KiSTi 한국과학기술정보연구원 Korea Institute of Science and Technology Information

2022

Advanced Topic in Research Data-centric Deep Learning

Lec 11: Transformer for Traffic Flow Prediction

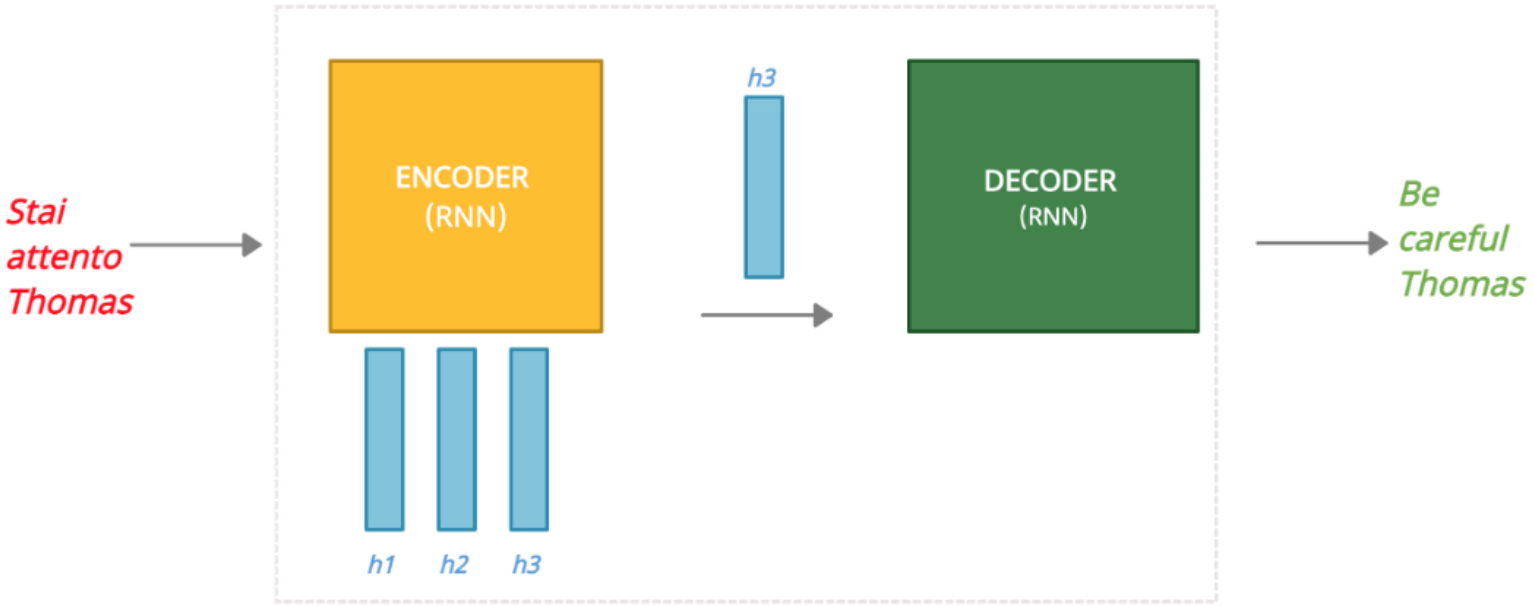
hsyi@kisti.re.kr

Hongsuk Yi (이홍석)

Reviewing the last class: Attention

Limitation of Seq2Seq model

- Main problem with seq2seq models
  - ✓compress all the information into one fixed-size vector results in information loss.
- This is the problem that attention solves!
  - ✓The last **hidden state (*h3*)** becomes the content that is sent to the decoder
  - ✓the encoder is “forced” to send only a single vector, regardless of the length of our input



Attention: a general formulation

- Attention weights, we apply the softmax of the “importance score”
- The “summary” is computed as a weighted sum

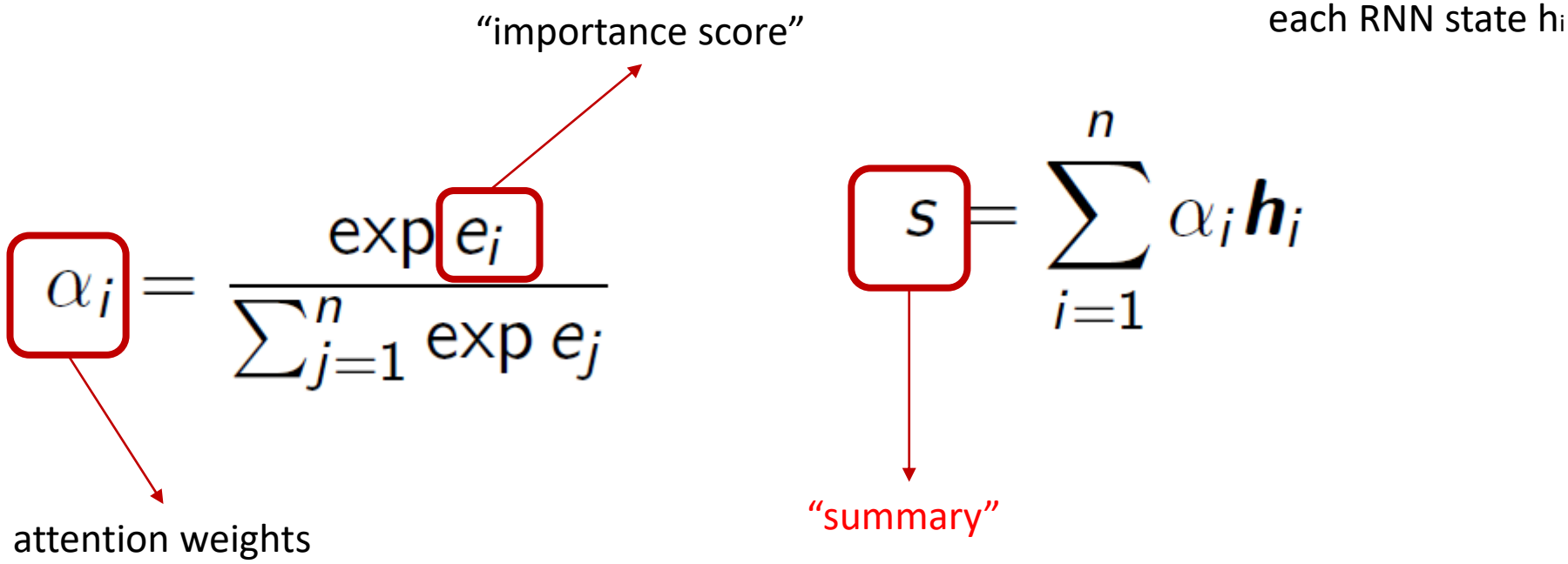


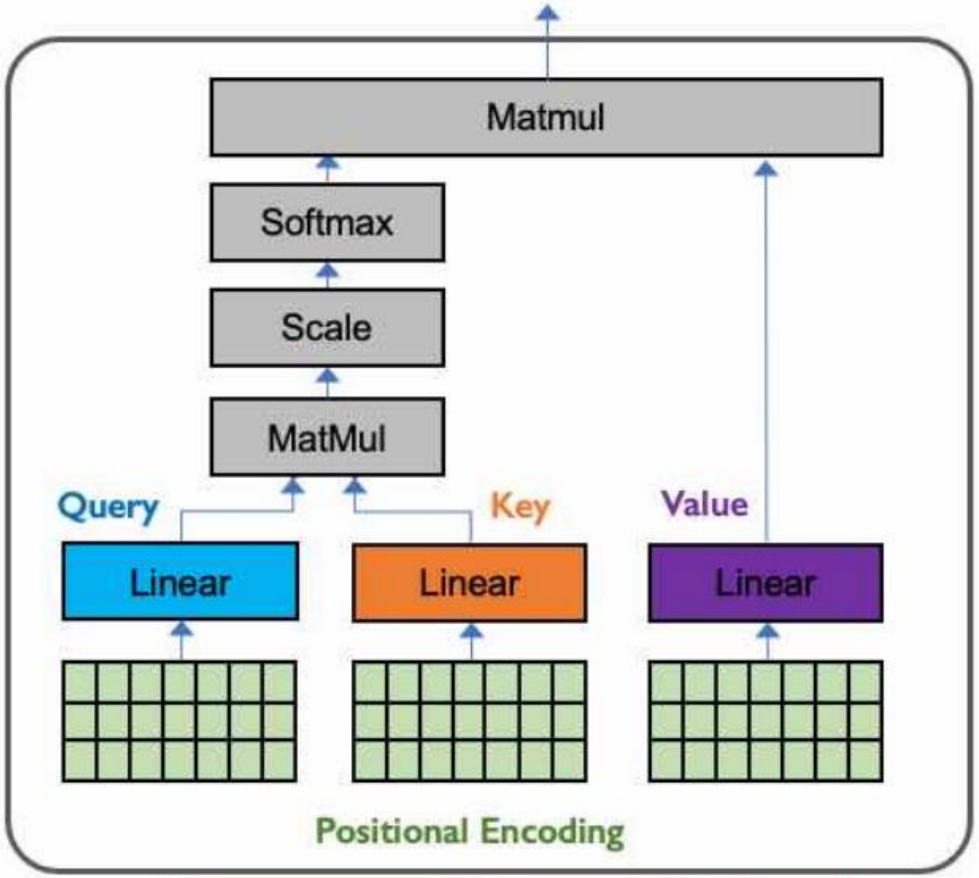
image borrowed from Richard Johansson (*Chalmers Technical University and University of Gothenburg)*

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

These operations form a self-attention head that can plug into a larger network. Each head attends to a different part of input.



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V$$

Attention with RNN

Image Captioning using spatial features

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, \ldots, y_T$

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

Encoder: $h_0 = f_W(\mathbf{z})$
where **z** is spatial CNN features
$f_W(.)$ is an MLP

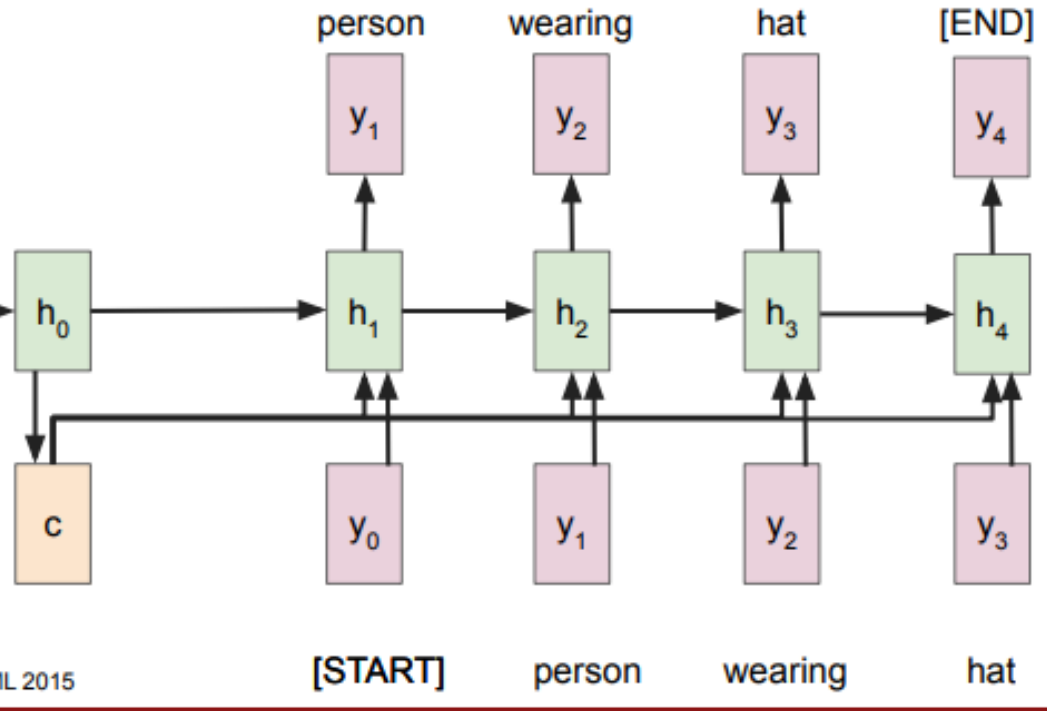
CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

MLP

Extract spatial features from a pretrained CNN

Features: H x W x D



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning using spatial features

Problem: Input is "bottlenecked" through c

- Model needs to encode everything it wants to say within c

This is a problem if we want to generate really long descriptions? 100s of words long

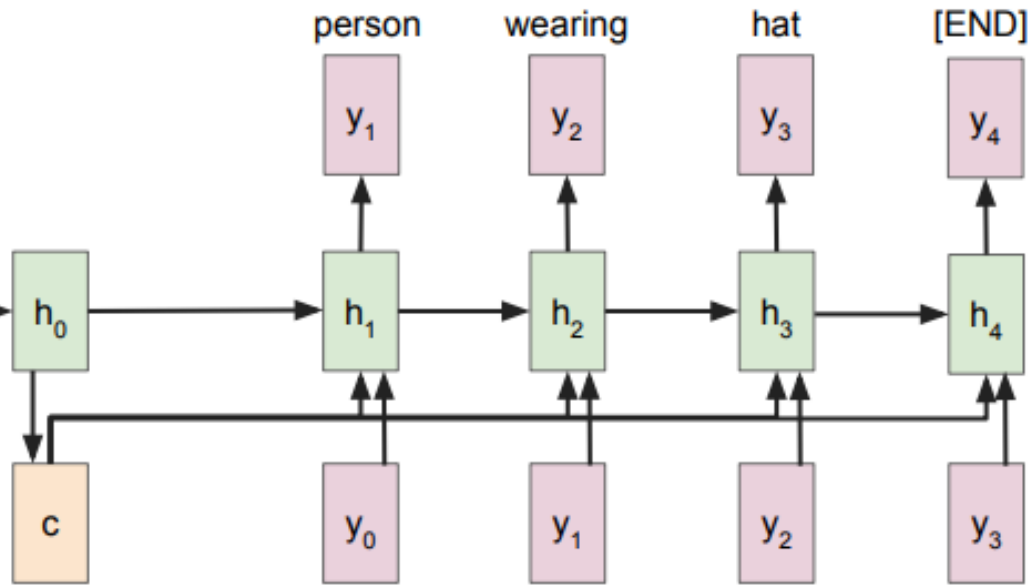
CNN

$z_{0,0}$ $z_{0,1}$ $z_{0,2}$
$z_{1,0}$ $z_{1,1}$ $z_{1,2}$
$z_{2,0}$ $z_{2,1}$ $z_{2,2}$

