LRP Tutorial

조소희 @ 설명가능인공지능 연구센터
kirarenctaon@gmail.com

실습 목표

설명가능인공지능 기술을 구현한 **파이썬** 코드를
이미지 데이터와 자연어 데이터에 실행하여,
향후 비슷한 **실세계 데이터**에 응용할 수 있도록 **연습하는 시간**

- **다루는 내용:**
  - LRP기술에 대한 직관적인 설명
  - 코드의 전반적인 흐름

- **다루지 않는 내용**
  -기술을 구현한 상세한 수식의 의미
  - CNN, LSTM모델에 대한 상세한 설명
  - 자연어 데이터 전처리에 대한 상세한 설명

L

R

P

Layer-Wise 레이어 단위

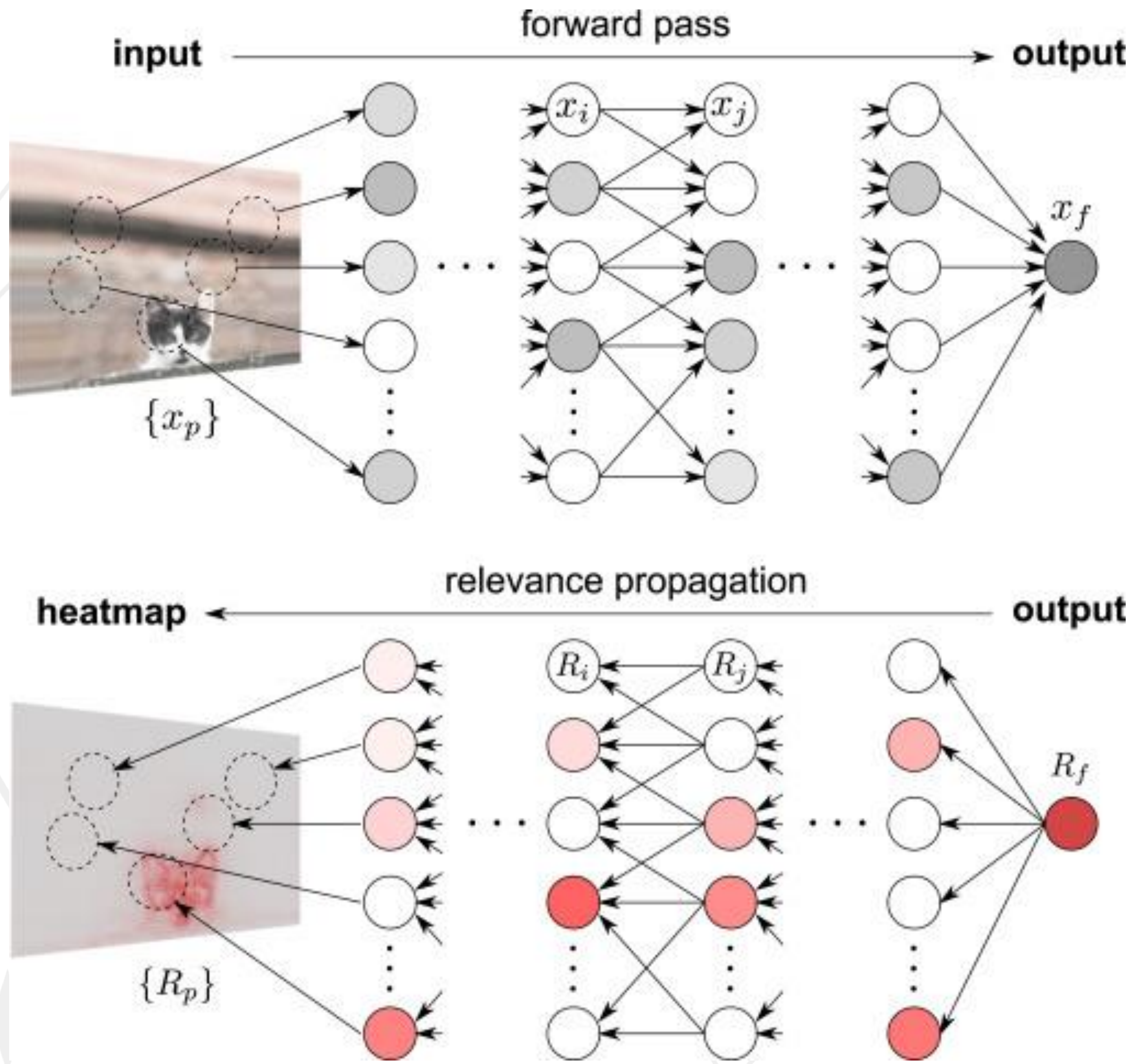
Relevance 관련성

Propagation 전파

Layer-Wise　레이어 단위로

Relevance　결과에 영향을 주는 관련성을 구하는

Propagation　역전파 기술

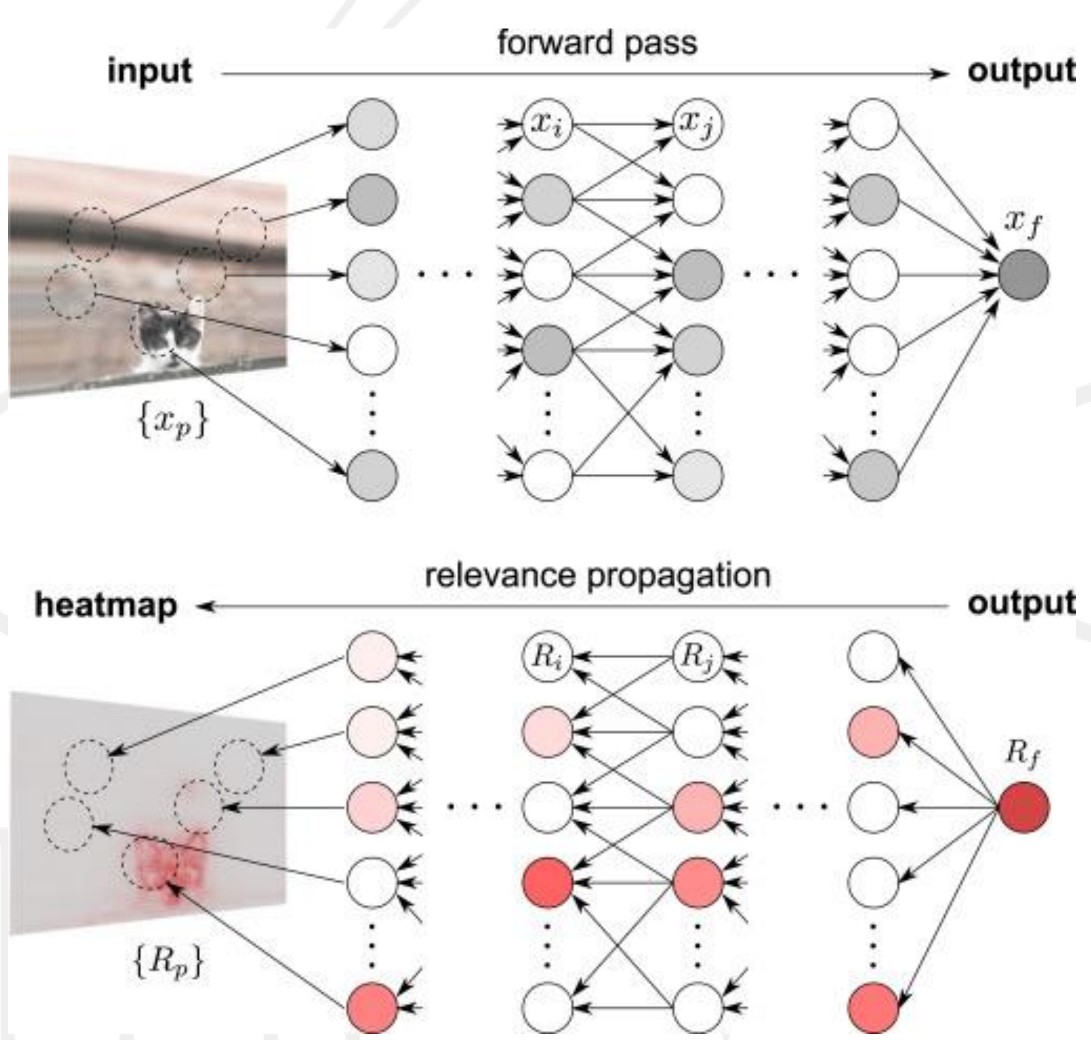


입력 데이터 관점에서 **분류 결과** 뿐만 아니라
결정에 영향을 미치는 구조를 설명

[Alexander Binder et al, Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers, ICANN, 2016]

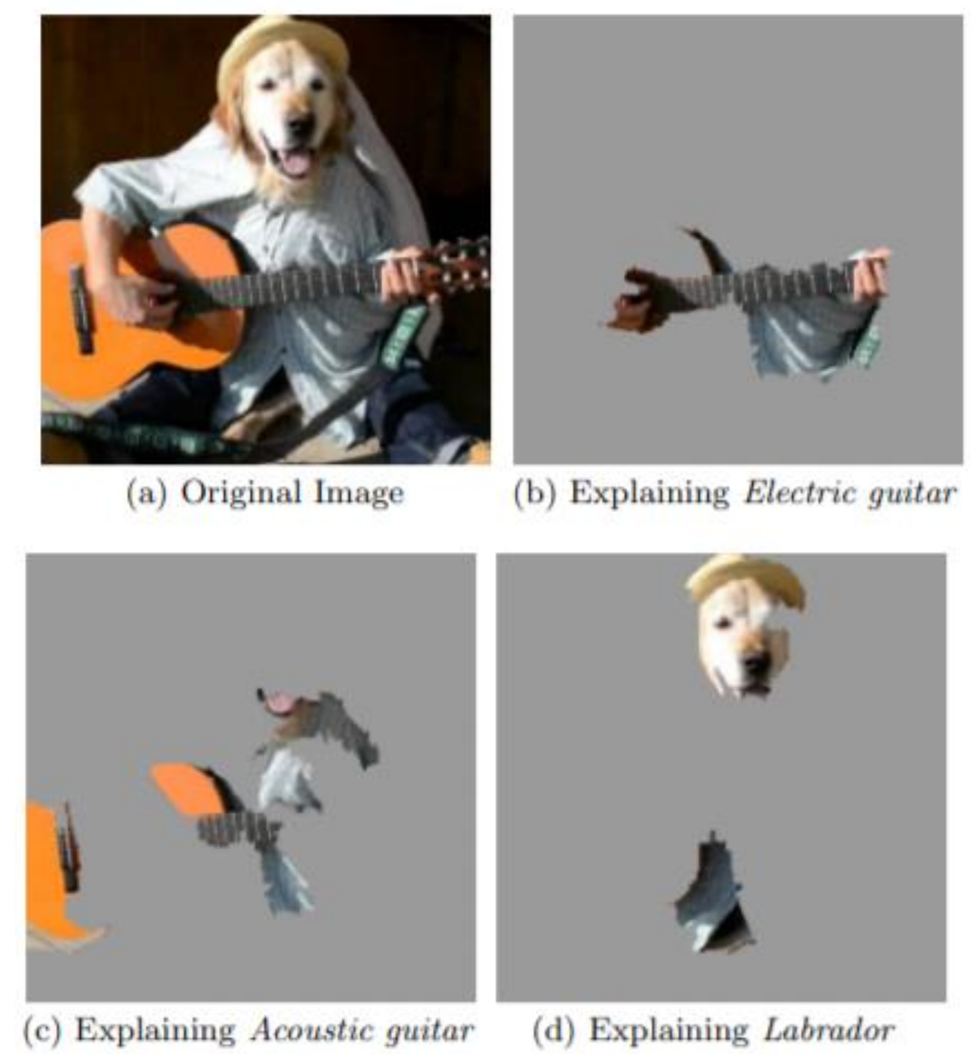
관련 기술들과 비교하면..

LRP



LIME

(Local Interpretable Model-agnostic Explanation)



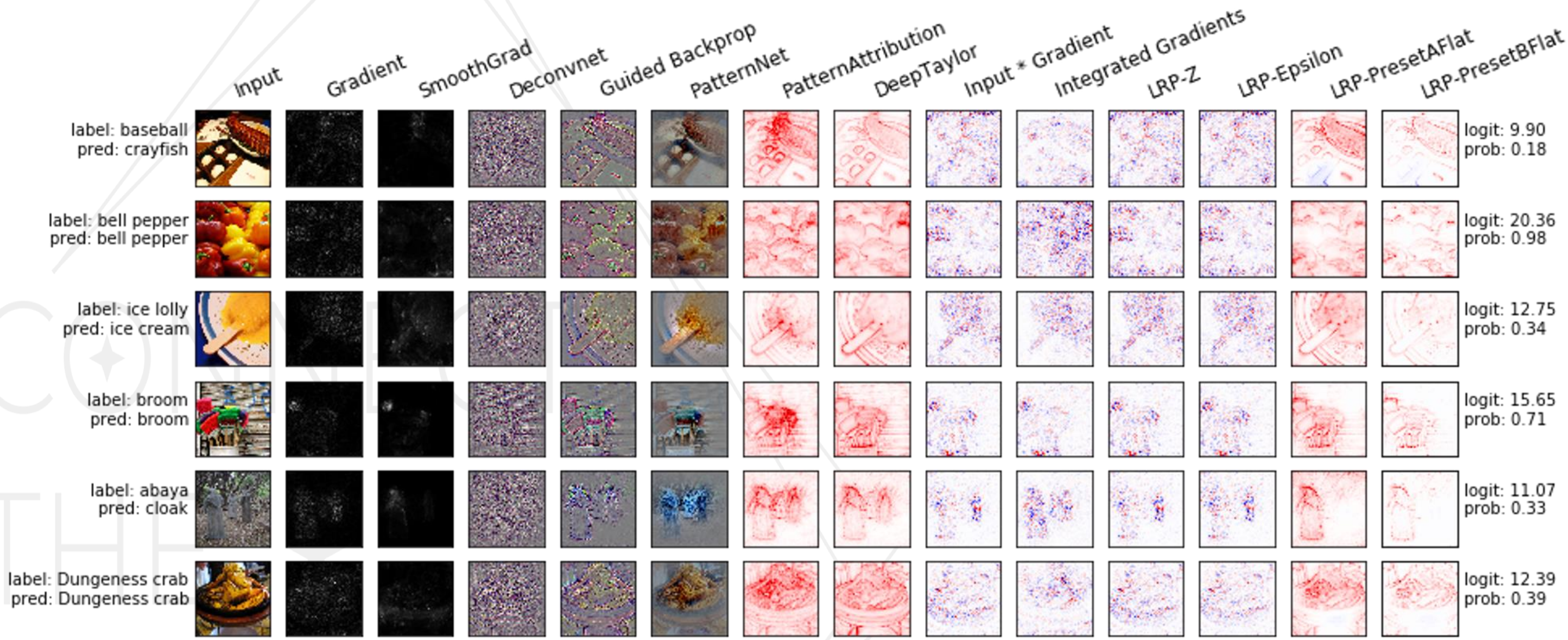
CAM

(Class Activation Mapping)

[Why Should I Trust You?: Explaining the Predictions of Any Classifier, KDD, 2016]
[Bolei Zhou et al, Learning Deep Features for Discriminative Localization, CVPR, 2016]

http://xai.unist.ac.kr/opensource/relatedproject/

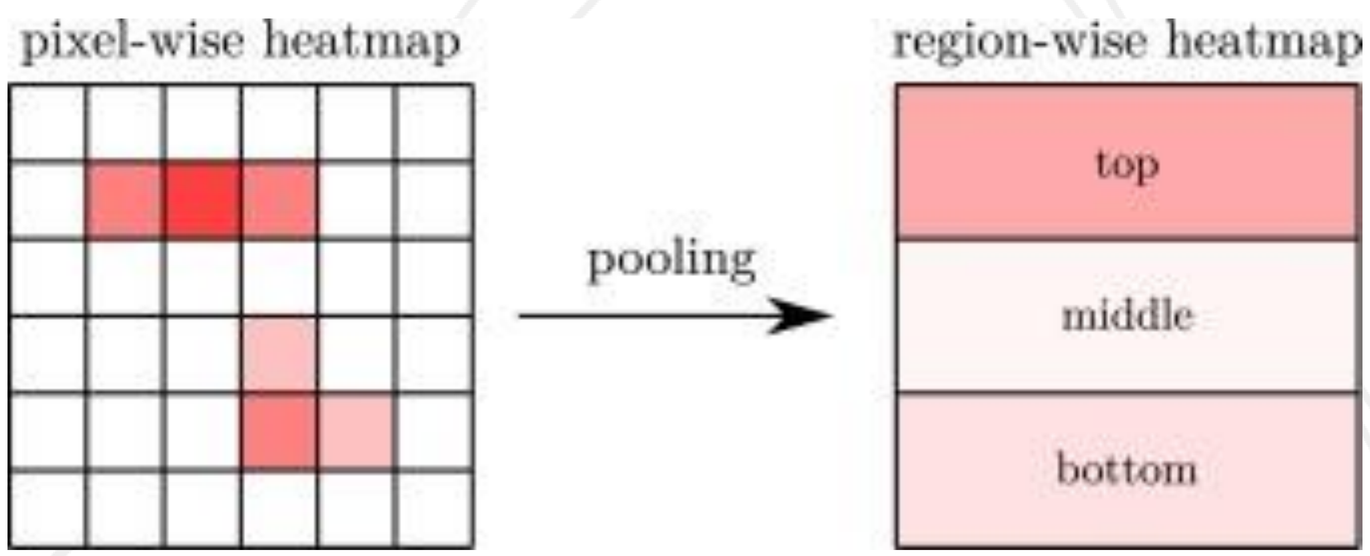
LRP 연구팀에서 발전시키고 있는 기술들과 비교하면..



