# Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting

## 시계열 예측 모델 최신 경향 분석 & 코드 리뷰 (1)

2021.03.23 백지윤

# Index

# 1-(1). attention 매커니즘

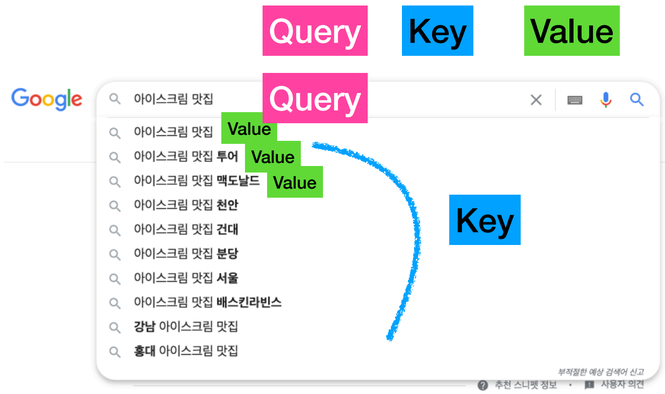
# attention 매커니즘 소개

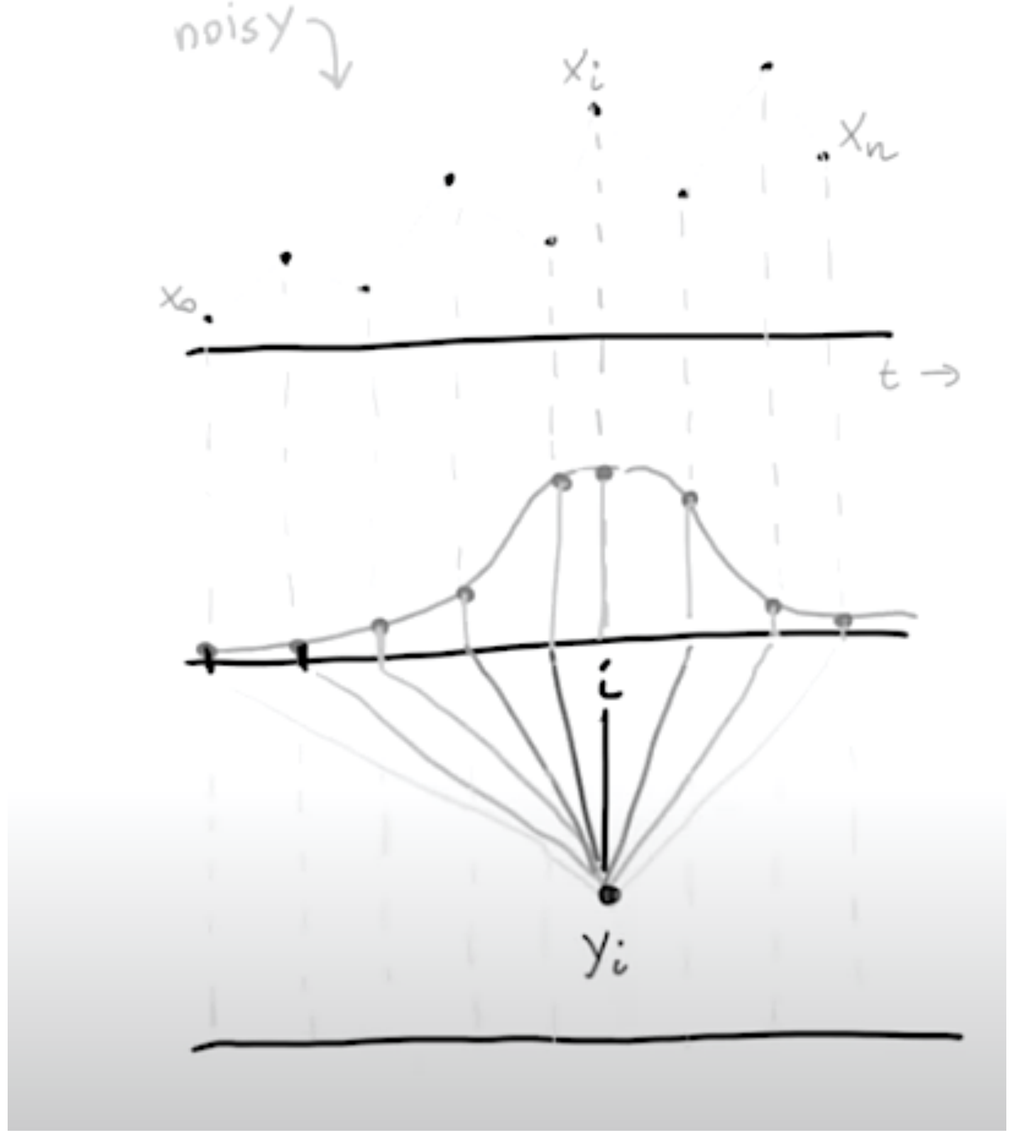
- 취지 : auto-regressive 특징을 갖는 데이터에서 특정 시점(t) 의 output 값을 예측할 때 이전 시점 들을 얼마나 고려할지 (각 time step 의 중요도에 따른) 각기 다른 weights 를 구해주자 !