Extract spatial features from a pretrained CNN

Features: H x W x D

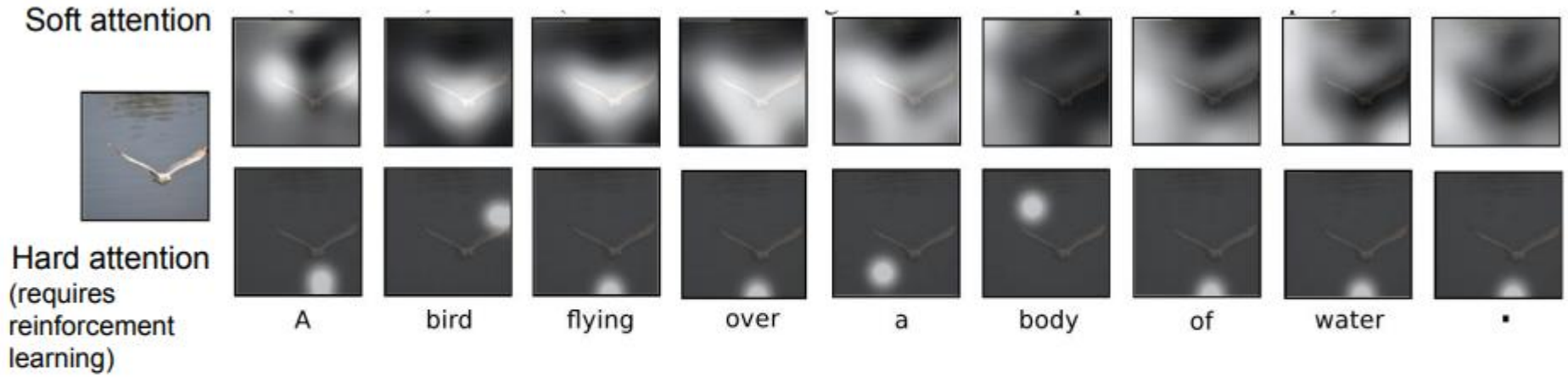
MLP



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

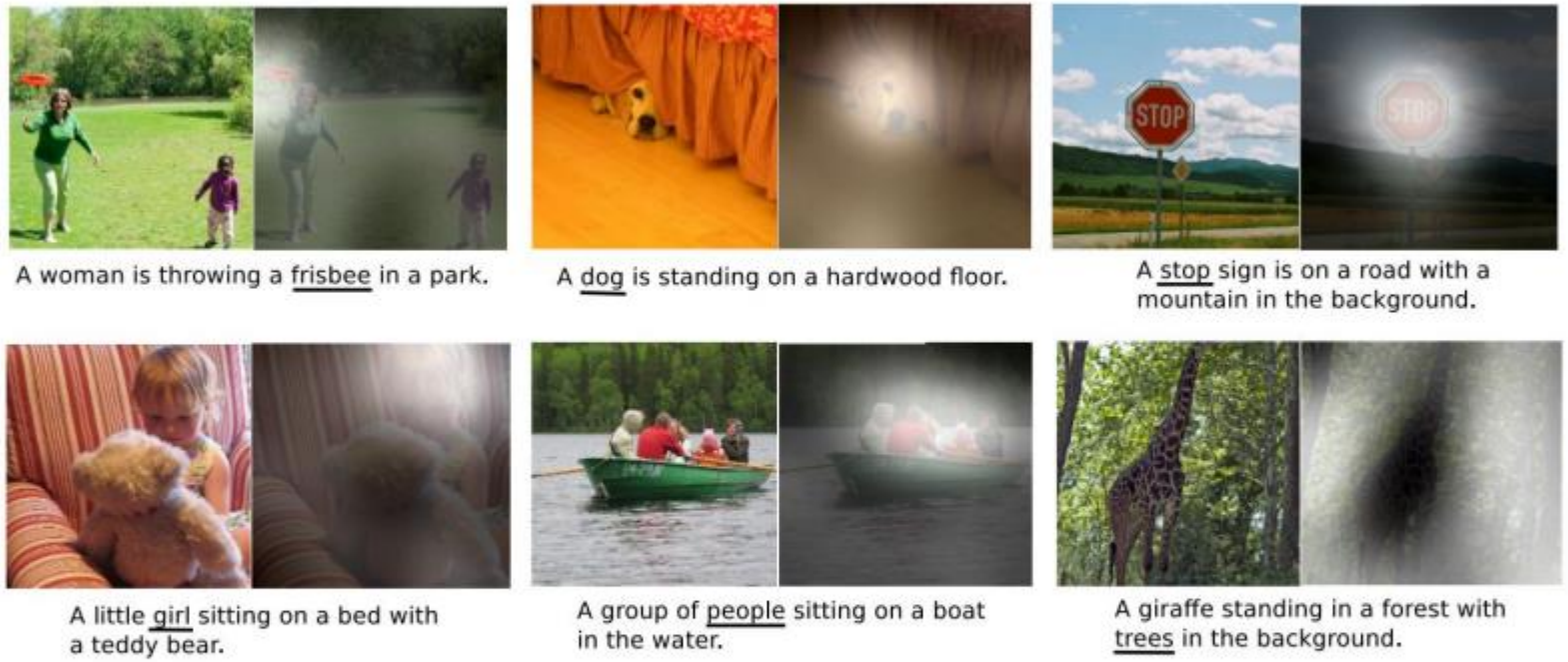
Image Captioning with RNNs & Attention

- Attention idea: New context vector at every time step.
- Each context vector will attend to different image regions



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



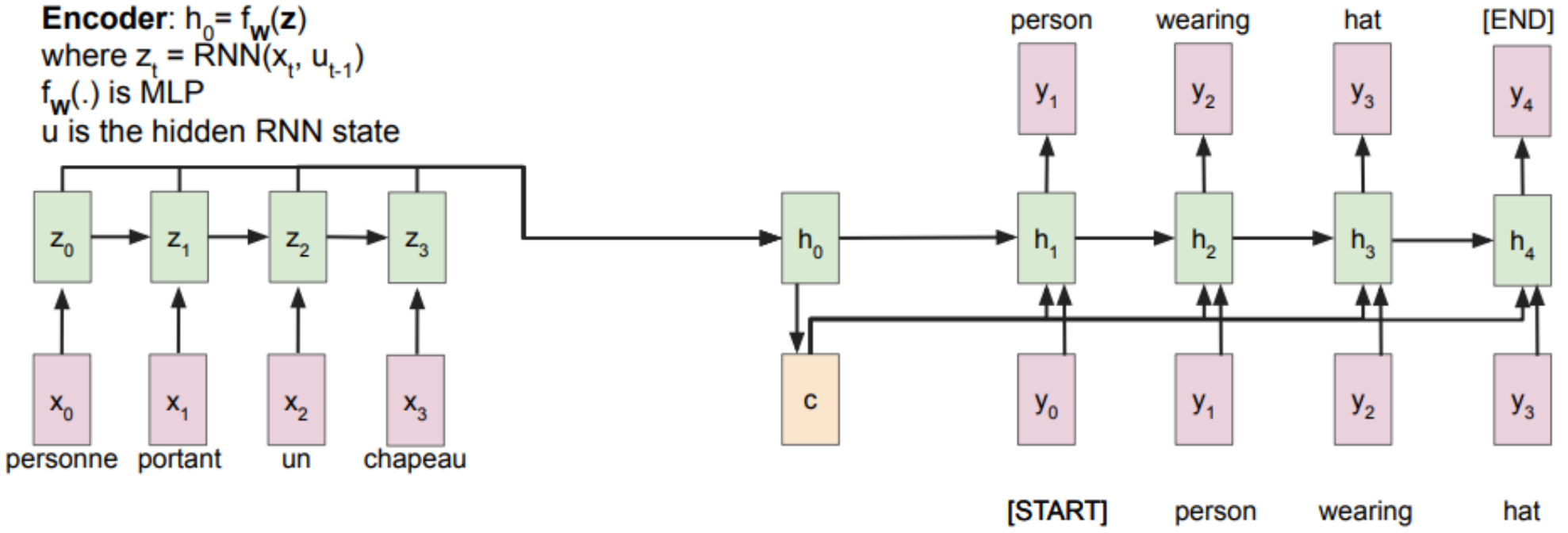
Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Similar tasks in NLP - Language translation example

Input: Sequence $\mathbf{x} = x_1, x_2, \ldots, x_T$
Output: Sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$

Encoder: $h_0 = f_W(\mathbf{z})$
where $z_t = \text{RNN}(x_t, u_{t-1})$
$f_W(.)$ is MLP
u is the hidden RNN state

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$



Review: NLP
Tockenizer and Seq2Seq

Tokenizing words

- Token : Language elements that we can't share anymore
  - ✓Word tokenization divides sentences based on spacing as follows.

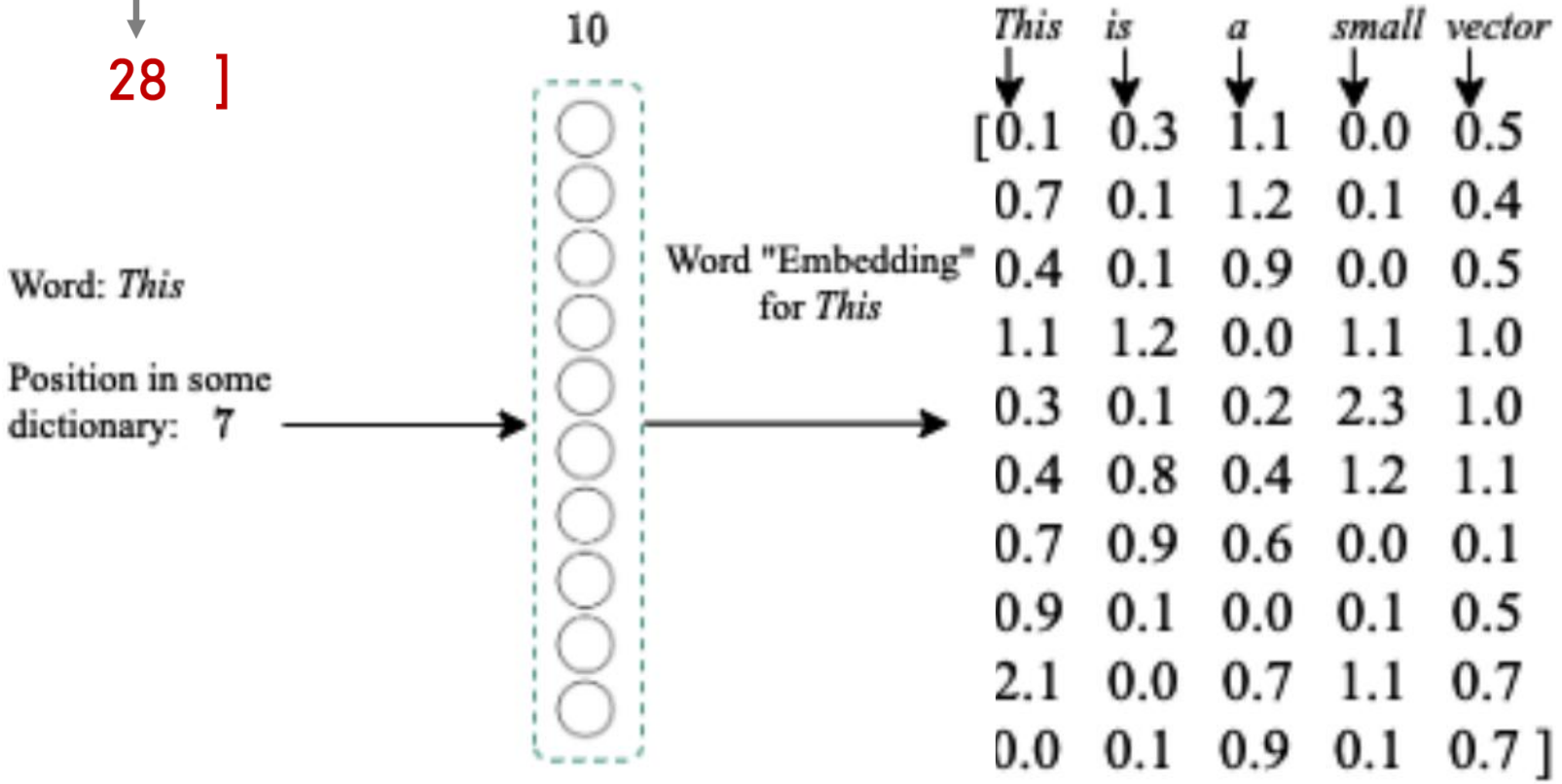
- Tokenizer : work to input text data into the neural network.
  - ✓The preprocessing process that converts it into an appropriate form through encoding

A simple example: Word embedding

- **vocab_size = 30, seq_length = 5**

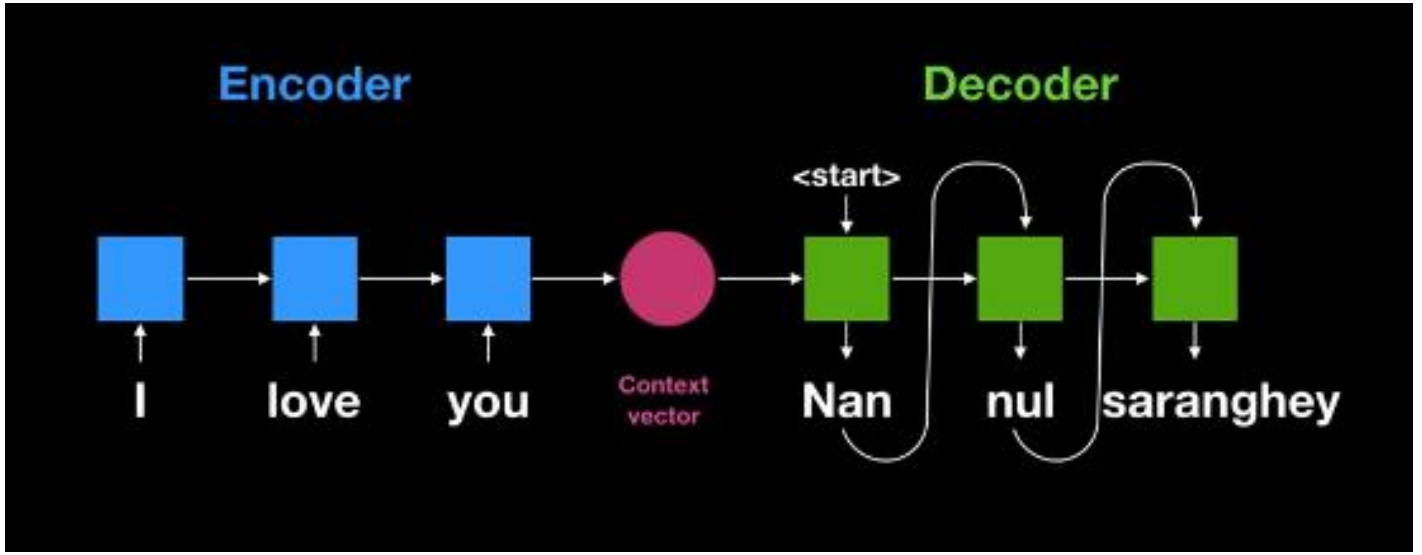
"This is a small vector"