https://github.com/albermax/innvestigate

오늘 실습에서 구현할 SA, LRP만 한번 더 확인

Sensitivity Anlysis

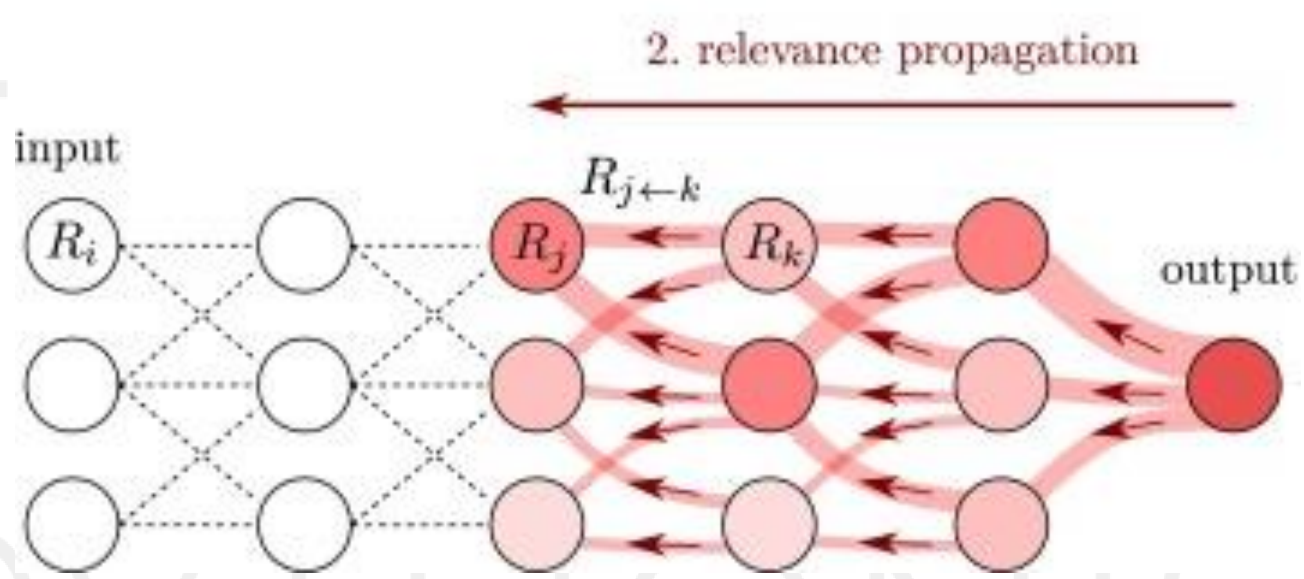


타켓클래스에 대한 예측함수

$$R_d = \left(\frac{\partial \boxed{f_c}}{\partial \boxed{x_d}}(\boldsymbol{x}) \right)^2$$

입력값

Layer-wise Relevance Propagation (Simplified)



전단계 뉴런의 활성화 함수 a와 가중치w의 곱

$$R_j = \sum_k \frac{\boxed{a_j w_{jk}^+}}{\boxed{\sum_j a_j w_{jk}^+}} \boxed{R_k}$$

전단계에서 구한 관련성

출력에서 전단계까지의 모든 뉴런의 활성화 함수 a와 가중치w의 곱

https://www.sciencedirect.com/science/article/pii/S1051200417302385

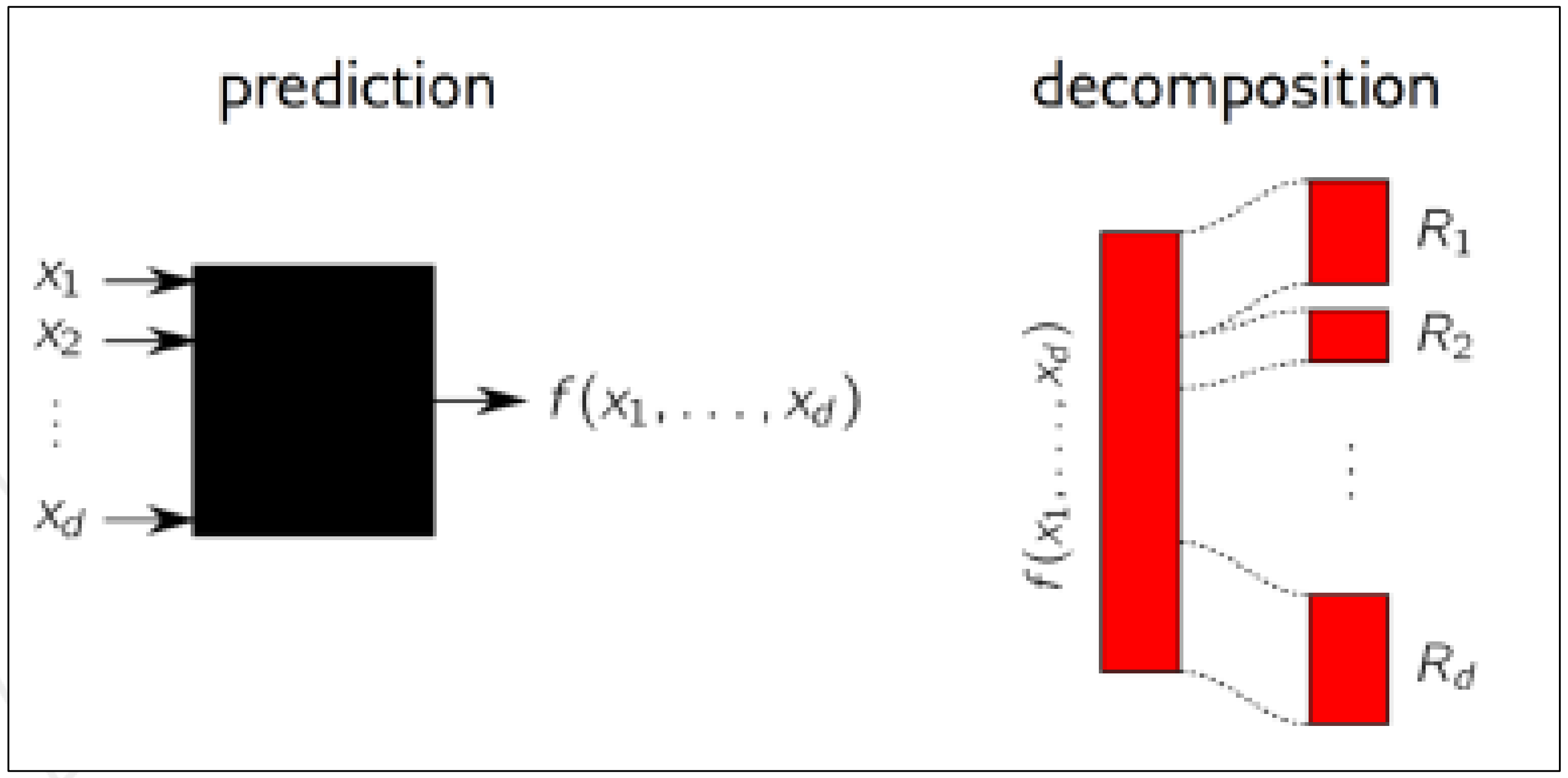
Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition

- Related work

Pixel-wise decomposition of a function

- explaining a neural network decision **by decomposing of function value** onto **input variables** in an amount that matches the respective **relevance of these input variables to the function value**.

- **Notation**
  - $x = \{x_p\}; a\ set\ of\ pixel\ values$
  - $f: \mathbb{R}^d \rightarrow \mathbb{R}^+$; $positive - valued\ function$
    - $f(x) = 0$: $absence\ of\ certain\ type\ of\ objects$
    - $f(x) = 1$: $presence\ of\ certain\ type\ of\ objects$
  - $R_P(x)$: $relevance\ score\ of\ x_p$
  - $\mathrm{R}(x) = \{R_P(x)\}$: heatmap



Pixel-wise decomposition

Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition

- Related work

Pixel-wise decomposition of a function

- **A heatmapping should satisfy properties that we define below:**

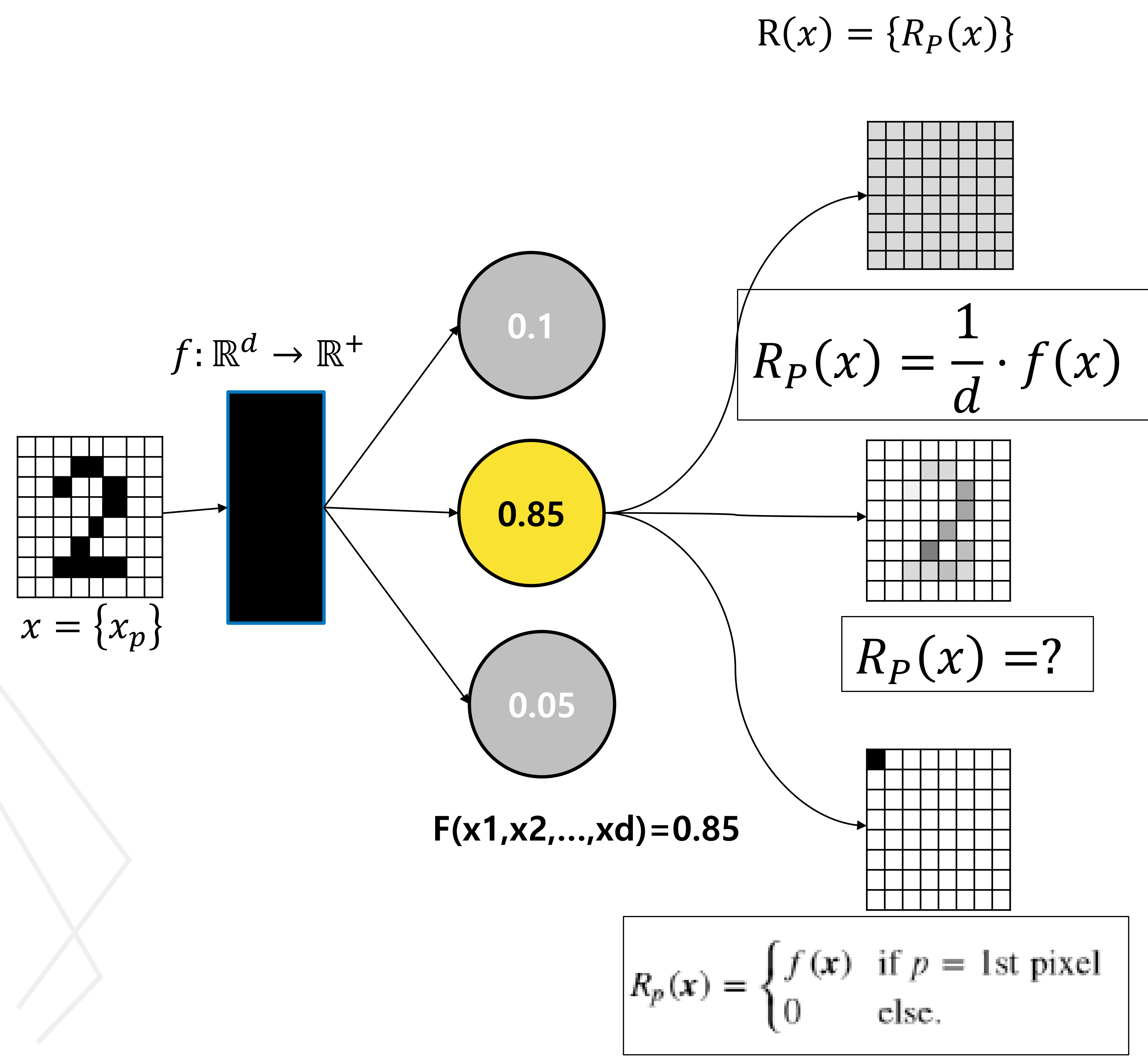
Definition 1. A heatmapping $\boldsymbol{R}(\boldsymbol{x})$ is *conservative* if the sum of assigned relevances in the pixel space corresponds to the total relevance detected by the model:

$$\forall\, \boldsymbol{x}: f(\boldsymbol{x}) = \sum_p R_p(\boldsymbol{x}).$$

Definition 2. A heatmapping $\boldsymbol{R}(\boldsymbol{x})$ is *positive* if all values forming the heatmap are greater or equal to zero, that is:

$$\forall\, \boldsymbol{x}, p: R_p(\boldsymbol{x}) \geq 0$$

Definition 3. A heatmapping $\boldsymbol{R}(\boldsymbol{x})$ is *consistent* if it is conservative *and* positive. That is, it is consistent if it complies with Definitions 1 and 2.



In practice, it is not possible to specify explicitly all properties that a heatmapping technique should satisfy.

Ginkyeng LEE Sailab

Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition

- Related work

Pixel-wise decomposition of a function

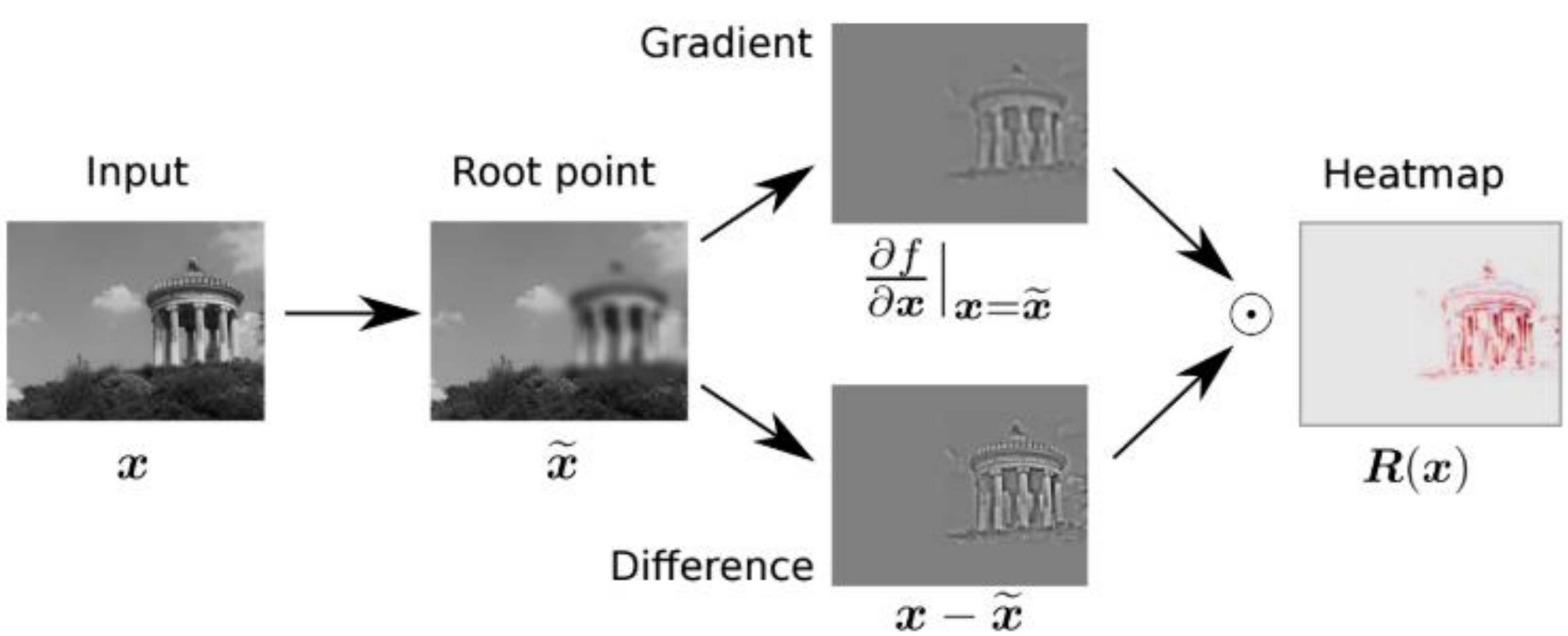
2 meaningful examples of decompositions that comply with the definition mentioned.

(2) Taylor decomposition

- **General case of arbitrary differentiable functions f(x)**
- **Decomposition method based on the Taylor expansion of the function** at some well-chosen root point $\tilde{x}$
- **The first-order Taylor expansion of f(x) is given by**

$$f(x) = f(\tilde{x}) + \left(\frac{\partial f}{\partial x}\Big|_{x=\tilde{x}}\right)^{T} \cdot (x - \tilde{x}) + \varepsilon = 0 + \sum_{p} \underbrace{\frac{\partial f}{\partial x_p}\Big|_{x=\tilde{x}} \cdot (x_p - \tilde{x}_p)}_{R_p(x)} + \varepsilon, \quad \Rightarrow \quad R(x) = \frac{\partial f}{\partial x}\Big|_{x=\tilde{x}} \odot (x - \tilde{x}).$$

$f(\tilde{x})=0$

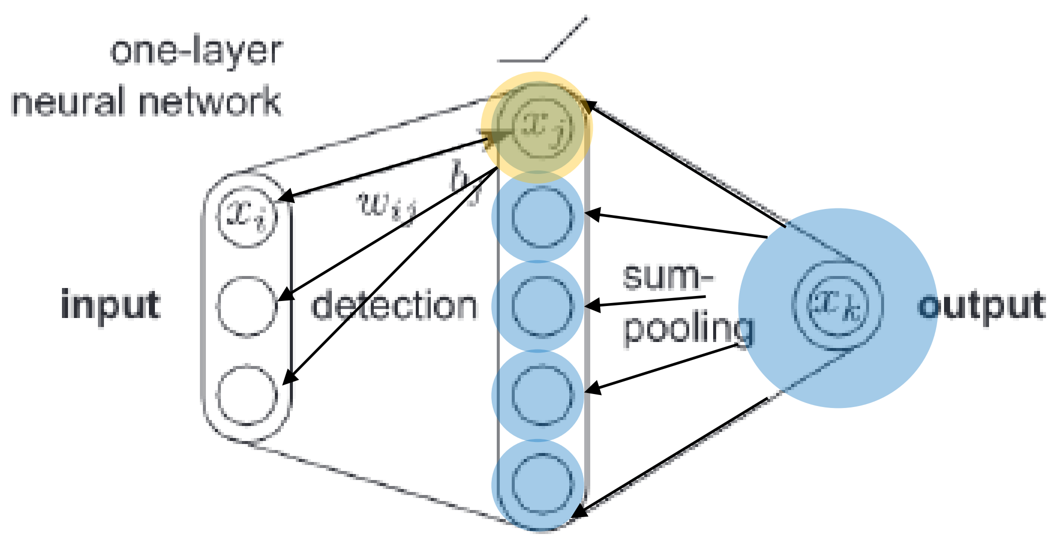


- Possibly NOT consistent!
  - Satisfying Definition 2.
  - Possible presence of non-zero higher order terms → not satisfying Definition1

Ginkyeng LEE Sailab

Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition

- Application to one-layer networks



$$x_j = \max\left(0, \sum_i x_i w_{ij} + b_j\right) \quad \text{and} \quad x_k = \sum_j x_j$$

1.Express the relevance for the top layer in terms of lower-layer neurons as: (mapping function)

$$R_k = \sum_j x_j$$

2. Redistribute R_k onto neurons $\{x_j\}$ using single Taylor

$$R_j = \frac{\partial R_k}{\partial x_j}\Big|_{\{\tilde{x}_j\}} \cdot (x_j - \tilde{x}_j).$$

We need to choose root points of this function.

$$\sum_j \tilde{x}_j = 0$$

- The root points should be admissible.
  - the root point must be positive.

➔ The only point that is both (root) + (admissible) is $\{\tilde{x}_j\} = 0$

$$R_j = x_j$$

1.Express R_j in terms of $\{x_i\}$ (mapping function)

$$R_j = \max\left(0, \sum_i x_i w_{ij} + b_j\right),$$

2. Redistribute R_j onto neurons $\{x_i\}$ using single Taylor decomposition.

$$R_i = \sum_j \frac{\partial R_j}{\partial x_i}\Big|_{\{\tilde{x}_i\}^{(j)}} \cdot (x_i - \tilde{x}_i^{(j)}).$$

How to choose root point?

