시계열용 DL 방법론

Pure deep learning - N-BEAT
RNN 계열 - deepAR

인과형 예측기법

Deep Learning 계열

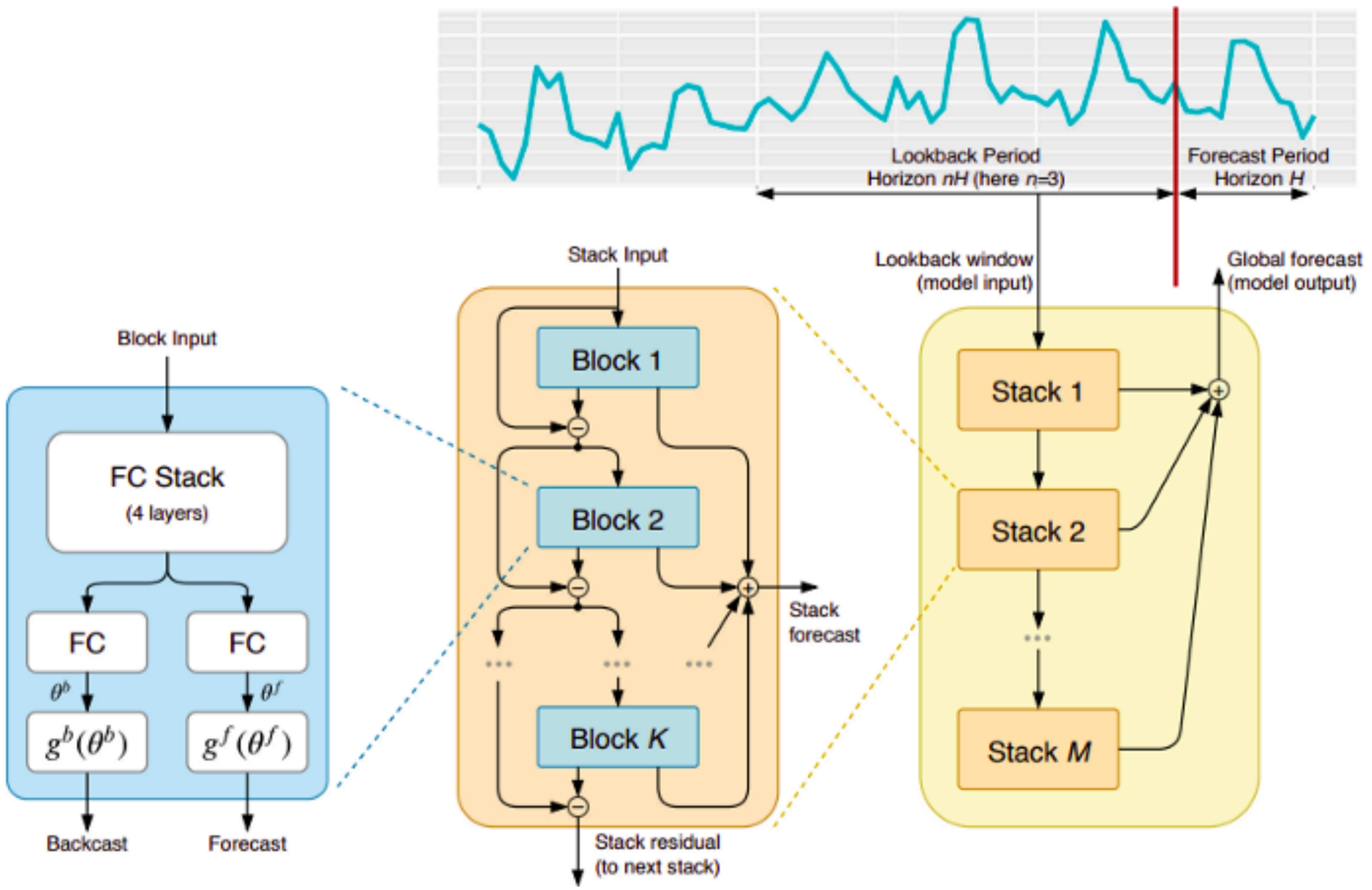
- 시계열 모형은 시계열의 현재값과 과거값 간 안정적인 함수관계를 가정
- Univer approximator property: ANN은 오류역전파 알고리즘을 통해, 임의의 함수를 임의의 정밀도로 근사해낼 수 있는 능력을 가지고 있으므로, 시계열 예측에도 활용할 수 있음
- 기계학습 방법론은 다변량 입력, 복잡한 비선형 관계, 결측치 존재라는 불리한 상황에서도 시계열 예측을 효과적으로 수행할 수 있다고 알려짐
- 시계열을 분석하기 위해 개발된 RNN(Recurrent Neural Networks) 이외도 DNN(Deep Neural Networks), CNN(Convolutional Neural Networks), Attention mechanism 등 다양한 방법들의 결합 활용한 모형들이 있음

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bigdataai&logNo=222043157198

N-BEAT

N-BEAT 개요

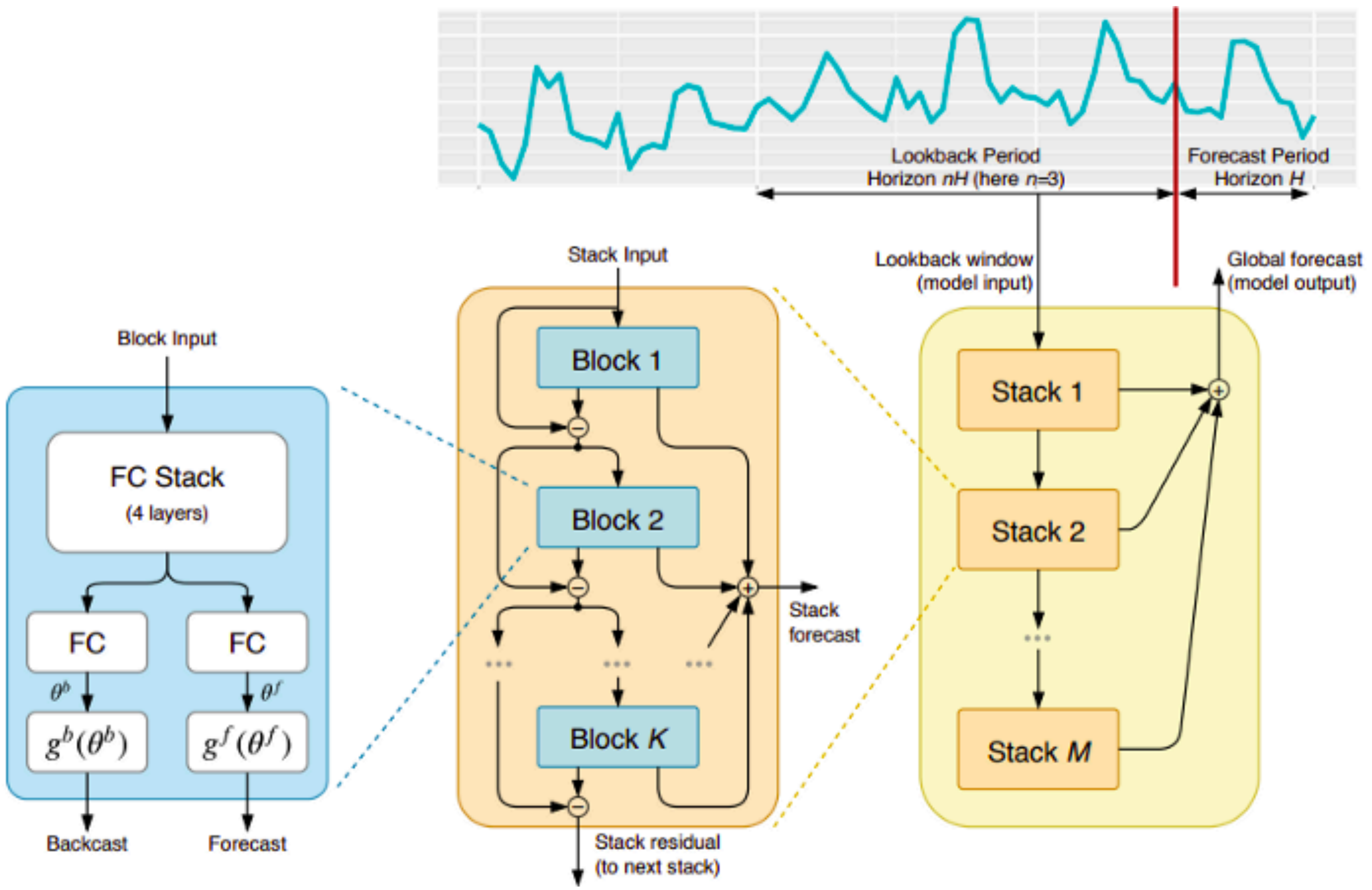
- Yoshua Bengio이 창업한 ElementAI에서 제안
- M-competition의 결과와는 달리 순수한 deep learning 구조만으로 시계열을 분석하고자 하는 시도
- 실용적으로 이용할 수 있는 해석 가능한 deep learning 구조를 고안하고자 함



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

특장점

- **표현하기 쉽고 사용하기 쉬움:**
    - 간단한 module 구조로 쉽게 이해할 수 있음
    - Feature engineering이 최소화 된 구조
- **다중 시계열 모델링 가능:**
    - 다른 스케일의 시계열을 동시에 모델링 할 수 있음
    - 이러한 성질은 inner procedure와 outer proceduer로 구성된 meta-learning을 통해서 이루어짐
    - Inner learning procedure은 block 안에서 이루어지며, local의 일시적인 성격을 모델링
    - Outer learning procedure는 stack 안에서 일어나며, 시계열 전반의 global한 성격을 학습



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

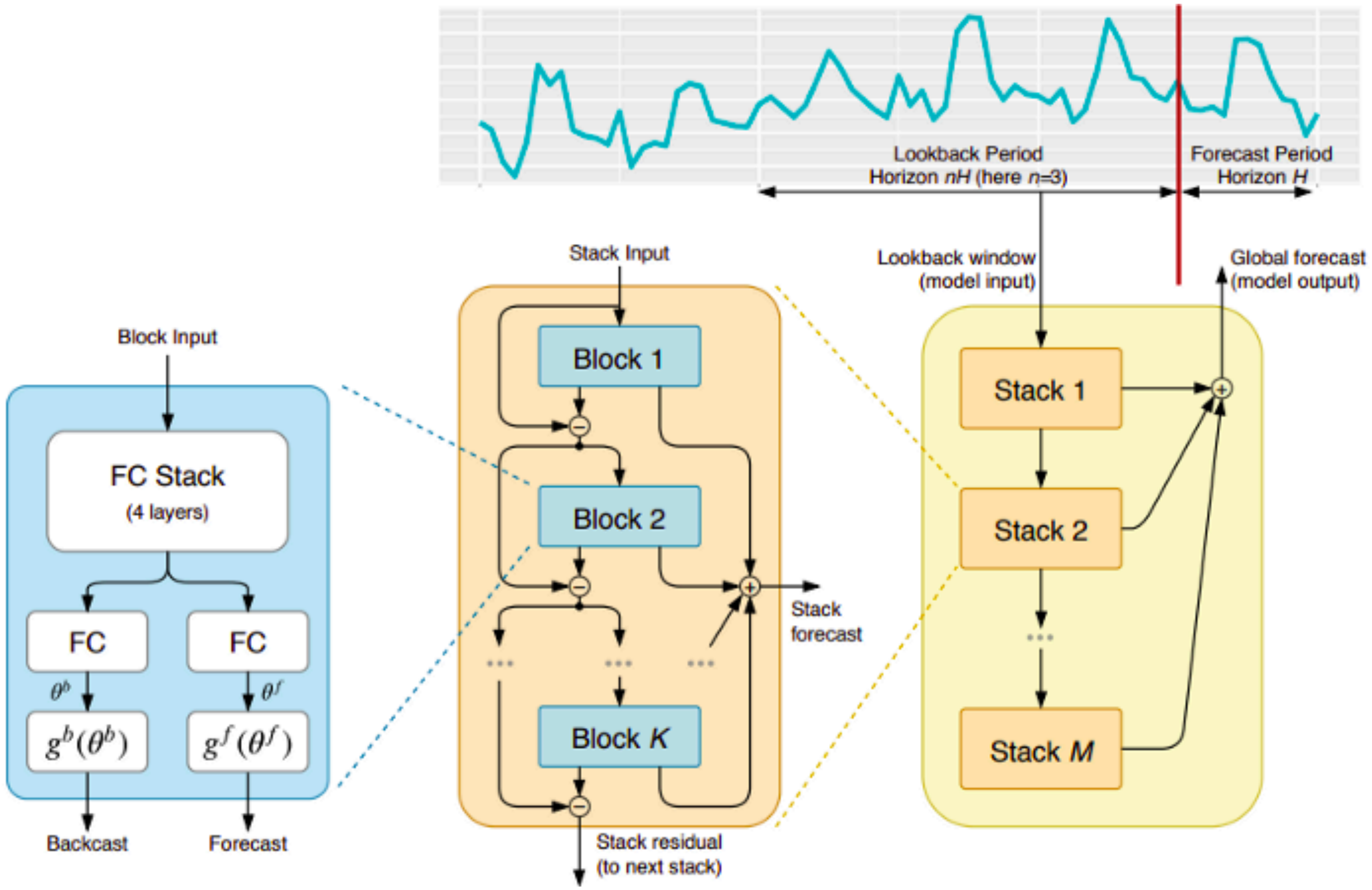
모형 상세

Basic Block (Doubly Residual Stacking)

- *N-BEATS*에서는 *lookback window*를 통해 입력된 데이터를 *backcast*와 *forecast*이라는 두개의 *residual branch* 형태로 활용. 두갈래이므로 *“doubly”*라는 이름이 붙음

- 연속된 block는 이전 block에서의 backcast의 재구성(reconstruction)을 통해 residual error를 모델링 한다는 측면에서 Box-Jenkins 방법을 흉내낸 것으로 볼 수 있음

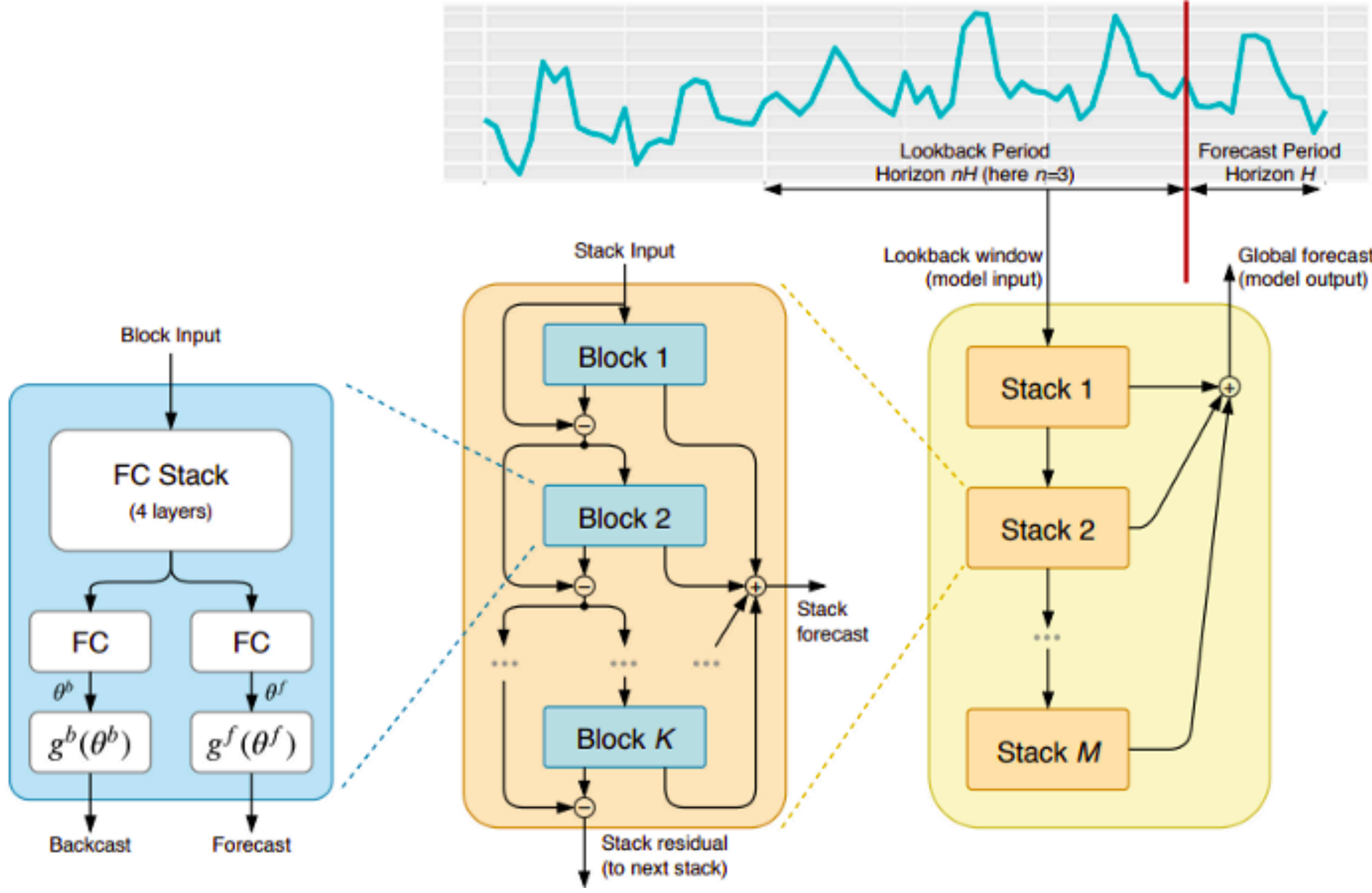
Note: The original *N-BEATS* implementation only works with univariate time series.



