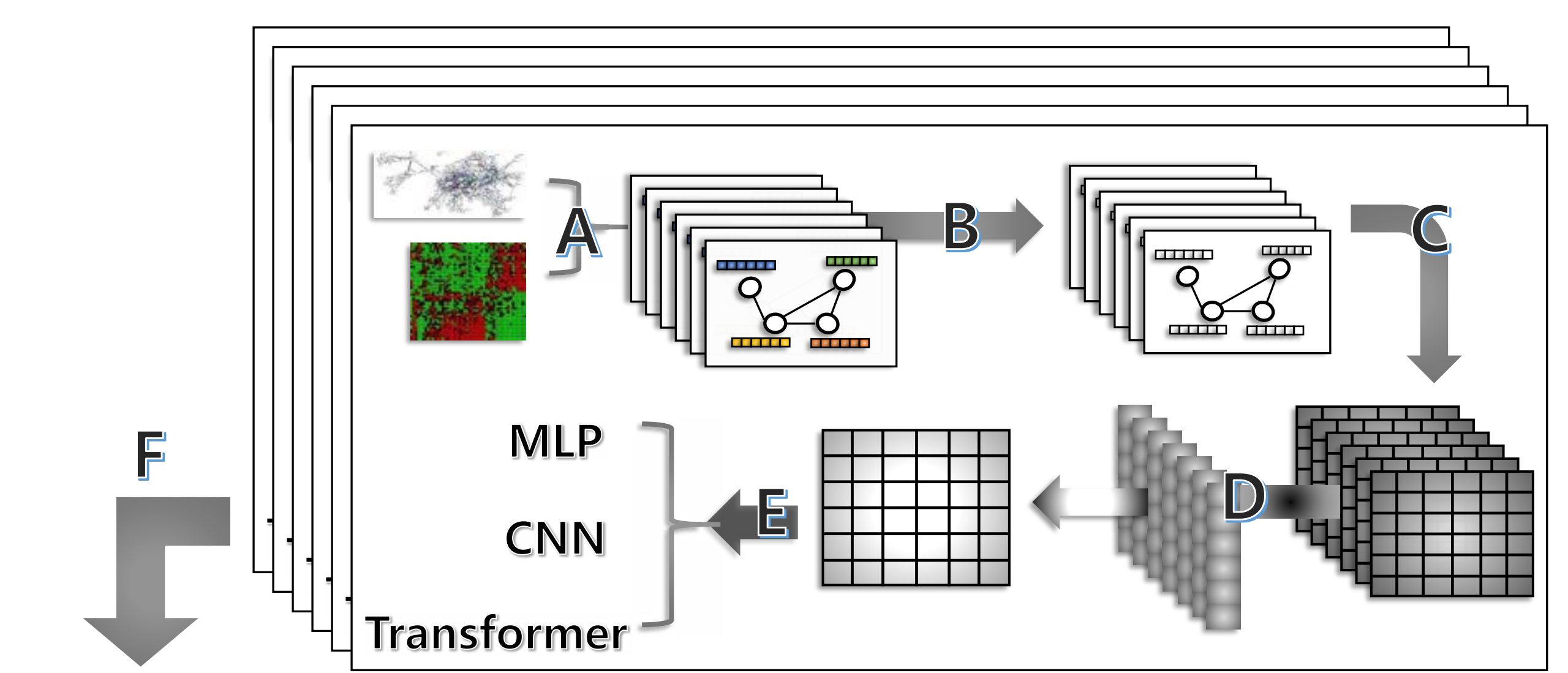
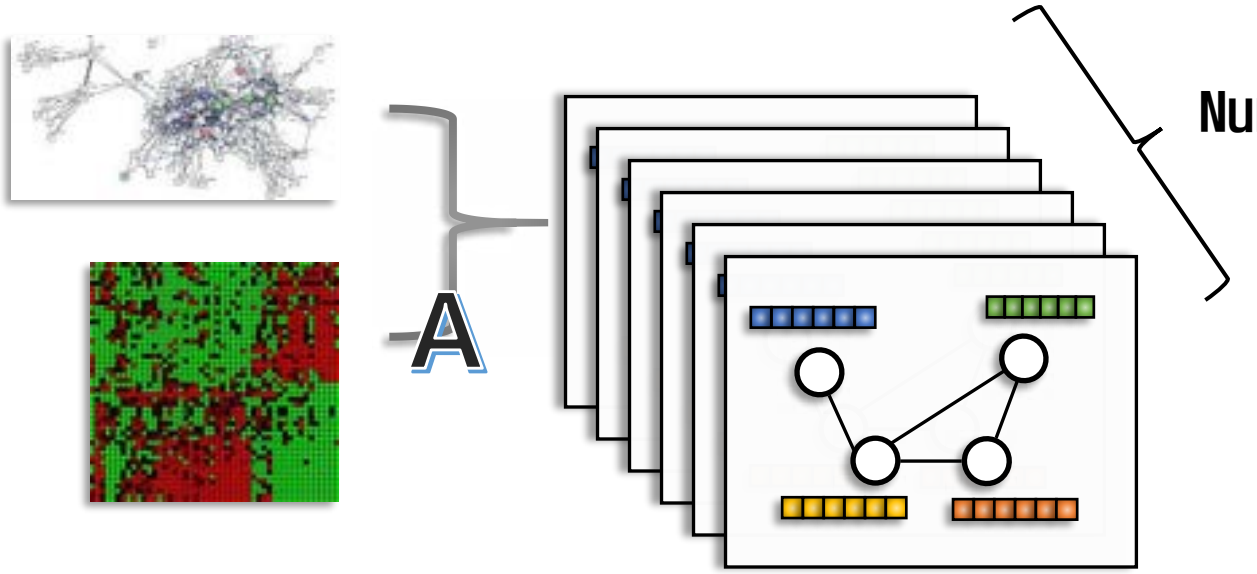
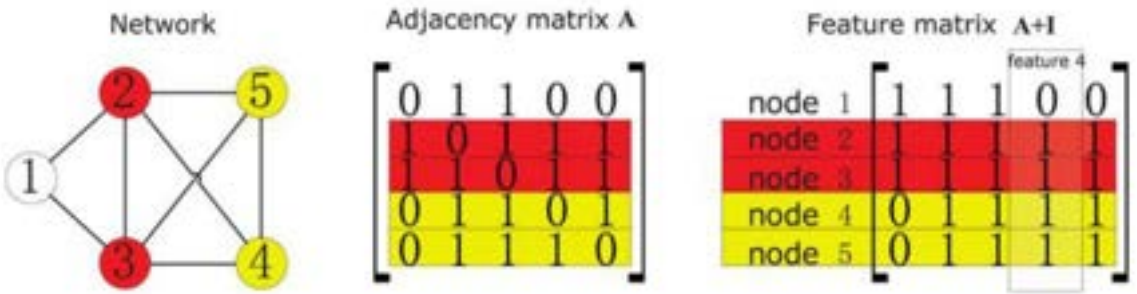