Various methods for choosing a root point
That consider the diversity of possible input domains

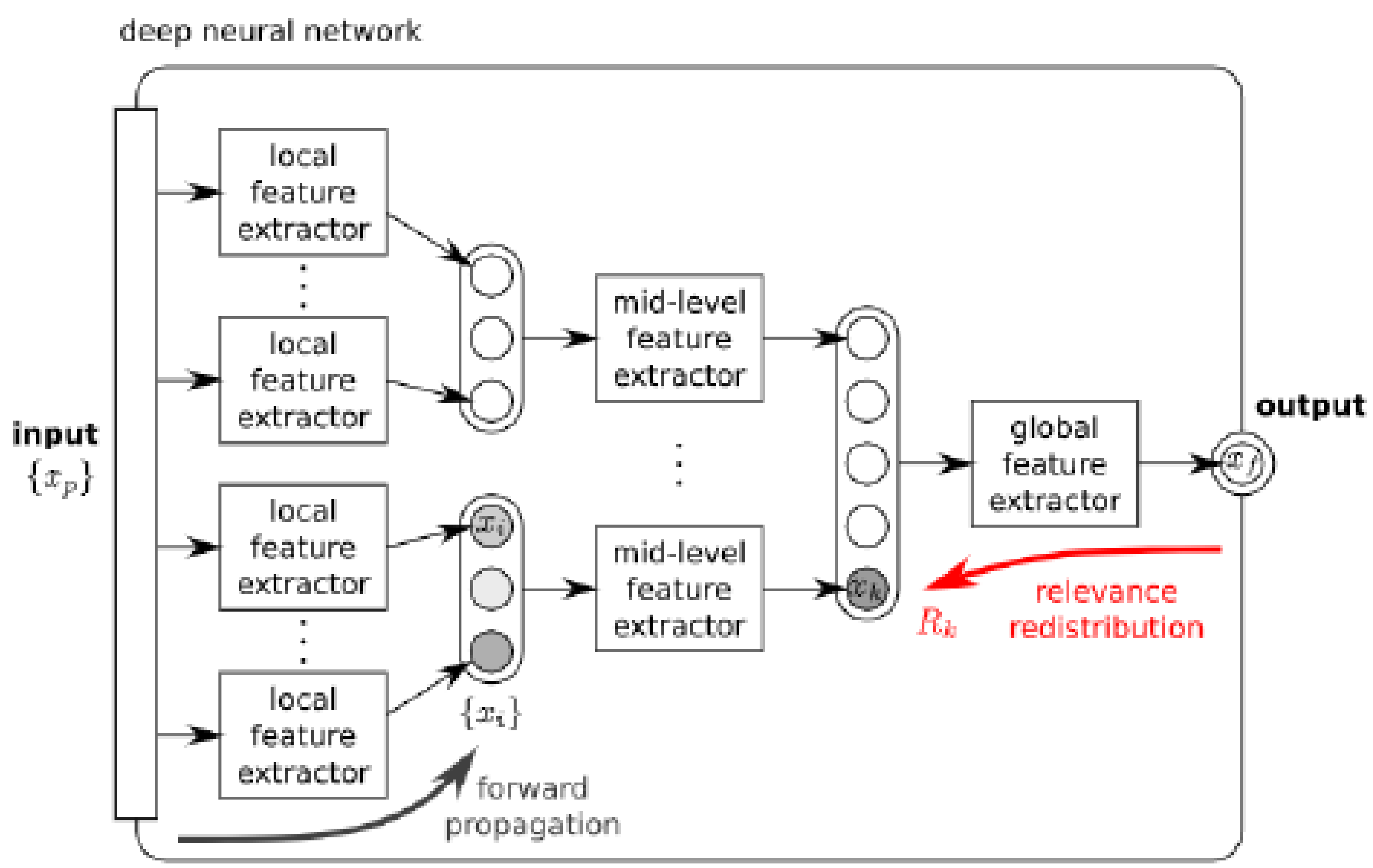
Ginkyeng LEE Sailab

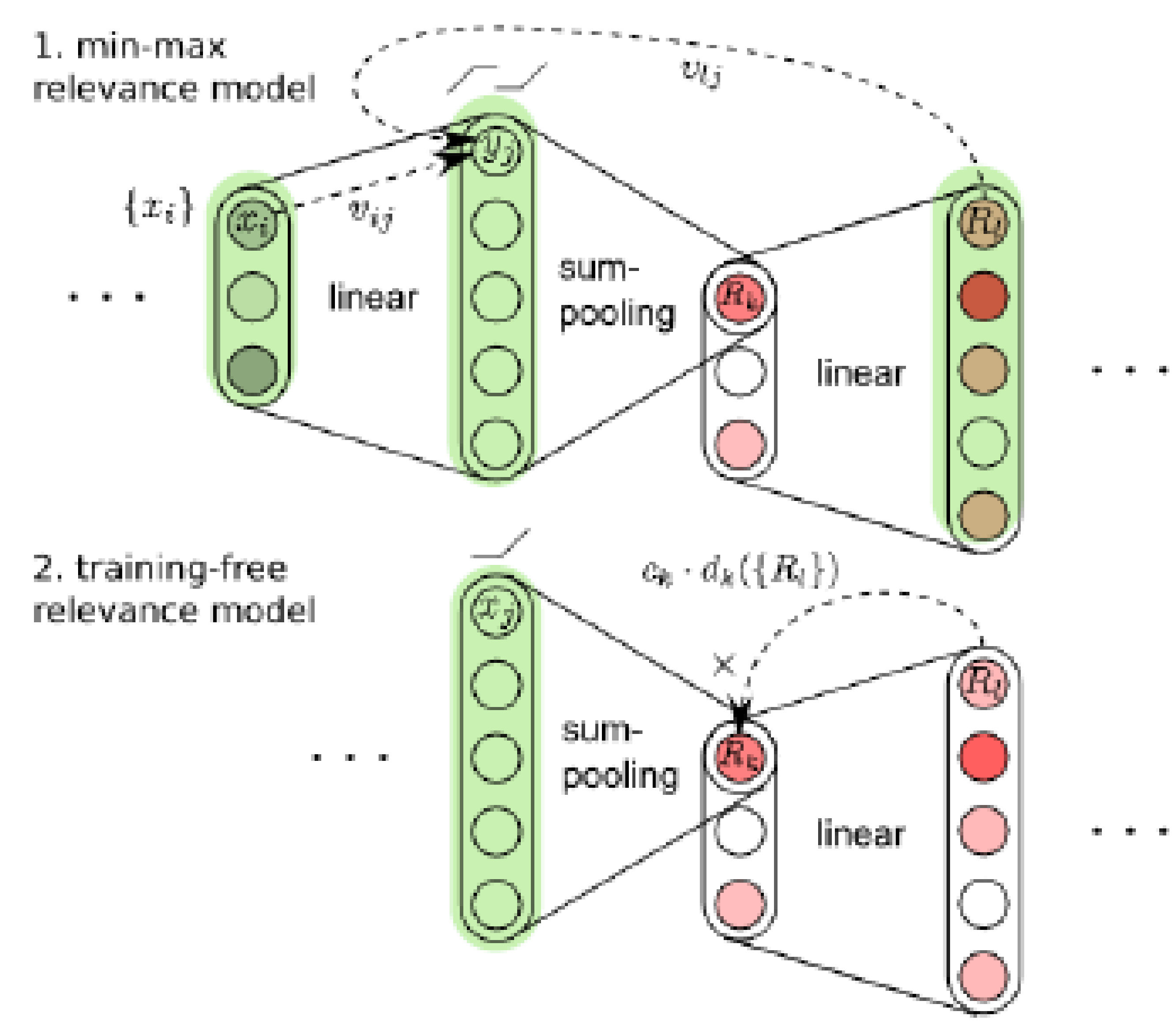
Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition
- Application to deep networks

The mapping may be unknown. In order to redistribute the relevance from higher to lower layers, One needs to make this mapping explicit. For this purpose, the paper introduce the concept of relevance model.

- Relevance model : function that maps a set of neuron activations at a given layer to the relevance of a neuron in a higher layer.

+ Upper-layer relevance is not only determined by input neuron activations , but also by high-level information that have been formed in the top layers of the network.

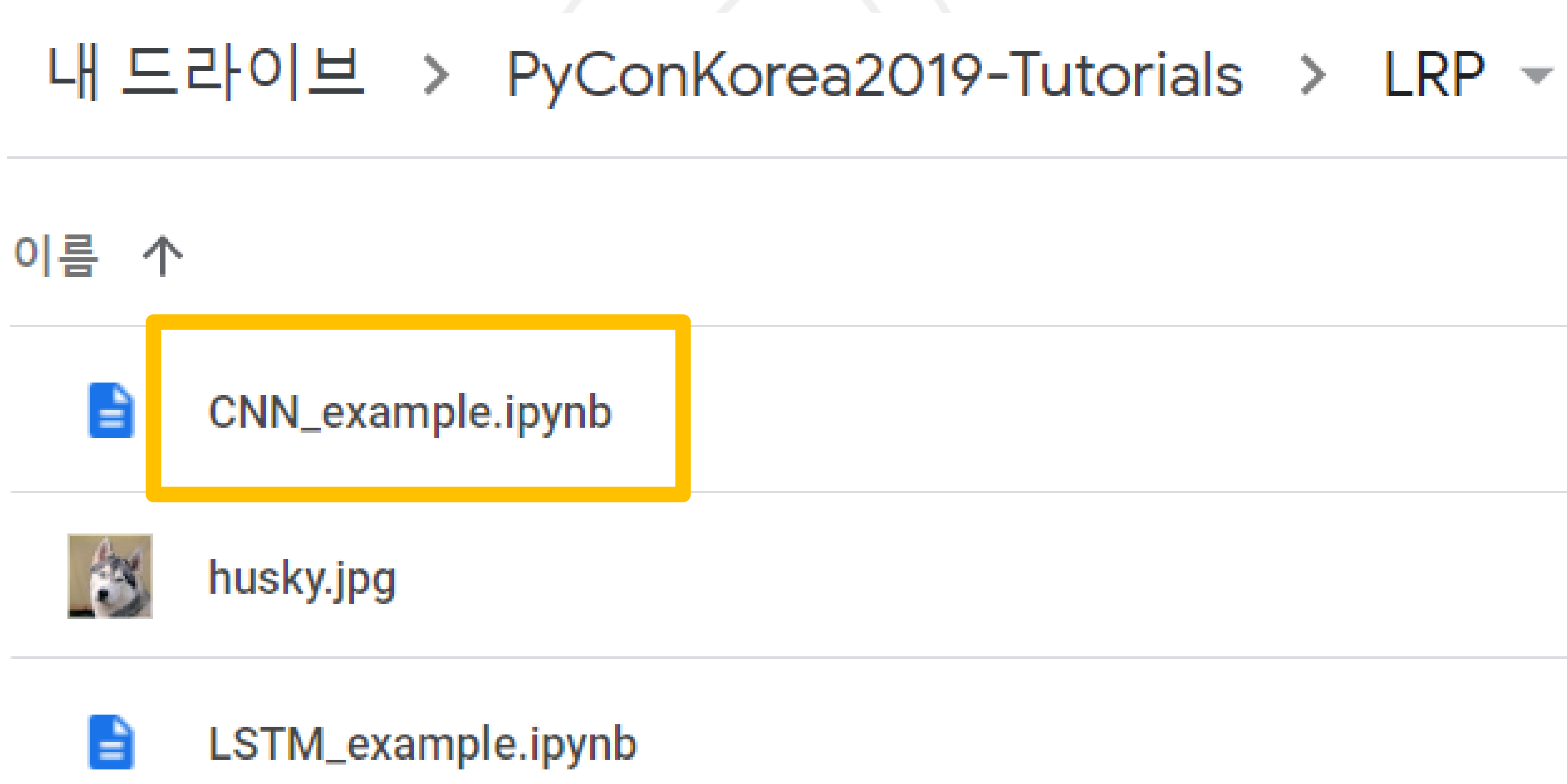


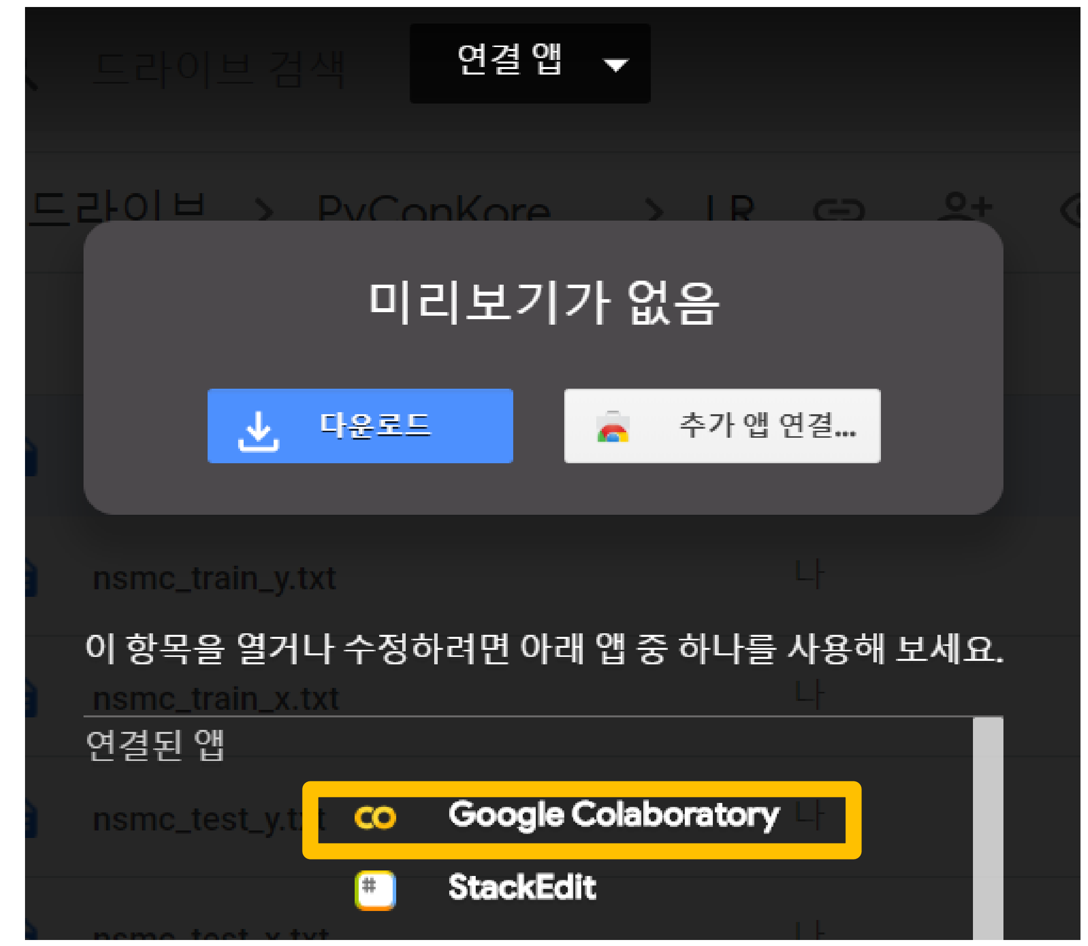


Ginkyeng LEE Sailab

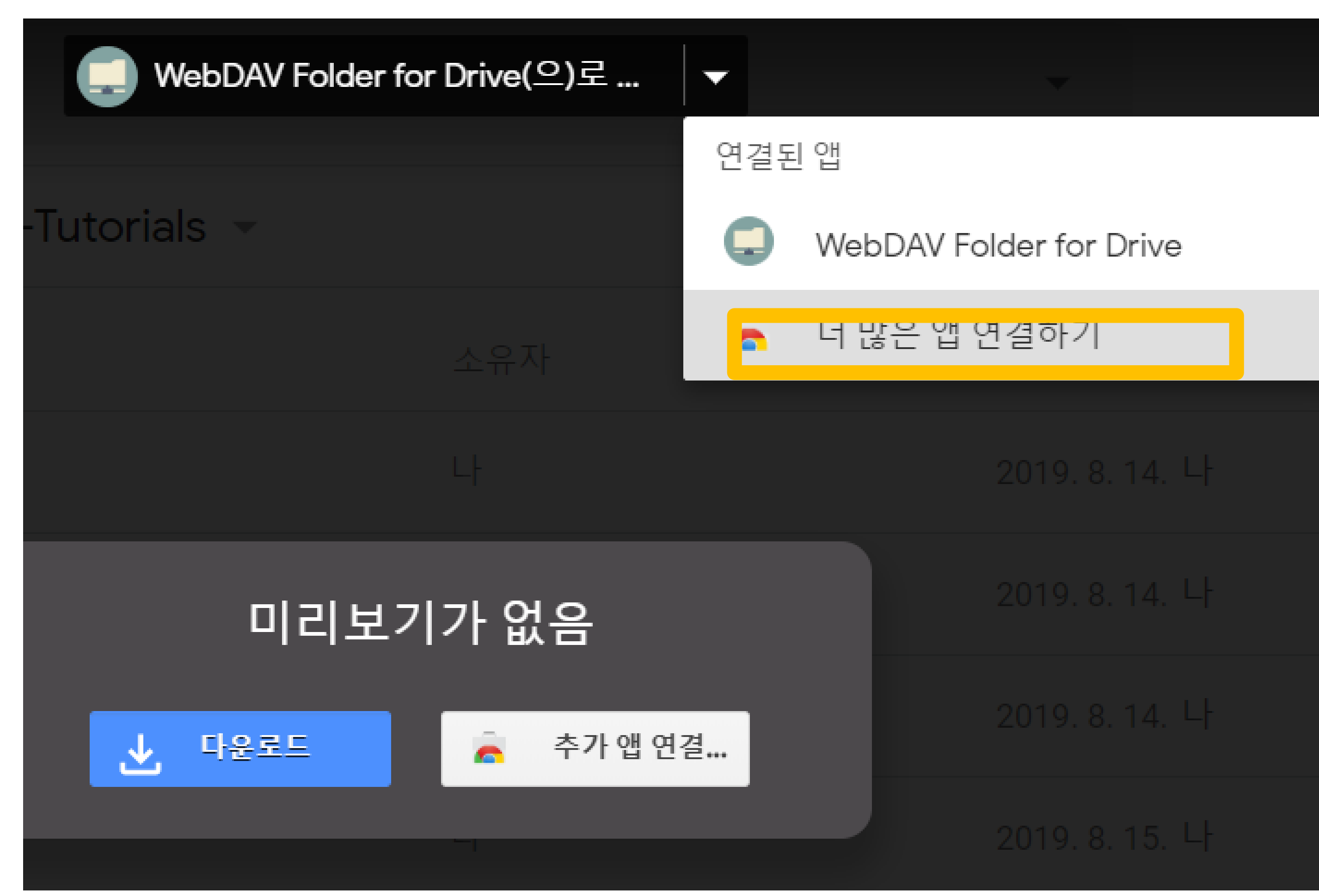
LRP for CNN

구글 드라이브에서 코랩으로 CNN_example.ipynb파일 연결





만약 '연결 앱'에 코랩이 없다면 '더 많은 앱 연결하기'를 이용



관련 내용을 좀더 자세하게 알고 싶다면

1. Innvestigate의 전체 코드와 다양한 예제 있는 공식 저장소
   https://github.com/albermax/innvestigate

2. LRP 연구팀이 튜토리얼, 논문 등을 업로드하는 공식 홈페이지
   http://heatmapping.org/

3. 김범수님이 한줄한줄 설명하며 텐서플로우로 직접 구현한 저장소
   https://github.com/1202kbs/Understanding-NN

를 추천드립니다.