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Basic Block (Doubly Residual Stacking)



- l번째 block은 입력으로 $\mathbf{x}_l$을 받고, 출력으로는 $\hat{\mathbf{x}}_l$과 $\hat{\mathbf{y}}_l$을 만들어 냄
- Input window의 크기는 forecast horizon H의 정수배 ($2H$~$7H$)로 둠
- 다른 블록들의 입력값 $\mathbf{x}_l$은 이전 블록들의 residual output (input + block을 통과한 출력을 합친 결과)
- $\hat{\mathbf{x}}_l$은 $\mathbf{x}_l$ lookback input에 대한 추정(backcast)이며, $\hat{\mathbf{y}}_l$은 H의 길이를 갖는 forward 예측값 벡터
- 각 *block*의 예측값은 *stack level*에서 한번 합해지고, 전체 *network*에서 합해짐 *(Interpretability*의 근거)

$$\mathbf{x}_l = \mathbf{x}_{l-1} - \hat{\mathbf{x}}_{l-1}, \quad \hat{\mathbf{y}} = \sum_l \hat{\mathbf{y}}_l$$

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Basic Block

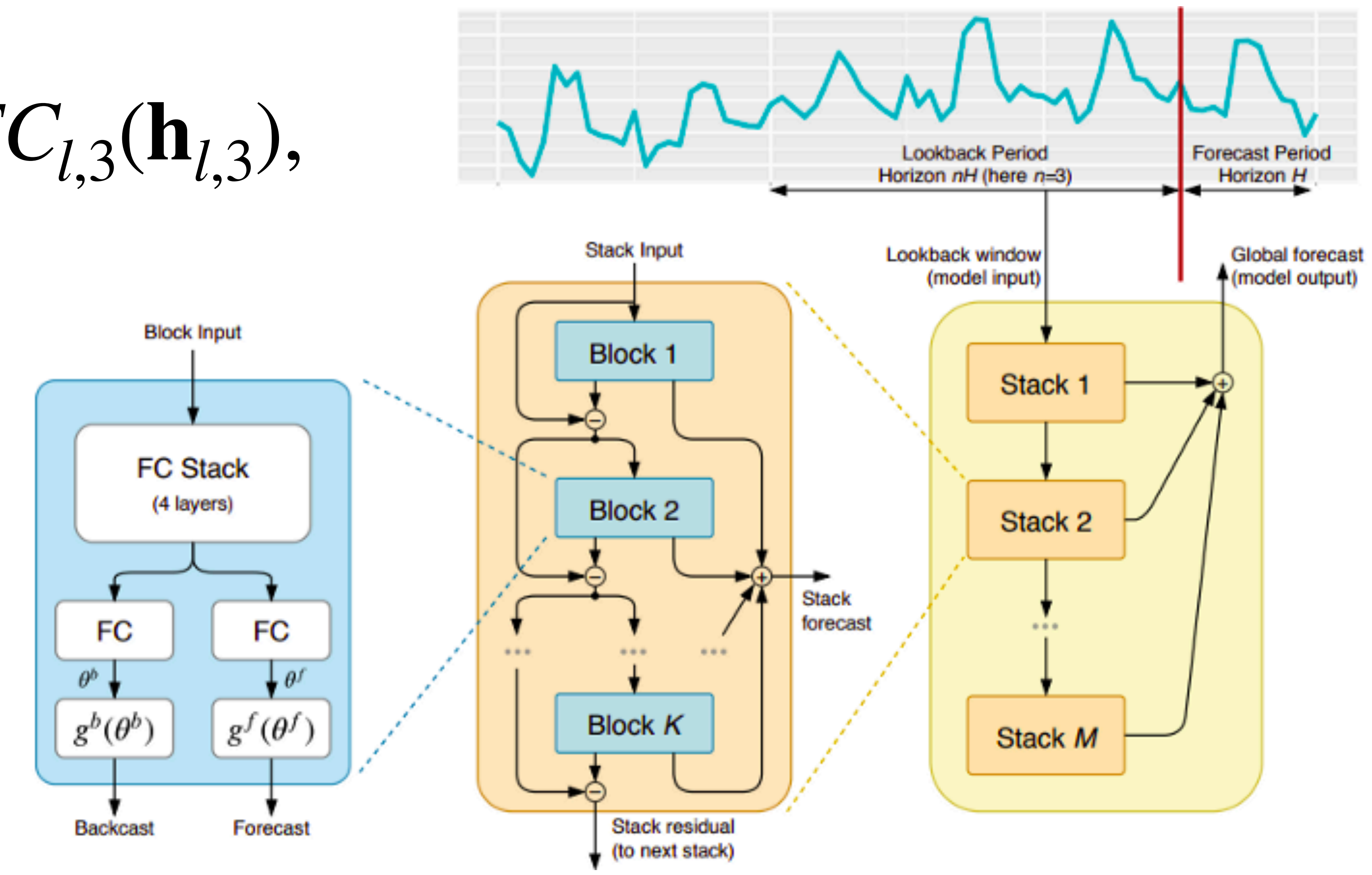
- FC network는 다음과 같이 쓸 수 있음

$$\mathbf{h}_{l,1} = FC_{l,1}(\mathbf{x}_l), \mathbf{h}_{l,2} = FC_{l,2}(\mathbf{h}_{l,1}), \mathbf{h}_{l,3} = FC_{l,3}(\mathbf{h}_{l,2}), \mathbf{h}_{l,4} = FC_{l,3}(\mathbf{h}_{l,3}),$$

FC = Linear

$$\theta_l^b = \text{LINEAR}_l^b(\mathbf{h}_{l,4}), \theta_l^f = \text{LINEAR}_l^f(\mathbf{h}_{l,4})$$

$$\theta_l^f = \mathbf{W}_l^f \mathbf{h}_{l,4}, \mathbf{h}_{l,1} = \text{RELU}(\mathbf{W}_{l,1}\mathbf{x}_l + \mathbf{b}_{l,1})$$



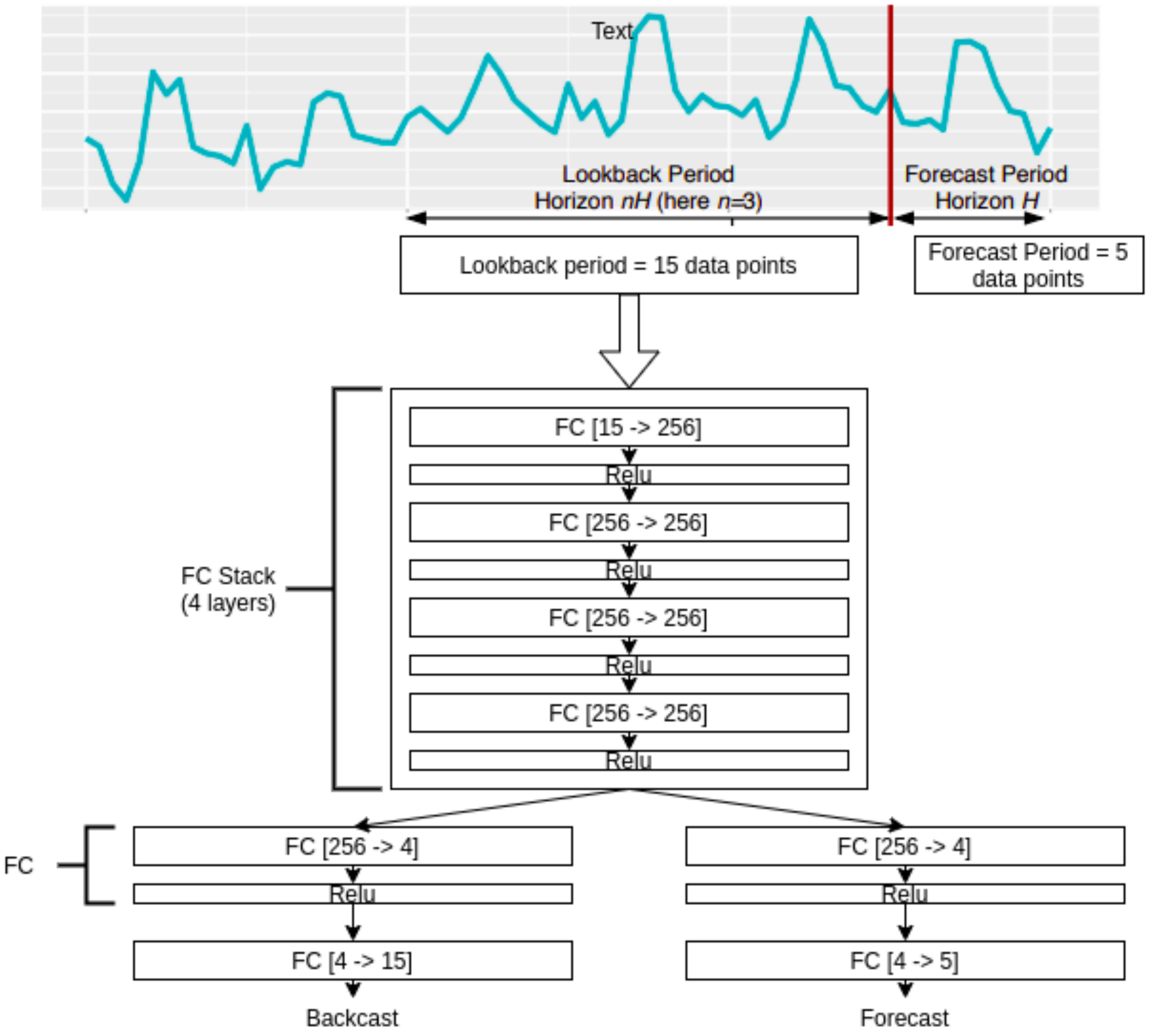
- g함수는 다음과 같이 정의됨

$$\hat{\mathbf{y}}_l = g^f(\theta^f) = \sum_{i=1}^{\dim(\theta_l^f)} \theta_{l,i}^f v_i^f, \quad \hat{\mathbf{x}}_l = g^b(\theta^b) = \sum_{i=1}^{\dim(\theta_l^b)} \theta_{l,i}^b v_i^b$$

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Basic Block



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Interpretability

- 일반 모형에서는 다음의 fully connected layer를 매 block의 마지막에 가짐

$$\hat{\mathbf{y}}_l = \mathbf{V}_l^f \theta_l^f + b_l^f, \quad \hat{\mathbf{x}}_l = \mathbf{V}_l^b \theta_l^b + b_l^b$$

- 설명을 위해서는 각 블록 내 마지막 fully connected layer를 제거하고, 그 대신 설명 가능한 layer를 추가

    -

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

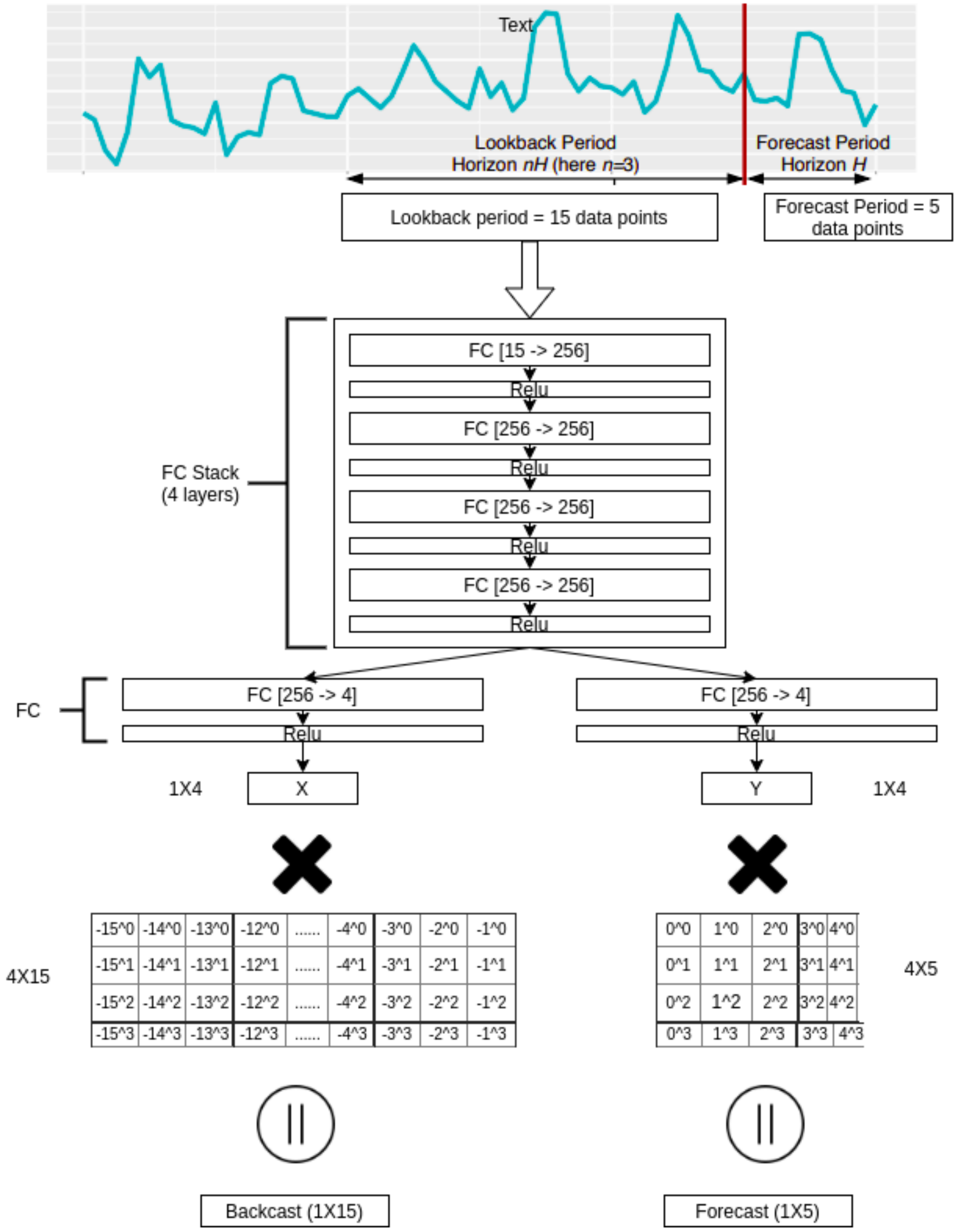
Interpretability

- Trend model
  - Trend model: 단조함수이며 천천히 변하는 함수

$$\hat{\mathbf{y}}_{s,l} = \sum_{i=0}^{p} \theta^{f}_{s,l,i} t^{i} \,,$$

matrix form으로는

$$\hat{\mathbf{y}}^{tr}_{s,l} = \mathbf{T}\theta^{f}_{s,l}, \quad \mathbf{T} = [\mathbf{1}, \mathbf{t}, \ldots, \mathbf{t}^{p}]$$



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

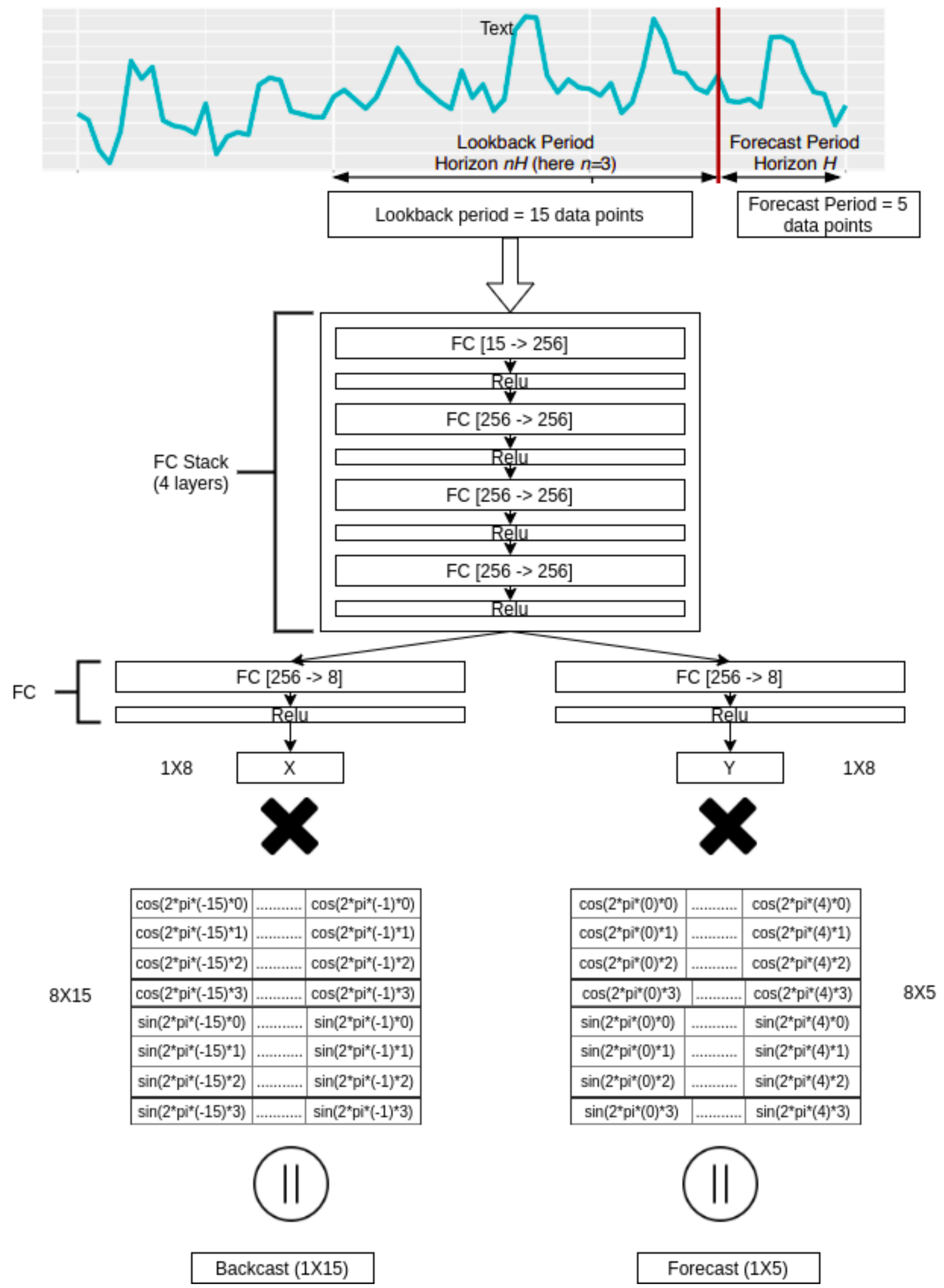
Interpretability

- Seasonality model

$$\hat{\mathbf{y}}_{s,l} = \sum_{i=0}^{\lfloor H/2-1 \rfloor} \theta^{f}_{s,l,i} \cos(2\pi i t) + \theta_{s,l,i+\lfloor H/2 \rfloor} + \sin(2\pi i t)$$

$$\hat{\mathbf{y}}^{season}_{s,l} = \mathbf{S}\theta^{f}_{s,l},$$

$$\mathbf{s} = [\mathbf{1}, \cos(2\pi\mathbf{t}), \ldots, \cos(2\pi\lfloor H/2-1 \rfloor \mathbf{t}), \sin(2\pi\mathbf{t}), \ldots, \sin(2\pi\lfloor H/2-1 \rfloor \mathbf{t}))]$$



https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

RNN

인과형 예측기법

RNN

- RNN(Recurrent Neural Networks)은 ANN 모형의 입력층에서 출력층으로의 일방향 데이터 전달 구조에 변형을 가하여 은닉층 또는 출력층에서 입력층으로의 피드백 구조를 도입, 네트워크가 기억력을 가지게 만들어 언어 데이터 처리에 우수한 능력을 보임

- 따라서, RNN 모형은 일반적으로 데이터에 존재하는 시간 의존성(Temporal Dependence)을 학습하는 기능을 지니며, 시계열 데이터에 내재되어 있는 temporal structure를 잘 파악해냄

- 특히 LSTM은 데이터에 존재하는 장기간에 걸친 연관관계를 잘 파악해낸다.

