문제만 엄청 풀고, LGCNS 사원은 정보처리기사를 공부해서 각각의 시험에서 성적을 내기 위해 노력했습니다.

하지만 BERT의 관점은 '이거저거 닥치는대로 책을 많이 본 사람'이 나중에 '어떤 시험을 쳐도 잘 보게 된다'는 것입니다.



[그림 10. (왼)BERT 이전의 자연어처리 모델 학습방식, (오)BERT의 학습 방식]

즉 '언어'라는 분야 전반에 걸쳐 지식을 두루 쌓은 '하나의 거대한 뇌'를 사전학습으로 만든다는 개념입니다.

BERT는 사전학습에서 상당한 양의 데이터(텍스트 코퍼스)를 커다란 모델로 학습시켰으며, 후속 태스크를 위한 Transfer Learning은 간략하게만 진행해도 좋은 성능을 낼 수 있었습니다.

무려 11개의 자연어처리 과제에서 1위를 했는데, 이는 텍스트를 대상으로 할 수 있는 거의 대부분의 과제라고 볼 수 있습니다.

이 때 BERT가 사전학습한 문서가 무려 33억 단어만큼이며, 16개의 TPUv3 칩을 활용하여 학습하였다고 합니다.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>

[그림 11. BERT 모델 공개 당시 모든 과제에서 성능 1위를 기록하였습니다.]

어찌 생각해보면 당연한 결과로 보입니다.

전 과목을 다 못하는 꼴지인데, 수학 한 과목에서만 천재인 학생이 있을까요?

모두의 학창시절이 비슷했겠지만 모든 과목을 포기하지 않고 두루 공부하는 학생이 각 과목의 상위권 또한 차지하곤 합니다.

지식이라는 것은 서로 연결되는 부분이 있어서 한 부분에서 습득했던 내용이 전혀 예기치 못한 다른 영역을 배우는데 도움을 줄 수 있기 때문이지요.

일반적으로 Self-Supervised Learning을 활용한 Pre-Trained 모델은 다양한 지식을 골고루 습득하는 것을 목적으로 하기에

대체로 모델의 사이즈가 큰 편이고 사전학습 규모가 어마어마하다는 특징이 있습니다.

어떻게 보면 GPU 학습 장비나 데이터 저장공간에 대한 비용 부담이 커서 실용적이지 않게 느껴지기도 합니다.

하지만 사전학습 모델은 한 번 잘 마련해놓으면 향후 어떤 과제든 적용할 수 있습니다.

장기적으로 볼 때 두고두고 여러군데 활용 가능한 Base 모델을 준비한다는 개념으로 생각해야 합니다.

## 8.4 마무리

딥러닝 기반의 AI 모델은 다양한 양질 데이터를 필요로 합니다.

또한 대부분은 모델의 학습을 위해 사람이 태깅한 정답 라벨을 필요로 합니다. (Human-labeled data)

데이터 자체를 많이 확보하기는 쉬울 지 몰라도, 라벨링 잘 된 데이터를 다양으로 구하는 것은 쉬운 일이 아닙니다.

이 때 활용할 수 있는 방법이 시스템적으로 라벨을 보유하고 학습할 수 있는 자가지도학습, Self-Supervised Learning입니다.

이 방식으로 모델은 전반적인 지식을 골고루 사전학습(Pre-Training)할 수 있으며, 향후 특정 태스크로 Transfer Learning 할 경우 좋은 성능을 보일 수 있습니다.

이번 시간은 Pre-Training과 Self-Supervised Learning에 대해 설명드렸습니다.

다음 시간에는 사람의 라벨링 작업을 최대한 효율적으로 할 수 있는 방법, '**Active Learning**'에 대해 자세히 알아보겠습니다.

감사합니다 😊

---

## 참고자료

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC), <http://www.image-net.org/challenges/LSVRC/>
- Google Youtube-8M Dataset, <https://research.google.com/youtube8m/index.html>
- WikiPedia database dump, [https://en.wikipedia.org/wiki/Wikipedia:Database\\_download](https://en.wikipedia.org/wiki/Wikipedia:Database_download)
- 나무위키 데이터베이스 덤프, <https://namu.wiki/w/%EB%82%98%EB%AC%B4%EC%9C%84%ED%82%A4:%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4%20%EB%8D%A4%ED%94%84>
- 국립국어원 언어정보나눔터, <https://ithub.korean.go.kr/user/guide/corpus/guide1.do>
- What is the difference between self-supervised and unsupervised learning?, Quora, <https://www.quora.com/What-is-the-difference-between-self-supervised-and-unsupervised-learning>
- Conversational AI, Insight+, 2019, 김명지, [Conversational AI<sup>51</sup>](#)
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, J. Devlin, et al., 2018, <https://arxiv.org/abs/1810.04805>

---

<sup>51</sup> <https://wire.lgcns.com/confluence/display/AI/Conversational+AI>

## 9 [9편] 족집게 데이터로 인공지능 학습하기

---

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/Cognitive AI LAB<sup>52</sup>

---

- 데이터의 바다, 정보의 홍수(see page 135)
  - Active Learning: 족집게 데이터로 공부하기(see page 137)
    - Active Learning의 절차(see page 138)
    - Query Strategy: 이 데이터를 제게 가르쳐 주십시오!(see page 139)
      - Uncertainty Sampling(see page 140)
      - Query by committee(see page 141)
  - 마무리(see page 142)
- 

오늘은 Human Labeling 공수를 최대한 효율적으로 하는 방법, Active Learning에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 9.1 데이터의 바다, 정보의 홍수

지난 시간에는 AI 스스로 라벨을 만들고 사전 학습할 수 있는 Self-supervised Learning에 대해 설명드렸습니다.

데이터 자체는 많지만, 인공지능 모델을 학습시킬 수 있을 만큼 경제되고 라벨이 잘 달려있는 데이터를 구하기는 어렵다는 데에서 착안된 기법인데요,

오늘 설명드릴 Active Learning(능동 학습)이라는 기술도 데이터는 많으나 인공지능을 '학습시킬 데이터'를 마련하기 쉽지 않을 때 이용할 수 있는 기술입니다.

---

<sup>52</sup><https://wire.lgcns.com/confluence/display/~78628>



[그림 1. 양이 많다고 다 좋은 데이터는 아닙니다]

강아지 이미지와 고양이 이미지를 구별하는 태스크라면 누구나 데이터를 라벨링 할 수 있어 적은 비용으로도 금방 데이터를 모으겠지만,

뇌 MRI 영상으로부터 파킨슨 병 여부를 판단하는 태스크라면 해당 분야의 전문 의사가 아니라면 몹시 어려운 일이지요.

게다가 이미지를 한장씩 보면서 파킨슨 병 여부를 일일이 태깅할만큼 시간 여유가 있는 뇌 전문의를 구한다는 것 또한 쉬운 일은 아닙니다.

하지만 뇌 전문의 다섯 명이 한 달 동안 매일 20장씩만 이미지를 보고 라벨링을 해줄 수 있다면 어떨까요?

수많은 뇌 MRI 영상 중 가능한 한 파킨슨 병인지 아닌지 가려내는데 효과적인 데이터만 뽑아서 정답을 알려달라고 하고 싶지 않나요?

**Active Learning**은 라벨링을 할 수 있는 인적 자원은 있지만, 많은 수의 라벨링을 수행할 수 없을 때 효과적으로 라벨링을 하기 위한 기법입니다.

수행하고자 하는 태스크가 너무나 특수하여 해당 도메인의 전문 인력만이 데이터를 라벨링 할 수밖에 없는 경우, 최대한 학습에 효과적인 데이터만을 뽑아내는 데에 쓰일 수 있습니다.

뇌 MRI 영상으로부터 파킨슨 병이라고 판단할 수 있을만한 대표적인 특징이 있겠지요.

하지만 누구나 파악할 수 있는 대표적인 특징을 가진 데이터 말고, 너무 애매모호해서 전문의가 아니고서는 판단하기 어려운 데이터를 확보할 수 있다면 인공지능 학습에 도움이 되지 않을까요?

## 9.2 Active Learning: 족집게 데이터로 공부하기

수능일이 얼마 남지 않았습니다.

수능일까지 남은 시간 동안 각 수험생이 풀어볼 수 있는 문제의 수는 한정되어있습니다.

고3 수험생 철수와 영희가 있다고 생각해봅시다.

철수와 영희는 각각 동일한 문제집 10권을 가지고 있습니다. 문제집 1권 당 문제가 100개씩 있습니다.

철수와 영희는 이 중 총 500 문제를 풀고 난 후 수능을 보게 됩니다.

철수의 학습 계획	철수는 문제집 5권을 임의로 뽑아서 안에 있는 문제를 전부 풀었습니다.
영희의 학습 계획	<p>영희는 문제집 3권을 임의로 뽑아서 안에 있는 문제를 전부 풀었습니다.</p> <p>영희는 풀었던 문제집 3권에서 많이 틀리는 유형들을 체크하였습니다.</p> <p>그 뒤 남은 7권의 문제집에서 많이 틀리는 유형에 해당하는 200문제를 추가로 찾아 풀었습니다.</p>

철수와 영희의 학습 방법이 위와 같을 때, 누가 더 효과적으로 학습했을까요?

일반적으로 본다면 영희가 더 효과적으로 학습했다고 말할 수 있습니다.

풀 수 있는 문제가 500개로 제한된 상황이라면 많이 틀리는 유형들의 문제를 중점적으로 공부해 틀리지 않도록 대비하는 것이 효과적인 방법이기 때문입니다.

철수의 방법이 AI 모델 학습을 위한 일반적인 데이터 라벨링 방식이라면 영희의 학습 계획이 Active Learning이라고 말할 수 있습니다.

위의 예시와 같이 풀 수 있는 문제의 수가 제한된 것처럼, 라벨링 할 수 있는 데이터의 수가 제한된 상황에서는 성능 향상에 효과적인 데이터를 선별하는 과정이 중요합니다.

**Motive**

모델이 잘 맞추기 어려운 데이터를 찾아 학습한다면,  
더 적은 훈련시간으로 더 좋은 성능을 낼 수 있을 것이다.

**Objective**

Labeling을 위한 예산이 한정되었을 때,  
모델의 성능을 극대화할 수 있는 Labeling 대상 데이터를 찾기

[그림 2. Active Learning의 동기와 목적]

Active Learning은 학습 데이터 중 모델 성능 향상에 효과적인 데이터들을 선별한 후, 선별한 데이터를 활용해 학습을 진행하는 방법입니다.

학습 데이터를 확보하는 과정은 데이터를 수집하는 것과 수집한 데이터에 라벨을 태깅하는 라벨링 작업으로 구성되어 있습니다.

일반적으로 라벨링 작업에 많은 시간과 인적 자원 활용 비용이 소요됩니다. 라벨링 작업에 특정 도메인의 전문성이 요구된다면 더더욱 많은 비용을 필요로 하겠지요.

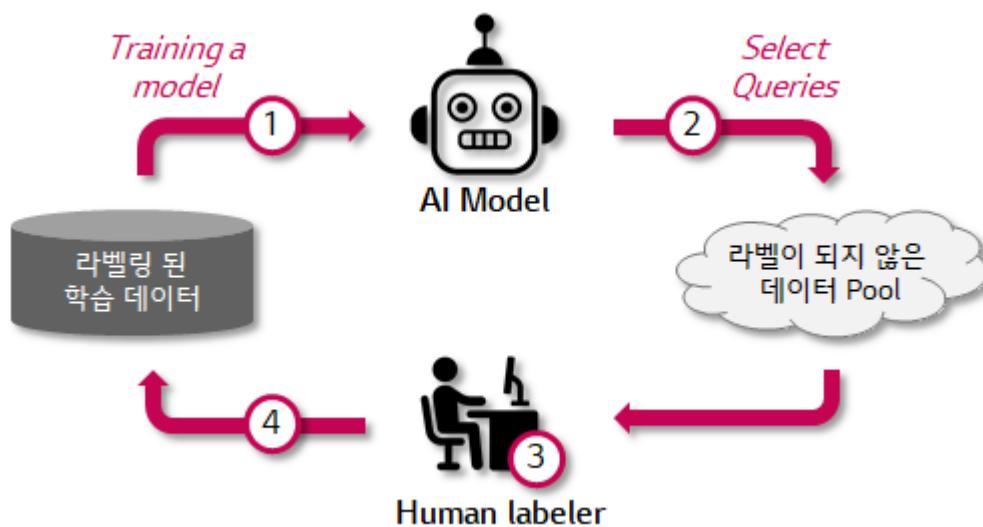
그렇기에 같은 수의 데이터에 라벨을 붙여서 학습할 때, 성능이 높게 나올 수 있도록 데이터를 선별한다면 효과적으로 딥러닝 모델을 학습할 수 있습니다.

이렇게 효과적인 데이터를 선별하는 방법을 연구하는 것이 Active Learning입니다.

이와 반대로 주어진 라벨 데이터만 가지고 모델을 학습하는 방법을 Passive Learning(수동 학습)이라고 합니다.

### 9.2.1 Active Learning의 절차

Active Learning은 크게 4단계로 구성되어 있습니다.



[그림 3. Active Learning의 4단계]

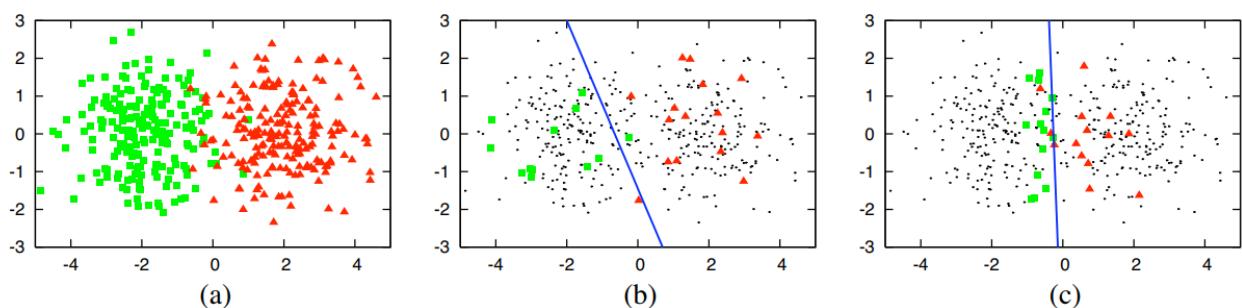
1. **Training a Model** : 초기 학습 데이터(labeled data)를 이용해 모델을 학습합니다.
2. **Select Query** : 라벨이 되지 않은 데이터 풀로부터 모델에게 도움이 되는 데이터를 선별합니다.
3. **Human Labeling** : 선별한 데이터를 사람이 확인하여 라벨을 태깅합니다.
4. 선별한 라벨 데이터를 기존 학습 데이터와 병합한 후, 다시 모델을 학습합니다.

목표하는 성능이 나올 때까지 위의 방법을 반복해 수행합니다.

### 9.2.2 Query Strategy: 이 데이터를 제게 가르쳐 주십시오!

Active Learning의 핵심은 성능 향상에 효과적인 데이터를 선별하는 방법입니다.

이러한 데이터 선별 방법을 ‘쿼리 전략(Query Strategy)’이라고 합니다.

[그림 4. 모델 학습에 효과적인 데이터 선별하기. 자료<sup>53]</sup>]

53 <http://burrsettles.com/pub/settles.activelearning.pdf>

위 [그림 4]를 예로 들어보겠습니다. (a)는 2차원 평면에 나타낸 두 집단의 분포입니다.

우리는 초록색 네모 집단과, 붉은 세모 집단을 구별하는 모형을 만들고자 합니다. 하지만 우리는 이러한 실제 모분포를 알 수 없습니다. (이해를 돋기 위해 그림에는 표시했지만요!)

우리는 두 집단이 있다는 것만 알고, 일부 데이터를 샘플링해와서 이 데이터가 초록 네모인지, 붉은 세모인지 라벨링하고 해당 데이터만으로 두 집단을 구별하는 모델을 만들고자 합니다.

(b)는 마구잡이로 데이터를 랜덤하게 샘플링해와서 초록 네모인지 붉은 세모인지 라벨링한 것입니다. 검은 점들은 선택되지 않은, Unlabeled data이기 때문에 어떤 집단인지 알 수 없습니다.

우리는 알고 있는 초록 네모와 붉은 세모 샘플만으로 모델을 학습하게 됩니다. 그 결과 두 집단을 전반적으로 나눌 수 있는 선 하나를 (b)처럼 그리게 됩니다.

하지만 (b)의 선은 실제 모집단의 분포를 70%만 제대로 분류할 수 있습니다.

(c)의 경우도 마찬가지로 일부 데이터만 샘플링해와서 라벨링하고 분류 모델을 만들게 되는데, 이 경우엔 샘플을 마구잡이로 고르는 것이 아니고 일정 기준에 따라 샘플링하게 됩니다.

이 때의 기준은 '어떤 집단에 속하는지 애매하고 헷갈리는 데이터'인데, (c) 그림상에서 볼 때 집단간 경계에 있는 모호한 샘플 위주로 추출된 것을 볼 수 있습니다.

모델이 헷갈릴만한 데이터 위주로 추출하여 정답을 알려주고 학습한다면 더 정교한 분류 모델이 만들어 질 수 있겠지요?