LRP for LSTM

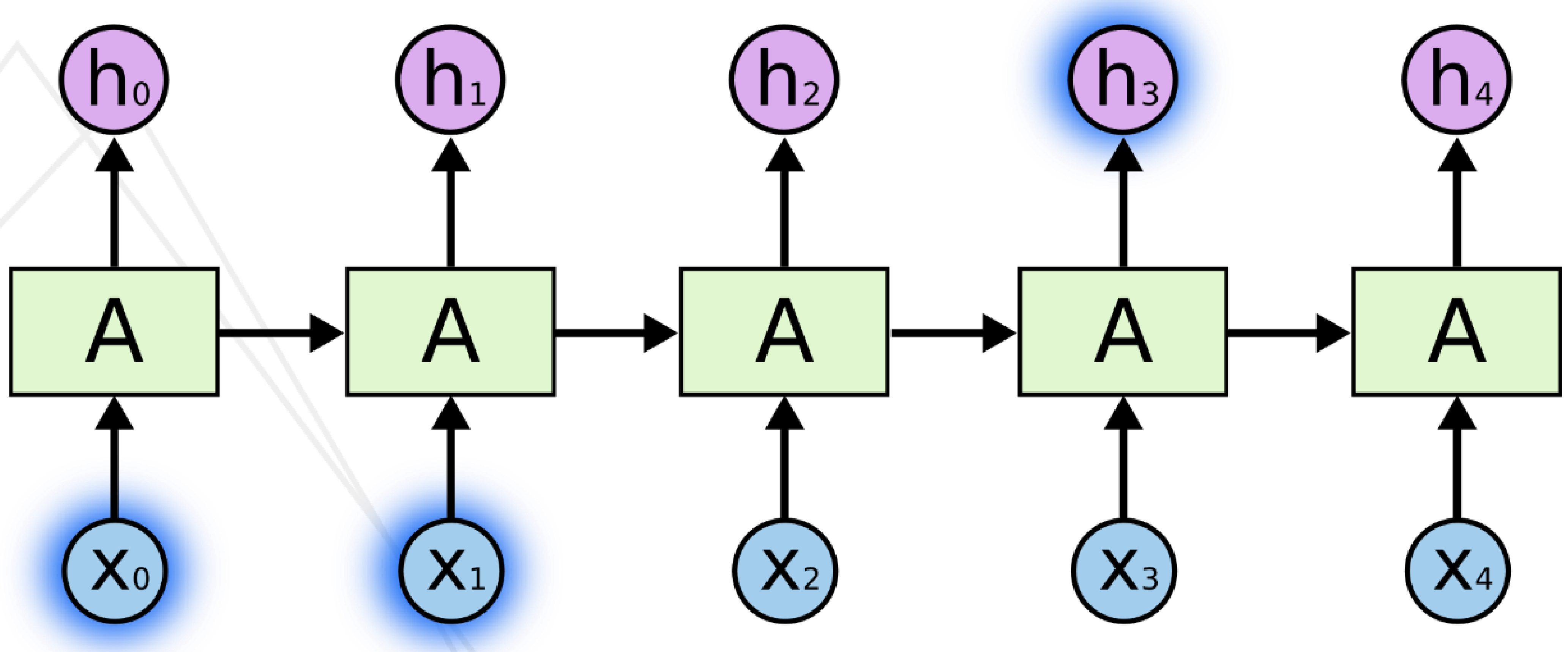
L S T M

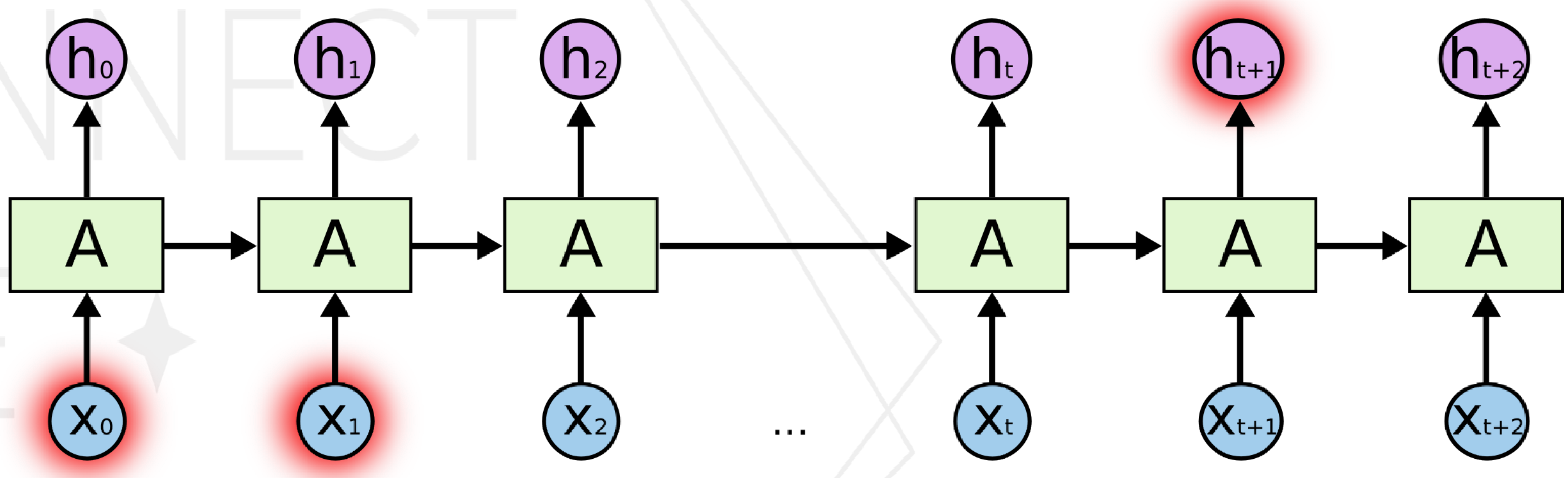
Long 장기적

Short 단기적

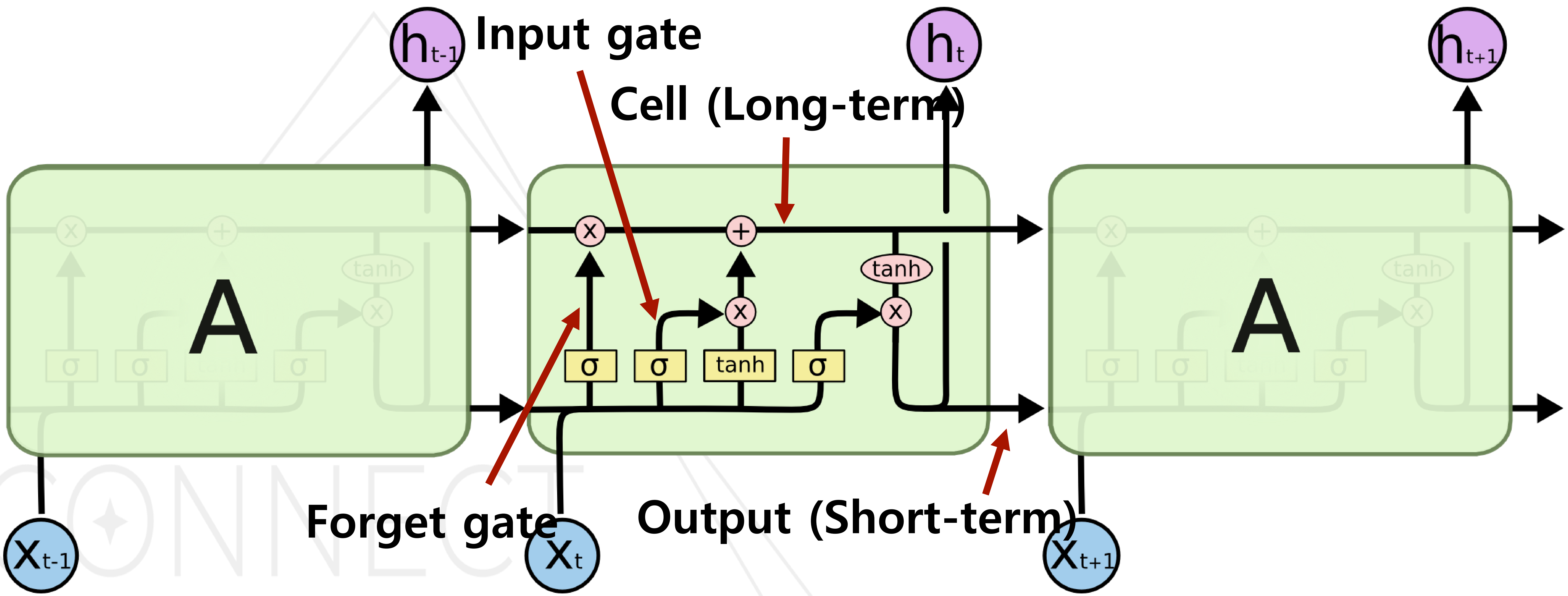
Term

Meromry 기억을 구분하는 순환신경망



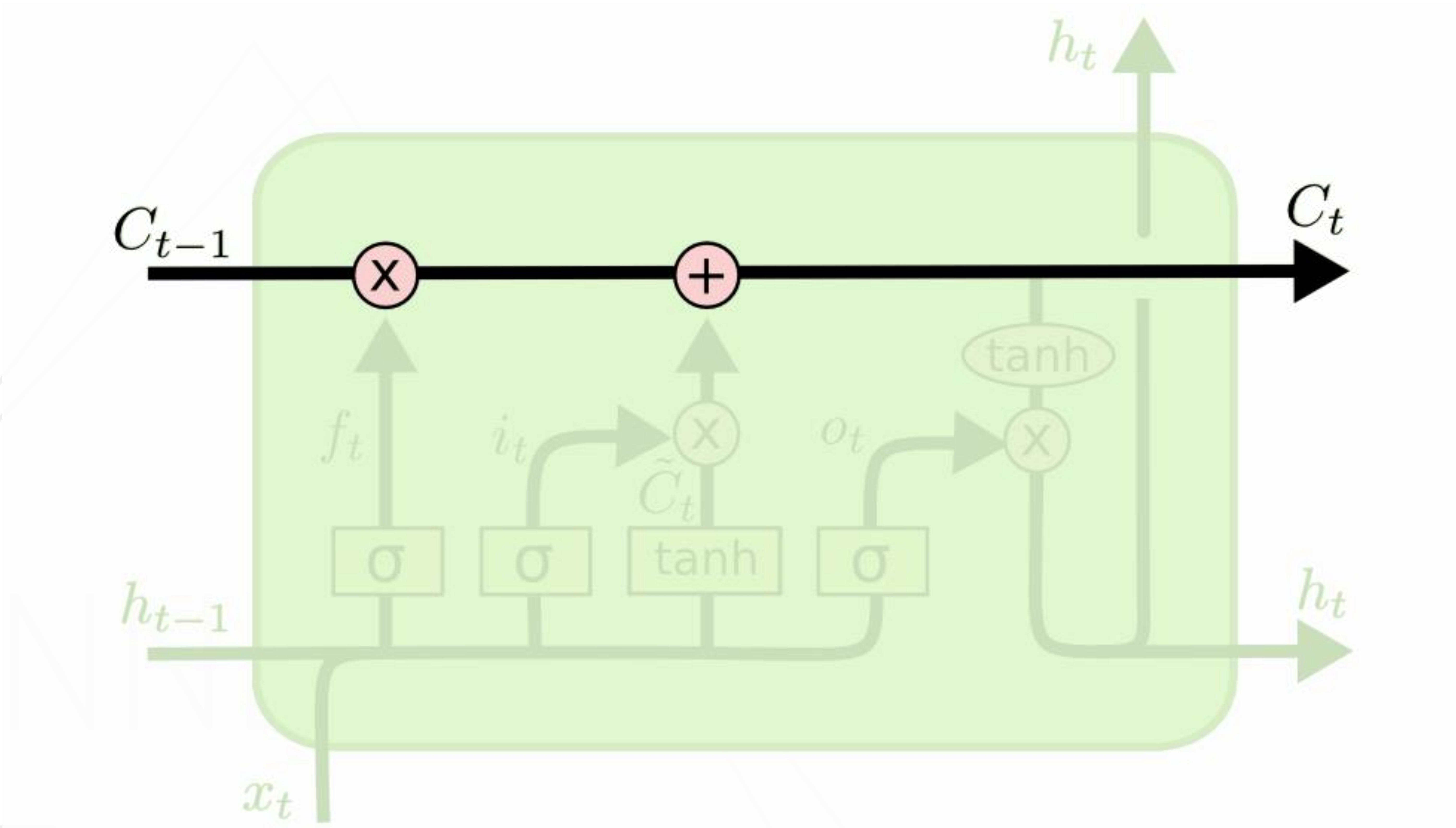


http://colah.github.io/posts/2015-08-Understanding-LSTMs/



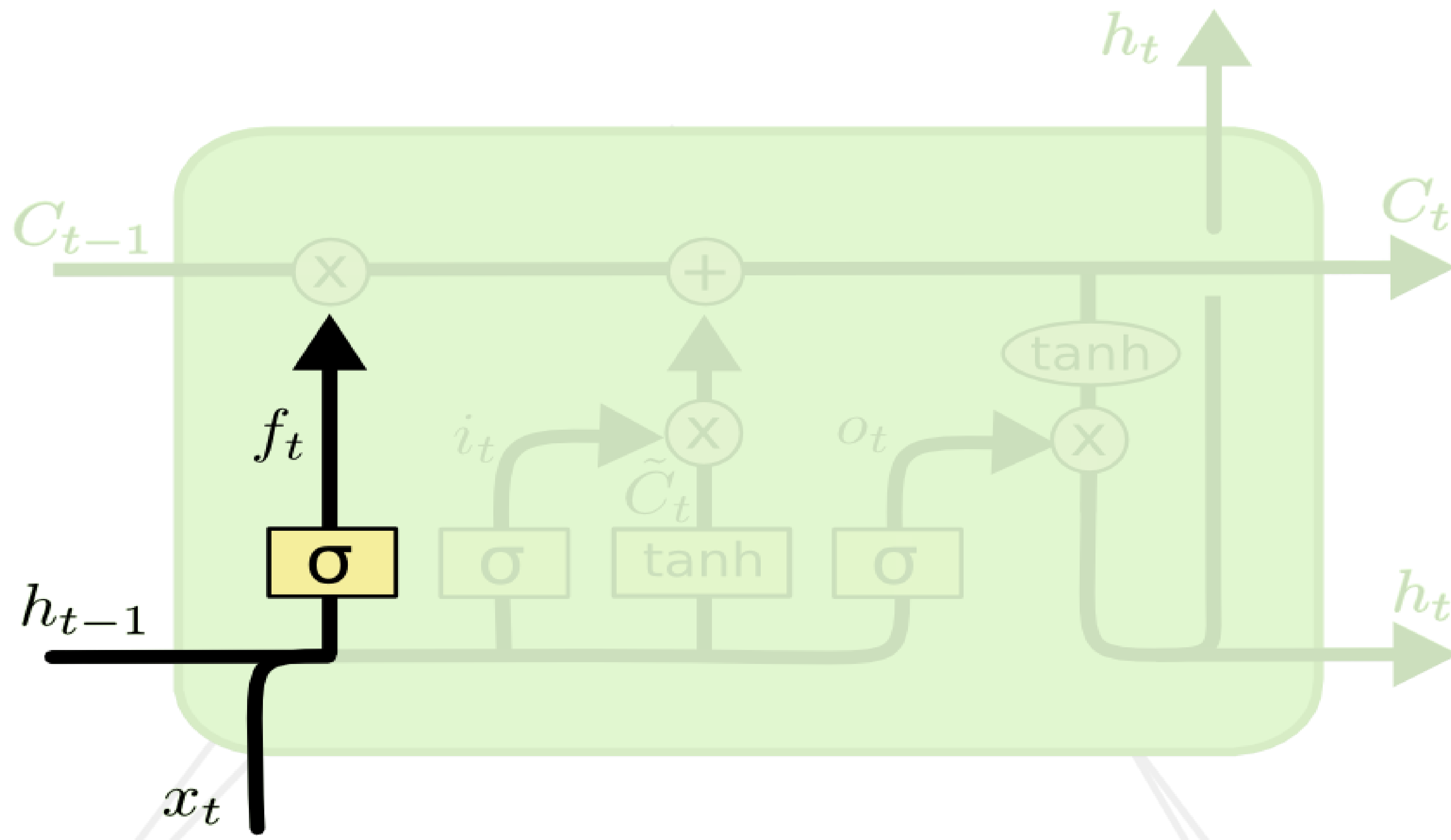
오랫동안 기억할 **Cell State**와
짧게 기억하고 잊을 **Hidden State**를 구분

