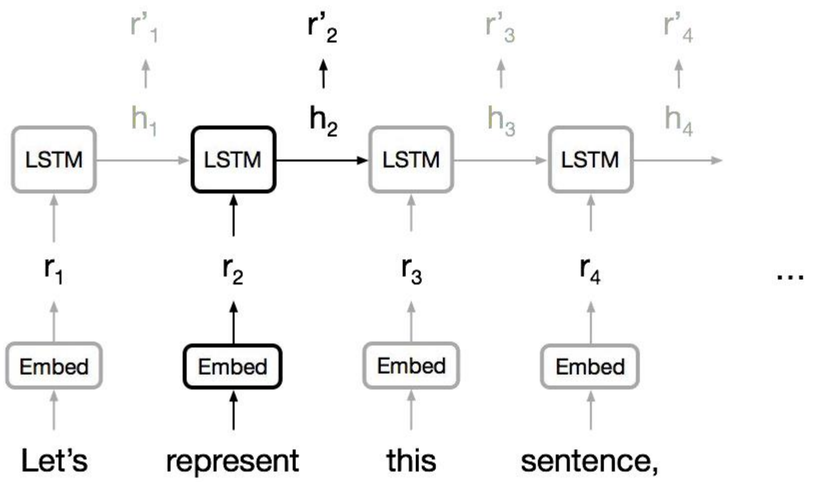
CS224n Lecture 14
Self-Attention For Generative Models

ESC 5조

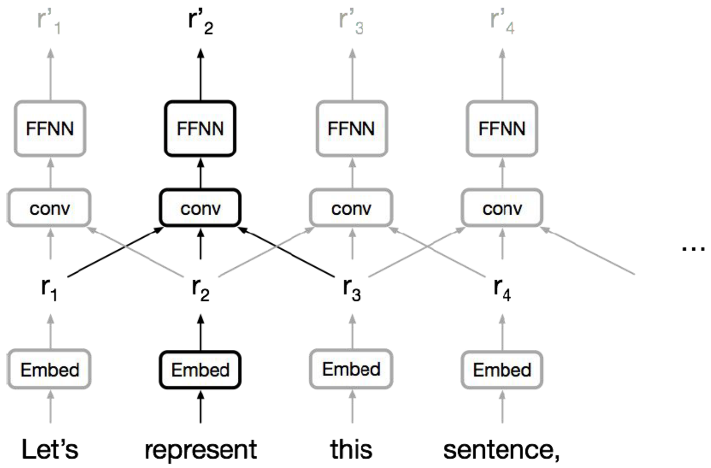
RNN



- Sequential computation -> parallelization 제한
- Want to model hierarchy
- Long / short range dependencies에 명확한 모델링 x

WASTEFUL!

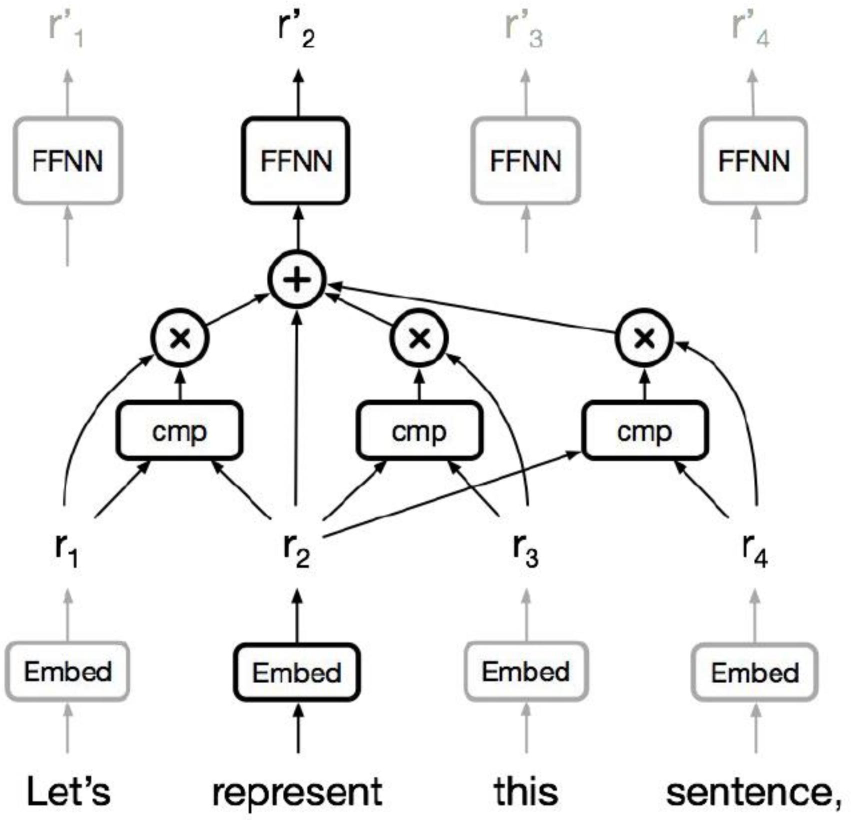
Convolutional Neural Networks



- Not sequentially -> But Depth
- Parallelize per layer <- apply convolutions simultaneously at every position
- Local dependencies
- Interaction distance btw positions linear or logarithmic

Long-distance Dependency
-> MANY LAYERS!

Self-Attention



…

어떤 단어의 새로운 representation을 만든다고 가정

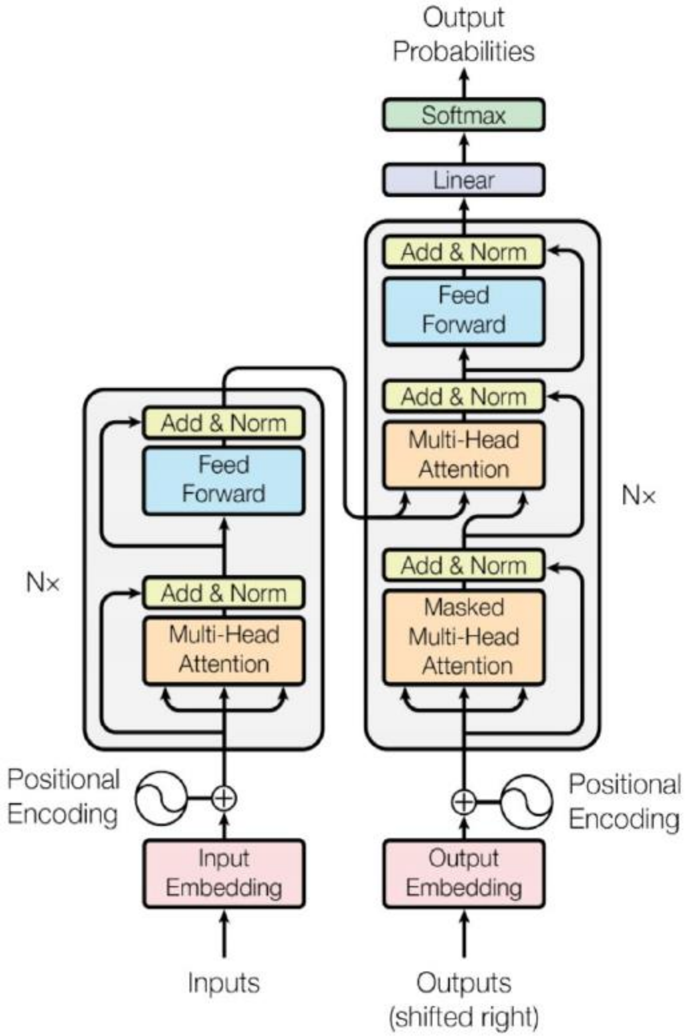
Attention의 역할 : re-expressing the word in certain terms of a weighted combination of its entire neighborhood

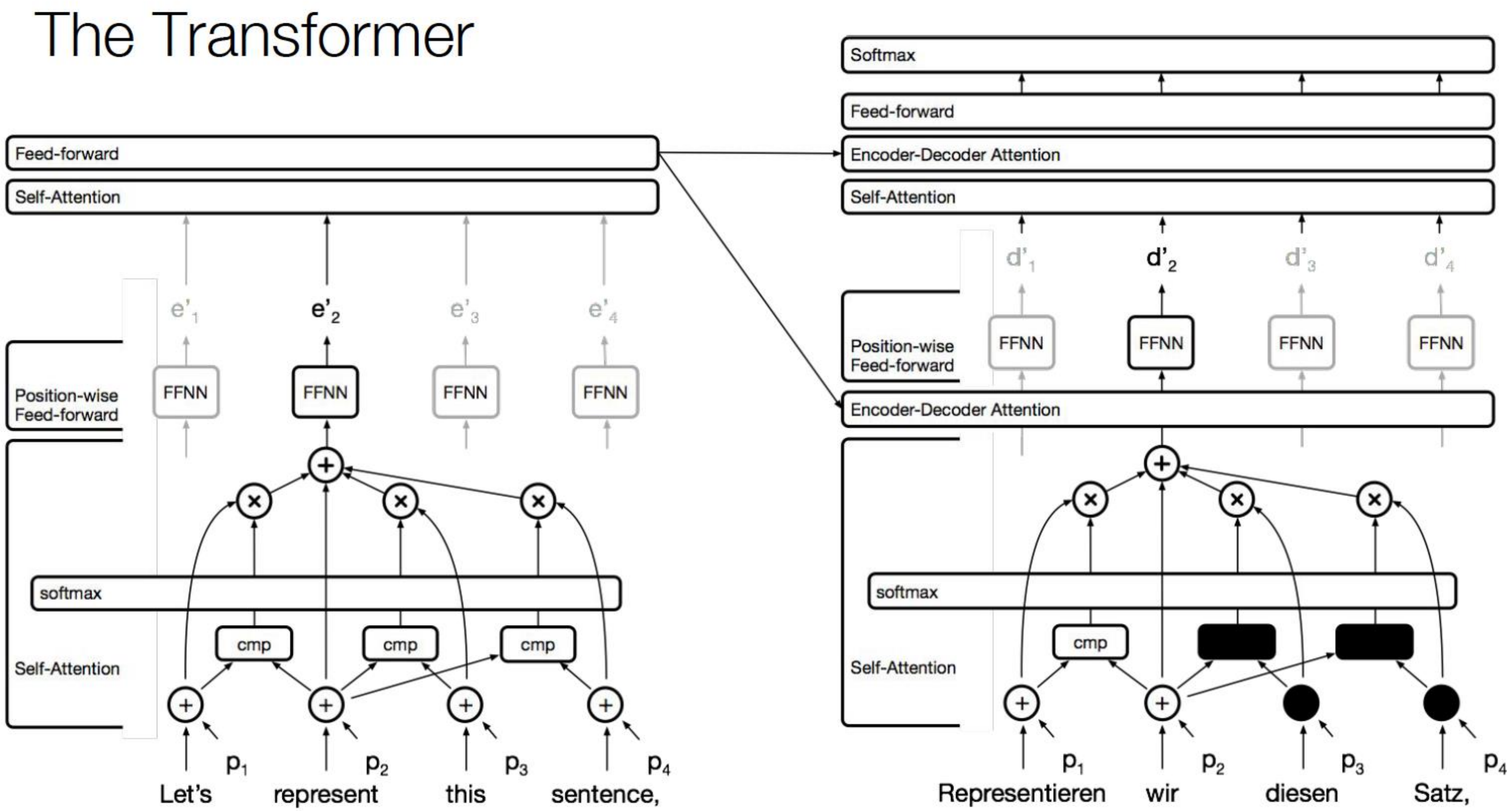
얘를 기반으로 모든 정보를 요약

- Constant 'path length' -> position can interact with any position
- Gating/multiplicative interactions
- Parallelize per layer <- attention just needs matmuls

Text Generation

Transformer – Using Attention primarily for computing Representations





Transformer – Why Attention? Cheap & Fast

Dimension이 length보다 훨씬 클 때
self-attention 모델이 아주 좋다

FLOPS

Self-Attention	O(length^2 · dim)	= $4 \cdot 10^9$
RNN (LSTM)	O(length · dim^2)	= $16 \cdot 10^9$
Convolution	O(length · dim^2 · kernel_width)	= $6 \cdot 10^9$

length=1000 dim=1000 kernel_width=3

Machine Translation: WMT-2014 BLEU
(Attention is All You Need)