(c)의 모델은 모집단의 분포를 90% 제대로 분류할 수 있도록 학습할 수 있습니다.

쿼리 전략을 어떻게 정하느냐에 따라서 선별할 데이터가 달라집니다.

- 학습된 모델의 판정 값을 기반으로 뽑는 Uncertainty Sampling
- 여러 개의 모델을 동시에 학습시키면서 많은 모델이 틀리는 데이터를 선별하는 Query by committee
- 데이터가 학습 데이터로 추가될 때, 학습된 모델이 가장 많이 변화하는 데이터를 선별하는 Expected Impact
- 데이터가 밀집된 지역의 데이터들을 선별하는 Density weighted method
- 데이터들을 최대한 고르게 뽑아서 전체 분포를 대표할 수 있도록 데이터를 선별하는 Core-set approach

이 중 대표적인 두 가지에 대해 간단히 설명드리겠습니다.

### 9.2.2.1 Uncertainty Sampling

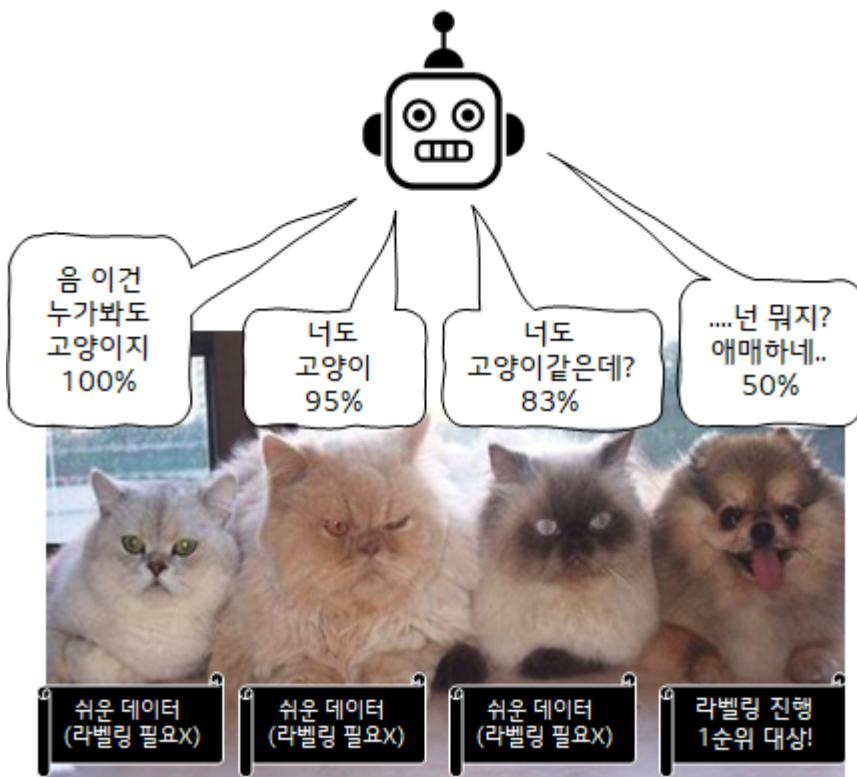
Uncertainty Sampling은 가장 단순한 쿼리 전략입니다.

AI 모델은 가장 불확실하다(least certain)고 생각하는 데이터를 추출하여 라벨링이 필요하다고 요청하게 됩니다.

예를 들어 강아지와 고양이를 분류하는 이진분류(binary classification) 태스크의 경우,

모델이 어떤 데이터에 대해 강아지일 확률과 고양이일 확률을 각각 50% 내외로 추론한다면 해당 데이터는 강아지인지 고양이인지 애매한 데이터겠죠?

이런 데이터를 라벨링하여 모델에게 알려준다면 분류 성능을 높이는 데 도움이 될 겁니다.



[그림 5. 애매한 데이터는 모델이 분류에 대한 확신이 낮을 것이다(불확실한 추론)]

### 9.2.2.2 Query by committee

Query by committee는 여러 AI 모델간의 의견불일치를 종합 고려하는 방식입니다.

여러 모델간 추론한 결과 불일치가 많은 데이터일수록 가장 헷갈리는 데이터, 즉 라벨링을 진행할 대상이 되는 것이죠.



[그림 6. 왼:확실한 데이터, 오:애매한 데이터]

강아지과 고양이를 분류하는 모델을 여러 개 학습했다고 해봅시다.

어떤 데이터를 넣었을 때 학습한 모델간 추론 의견이 일치한다면 그 데이터는 확실히 강아지거나 고양이인 데이터일 것입니다.

그만큼 정보는 부족하겠지요. 이런 데이터는 라벨링을 진행하지 않아도 이미 여러 모델이 잘 맞춘다는 뜻이므로 넘어가도 좋습니다.

하지만 어떤 데이터를 넣었을 때 모델간 추론 결과가 제각각이라면 그 데이터는 고양이인지 강아지인지 애매한, 정 보가 많은 데이터입니다.

이러한 경우 라벨링을 진행하여 모델 학습에 이용하면 분류 성능 향상에 도움이 될 것입니다.

이외에 다양한 쿼리 전략이 있습니다만, 어떤 방법이든 간에 가장 정보가 많은 데이터를 선정해서 라벨링해야 모델 학습에 도움이 될거라는 생각은 동일합니다.

예시를 통해 가장 단순한 두 방법에 대해 이해하셨다면 쿼리 전략이라는 것이 어떤것인지 이제는 아셨겠죠?

### 9.3 마무리

데이터 자체는 손쉽게 대량으로 확보할 수 있으나 모델이 학습에 사용할 수 있는 '유의미한 라벨 정보가 포함된 데이터'는 극소수입니다.

데이터는 많이 있지만, 역설적이게도 AI 모델 학습을 위해 '쓸모있는 데이터'는 많지 않습니다.

풀고자 하는 태스크를 위한 라벨 정보를 새로 만드는 것은 시간 및 비용에 의해 현실적으로 불가능한 경우가 대부분입니다.

이러한 어려움을 조금이라도 해결하기 위해 연구되어온 방법이 Active Learning입니다..

이번 주 내용을 읽어보고 느끼셨겠지만, Active Learning은 그 자체로 어떤 딥러닝 기술이라기보다는 효과적인 학습을 위한 시스템이라고 보는 것이 맞습니다.

그리고 그 시스템의 일부분에서 반드시 인간의 라벨링 작업을 필요로 하지요.

AI는 어떤 식으로든 이와 같이 조금이라도 사람의 도움(라벨링과 같은)을 필요로 합니다.

여러 사례들을 보면 Active Learning은 AI 모델 학습을 시작하는 초기 개발 단계에 매우 효과적입니다.

어차피 가르쳐줘야 할 것이라면 조금이라도 효율적으로, 더 도움이 될 수 있는 방향으로 작업을 할 수 있는 것이 좋겠죠?

그렇다 해도 Active Learning이라는 방법 하나로는 데이터에 관한 모든 문제를 해결할 수는 없습니다.

필요한 라벨 데이터의 개수가 줄어들 수는 있지만, 결국은 사람이 직접 라벨링을 수행해야 하기 때문입니다.

하지만 Active Learning은 인공지능 기술이 효율적으로 접목될 새로운 가능성을 열어줄 수는 있습니다

이번 시간은 Active Learning에 대해 설명드렸습니다.

다음 시간에는 어텐션 메커니즘(Attention mechanism)에 대해 소개하겠습니다.

감사합니다 😊

---

## 참고자료

- 족집게 데이터가 '전교 1등' AI 만든다!, 김도연 연구원(AI기술팀), LG CNS 블로그, <https://blog.lgcns.com/2275?category=854507>
- Introduction to Active Learning, Jennifer Prendki, KDnuggets, <https://www.kdnuggets.com/2018/10/introduction-active-learning.html>
- Active Learning tutorial, Ori Cohen, towards data science, <https://towardsdatascience.com/active-learning-tutorial-57c3398e34d>
- Active Learning Literature Survey, Burr Settles, 2010, University of Wisconsin–Madison, <http://bursettles.com/pub/settles.activelearning.pdf>

## 10 [10편] 뭣이 중한지 알아보는 인공지능

---

안녕하세요, CTO AI빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI빅데이터연구소 AI기술팀 김명지 책임연구원/Cognitive AI LAB<sup>54</sup>

---

- 긴 입력 데이터 처리하기(see page 144)
  - 어텐션 메커니즘(Attention mechanism)(see page 146)
    - 어텐션 스코어(Attention score)(see page 148)
    - 컨텍스트 벡터(Context vector)(see page 148)
  - XAI로서의 어텐션(see page 151)
    - 텍스트에서의 어텐션(see page 153)
    - 이미지에서의 어텐션(see page 153)
  - Attention 전성시대, Transformer(see page 154)
  - 마무리(see page 156)
- 

오늘은 인공신경망 모델이 특정 영역에 더 집중해서 의사결정하는 방법인 어텐션 메커니즘(Attention mechanism)에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 10.1 긴 입력 데이터 처리하기

지난시간에 배웠던 RNN에 대해 기억하시나요? RNN은 시간 순서에 따른 데이터를 처리하는 인공신경망이었습니다.([참고\(see page 79\)](#))

RNN의 한계는 데이터가 길어질 경우, 즉 긴 시간에 대해 누적된 데이터가 입력될 경우 먼 과거의 내용을 잘 반영하기 어렵다는 점이었는데요,

이는 사람도 마찬가지로, 사람도 최근의 내용만 기억에 남기고 먼 옛날의 사건일수록 기억이 가물가물하다는 점을 설명드렸습니다.

연말 평가입력을 위해 수행한 내용들을 작성할 때, 최근에 한 일에 대해서는 구체적으로 작성할 수 있지만 1,2월에 했던 내용은 어떤가요? 바로 기억이 술술 떠오르는 분은 잘 없으시지요?

그 때의 업무 일지나, 메일이나 플래너, 주간보고등을 다시 찾아보며 어떤 일이 있었는지 되짚어보며 평가 입력에 참고하게 되지요.

---

<sup>54</sup><https://wire.lgcns.com/confluence/display/~78628>

또 다른 예로, 기계번역 태스크를 생각해보겠습니다.

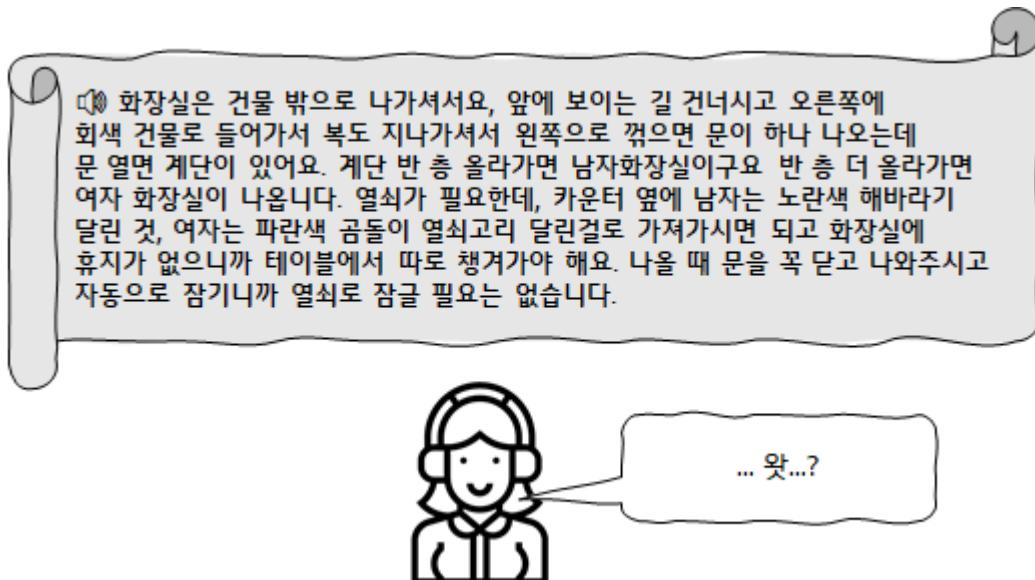
다음 문장이 음성으로 들려온다고 생각하고, 다 듣고 난 뒤 영어로 한번 번역해보시겠어요?



[그림 1. 이 정도 번역은 꼼이죠!]

참쉽죠? 여행 필수 표현으로 많이 공부했던 문장이기도 하고, 짧은 문장이기도 합니다.

그렇다면 다음을 한번 다 듣고 난 뒤 영어로 번역해야 한다고 생각해보겠습니다.



[그림 2. 져라면 이 화장실 안갑니다.]

어떤가요? 다시 들을 기회가 없다고 할 때 한번 듣는 것 만으로 전부 번역할 수 있으신가요?

내용은 어렵지 않습니다. 하지만 다량의 내용이 담긴 긴 문장(또는 문단)을 딱 한 번만 들어서는 번역을 정확하게 할 수 없을 겁니다.

우리가 긴 문장을 번역할 때엔 한 단어 또는 구절을 영어로 옮길 때마다 필요한 내용을 한국어 원문에서 되짚어가며 하겠지요.

한국어와 영어의 어순이 다르니 뒷쪽을 봤다가 다시 앞쪽을 참고하기도 하고, 여러 어절을 한 영어 단어로 번역하거나 반대로 한 어절을 여러 구로 번역하기도 할 겁니다.

문장이 문단이 되고, 문서가 될수록 더욱 더 여러 번 원문을 재참조해야 합니다.

우리의 기억력에는 한계가 있어서, 모든 인풋을 한 번 읽고 아웃풋을 한 번에 생성하는 것보다는 그때 그때 재확인하면서 필요한 부분을 보는 것이 더 도움이 됩니다.

사람이 원문(Input data)를 전체적으로 훑으며 중요한 부분을 다시 참고하듯, 인공신경망 학습 때에도 이러한 모티브를 녹여낼 수 없을까요?

## 10.2 어텐션 메커니즘(Attention mechanism)

팀 주간회의를 할 때, 내 업무 뿐 아니라 다른 팀원들의 모든 내용을 빠짐없이 다 주의깊게 들으시는 분 있으신가요? (※팀장님 제외..)

아마 대부분은 이러면 안된다는 것을 알면서도... 나와 관련된 업무 이야기를 할 때만 반짝 주의하여 듣고 나머지 시간은 노트에 낙서를 하거나 다른 생각을 하고 계시겠지요.

그러다 질문이 들어왔는데 맥락을 이해하지 못하고 어버버 거리면 팀장님이 "집중좀 합시다!!!" 한소리 하실 수도 있겠구요.



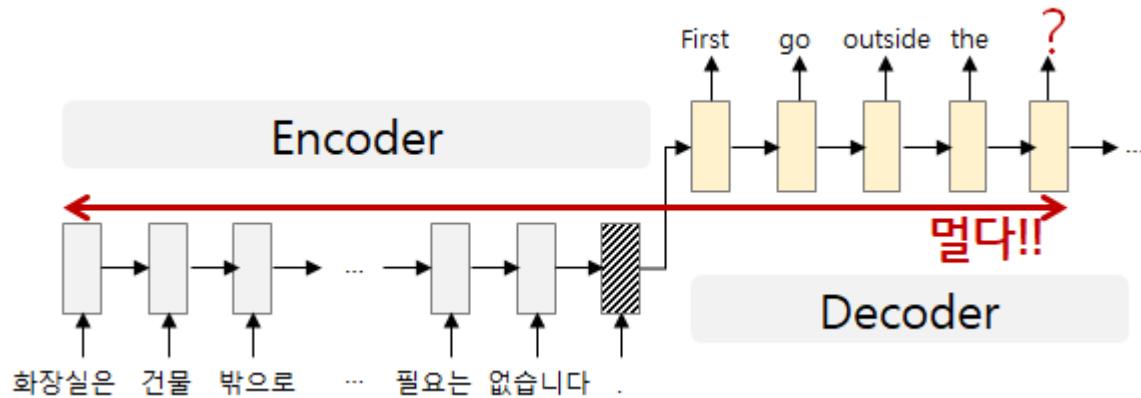
[그림 3. 다양한 과제 이야기가 오가는 팀 주간회의. 떡볶이 맛집 댓글추천받습니다.]

인공신경망 모델에서의 집중(Attention)이란 무엇일까요?

인공신경망이 수행하는 '집중', 어텐션 메커니즘에 대해 알아봅시다.

어텐션 메커니즘이란 인공신경망이 입력 데이터의 전체 또는 일부를 되짚어 살펴보면서 어떤 부분이 의사결정에 중요한지, 중요한 부분에 >>집중<<하는 방식입니다.

일단, 어텐션 없이 기본 RNN만으로 구성한 기계번역 모델은 아래와 같습니다.