[7 0 6 1 28]



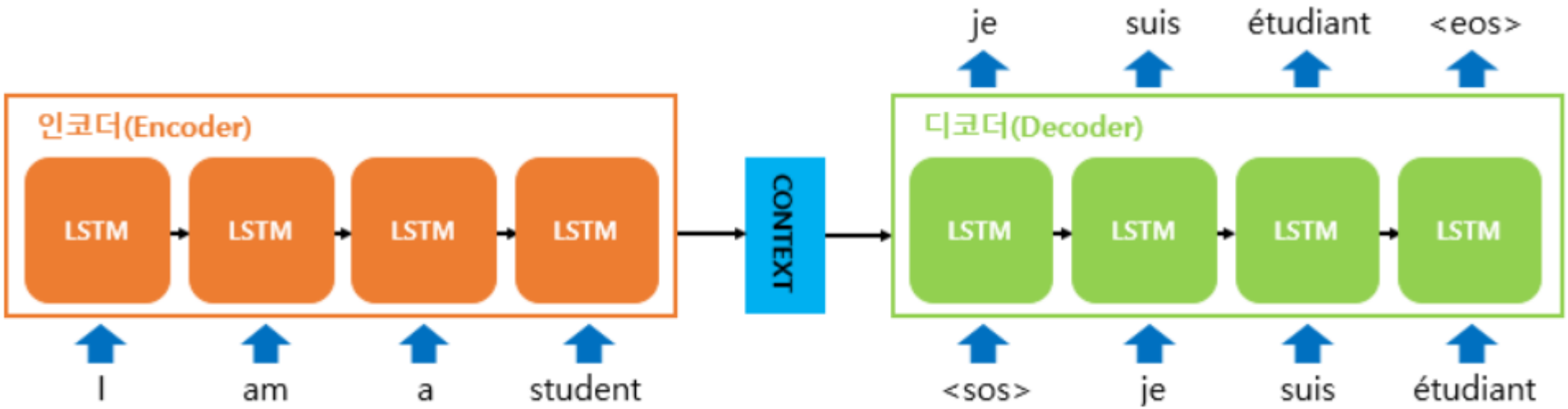
What is a Seq2Seq model?

- Sequence-to-sequence (Seq2Seq)
  - ✓convert sequences from one domain to sequences in another domain
  - ✓This can be used for machine translation
- A Seq2Seq model usually consists of Encoder, decoder, context
  - context vector is sent to the decoder which formulates the output sequence



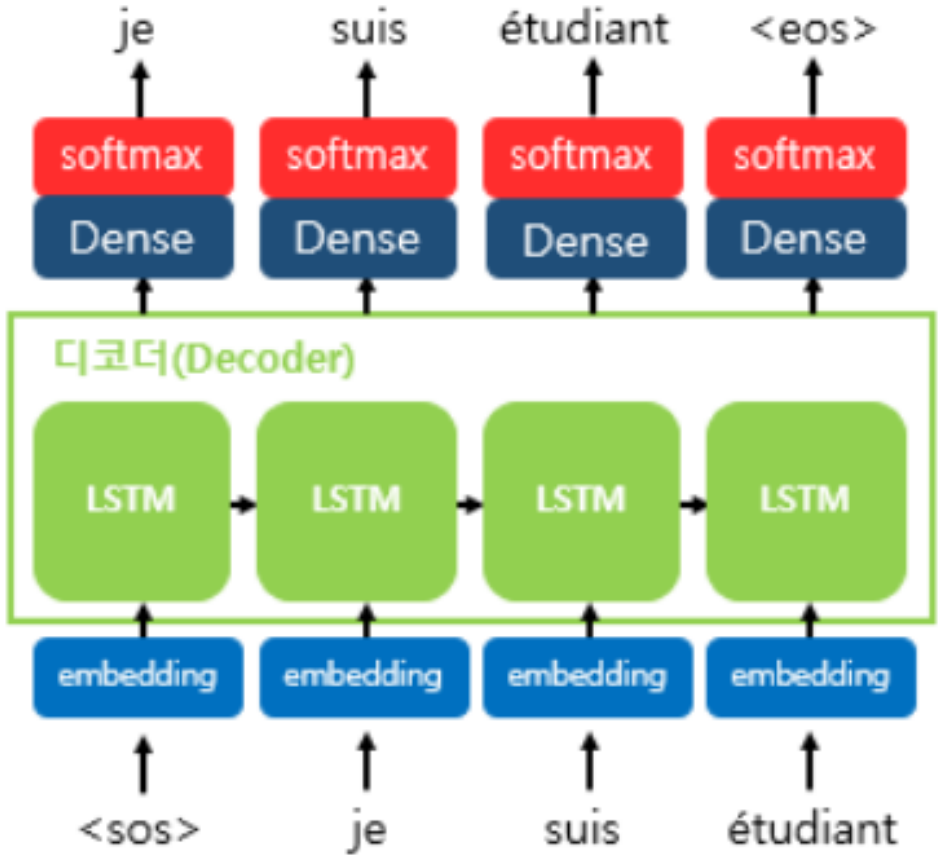
What is a Seq2Seq model?

- Encoder and decoder are RNN architectures
- context vector
  - ✓The context vector is the first hidden state of the decoder RNN cell

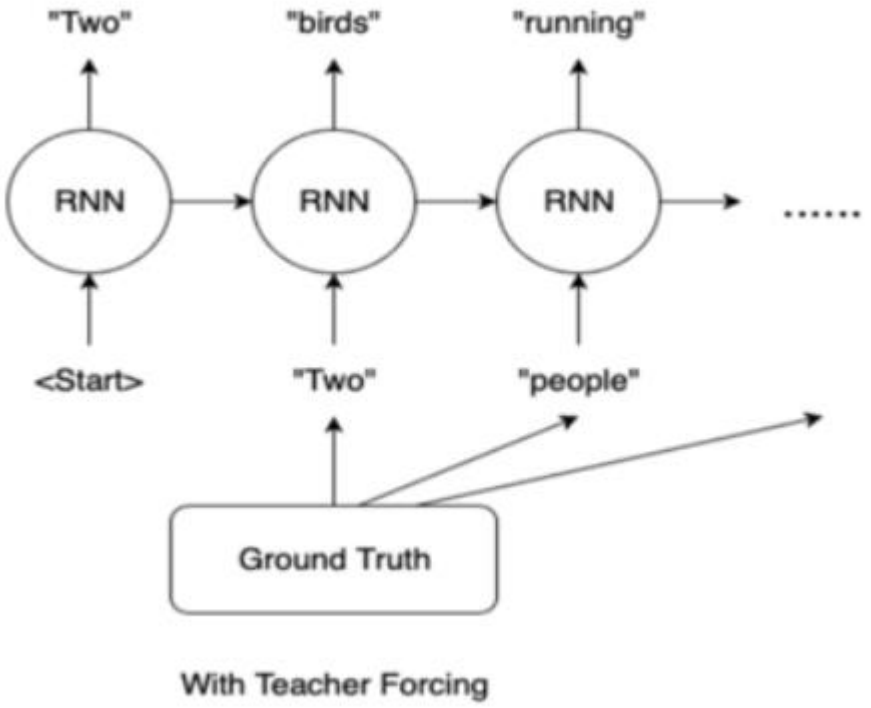


Seq2Seq: Decoder part

- Softmax for next prediction word

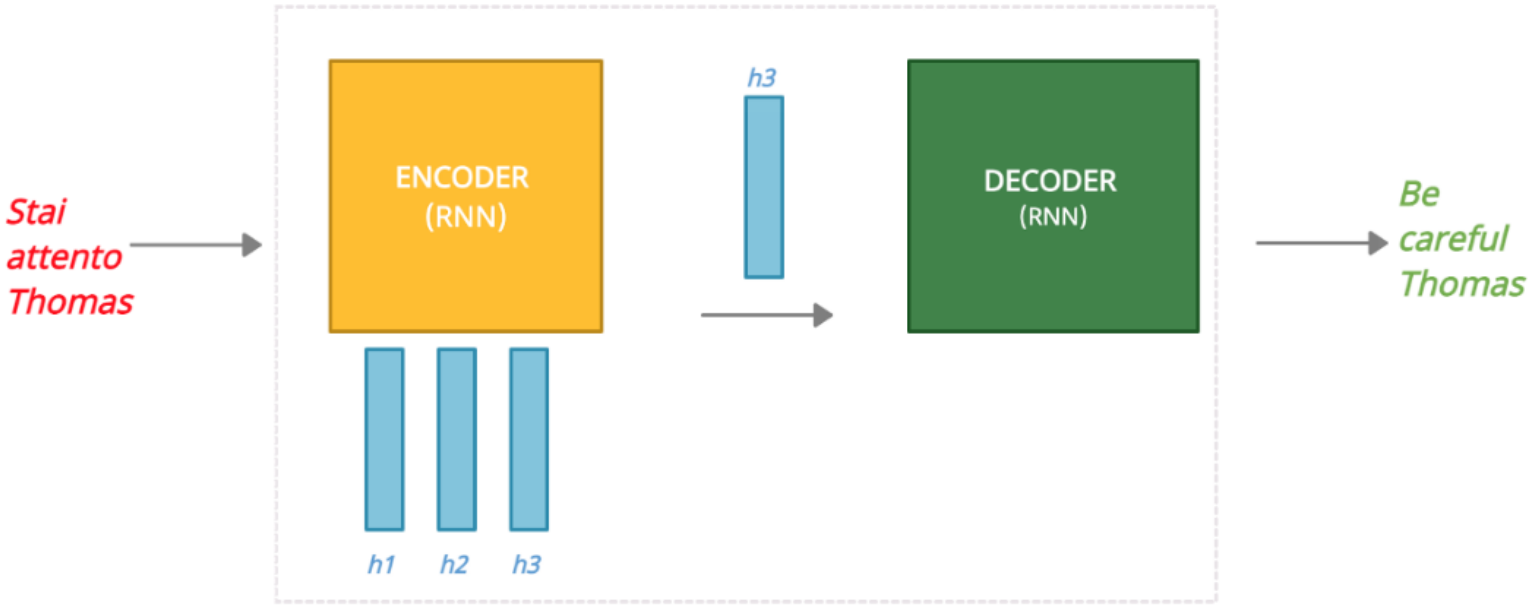


Teacher Forecing Learning



Limitation of Seq2Seq model

- Main problem with seq2seq models
  - ✓compress all the information into one fixed-size vector results in information loss.
- This is the problem that attention solves!
  - ✓The last **hidden state (*h3*)** becomes the content that is sent to the decoder
  - ✓the encoder is “forced” to send only a single vector, regardless of the length of our input



A general formulation

- ✓ for the attention weights, we apply the softmax
- ✓ the "summary" is computed as a weighted sum