	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	**28.4**	**41.8**

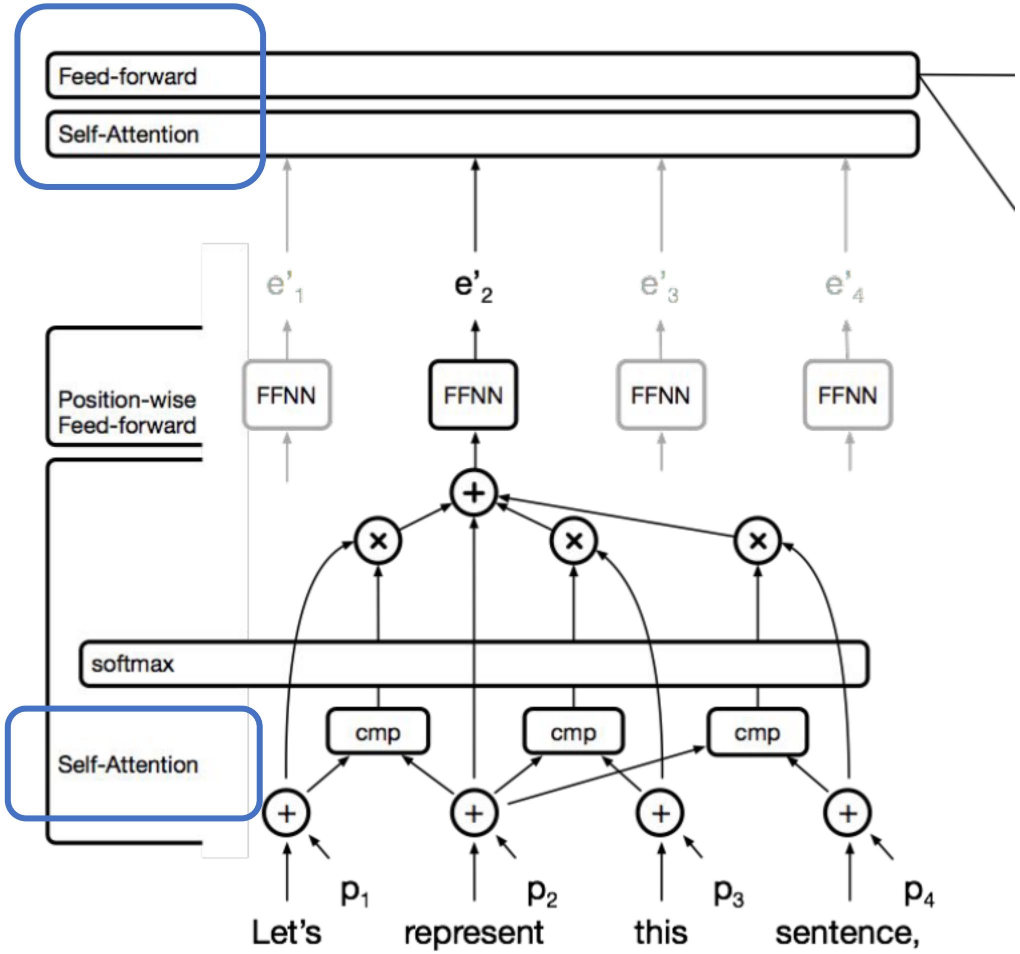
*Transformer models trained >3x faster than the others.

Transformer

Encoder Self-Attention : Simplicity & Speed

repeat

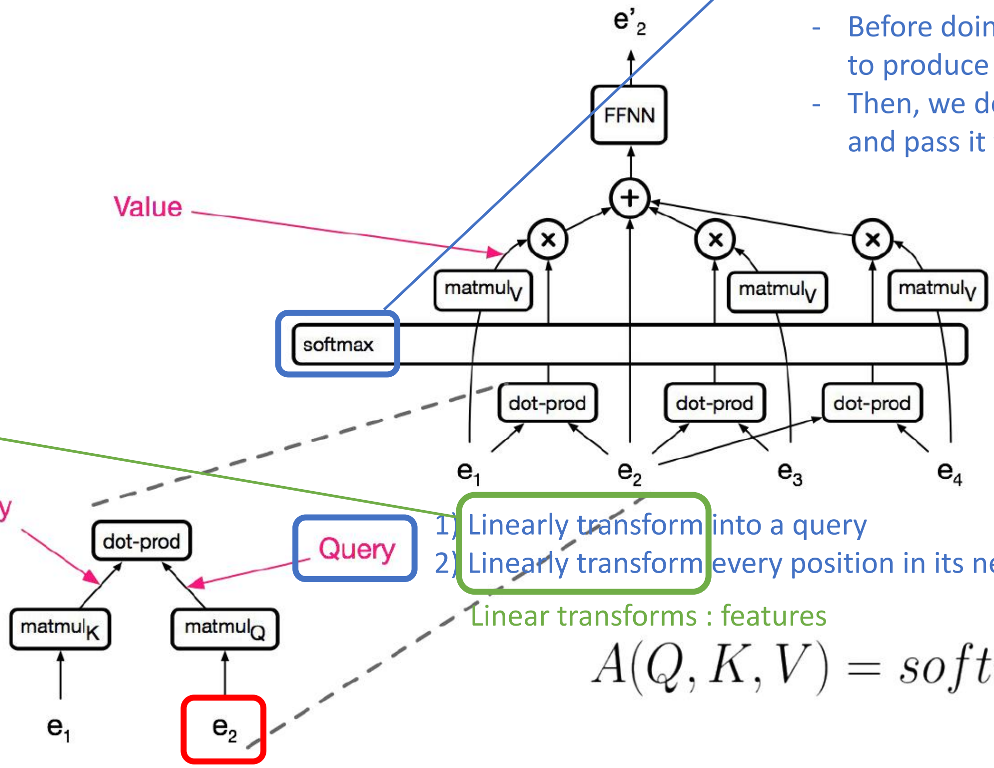
Just computes the representation for every position simultaneously using attention



Transformer

Encoder Self-Attention : Simplicity & Speed

Softmax computer convex combination
- Based on this convex combination, re-express e2 or in terms of this convex combination of all the vectors of all these positions
- Before doing the convex combination, again do a linear transformation to produce values
- Then, we do a second linear transformation just to mix this information and pass it through a feedforward layer



Value

Bilinear form
-> projecting these vectors into a space where dot product is a good proxy for similarity

Key

Query

1) Linearly transform into a query
2) Linearly transform every position in its neighborhood

Linear transforms : features

$$A(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

간단히 이 두개의 matrix multiplication으로 표현 가능

Scaling factor

얘를 re-represent하는 task라고 하자

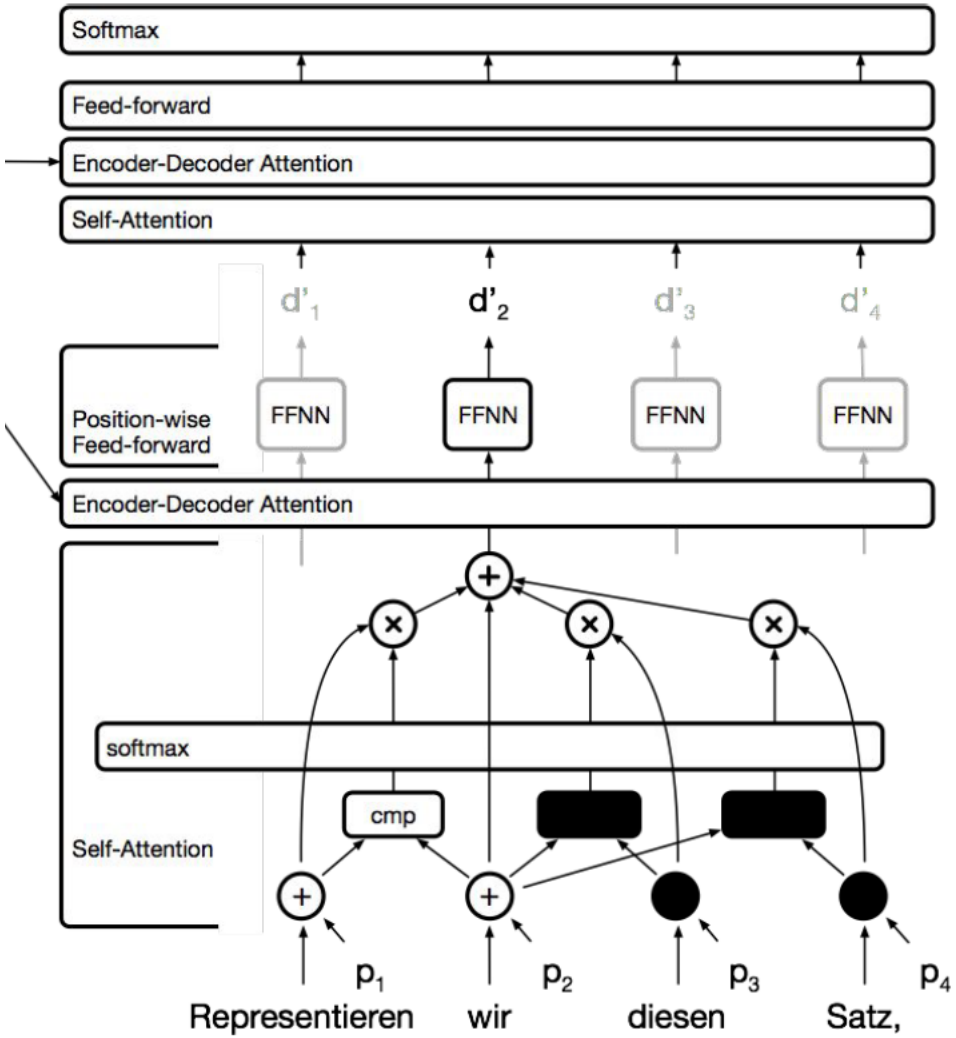
*) Residual Connections
: Btw every layer, and the input, we have a skip connection that just adds the activations

Transformer

Decoder Self-Attention : Simplicity & Speed

- Mimic a language model using self-attention to impose causality by just masking out positions that you can look at
- Has causal self-attention layer(followed by encoder-decoder attention)
- It cannot look forward. But it can look at itself because input was shifted.

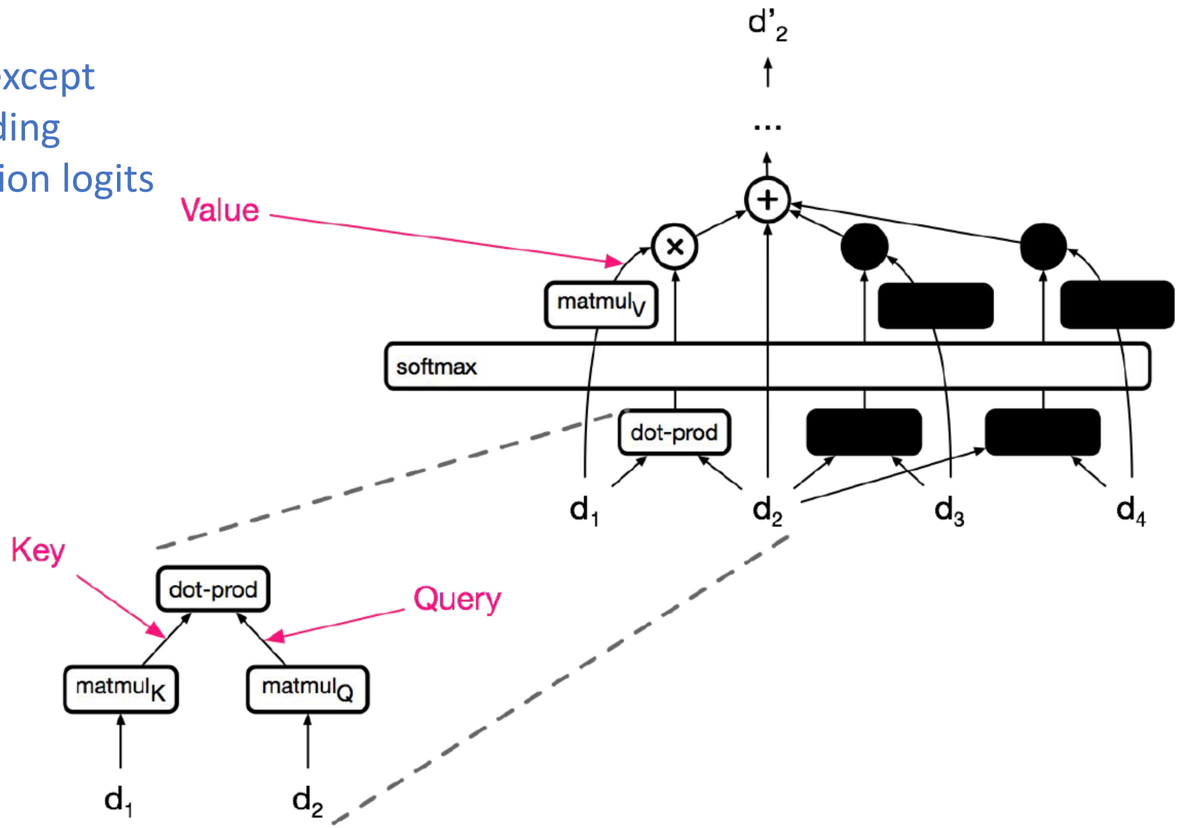
Standard cross-entropy loss



Transformer

Decoder Self-Attention : Simplicity & Speed

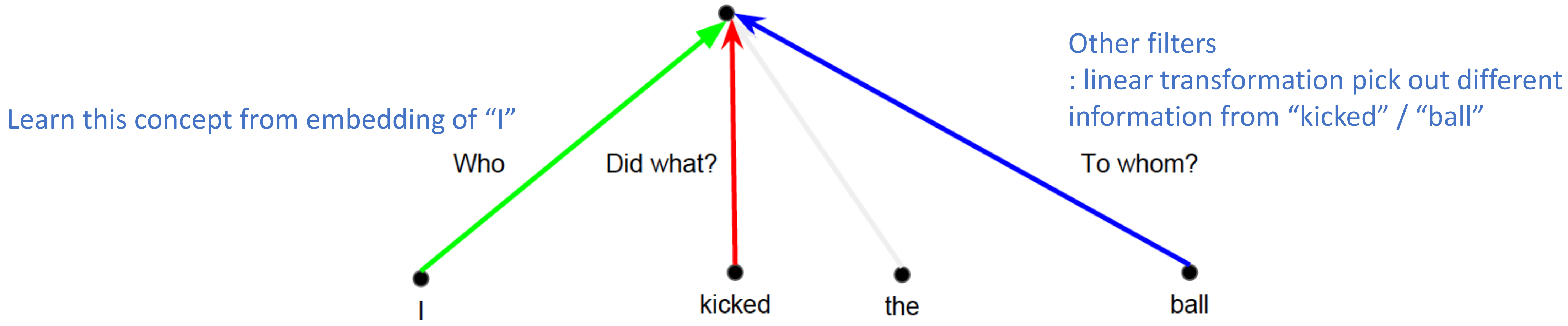
Exactly same same as the encoder except that we impose causality by just adding highly negative values on the attention logits