[그림 4. 기본 RNN의 인코더/디코더를 활용한 기계번역]

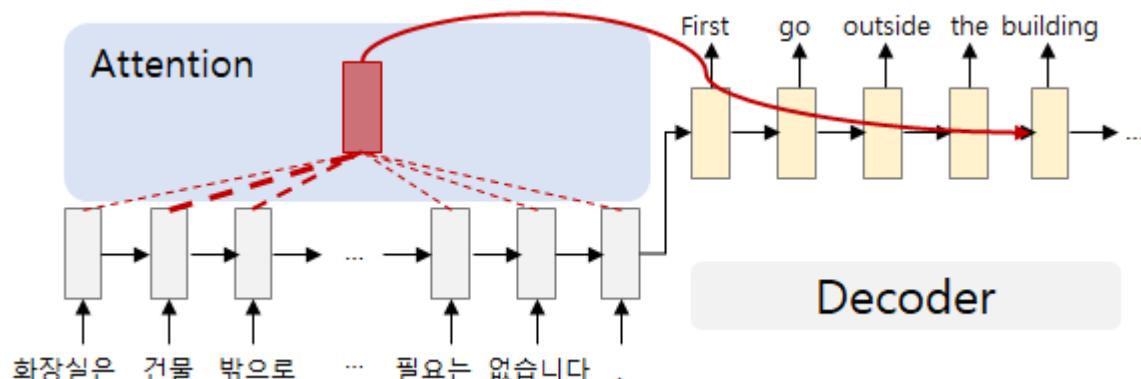
RNN은 입력 문장의 단어 하나 하나를 누적하여 압축해서 인코딩하고 있다가, 모든 문장이 다 들어오게 되면 영어로 한 단어씩 번역을 수행(디코딩)합니다

이 때 디코더가 참고하는 문맥은 그림상의 ■에 해당하는, 입력문이 전부 압축된 하나의 벡터입니다.

이 벡터는 긴 문장을 모두 누적하고 있지만, 문장 앞부분의 내용은 너무 압축된 나머지 정보를 거의 잊어버린 것이나 마찬가지입니다.

여기서 어텐션 메커니즘을 끼얹어, 번역시에 원문을 다시 재참조하며 현재 디코딩할 단어와 연관된 중요 부분에 집중케 하면 어떨까요?

아래와 같이 구성해볼 수 있습니다.



[그림 5. 어텐션 메커니즘을 추가로 적용한 RNN 기계번역]

"building"이란 단어를 생성할 때, 인공신경망은 전체 한국어 입력 문장을 되짚어보며 현재 단어를 디코딩하기 위해 중요한 부분이 어디일까를 생각하게 됩니다.

그 결과 수많은 입력 단어 중 "건물"에 해당하는 단어에 조금 더 주의를 기울일 필요가 있다고 판단,

해당 단어에 조금 더 집중하여 전체 입력을 다시 한번 재조정한 입력 데이터 인코딩 벡터를 만듭니다.

이렇게 하면 입력 문장이 매우 길어진다고 해도 전체 문맥을 골고루 참고할 수 있게 되므로 더 좋은 번역을 할 수 있습니다.

### 10.2.1 어텐션 스코어(Attention score)

이 때, 중요한 단어에 집중한다는 것은 어텐션 스코어를 계산한다는 것인데요,

어텐션 스코어는 인공신경망 모델이 각 인코딩 timestep마다 계산된 특징(feature)를 가지고 자동으로 계산하는 0~1 사이의 값입니다.

어떤 step은 더 집중해서 봐야하고(1에 가까운 스코어), 어떤 스텝은 지금은 중요하지 않으므로 대충대충 살피도록(0에 가까운 스코어) 하는 것이죠.

[그림 5]에서는 붉은 점선의 굵기로 어텐션 스코어를 표시해보았습니다.

각 단어에 대한 주의 집중 가중치라고 볼 수도 있겠네요.

### 10.2.2 컨텍스트 벡터(Context vector)

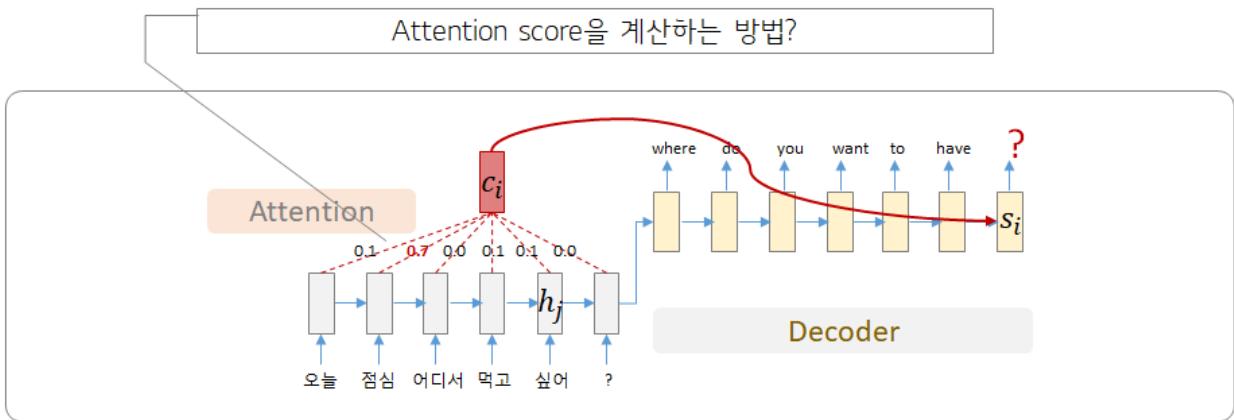
이렇게 어디를 더 살펴보고 어디는 대충 볼지에 대해 어텐션 스코어를 구하고 나면 현재 디코딩할 단어와의 관련성을 반영하여 다시 입력 문장을 인코딩하게 되는데

이는 중요도에 따라 전체 문맥의 정보를 잘 반영하고 있다고 하여 컨텍스트 벡터(Context vector)라고 부릅니다.

어텐션 스코어와 컨텍스트 벡터를 만드는 방식은 여러 가지가 있겠으나, 수학 연산 종류의 차이일 뿐이므로 깊게 다루지는 않겠습니다.

혹시 궁금하시다면 참고 자료를 봐주세요.

**참고 : RNN에서 attention score 및 context vector 계산 예 (자료:딥러닝실무과정)**



- 디코딩하는 i번째 타임스텝 직전의 hidden을  $s_{i-1}$ ,
- Attention 대상 토큰 중 j번째에 대한 hidden을  $h_j$  라고 할 때, i번째 타임스텝의 hidden은 다음과 같이 구한다.

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \quad e_{ij} = a(s_{i-1}, h_j) = \begin{cases} s_{i-1}^\top h_j \\ s_{i-1}^\top W_a h_j \\ v_a^\top \tanh(W_a[s_{i-1}; h_j]) \end{cases}$$

context vector      where  $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$

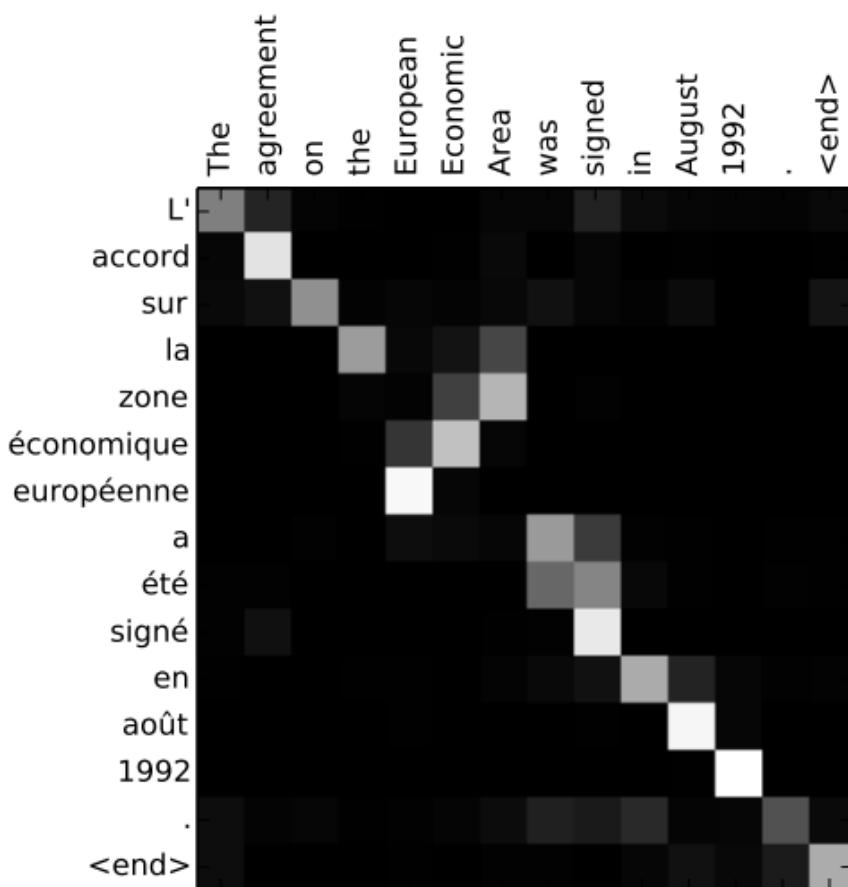
중요한 것은 스코어 계산에 필요한 수식이 아니라, 어텐션 메커니즘이 매번 디코딩마다 직전 단계의 벡터 뿐 아니라 과거의 모든 데이터의 특징(feature)들을 고려한다는 점입니다.

그리고 또 하나의 포인트는, 딥러닝 모델이 스스로 집중할 영역을 파악한다는 것이지요.

데이터를 더 잘 맞추도록 학습하는 과정에서 어떤 부분이 중요한지 사람이 알려주지 않아도 딥러닝 모델은 알아서 집중할 영역을 찾아냅니다.

딥러닝을 활용한 기계번역을 위해 어텐션 메커니즘을 처음 도입한 논문([참고<sup>55</sup>](#))에 보면 이런 자료가 있습니다.

<sup>55</sup> <https://arxiv.org/pdf/1409.0473.pdf>



[그림 6. English-French 번역에서의 attention]

위 그림은 영어 문장을 프랑스어로 번역하는 과제를 할 때, 각 단어 번역시 인공신경망이 어떤 영어 단어쪽에 집중했는지 어텐션 스코어를 시각화한 그림입니다.

하얀색으로 표시될수록 딥러닝 모델이 해당 단어를 더 주의깊게 봤다는 뜻이 되고, 이는 사람이 알려준 것이 아닌 신경망 스스로가 학습한 집중 패턴입니다.

저는 프랑스어를 잘 모르지만... 그림을 보면 영어와 프랑스어는 대강 비슷한 어순을 가지는 것으로 보이는군요!

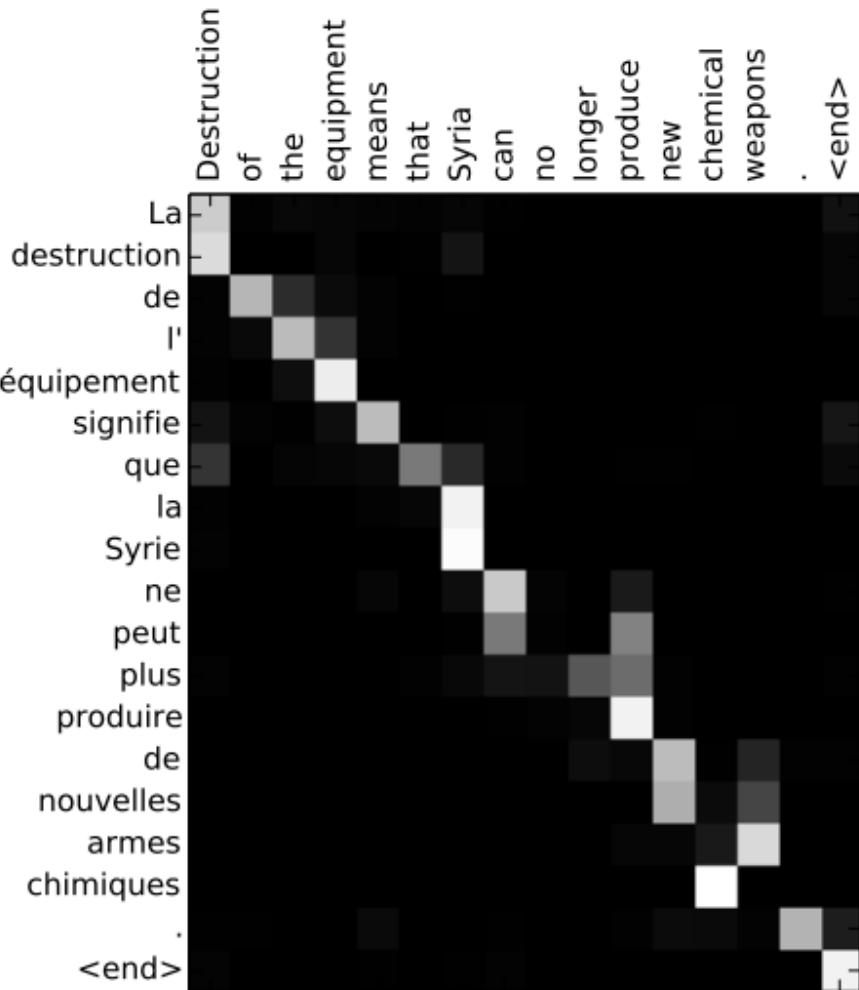
특이한점은 영어의 "European Economic Area"가 프랑스어의 "zone économique européenne"로 번역될 수 있는데, 이 부분은 영어와 프랑스어 어순이 반대입니다.

신기하게도 어텐션 스코어를 보면 딥러닝 모델도 이 부분에서는 순서를 반전시켜가며 주의를 기울이고 있네요.

완전히 동일한 어순을 가지며 단어간 1:1 매칭이 되는 언어끼리의 번역이 아니라면 그 때 그 때 유연하게 집중해야만 하는데

이 경우 어텐션 메커니즘이 그 역할을 훌륭히 해냈습니다.

하나 예를 더 살펴보겠습니다.



[그림 7. English-French 번역에서의 attention]

위 예에서 불어의 "La destruction"란 단어를 번역할 땐 영어의 "Destruction"에 집중하고, "la Syrie"를 번역할 땐 영어의 "Syria"에 집중한 걸 볼 수 있습니다.

프랑스어에서 단어 앞에 관사를 붙이는 점이 영어와 다른 특징이라고 볼 때, 해당 언어별 특징을 잘 이해하고 집중할 부분을 선택한 것을 확인할 수 있네요.

### 10.3 XAI로서의 어텐션

예제에서 알 수 있듯이 어텐션 메커니즘은 기계가 판단시 중요하게 생각하는 부분을 우리에게 알려주는 역할도 합니다.

설명가능한 인공지능(explainable AI; XAI)으로서의 기능을 수행하는데요, 이러한 영역만을 따로 연구하는 분야가 있을 정도로 인공지능의 추론 결과를 해석하는 것은 오늘날 중요한 영역입니다.

이를 해석가능한 인공지능(interpretable AI)라고도 부릅니다.

딥러닝 기반의 인공지능은 일반 머신러닝 기반이나 전통적 를 기반의 프로그래밍에 비해 예측 정확도는 좋지만, 그 모델이 너무 복잡하고 해석하기 어렵다는 단점이 있는데요,

이러한 단점은 특히 법률, 의학, 금융 등, 민감한 내용을 다루는 도메인에서 문제가 되곤 합니다.

인공신경망의 무수한 파라미터를 사람이 이해할 수 있는 방법으로 표현하기가 쉽지 않거든요.

예를 들어 미래에 AI 판사가 등장하여 인간 판사를 대체할 수 있을 수준이 되었다고 가정해봅시다.

AI 판사는 아주 실력이 좋은 유능하면서도 24시간 일할 수 있고, 공정하며, 뇌물수수의 유혹에도 흔들리지 않기에 인간은 전적으로 AI의 판단에 결정을 맡기게 되었습니다.

하지만 다짜고짜 AI 판사가 여러분에게 이렇게 선언했다고 해봅시다.

**AI :** "@@씨, 당신은 유죄입니다! 감옥에서 종신형을 선고합니다."

**나 :** "...? 왜... 왜죠? 저는 납득할수가 없어요! 제가 유죄라니요!"

**AI :** "이유는 설명할 수 없습니다. 하지만 제 판단은 정확합니다! 유죄! 굳이 뭐가 필요하시다면 제 모델의 **weight** 와 **bias** 파라미터들을 전부 까보여드리겠습니다!"

**나 :** (질질끌려나가며) "이게뭐죠... 이 몇백만의 숫자들이 무슨의미가있나요...ㅠㅠ"

당연히 납득할 수 없겠죠? AI 판사가 평범한 인공신경망으로 이루어져있다면 죄명이 무엇인지, 뭘 그렇게 얼마나 잘 못했길래 이런 판단을 내렸는지 설명할 수 없습니다.

반면 인간 판사는 이러한 점을 설명해줄 수 있습니다.

왜 이렇게 판단했는지, 죄명은 무엇이고, 어떤 점 때문에 형벌의 강도를 심하게, 또는 약하게 정했는지 등을요.

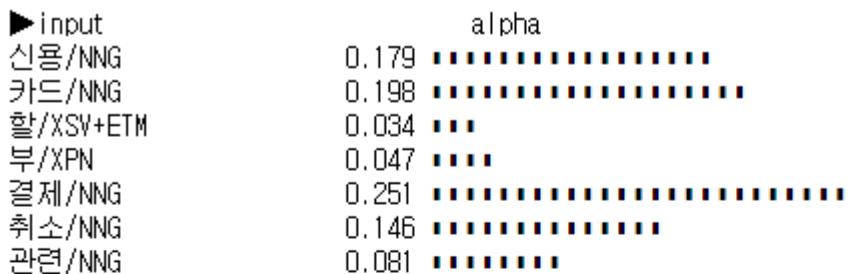
벌을 달게 받는다는게 쉬운 일은 아니지만 그래도 설명이 있다면 납득하고 받아들일 수는 있을 겁니다.