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bigdataai&logNo=222043157198

DeepAR

- 개념
- 특장점
- 한계점
- 알고리즘 설명
- Parameter 요약
- 현업 유의점
- Code - 별첨

DeepAR 개요

- **수없이 많은 시계열을 동시에 예측해야 하는 상황이 더욱 빈번해짐**
  - 기존에도 여러 시계열의 정보를 활용하는 multivariate timeseries 기법들이 있었지만, scalable하지 않음
  - 이를 위해 주로 가장 중요한 품목들 위주로 분석을 진행하거나 상위 카테고리로 그룹화 한후 분석 진행

- **기존의 전통적 시계열방법은 판단과 수작업이 많이 들어감**
  - 시계열의 정상성, 단위근 파악 등
  - 특성 파악후 많은 전처리 작업이 필요
    - Ex) 추세 제거, 차분, 지시변수 생성 등
  - 이러한 작업들 역시 scalability에 영향을 줌

Figure 1: Log-log histogram of the number of items versus number of sales for the 500K time series of ec, showing the scale-free nature (approximately straight line) present in the ec dataset (axis labels omitted due to the non-public nature of the data).

https://www.youtube.com/watch?v=okyo61ZZivA

특장점

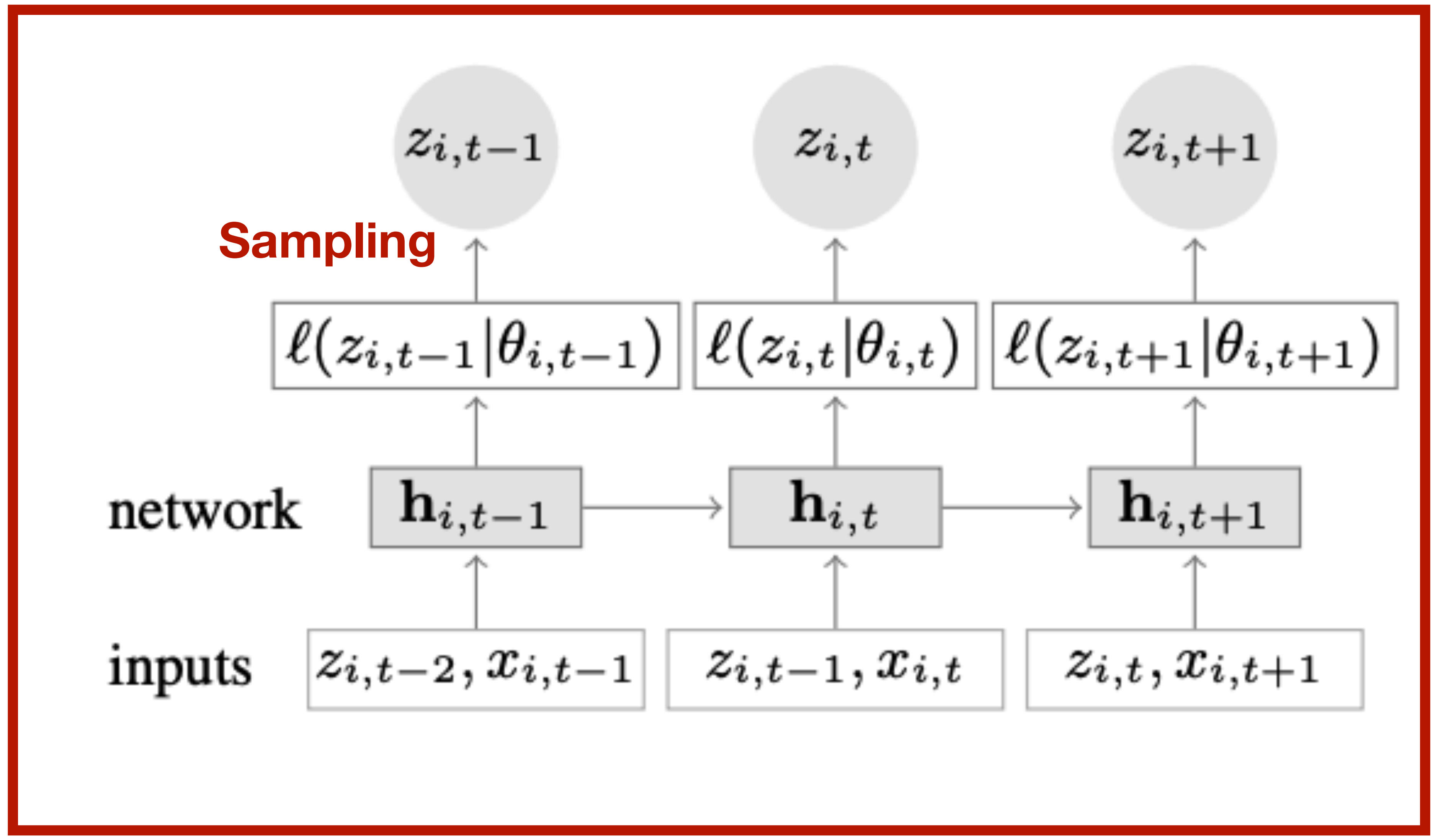
- **Feature engineering의 수고가 줄어듬**
  - 원래 시계열을 거의 그대로 입력 (내부적으로 autoscaling 적용)
- **확률적인 예측이 가능**
  - 신뢰구간 제공이 가능
  - loss의 종류에 따라서, 기대값이 아니라, 기대 quantile을 구할 수 있음
- **적은 데이터에도 효과가 있음**
  - 수많은 시계열의 정보를 빌려올 수 있음
  - 특히 서로 다른 scale(혹은 분포)의 시계열을 동시 예측 가능
- **다양한 정보를 모델링 할 수 있음**
  - Static한 variable (독립변수)
  - 기타 시계열 혹은 미래의 알려진 값까지 모델링이 가능
- **다양한 likelihood를 고려하여, 다양한 형태의 데이터를 모델링할 수 있음**
  - Binary response, Poison response

Figure 1: Log-log histogram of the number of items versus number of sales for the 500K time series of ec, showing the scale-free nature (approximately straight line) present in the ec dataset (axis labels omitted due to the non-public nature of the data).

https://www.youtube.com/watch?v=okyo61ZZivA

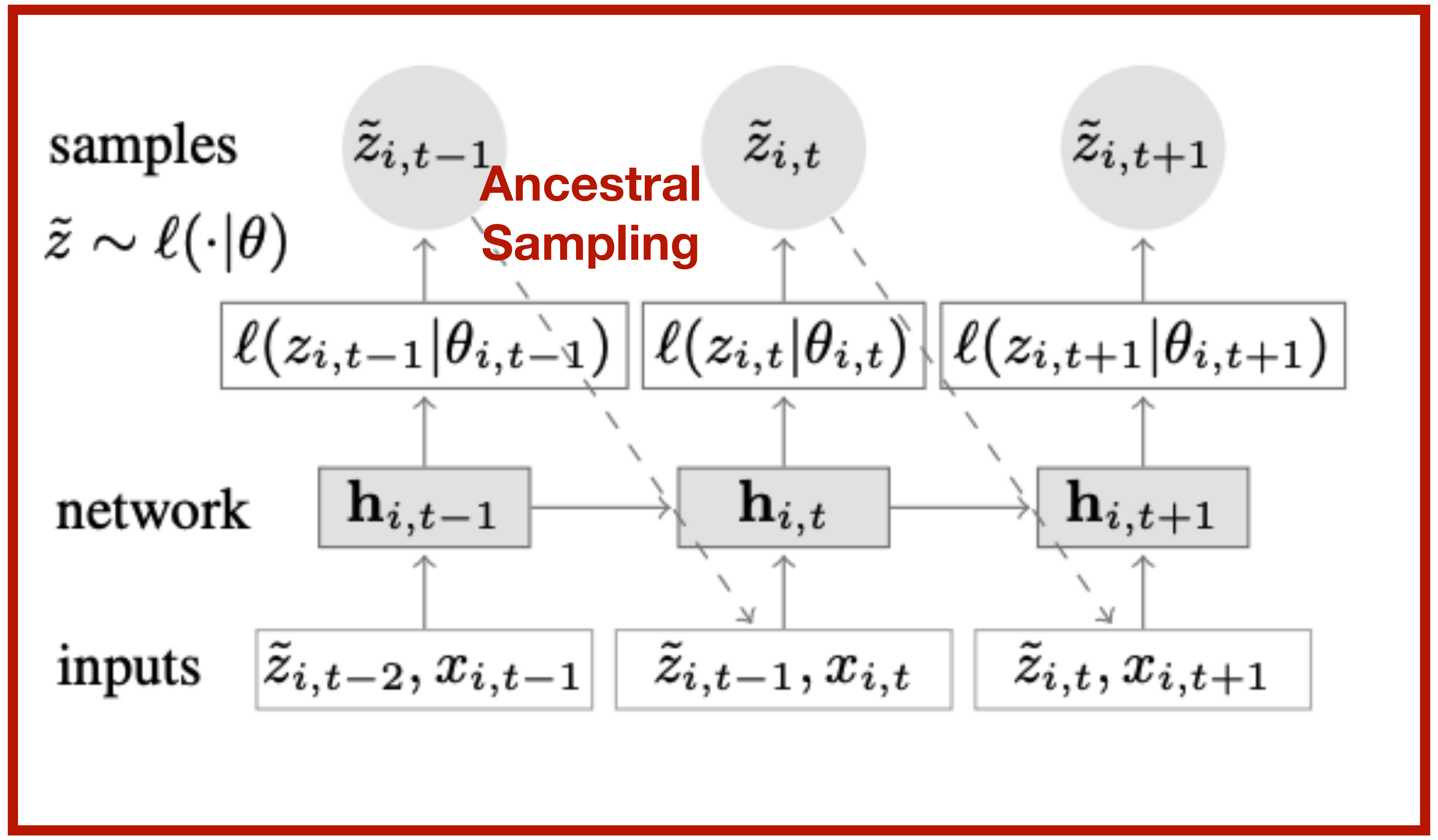
모형 상세

Sequence to sequence 구조



Encoder

Training Phase



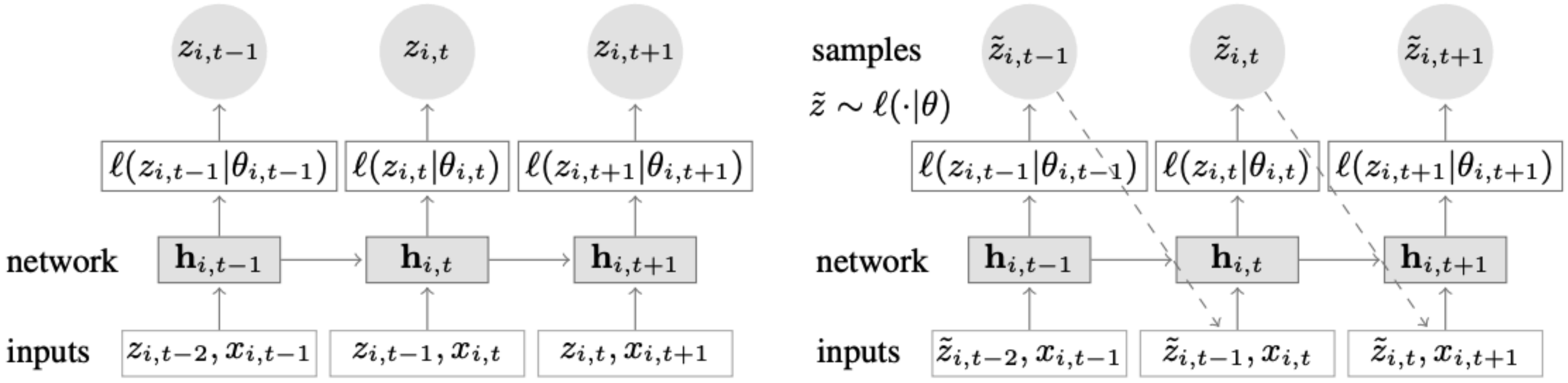
Decoder

Prediction Phase

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Likelihood



- 신경망은 시계열 정보, z,의 likelihood, $l(z|\theta)$,의 paramter를 예측하는 데 활용
- 논문에서는 두개의 likelihood를 사용
  - Gaussian likelihood

Softplus activation

$$l_G(z\,|\,\mu,\sigma) = (s\pi\sigma^2)^{-1/2}\exp\left(-(z-\mu)^2/(2\sigma^2)\right), \quad \mu(\mathbf{h}_{i,t}) = \mathbf{w}_\mu^T\mathbf{h}_{i,t}, \quad \sigma(\mathbf{h}_{i,t}) = \log(1+\exp(\mathbf{w}_\sigma^T\mathbf{h}_{i,t}+b\sigma))$$

  - Negative binomial likelihood: positive count

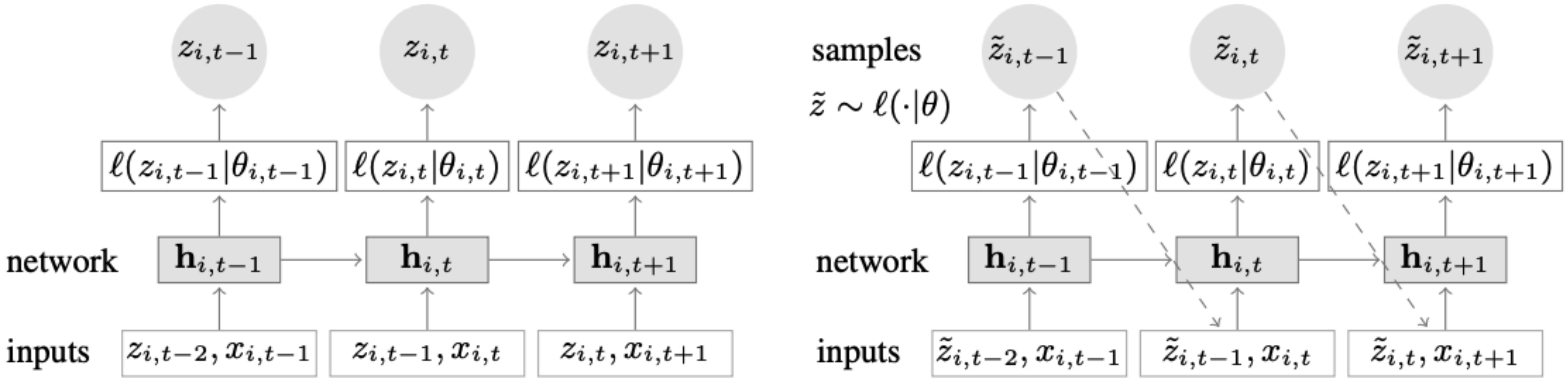
$$l_{NB}(z\,|\,\mu,\alpha) = \frac{\Gamma\left(z+\frac{1}{\alpha}\right)}{\Gamma(z+1)\Gamma\left(\frac{1}{\alpha}\right)}\left(\frac{1}{1+\alpha\mu}\right)^{1/\alpha}\left(\frac{\alpha\mu}{1+\alpha\mu}\right)^{z}, \quad \mu(\mathbf{h}_{i,t}) = \log(1+\exp(\mathbf{w}_\mu^T\mathbf{h}_{i,t}+b\mu)), \quad \alpha(\mathbf{h}_{i,t}) = \log(1+\exp(\mathbf{w}_\alpha^T\mathbf{h}_{i,t}+b\alpha))$$