Compare

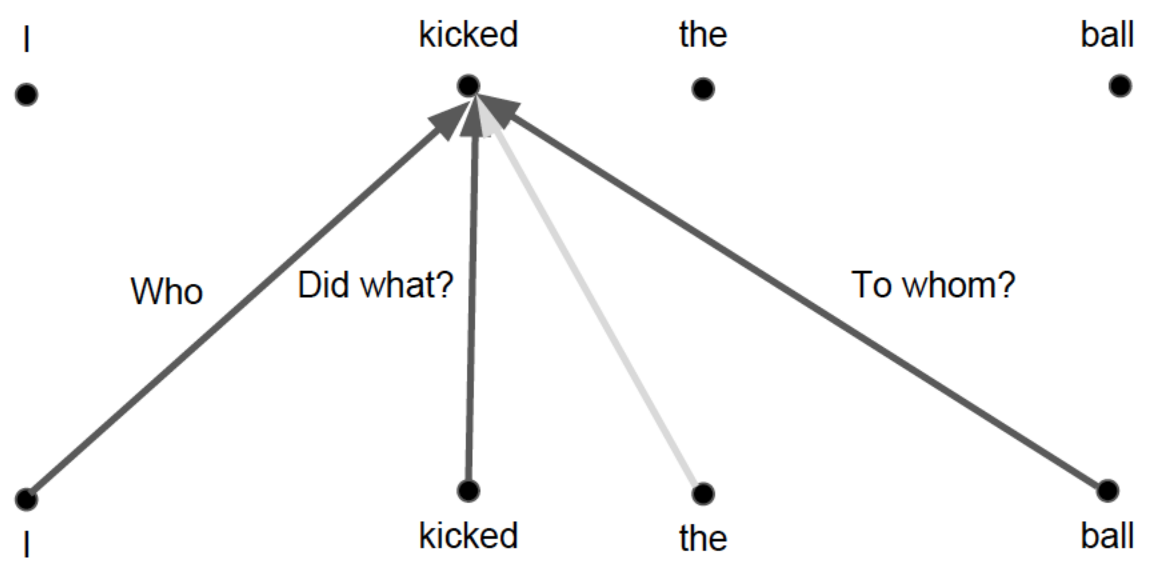
Convolutions

Convolution filters
: different transformation based on
relative distances



Compare

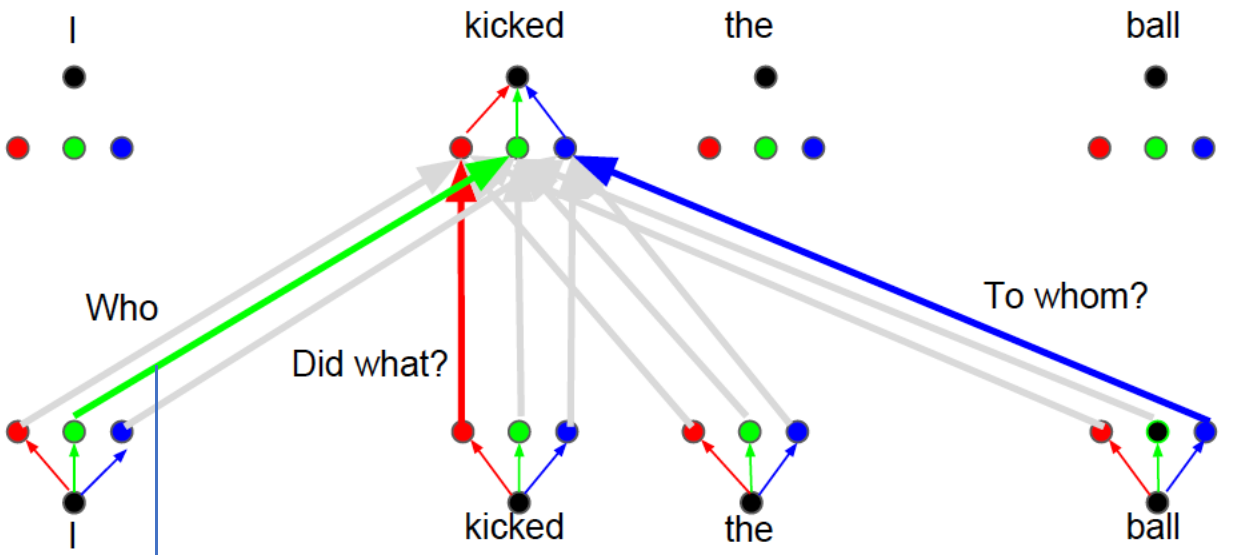
Self-attention – Single layer



Only possible
: just mixing proportions
-> it's impossible to pick out different pieces of information from different places

Compare

Self-attention – Parallel attention heads(Multihead Attention)



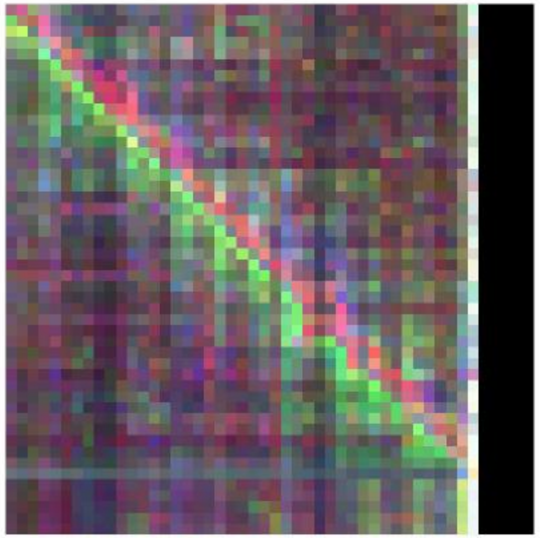
One attention layer for who
: feature detector
-> it carries with a linear transformation, so it's projecting them in a space which starts caring about syntax
Or, it would start caring about "who" or not

All of these can actually done in parallel!

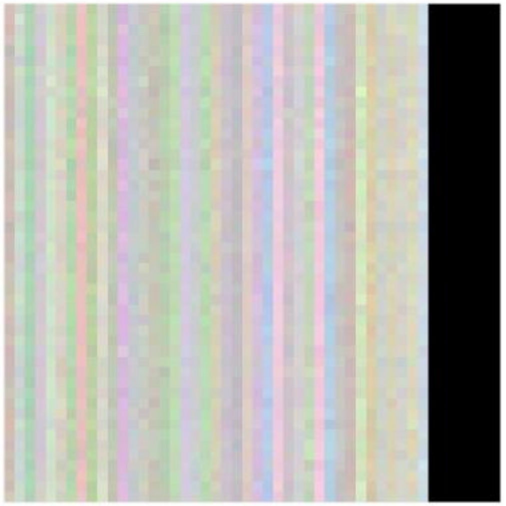
- For efficiency : instead of having these dimensions operating in a large space, we just reduce the dimensionality of all these heads
- Operating these attention layers in parallel is ort of bridging the gap

Importance of Residuals

Residuals – carry positional information to higher layers

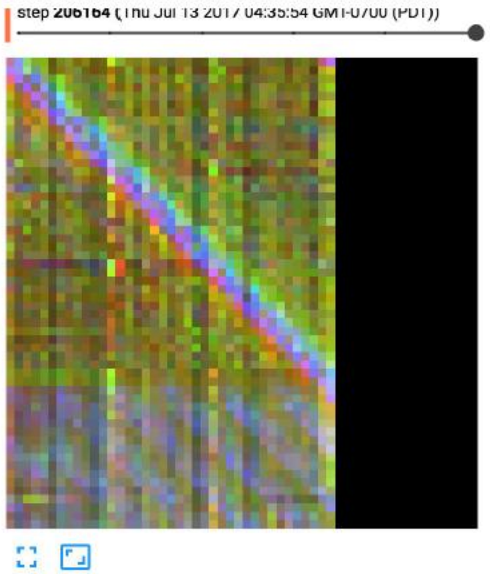


With residuals



Without residuals

Unable to pick diagonal

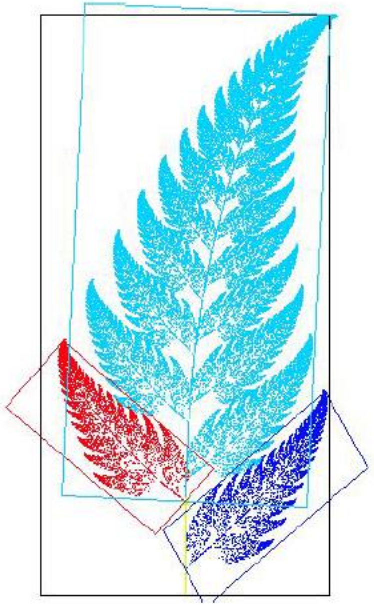


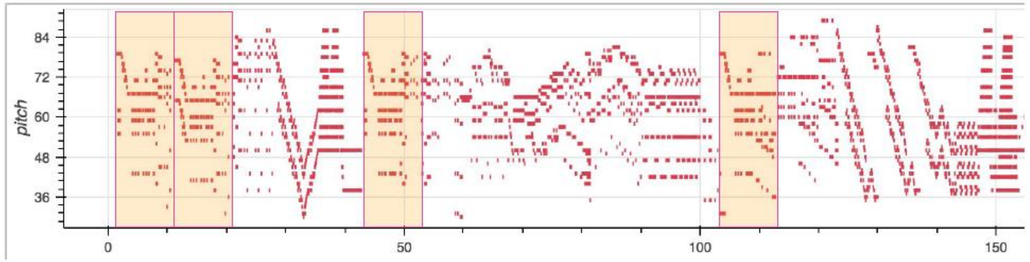
Without residuals,
with timing signals

Added position info back in at every layer
-> accuracy was not recovered but diagonal focus is back!!

Self-similarity

In many cases, images and music have <u>repetition</u> of patches in <u>different scales</u>





Probabilistic Image Generation

Can self-attention help us in modeling other objects like image?

Self attention으로 단어 뿐만 아니라 image 등도 모델링할 수 있을까?

- model the joint distribution of pixels
- standard auto-regressive image modeling 사용

    GAN이랑은 다름! Language Modeling을 image에 적용한 거라고 생각하자!

- probabilistic image modeling

Probabilistic Image Generation

Self-attention is a good computation mechanism!

CNN에 비해 self-attention을 image에 적용했을 때 좋은점?

- 기존까지 image generation task에서 SOTA를 달성한 모델은 모두 CNN 기반 (PixelRNN, PixelCNN)

  CNN은 병렬연산이 가능하기 때문에 계산속도가 더 빠르다

- But long-range dependencies matter for images!

  이미지에서는 어느 정도 원거리 유사성이 고려되어야 한다 (ex. symmetry)

  image size도 점점 커지는 추세 => 이러한 문제점을 해결하기 위해

CNN을 사용하면...?

- more layers ! 학습이 어려워진다..
- larger kernels !

  parameter/computation cost 증가..

Self-attention을 사용하면...?

- large receptive field => get attention at a lower computational cost
- *no need for layers!*

  (CNN의 경우 멀리 떨어진 pixel과의 관계를 찾기 위해 layer가 많이 필요)

Probabilistic Image Generation

Attention is Cheap!

only if <u>length</u> << dimension..

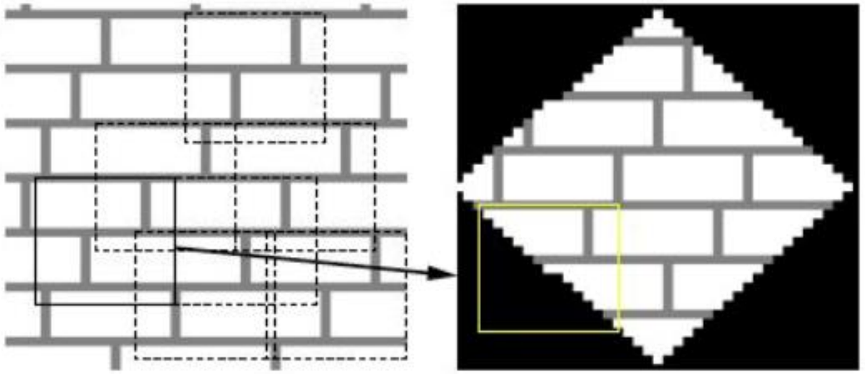
FLOPs

그냥 CNN 쓰자..