http://colah.github.io/posts/2015-08-Understanding-LSTMs/



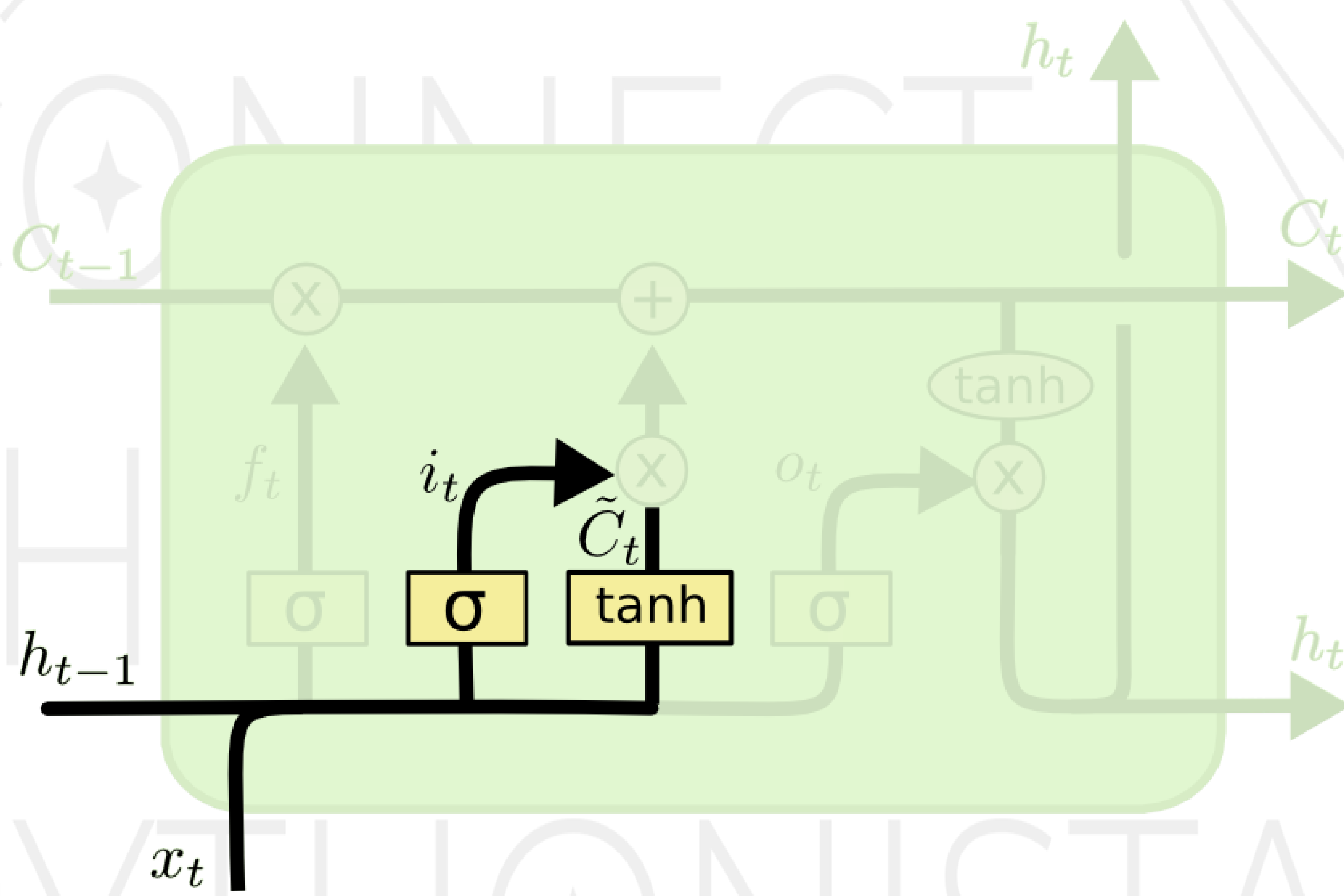
장기 기억이 가능하게 하는 **Cell State**가 **LSTM**의 핵심

http://colah.github.io/posts/2015-08-Understanding-LSTMs/



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

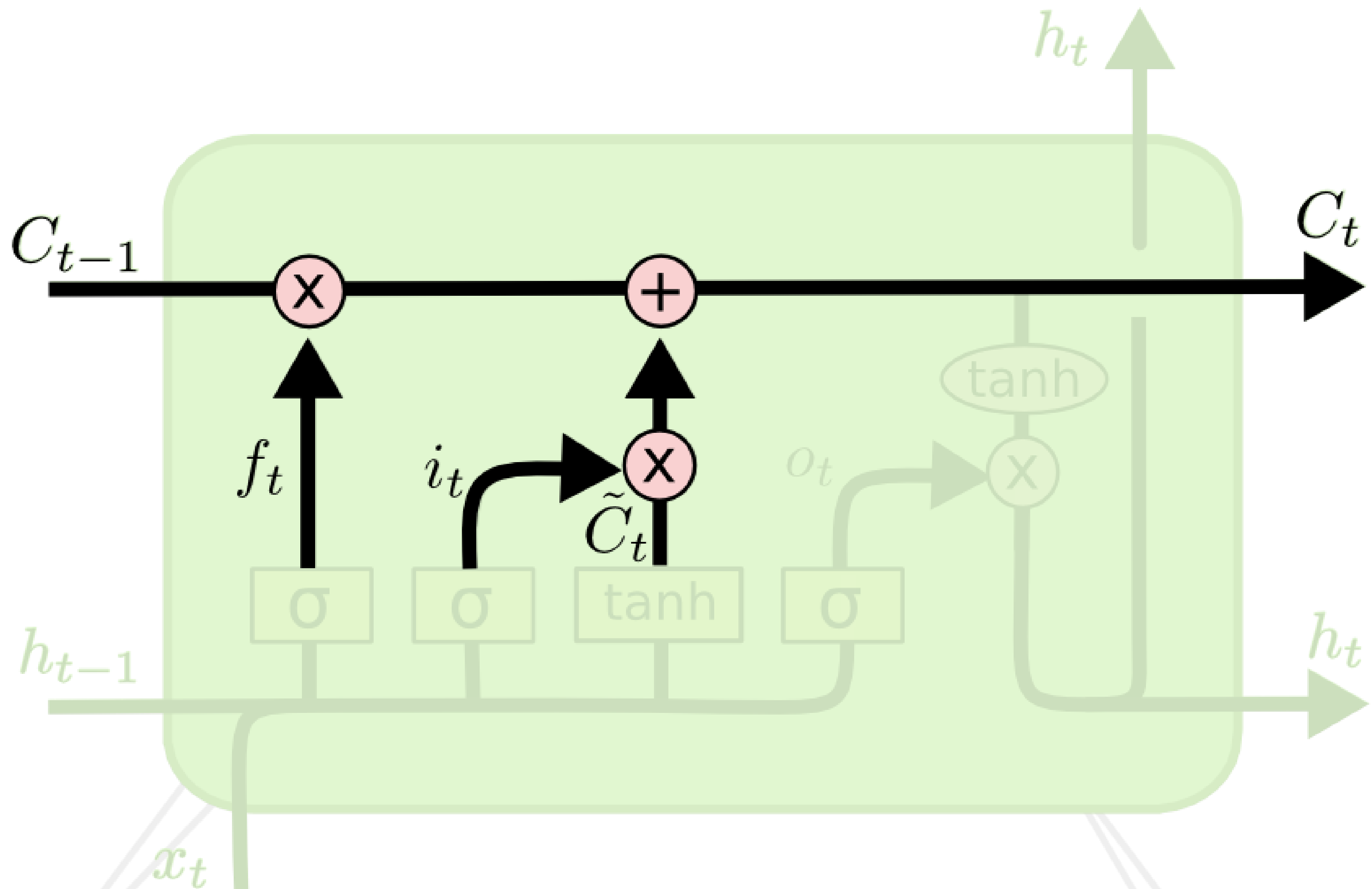
잊을 정보 f를 정함



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

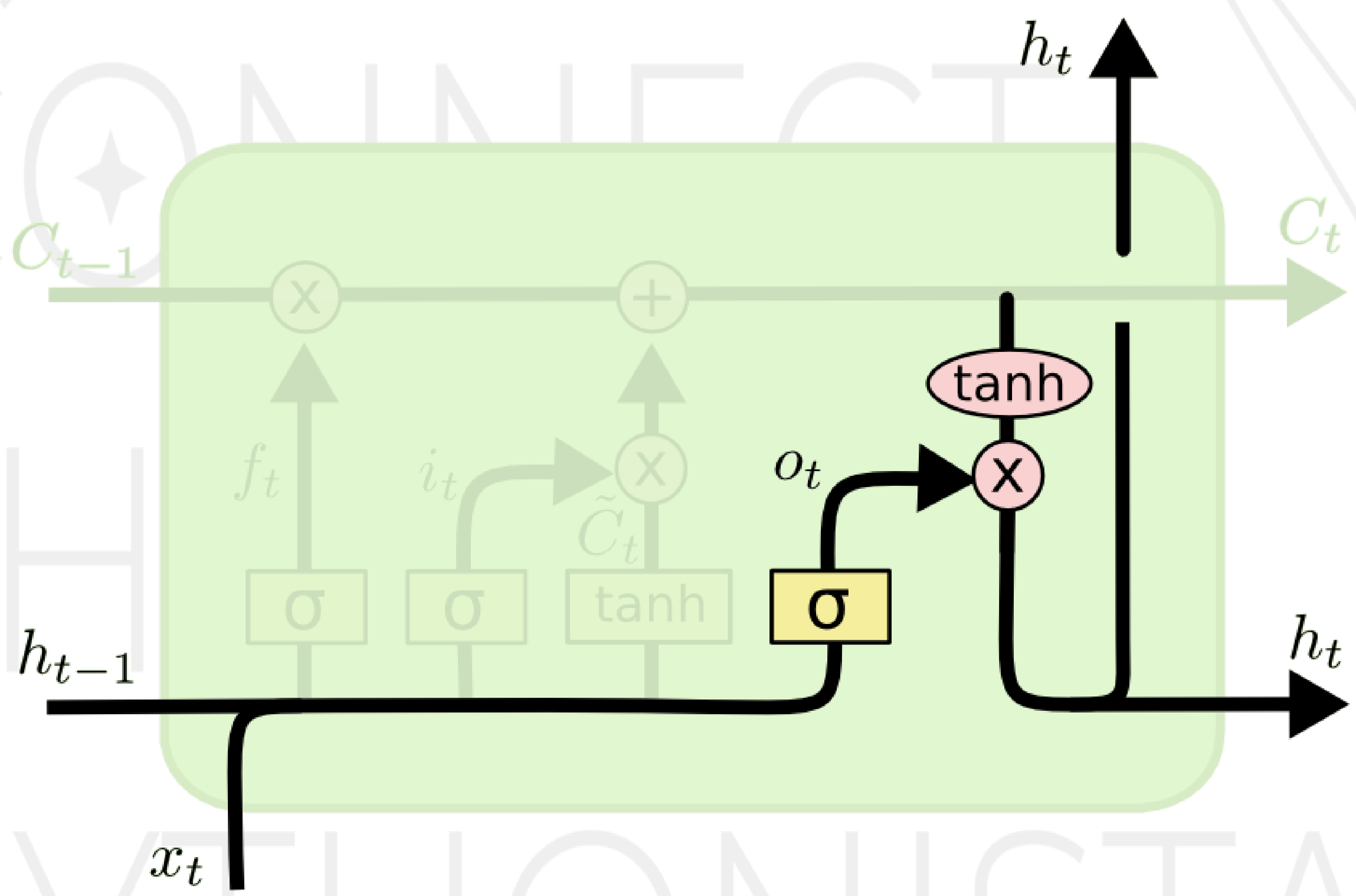
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

장기 기억 정보들의 후보를 정함



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

실제 장기 기억할 정보 C를 확정



$$o_t = \sigma \left(W_o \ [h_{t-1}, x_t] \ + \ b_o \right)$$

$$h_t = o_t * \tanh \left(C_t \right)$$

결과값 o과
단기 기억할 h를 확정

Explaining Recurrent Neural Network Predictions in Sentiment Analysis

Leila Arras[1], Grégoire Montavon[2], Klaus-Robert Müller[2,3,4], and Wojciech Samek[1]
[1]Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany
[2]Machine Learning Group, Technische Universität Berlin, Berlin, Germany
[3]Department of Brain and Cognitive Engineering, Korea University, Seoul, Korea
[4]Max Planck Institute for Informatics, Saarbrücken, Germany
{leila.arras, wojciech.samek}@hhi.fraunhofer.de

Abstract

Recently, a technique called Layer-wise Relevance Propagation (LRP) was shown to deliver insightful *explanations* in the form of input space relevances for understanding feed-forward neural network classification decisions. In the present work, we extend the usage of LRP to recurrent neural networks. We propose a specific propagation rule applicable to multiplicative connections as they arise in recurrent network architectures such as LSTMs and GRUs. We apply our technique to a word-based bi-directional LSTM model on a five-class sentiment prediction task, and evaluate the resulting LRP relevances both qualitatively and quantitatively, obtaining better results than a gradient-based related method which was used in previous work.

1 Introduction

Semantic composition plays an important role in sentiment analysis of phrases and sentences. This includes detecting the scope and impact of negation in reversing a sentiment's polarity, as well as quantifying the influence of modifiers, such as degree adverbs and intensifiers, in rescaling the sentiment's intensity (Mohammad, 2017).

Recently, a trend emerged for tackling these challenges via deep learning models such as convolutional and recurrent neural networks, as observed e.g. on the SemEval-2016 Task for *Sentiment Analysis in Twitter* (Nakov et al., 2016).

As these models become increasingly predictive, one also needs to make sure that they work as intended, in particular, their decisions should be made as transparent as possible.

Some forms of transparency are readily obtained from the structure of the model, e.g. recursive nets (Socher et al., 2013), where sentiment can be probed at each node of a parsing tree.

Another type of analysis seeks to determine what input features were important for reaching the final top-layer prediction. Recent work in this direction has focused on bringing measures of feature importance to state-of-the-art models such as deep convolutional neural networks for vision (Simonyan et al., 2014; Zeiler and Fergus, 2014; Bach et al., 2015; Ribeiro et al., 2016), or to general deep neural networks for text (Denil et al., 2014; Li et al., 2016a; Arras et al., 2016a; Li et al., 2016b; Murdoch and Szlam, 2017).

Some of these techniques are based on the model's local gradient information while other methods seek to redistribute the function's value on the input variables, typically by reverse propagation in the neural network graph (Landecker et al., 2013; Bach et al., 2015; Montavon et al., 2017a). We refer the reader to (Montavon et al., 2017b) for an overview on methods for understanding and interpreting deep neural network predictions.

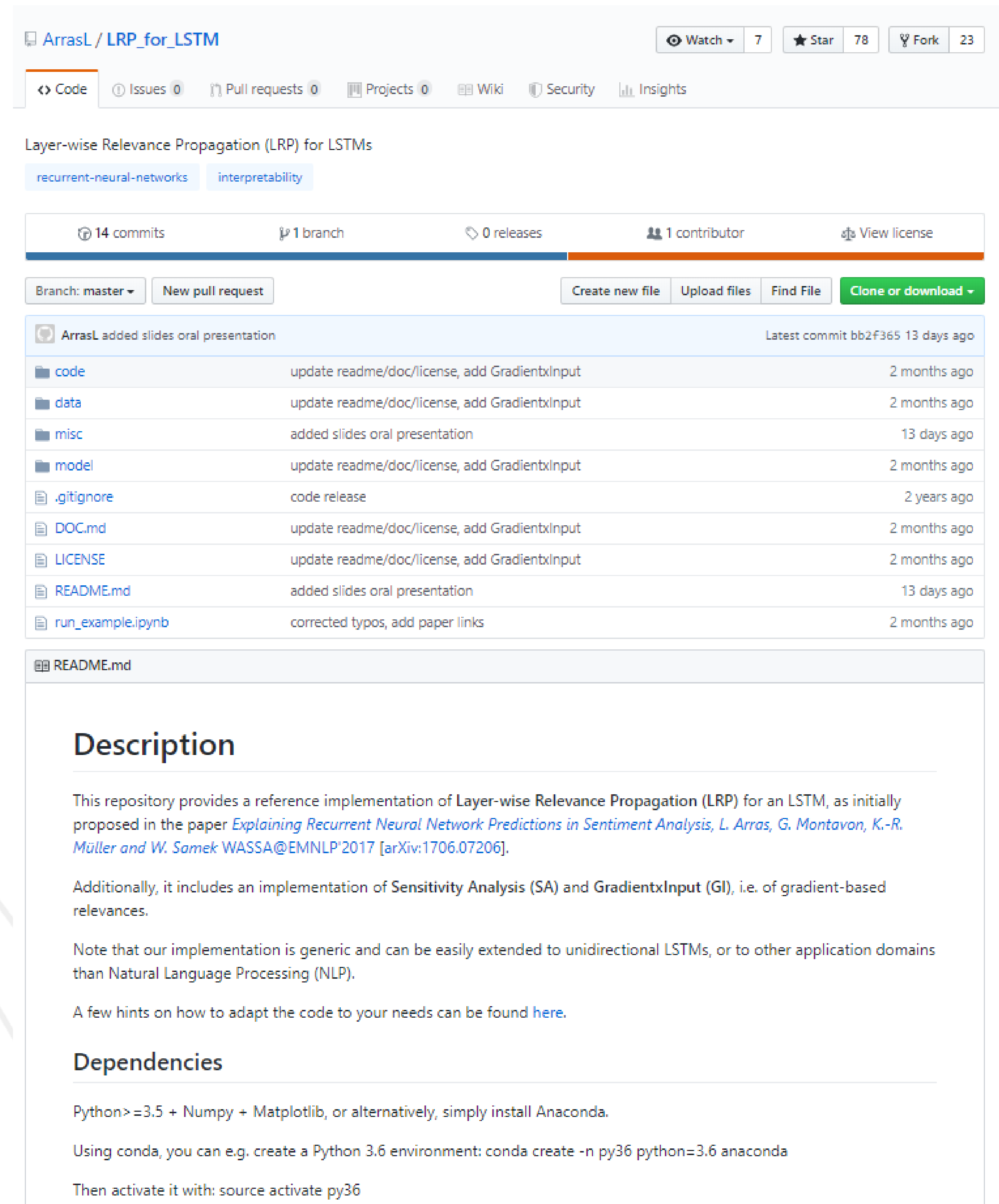
Bach et al. (2015) proposed specific propagation rules for neural networks (LRP rules). These rules were shown to produce better explanations than e.g. gradient-based techniques (Samek et al., 2017), and were also successfully transferred to neural networks for text data (Arras et al., 2016b).

In this paper, we extend LRP with a rule that handles multiplicative interactions in the LSTM model, a particularly suitable model for modeling long-range interactions in texts such as those occurring in sentiment analysis.