α: Shape parameter

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Likelihood



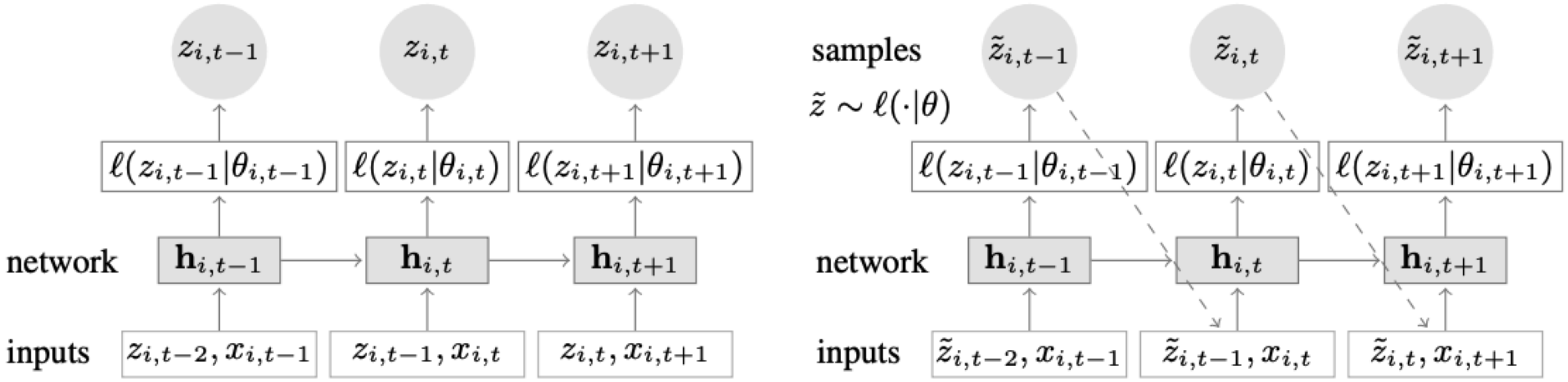
- Input:
  - 과거의 관측치 값: $z_1, \ldots, z_{t_0-1}$
  - 독립변수들: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$
- Output: 추정된 결합확률 분포: $P(Z_{t_0}, Z_{t_0+1}, \ldots, Z_T)$
- 정해진 크기의 임의의 window 내의 값들을 이용하여, MLE를 구하는 방식으로 학습
- 결국 likelihood는 다음과 같음

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \log l(z_{i,t} \mid y_t(\mathbf{h}_{i,t}))$$

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Scale problem, long-tail problem



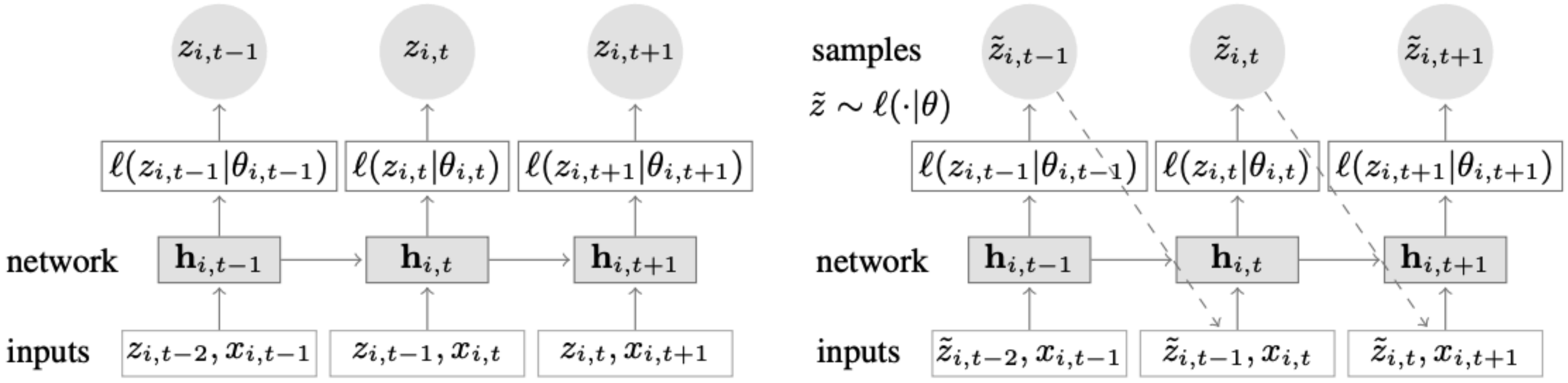
- 두가지 문제가 존재함
  - 시계열의 scale이 크게 다르다. (대다수의 품목이 아주 적게 팔린다)
  - 주로 관심을 가져야 할 시계열의 개수가 크게 적다. (소수의 품목만이 아주 많이 팔린다.)

- 두가지 해결책을 제시
  - Automatic Scaling
  - Weighted sampling to counter-balance power-law behavior

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Automatic scaling



Automatic scaling:

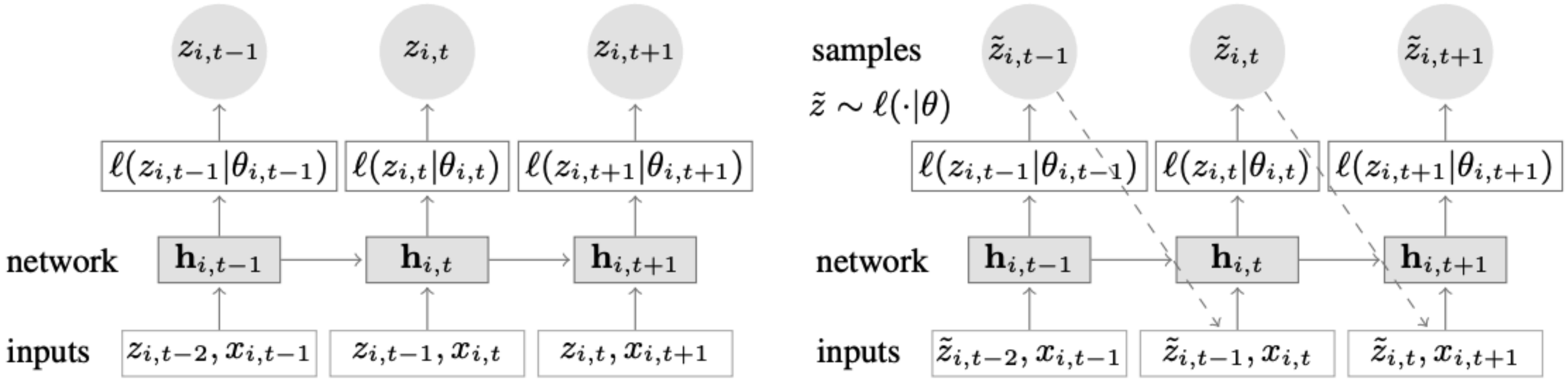
- ***i*번째 시계열의 자기상관 입력인** z 를 v_i라는 scaling factor를 이용해서 scaling함
- v_i는 단순히 시계열의 평균값으로, 논문에서는 다음의 식을 활용 (Heuristic method)

$$v_i = 1 + \frac{1}{t_0} \sum_{t=1}^{t_0} z_{i,t}$$

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

모형 상세

Long tail



Handling heavy tail distribution (Weighted sampling):

- Scale에 비례해서 학습셋에 포함될 확률을 높여줌

https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

Attention Mechanism

Seq2seq & its limit

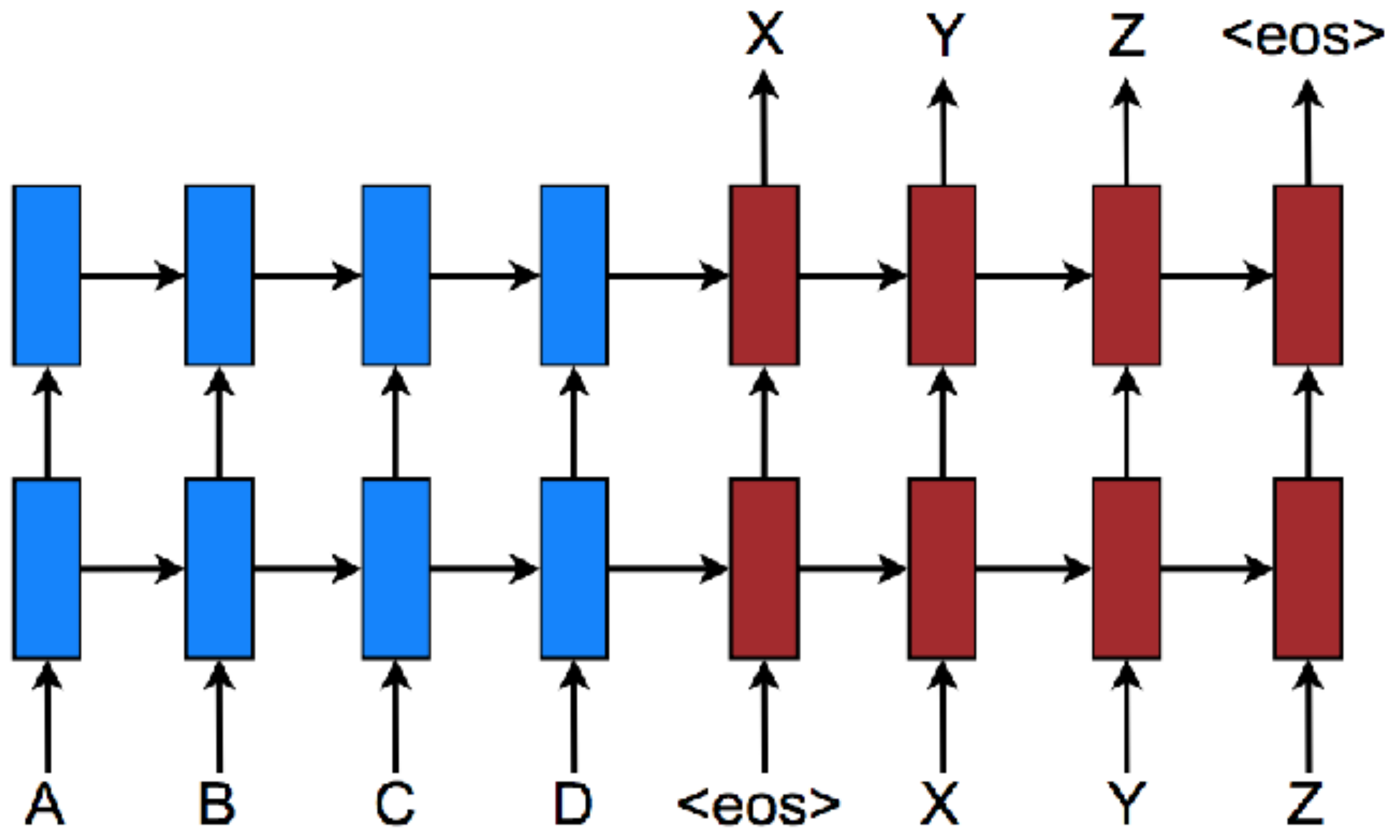


Figure 1: **Neural machine translation** – a stacking recurrent architecture for translating a source sequence `A B C D` into a target sequence `X Y Z`. Here, `<eos>` marks the end of a sentence.

When the source sequence is too long and contains multiple information-rich phrases apart from each other

What if we had a mechanism to allow the decoder to selectively (dynamically) focus on the information-rich phrases in the source sequence?

Attention mechanism

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

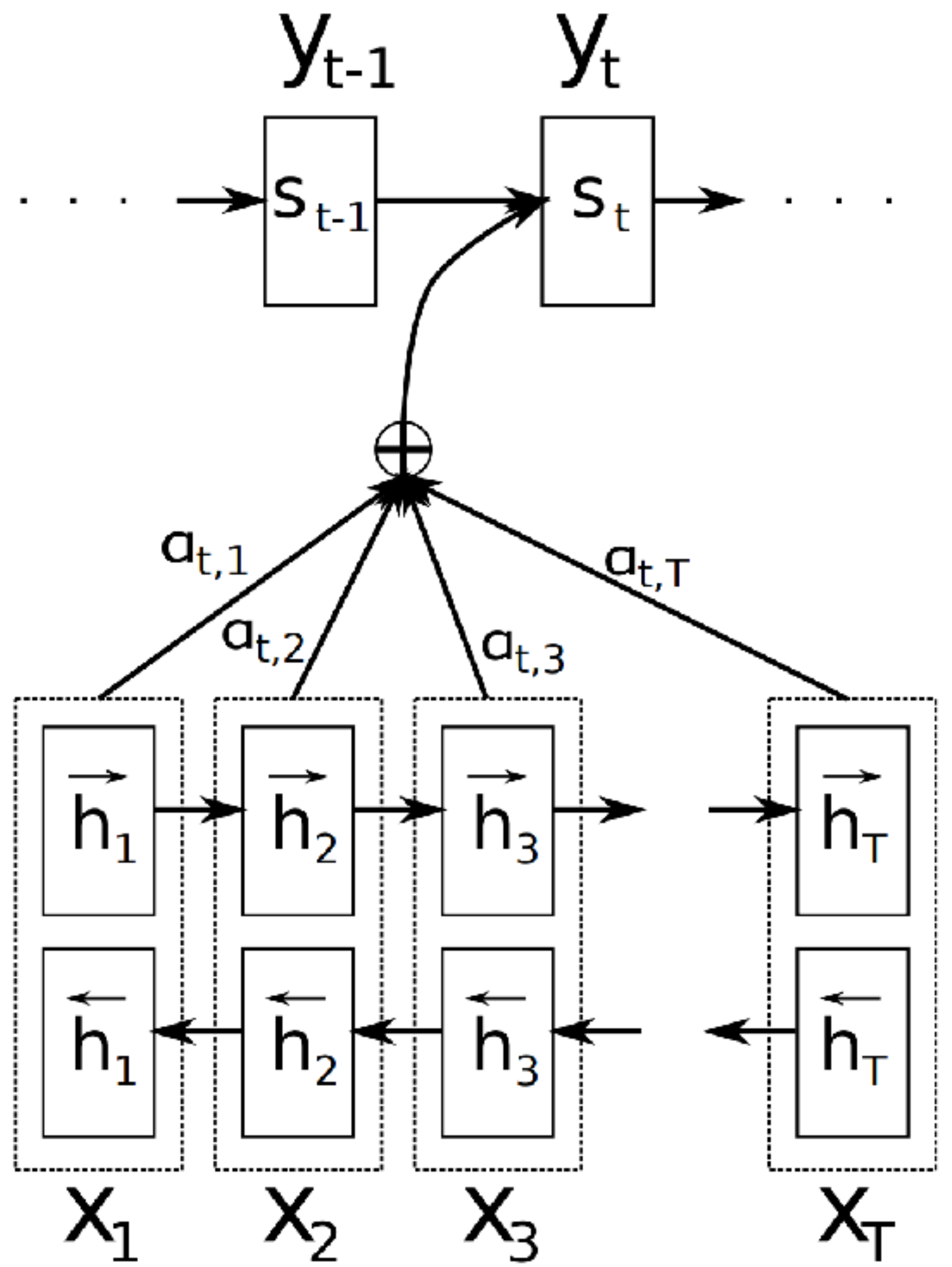


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Attention mechanism can do

Long term memories - attending to memories

- Dealing with gradient vanishing problem

Exceeding limitations of a global representation

- Attending/focusing to smaller parts of data
- patches in images - words or phrases in sentences

http://download.mpi-inf.mpg.de/d2/mmalinow-slides/attention_networks.pdf

Attention mechanism can do

Decoupling representation from a problem

- Different problems required different sizes of representations
- LSTM with longer sentences requires larger vectors

Overcoming computational limits for visual data

- Focusing only on the parts of images
- Scalability independent of the size of images

Adds some interpretability to the models (error inspection)

How to implement Attention

Attention mechanism

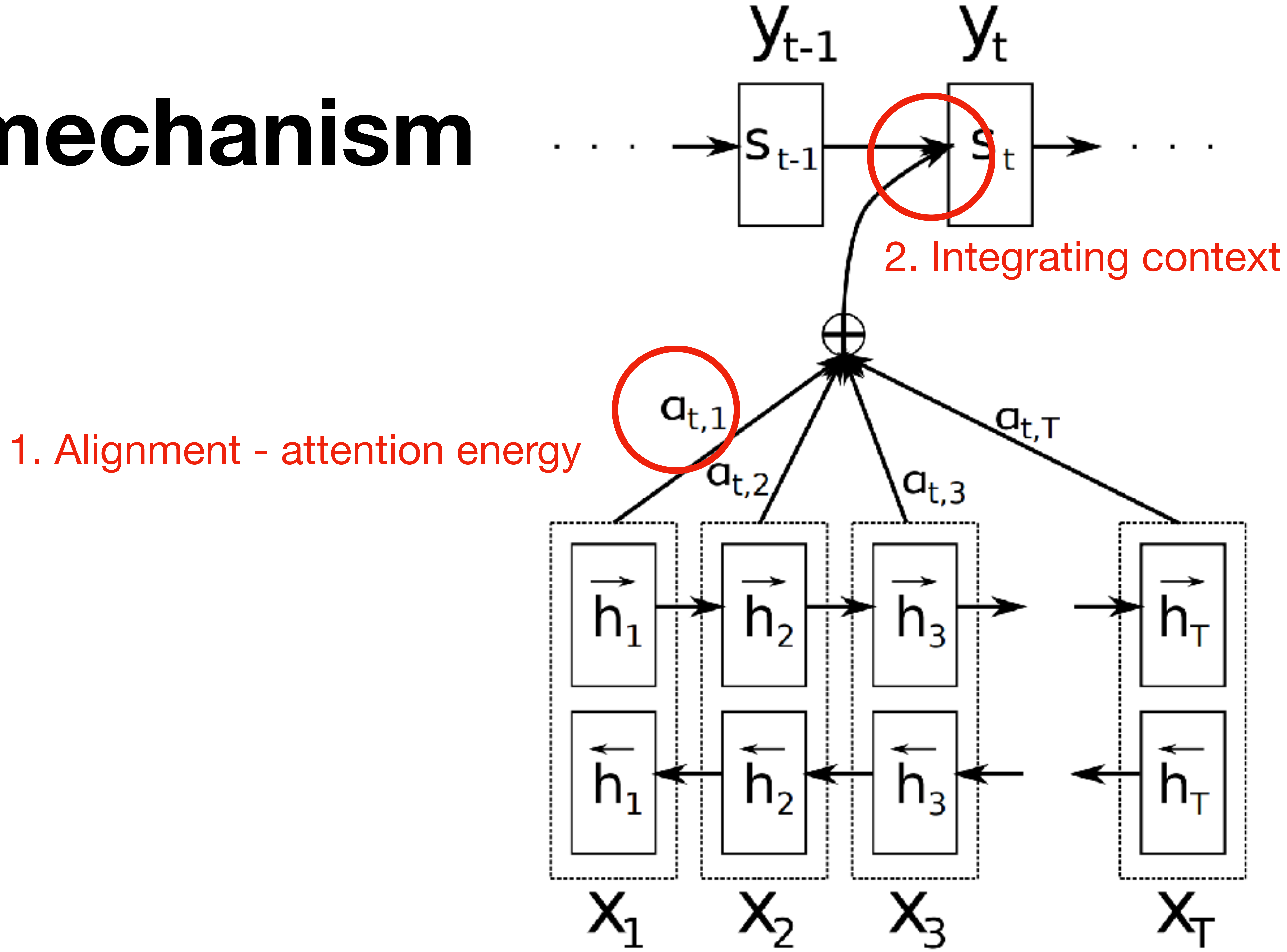


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Université de Montréal
KyungHyun Cho Yoshua Bengio*

Alignment

Neural Machine Translation by Jointly Learning to Align and Translate
(https://arxiv.org/pdf/1409.0473.pdf)

A.1.2 ALIGNMENT MODEL

The alignment model should be designed considering that the model needs to be evaluated $T_x \times T_y$ times for each sentence pair of lengths T_x and T_y. In order to reduce computation, we use a single-layer multilayer perceptron such that

$$a(s_{i-1}, h_j) = v_a^\top \tanh \left(W_a s_{i-1} + U_a h_j \right),$$

where $W_a \in \mathbb{R}^{n \times n}, U_a \in \mathbb{R}^{n \times 2n}$ and $v_a \in \mathbb{R}^n$ are the weight matrices. Since $U_a h_j$ does not depend on i, we can pre-compute it in advance to minimize the computational cost.