IC50 Prediction



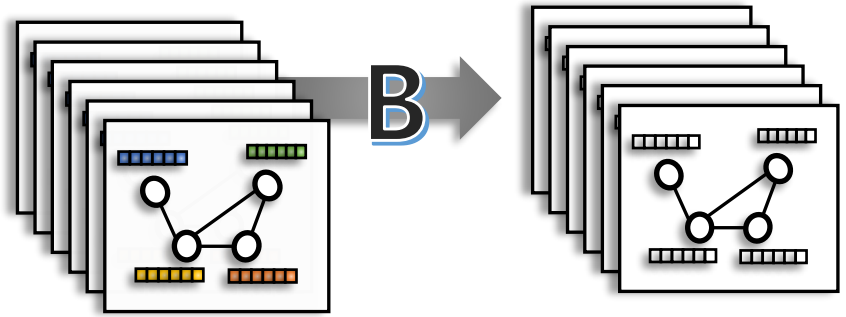


STEP A_1

- KEGG DB에서 Pathway Data 수집 (n = 308)
- GDSC DB에서 Cancer에 대한 gene expression data 수집 (n = 6308)

STEP A_2

- 수집한 Pathway Data를 통해 Pathway별로 Adjacency Matrix 생성
- Pathway에 속해 있는 gene들의 expression을 확인하여 각 vertex(gene)의 Feature로 하는 Node Feature Matrix 생성