We then apply the extended LRP method to a bi-directional LSTM trained on a five-class sentiment prediction task. It allows us to produce reliable explanations of which words are responsible for

159

Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 159–168
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics



LRP는 CNN 계열 뿐만 아니라 다른 신경망에도 적용이 가능하기에 RNN, **LSTM도 구현가능**하고 관련 논문과 코드도 공개되어 있음

https://github.com/ArrasL/LRP_for_LSTM/blob/master/code/LSTM/LSTM_bidi.py

[Explaining Recurrent Neural Network Predictions in Sentiment Analysis, Leila Arras, Grégoire Montavon, Klaus-Robert Müller, Wojciech Samek, 2017]

공개된 코드는
학습된 가중치를 불러와 LRP에 적용해 커스터마이징이 어렵고,
Bi-LSTM으로 구현되어 코드가 굉장히 복잡하여

1. 원본 코드

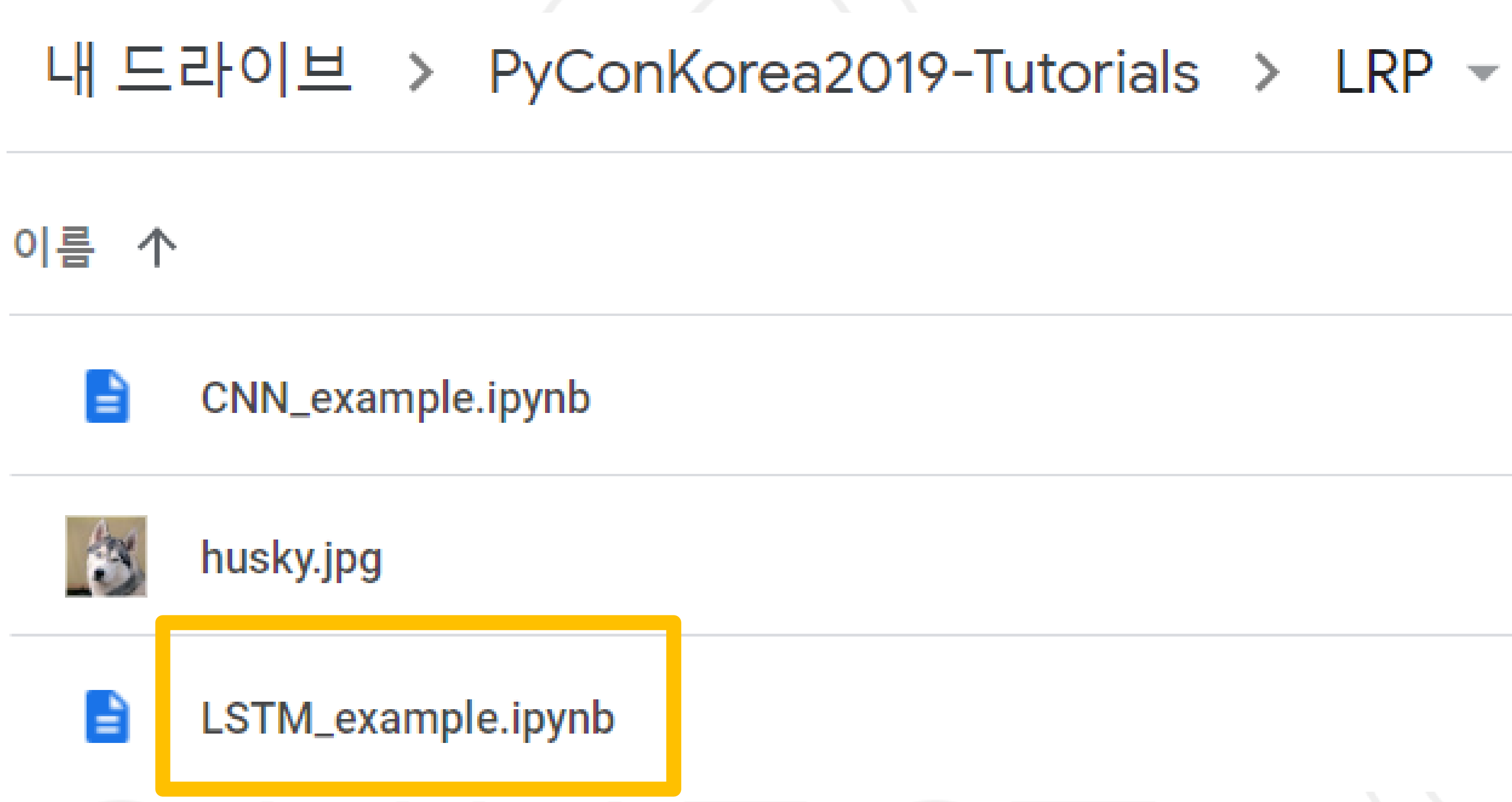
https://github.com/ArrasL/LRP_for_LSTM/blob/master/code/LSTM/LSTM_bidi.py

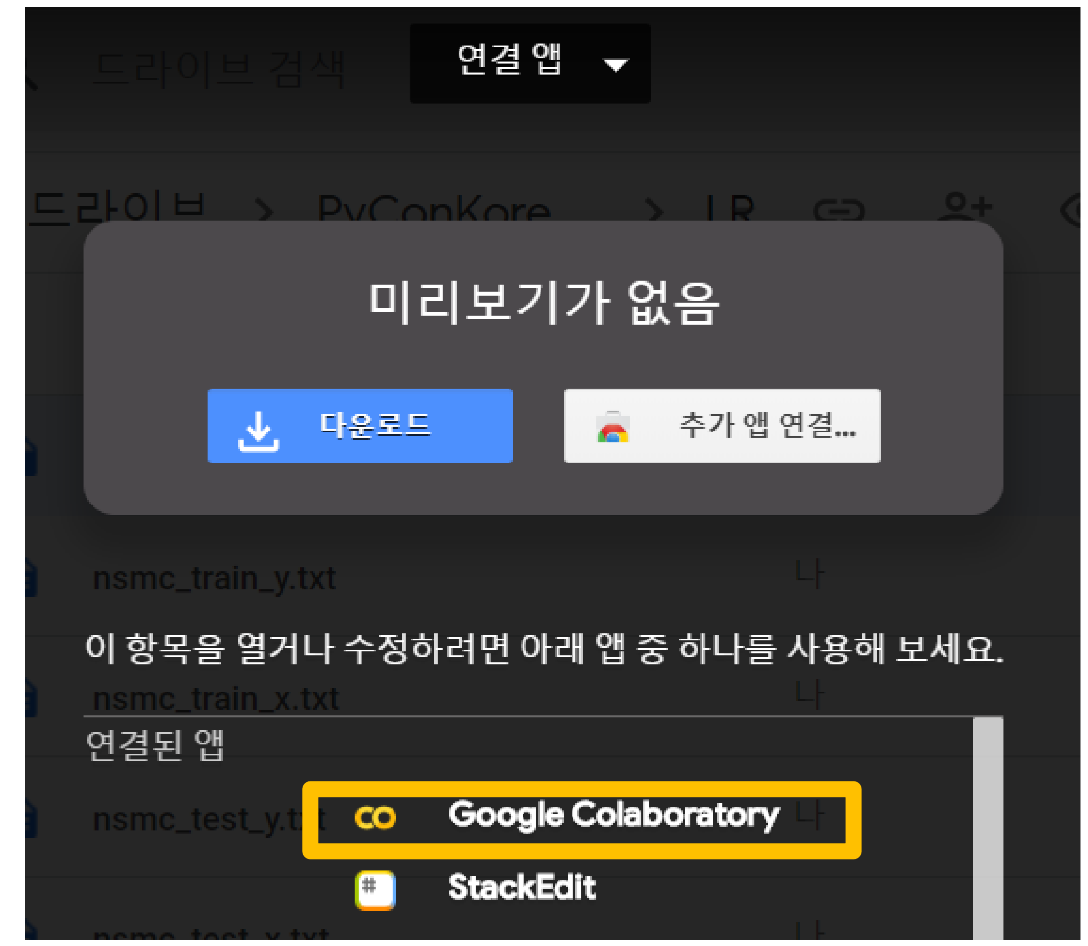
2. 한국어데이터 전처리 과정

https://cyc1am3n.github.io/2018/11/10/classifying_korean_movie_review.html

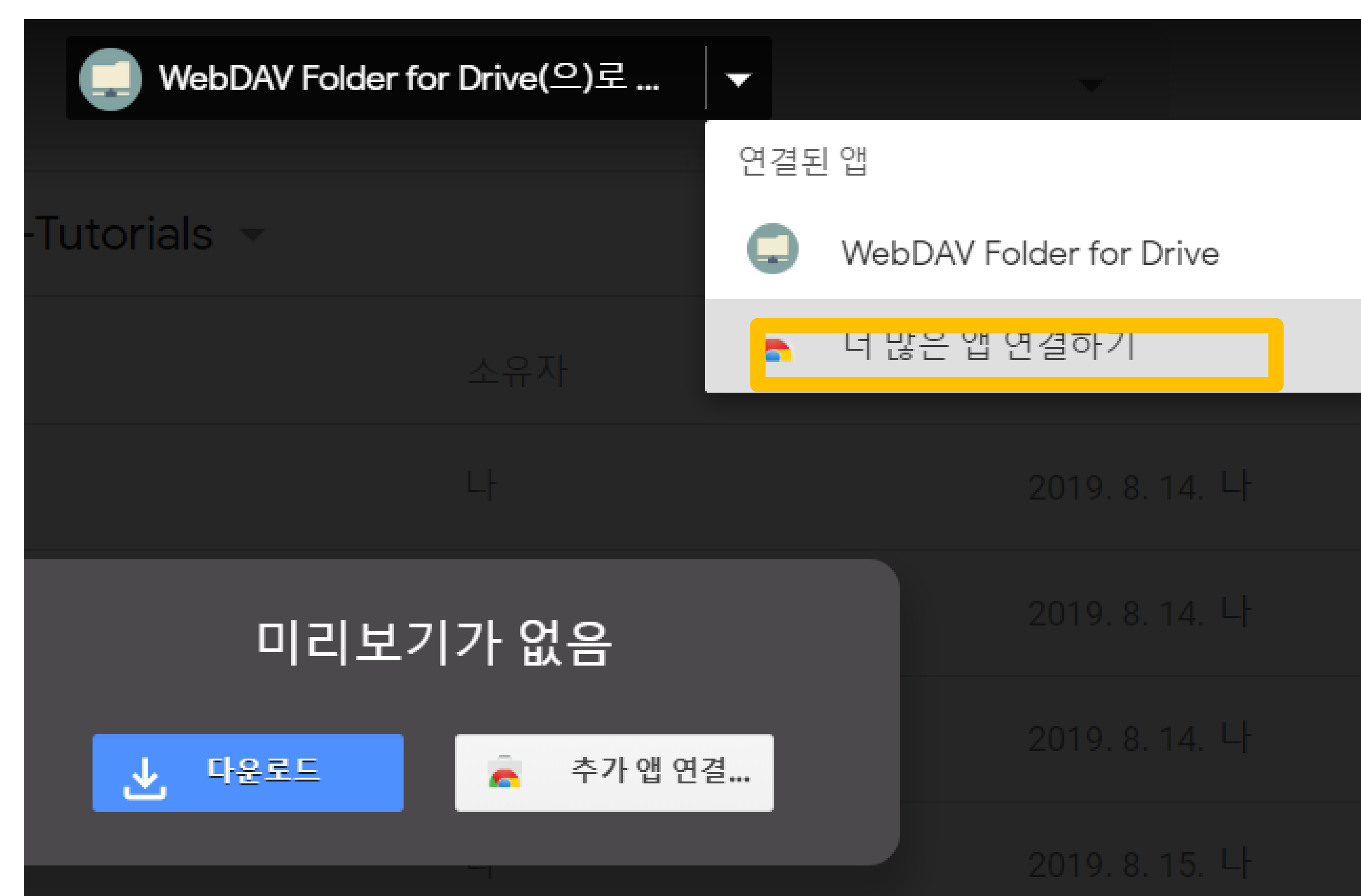
에 기반하여 한국어 자연어 처리가 가능하게 커스터마이징함

구글 드라이브에서 코랩으로 LSTM_example.ipynb파일 연결





만약 '연결 앱'에 코랩이 없다면 '더 많은 앱 연결하기'를 이용



전처리 과정

1. 한국어 문장 데이터를

```
words='이 튜토리얼이 도움이 되신하면 깃허브 스타를 눌러주세요.'
```

2. 형태소 분석을 통해 단어로 나누고

```
from konlpy.tag import Okt

okt = Okt()
print(okt.pos(words))
```

```
[('이', 'Noun'), ('튜토리얼', 'Noun'), ('이', 'Josa'), ('도움', 'Noun'), ('이', 'Josa'), ('되', 'Verb'), ('신하', 'Noun'), ('면', 'Josa'), ('깃허브', 'Noun'), ('스타', 'Noun'), ('를', 'Josa'), ('눌러주세요', 'Verb'), ('.', 'Punctuation')]
```

3. 나눠진 단어를 많이 등장하는 순서로 정렬해 단어 사전을 만든 후

```
selected_words = [f[0] for f in text.vocab().most_common(9999)]
```

4. 문장 데이터를 단어의 인덱스로 변경하여 정수화

```
[3993, 320, 13, 4, 9999, 14, 4301, 4925, 14, 9999, 184, 0]
```