- Ex) I love cats -> 나는 ? ?
  0.8 0.1 0.1

- 초기 어텐션 : timesteps 0~3까지 있다고 하면
  Y[i]=softmax(X[i] · X[0]) * X[0] +softmax(X[i] · X[1])*X[1] + softmax(X[i] ·X[2])*X[2]

- softmax(X[i] · X[0]) * X[0]



# attention 매커니즘 소개

## Self-attention

$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

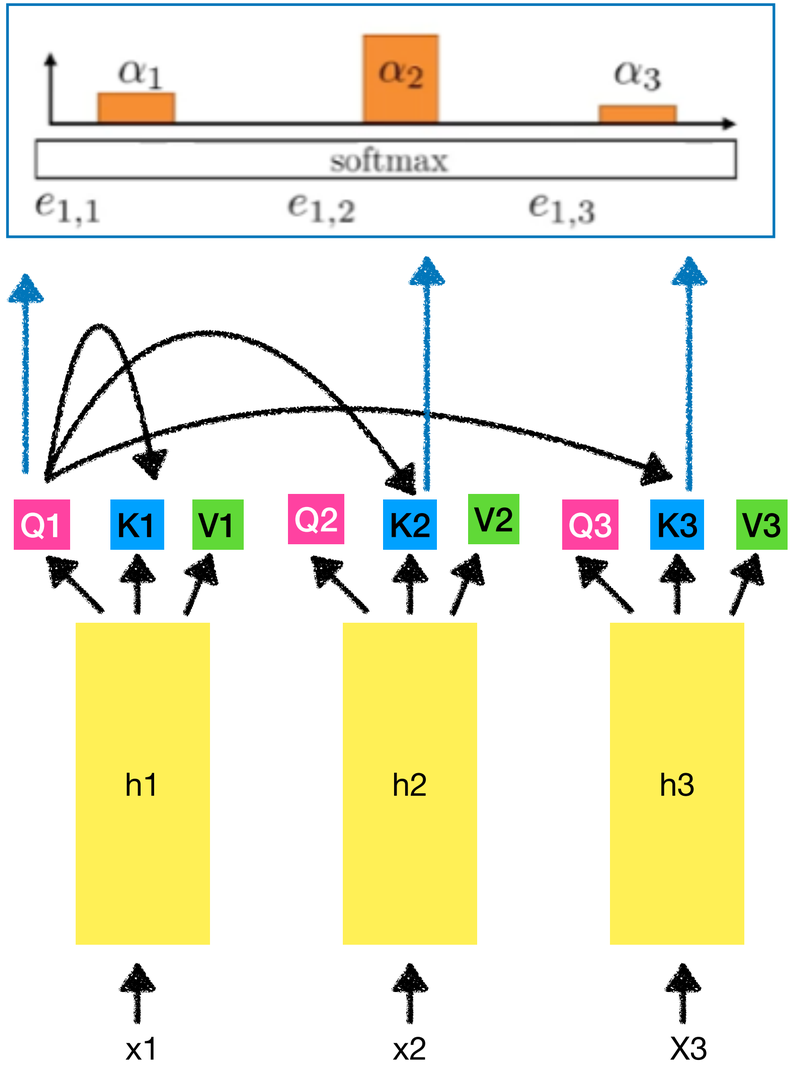
we'll see why this is important soon

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

This is not a recurrent model !
But still weight sharing :
h[t] = sigmoid(Wx[t] +b)