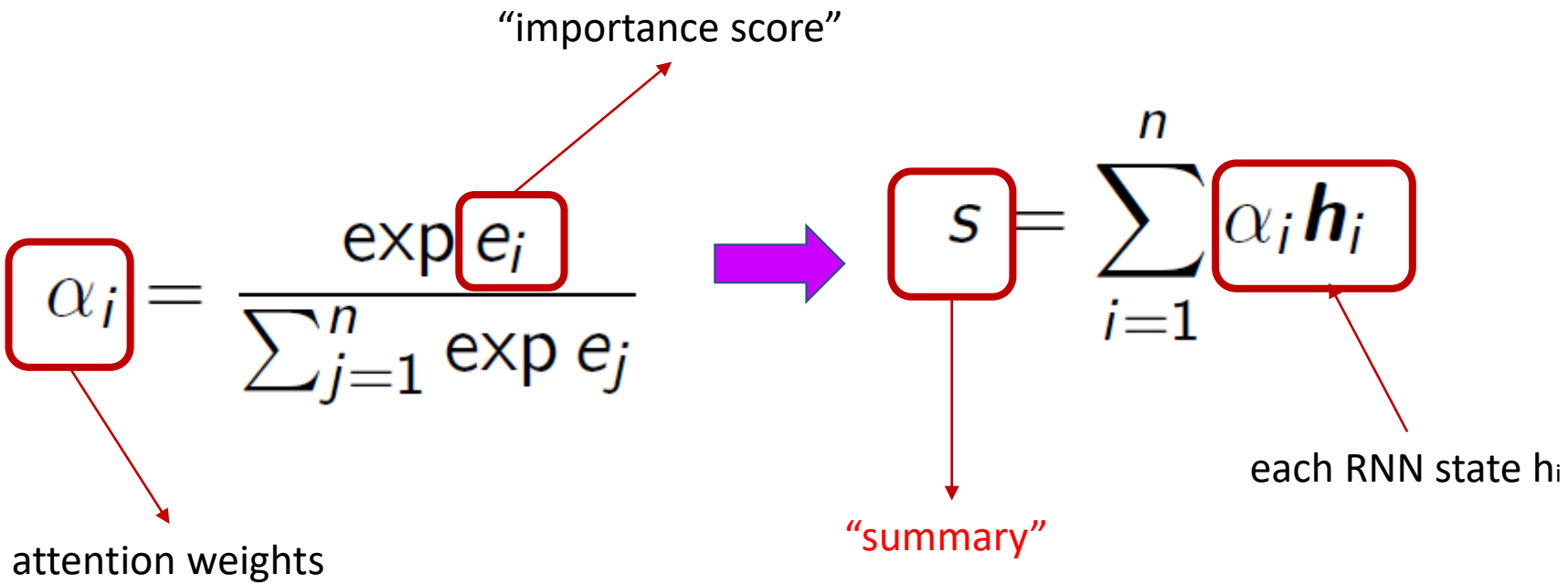


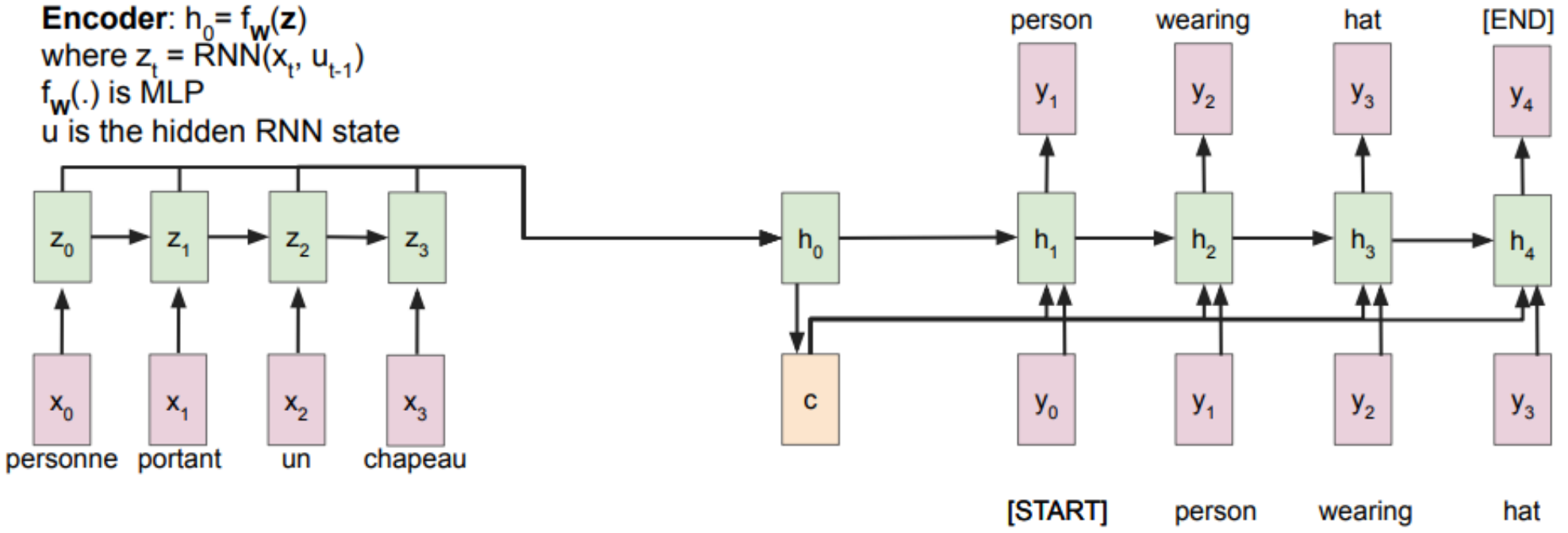
image borrowed from Richard Johansson (*Chalmers Technical University and University of Gothenburg*)

Similar tasks in NLP - Language translation example

Input: Sequence $\mathbf{x} = x_1, x_2, \ldots, x_T$
Output: Sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$

Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where $z_t = \text{RNN}(x_t, u_{t-1})$
$f_{\mathbf{W}}(.)$ is MLP
u is the hidden RNN state

Decoder: $y_t = g_{\mathbf{V}}(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

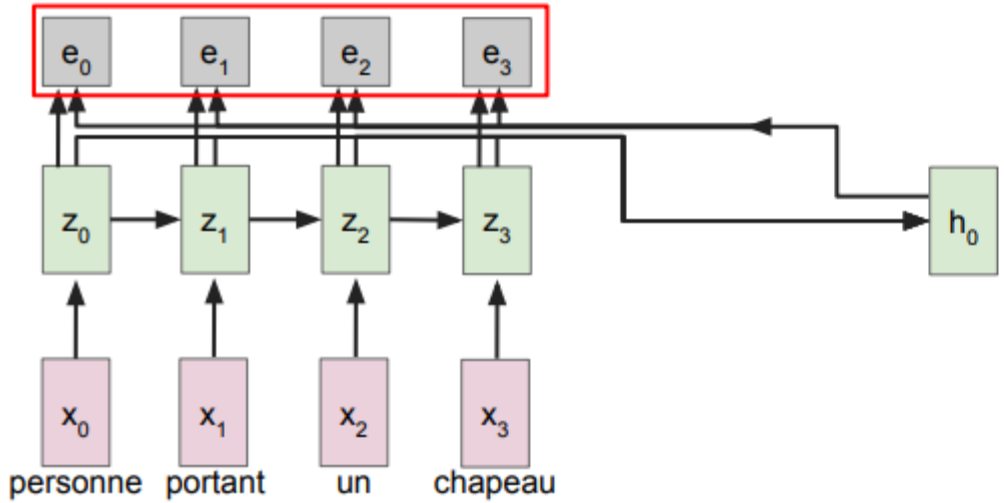


Attention in NLP - Language translation example

Compute alignments scores (scalars):

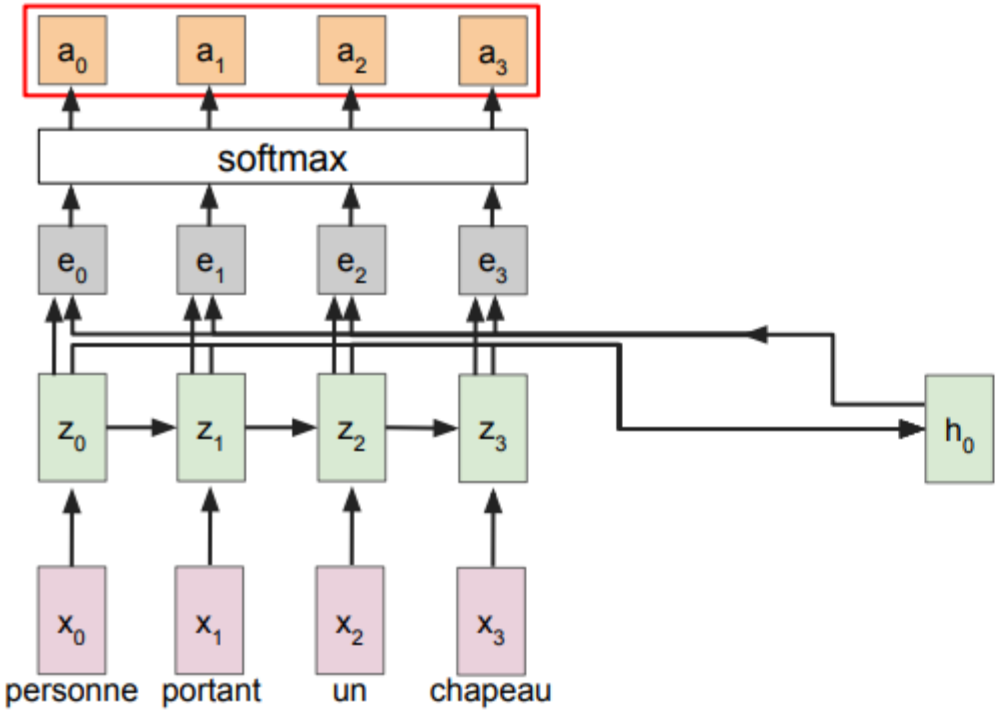
$$e_{t,i} = f_{att}(h_{t-1}, z_i)$$

$f_{att}(.)$ is an MLP



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention in NLP - Language translation example



Compute alignments scores (scalars):

$$e_{t,i} = f_{att}(h_{t-1}, z_i)$$

$f_{att}(.)$ is an MLP

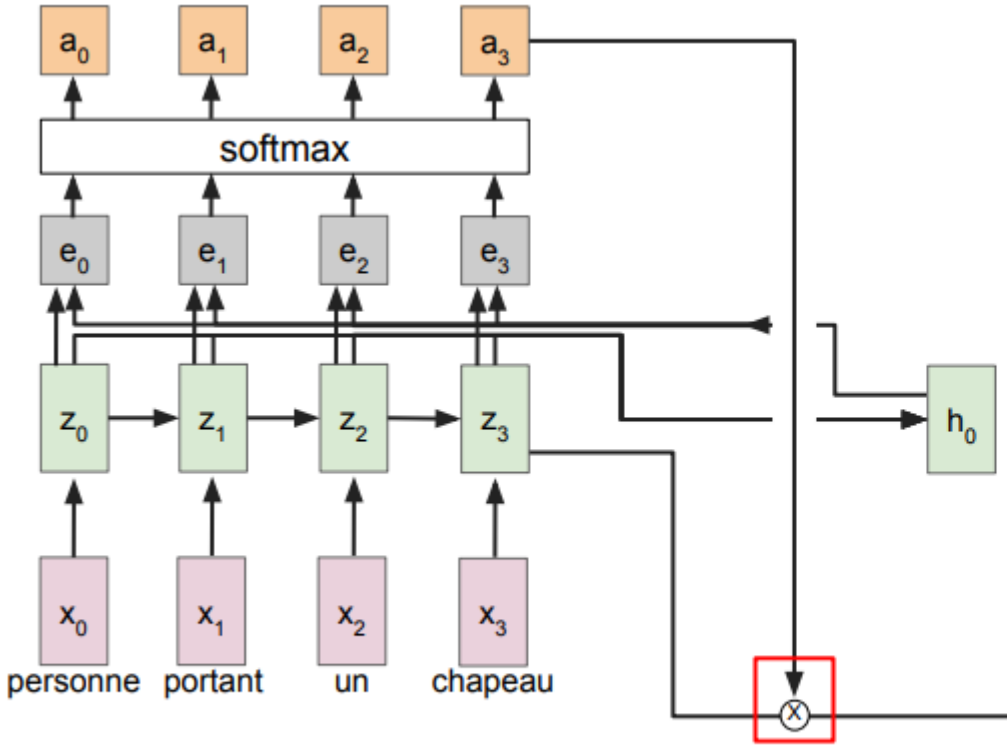
Normalize to get attention weights:

$$a_{t,:} = softmax(e_{t,:})$$

$0 < a_{t,i,j} < 1$,
attention values sum to 1

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Attention in NLP - Language translation example



Bahdanau et al, "Neural machine translaAon by jointly learning to align and translate", ICLR 2015

Compute alignments scores (scalars):

$$e_{t,i} = f_{att}(h_{t-1}, z_i)$$

$f_{att}(.)$ is an MLP

Normalize to get attention weights:

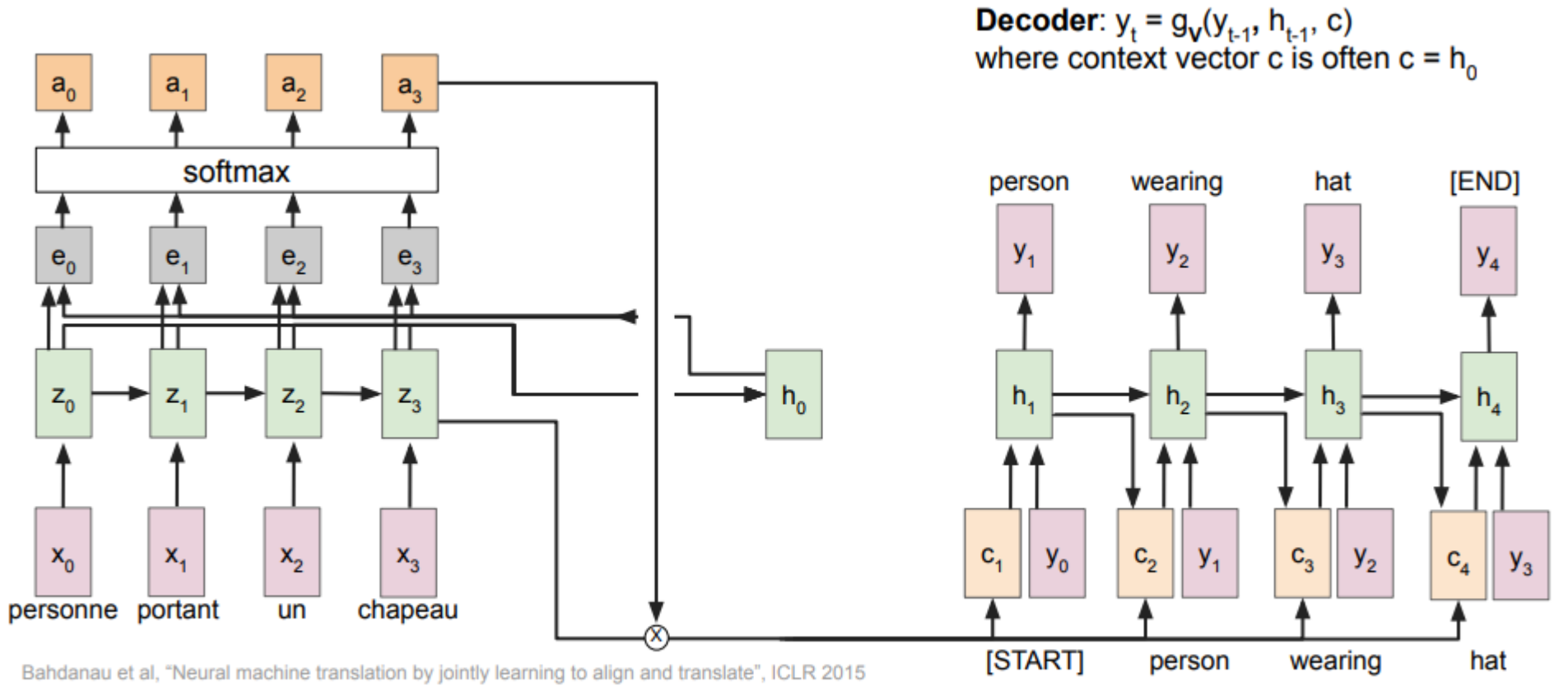
$$a_{t,:} = softmax(e_{t,:})$$

$0 < a_{t,i,j} < 1$,
attention values sum to 1

Compute context vector:

$$c_t = \sum_i a_{t,i} z_{t,i}$$

Attention in NLP - Language translation example



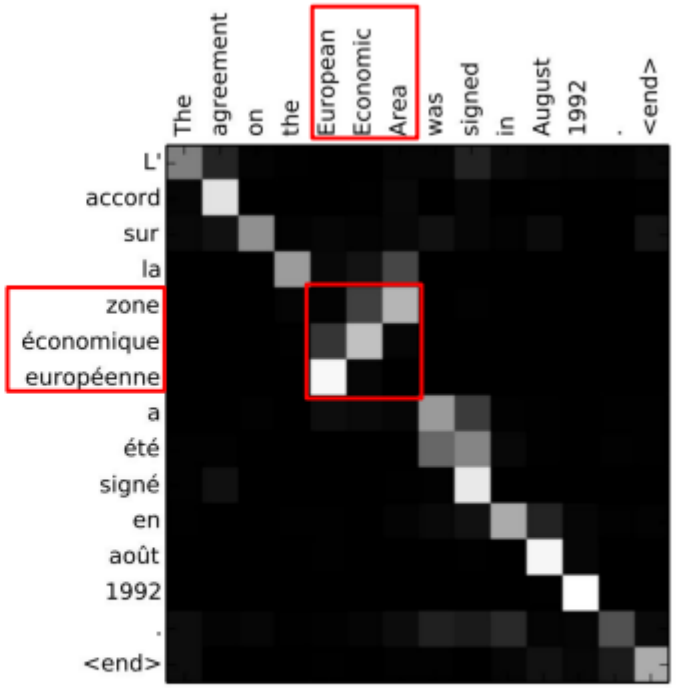
Similar visualization of attention weights

English to French translation example:

Input: "The agreement on the European Economic Area was signed in August 1992."