Self-Attention	O(length^2 · dim) (length=3072 for images)
RNN (LSTM)	O(length · dim^2)
Convolution	O(length · dim^2 · kernel_width)

Self-Similarity

Examples



Texture Synthesis with Self Similarity

Non-local Means (**image denoising**)

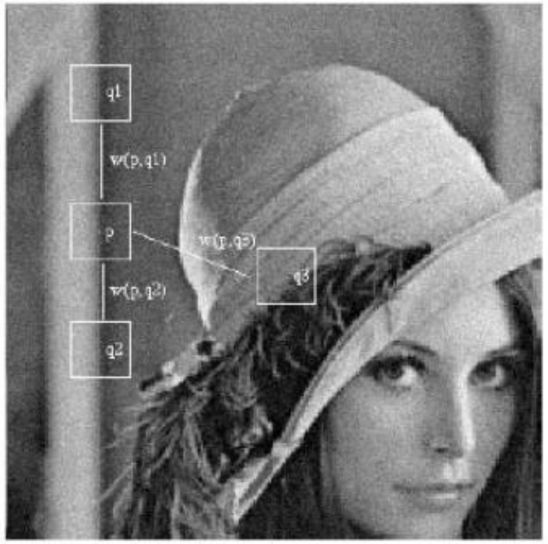


Figure 1. Scheme of NL-means strategy. Similar pixel neighborhoods give a large weight, w(p,q1) and w(p,q2), while much different neighborhoods give a small weight w(p,q3).

"denoise patch ***p***"

↓

based on similarity btw all other patches in image,

↓

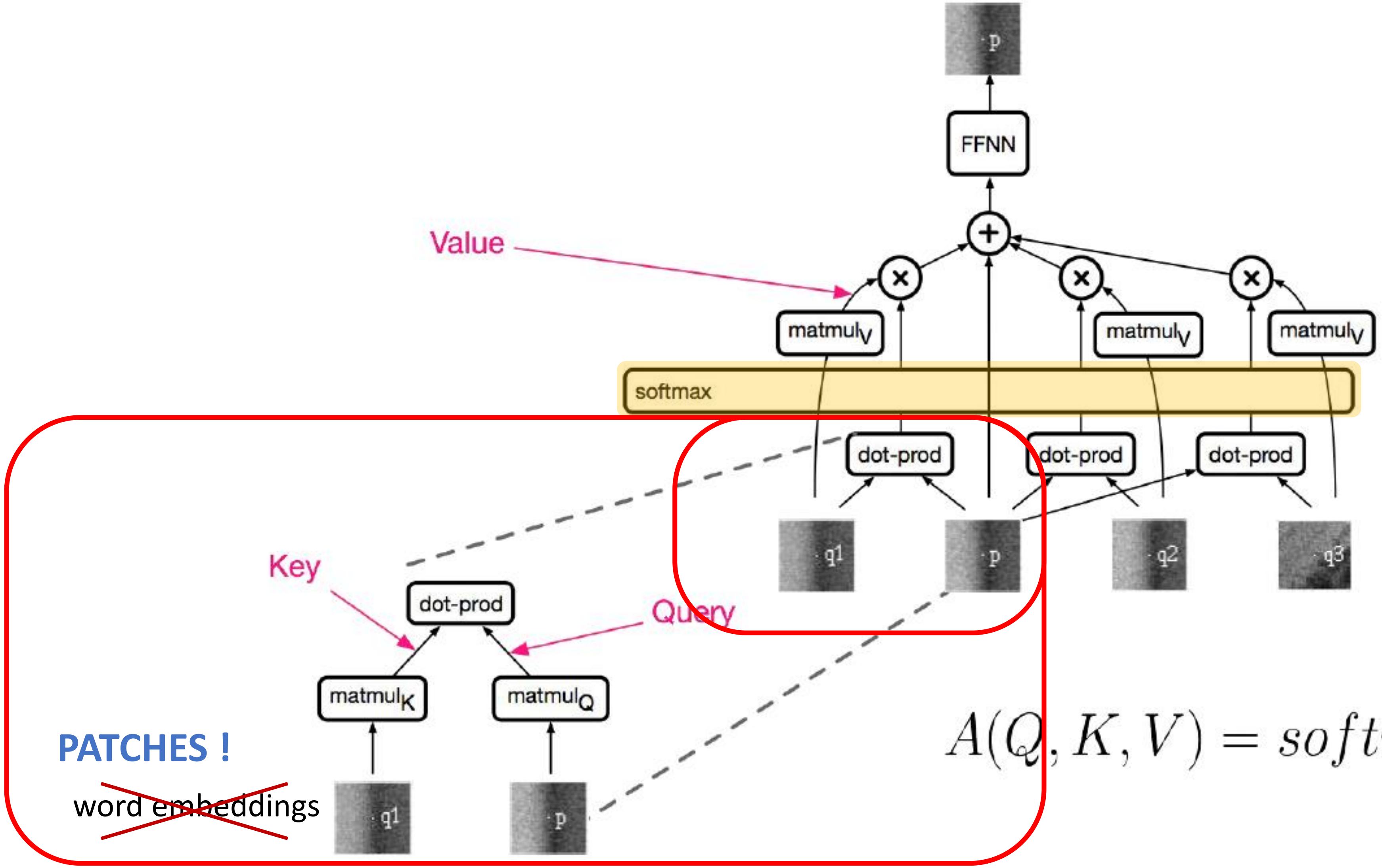
compute function of **content-based similarity**

↓

get information

Self-Similarity

patches instead word embeddings !



constructs convex combinations

↓

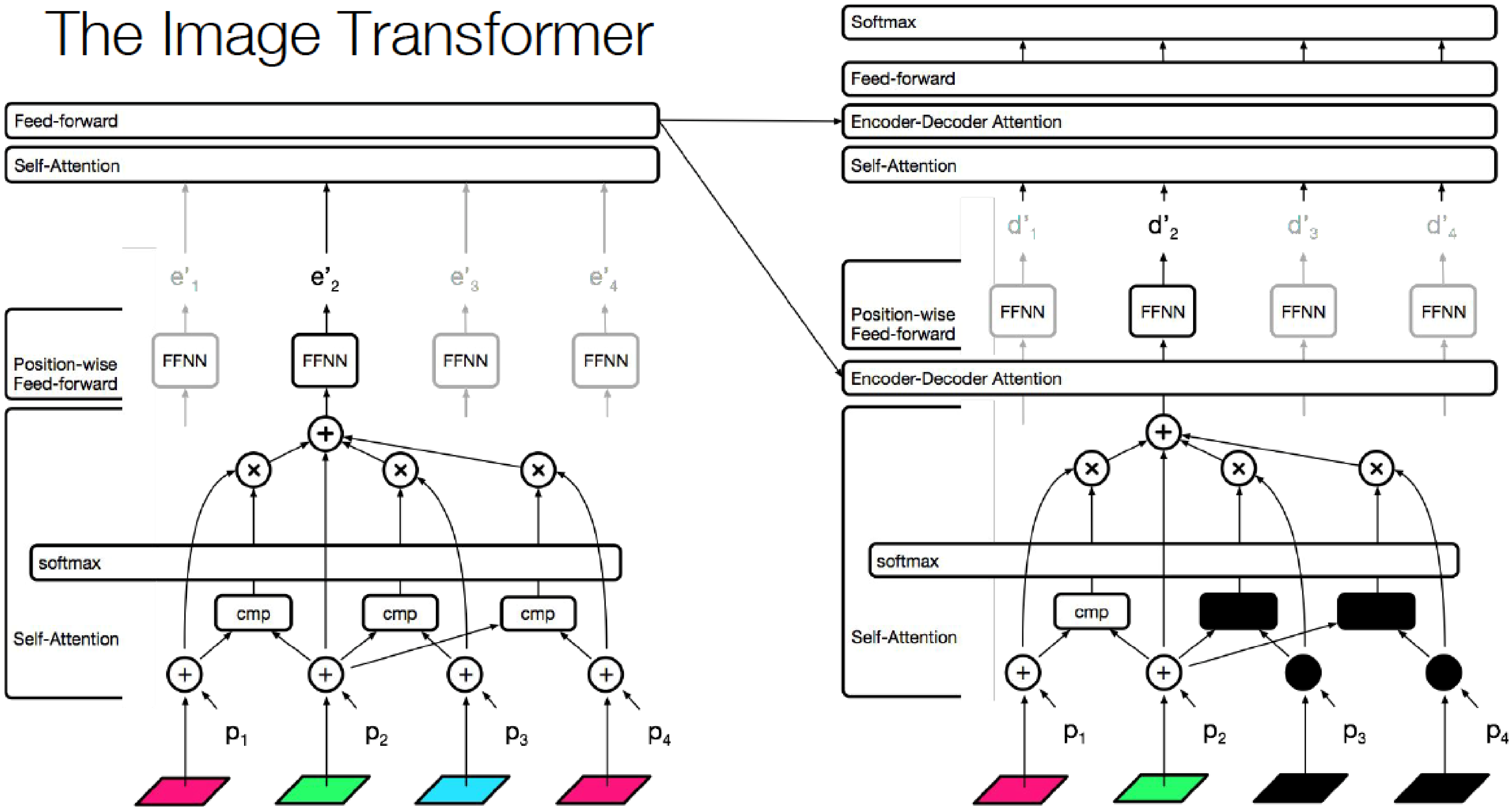
bring things together!

이미지의 서로 다른 patch끼리의

content-based similarity 계산

$$A(Q,K,V) = softmax\left(\frac{\boxed{QK^T}}{\sqrt{d_k}}\right)V$$

Overall Structure



Combine with Locality

Attention is not cheap for images :(

FLOPs

Self-Attention	O(length^2 · dim) (length=3072 for images)
RNN (LSTM)	O(length · dim^2)
Convolution	O(length · dim^2 · kernel_width)

- Restrict attention windows to be local neighborhoods

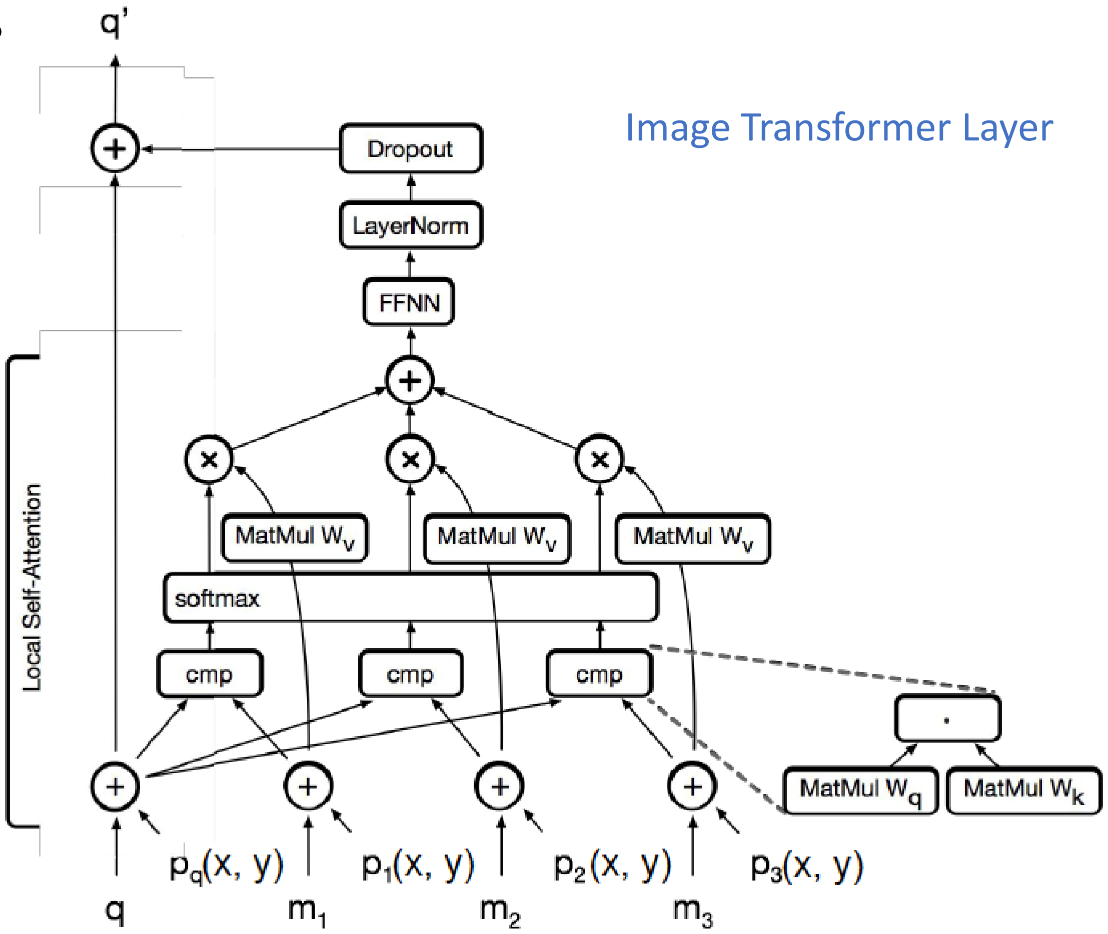


Image Transformer Layer

Usages and Results

Still more to go!