어텐션은 인공신경망의 이러한 설명 부족 문제를 일부 해소해줄 수 있습니다.

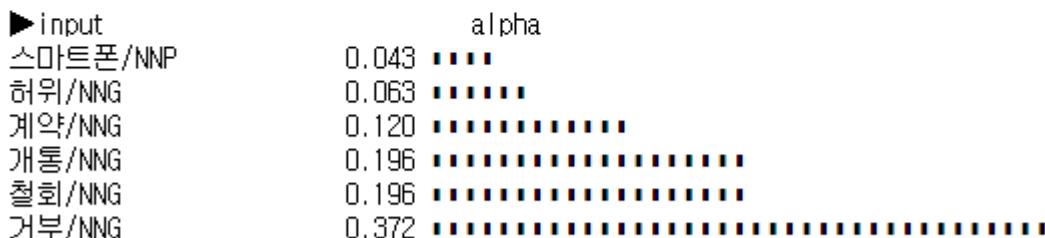
물론 우리가 알아들을 수 있는 말로 "이렇고 저렇게 때문에~ 그렇게 판단했어~"라고 알려주는 것은 아니구요,

해당 결정을 내릴 때 어떤 부분에 집중해서 판단했는지를 시각화해 보여줄 수 있습니다.

### 10.3.1 텍스트에서의 어텐션



- ▶ 모형정답 : (1) 금융
- ▶ 모형정답 : (2) 정보통신서비스
- ▶ 모형정답 : (3) 의류 · 섬유신변용품



- ▶ 모형정답 : (1) 정보통신서비스
- ▶ 모형정답 : (2) 정보통신기기
- ▶ 모형정답 : (3) 도서 · 음반

[그림 8. 소비자 민원 주제 자동분류 예 (AI기술팀, 2017)]

위 그림은 1372 소비자민원센터의 민원 제목을 보고 딥러닝 모델로 민원 카테고리를 자동 분류하는 실험 예입니다.

민원 제목을 형태소분석한 뒤 RNN으로 주제 분류를 한 것인데, 어텐션 메커니즘을 집어넣어 모델이 어떤 키워드를 더 집중해서 보고 주제를 분류했는지 시각화해보았습니다.

첫 번째 민원에서는 '신용', '카드', '결제'와 같은 키워드에 집중해서 '금융' 관련 민원으로 분류를 했군요.

두 번째 민원에서는 '개통', '철회', '거부'에 초점을 맞추어 '정보통신서비스' 관련 민원이라는 것을 파악했네요.

만일 스마트폰 자체에 문제가 있어서 해당 키워드쪽에 더 집중을 했다면 '정보통신서비스' 보다는 '정보통신기기'쪽으로 분류했겠죠?

### 10.3.2 이미지에서의 어텐션

문장을 예로 들어 어텐션을 설명했지만, 입력 데이터의 전반적인 내용을 다시 살피며 중요한 부분에 집중한다는 사고방식은 이미지에도 적용할 수 있습니다.

아래는 Show, attend and Tell<sup>56</sup>이라는 논문에 포함된 자료입니다.



[그림 9. A woman is throwing a frisbee in a park.]

이 논문에서는 이미지 캡셔닝(Image captioning) 과제를 수행하는데, 이미지를 입력으로 주면 딥러닝 모델이 간단한 설명문을 생성하는 과제입니다.

[그림 9]의 첫 번째 사진을 입력으로 주니 인공신경망이 "A woman is throwing a frisbee in a park."이라는 문장을 생성했네요!

뒤의 그림은 모델이 각 단어를 생성할 때 이미지의 어떤 영역을 집중해서 보았는지에 대해 어텐션 스코어를 시각화 한 사진입니다.

'woman'을 생성할 때엔 사진에서 사람이 있는 부분을, 'park'를 생성할 땐 사람보다 주변의 배경에 집중하는 것을 볼 수 있습니다.

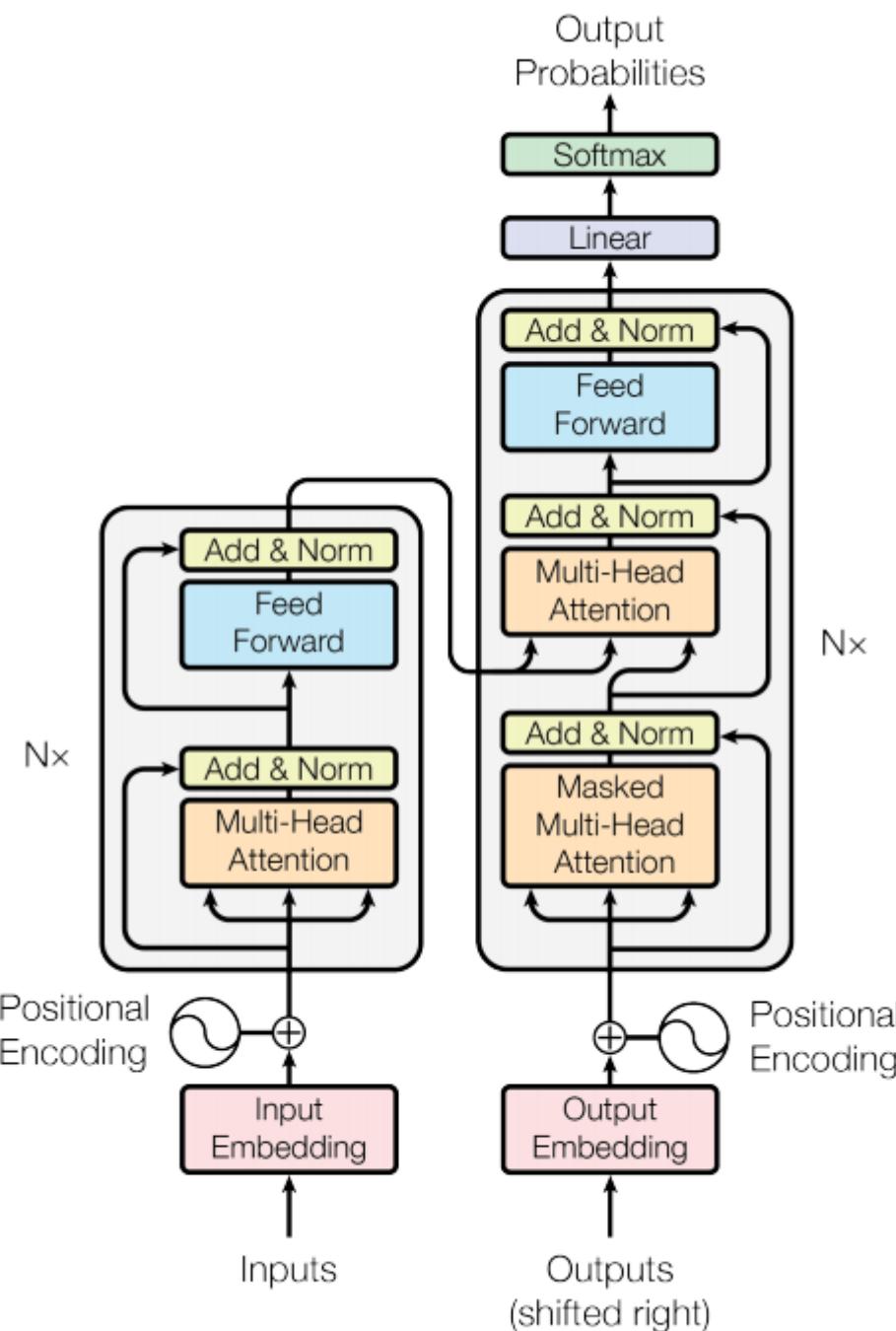
가르쳐주지 않았는데도 스스로 집중할 포인트를 찾아낸다는 게 신기하지 않나요?

## 10.4 Attention 전성시대, Transformer

이 어텐션이라는것이 어찌나 좋은지, 요즘에는 어텐션만으로 이루어진 인공신경망 구조가 새로 등장했습니다.

56 <https://arxiv.org/pdf/1502.03044.pdf>

오죽하면 그 시초가 되는 논문의 제목도 "Attention is all you need"<sup>57</sup>"입니다.



**Figure 1: The Transformer - model architecture.**

[그림 10. 트랜스포머 구조]

57 <https://arxiv.org/abs/1706.03762>

트랜스포머(Transformer)라는 인공신경망은 입력 데이터끼리의 self-attention을 통해 상호 정보교환을 수행하는 것 이 특징입니다.

문장 내의 단어들이 서로 서로 정보를 파악하며, 나와 내 주변 단어간의 관계, 문맥을 더 잘 파악할 수 있게 되는 것이지요.

트랜스포머라는 구조는 오늘날 인공신경망 발전에(특히 자연어 이해에) 큰 획을 긋고 있습니다.

순차적 계산이 필요 없기 때문에 RNN보다 빠르면서도 맥락 파악을 잘하고, CNN처럼 일부씩만을 보는 것이 아니고 전 영역을 아우릅니다.

하지만 이해력이 좋은 대신에 모델의 크기가 엄청 커지며 고사양의 하드웨어 스펙을 요구한다는 단점이 있는데요, 이러한 한계를 보완하기 위한 다양한 경량화 방안이 연구되고 있습니다.

기회가 되면 다른 시간에 다양한 경량화 기법에 대해서도 알아보겠습니다.

## 10.5 마무리

입력 데이터의 크기가 커지면 인공지능은 정보를 효율적으로 처리하기 어렵습니다.

사람도 인공지능만큼은 아니지만, 갑자기 많은 양의 정보를 한번에 받아들이면 인식하기가 어렵죠.

하지만 사람은 데이터를 살펴보며, 어떤 부분이 중요하고 어떤 부분은 지금 당장 보지 않아도 괜찮은지 중요도를 파악하여 필요한 부분에 '집중'할 수 있습니다.

이런 모티브를 인공신경망에 녹여낸 것이 '어텐션 메커니즘'입니다.

이로써 신경망도 의사결정을 할 때 집중할 영역을 찾아 가중치를 더 반영할수 있게 되고, 또 모델의 판단을 사람이 해석하기 쉽게 만들어 줍니다.

이번 시간은 '어텐션 메커니즘(Attention mechanism)'에 대해 설명드렸습니다.

다음 시간에는 'AutoML'에 대해 소개하겠습니다.

감사합니다 😊

## 참고자료

- 자사 딥러닝 실무과정 교재보기, [\[교재보기\] 딥러닝 실무<sup>58</sup>](#)
- Neural Machine Translation By Jointly Learning To Align And Translate, D. Bahdanau, et al., 2015, <https://arxiv.org/abs/1409.0473>
- NLP의 궁예 등장? 관심법으로 번역을 잘해보자, J. Park, <https://jiho-ml.com/weekly-nlp-23/>
- 1372 소비자민원상담 데이터 주제분류, 2017, AI기술팀
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, K. Xu, et al., 2015, <https://arxiv.org/abs/1502.03044>
- Attention Is All You Need, A. Vaswani, et al., 2017, <https://arxiv.org/abs/1706.03762>

<sup>58</sup> <https://wire.lgcns.com/confluence/pages/viewpage.action?pageId=73005264>

## 11 [11편] 스스로 진화하는 인공지능, AutoML

---

안녕하세요, CTO AI 빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI 빅데이터연구소 AI 기술팀 김명지 책임연구원/Cognitive AI LAB<sup>59</sup>

---

- 사람의 손을 필요로 하는 인공지능(see page 157)
- 스스로 진화하는 인공지능, AutoML(see page 159)
  - 하이퍼파라미터 탐색 자동화(see page 160)
  - 아키텍처 탐색 자동화(see page 162)
- AutoML 특징(see page 162)
- AutoML 서비스(see page 164)
- 마무리(see page 166)

오늘은 스스로 진화하는 인공지능, AutoML(Automated Machine Learning)에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 11.1 사람의 손을 필요로 하는 인공지능

AI, 그 중에서도 데이터만 있다면 높은 성능을 보일 수 있는 딥러닝에 대해 어느정도 공부해 봤습니다.

이제 공부한 내용을 바탕으로 실제 AI 모델을 만들기 위해 실험을 반복할 준비가 되었는데요,

AI 사이언티스트의 길을 걷다보면 이 때부터 누구나 튜닝(Tuning)의 장벽과 마주하게 됩니다.

튜닝이라 함은 현재 실험의 결과 양상을 보고 문제점을 진단하고, AI 모델을 조금 더 나은 방향으로 만들고자 실험을 개선하는 것을 말합니다.

여기에는 아키텍처를 변경하거나, 하이퍼파라미터를 조절하거나 하는 등의 역할이 포함됩니다. ([참고:AI 테크레터 6 편\(see page 0\)](#))

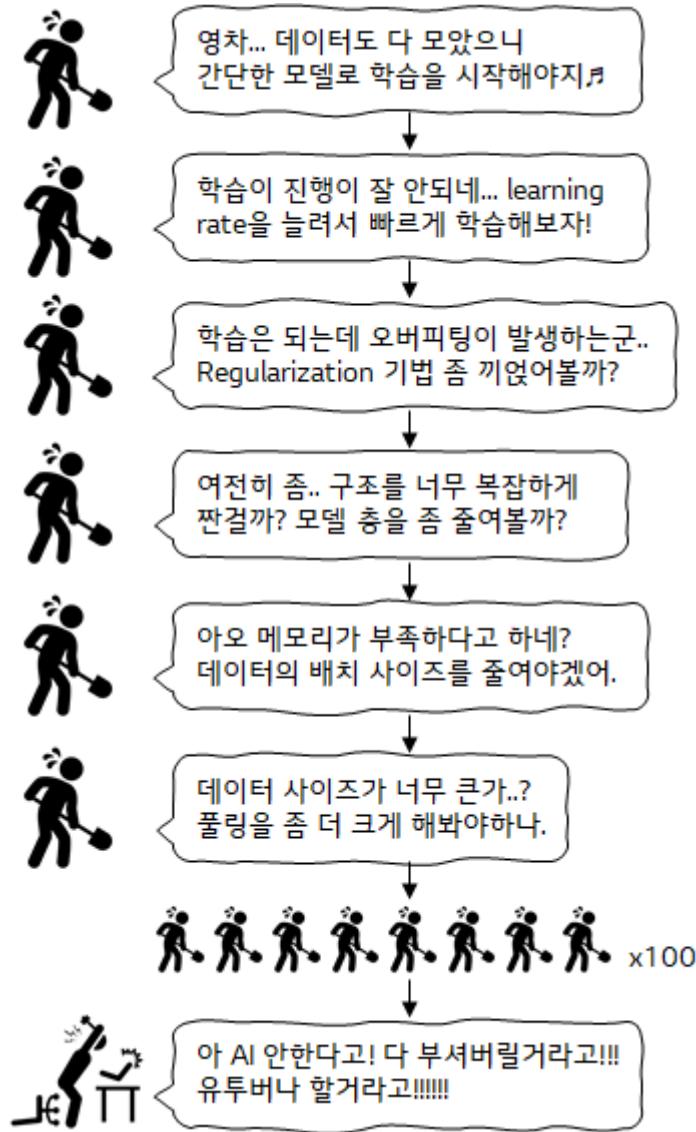
하지만 결과가 이럴 땐 이렇게 하는게 효과적이더라~ 하는 매뉴얼이 딱 있는 것은 아니며, 있다고 하더라도 현장의 다양한 태스크에 꼭 들어맞지 않습니다.

왜 이런 결과가 나왔는지를 해석하는 것도 사람의 뭇이기 때문에 우리는 문제의 원인을 유추하고 이에 기대어 현상을 개선하기 위한 방법을 찾아야 합니다.

따라서 실험을 진행하는 사람의 탄탄한 이론 배경과 더불어 경험과 노하우까지 풍부해야만 불필요한 실험의 반복 횟수를 줄일 수 있지요.

---

<sup>59</sup><https://wire.lgcns.com/confluence/display/~78628>



[그림 1. AI 학습은 반복 시도와 실패의 연속. 그렇지만 반복 시도라고 해도 한계가 있다.]

우리가 튜닝해야 할 대상은 너무나 다양합니다.

딥러닝 모델로 한정짓는다 해도 FNN, CNN, RNN, Transformer... 어떤 계열의 모델 구조를 이용하는게 좋을까요?

인공신경망의 층 수는요? 얼마나 깊게 하는게 좋을까요? 한 층에 들어갈 인공 뉴런의 수(필터 사이즈, 필터 수 등등..)는요?

얼마나 성큼성큼 학습시키는 것이 좋을까요(learning rate)? 한 번에 학습할 데이터의 수는 어느 정도가 적당할까요(mini-batch size)

몇 번이나 반복해서 보여줘야 적당할까요(epoch)? 어떤 최적화 기법을 쓸 것이며(optimizer), 손실 함수는 어떤 것을 쓰는게 효과적이며(cost function), 활성화 함수는 무엇이 좋을까요?

모델 학습을 위해 사람이 결정해주어야 하는 설정이 한두가지가 아닙니다.