Output: "L'accord sur la zone économique européenne a été signé en août 1992."

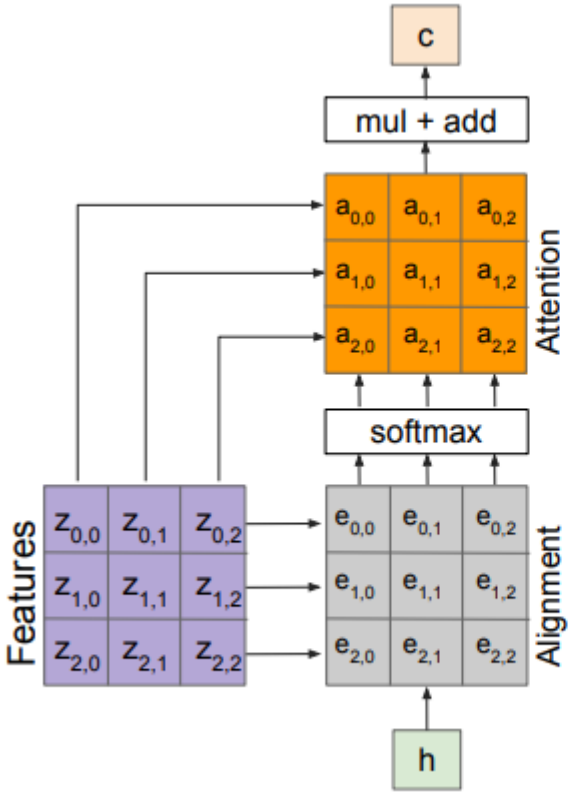
Without any attention supervision, model learns different word orderings for different languages



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Self-attention

Attention we just saw in image captioning

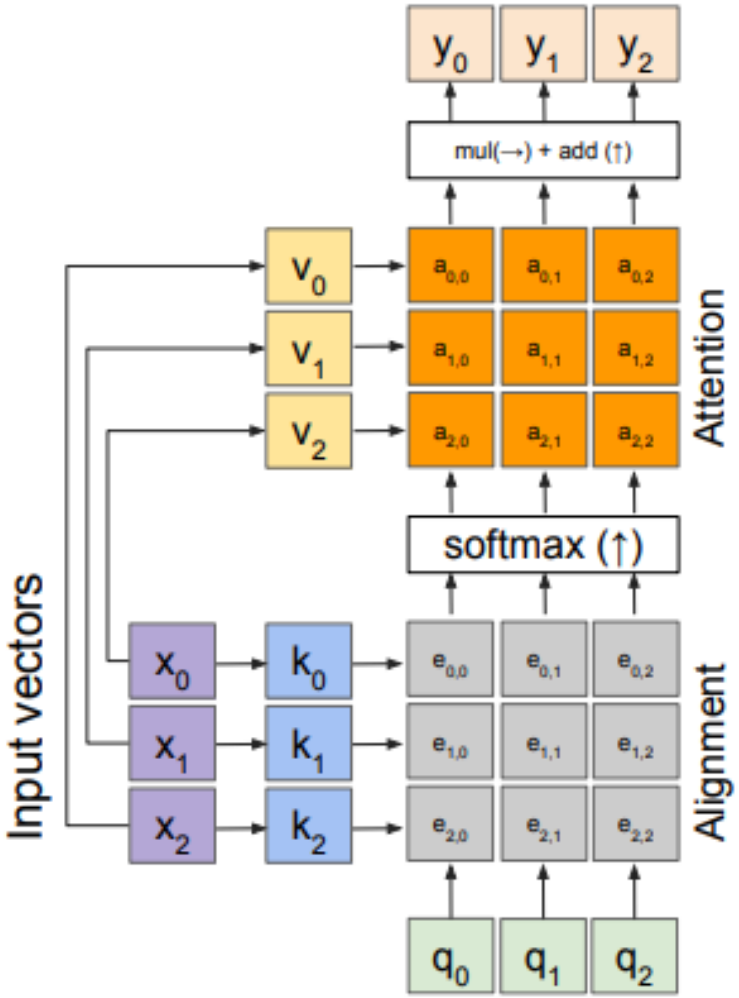


Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_{i,j} a_{i,j} z_{i,j}$

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

General attention layer



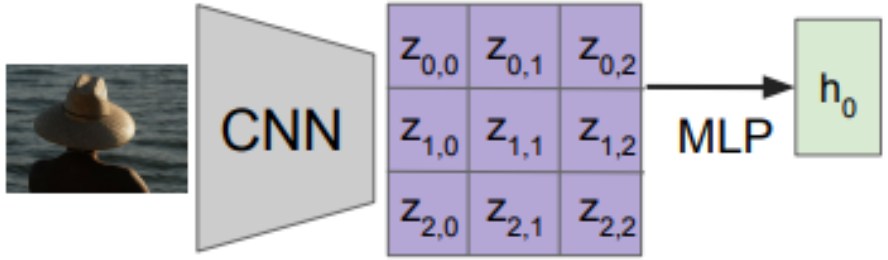
Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $a = \text{softmax}(e)$
Output: $y_j = \sum_i a_{i,j} v_i$

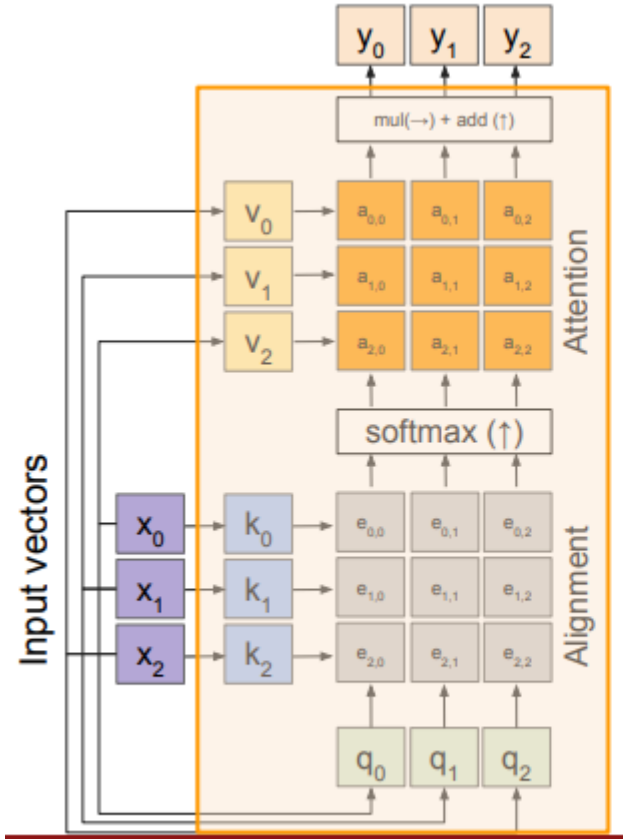
Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

Recall that the query vector was a function of the input vectors

Encoder: $h_0 = f_W(z)$
where **z** is spatial CNN features
$f_W(.)$ is an MLP



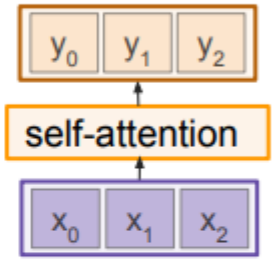
Self attention layer - attends over sets of inputs



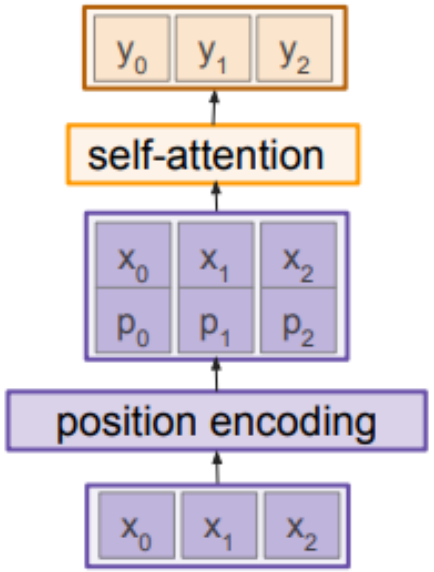
Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Query vectors: $q = xW_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)



Positional encoding



Concatenate special positional encoding p_j to each input vector x_j

We use a function *pos*: $N \rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, $p_j = pos(j)$

Options for *pos*(.)

1. Learn a lookup table:
   - Learn parameters to use for *pos*(t) for $t \in [0, T)$
   - Lookup table contains T x d parameters.
2. Design a fixed function with the desiderata
   -

$$p(t) = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_d$$

Intuition:

0:	0	0	0	0	8:	1	0	0	0
1:	0	0	0	1	9:	1	0	0	1
2:	0	0	1	0	10:	1	0	1	0
3:	0	0	1	1	11:	1	0	1	1
4:	0	1	0	0	12:	1	1	0	0
5:	0	1	0	1	13:	1	1	0	1
6:	0	1	1	0	14:	1	1	1	0
7:	0	1	1	1	15:	1	1	1	1

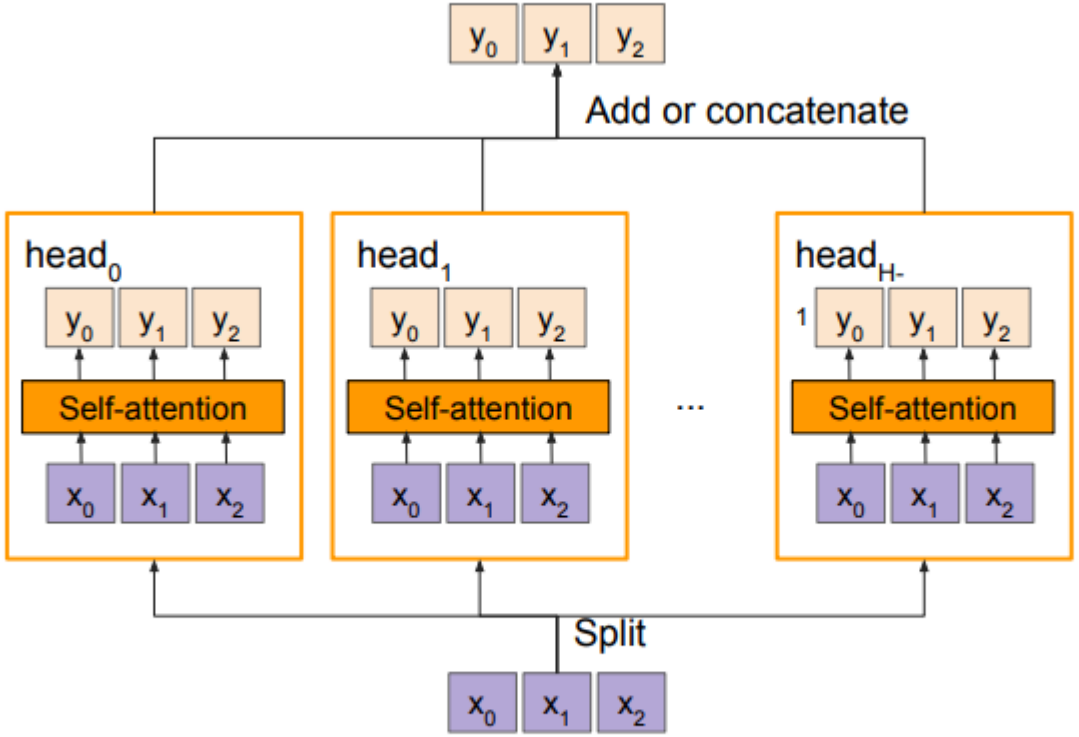
where $\omega_k = \frac{1}{10000^{2k/d}}$

image source

Vaswani et al, "Attention is all you need", NeurIPS 2017

Multi-head self attention layer

- Multiple self-attention heads in parallel



Comparing RNNs to Transformers

RNNs

(+) LSTMs work reasonably well for long sequences.

(-) Expects an ordered sequences of inputs

(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

Transformers:

(+) Good at long sequences. Each attention calculation looks at all inputs.

(+) Can operate over unordered sets or ordered sequences with positional encodings.

(+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.