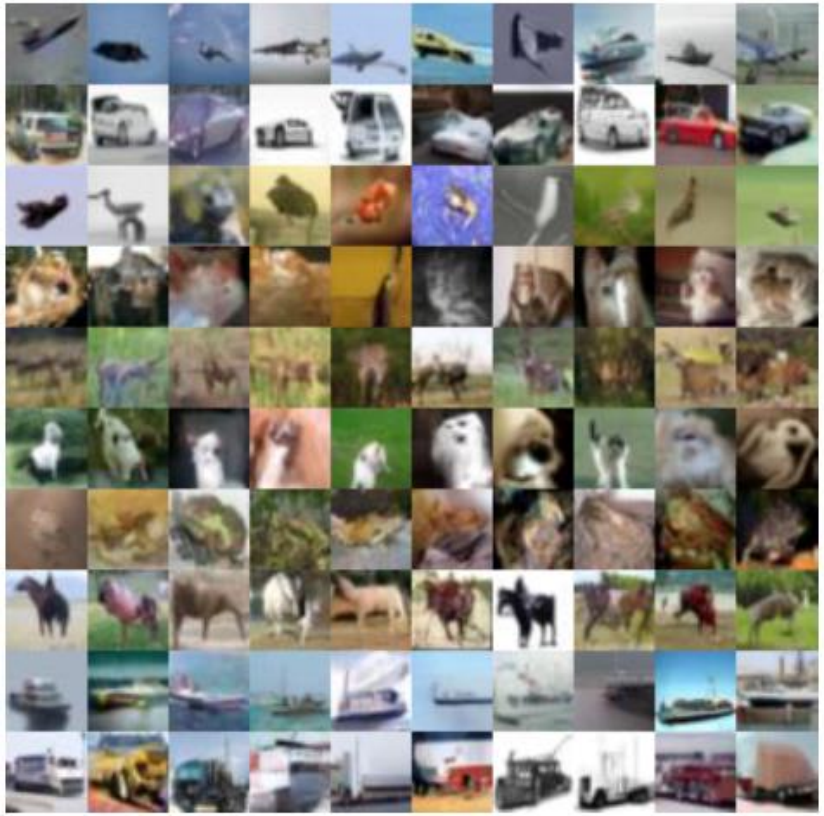
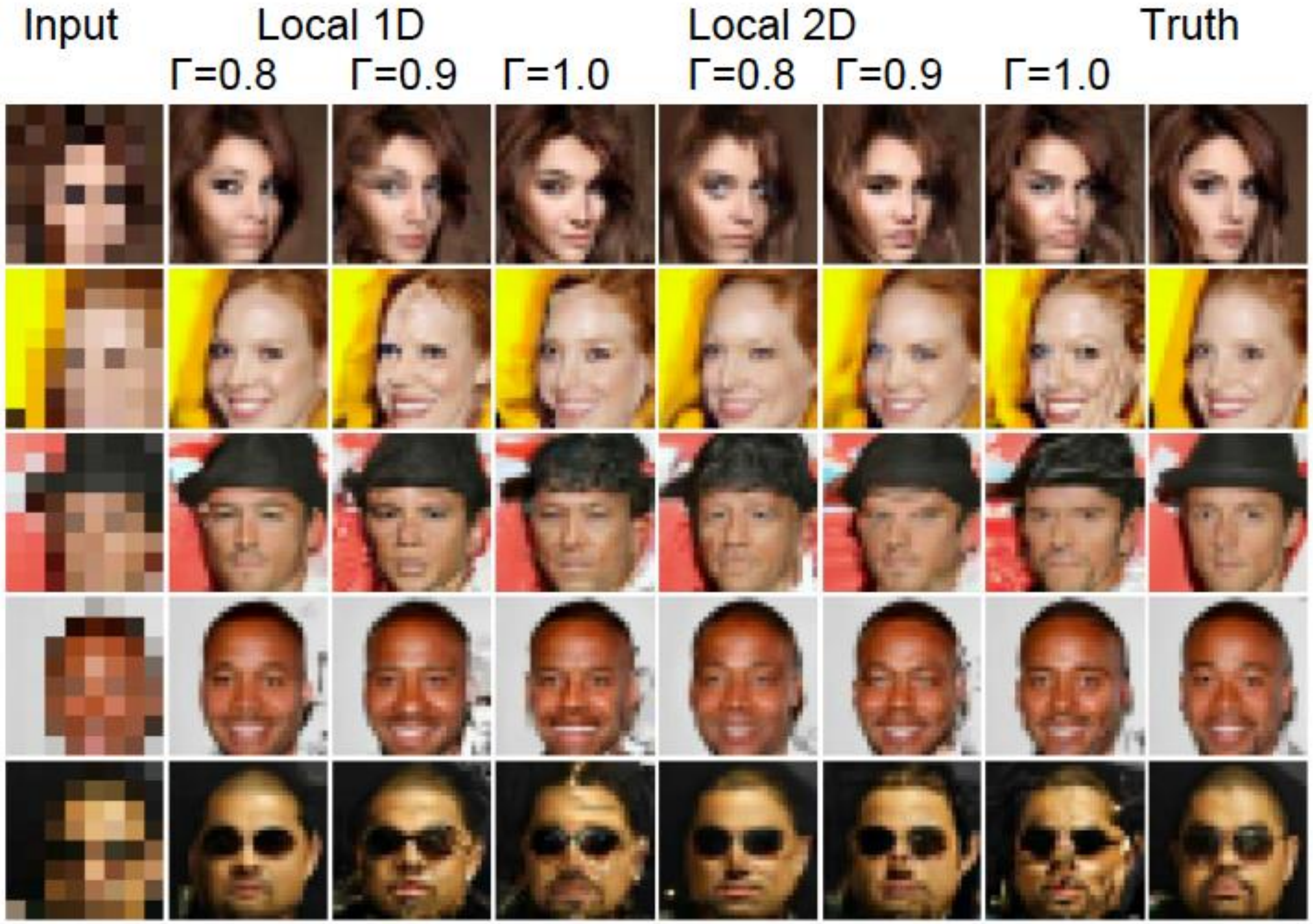


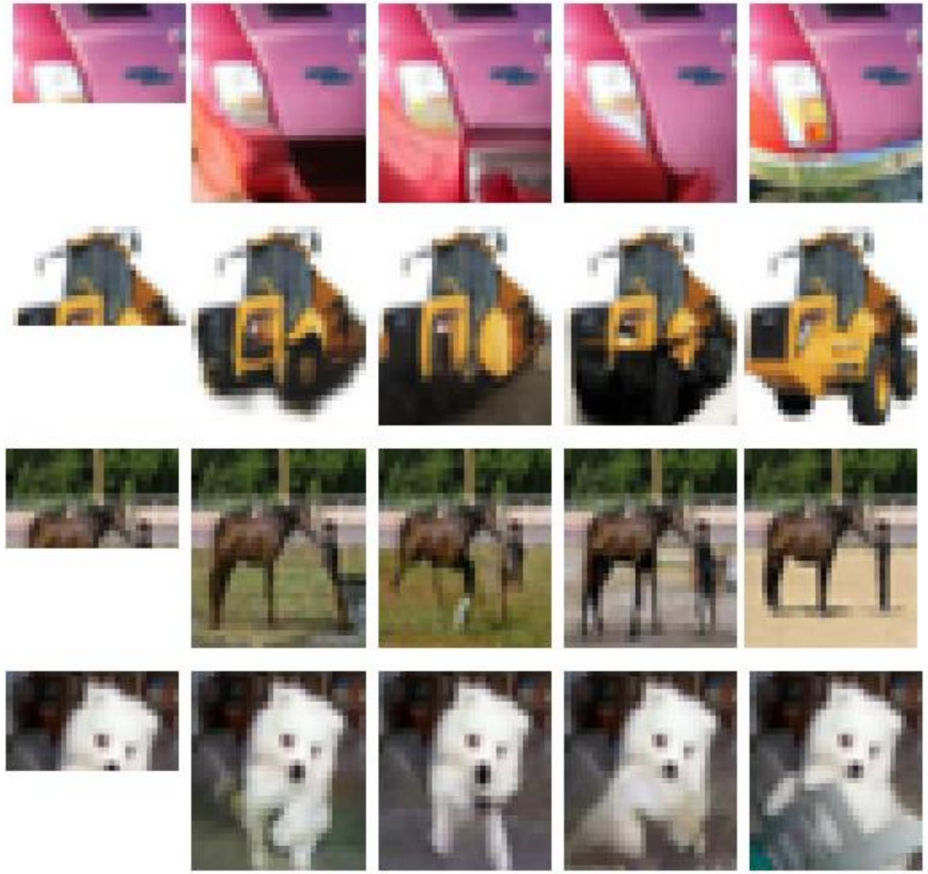
Image Generation
: Not as good as GANs..



Super Resolution : 꽤 좋은 성과!

Usages and Results

Still more to go!

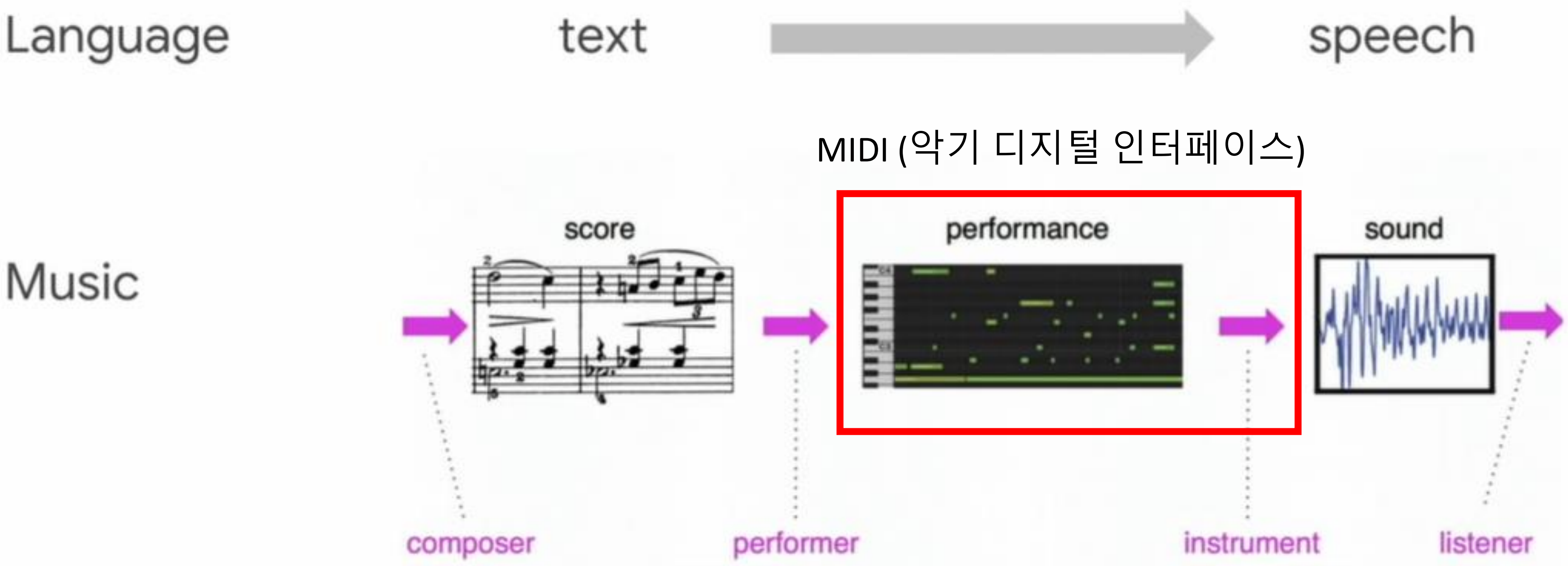


Conditional Image Completion

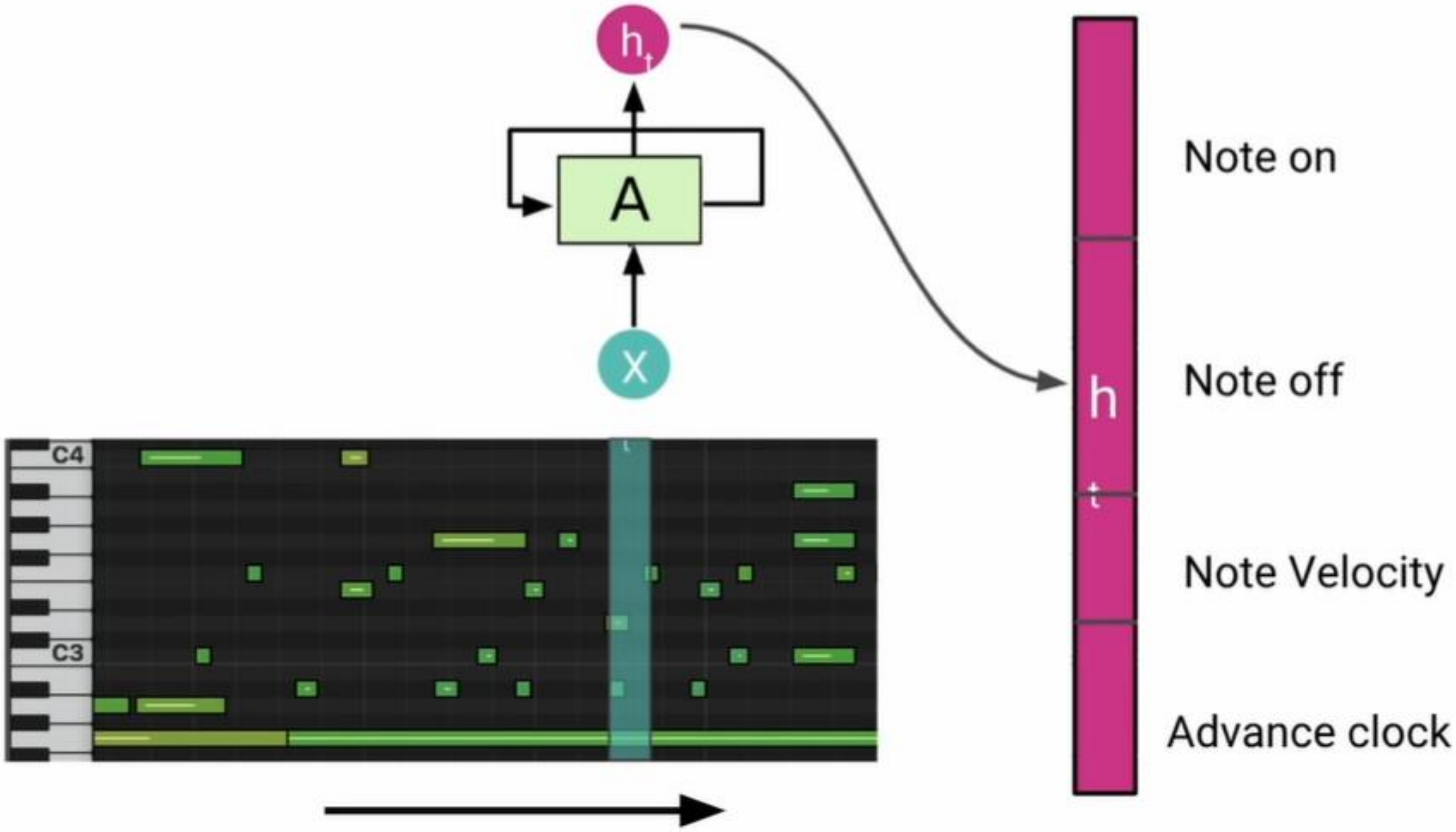
Super Resolution : 꽤 좋은 성과!