딥러닝 기반의 AI가 머신러닝이나 률기반의 AI보다는 수작업의 공수가 적다고 했지만, 여전히 사람의 개입이 필요한 부분이 있습니다.

조금 귀찮은데, 스스로 척 하면 척 알아서 학습할 수 있는 인공지능은 없을까요?

## 11.2 스스로 진화하는 인공지능, AutoML

다행히 우리가 수많은 시행착오를 겪으며 스트레스를 받지 않아도, 자동으로 적절한 인공지능이 학습되도록 도와주는 기법이 있습니다.

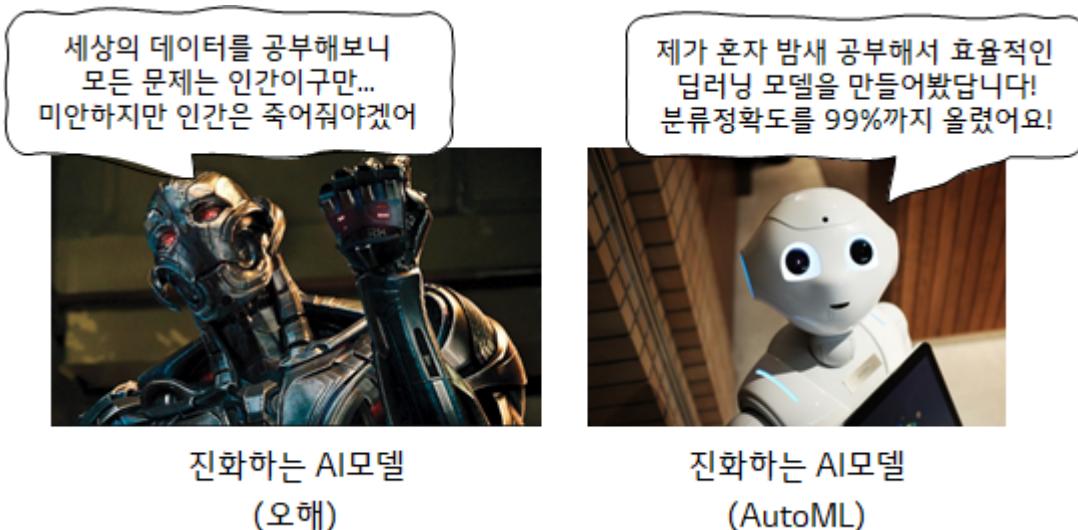
이를 **AutoML(Automated Machine Learning)**, 말 그대로 자동화된 기계학습이라고 부릅니다.

테크레터 1편에서는 인공지능이 스스로 똑똑해진다는 것은 오해일 뿐이라고 설명한 적이 있습니다. ([참고\(see page 0\)](#))

스스로 진화한다는 표현이 좀 과장되기는 했지만, 여기서 말하는 진화란 한정적인 의미입니다.

여기서는 '특정 태스크를 위한 모델 학습'에 한하여 사람이 주기적으로 실험에 개입하지 않아도 AI 스스로가 반복실험을 통해 성능을 개선하는 것을 말합니다.

가만히 놔두면 인공지능 스스로가 똑똑해져서 이것도 배우고 저것도 배우며 결국엔 사람을 지배하는 스토리의 진화가 결코 아니라는 점을 말씀드리고 싶습니다.



[그림 2. 인공지능이 스스로 똑똑해진다는 것은...]

AutoML의 역할은 크게 세 가지로 나누어 볼 수 있습니다.

첫 번째는 AI 모델을 학습하기 위해 데이터로부터 중요한 특징(feature)을 선택하고 인코딩하는 방식에 대한 Feature Engineering 자동화입니다.

두 번째는 AI 모델 학습에 필요한 사람의 설정들, 하이퍼파라미터를 자동으로 탐색해주는 것입니다.

세 번째는 AI 모델의 구조 자체를 더 효율적인 방향으로 찾아주는 아키텍처 탐색입니다.

딥러닝 모델은 비정형 데이터를 깊은 인공신경망에 태워서 자동으로 특징을 추출한다는 장점이 있기 때문에 이 중 feature engineering에 대해서는 별도로 다루지 않도록 하겠습니다.

하이퍼파라미터 탐색 자동화와 아키텍처 탐색 자동화에 대해 알아봅시다.

### 11.2.1 하이퍼파라미터 탐색 자동화

딥러닝 모델 학습에 필요한 하이퍼파라미터는 다양한 종류가 있습니다.

모델의 파라미터 업데이트를 얼마나 큰 단위로 할지를 결정하는 학습률(learning rate), 데이터를 얼마나 쪼개어 학습할지의 단위인 미니배치 사이즈(mini-batch size),

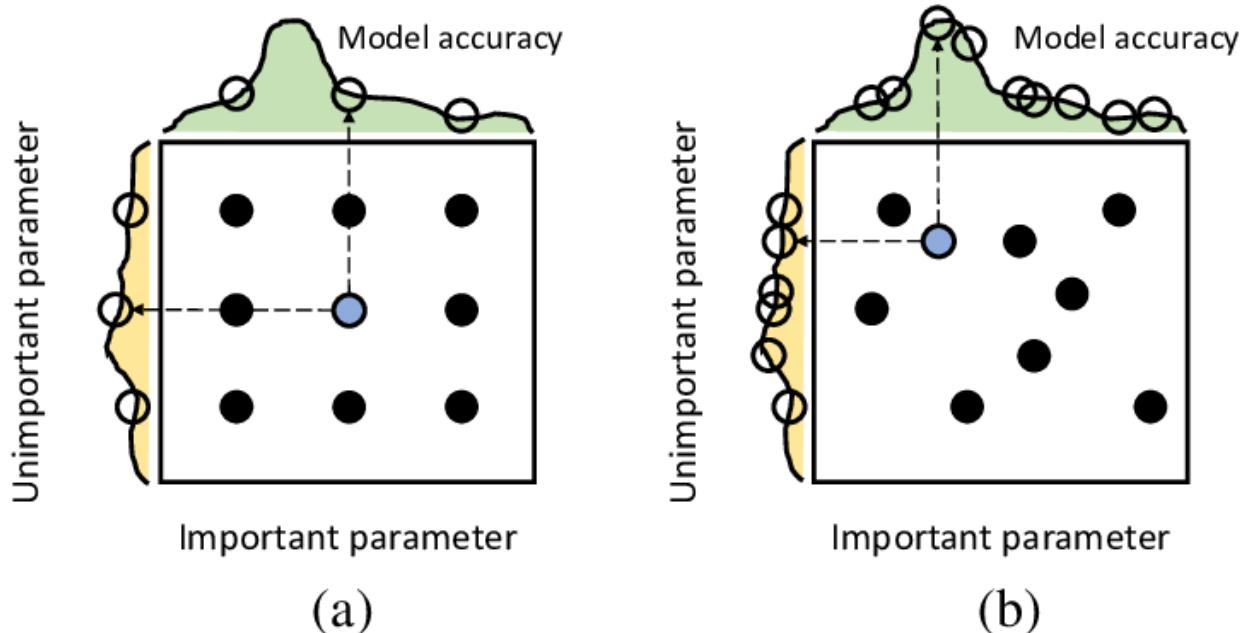
데이터를 몇 번 반복 학습할지에 대한 단위 에폭(epoch), 이외에도 모멘텀이라든지, 컨볼루션 필터의 수, 스트라이드 등등... 사람이 설정해주지 않아도 자동으로 결정되는 값은 하나도 없습니다.

많은 경우 딥러닝 학습 프레임워크(TensorFlow, PyTorch 등)에서는 기본적으로 잘 작동하는 설정을 디폴트로 제공하고 있지요.

하지만 기본 설정으로도 학습이 잘 되지 않는다면 실험 결과를 살핀 뒤 하이퍼파라미터를 조금씩 튜닝을 해줘야 합니다.

이게 위낙에 귀찮은 작업이다보니, 기존에 이미 여러 가지 하이퍼파라미터의 조합을 찾고자 하는 시도가 있었습니다.

자주 쓰이는 것 두 가지만 들자면 **그리드 서치(Grid search)**와 **랜덤 서치(Random search)** 방식이 있습니다.



[그림 3. (a)Grid search 방식, (b)Random search 방식]

그리드 서치 방식은 최적화할 하이퍼파라미터의 값 구간을 일정 단위로 나눈 후, 각 단위 조합을 테스트하여 가장 높은 성능을 낸 하이퍼파라미터 조합을 선택하는 방식입니다.

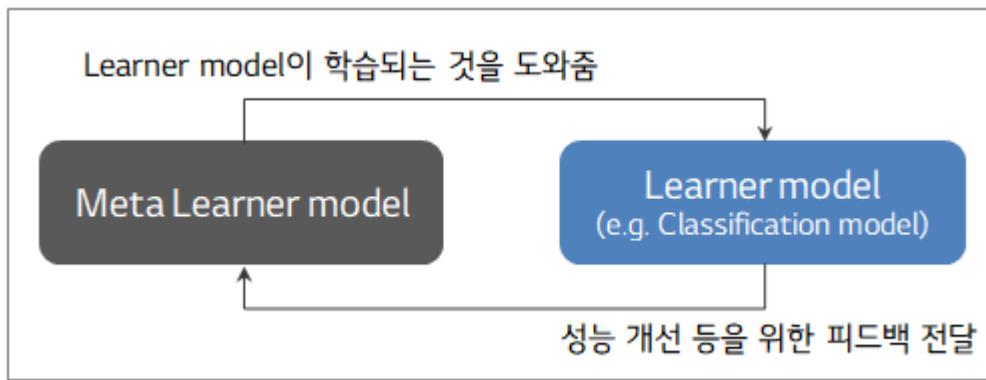
단순하지만 최적화 대상이 되는 하이퍼파라미터가 많다면 경우의 수가 기하급수적으로 많아져서 탐색에 오랜 시간이 걸립니다.

또한 불필요한 탐색에 시간을 허비하기도 하죠. 예를 들어 [그림 3]의 (a)에서 맨 왼쪽 열의 조합들은 굳이 세 번을 다 학습해볼 필요가 없지만 그리드 서치 방식을 이용할 경우 어쩔 수 없이 다 탐색을 수행해야 합니다.

반면 오른쪽의 랜덤 서치 방식은 랜덤하게 하이퍼파라미터의 조합을 테스트하는 방식인데, 그리드서치에 비해 비교적 빠르게 최적의 조합을 찾아내곤 합니다.

위의 두가지 방식은 어찌 보면 경우의 수를 찾는 단순 탐색법에 불과한데요, 단순히 for문을 이용하여 구현할 수도 있습니다.

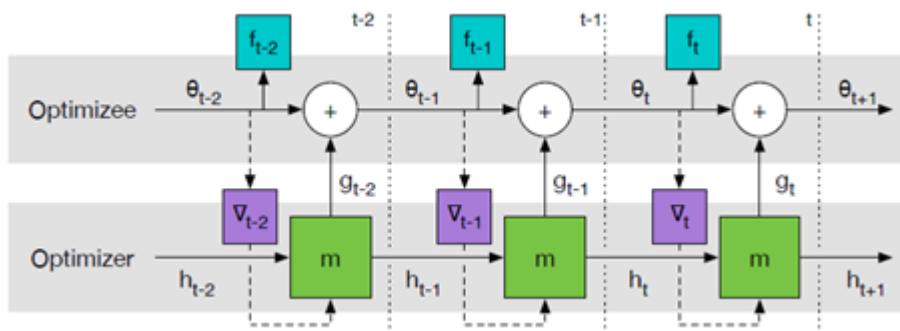
그러나 최근의 AutoML 방식에서는 하이퍼파라미터도 모델을 통해 탐색하곤 합니다.



[그림 4. 하이퍼파라미터를 찾아주는 Meta Learner]

주어진 태스크를 수행하는 Learner가 본래 우리가 알고 있던 AI 모델이라면, 이 AI모델이 좋은 성능을 달성하기 위한 최적의 하이퍼파라미터의 조합을 찾아주는 **Meta Learner**가 별도로 있습니다.

Meta Learner는 대부분 RNN과 강화학습을 활용하여 최적의 하이퍼파라미터를 탐색합니다.



[그림 5. Meta learner의 하이퍼파라미터 탐색]

Meta Learner의 하이퍼파라미터 조합대로 학습한 Learner의 학습 성능 결과를 Meta Learner로 다시 전달하고, Meta Learner는 이를 또 개선하기 위한 다른 하이퍼파라미터 조합을 내며 Learner는 이 조합으로 또다시 학습하고... 이러한 과정을 반복하다 보면 최적의 조합을 찾아낸다는 이론입니다.

이 때 Meta Learner가 수행하는 학습에 대해, (Learner의)학습을 위한 학습이라는 뜻에서 '메타학습', 또는 'Learn to Learn'이라고 표현하기도 합니다.

## 11.2.2 아키텍처 탐색 자동화

하이퍼파라미터 뿐 아니라 최적의 아키텍처를 찾아주는 방법도 있습니다.

아키텍처는 모델을 이루는 구조를 말하는데, 사람이 어떤 방식으로 모델 구조를 짠지 생각하지 않아도 자동 탐색을 통해 최적 구조를 찾을 수 있습니다.

특히 딥러닝 모델의 경우에는 인공신경망을 활용하기 때문에, **NAS(Neural Architecture search)**라고 부릅니다.

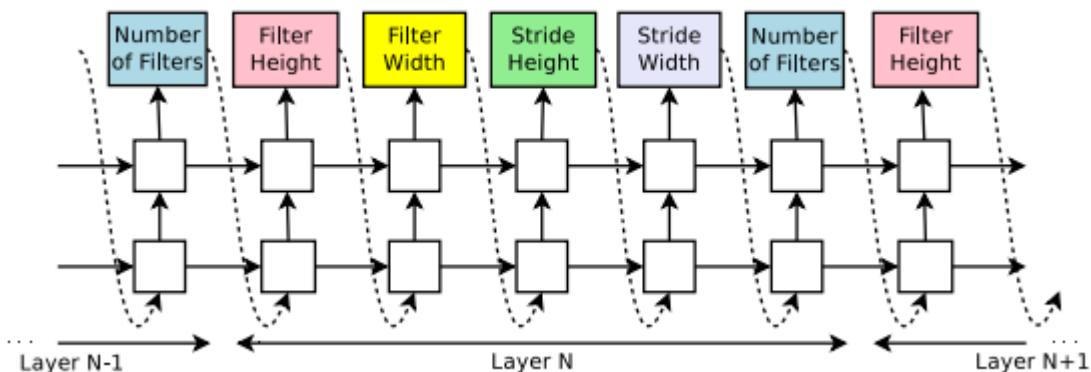
(참고: AutoML이라는 용어를 사용할 때 많은 곳에서 좁은 의미로 NAS만을 일컫기도 하지만, 넓은 의미에서 AI 자동 학습 기법이라는 측면을 더 알아주시면 좋겠네요.)

NAS의 경우도 마찬가지로 대부분 Meta Learner와 Learner로 이루어져 있어서, Learner가 본 과제를 수행하는 AI 모델이라면

Meta Learner가 어떤 구조의 신경망을 만들면 좋은지, 아키텍쳐 구성을 고민하게 됩니다.

Meta Learner는 역시 RNN과 강화학습을 접목한 형식으로 구성해볼 수 있습니다.

Meta Learner는 Learner의 인공신경망 아키텍처가 어떻게 구성되면 좋을지를 결정하여, Learner의 태스크 수행 결과를 보상으로 활용합니다.



[그림 6. 컨트롤러 역할을 수행하는 Meta Learner를 RNN으로 구성한 것]

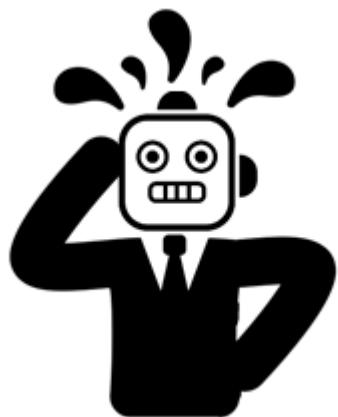
이외에도 진화 알고리즘이나 경사하강법을 기반으로 한 NAS 방식도 있습니다.

## 11.3 AutoML 특징

AutoML을 활용하면 사람이 한 땀 한 땀 구조를 고민하고, 하이퍼파라미터를 튜닝할 필요 없이 최적의 환경을 결정해 줍니다.

AutoML은 일반적으로는 사람이 고민한 모델 이상의 성능을 낼 수 있다고 합니다.

기계는 사람이 생각도 못한 조합의 설정이나 구조를 시도해볼 수 있고, 기존의 설정 관습이나 제약에 얹매이지 않기 때문이죠.



[그림 7. 상상도 못한 아키텍처!]