Alignment

These are basically ***unnormalized scores*** of alignment between decoder state ***s*** and the hidden states H.

The mapping from $(\{h_i\}_{i \in [0,T]}, s_j)$ to the attention energies is known as the *alignment* model.

The idea of a global attentional model is to consider all the hidden states of the encoder when deriving the context vector c_t. In this model type, a variable-length alignment vector $\boldsymbol{a}_t$, whose size equals the number of time steps on the source side, is derived by comparing the current target hidden state $\boldsymbol{h}_t$ with each source hidden state $\bar{\boldsymbol{h}}_s$:

$$\begin{aligned} \boldsymbol{a}_t(s) &= \operatorname{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \qquad (7) \\ &= \frac{\exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

Effective Approaches to Attention-based Neural Machine Translation
(https://arxiv.org/pdf/1508.04025.pdf)

Alignment models

Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Multiplicative model

Additive model

Effective Approaches to Attention-based Neural Machine Translation
(https://arxiv.org/pdf/1508.04025.pdf)

Alignment models

Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Multiplicative model

Additive model

Equivalent to

$$v_a^T \tanh\left(\mathbf{U}_a \mathbf{h}_t + \mathbf{W}_a \bar{\mathbf{h}}_s\right)$$

Effective Approaches to Attention-based Neural Machine Translation
(https://arxiv.org/pdf/1508.04025.pdf)

Alignment models

Multiplicative Models

$$score(h_i, s_j) = \begin{cases} h_i^T s_j & dot \\ h_t^T W_a s_j & general \end{cases}$$

Additive Models

$$score(h_i, s_j) = \begin{cases} v_a^T \tanh(u_a h_i + W_a s_j) & linear \\ v_a^T \tanh(W_a [h_i ; s_j]) & general \end{cases}$$

In general, the performance of multiplicative and additive functions are similar but the multiplicative function is faster and more space-efficient.

Integrating context

Specifically, given the target hidden state $\boldsymbol{h}_t$ and the source-side context vector $\boldsymbol{c}_t$, we employ a simple concatenation layer to combine the information from both vectors to produce an attentional hidden state as follows:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \tag{5}$$

The attentional vector $\tilde{\boldsymbol{h}}_t$ is then fed through the softmax layer to produce the predictive distribution formulated as:

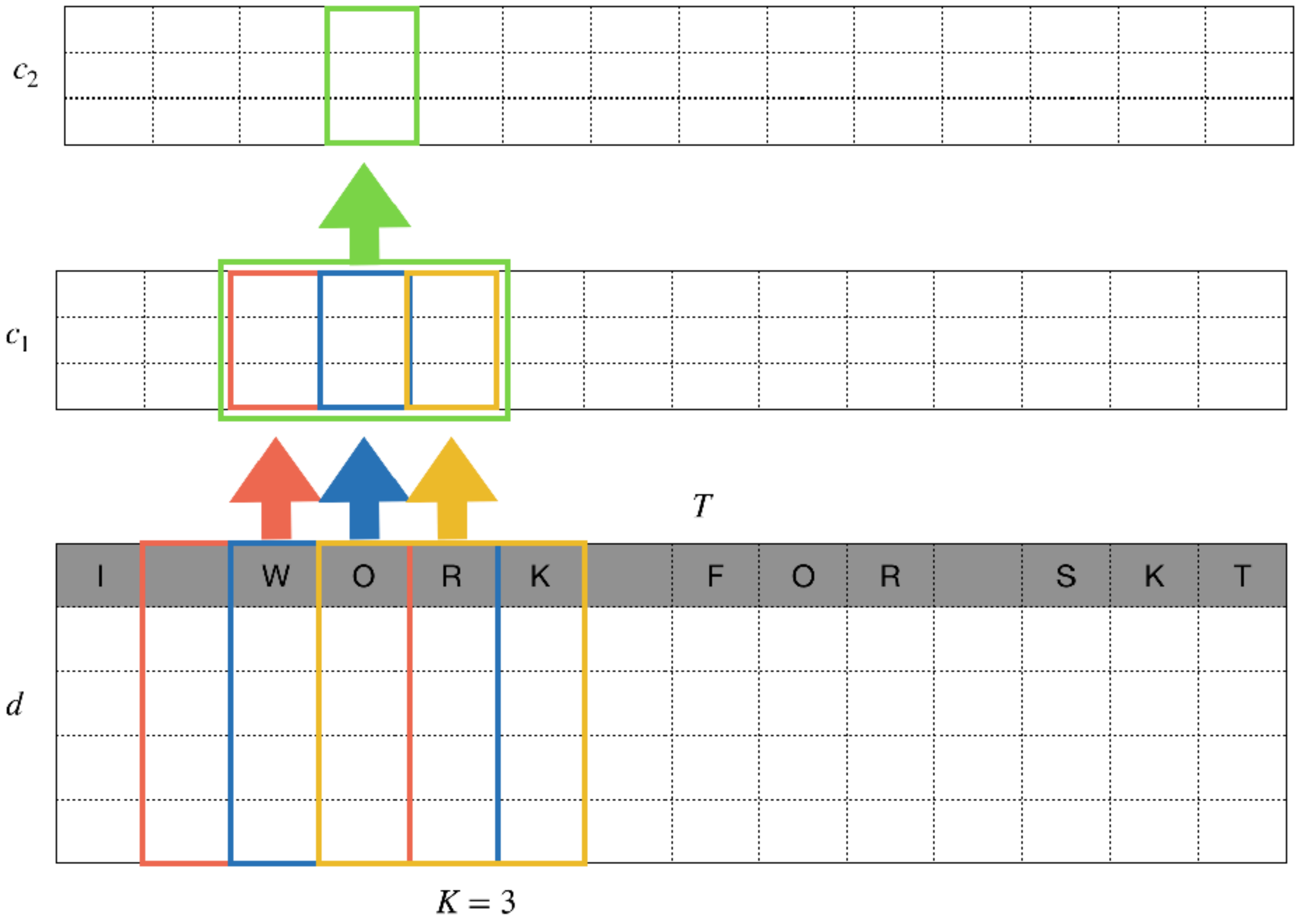
$$p(y_t|y_{<t}, x) = \mathrm{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t) \tag{6}$$

We now detail how each model type computes the source-side context vector $\boldsymbol{c}_t$.

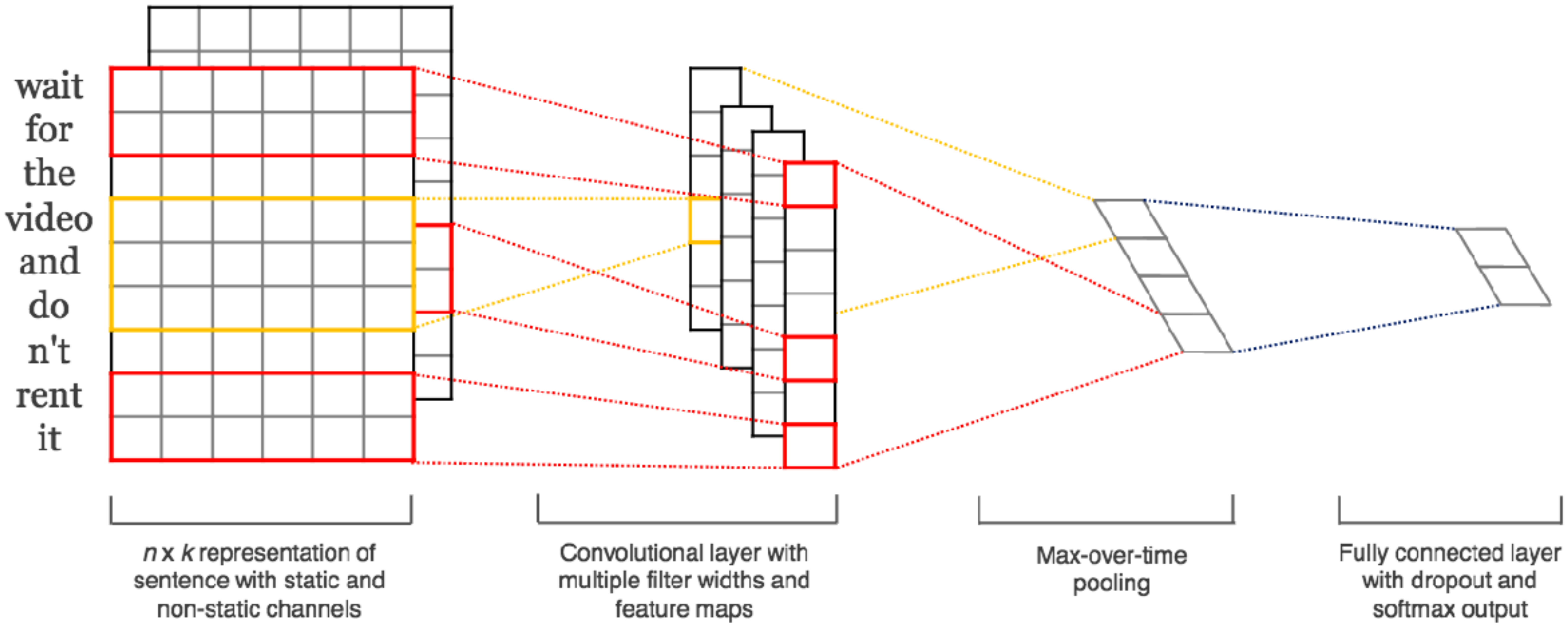
Self Attention

N-GRAM & CNN

***n*-gram is a contiguous sequence of *n* items from a given sample of text**



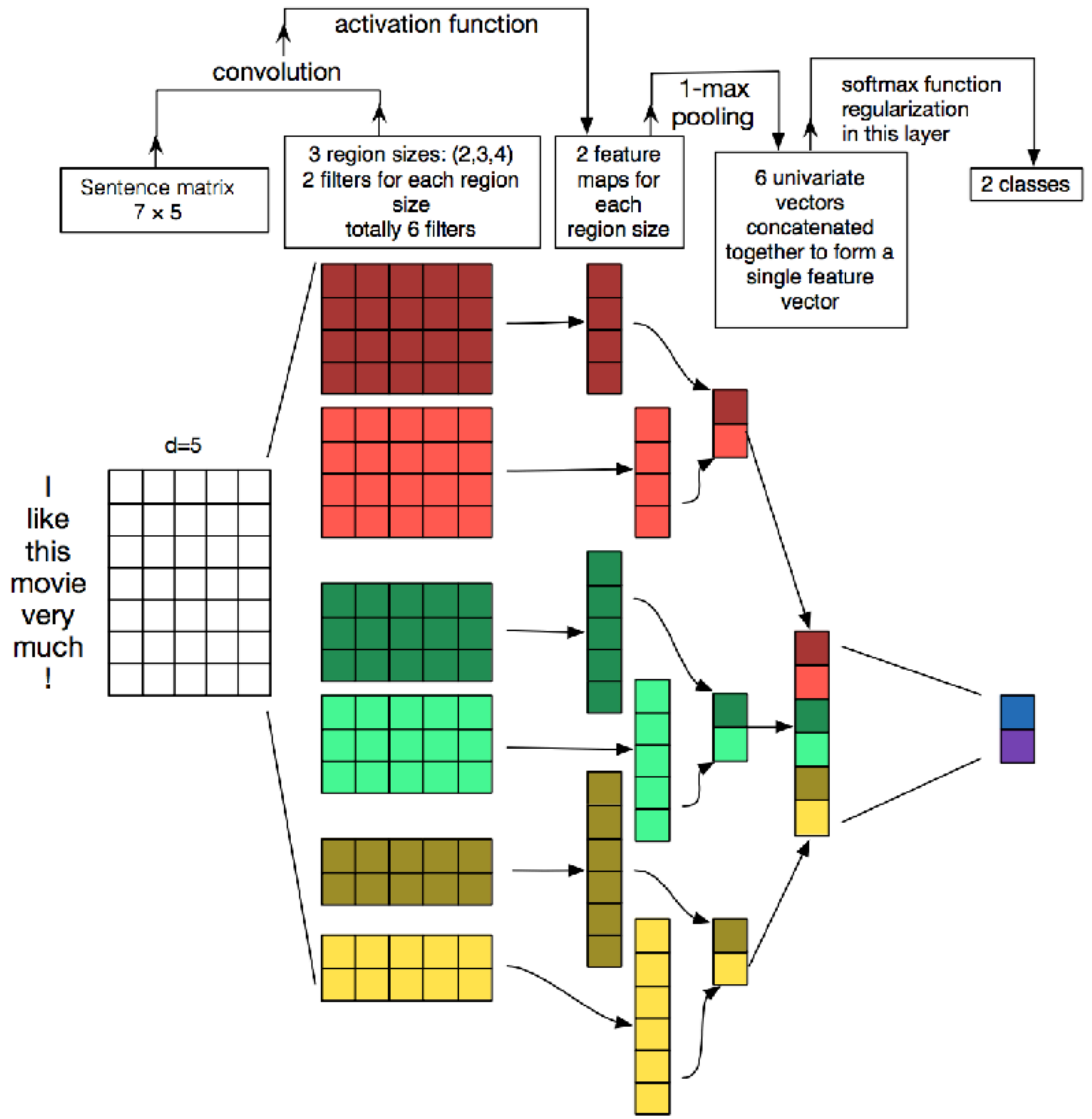
CNN for NLP



Yoon Kim (2014)

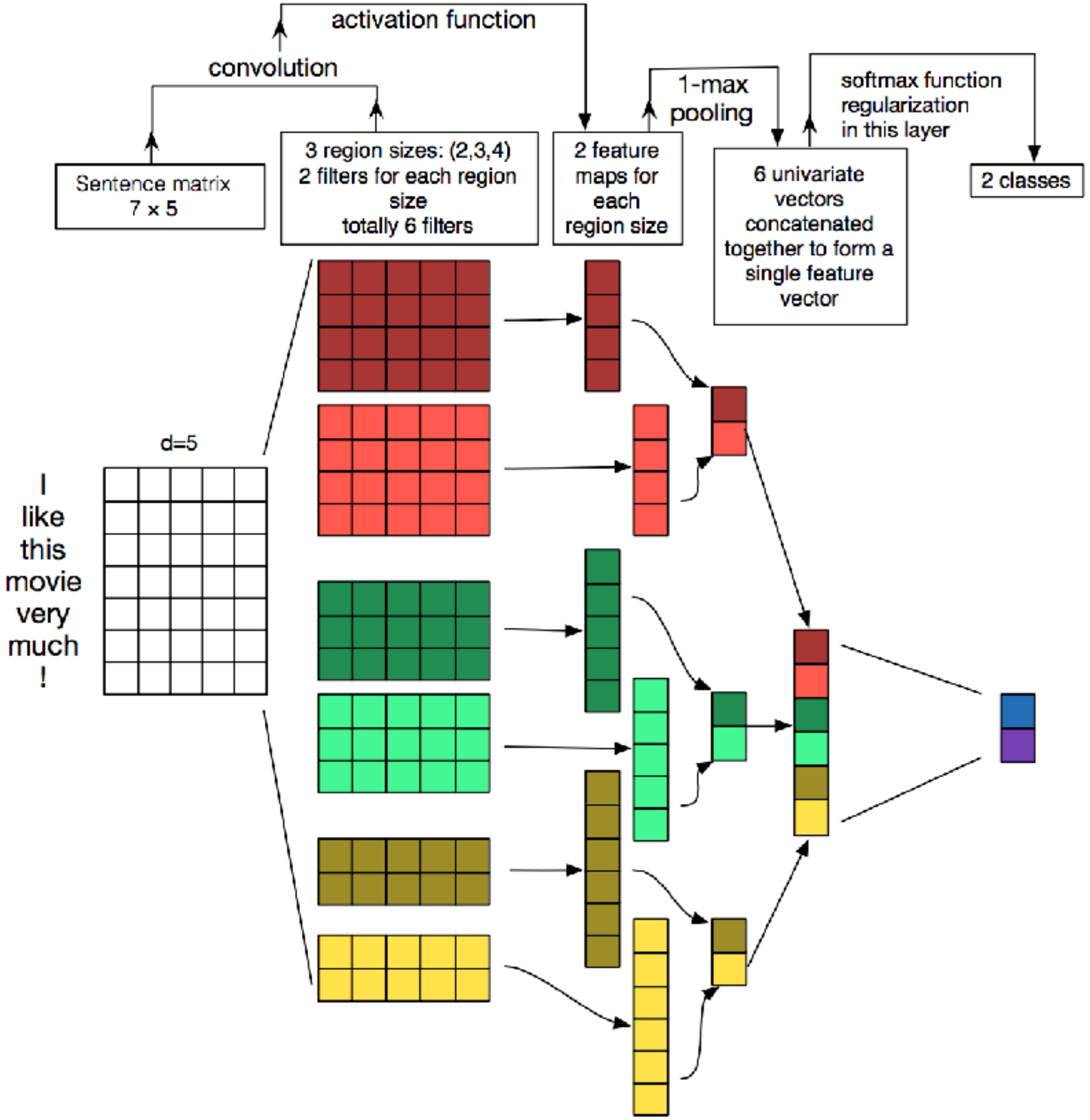
https://arxiv.org/abs/1408.5882

CNN for NLP



[A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification]