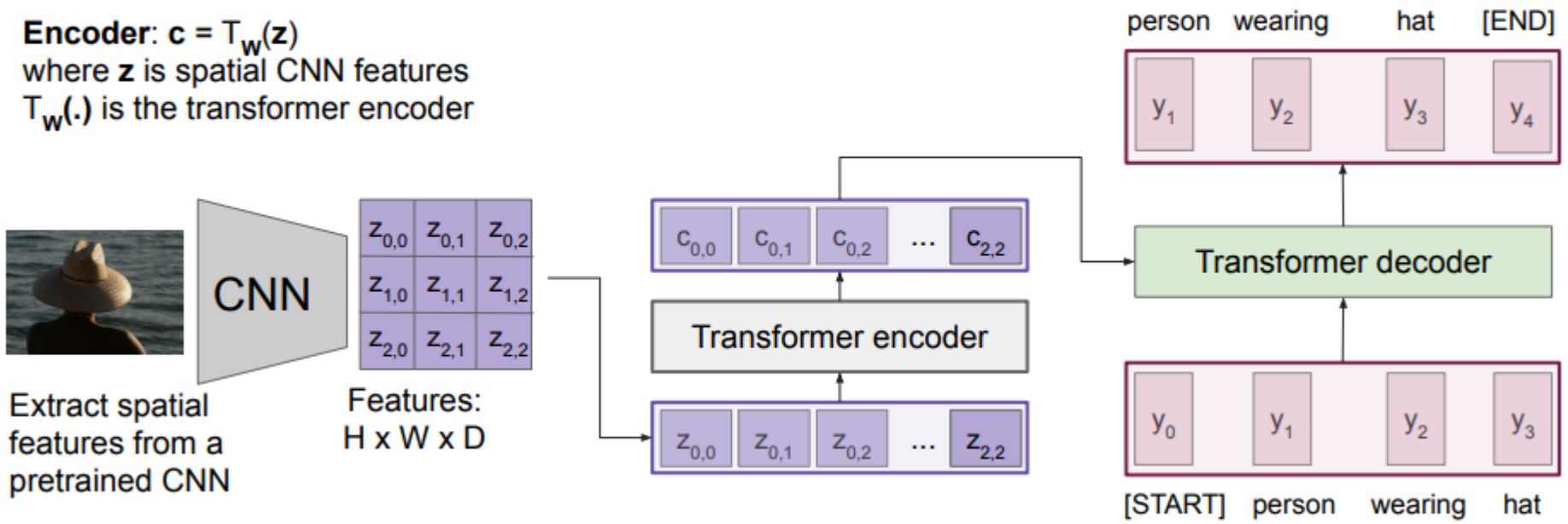
(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

Image Captioning using transformers

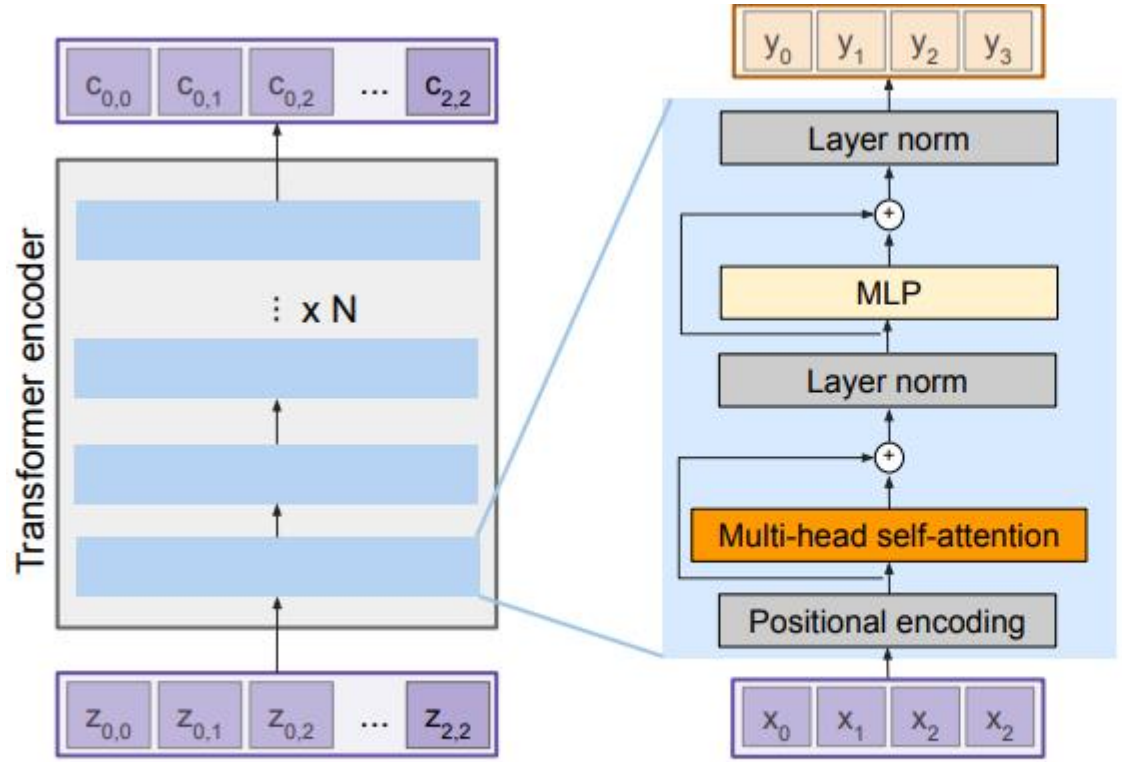
Input: Image **I**
Output: Sequence **y** = $y_1, y_2, \ldots, y_T$

Encoder: $\mathbf{c} = T_W(\mathbf{z})$
where **z** is spatial CNN features
$T_W(.)$ is the transformer encoder

Decoder: $y_t = T_D(y_{0:t-1}, \mathbf{c})$
where $T_D(.)$ is the transformer decoder



The Transformer encoder block



Transformer Encoder Block:

Inputs: Set of vectors **x**
Outputs: Set of vectors **y**

Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, "Attention is all you need", NeurIPS 2017

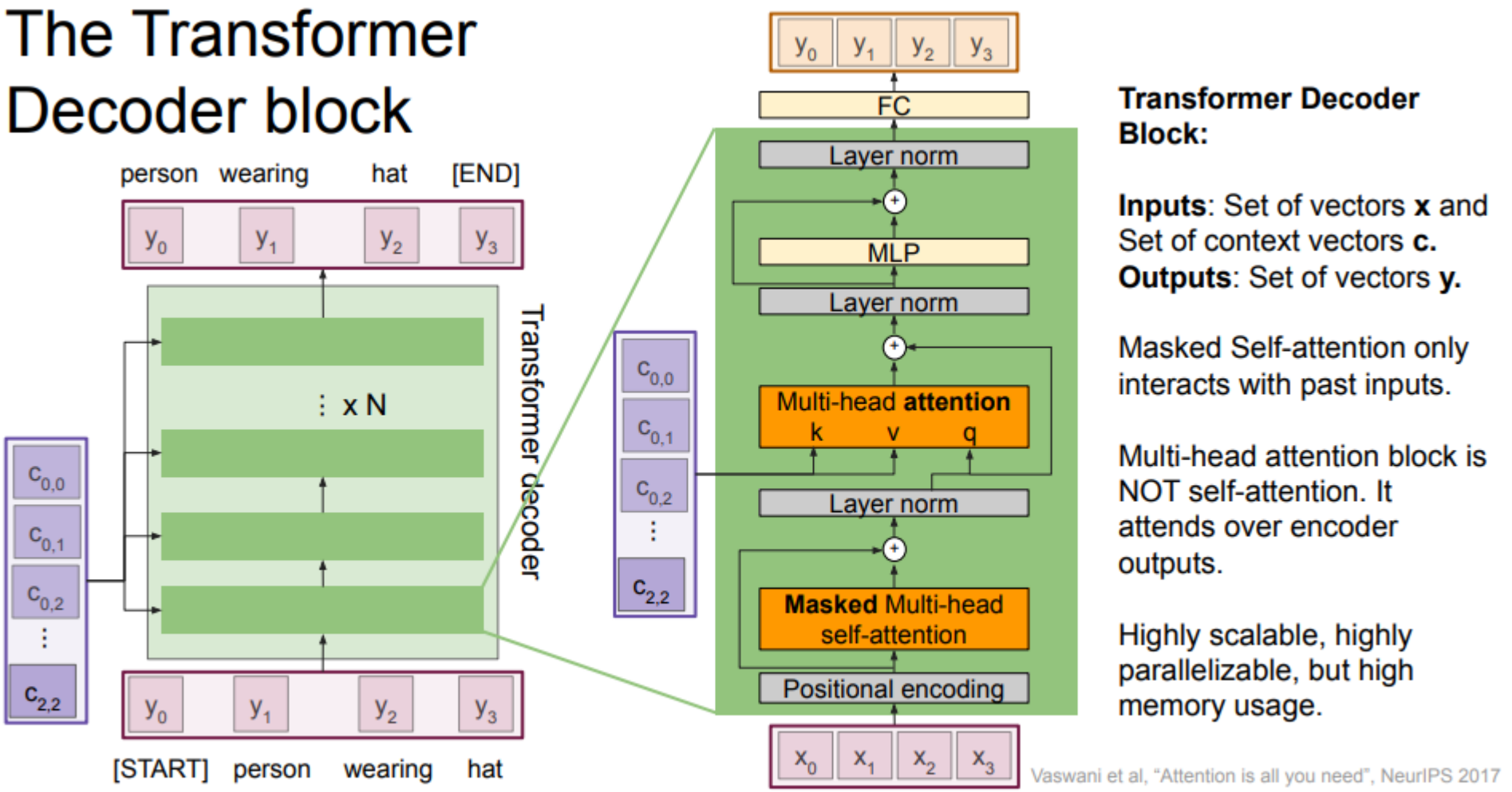
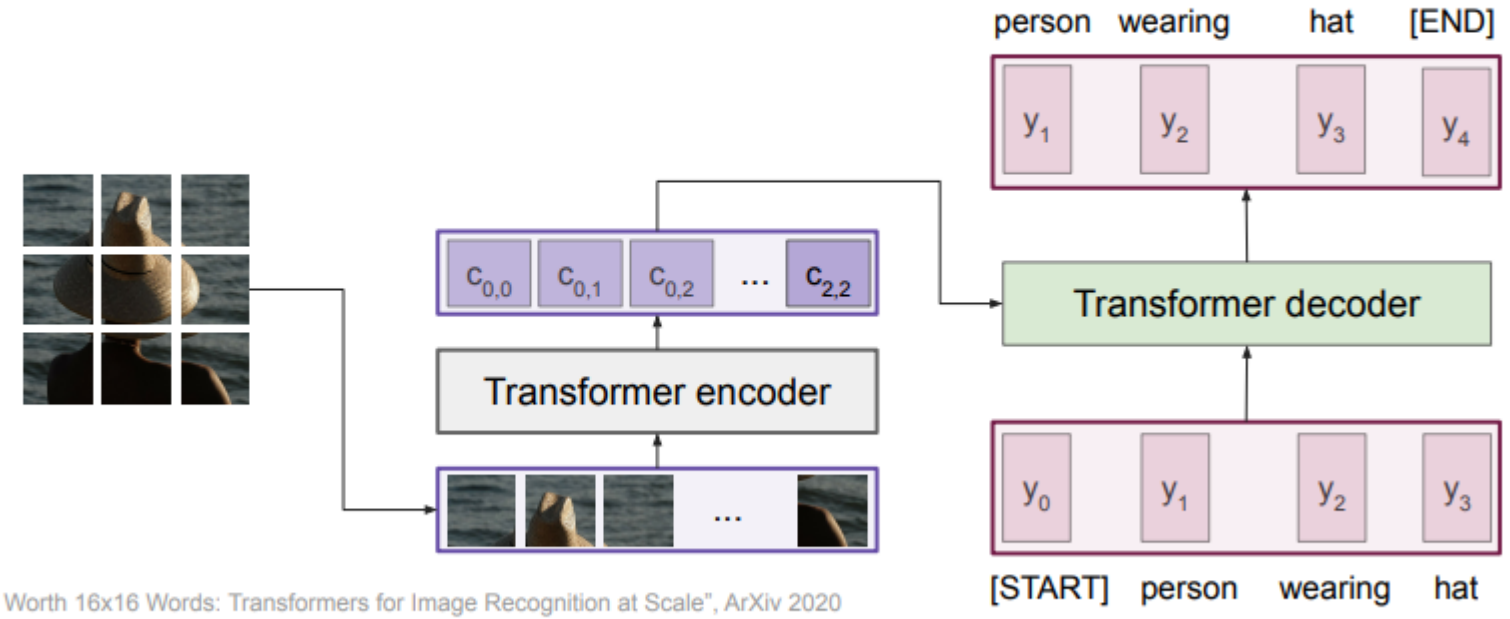

The Transformer Decoder block
person wearing hat [END]
y_0 y_1 y_2 y_3
x N
Transformer decoder
$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ $c_{2,2}$
[START] person wearing hat
FC
Layer norm
MLP
Layer norm
Multi-head attention
k v q
Layer norm
Masked Multi-head self-attention
Positional encoding
x_0 x_1 x_2 x_3
Transformer Decoder Block:
Inputs: Set of vectors x and Set of context vectors c.
Outputs: Set of vectors y.
Masked Self-attention only interacts with past inputs.
Multi-head attention block is NOT self-attention. It attends over encoder outputs.
Highly scalable, highly parallelizable, but high memory usage.
Vaswani et al, "Attention is all you need", NeurIPS 2017
Fei-Fei Li, Ranjay Krishna, Danfei Xu
Lecture 11 - 79
May 06, 2021

Image Captioning using ONLY transformers

- **Transformers from pixels to language**



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020
Colab link to an implementation of vision transformers

Summary

- Transformers are a type of layer that uses self-attention and layer norm.
  - ✓It is highly scalable and highly parallelizable
  - ✓Faster training, larger models, better performance across vision and language tasks
  - ✓ They are quickly replacing RNNs, LSTMs, and may even replace convolutions.