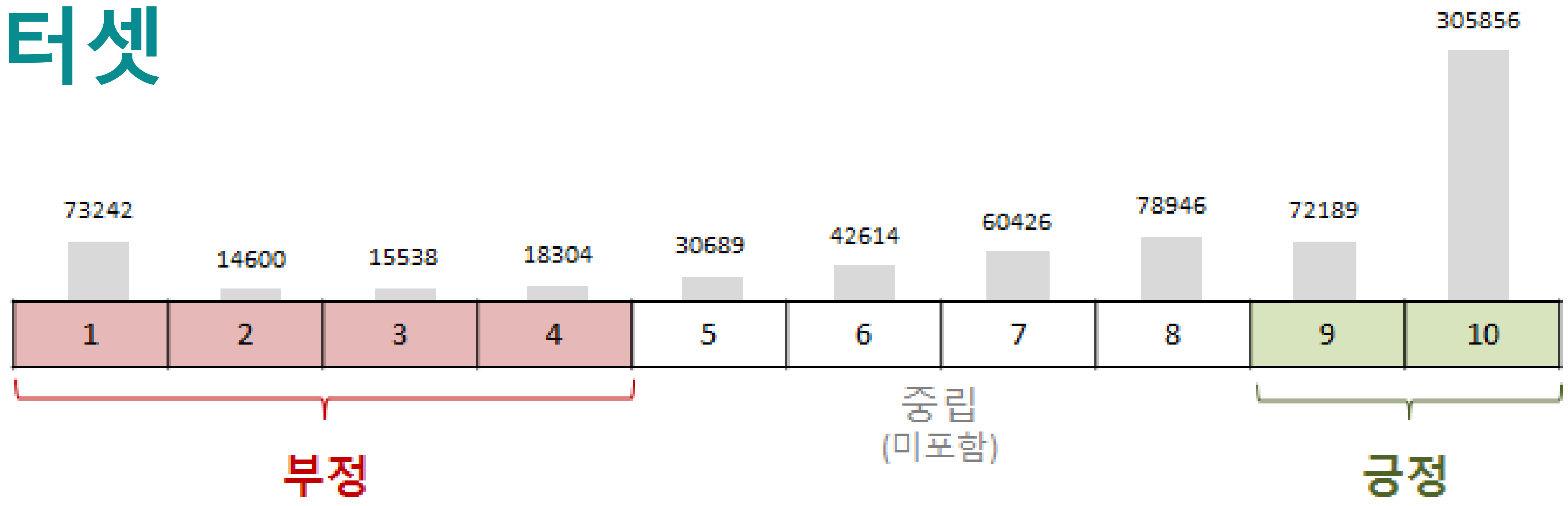
시간관계상 저장된 파일을 불러옴

5. 제로패딩으로 모든 문장 길이를 맞추고

케라스 모듈 이용

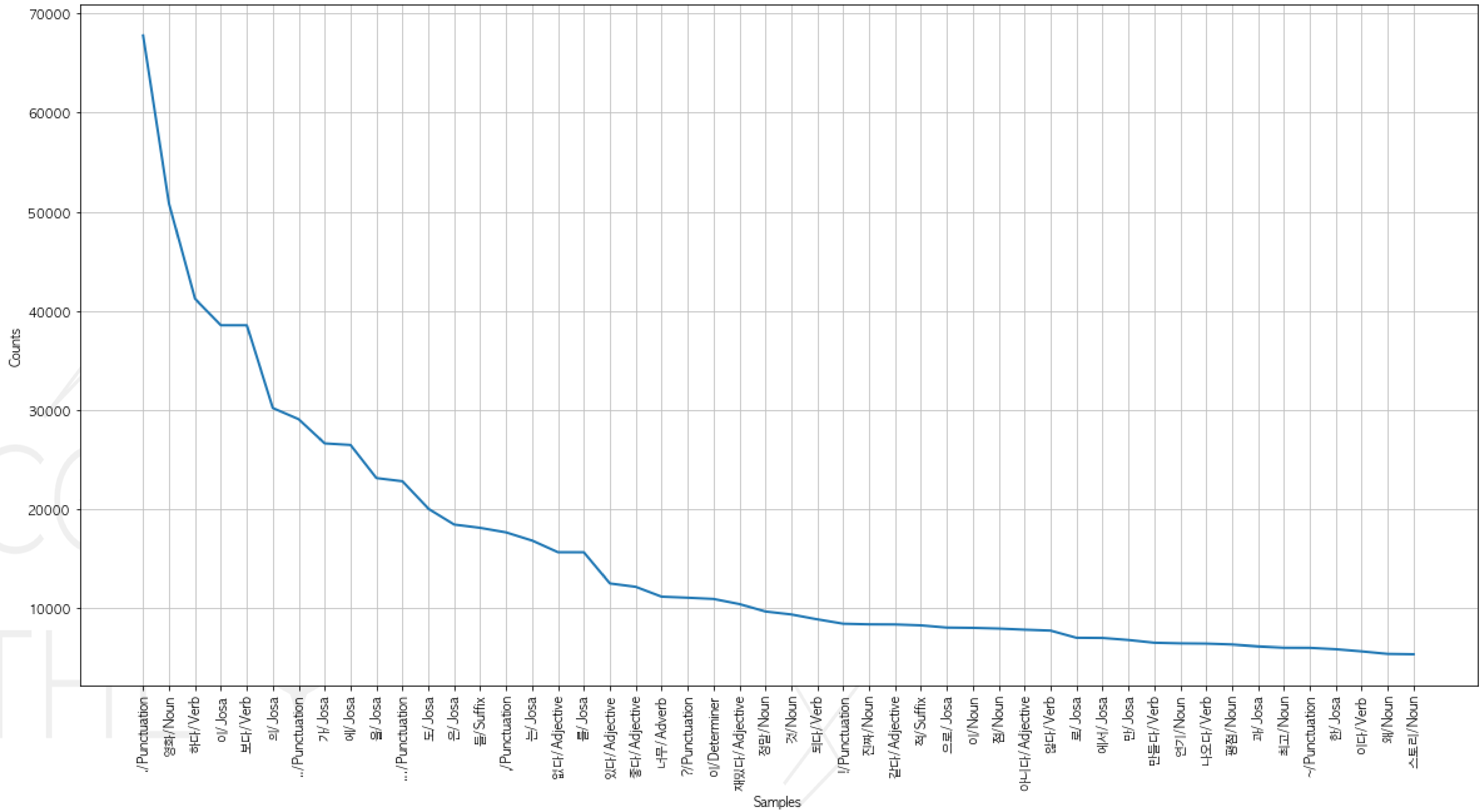
6. 워드 임베딩을 통해 좀더 밀집하게 표현

NSMC 데이터셋

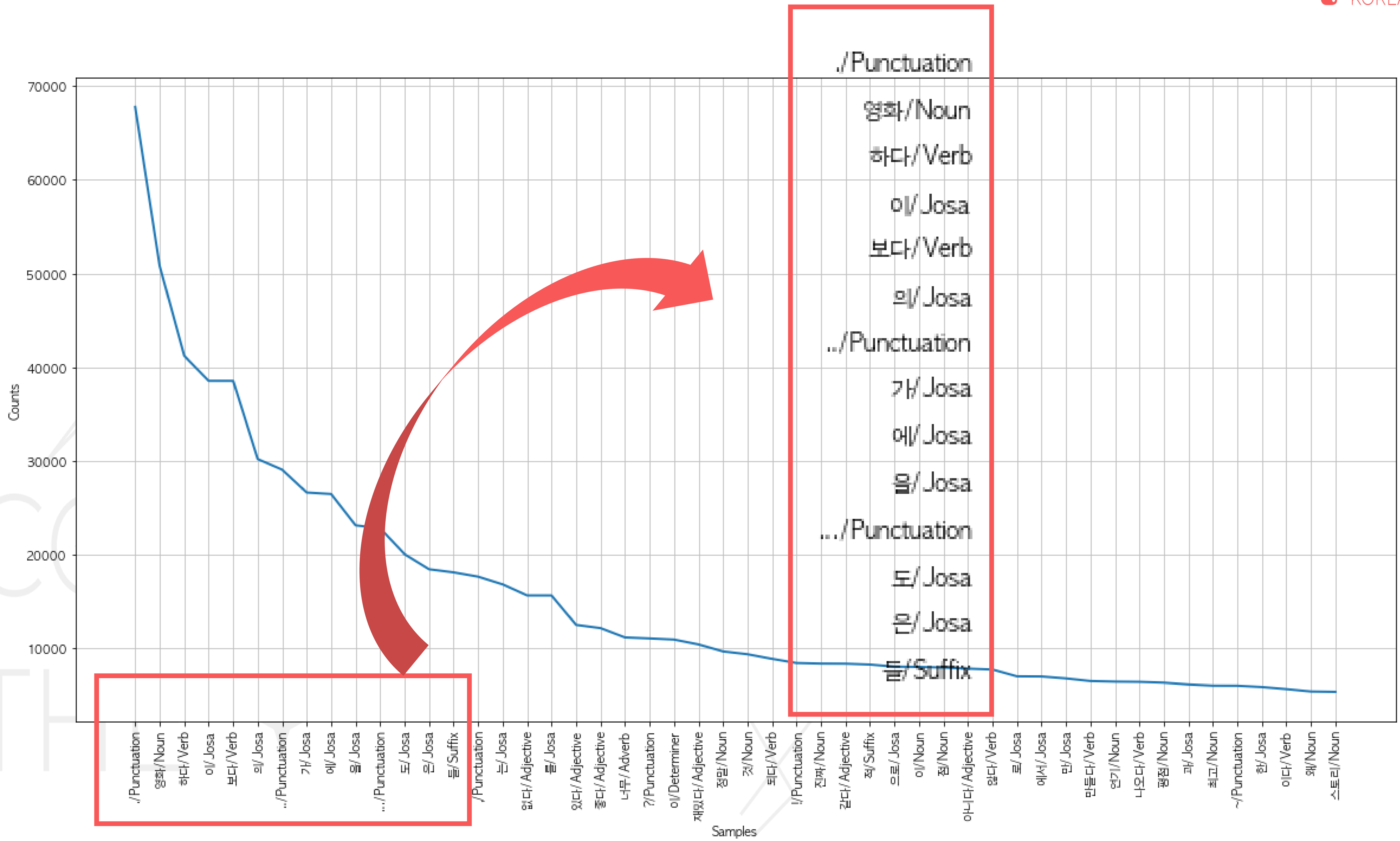



```
'''
OUTPUT:
id	document	label
9976970	아 더빙.. 진짜 짜증나네요 목소리	0
3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
10265843	너무재밓었다그래서보는것을추천한다	0
9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0
6483659	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 던스트가 너무
5403919	막 걸음마 뗀 3세부터 초등학교 1학년생인 8살용영화.ㅋㅋㅋ...별반개도 아까움.	0
7797314	원작의 긴장감을 제대로 살려내지못했다.	0
9443947	별 반개도 아깝다 욕나온다 이응경 길용우 연기생활이몇년인지..정말 발로해도 그것보단 낫겟다 납치.감금
7156791	액션이 없는데도 재미 있는 몇안되는 영화	1
'''
```

https://cyc1am3n.github.io/2018/11/10/classifying_korean_movie_review.html



https://cyc1am3n.github.io/2018/11/10/classifying_korean_movie_review.html



https://cyc1am3n.github.io/2018/11/10/classifying_korean_movie_review.html

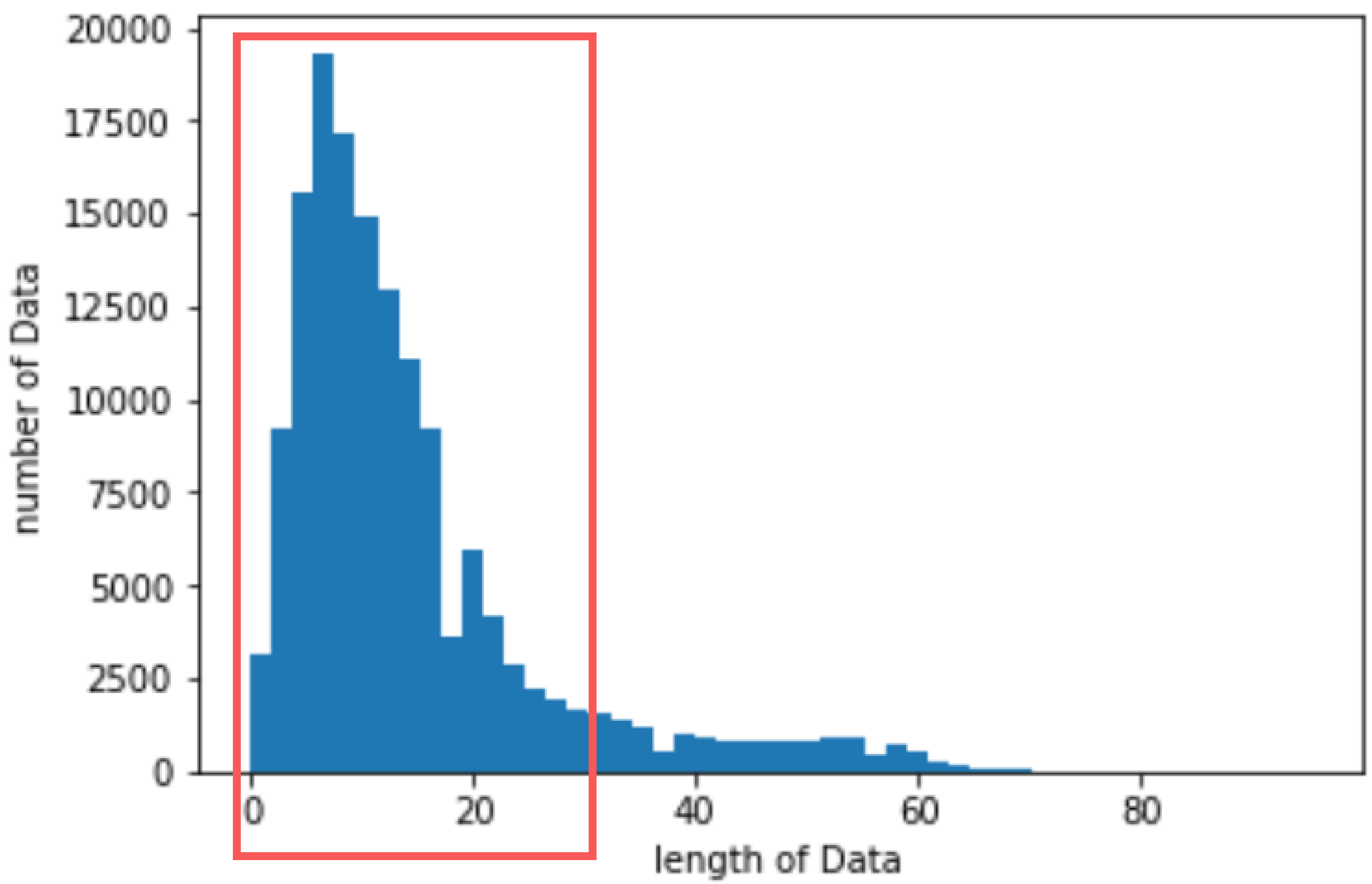
한국어 형태소 분석기 성능 비교 https://iostream.tistory.com/144

khaiii	한나눔	꼬꼬마	KOMORAN	OKT
개봉/NNG	개봉/N	개봉/NNG	개봉/NNG	개봉/Noun
하/XSV	하/X	하/XSV	하/XSV	했을/Verb
였/EP	었을/E	었/EPT	았/EP	때/Noun
을/ETM	때/N	을/ETD	을/ETM	부터/Josa
때/NNG	부터/J	때/NNG	때/NNG	지금/Noun
부터/JX	지금/M	부터/JX	부터/JX	까지/Josa
지금/NNG	까지/J	지금/NNG	지금/NNG	마음/Noun
까지/JX	마음이답답하거/N	까지/JX	까지/JX	이/Josa
마음/NNG	나/J	마음/NNG	마음/NNG	답답하거나/Adjectiv
이/VCP	힘들/P	이/JKS	이/JKS	힘들/Adjective
답답/NNG	ㄹ/E	답답/XR	답답/XR	때/Noun
하/XSA	때/N	하/XSA	하/XSA	이영화/Noun

입력 문장의 **길이**가 서로 **달라서**

max_len보다 **긴 문장은** max_len까지 **자르고**,

max_len보다 **짧은 문장은** max_len까지 **0으로 채움**

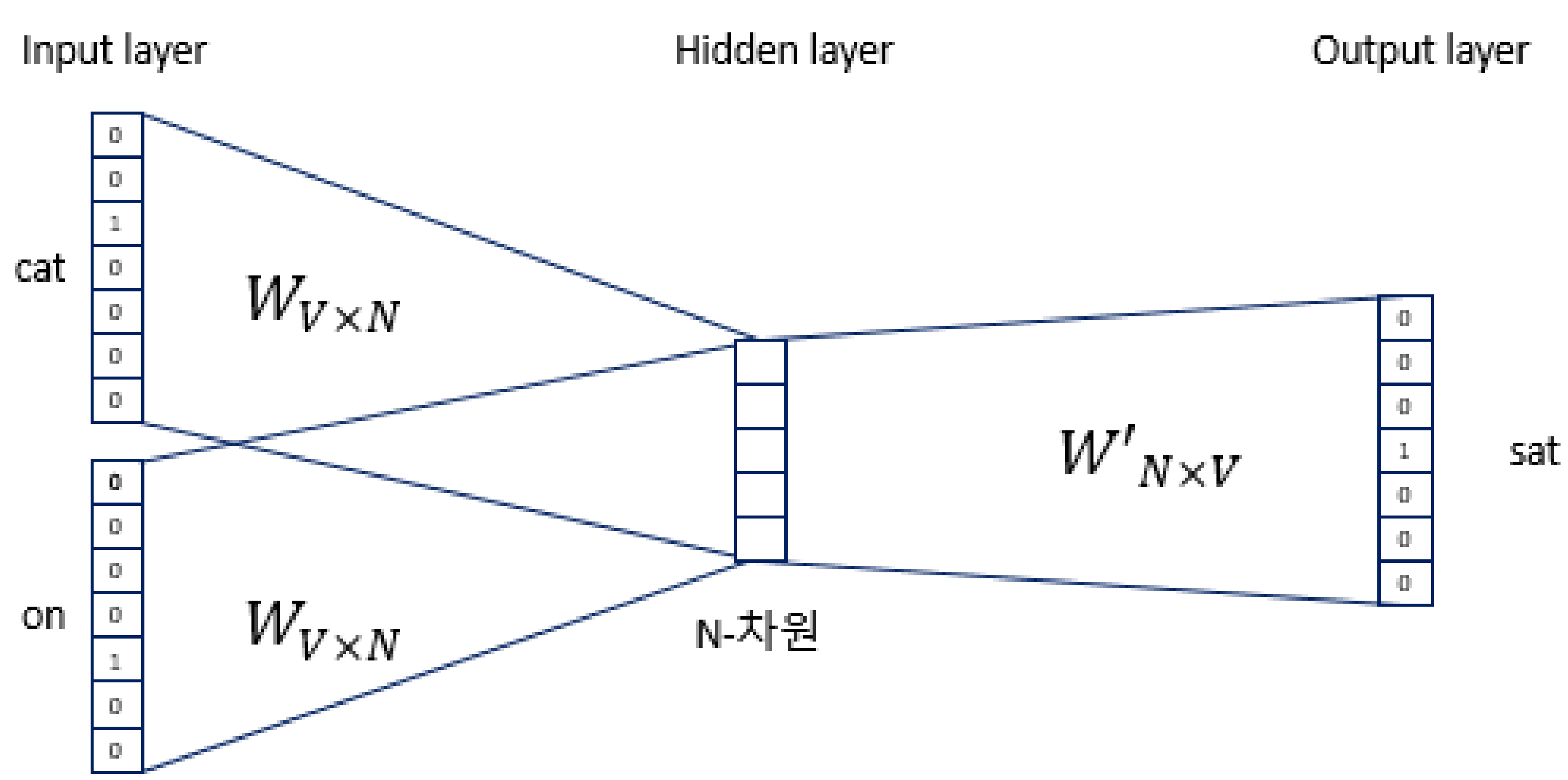


Max_len 값을 결정하기 위해 전체 데이터의 길이 분포를 고려

Model

```
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='sigmoid'))
```

단어 사전의 길이인 10000차원의 **희소**(sparse)한 특징을



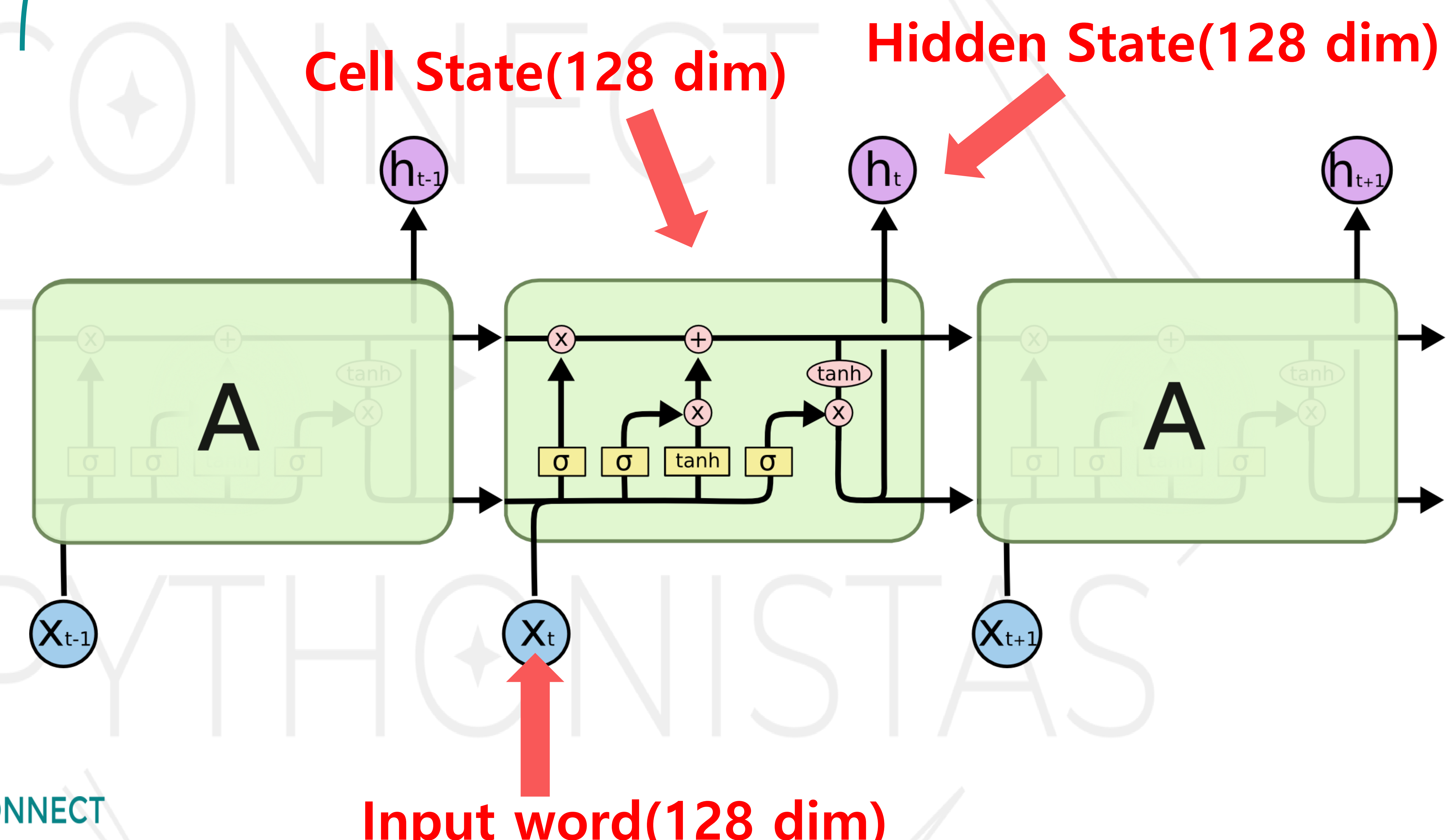
128차원으로 **밀집**(dense)해서 표현

https://wikidocs.net/22660
https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space?hl=ko

Model

```
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='sigmoid'))
```

입력 문장의 길이(max_len) = 30



LRP

저자들이 구현한 Forward (LSTM) 원본코드 forward pass

```python
def forward(self):
    """
    Standard forward pass.
    Compute the hidden layer values (assuming input x/x_rev was previously set)
    """
    T      = len(self.w)
    d      = int(self.Wxh_Left.shape[0]/4)
    # gate indices (assuming the gate ordering in the LSTM weights is i,g,f,o):
    idx    = np.hstack((np.arange(0,d), np.arange(2*d,4*d))).astype(int) # indices of gates i,f,o together
    idx_i, idx_g, idx_f, idx_o = np.arange(0,d), np.arange(d,2*d), np.arange(2*d,3*d), np.arange(3*d,4*d) # indices of ga

    # initialize
    self.gates_xh_Left  = np.zeros((T, 4*d))
    self.gates_hh_Left  = np.zeros((T, 4*d))
    self.gates_pre_Left = np.zeros((T, 4*d))  # gates pre-activation
    self.gates_Left     = np.zeros((T, 4*d))  # gates activation

    for t in range(T):
        self.gates_xh_Left[t]     = np.dot(self.Wxh_Left, self.x[t])
        self.gates_hh_Left[t]     = np.dot(self.Whh_Left, self.h_Left[t-1])
        self.gates_pre_Left[t]    = self.gates_xh_Left[t] + self.gates_hh_Left[t] + self.bxh_Left + self.bhh_Left
        self.gates_Left[t,idx]    = 1.0/(1.0 + np.exp(- self.gates_pre_Left[t,idx]))
        self.gates_Left[t,idx_g]  = np.tanh(self.gates_pre_Left[t,idx_g])
        self.c_Left[t]            = self.gates_Left[t,idx_f]*self.c_Left[t-1] + self.gates_Left[t,idx_i]*self.gates_Left[
        self.h_Left[t]            = self.gates_Left[t,idx_o]*np.tanh(self.c_Left[t])
```

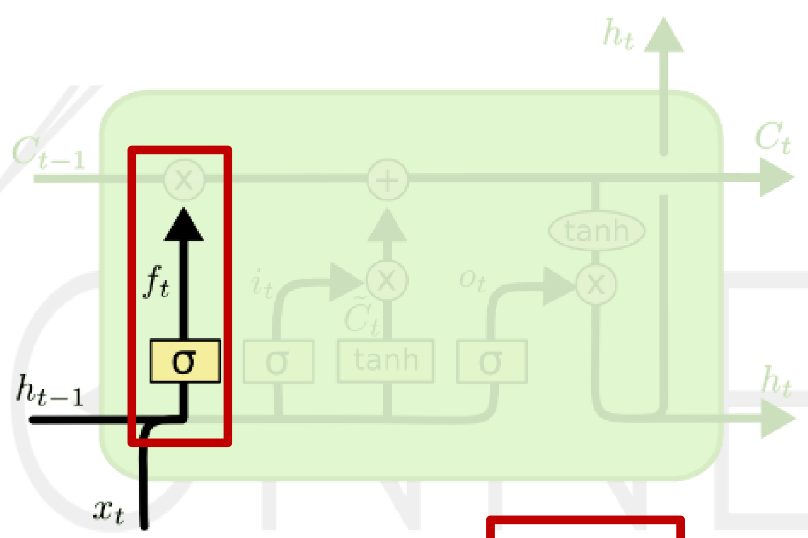
gates ?

gates_pre ???