CNN for NLP

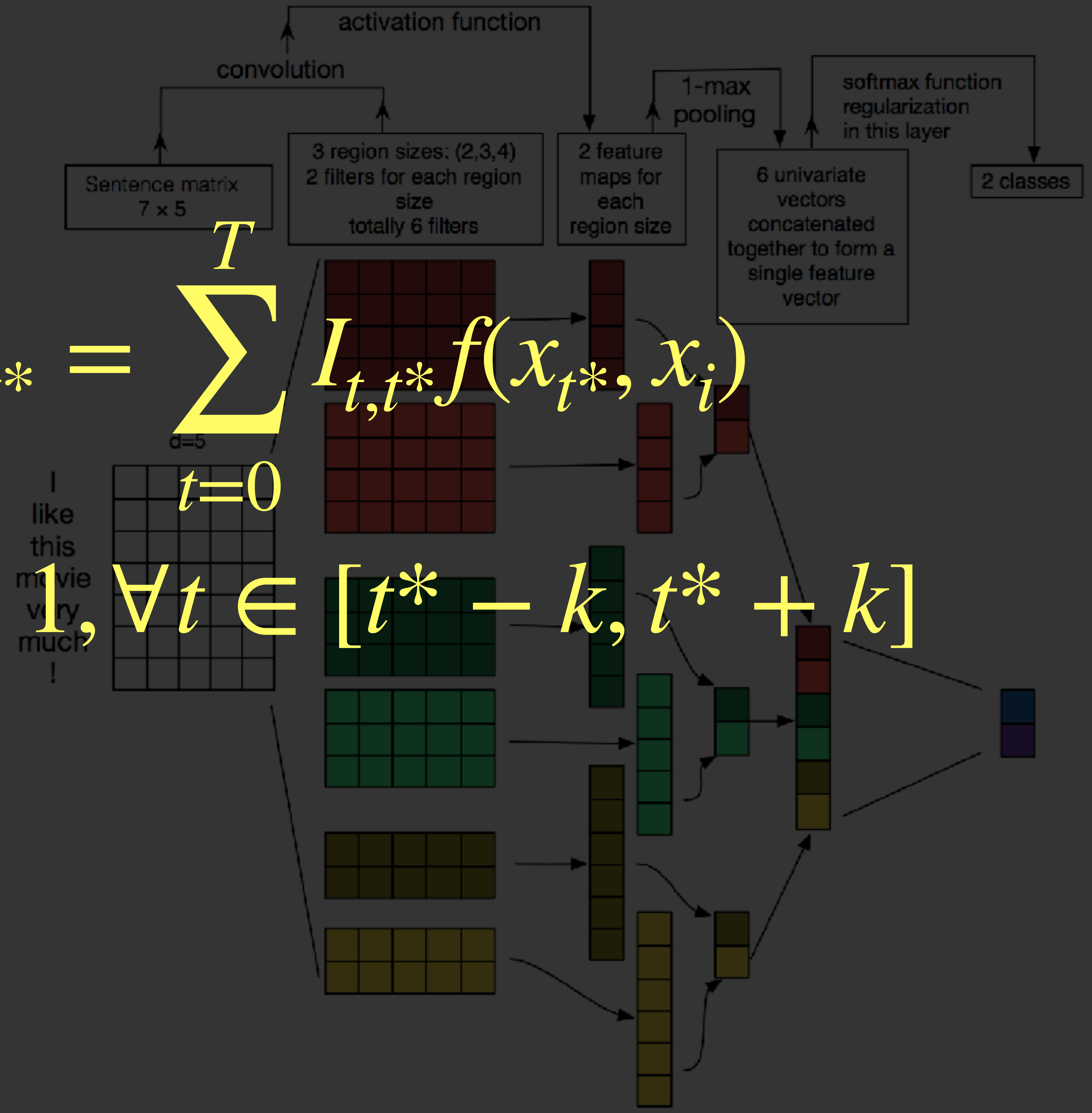


Wants to increase Receptive fields?

Dilated convolution (wavenet)

[A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification]

CNN for NLP



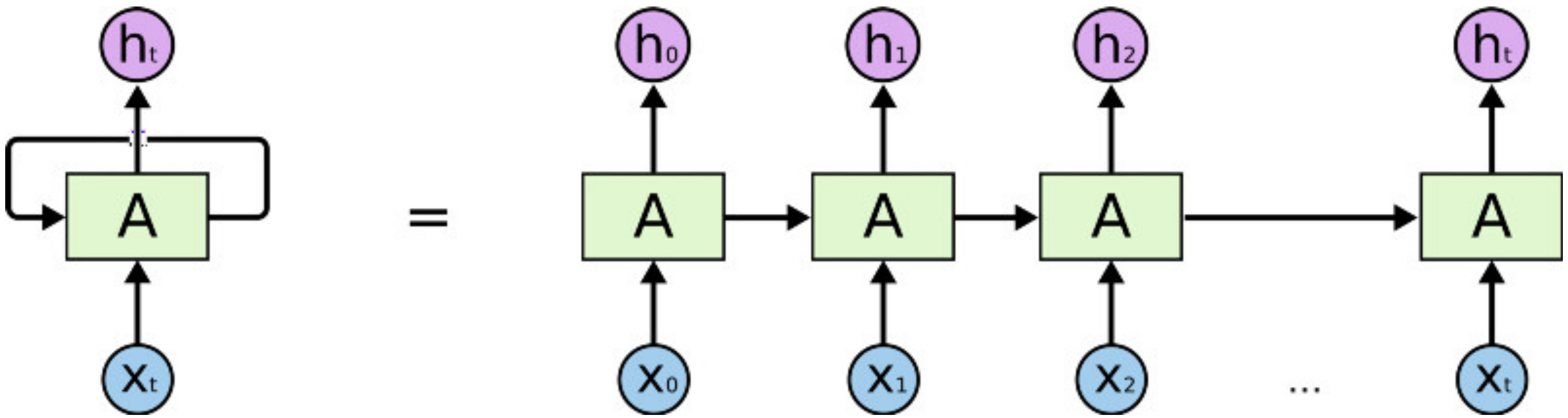
$$h_{t^*} = \sum_{t=0}^{T} I_{t,t^*} f(x_{t^*}, x_i)$$

$$I_{t,t^*} = 1, \forall t \in [t^* - k, t^* + k]$$

[A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification]

RNN for NLP

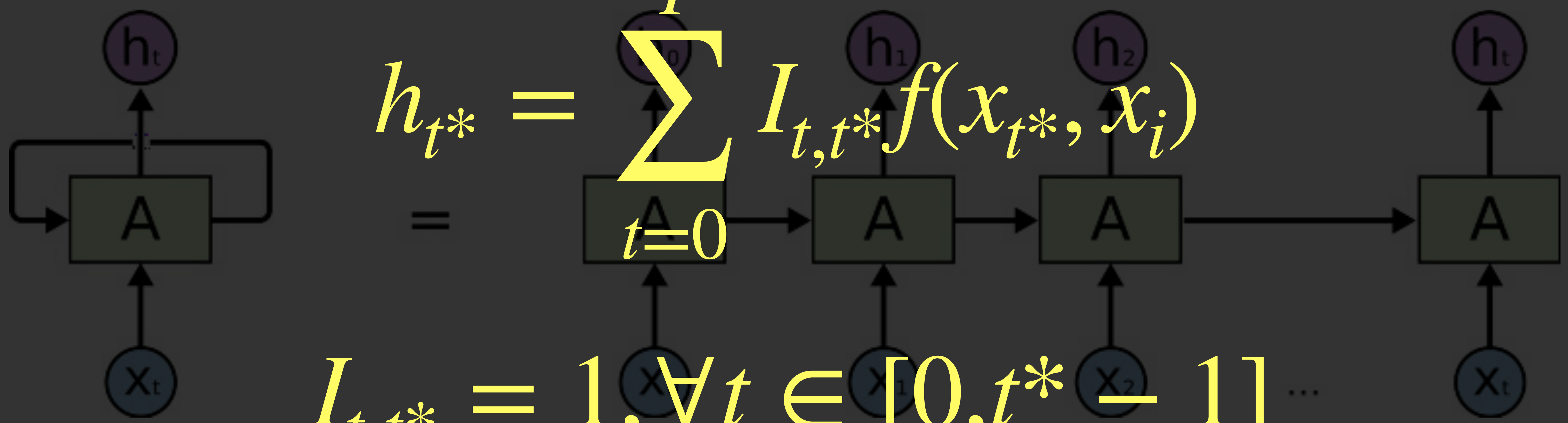
Unfolding **Computational Graphs**



$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$

RNN for NLP

Unfolding **Computational Graphs**



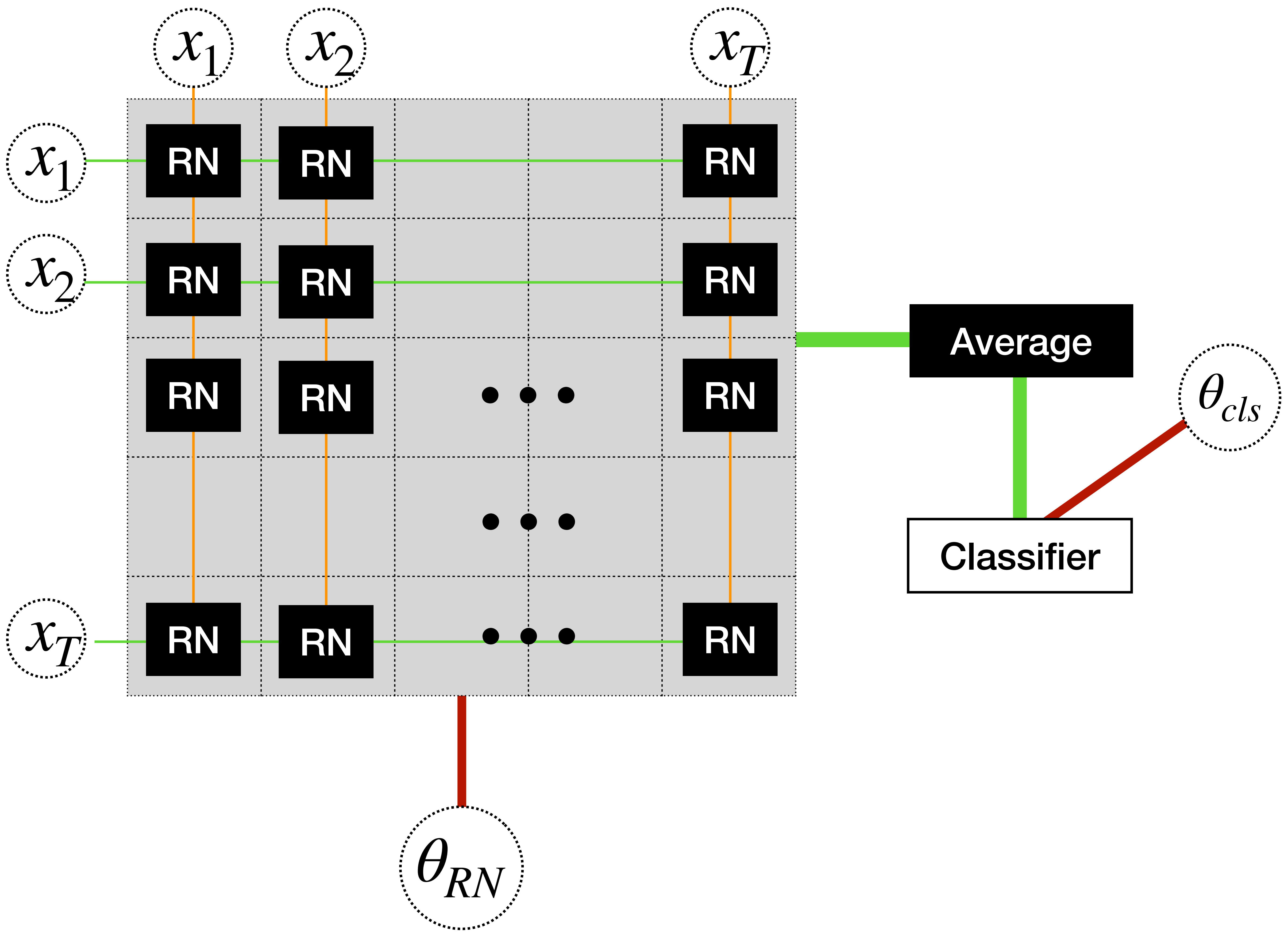
$$h_{t^*} = \sum_{t=0}^{T} I_{t,t^*} f(x_{t^*}, x_i)$$

$$I_{t,t^*} = 1, \forall t \in [0, t^* - 1]$$

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$

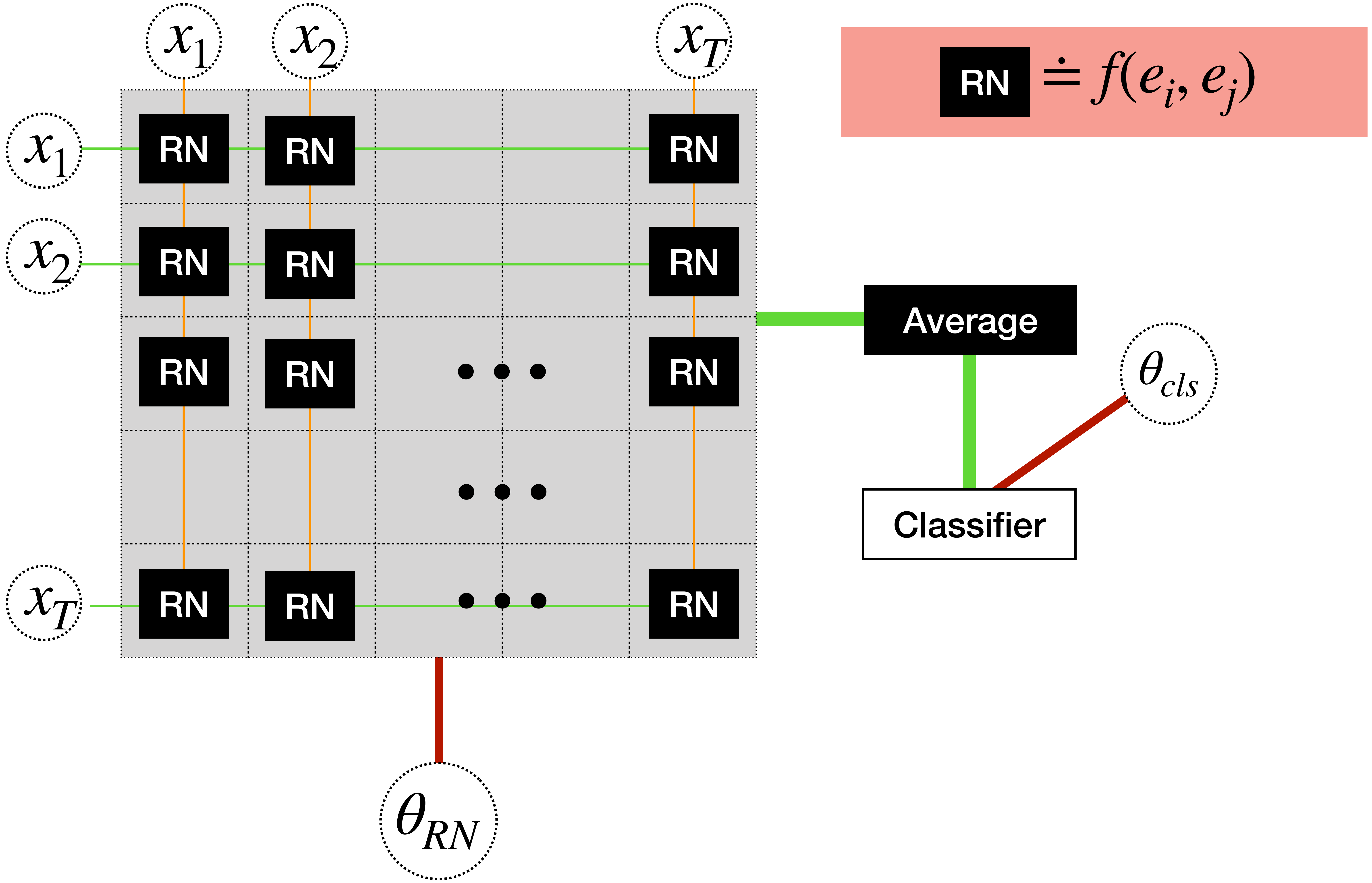
Relation Network (Skip bigram)