# attention 매커니즘 소개

## Self-attention

$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

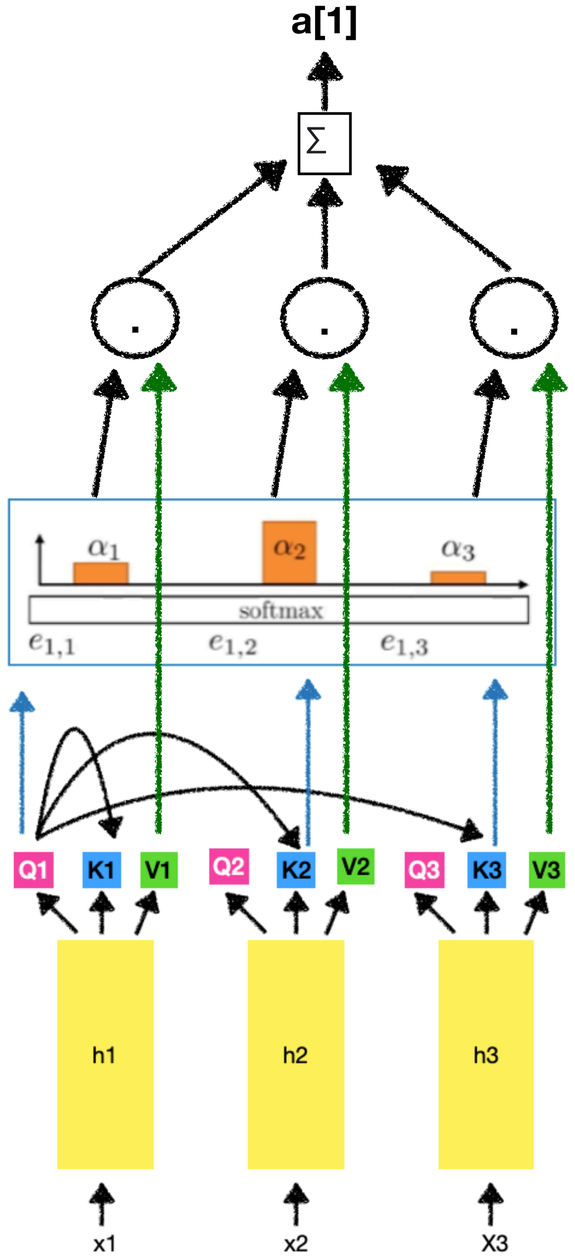
we'll see why this is important soon

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

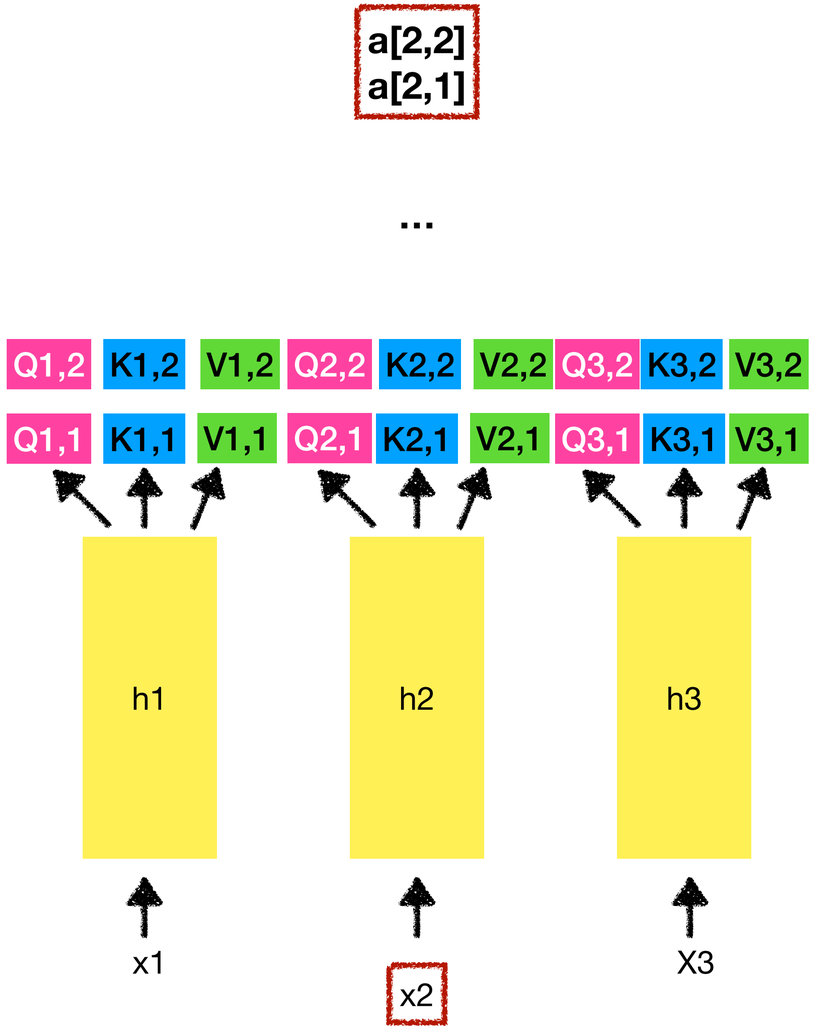
This is not a recurrent model !
But still weight sharing :
h[t] = sigmoid(Wx[t] +b)