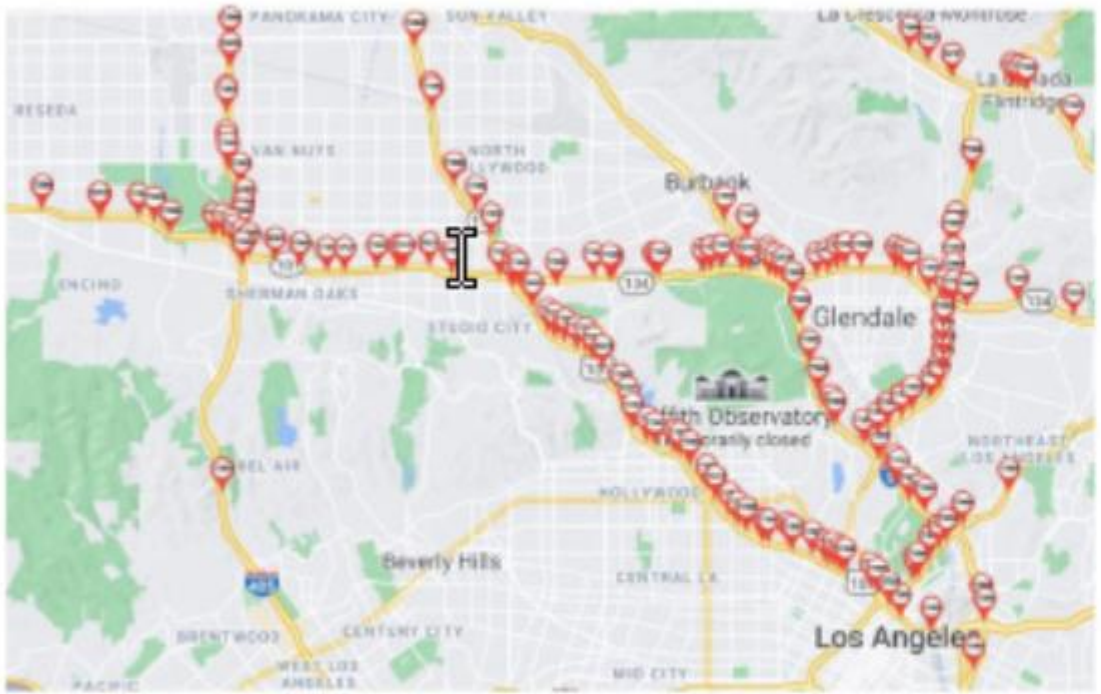
Transformer Model for Traffic Flow Prediction

Daejeon City Traffic Data: UVDS

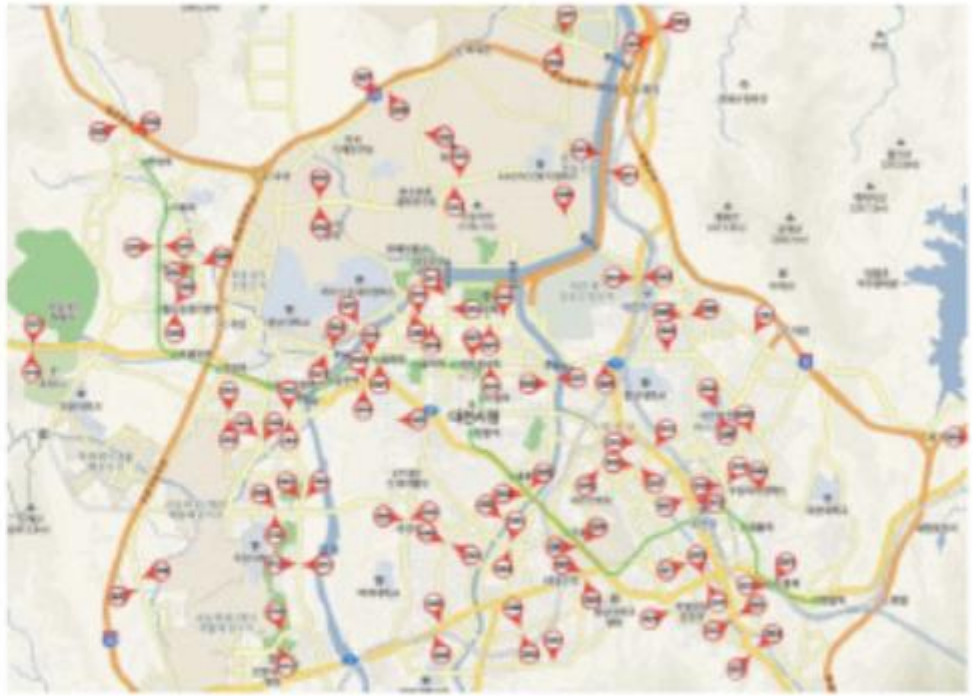
❖ 약 1.4km 왕복 4차로, RSE 3개, 영상검지기 5개(현재 3개, 향후 2개)

UVDS: Urban VDS Dataset

- Tra±c data have been collected from VDS
  - ✓speed, traffic volume, occupancy, and vehicle types.

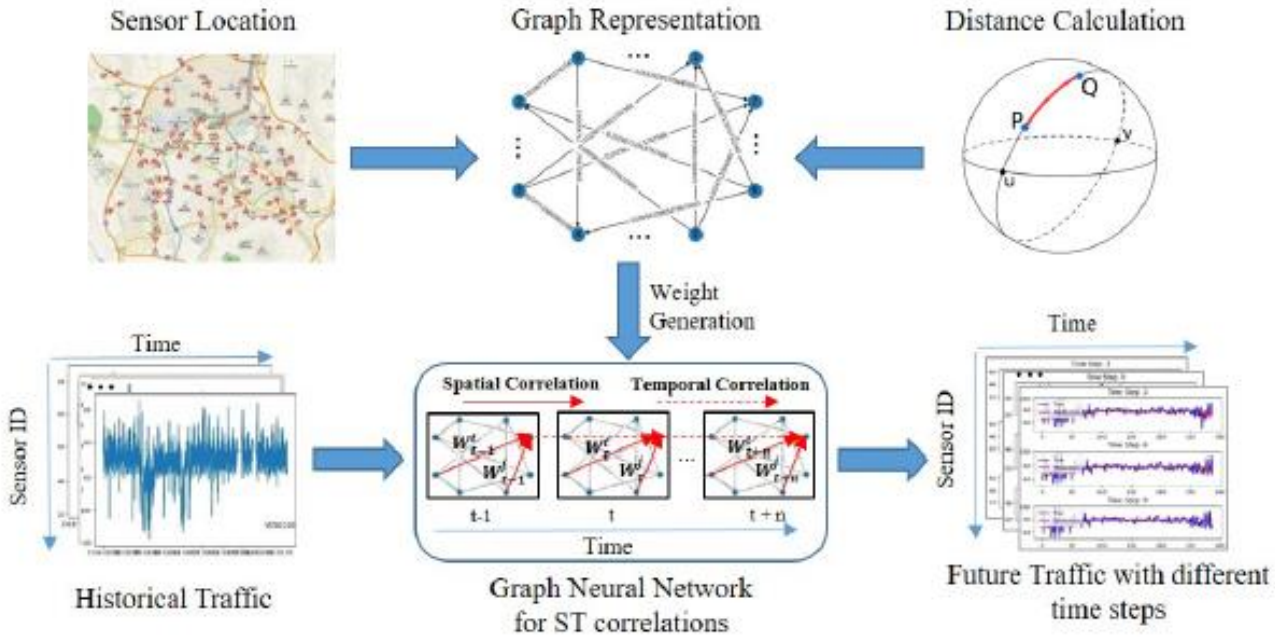


(a) METR-LA



(b) UVDS

UVDS: A New Dataset for Trac Forecasting with Spatial-Temporal Correlation



Spatial–temporal Graph neural Network

For graph construction, UVDS based on the geometric distances between sensors (N=104)

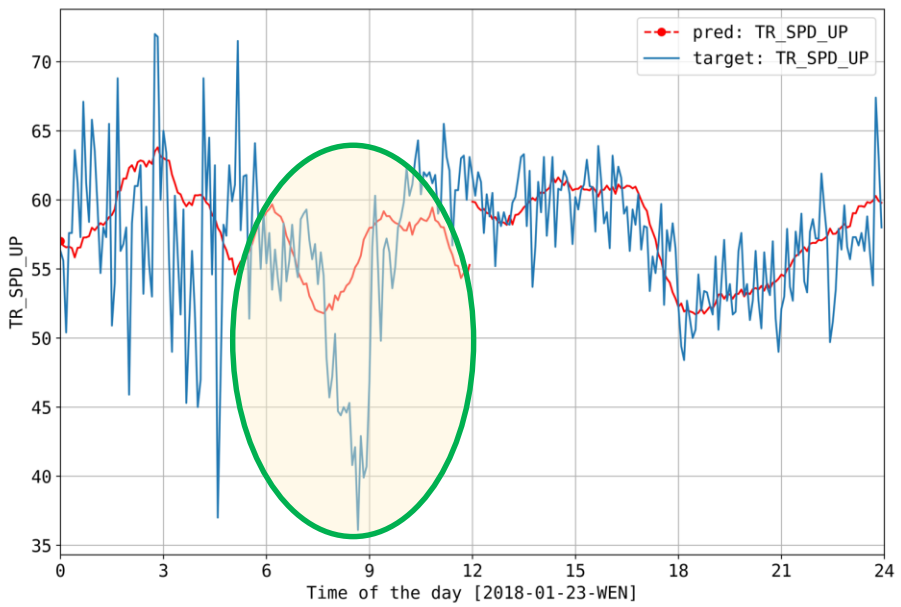
$$W_{i,j} = \begin{cases} \exp\left(-\frac{d_{v_i,v_j}^2}{\sigma^2}\right), & \text{if } \exp\left(-\frac{d_{v_i,v_j}^2}{\sigma^2}\right) \geq \beta, \\ 0, & \text{otherwise,} \end{cases}$$

$$d_{v_i,v_j} = 2r\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_j - \phi_i}{2}\right) + \cos(\phi_i)\cos(\phi_j)\sin^2\left(\frac{\varphi_j - \varphi_i}{2}\right)}\right)$$

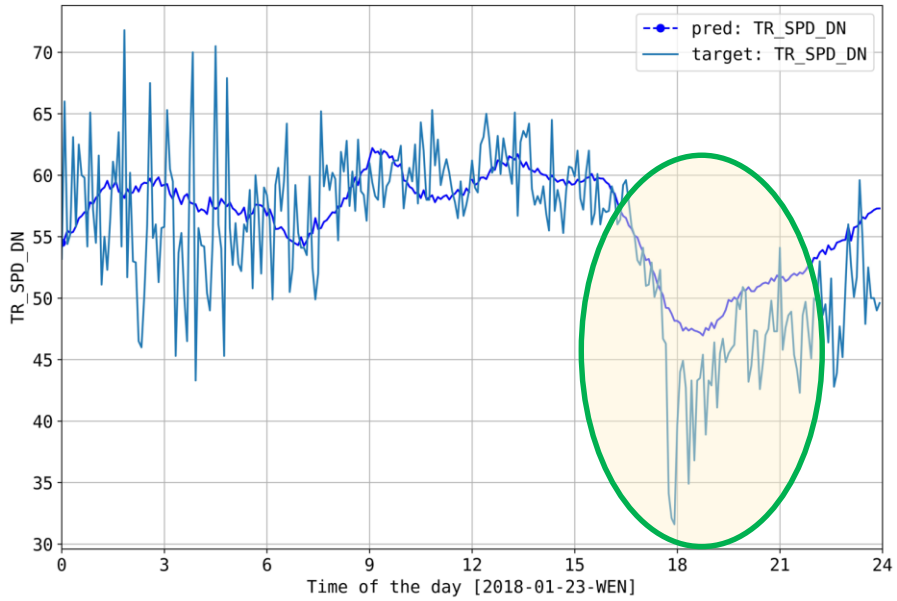
LSTM을 이용한 교통 흐름 예측 및 새로운 모델의 필요

LSTM 예측은 RNN에 비하여 성능은 좋은나, Long-Sequence 장기예측에는 성능저하가 발생한다.

상행 12시간예측 (2018.01.23.,수요일)



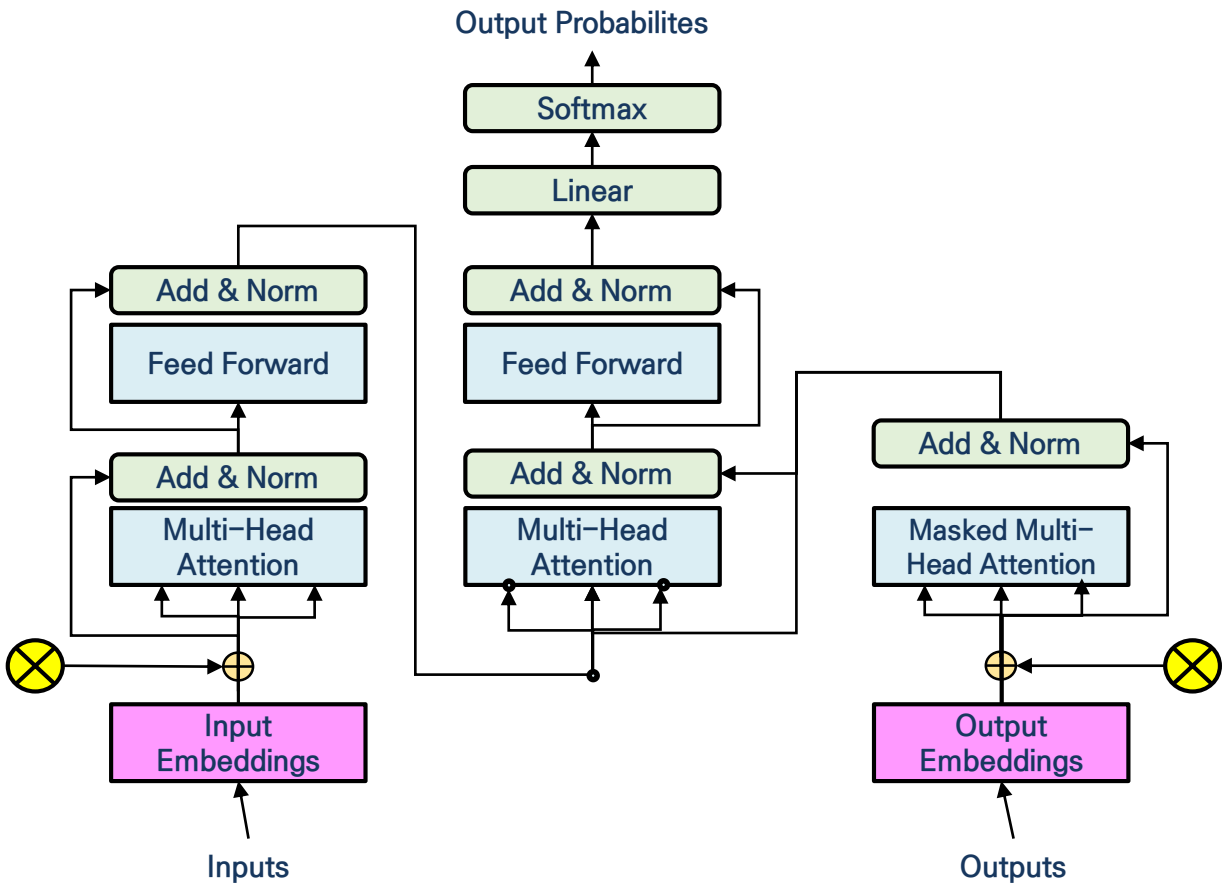
하행 12시간예측 (2018.01.23.,수요일)



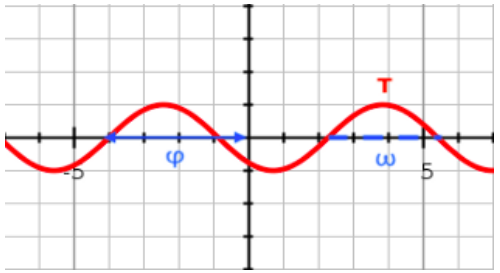
LSTM 장기예측(12시간)과 예측 정확은 재고할 필요가 있다. → 새로운 모델이 필요하다.

☆ LSTM 장기 예측(12시간) 결과는 상대적으로 정확도가 현저히 떨어진다.