To summarize sentence based on pairwise relationship



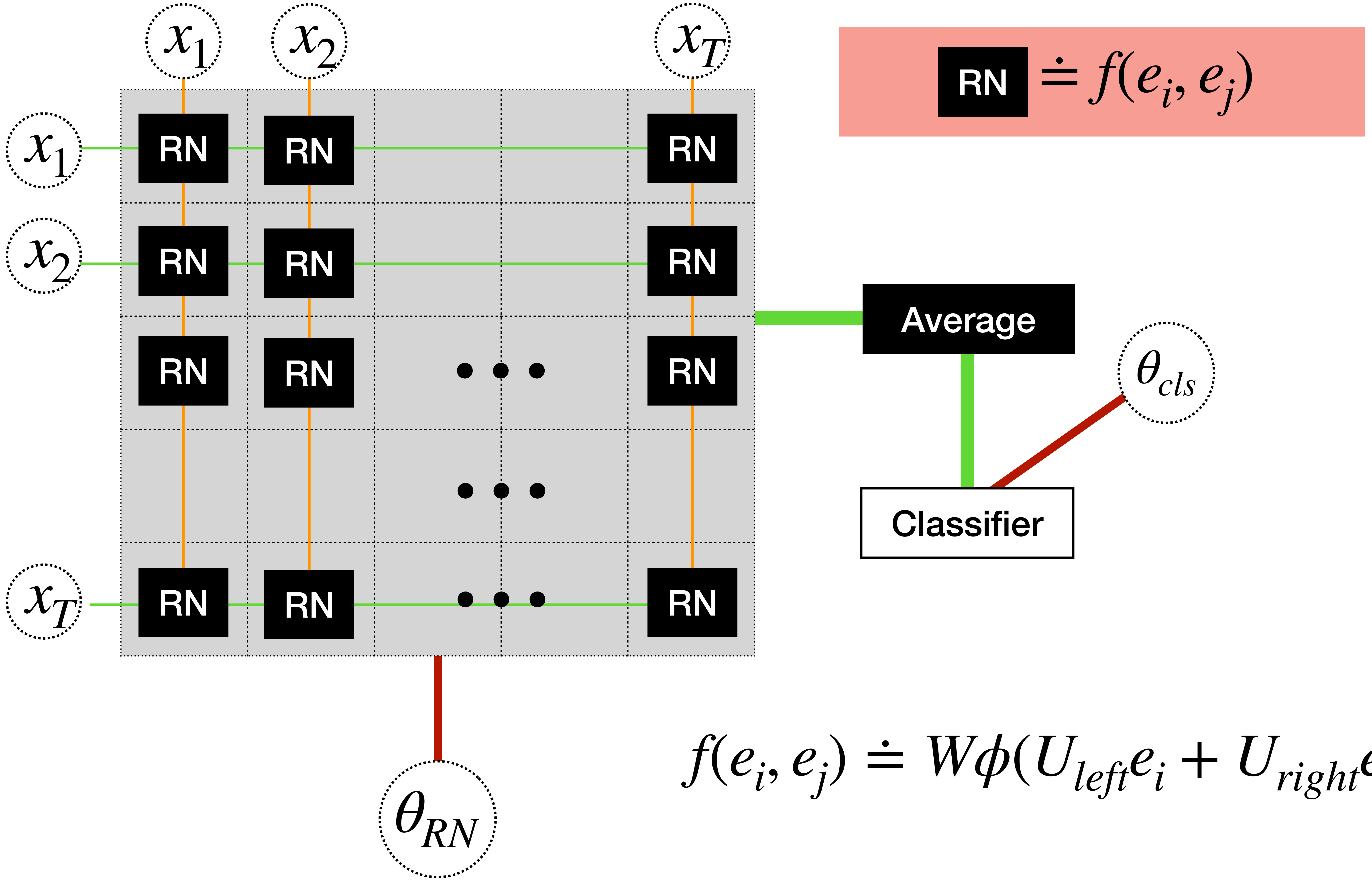
Relation Network (Skip bigram)

To summarize sentence based on pairwise relationship



Relation Network (Skip bigram)

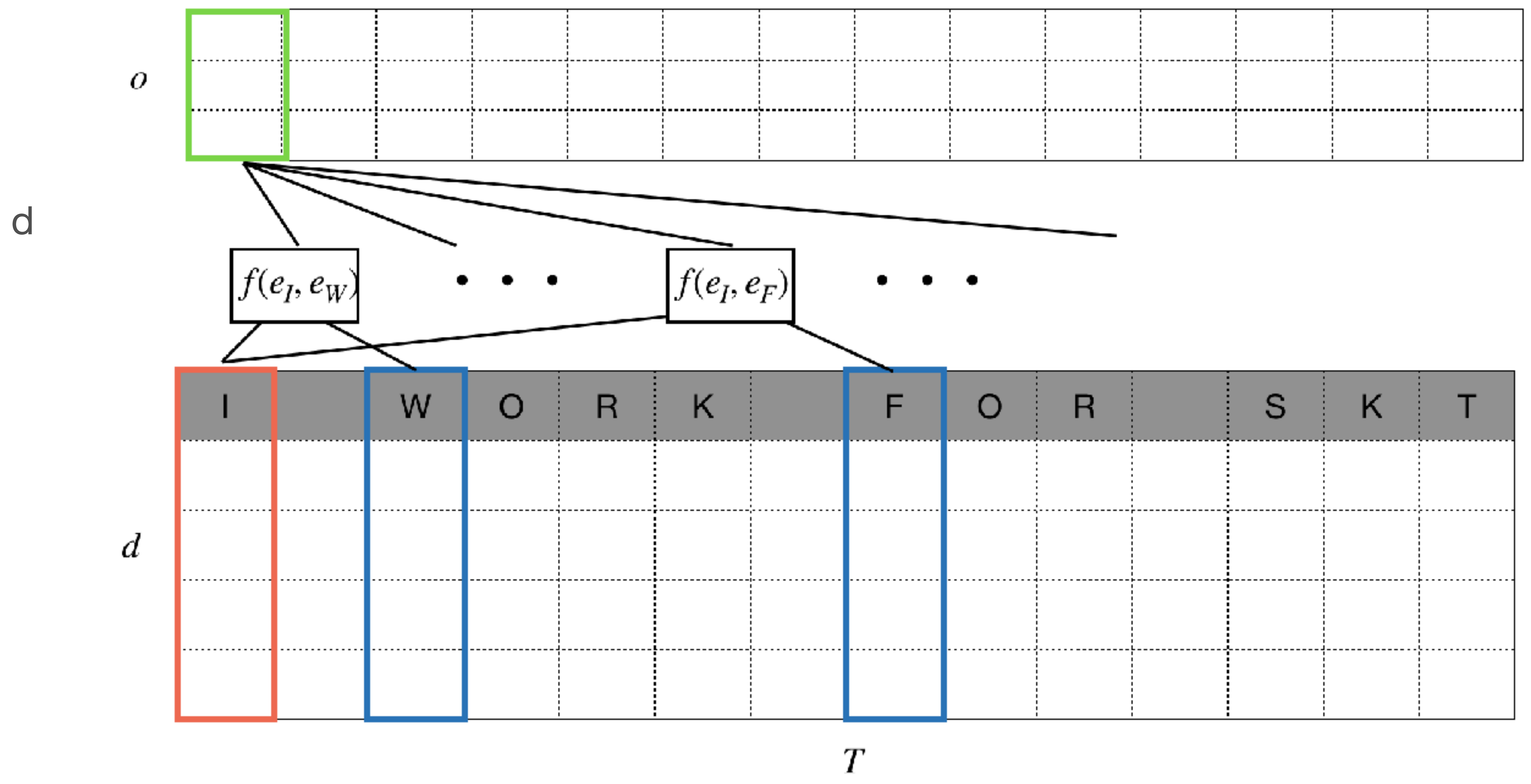
To summarize sentence based on pairwise relationship



$$f(e_i, e_j) \doteq W\phi(U_{left}e_i + U_{right}e_j)$$

Skip bigram - Relation Net

A pair of words may give us more clear information about the sentence.
ex) ‘like’ in ‘I like’ vs ‘like’ in ‘like this’

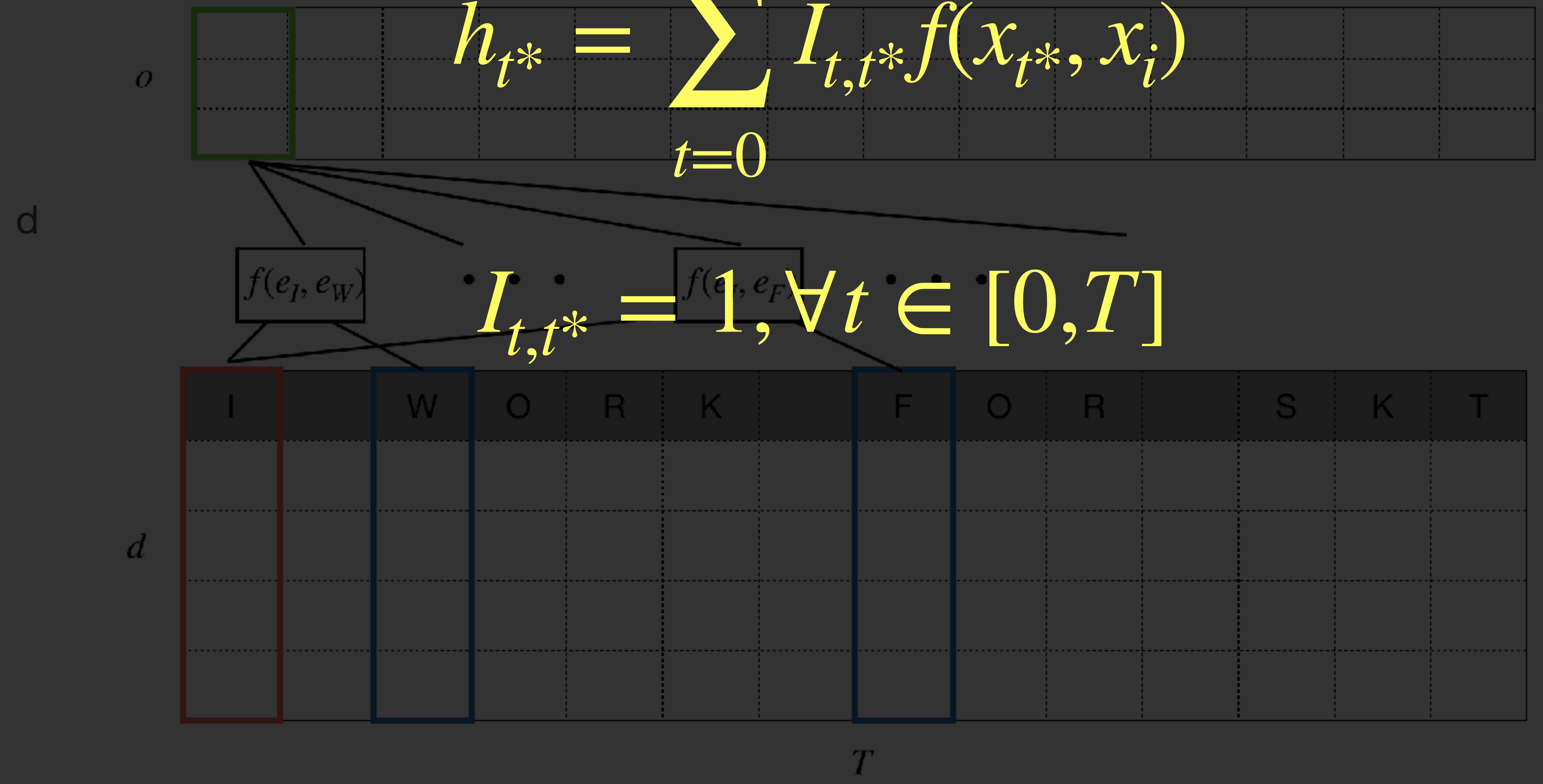


Skip bigram - Relation Net

A pair of words may give us more clear information about the sentence.
ex)'like' in 'I like' vs 'like' in 'like this'

$$h_{t*} = \sum_{t=0}^{T} I_{t,t*} f(x_{t*}, x_i)$$

$$I_{t,t*} = 1, \forall t \in [0,T]$$

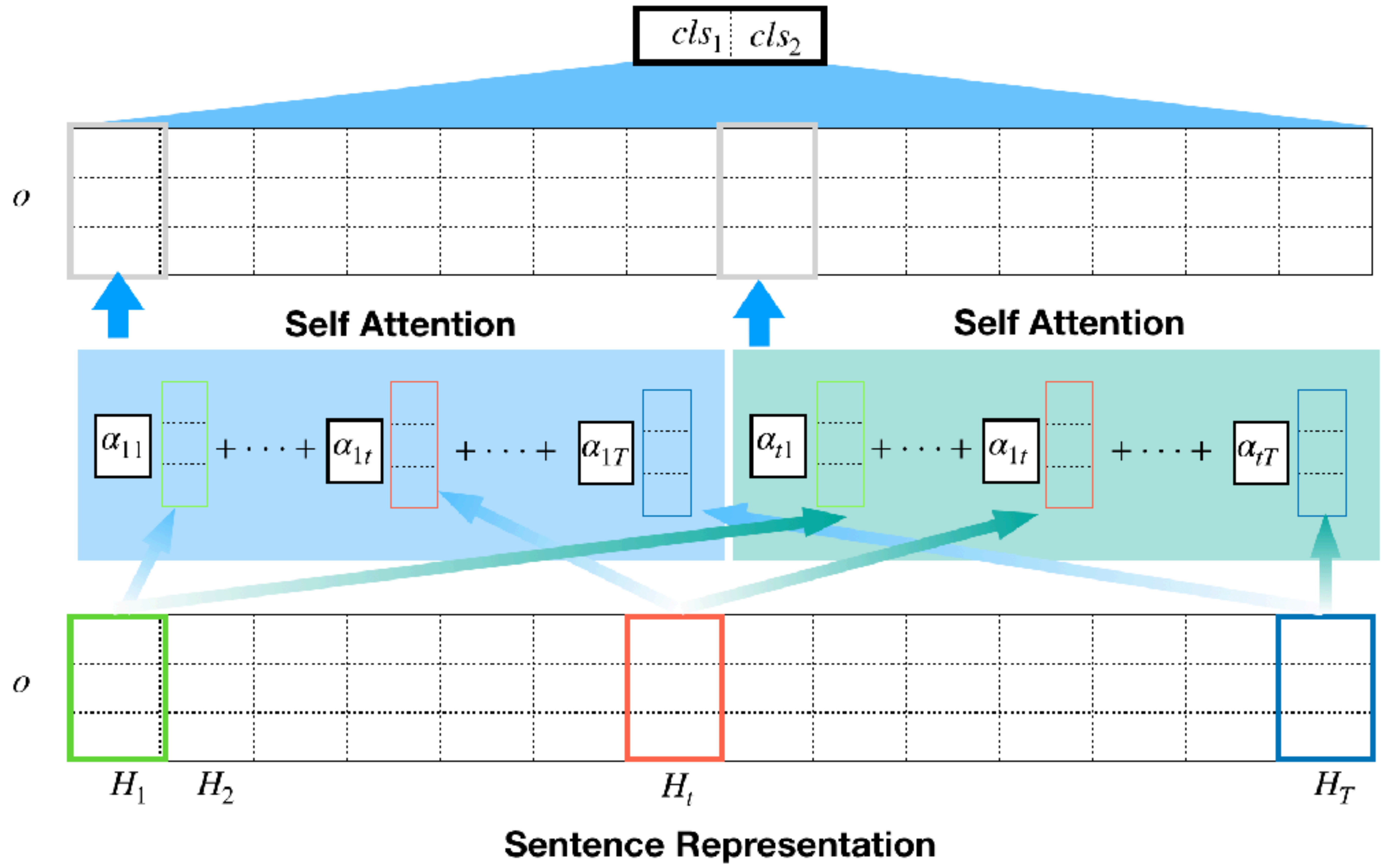


Idea of Self Attention

Why don’t we have more generator function than just indicator?

$$h_{t*} = \sum_{t=0}^{T} g(t, t^*) \cdot f(x_t, x_{t^*})$$

Self - Attention



Self - Attention

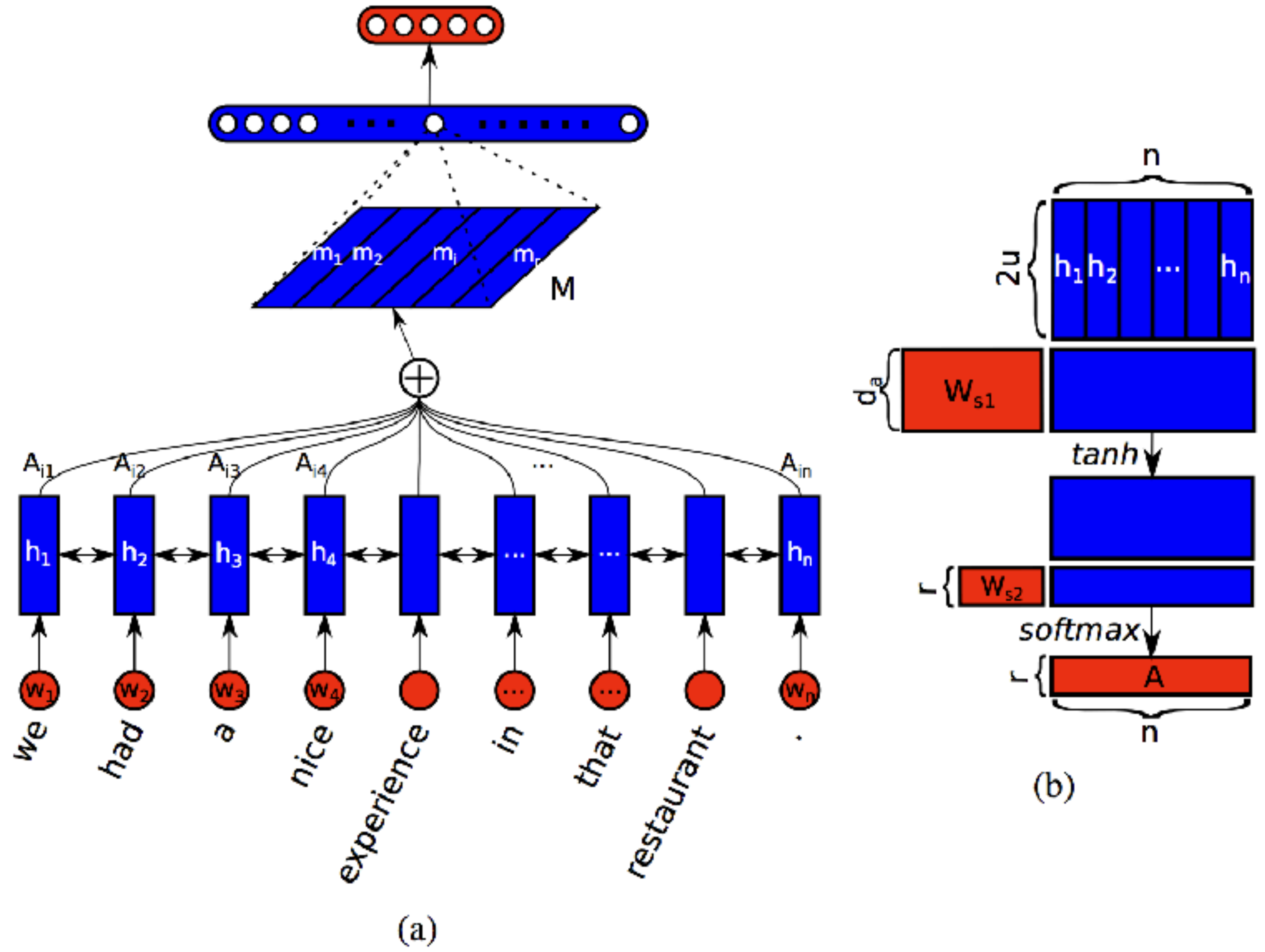


Figure 1: A sample model structure showing the sentence embedding model combined with a fully connected and softmax layer for sentiment analysis (a). The sentence embedding M is computed as multiple weighted sums of hidden states from a bidirectional LSTM ($\mathbf{h_1}, ..., \mathbf{h_n}$), where the summation weights ($A_{i1}, ..., A_{in}$) are computed in a way illustrated in (b). Blue colored shapes stand for hidden representations, and red colored shapes stand for weights, annotations, or input/output.