# attention 매커니즘 소개

## Multi-head attention

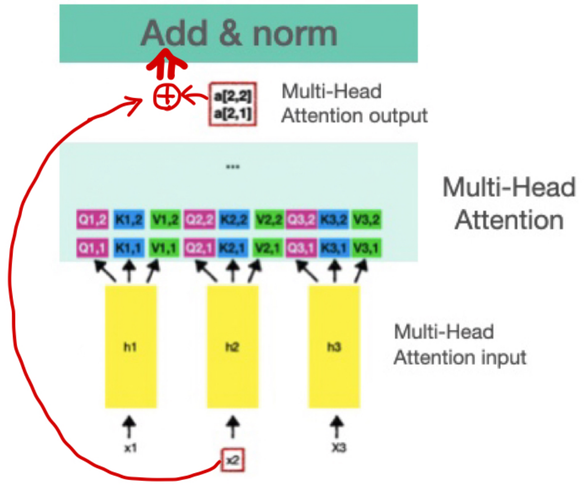
- 일반적으로 8개 정도의 head 사용
- 각 head 에서 나온 attention 결과를 concat
- Multi – heads 개념은 cnn 의 필터 개념과 흡사



# 1-(2). 부가적인 개념

# 부가적인 개념

**Layer normalization (add & norm)**
**Positional encoding**



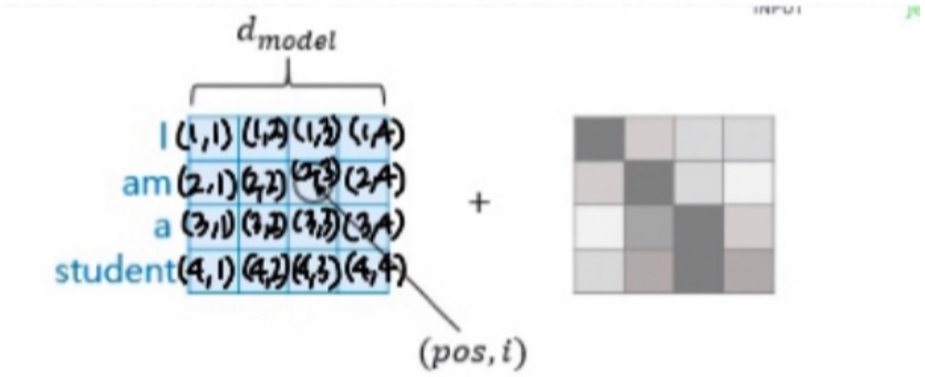

- add&norm 은 레즈넷의 개념과 유사
- 어텐션 메커니즘은 rnn 과 달리 위치 기반이 아니기 때문에

  Positional encoding 도입
- Pos 는 전체 시퀀스에서 몇 번째 단어인지를 의미
- i 는 임베딩 벡터 내의 인덱스
- Positional encoding 값 = sin(pos/10000^(2k/임베딩 차원))

  (i=2k, 즉 i 가 짝수일 때)

  Positional encoding 값 = cos(pos/10000^(2k/임베딩 차원))

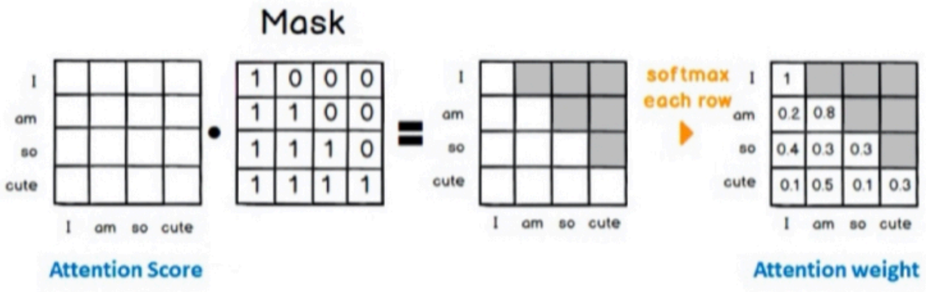
  (i=2k+1,즉 i 가 홀수일 때)