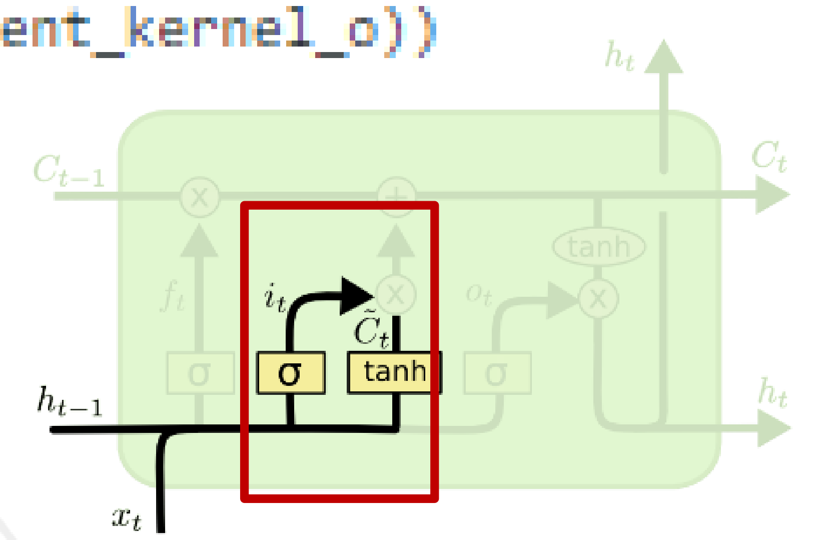
sigmoid ???

비교를 위해 모델 학습에 사용한 keras LSTM 모델을 보면...

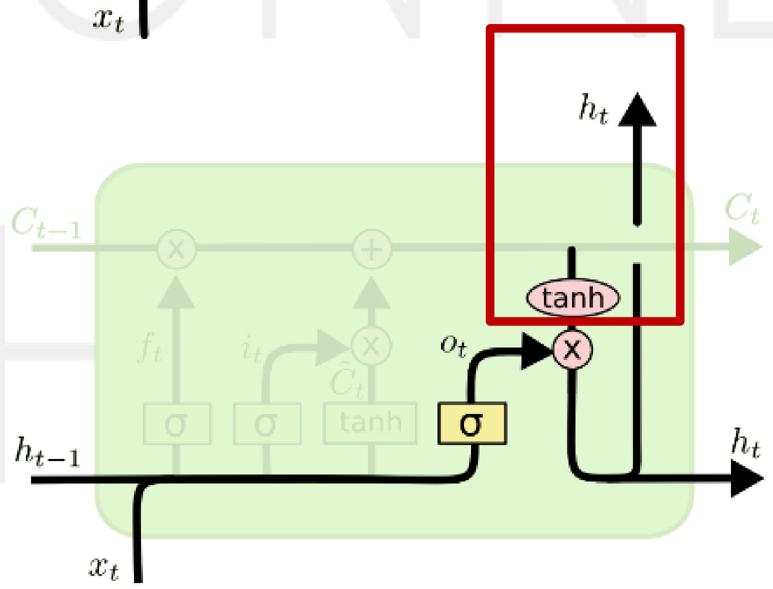
```
i = self.recurrent_activation(x_i + K.dot(h_tm1_i,
                                          self.recurrent_kernel_i))
f = self.recurrent_activation(x_f + K.dot(h_tm1_f,
                                          self.recurrent_kernel_f))
c = f * c_tm1 + i * self.activation(x_c + K.dot(h_tm1_c,
                                                self.recurrent_kernel_c))
o = self.recurrent_activation(x_o + K.dot(h_tm1_o,
                                          self.recurrent_kernel_o))
```



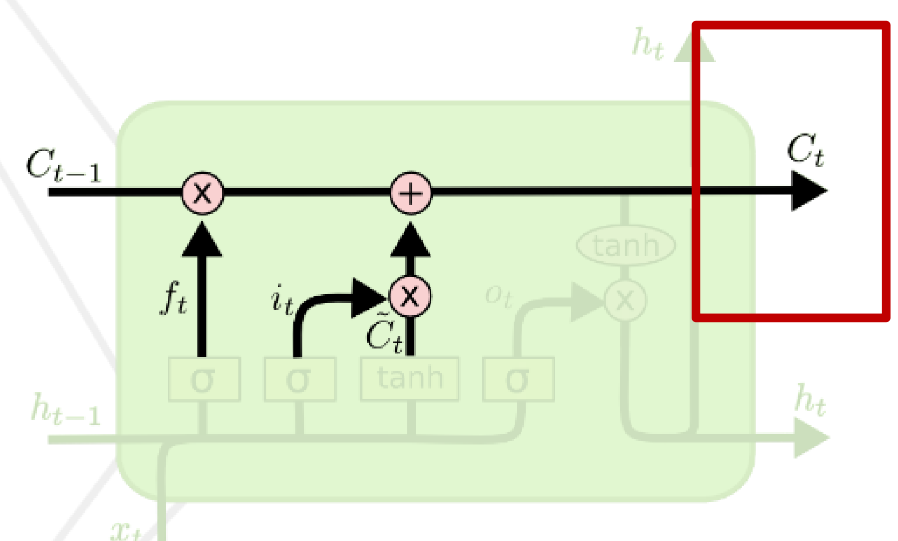
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

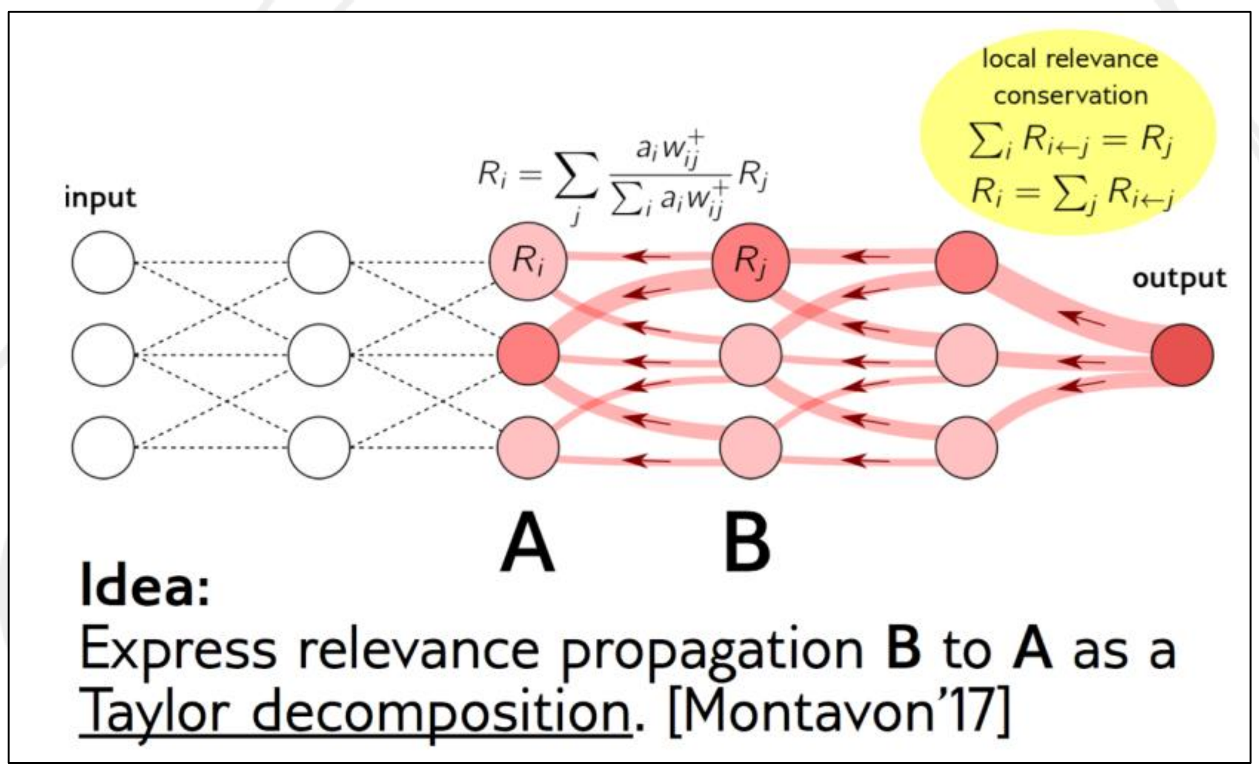


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

논문의 수식을 그대로 구현함

https://github.com/keras-team/keras/blob/master/keras/layers/recurrent.py#L2051

LRP의 backpropagation을 activation 단계에 쉽게 적용하기 위해서



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$\boxed{f'(x) = f(x)(1 - f(x))}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$\boxed{f'(x) = 1 - f(x)^2}$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

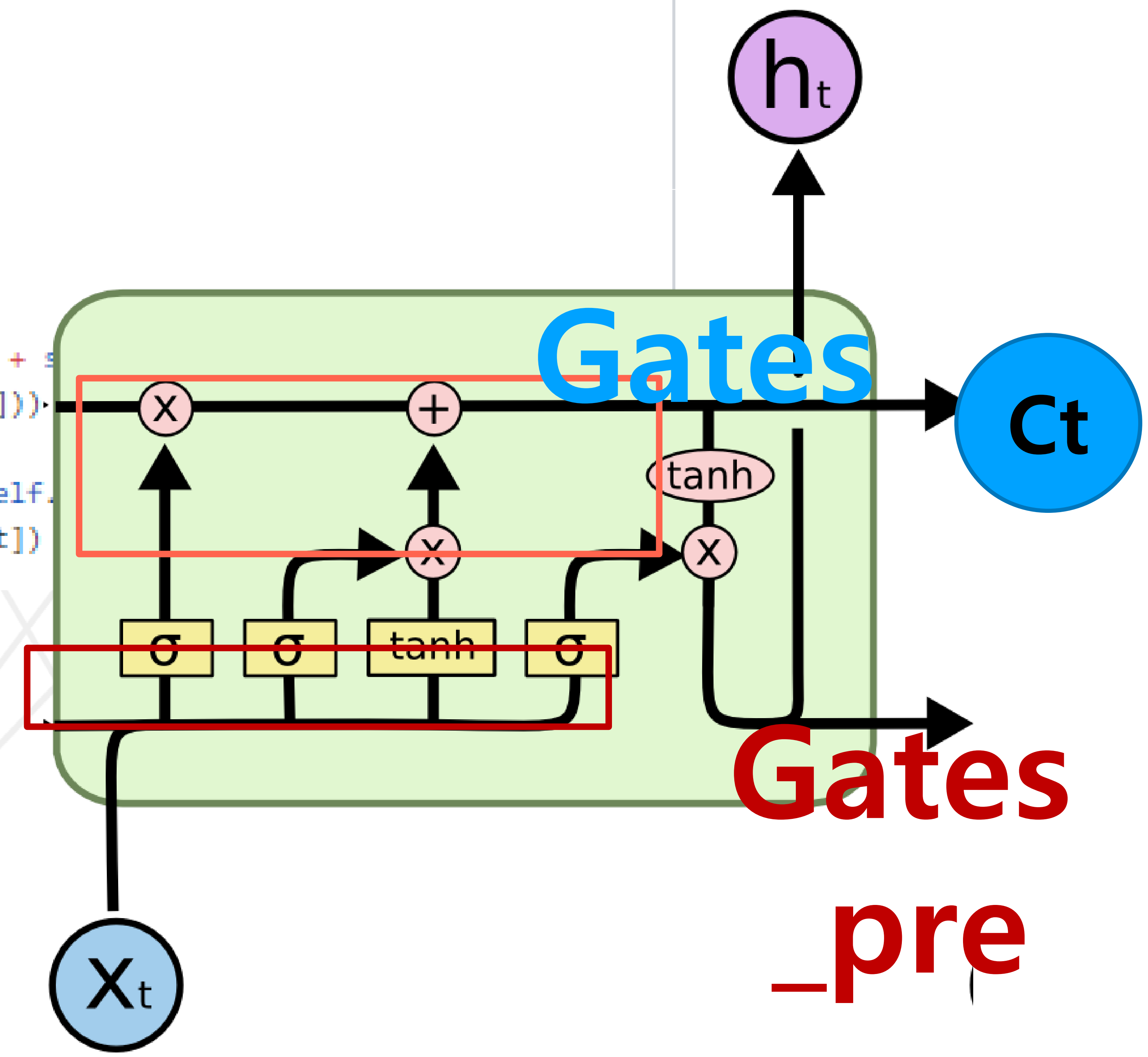
http://aikorea.org/cs231n/optimization-2/

LRP의 핵심인 derivation을 쉽게 구현하기 위해서!!

```python
def forward(self):
    """
    Standard forward pass.
    Compute the hidden layer values (assuming input x/x_rev was previously set)
    """
    T      = len(self.w)
    d      = int(self.Wxh_Left.shape[0]/4)
    # gate indices (assuming the gate ordering in the LSTM weights is i,g,f,o):
    idx    = np.hstack((np.arange(0,d), np.arange(2*d,4*d))).astype(int) # indices of gates i,f,o together
    idx_i, idx_g, idx_f, idx_o = np.arange(0,d), np.arange(d,2*d), np.arange(2*d,3*d), np.arange(3*d,4*d) # indices of ga

    # initialize
    self.gates_xh_Left  = np.zeros((T, 4*d))
    self.gates_hh_Left  = np.zeros((T, 4*d))
    self.gates_pre_Left = np.zeros((T, 4*d))  # gates pre-activation
    self.gates_Left     = np.zeros((T, 4*d))  # gates activation

    for t in range(T):
        self.gates_xh_Left[t]      = np.dot(self.Wxh_Left, self.x[t])
        self.gates_hh_Left[t]      = np.dot(self.Whh_Left, self.h_Left[t-1])
        self.gates_pre_Left[t]     = self.gates_xh_Left[t] + self.gates_hh_Left[t] + s
        self.gates_Left[t,idx]     = 1.0/(1.0 + np.exp(- self.gates_pre_Left[t,idx]))
        self.gates_Left[t,idx_g]   = np.tanh(self.gates_pre_Left[t,idx_g])
        self.c_Left[t]             = self.gates_Left[t,idx_f]*self.c_Left[t-1] + self.
        self.h_Left[t]             = self.gates_Left[t,idx_o]*np.tanh(self.c_Left[t])
```



LRP구현

저자들이 구현한 Forward (LSTM) 원본코드 forward pass

```
ds                      = np.zeros((C))
ds[sensitivity_class] = 1.0
dy_Left               = ds.copy()
dy_Right              = ds.copy()

self.dh_Left[T-1]     = np.dot(self.Why_Left.T,  dy_Left)
self.dh_Right[T-1]    = np.dot(self.Why_Right.T, dy_Right)

for t in reversed(range(T)):
    self.dgates_Left[t,idx_o]    = self.dh_Left[t] * np.tanh(self.c_Left[t])  # do[t]
    self.dc_Left[t]             += self.dh_Left[t] * self.gates_Left[t,idx_o] * (1.-(np.tanh(self.c_Left[t]))**2) # dc[t]
    self.dgates_Left[t,idx_f]    = self.dc_Left[t] * self.c_Left[t-1]         # df[t]
    self.dc_Left[t-1]            = self.dc_Left[t] * self.gates_Left[t,idx_f] # dc[t-1]
    self.dgates_Left[t,idx_i]    = self.dc_Left[t] * self.gates_Left[t,idx_g] # di[t]
    self.dgates_Left[t,idx_g]    = self.dc_Left[t] * self.gates_Left[t,idx_i] # dg[t]
    self.dgates_pre_Left[t,idx]  = self.dgates_Left[t,idx] * self.gates_Left[t,idx] * (1.0 - self.gates_Left[t,idx]) # d ifo pre[t]
    self.dgates_pre_Left[t,idx_g]= self.dgates_Left[t,idx_g] * (1.-(self.gates_Left[t,idx_g])**2) # d g pre[t]
    self.dh_Left[t-1]            = np.dot(self.Whh_Left.T, self.dgates_pre_Left[t])
    self.dx[t]                   = np.dot(self.Wxh_Left.T, self.dgates_pre_Left[t])
```

D sigmoid

D tanh