Music generation using relative self-attention

Raw representations in music and language



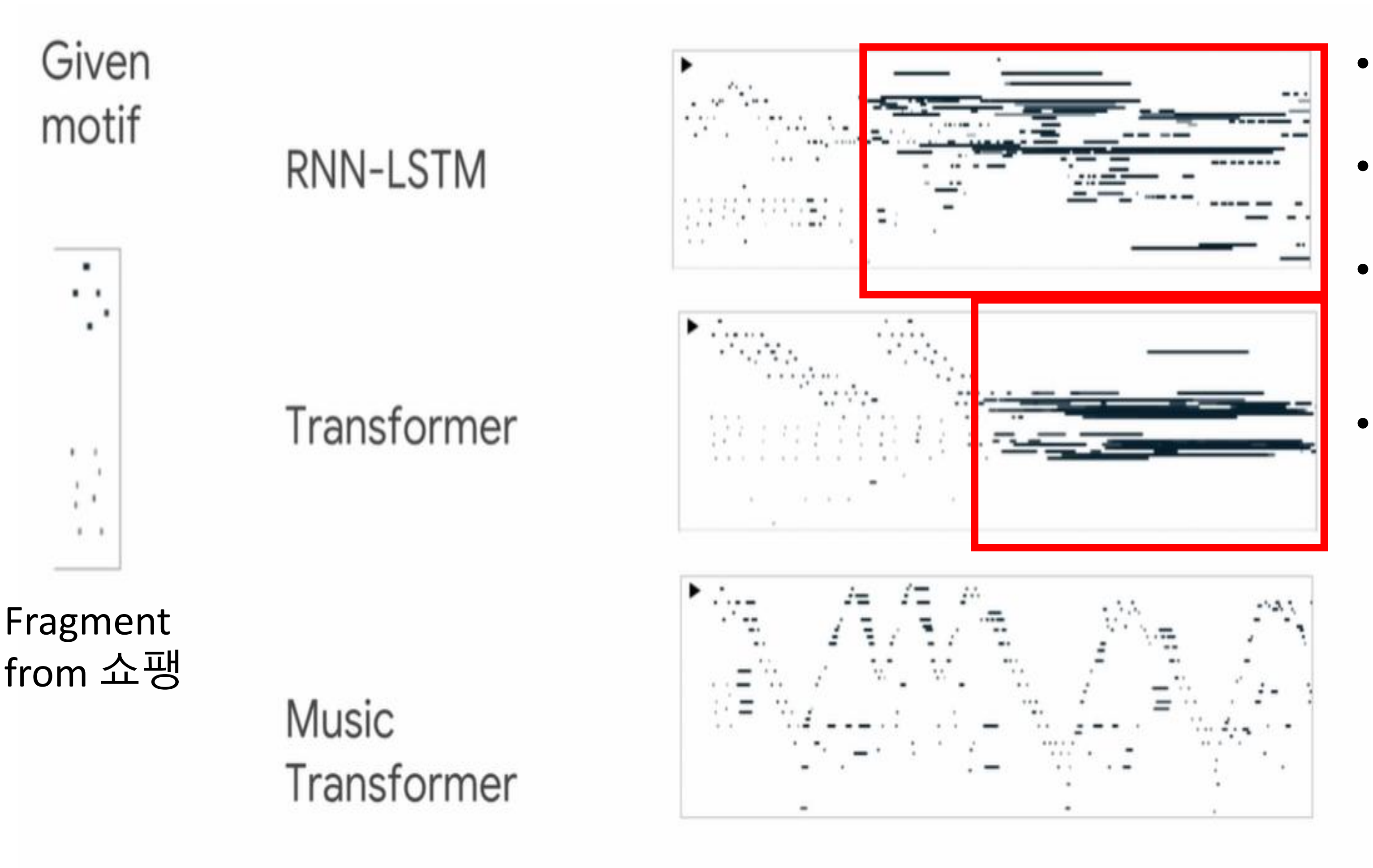
(Image from Simon & Oore, 2016)



Output:
Turn this note on
How loud
Turn this note on
Dynamics information
Etc.
…

RNN을 통한 연구가 이전에 많았지만 문제점이 많다!
-Long sequence가 fixed length Vector로 줄어드는 것
-Repetition이 있을 때 더욱 문제가 됨.

Motif를 주고 그 뒤는 model이 알아서 만들게 시킴



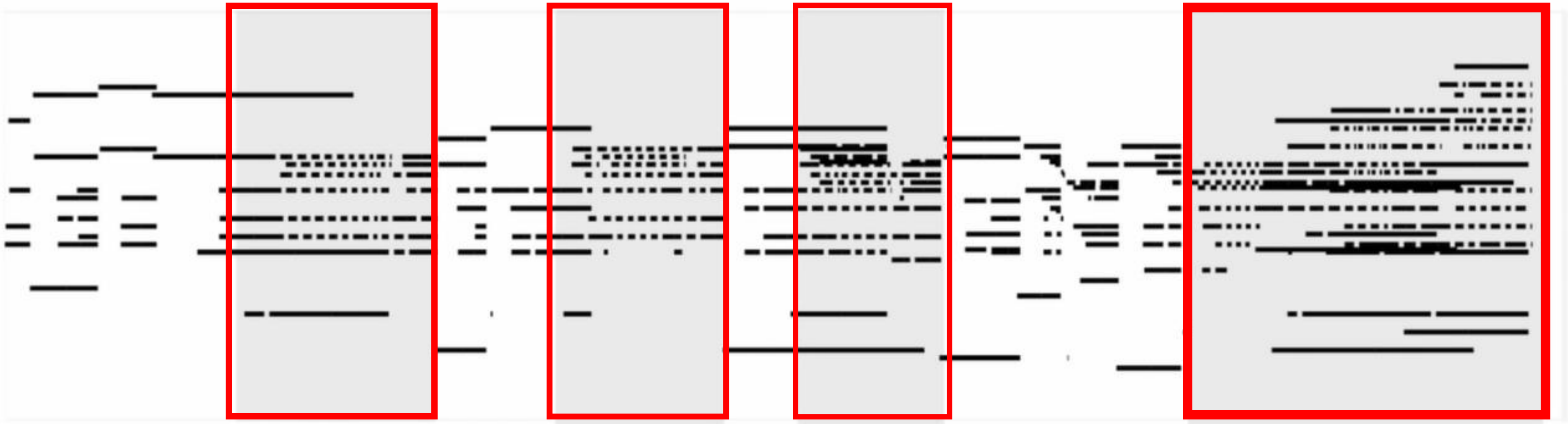
Fragment from 쇼팽

- 처음에는 given motif를 반복하려고 하지만 뒤로 갈 수록 이상해짐
- It is not able to directly look back to what happened in the past.
- Becomes more and more blurry