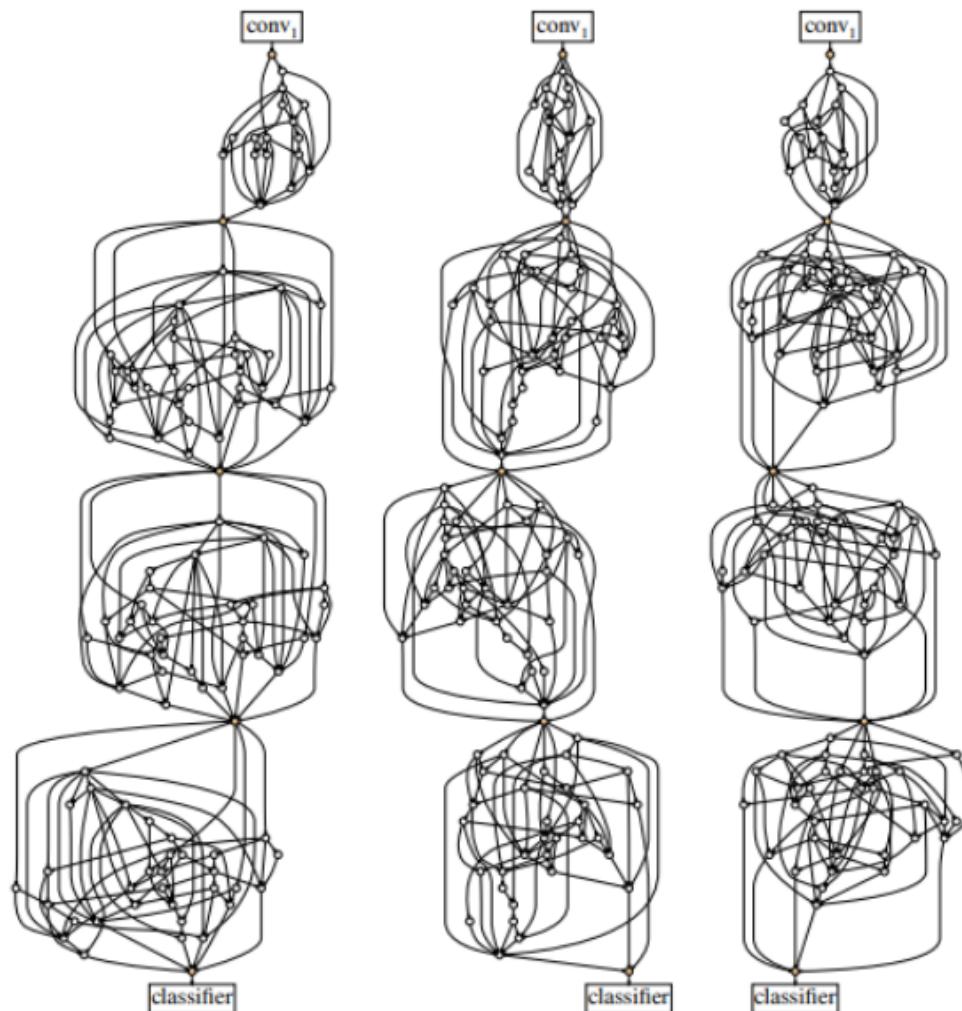
아래 그림은 작년 AI 커뮤니티를 뜨겁게 달구었던 논문의 신경망 모델인데, AutoML을 통해 자동으로 탐색된 네트워크 구조입니다. (S. Xie, et al., 2019, FAIR)

이 모델은 보기에는 불펜뚱처럼(?) 보여도 이미지 인식 과제에서 사람이 만든 신경망 모델보다 훨씬 적은 연산량으로 유사한 성능, 또는 더 좋은 성능을 보였다고 합니다.

기존의 NAS 방법들은 Meta Learner가 탐색할 아키텍처의 공간이 한정되어 있어서, 그 중 최적의 아키텍처를 찾고자 하였다면,

이 논문 저자는 이러한 제약마저 없어져야 진정한 의미의 AutoML이라고 할 수 있지 않을까? 하고 생각했다고 합니다.

이러한 모티브에서, 랜덤 그래프 생성 방법론을 기반으로 NAS를 진행하자 아래 그림처럼 기존 구성 방법의 틀을 완전히 깬 모델이 만들어졌습니다.



[그림 8. AutoML을 통해 자동으로 탐색된 인공신경망 아키텍처]

하지만 사람의 손길을 덜 필요로 하고 좋은 성능 결과를 얻는 대신, 기계가 다양한 시도를 해보도록 오랜 시간을 기다려야 합니다.

(그런데 또... 어찌 보면 사람이 반복 실험을 하며 겪는 시간보다는 적을 수도 있습니다)

또한 고사양의 하드웨어 스펙이 뒷받침 되어야만 이러한 창조적인 시도를 지원할 수 있습니다.

Learner 모델과 Meta Learner 모델이 동시다발적으로 학습을 해야 하니까요.

결론적으로 사람이 하이퍼파라미터를 찾거나 구조를 고민하기 귀찮다면 기계에게 시간과 돈을 써야 한다는 말이 되겠네요...

이런 고민을 조금이나마 덜어주기 위해 AutoML 서비스를 제공하는 업체들이 있습니다.

## 11.4 AutoML 서비스

AWS, Azure, GCP(Google Cloud Platform)와 같은 CSP 업체는 모두 일종의 AutoML 서비스를 제공하고 있습니다.

단 하이퍼파라미터 선택이나 모델 구조 선택, 이미지/텍스트 관련 태스크별 지원 기능에 차이가 있으므로 원하는 기능이 제공되는지 살펴보고 이용하시기 바랍니다.

최신 기술이니만큼 수시로 업데이트되고 있으니 당장은 제공하지 않는 기능이라 해도 향후에는 지원될 가능성이 높습니다.

CSP 서비스 중 대표적인 구글 클라우드 플랫폼은 제공하는 AutoML 기능이 가장 다양해보입니다.

시각	<b>AutoML Vision</b>	<b>AutoML Video Intelligence</b> <small>베타</small>
	클라우드나 에지의 이미지에서 유용한 정보를 도출합니다.	강력한 콘텐츠 탐색 기능과 매력적인 동영상 환 경을 지원합니다.
	<a href="#">자세히 알아보기</a>	<a href="#">자세히 알아보기</a>
언어	<b>AutoML Natural Language</b>	<b>AutoML Translation</b>
	머신러닝을 통해 텍스트의 구조와 의미를 드러 냅니다.	언어를 동적으로 감지하고 각 언어로 번역합니 다.
	<a href="#">자세히 알아보기</a>	<a href="#">자세히 알아보기</a>
구조화된 데이터	<b>AutoML Tables</b> <small>베타</small>	
	구조화된 데이터에서 최신 머신러닝 모델을 자 동으로 빌드하고 배포합니다.	
	<a href="#">자세히 알아보기</a>	

[그림 9. Google AutoML 서비스]

Google AutoML은 구글 계정을 만들고 크레딧을 지불하여 이용할 수 있습니다.

제공하는 기능은 이미지에 대한 분류(classification)와 객체 탐지(detection), 동영상에 대한 분류(classification)와 객체추적(visual tracking),

자연어에 대한 분류(classification), 객체명인식(named entity recognition), 감정분류(sentiment classification), 번역(translation),

정형 테이블 데이터에 대한 회귀(regression) 및 분류(classification)이며,

위 기능들을 엣지 레벨(엣지 기기 탑재)과 클라우드 레벨(API형식)로 제공합니다.

학습은 데이터 및 수행 과제에 따라 달라지지만, 대체로 몇 시간~1일 이내의 학습 시간을 보입니다.

비용도 상당한데요, 모든 계정에 기본으로 지급되는 무료 크레딧(100\$상당)을 활용하여 1~2회 정도의 학습을 수행할 수 있으며

비용이 걱정되는 경우 예산 커트라인을 지정하여 해당하는 만큼만 학습할 수도 있습니다.

비싸다고 생각할 수도 있지만, 사람이 직접 튜닝한다고 했을 때 코드 구현 및 시행착오에 걸리는 시간과 GPU 서버 사용료를 생각하면 더 저렴할지도 모르겠습니다.

Google AutoML로 학습된 모델은 어떤 하이퍼파라미터를 활용하며, 어떤 네트워크 아키텍쳐를 가지고 있는지 알 수 없습니다.

단지 우리는 비용을 지불하고, 해당 데이터에 대해 최적으로 맞춰진 모델을 이용할 수 있을 뿐입니다.

표준 데이터로 실험했을 때 대체로 사람이 학습시킨 모델과 유사하거나 더 높은 성능을 내곤 합니다.

딥러닝에 대해 잘 모르지만, 비용과 데이터가 있다면 활용해보는 것도 좋겠죠?

이외에 다양한 제공처가 있으니 나에게 맞는 서비스를 찾아보시기 바랍니다.

Tool	Platform	Input data sources		Data pre-processing	Data types detected				Feature engineering	ML Tasks	Model selection and Hyperparameter optimization		Quick start / early stop	Model evaluation / Result analysis / Visualization
		Spreadsheet datasets	Image, text		Numerical	Categorical	Datetime	Time-series	Other (Hierarchical types) (7)	Datetime, categorical processing Imbalance, missing values Feature selection, reduction Advanced feature extraction (8*)	Unsupervised learning (10*)	Ensemble Genetic algorithm Random search Bayesian search Neural architecture search	Quick finding of starting model	Allow maximum limit search time Restrict time consuming combination of components Model dashboard
TransmogrifAI	Apache Spark	Y	N	Y(*)	Y Y	Y Y	Y Y	Y Y	Y Y	Y Y	N Y N Y Y N N			Y Y
H2O-AutoML	AWS, GCP, Azure	Y	N	Y	Y Y	Y Y	Y Y	N Y	Y Y	Y N Y	N Y N Y N N N			Y Y Y Y Y Y
Darwin (+)	GCP	Y	N	Y	Y Y	Y Y	Y Y	N Y	Y Y	Y Y	Y Y N N Y Y			Y N Y Y Y Y
DataRobot (+)	AWS, GCP, Azure	Y	Y	Y	Y Y	Y Y	Y Y	N Y	Y Y	Y Y	Y Y Y Y Y N	Y(12*)		Y Y Y Y
Google AutoML (+)	Google Cloud	N	Y	Y					N Y Y Y Y Y Y		Y Y Y Y Y Y Y			Y Y Y Y Y Y
Auto-sklearn		Y	N	N	N N N N N N	N N N N	Y(2*)	Y Y Y Y Y Y	N Y N Y Y N Y					Y Y Y Y Y Y
MLjar (+)	MLJAR Cloud	Y(3*)	N	Y	Y Y	N N N N	N Y	Y(4*)	N N N Y(5*)	N Y N Y N N N N	N N N N N N N N			Y Y Y Y Y N
Auto_m1		Y	N	N	N N N N N N	N N N N	Y	Y Y Y Y Y Y	N Y N Y N N N N					N N Y Y Y Y
TPOT		Y	N	N	N N N N N N	N N N N	Y	N Y Y Y Y Y	N Y N Y N N N N					Y N Y Y Y N
Auto-keras		Y	Y	N	N N N N N N	N N N N	Y	Y Y N Y N N N N	Y Y N N N N Y Y Y Y					Y N Y N Y
Ludwig		Y	Y	Y(*)	Y Y	N Y Y N	Y Y N	Y Y Y Y Y Y	N Y N Y Y Y Y					Y N Y Y N N
Auto-Weka		Y	N	N	Y Y	N N N N N N	N Y	Y Y N Y N Y N Y N	Y N Y N Y Y N N					Y Y Y N N N
Azure ML (+)	Azure	Y	Y	Y(6*)	Y Y	Y Y	N Y	Y Y Y Y Y Y	N Y N Y N Y Y N					Y Y Y Y Y Y
H2O-Driverless AI (+)	AWS, GCP, Azure	Y(3*)	Y	Y	Y Y	Y Y	Y Y	Y Y Y Y Y Y	Y Y Y Y Y Y Y N					Y Y Y Y Y Y

[그림 10. AutoML Tool. 2019년 기준 (자료<sup>60</sup>)]

## 11.5 마무리

딥러닝은 자동화의 패러다임을 바꾸었습니다.

기존의 자동화 프로그래밍이 인간의 지식을 프로그래밍 언어로 명문화하여 기계에게 주입하는 방식이었다면,

딥러닝은 인간의 사고 방식 자체, 즉 인공적인 신경망을 기계에 구현한 뒤 수많은 데이터를 보여주며 말로는 표현할 수 없는 지식을 학습시켰지요.

AutoML은 여기에 다시 한 번 자동화를 추가하고 있습니다.

이제는 사고 방식이 탄생될 수 있는 환경만 조성해주면 기계가 알아서 나머지를 수행하게 됩니다.

사람은 태스크 수행을 위한 최적의 뇌 구조가 만들어질 수 있도록 지원하고 결과를 지켜보는 역할을 합니다.

60 <https://arxiv.org/pdf/1908.05557.pdf>

이번 시간은 'AutoML'에 대해 설명드렸습니다.

다음 시간에는 2020년 마지막 AI 테크레터 주제로 찾아뵙겠습니다.

감사합니다 😊

---

## 참고자료

- A Kernel Design Approach to Improve Kernel Subspace Identification, K. Pilario, et al., 2020, [https://www.researchgate.net/publication/341691661\\_A\\_Kernel\\_Design\\_Approach\\_to\\_Improve\\_Kernel\\_Subspace\\_Identification](https://www.researchgate.net/publication/341691661_A_Kernel_Design_Approach_to_Improve_Kernel_Subspace_Identification)
- 최신 딥러닝 기술 동향, AI빅데이터연구소 이주열 위원
- Neural Architecture Search with Reinforcement Learning, Zoph & Le, 2016, <https://arxiv.org/pdf/1611.01578.pdf>
- Learning Transferable Architectures for Scalable Image Recognition, B. Zoph, et al., 2017, <https://arxiv.org/pdf/1707.07012.pdf>
- 자동 기계학습(AutoML) 기술 동향, ETRI, 2019, [https://ettrends.etri.re.kr/ettrends/178/0905178004/34-4\\_32-42.pdf](https://ettrends.etri.re.kr/ettrends/178/0905178004/34-4_32-42.pdf)
- Exploring Randomly Wired Neural Networks for Image Recognition(FAIR), S. Xie, et al., 2019, <https://arxiv.org/abs/1904.01569>
- 박경찬 - Exploring Randomly Wired Neural Network For Image Recognition, [https://www.youtube.com/watch?v=kKaUNLrkjJM&ab\\_channel=KoreaUnivDSBA](https://www.youtube.com/watch?v=kKaUNLrkjJM&ab_channel=KoreaUnivDSBA)
- “데이터 과학자 없는 머신러닝” AutoML의 이해, itworld, <https://www.itworld.co.kr/news/129362>
- Google Cloud Platform AutoML, <https://cloud.google.com/automl>
- Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools, A. Truong, et al., 2019, <https://arxiv.org/abs/1908.05557>

## 12 [12편] 설명 가능한 인공지능, XAI

---

안녕하세요, CTO AI 빅데이터연구소입니다.

한 달에 두 번씩 **AI 테크레터**를 통해 인공지능 지식을 임직원 여러분들께 공유드리고 있습니다.

모든 CNSer가 이해하실 수 있도록 쉽게 작성하려고 하니, 상세 기술에 대한 궁금증이 생기시면 댓글이나 이메일을 통해 언제든 연락 바랍니다 😊

본 업로드는 [TECH wiki AI게시판\(see page 7\)](#)에서 연재됩니다.

작성 : CTO AI 빅데이터연구소 AI 기술팀 김명지 책임연구원/Cognitive AI LAB<sup>61</sup>

- 종종 이해할 수 없는 결정을 내리는 AI(see page 168)
- 설명 가능한 인공지능, XAI(explainable Artificial Intelligence)(see page 171)
  - XAI의 필요성(see page 171)
- XAI를 위한 접근법(see page 172)
  - 어텐션 메커니즘(Attention Mechanism)을 활용한 XAI(see page 172)
  - 설명하는 법 학습하기(Learn to explain)(see page 174)
- 마무리(see page 175)
- 2020 AI 테크레터 연재를 마무리하며...(see page 176)

오늘은 설명 가능한 인공지능, Explainable AI에 대해 알아보겠습니다.

지난 시간까지의 내용이 궁금하신 분은 ★[AI Tech Letter\(see page 7\)](#)★를 확인하시기 바랍니다.

### 12.1 종종 이해할 수 없는 결정을 내리는 AI

속을 알 수 없는 AI가 있다??

네 여기 있습니다. 인공신경망 기반의 딥러닝 모델은 사람이 그 결과를 해석하기 어렵다는 단점이 있는데요, 딥러닝 모델은 데이터가 충분한 경우 일반적으로 룰 기반의 모델이나 머신러닝 기반의 모델보다 좋은 성능을 보일 순 있지만,

어째서 모델이 이런 결과를 도출했는지에 대해 사람의 결과 해석 여지가 매우 부족합니다.

룰 기반 모델이라면 모델이 어떤 입력에 대해 정답을 맞추거나 틀리더라도

"아~ A 규칙에는 걸렸지만 B 규칙에 안맞아서 틀렸구나, 이러한 데이터도 맞출 수 있으려면 B에 예외 규칙을 좀 더 넣어야겠어!"