# 부가적인 개념

## Masked multi-head attention

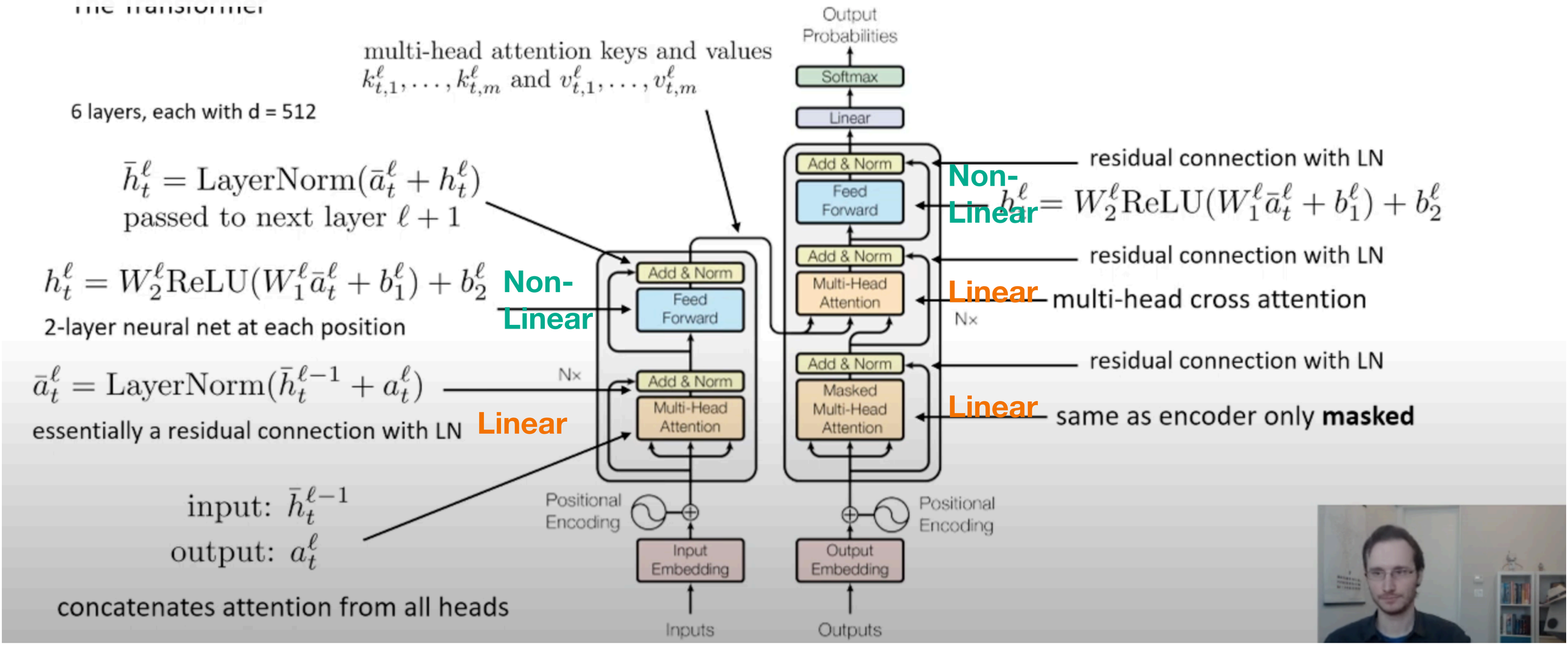
- Attention 은 rnn 처럼 이전 시점의 인풋 값들로만 output 을 추측하는 것이 아닌 오픈북 테스트
- 이를 방지하기 위해 실질적으로 결과물을 뱉어야하는 디코더에서는 masked multi-head attention 개념 도입



# 1-(3). 트랜스포머 전체 구조 & 코드

# 부가적인 개념

## Masked multi-head attention



# “Why transformers >> rnn ? ”

**Much better long-range connections**
**Much easier to parallelize**
**Can make it much deeper than RNN**

**코드 : https://github.com/aladdinpersson/Machine-Learning-Collection/blob/master/ML/Pytorch/more_advanced/transformer_from_scratch/transformer_from_scratch.py**

출처 : https://www.youtube.com/playlist?list=PL_iWQOsE6TfVmKkQHucjPAoRtIJYt8a5A&fbclid=IwAR3mDMdJgK-3bvyO2P95drPcvQd1VonMQ8cYJ06efPKKWW6Ui9-XwYIO3PA