Transformer

Sorely based on attention-mechanism without underlying RNNs

RNNs are ***slow***, especially for LSTM and ***loose long-term dependency***

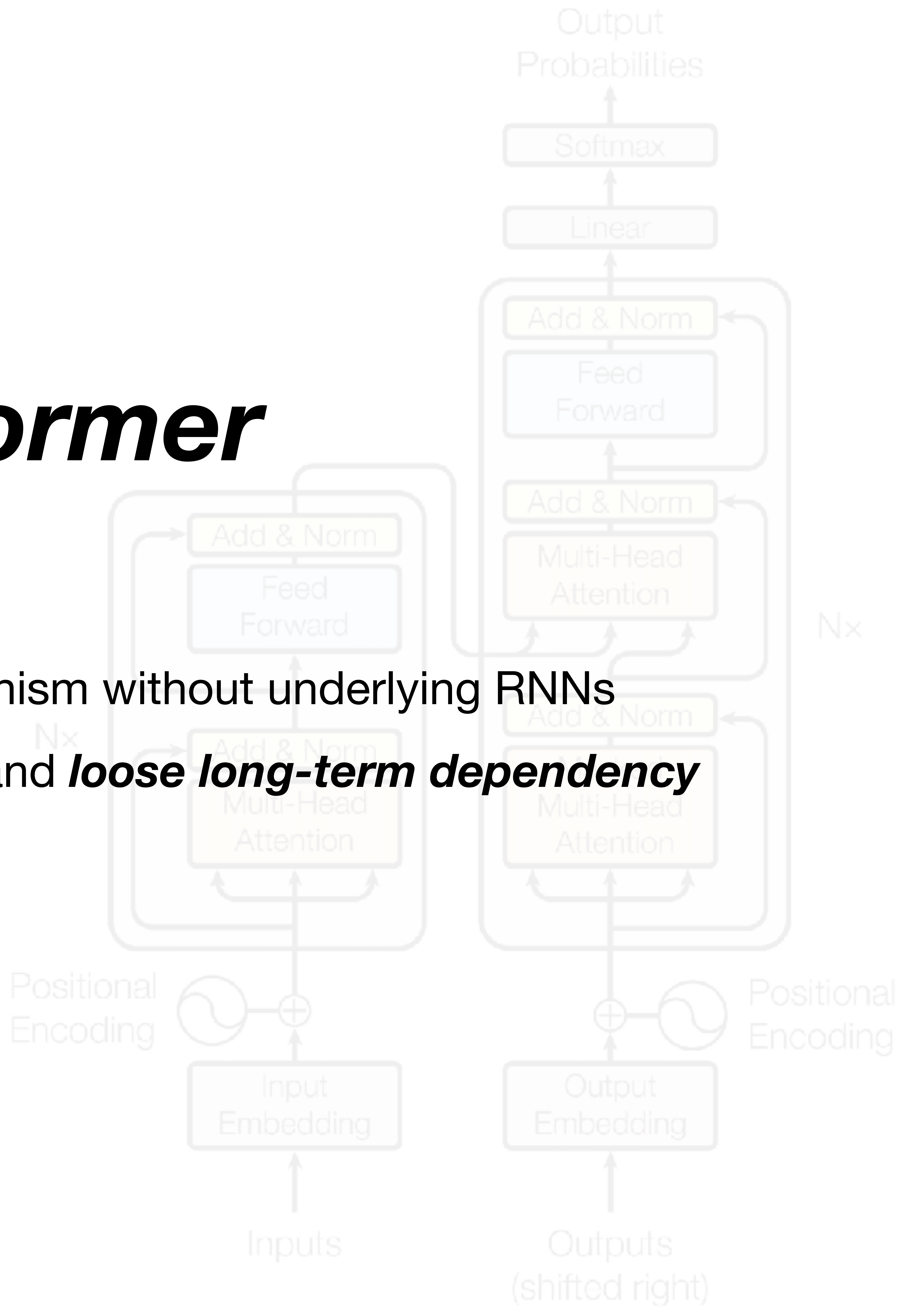


Figure 1: The Transformer - model architecture.

Transformer

Sorely based on attention-mechanism without underlying RNNs

RNNs are ***slow***, especially for LSTM and ***loose long-term dependency***

Much ***faster*** than RNN by putting ***sequence data all at once***

Can capture ***long-term dependency*** by employing ***attention-mechanism***

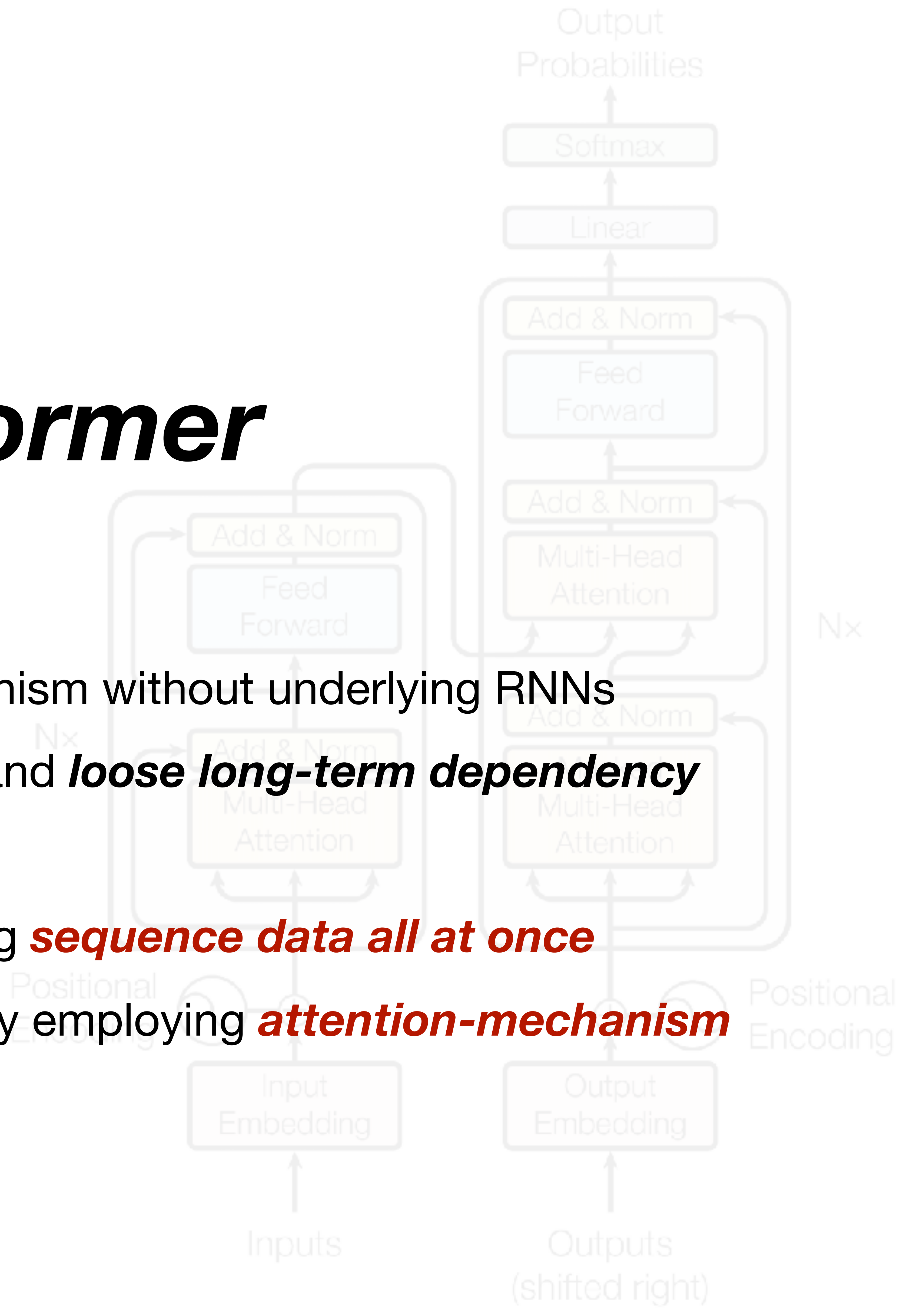


Figure 1: The Transformer - model architecture.

Transformer

Sorely based on attention-mechanism without underlying RNNs

RNNs are ***slow***, especially for LSTM and ***loose long-term dependency***

Much ***faster*** than RNN by putting ***sequence data all at once***

Can capture ***long-term dependency*** by employing ***attention-mechanism***

Known as the core engine for STOA neural machine translation engine

Building blocks of BERT

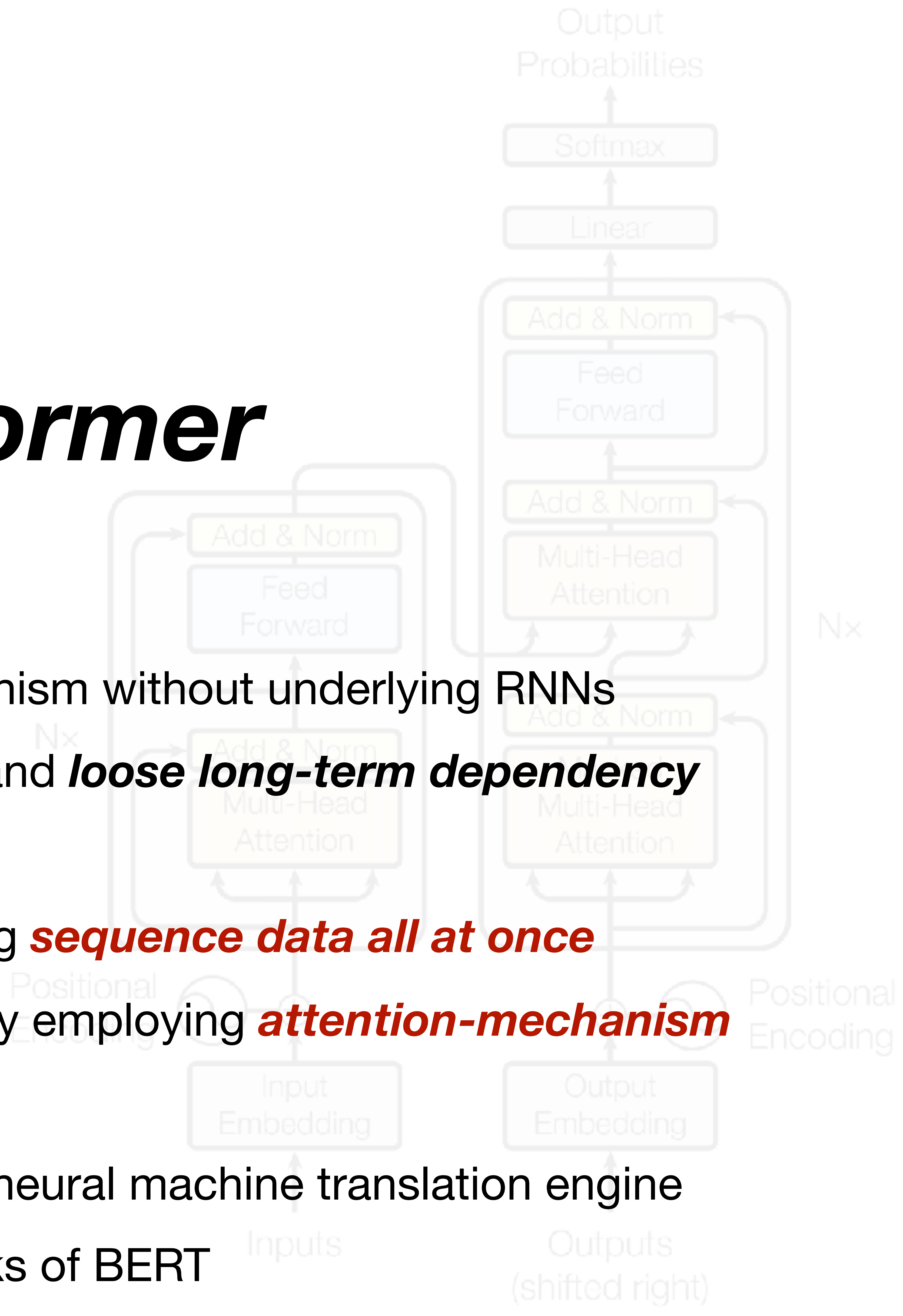


Figure 1: The Transformer - model architecture.

Transformer Architecture

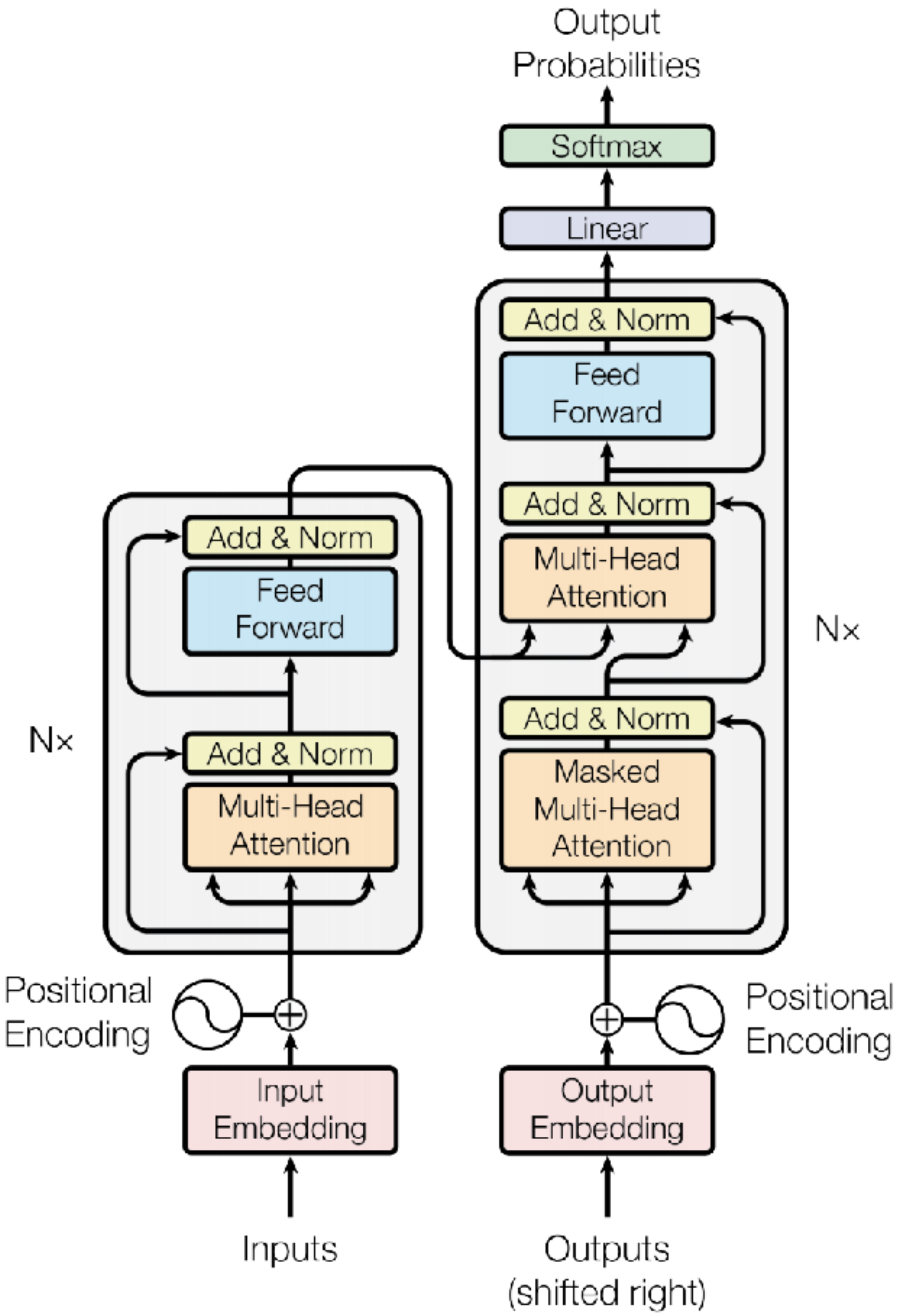


Figure 1: The Transformer - model architecture.