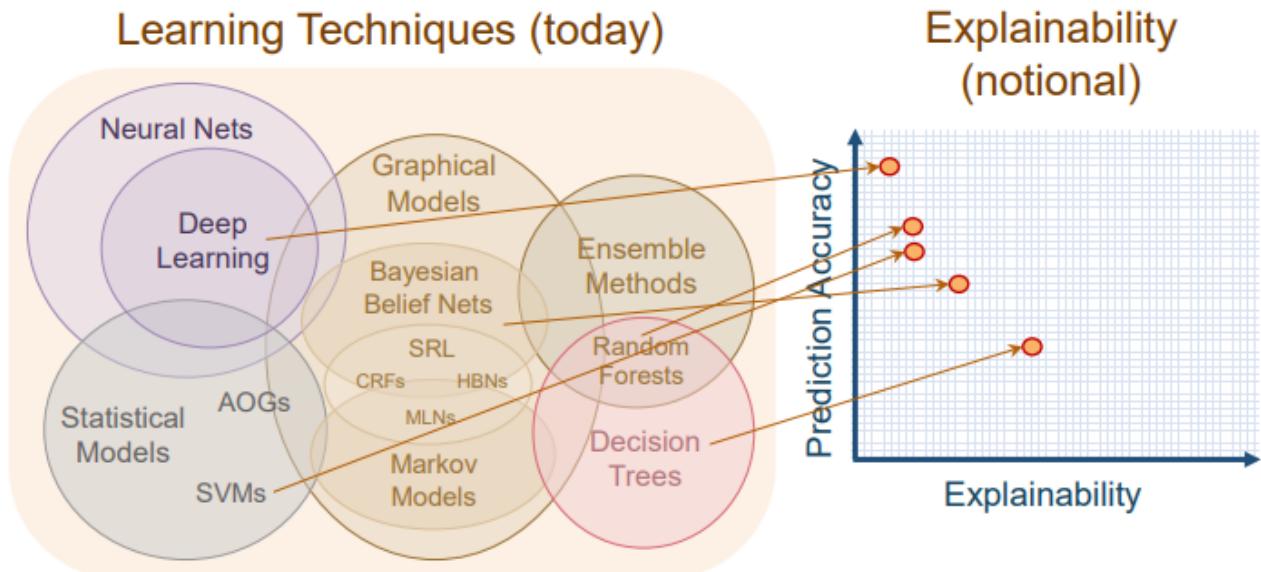
같은 모델 개선방안이 나올 수 있습니다.

머신러닝 기반 모델의 경우도 마찬가지입니다.

---

<sup>61</sup><https://wire.lgcns.com/confluence/display/~78628>

회귀 모형이나 의사 결정 나무 같은 경우에도 어떤 입력 변수가 얼마나 영향을 미치는지, 학습된 모델을 통해 해석할 수 있는 여지가 있습니다.



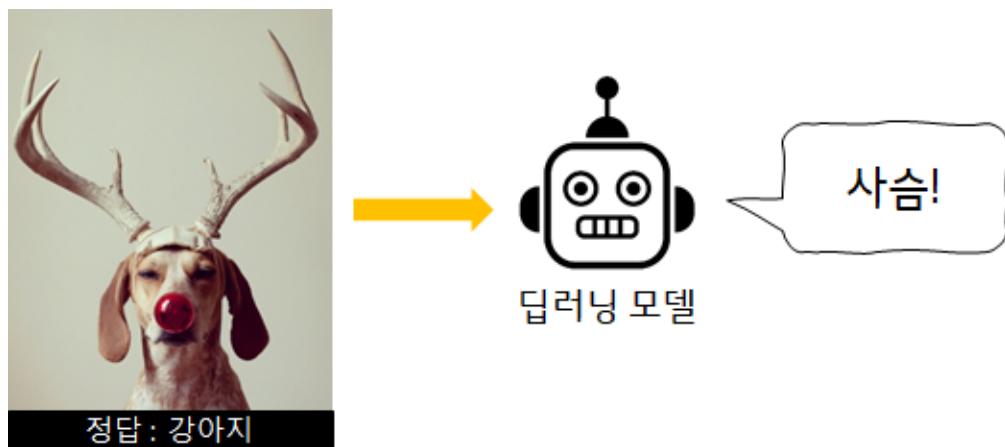
[그림 1. AI 기술별 정확도와 설명력 (자료:DARPHA)]

하지만 딥러닝 모델은요...?

규칙이나 몇 가지의 입력 변수만으로 판단하기 어려운 태스크(특히 비정형 데이터에 대한 태스크)를 수행하는 경우, 딥러닝 방식은 말로 표현하기 어려운 특징까지도 잘 포착하여 좋은 성능을 낼 수 있습니다.

수많은 파라미터로 복잡하게 얹힌 인공신경망의 연산이 우리는 뭔지 모를 어떤 판단을 잘 내리고 있는 것 같긴 한데, 간혹 "윙? 이 쉬운 걸 틀린다고?" 하는 경우가 발생합니다.

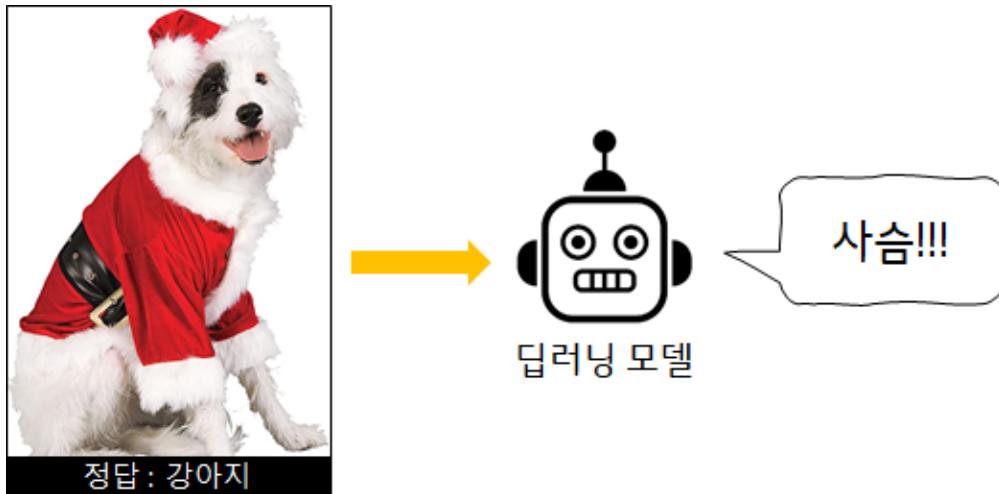
하지만 딥러닝 모델은 설명력이 부족하기 때문에 어째서 이런 결과가 나왔는지 우리는 알 수가 없습니다.



[그림 2. 딥러닝 모델의 추론 미스(1)]

그래도 위 예시같은 정도의 오답은 왜 틀렸을지 대충 짐작할 수 있습니다.  
 비록 모델은 사슴이라는 판단 결과 외에 아무런 추가 단서를 제공해주지 않았지만,  
 우리는 '그래, 뿔이 달렸으니까... 뭐 기계는 사슴이라고 생각할수도 있겠지..' 하고 정상참작 해줄 수 있습니다.

하지만...?



[그림 3. 딥러닝 모델의 추론 미스(2)]

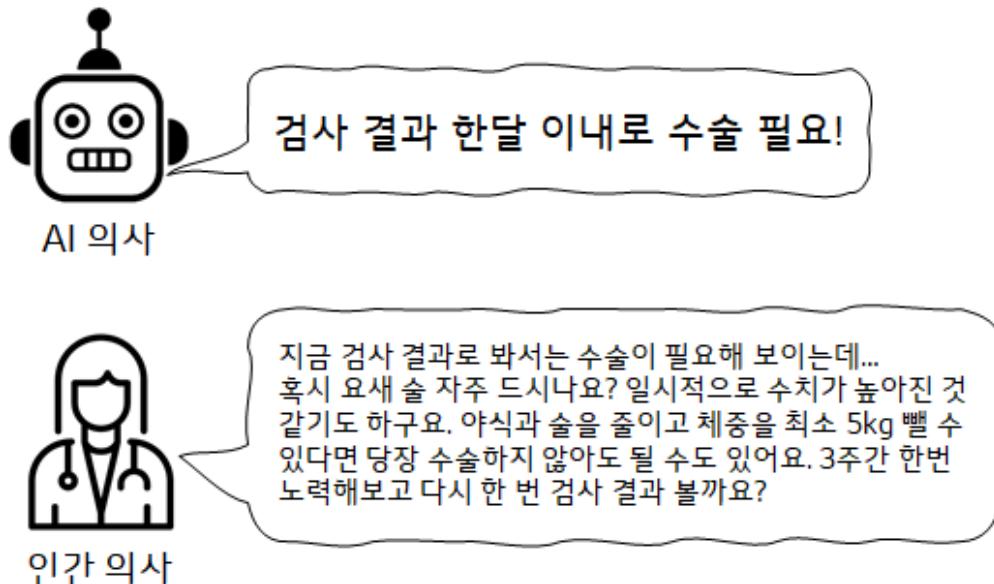
이번엔 왜 틀린 걸까요? 어딜 봐도 사슴스러운 구석이라고는 없는데, 왜 사슴이라고 한걸까요?  
 차라리 산타라고 하면 모를까, 왜 산타라고는 안한거죠? 왜 강아지라고는 판단하지 않았을까요?  
 어떨 때 딥러닝은 잘 판단할 수 있고, 어떨 때는 실패하는 걸까요?  
 언제 딥러닝 모델을 신뢰할 수 있을까요?  
 이러한 판단 오류를 수정하기 위해서는 무엇을 더 해야 할까요?

애석하지만 위 어느 질문에도 딥러닝 모델은 대답해줄 수 없습니다.  
 우리는 단지 그럴싸한 가정을 찾아서 이를 해결하는 방법이라고 알려진 보완 사항을 덧붙여 모델 추가 학습을 하고,  
 해결이 되면 '음 그게 문제였나보네... 어쨌든 고쳐졌으니 됐는걸.' 식의 마무리를 할 수밖에도.  
 동물의 종류를 구별하는 정도의 과제를 수행한다면 큰 문제 없을겁니다.  
 하지만 이게 만일 사람의 생명을 다루는 의료 과제였다면? 법적인 문제나, 국가 보안이나, 개인/기업의 신용과 관련된 민감한 주제였다면?  
 사람이라면 범하지 않을 비상식적인 결과를 도출하는 AI 모델에 대해, 아무리 전반적인 성능이 뛰어나다고 해도 결과가 타당한지 신뢰하기는 어렵겠죠.  
 이런 인공지능의 한계를 위해, 인공지능에 설명력을 부여하는 연구 분야가 있습니다.

## 12.2 설명 가능한 인공지능, XAI(eXplainable Artificial Intelligence)

### 12.2.1 XAI의 필요성

AI의 판정 정확도가 사람을 능가하는 시대가 왔다고 할 때,  
AI 의사와 인간 의사, 누구에게 내 운명을 맡기고 싶으신가요?



[그림 4. AI 의사와 인간 의사 (실제 사건 아님 주의)]

부연설명이 없고 단호박이지만 그래도 우수한 정확도를 보이는 AI 의사의 의견을 따르시겠습니까?

아니면 틀릴 가능성은 조금 있겠지만, 왜 이렇게 결정했는지 이유를 설명해주고 다른 결정을 내리기 위해 문제되는 부분을 지적해주는 인간 의사의 의견을 따를까요?

보통은 사람이 이해하기 어렵지만 정확도가 높은 모델 보다는 해석이 가능한 모델을 선택했을 때 더 안정감을 느낄 것이라고 생각되네요.

그렇다면 AI 의사도 "간수치가 높고 체중이 몇 KG 이상인데다 이러저러하기 때문에 수술이 필요하다고 결정했음."이라는 부연설명을 해주면 되지 않을까요?

아무리 좋은 AI 모델이라고 해도 고객사에 도입할 모델이라고 친다면,

"아 글쎄요... 이게 웬만해서는 잘 안틀리는 모델인데.. 왜 그런지는 면밀히 조사해보겠습니다(사실 조사해도 알수없음)." 보다는

"@#@하고 ###하기 때문에 \$\$\$같은 케이스에서 판단 오류가 발생했습니다. \*\*\* 처리를 이번주 중 수정하면 다시는 이런 오류가 발생하지 않을 거라 생각합니다."

라고 대처하고 싶겠죠?

인공신경망으로 구성된 딥러닝 모델은 알고리즘의 복잡성으로 인해 "블랙박스"라는 별칭으로도 불립니다.

때문에 딥러닝 모델에 설명력을 부여하는 것은 사용자가 모델의 최종 결정을 이해하고, 결과의 타당성을 확인할 수 있게 해줍니다.

즉 XAI에 대한 연구는 모델에 대한 신뢰성으로 연결될 수 있다고 말씀드릴 수 있겠네요.



[그림 5. 설명없는 딥러닝 모델...]

인공지능에게 설명력을 부여하는 방법에 대한 연구 분야를 **XAI(eXplainable AI)**, 설명 가능한 인공지능이라고 합니다.

XAI는 모델에 설명 가능한 근거와 해석력을 부여해서 투명성, 신뢰성을 확보하고자 하는 것이 목적입니다.

이렇게 되면 AI 모델의 비즈니스 활용에 있어 한계를 극복할 수 있는 가능성성이 생기겠죠?

### 12.3 XAI를 위한 접근법

XAI를 가장 본격적으로 연구하는 대표 기관에는 DARPA(Defense Advanced Research Projects Agency)가 있습니다.

DARPA는 미국 국방성의 연구 개발을 담당하는 기관으로, 2017년부터 XAI를 연구해왔습니다.

DARPA의 자료에 따르면 AI 모델을 위한 XAI로 크게 세 가지의 접근 방식이 있습니다.

첫 번째로, **기준 AI 모델에 설명할 수 있는 어떤 모듈을 덧붙이는** 방식입니다.

여기에는 다시 몇 가지 구체적인 방법이 있습니다.

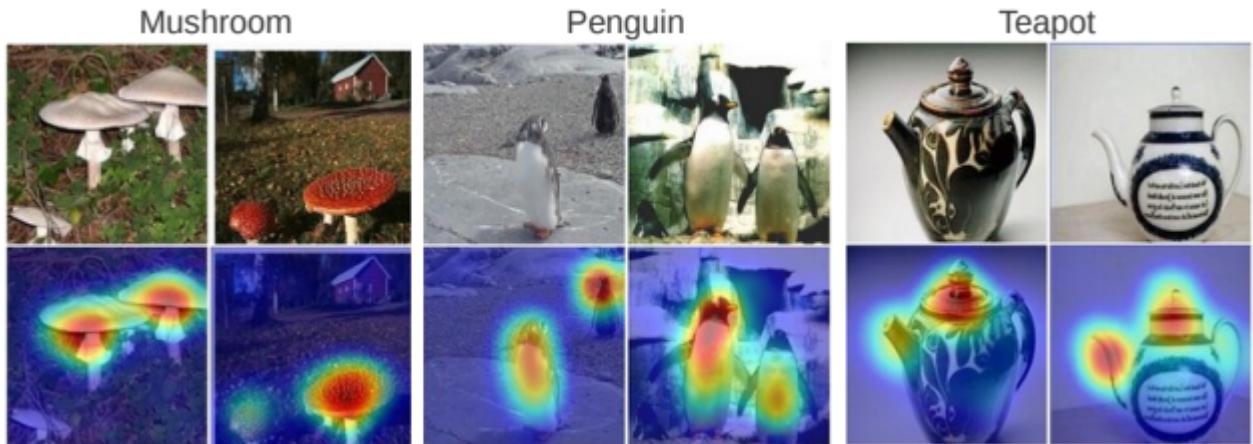
#### 12.3.1 어텐션 메커니즘(Attention Mechanism)을 활용한 XAI

지난 시간의 [테크레터\(참고\)\(see page 144\)](#)에서 소개된 어텐션 메커니즘은 XAI로서의 기능을 수행합니다.