- 처음에는 좋은 성능을 보이지만 Input 보다 더 긴 sequence는 만들어내지 못함

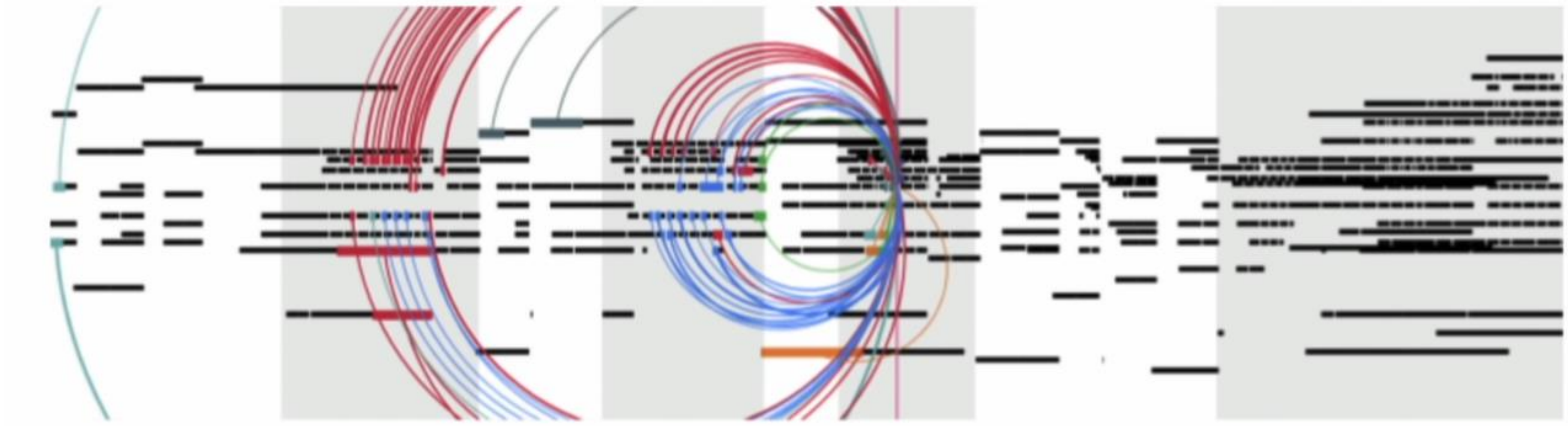
Unconditional sample from model

A lot of Replications with gaps on between



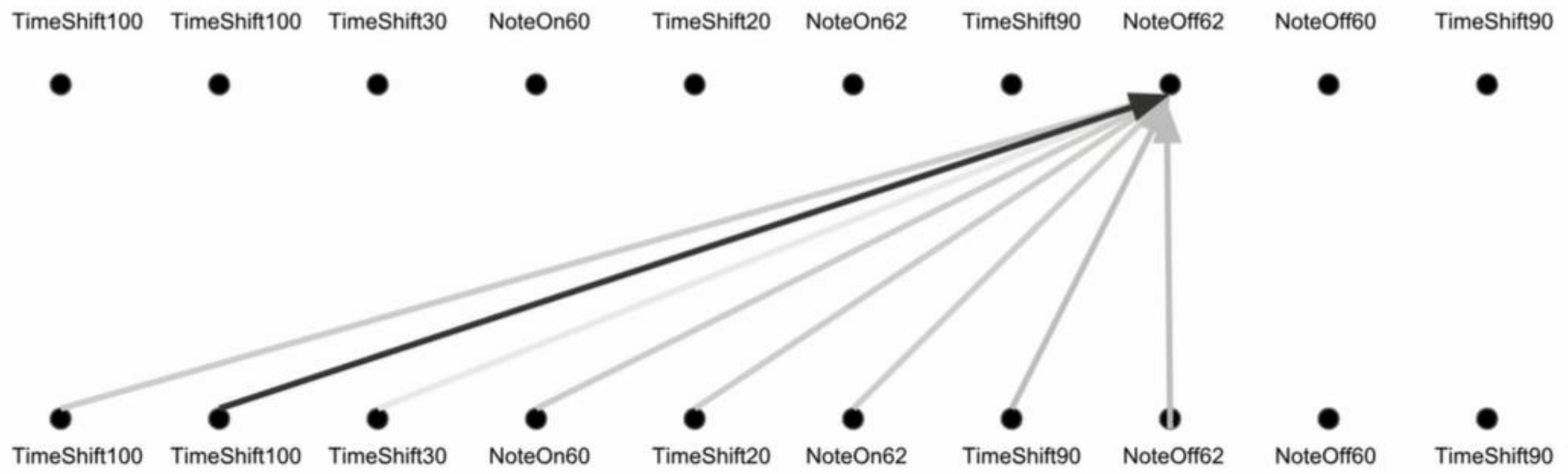
음영: motif

Self Attention Structure



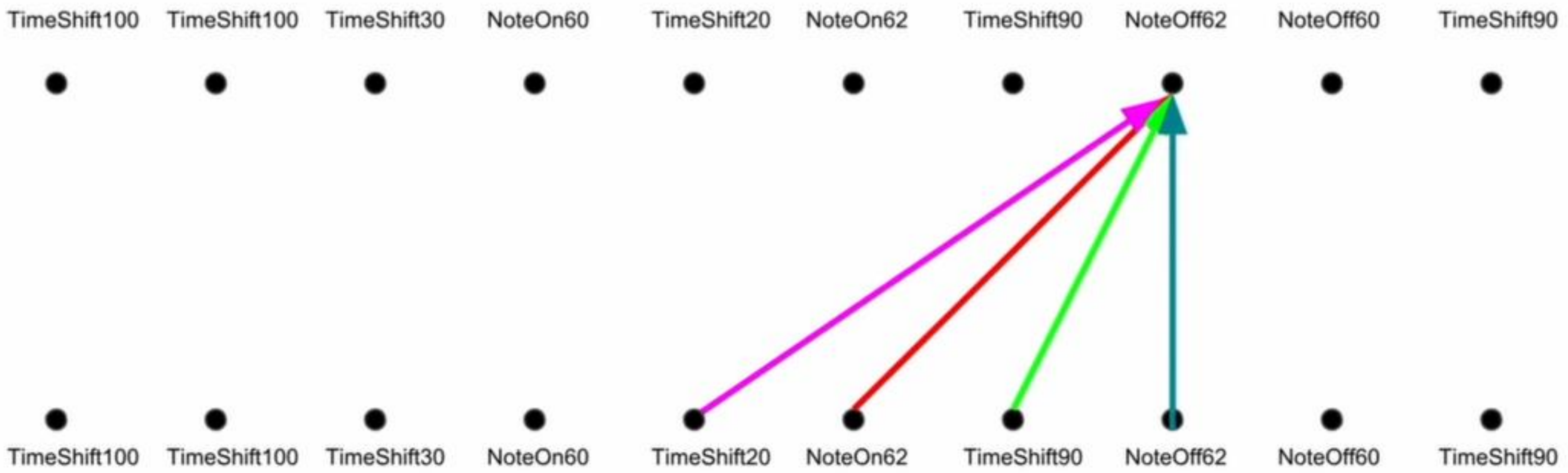
- Looking at the relevant parts even if it was not immediately
- 다른 색들 → different attention head, 각각 motif 부분에서 다른 부분에 집중
- How?

Attention: a weighted average



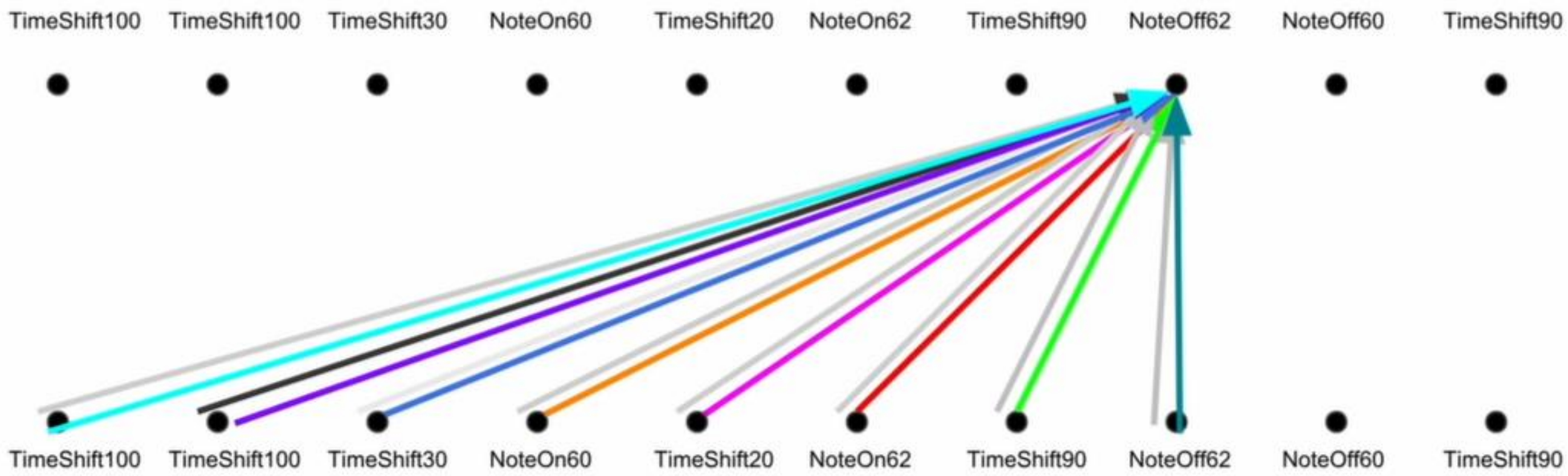
- Regular attention으로 시작
- 아무리 멀리 떨어져 있어도 Direct access 가능
- 작품 초반에 motif가 발생하더라도, 비슷한 것이 있었다는 것을 기억하고 반영할 수 있다.
- 하지만 과거 데이터들이 위치를 기억할 수 없다.
- **positional sinusoids 사용**

Convolution: Different linear transformations by relative position



- Fixed filter that's moving around that captures the relative distance.
- Bring in the distance information very explicitly.

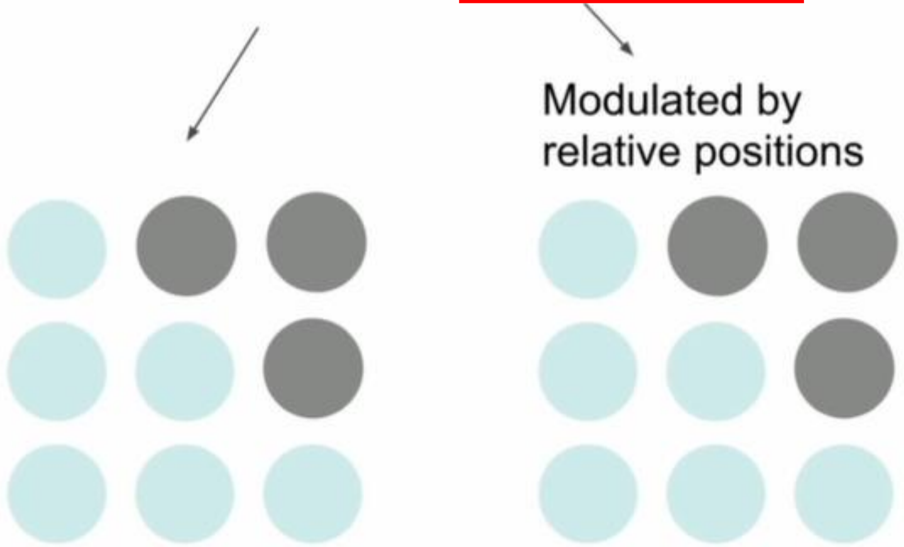
Relative attention (Shaw et al, 2018)
Multihead attention + convolution?



- Can access the history very directly.
- Also know how relate to this history.
- Relative attention을 통해 Music transformer는 train할 때 주어진 데이터 보다 긴 데이터를 학습할 수 있음.

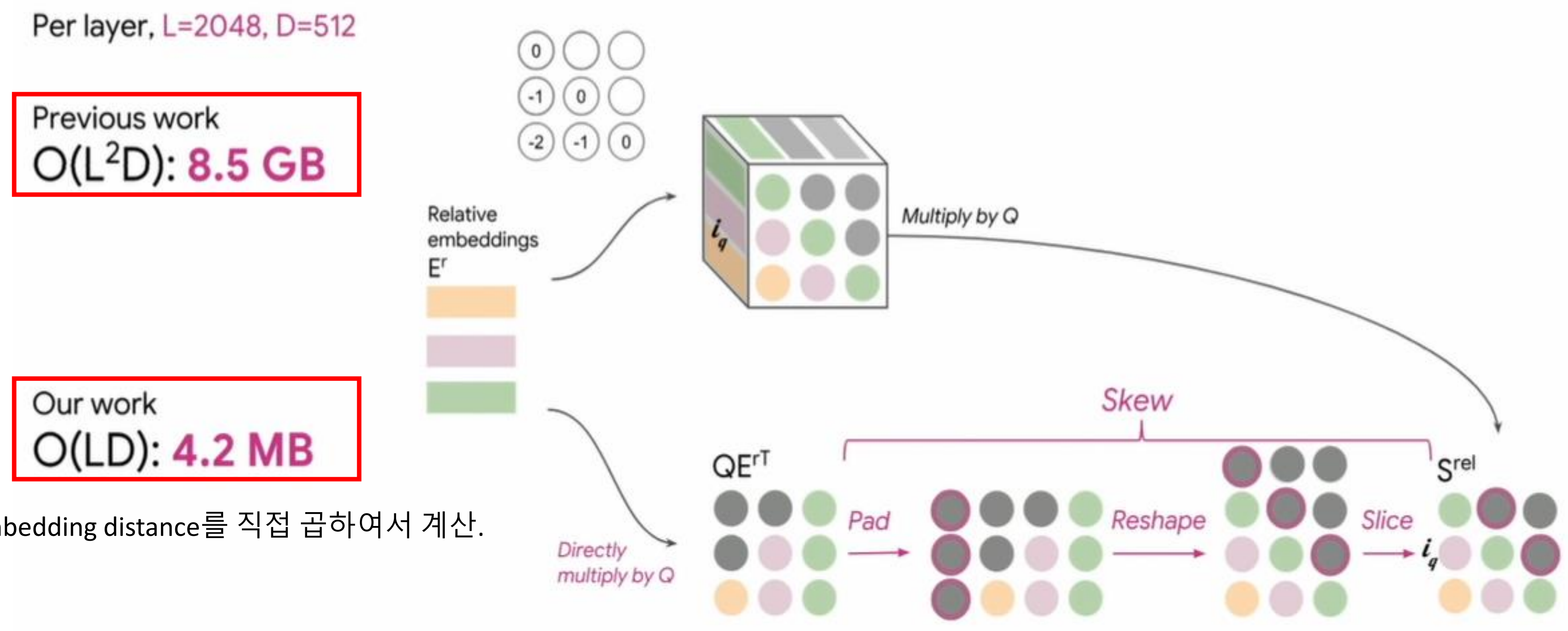
Closer look at relative attention

$$softmax(QK^\top + \boxed{Qf(E_{rel})})$$



- 기존 transformer는 모든 query와 key들을 비교해서 square matrix 형성
  → self similarity
- Relative attention은 얼마나 떨어져 있는지도 고려
  → similarity between positions
- 기존 attention score에 relative positional attention score 추가
- Gather the embedding that's irrelevant to the query key distance on the logits.

기존: 3D tensor를 만드는 것이라 메모리를 차지하는 비중과 연산량이 굉장히 높음

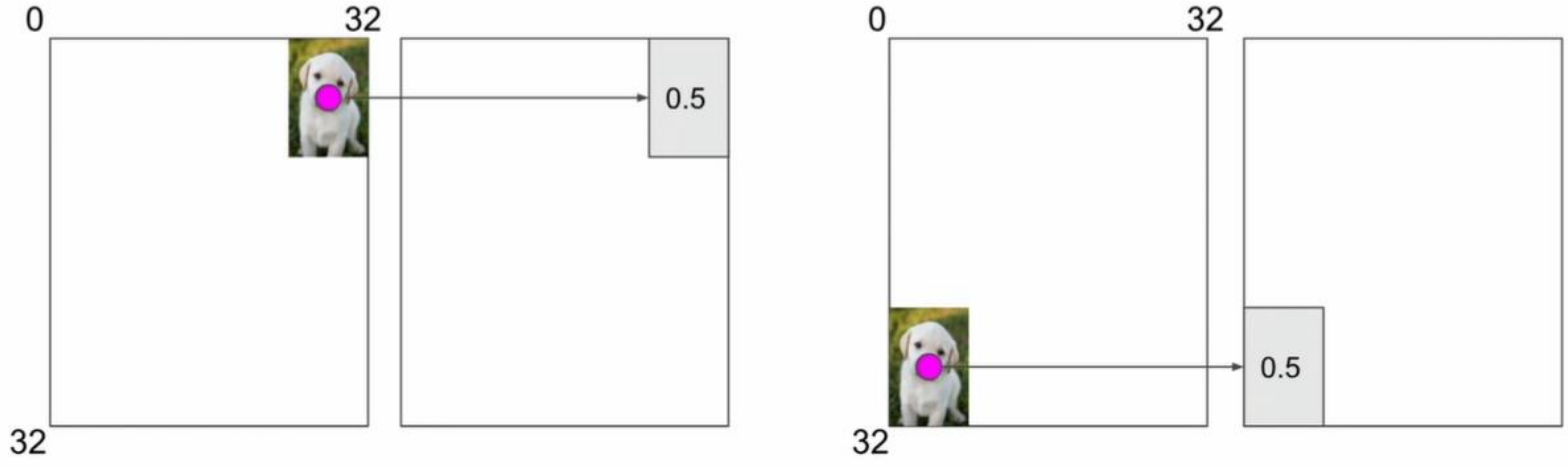


Our work: query들과 embedding distance를 직접 곱하여서 계산.

- 기존 relative attention은 모든 페어를 고려해서 relative distance를 구하기 때문에 L X L X D 만큼의 연산이 필요함

- Our work에서는 연산량은 줄어들었지만, 결과값의 순서가 query ordered by a relative distance이기 때문에 이를 query ordered by keys로 변환하는 과정이 필요함 → Do a series of skewing!