Positional Encoding : Time2Vec



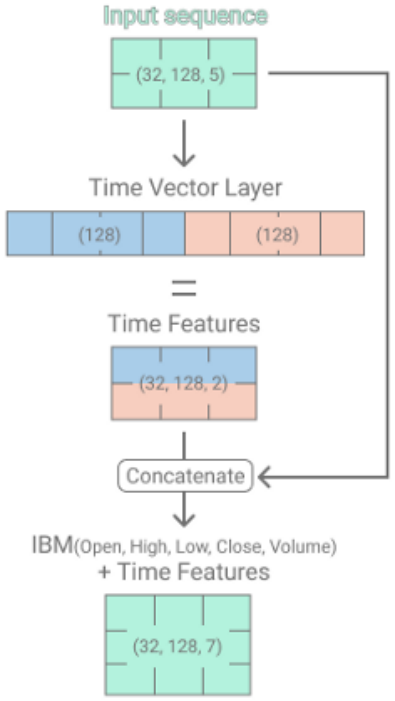
$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$



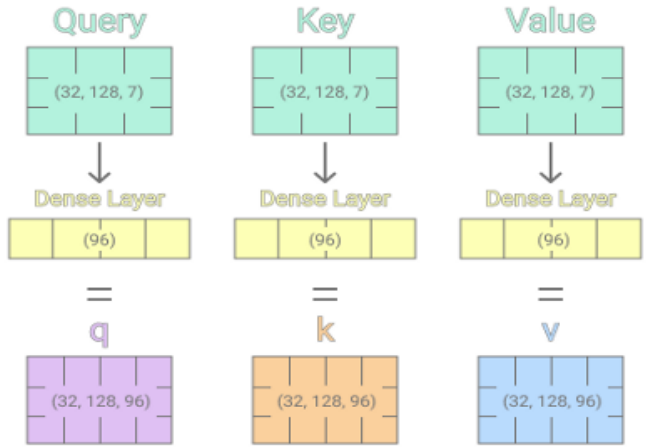
Attention : Query, Key, Value

Time2Vec

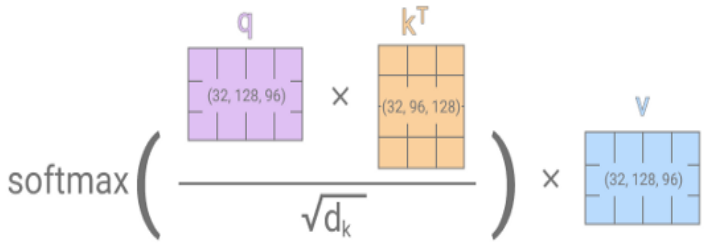
- 5개 VDS 17, VDS 18, VDS 19, VDS 20, VDS 21
- 트랜스포머 모델 입력크기(input-size)
  - batch_size: 32, sequence_length: 128, feature : 5



Single Head Attention : Query, Key, Value



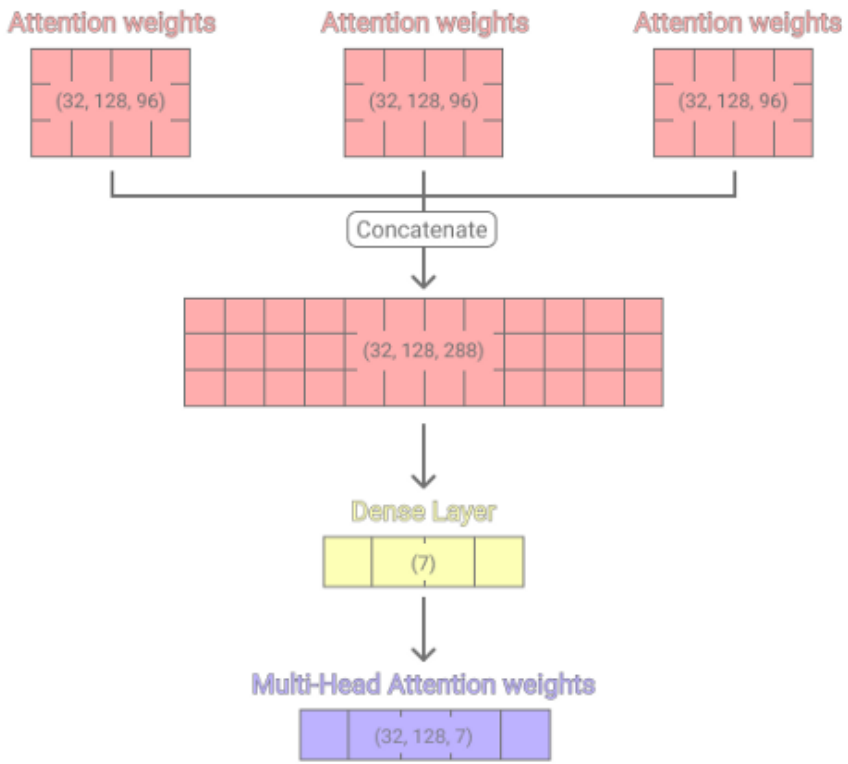
- Attention Weight는 Softmax를 사용함. Q*K*V



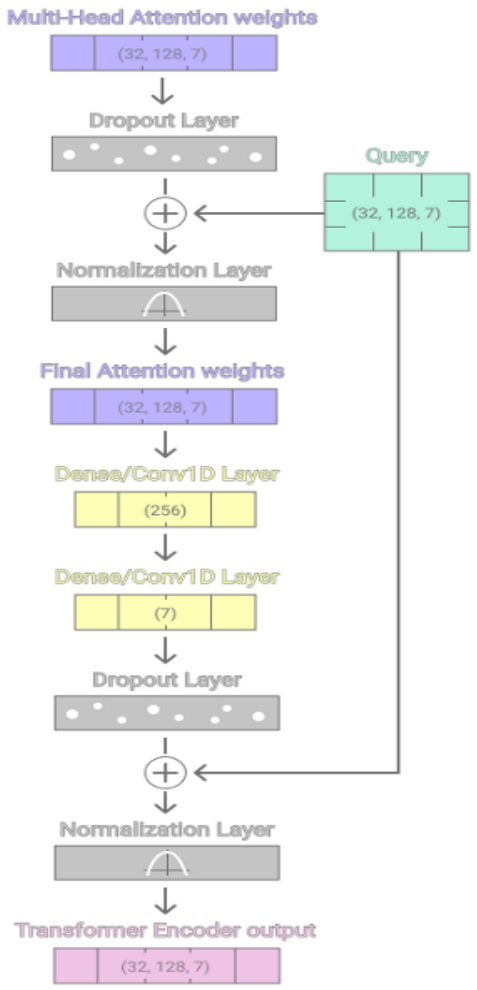
Transformer Traffic Model

Mulit-Head Attention으로 병렬처리가 가능함

❖ VDS 17, VDS 18 데이터 구조 (train, test, validation)

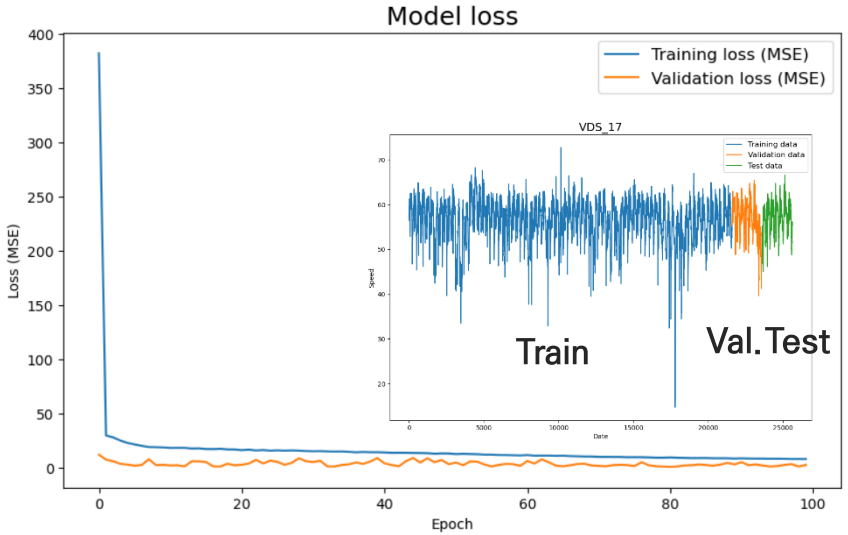


Transformer Encoder 출력이되 Decoder에 사용됨



Long-Term prediction of Traffic Speed and Volume

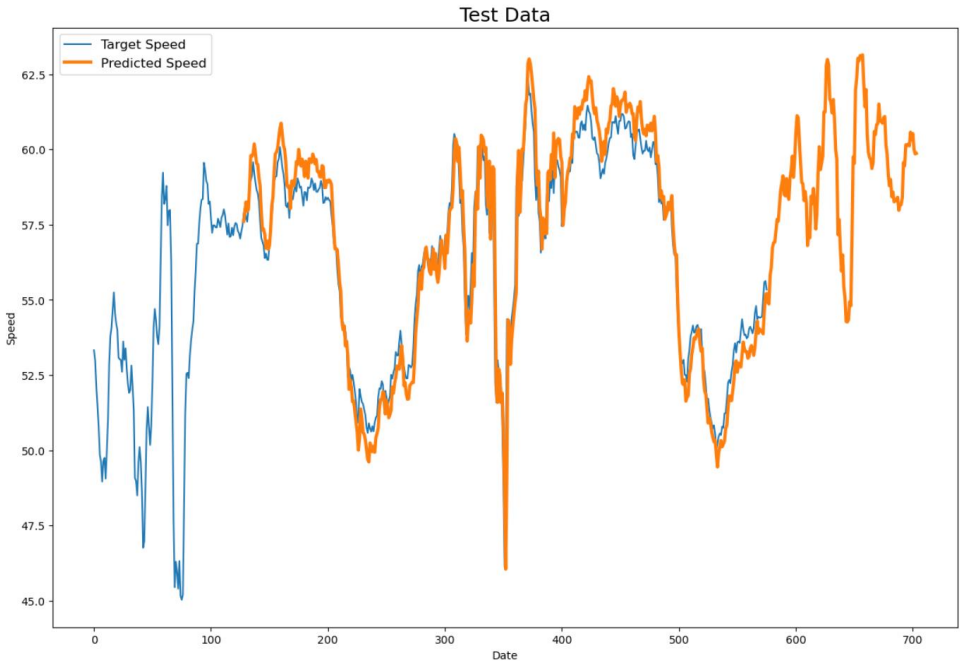
대전시 차량검지기(VDS) 데이터 105개



- 대학로 관련 5개 위치 데이터 90일
  - 데이터 개수: 25,920개
  - 90일*24*12 (5분 단위)

장기예측 성능 : 2016개 예측(1주일, 288*7)

- TransformerVDS: Transformer+TimeEmbedding



- TransformerVDS 결과와 대전 UVDS 데이터에 적용하고 있으며, SCI 논문에 투고할 예정

DSTWN : Dynamic Spatial Transformer WaveNet Network로 Spatial-Temporal Graph

❖ 기존 Spatial-Temporal Graph Network과 비교 성능 향상 목표

❖ 개발한 DSTWN 알고리즘 성능 벤치마크 : Metr-LA 데이터

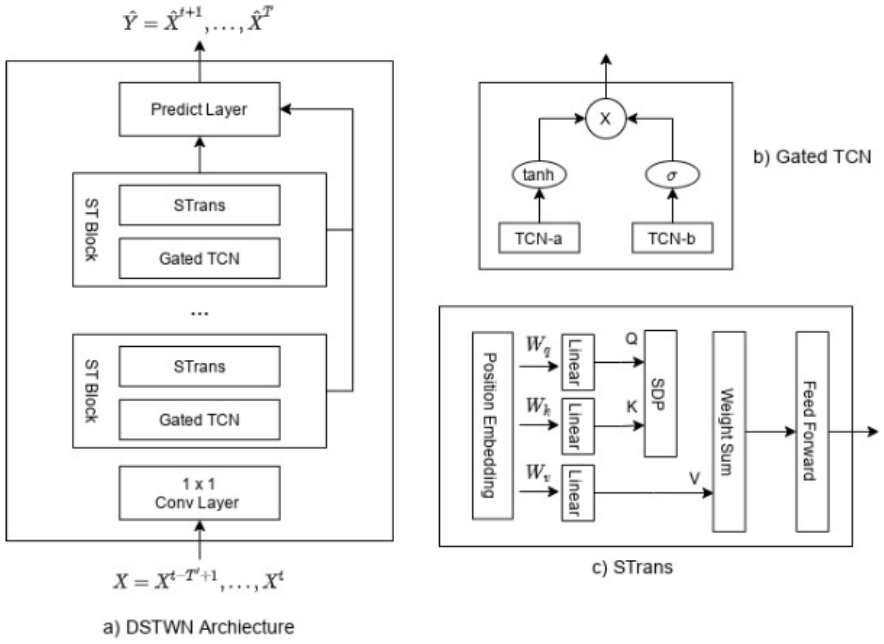


Fig. 3. The DSTWN Architecture

Table 2. Results on METR-LA dataset.

Step	Metrics	GraphWN	MTGNN	STWaNet	STTN	DSTWN
1	MAE	2.2408	2.2441	2.2791	2.4080	2.3220
	RMSE	3.8636	3.9185	3.9793	4.2339	4.0704
	MAPE	0.0540	0.0551	0.0550	0.0588	0.0567
3	MAE	2.7127	2.6762	2.7457	2.9160	2.8758
	RMSE	5.1690	5.1428	5.3162	5.6556	5.6285
	MAPE	0.0695	0.0686	0.0711	0.0778	0.0759
6	MAE	3.0974	3.0605	3.0947	3.3819	3.2909
	RMSE	6.1839	6.2002	6.2781	6.8651	6.7740
	MAPE	0.0847	0.0816	0.0838	0.0958	0.0900
9	MAE	3.3617	3.3100	3.3239	3.6961	3.5461
	RMSE	6.8279	6.8380	6.8639	7.5749	7.3837
	MAPE	0.0950	0.0912	0.0924	0.1085	0.0984
12	MAE	3.5760	3.4937	3.5036	3.9533	3.7446
	RMSE	7.2883	7.2421	7.2761	8.1262	7.8246
	MAPE	0.1035	0.0982	0.0993	0.1181	0.1047
Training		45.6927	62.8770	54.5744	77.5496	133.2374
Inference		1.4630	1.6825	1.5830	6.6945	3.8088

KISTI SINCE 1962

2022 TRUST KISTI

Korea Institute of Science and Technology Information