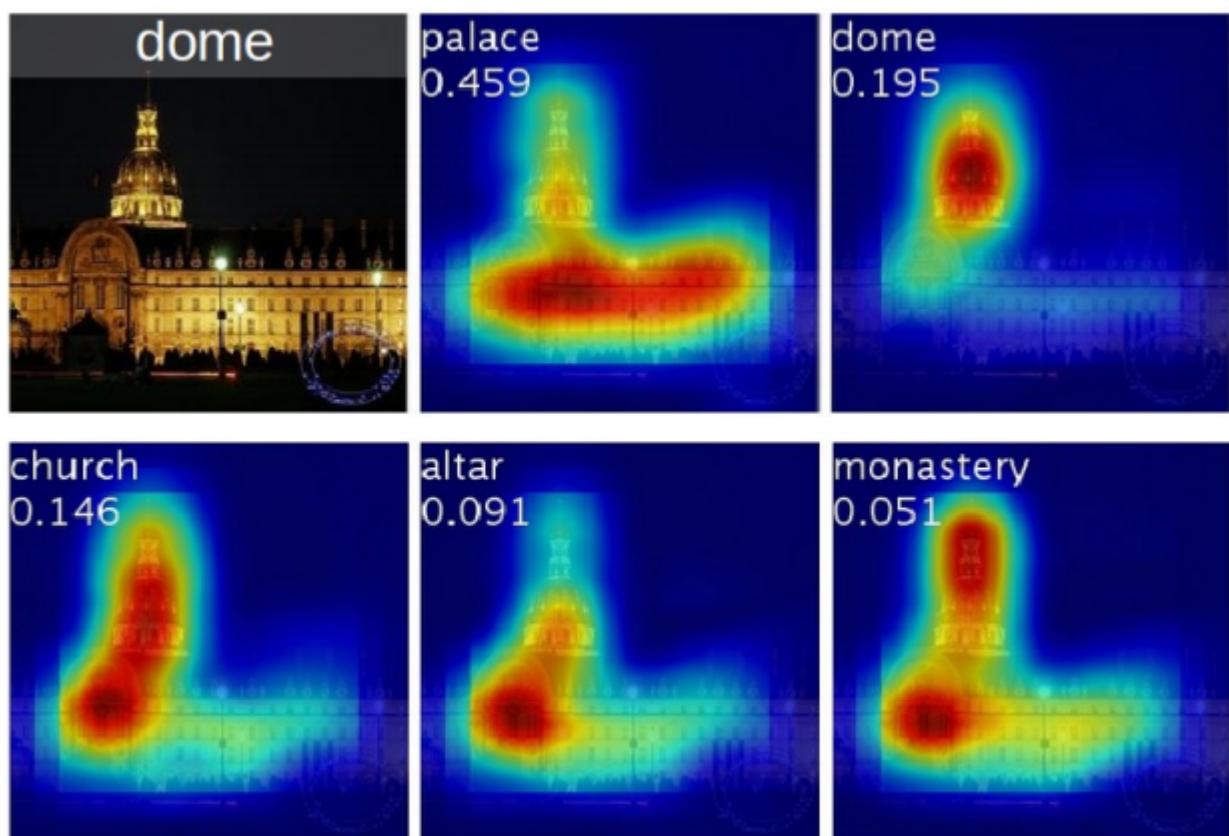
어텐션 메커니즘은 딥러닝 모델이 어떤 결정을 내릴 때, 입력 데이터의 어떤 부분에 집중해서 판단했는지를 시각화해 보여줄 수 있습니다.

집중된 영역이 반드시 판단을 내리게 된 '원인'이라고 볼 수만은 없지만,

'아 이런 부분이 중요하다고 생각해서 집중한 결과 이렇게 판단했구나~'라는 인사이트를 얻을 수 있습니다.



[그림 6. 딥러닝 모델이 버섯, 펭귄, 찻주전자로 분류한 이미지에 대해 입력 데이터의 어떤 부분을 집중했는지 시각화한 것]



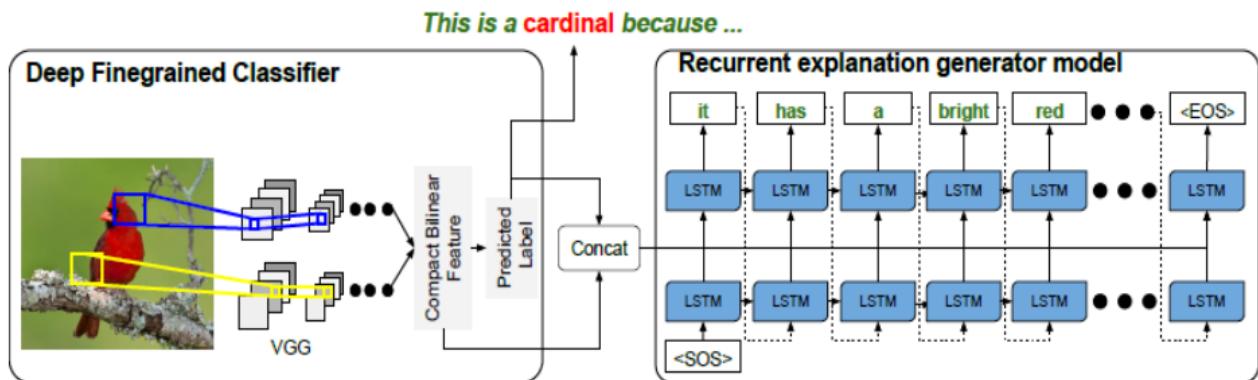
[그림 7. 첫 번째 사진에 대해, 딥러닝 모델이 분류한 상위 5개 결과에 대해 집중한 영역을 시각화한 것]

모델이 건물의 어떤 부분에 집중하는지에 따라 궁전으로도, 둑으로도, 교회로도 분류할 수 있는 점이 신기하죠?

이러한 시각적 정보는 우리에게 AI 모델의 판단을 조금이나마 이해할 수 있게 합니다.

### 12.3.2 설명하는 법 학습하기(Learn to explain)

이 방식은 판단을 하는 딥러닝 모델에 RNN 모듈 등을 덧붙여 인간이 이해할 수 있는 방식의 설명을 생성하도록 하는 방식입니다.



[그림 8. CNN 계열의 모델이 이미지에 대해 판단하면 RNN 모듈이 설명을 생성]

이렇게 만든 모델은 어째서 이러한 판단을 결정했는지에 대해 문장을 작성할 수 있습니다.



[그림 9. 새 종류를 분류하는 모델이 판단 결과를 설명]

하지만 설명 생성에 한계가 있어 널리 활용되는 방식은 아닙니다.

이 외에도 딥러닝 모델이 해석 가능한 모듈 구성요소로 이루어진 경우, 판정 결과가 어떤 모듈 경로를 따라 연산되는지 파악하는 모듈러 네트워크(Modular Networks) 방식,

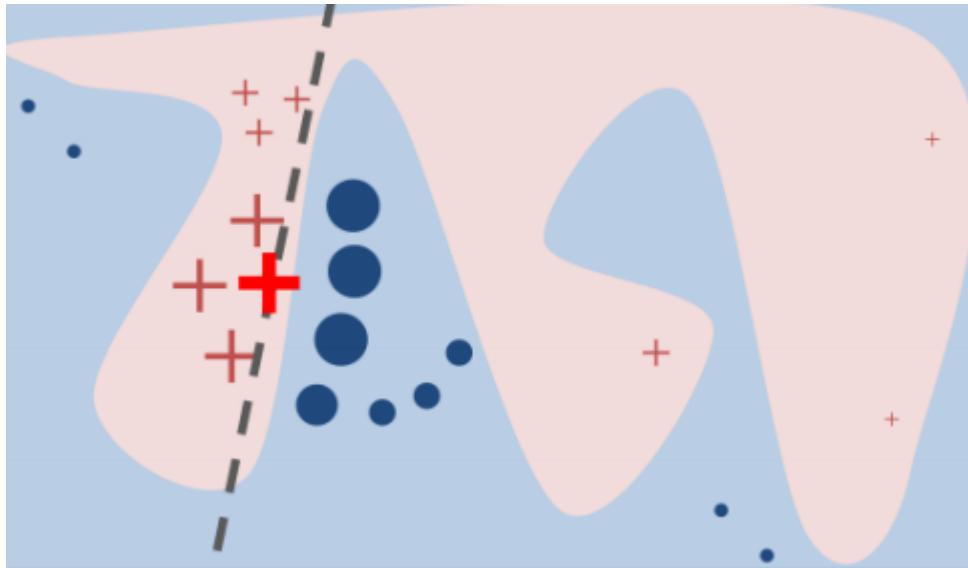
딥러닝 모델에서 “설명가능”한 특징을 학습한 노드를 찾아서 그 특징(Feature)에 ‘설명 라벨’을 붙이는 Feature Identification 방법 등이 있습니다.

두 번째로, 애초에 설명력있는 모델을 만드는 방법이 있습니다.

설명력 있는 모델의 예로는 의사결정나무나 선형회귀분석 모델 등이 있는데,

딥러닝 모델은 설명력이 부족한 인공신경망을 기반으로 하므로 이 방법을 적용할 수 없어 여기서는 따로 다루지 않겠습니다.

세 번째로는 인공신경망처럼 복잡한 블랙박스 모델의 일부분을 설명해 줄 수 있는 다른 모델을 활용하여 유추하는 방식입니다.



[그림 10. 블랙박스 모델(붉은색과 푸른색 칠해진 배경)과 설명가능한 모델(회색 점선)]

위 그림에서 연한 붉은색과 푸른색으로 칠해진 영역이 복잡한 인공신경망의 분류 결과라고 가정하겠습니다.

인공신경망은 복잡한 분류를 잘 수행했지만 설명력이 부족하기 때문에,

이 중 일부 영역의 데이터(+와 O)를 활용해 설명력이 좋은 모델을 별도로 하나 더 만들어 학습시킵니다.

그림상에서 추가로 학습한 설명력이 좋은 모델은 회색 점선에 해당합니다.

회색 점선 모델은 붉은색과 푸른색에 해당하는 전체 영역을 분류할 수는 없지만, 일부 하위 영역을 분류할 수는 있습니다.

그리고 매우 단순하기 때문에 사람이 결과를 해석하기도 쉽죠.

LIME이나 SP-LIME 등의 기술이 이에 해당합니다.

## 12.4 마무리

인간 또한 말로는 설명하기 어렵지만 '축'이나 '감'으로부터 결정을 먼저 내린 후 뒤이어 근거를 설명하는, 先결정 後 설명을 하는 경우가 많습니다.

축이 좋은 사람의 판단을 인정하기는 하겠지만, 그렇다고 누군가의 축만 믿고 큰 돈을 선뜻 투자하는 등의 일은 쉽지 않을 겁니다.

구체적이고 객관적인 설명을 통해 충분히 남을 납득시킬 수 있다면 당연히 모두 설득되어 동의할 수 있겠지요.

XAI는 인공지능에게 결과만이 아닌, 과정에 대한 해석력을 부여하는 것을 중요하게 생각하는 연구 분야입니다.



[그림 11. 설명할 수 없다면 제대로 이해한 것이 아니다.]

XAI가 중요한 만큼, 여러 기술들이 연구되고는 있으나 실질적으로 좋은 성능을 유지하면서도 사람이 납득할만한 뛰어난 설명을 제공하는 방식은 아직까지는 없는 듯 합니다.

현재의 수준은 여러 가능성을 시험해보는 초기 연구 단계에 불과하다고 말씀드릴 수 있겠네요.

하지만 어느 정도 딥러닝 기반 AI 모델의 성능이 상향 평준화된 지금, 민감한 산업 도메인으로의 AI 적용 또한 활발하게 검토되고 있기 때문에

곧이어 좋은 기법들이 연구되어 높은 성능과 해석가능성, 두 마리 토끼를 잡게 해주리라 기대합니다.

## 12.5 2020 AI 테크레터 연재를 마무리하며...

이렇게 올해 12편의 AI 테크레터가 마무리되었습니다.

어느새 우리의 삶에 부쩍 가까이 다가와 더 이상 미래의 이야기가 아닌, 당장 오늘의 큰 주제가 되는 인공지능이 단지 어려워 보인다거나 또 수학, 코딩을 잘 해야만 배울 수 있을 것 같다는 이유로 외면받는 것이 너무나 안타깝습니다.

그래서 문돌이였던 제가 삽질을 하며 느리게 조금씩 터득했던 지식을, 비전공자 독자를 가정하고 조금이라도 쉽고 재밌게 전달하기 위해 노력했습니다.

혹시 부족한 저의 지식이 잘못된 내용을 전달하는 것은 아닐까, 매 회 수많은 자료를 검색하고 참고하며 작성했습니다.

딥러닝은 오픈 사이언스로, 배우고자 하는 의지만 있다면 누구나 세계 유명 대학 교수의 강의자료, 테크 자이언트 기업의 연구 논문,

그것을 구현한 코드와 데이터 등 수많은 양질의 자료에 무료로 접근할 수 있습니다.

대부분의 사람, 어쩌면 모든 사람에게 영향을 미칠 인공지능이라는 것이 어느 특정 전공자 집단이나 연구 기관의 전유물이 되지 않기를 바랍니다.

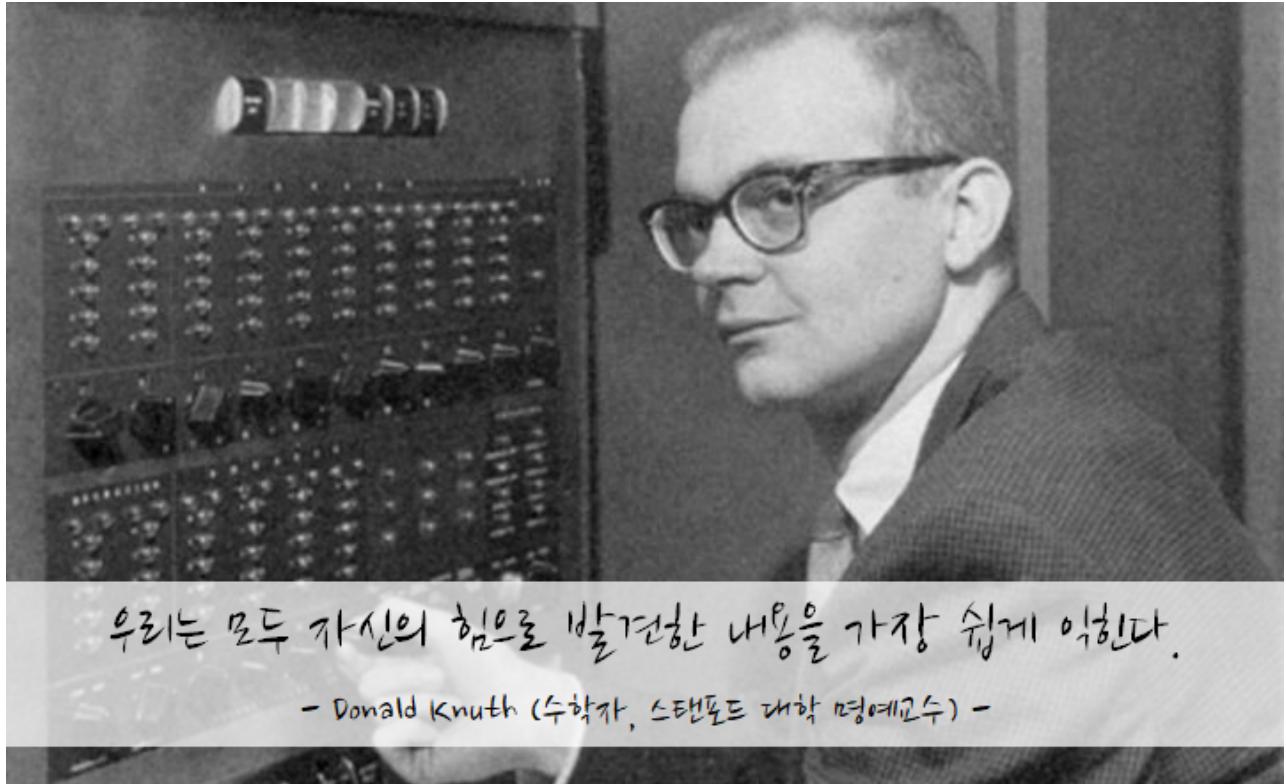
때문에 미흡한 컨텐츠임에도 불구하고 모든 임직원 여러분들께 메일을 쏴가며 읽어주시기를 호소했습니다.

부족한 부분이 있었다면 부디 너그러운 연말 직장인의 마음으로 이해해주시길 바랍니다.

AI 테크레터와 관련된 것, 또는 딥러닝과 관련된 어떠한 것이라도 질문이 생긴다면 언제든지 편하게 알려주세요.

AI 테크레터를 읽어주셔서 감사합니다.

모두 행복한 연말 보내시기 바랍니다 😊



## 참고자료

- Explainable Artificial Intelligence (XAI), David Gunning, DARPA/I2O, [https://www.cc.gatech.edu/~alanwags/DLAI2016/\(Gunning\)%20IJCAI-16%20DLAI%20WS.pdf](https://www.cc.gatech.edu/~alanwags/DLAI2016/(Gunning)%20IJCAI-16%20DLAI%20WS.pdf)
- Explainable Artificial Intelligence (XAI), Dr. Matt Turek, <https://www.darpa.mil/program/explainable-artificial-intelligence>
- 딥러닝 이후, AI 알고리즘 트렌드, ETRI Insight Report, 2018
- 설명 가능한 인공지능(explainable AI, XAI) 소개, 금융보안원 보안기술연구팀, 2018
- 위키피디아-방위 고등 연구 계획국, <https://ko.wikipedia.org/wiki/%EB%B0%A9%EC%9C%84%EA%B3%A0%EB%93%B1%EC%97%B0%EA%B5%AC%EA%B3%84%ED%9A%8D%EA%B5%AD>
- Class Activation Mapping and Class-specific Saliency Map, <http://cnnlocalization.csail.mit.edu/>
- Title: Top-down Visual Saliency Guided by Captions, V. Ramanishka, et al., 2017, <https://arxiv.org/abs/1612.07360>
- Learning Deep Features for Discriminative Localization, B. Zhou, et al., 2015, <https://arxiv.org/abs/1512.04150>
- AI와 최신 딥러닝 기술 동향, AI빅데이터연구소 이주열수석위원, 2019, [http://delab.cju.ac.kr/seminar/LG\\_ai.pdf](http://delab.cju.ac.kr/seminar/LG_ai.pdf)
- Generating Visual Explanations, L. Hendricks, et al., 2016, <https://arxiv.org/abs/1603.08507>

- "Why Should I Trust You?": Explaining the Predictions of Any Classifier, M. Ribeiro, et al., 2016, <https://arxiv.org/abs/1602.04938>