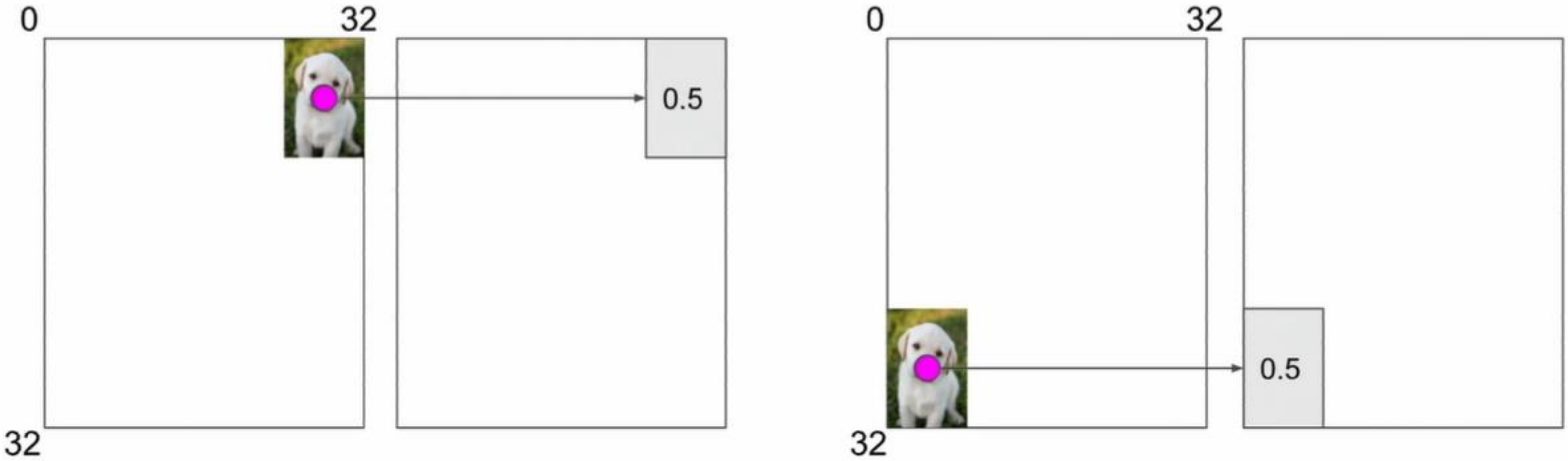
Convolutions and Translational Equivariance



- Translation equivariance: input의 위치가 변하면 출력도 동일하게 위치가 변한채로 나온다.
- 빨간 점 혹은 강아지에 대한 계산을 할 때 그림에서 어느 위치에 있는 지에 영향을 받지 않는다.
- In the large image, it just doesn't depend on its absolute location. It's going to produces the same activation.
- Convolution 연산을 하면 translation equivariance 특성과 더불어 파라미터를 공유하기 때문에 필터 하나로 다양한 위치에서 다양한 특징들을 추출할 수 있다.

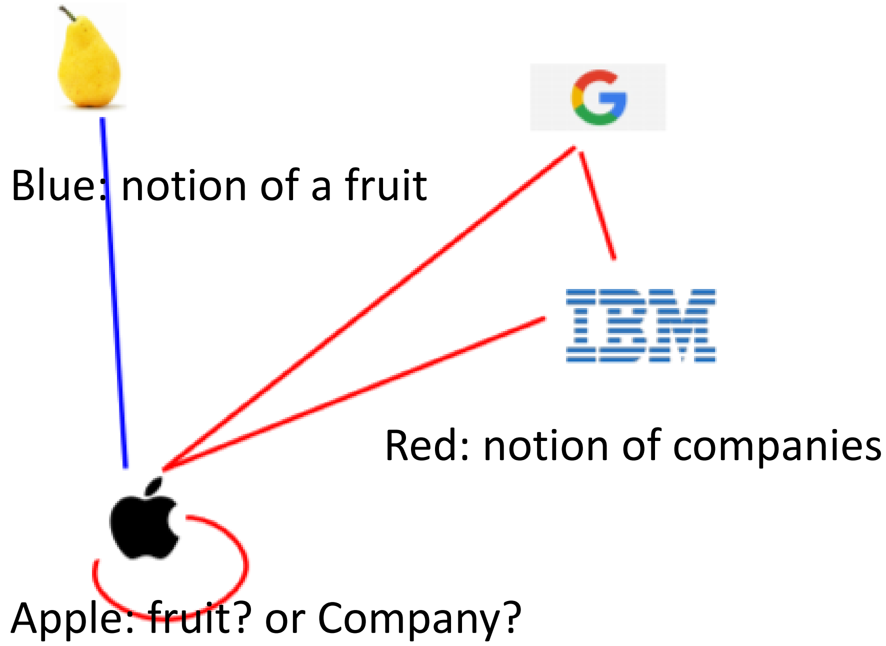
https://medium.com/@seoilgun/cnn%EC%9D%98-stationarity%EC%99%80-locality-610166700979

Relative positions Translational Equivariance

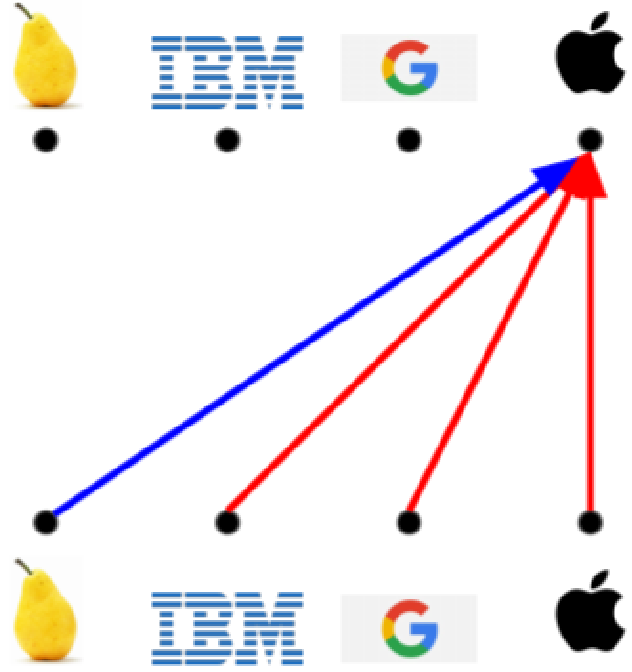


- Positions 혹은 relative attention도 비슷한 효과를 보인다.
- Absolute position에 대한 정보를 지워도 좋은 성능을 보였다 .
  → Translation equivariance which is great property for images.
- Semis 같은 작업을 하는 경우 unlabeled data로 모델을 학습시킨 다음 위치를 변동시키기 때문에 Generative models of images가 유용하다.

Relative Attention And Graphs



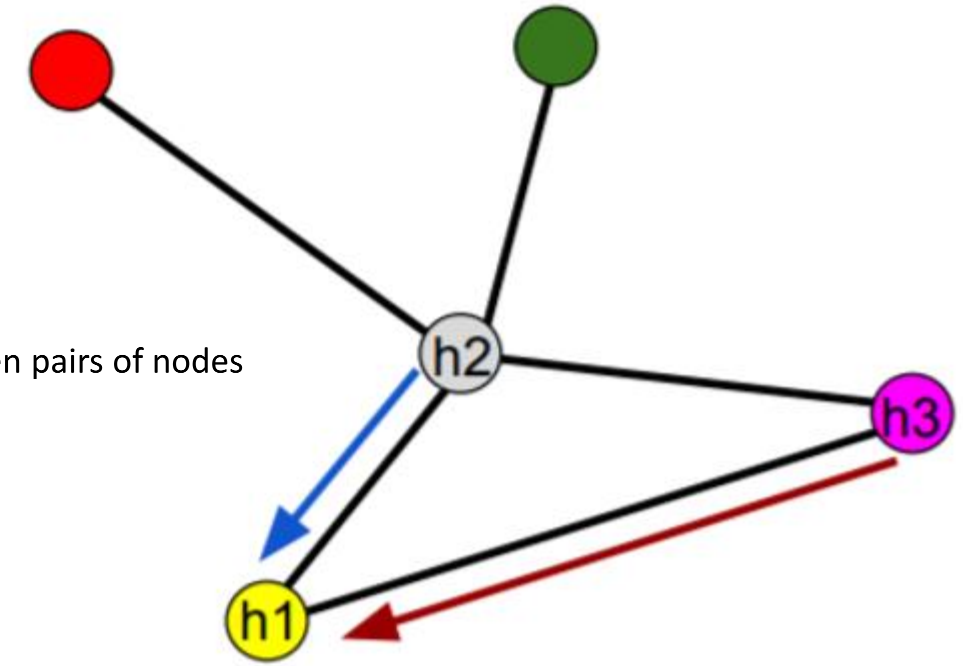
- Relative attention으로 다양한 뜻을 가진 단어에 서로 다른 similarity를 부여할 수 있음.



If you have graph problems then relative self attention might be a good fit for you.

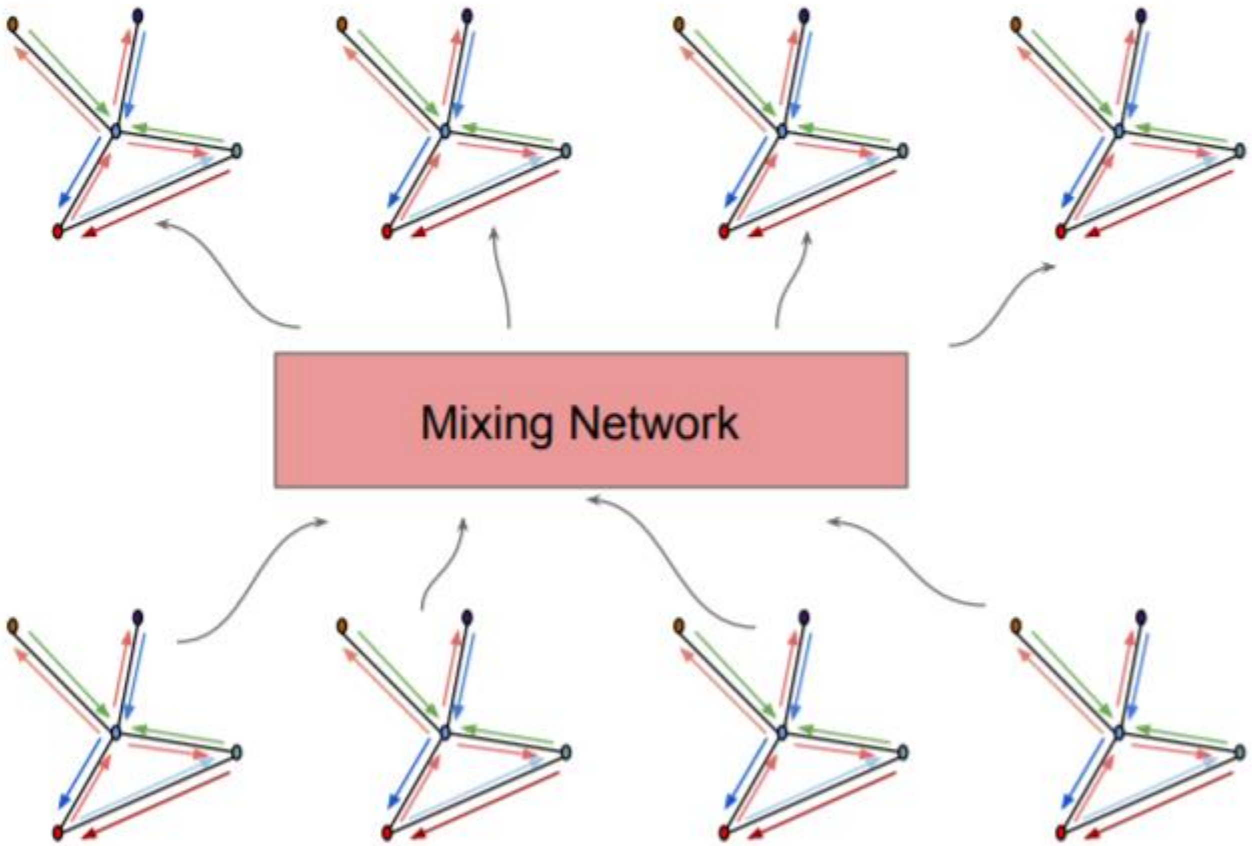
Message Passing Neural Networks

Passing messages between pairs of nodes



- Self attention imposes a fully connect, then now message passing neural networks did exactly that they were passing messages between nodes as well.
- 기존 message passing은 연결되어 있는 node들만 고려했지만, self attention을 사용한다면 softmax 함수를 사용하기에 모든 node들 사이의 interaction을 고려할 수 있다.

Multiple Towers



MPNN: Message Passing Neural Network

- Run k smaller copies of the MPNN in parallel.
- Mix node states after each message pass.
- Offers a factor of k speedup for the same node dimension d (> 2x speedup when d=200).
- Also helped improve performance when used with matrix multiply message function.

Self Attention

- Constant ‘path length’ between any two positions.
- Unbounded memory.
- Trivial to parallelize (per layer).
- Models Self-Similarity.
- Relative attention provides expressing timing, equivariance, and extends naturally to graphs

감사합니다.