STEP B_1

- Graph Convolution 연산 수행

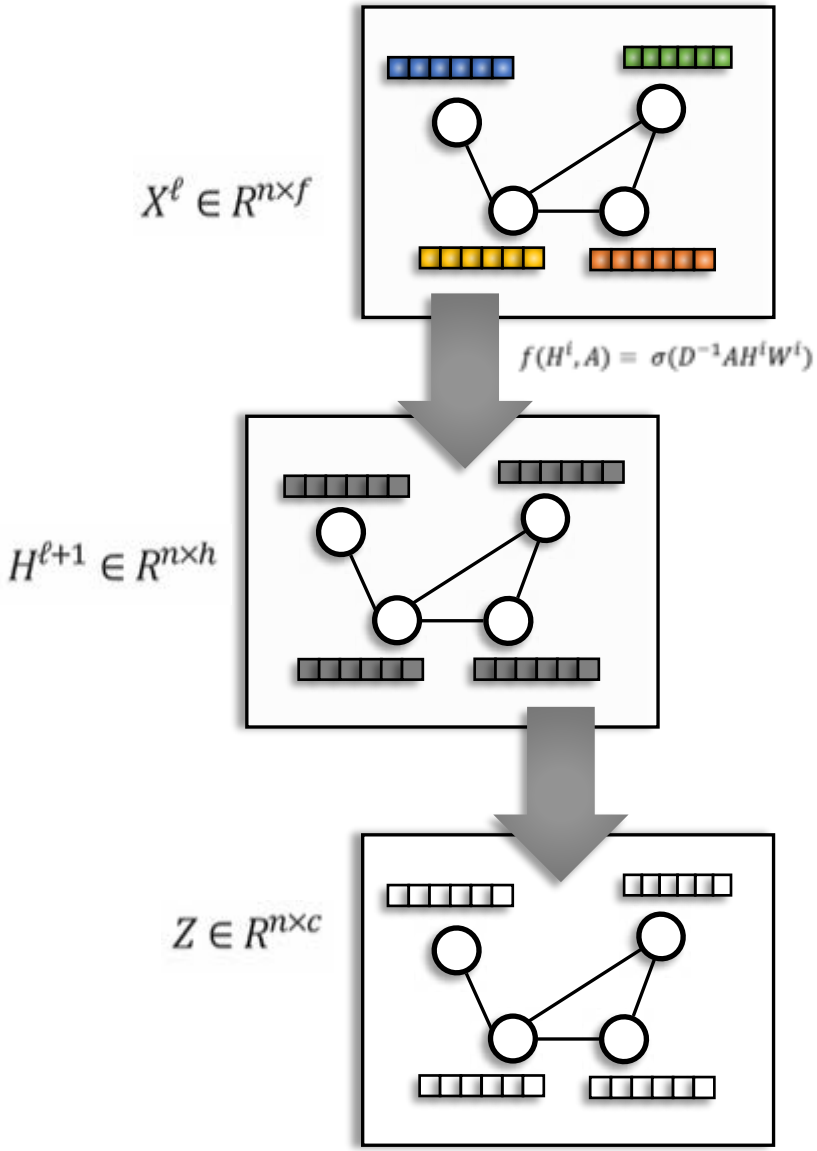
$$H^{\ell+1} = \sigma(\hat{A}X^{\ell}W^{\ell} + b^{\ell})$$

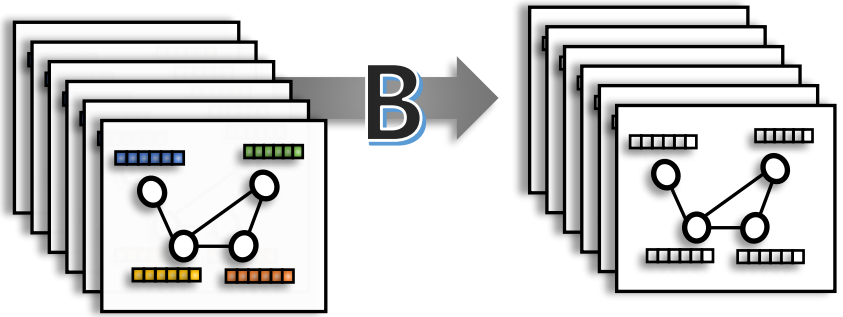
$$\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

$$\hat{A} = \tilde{D}^{-1/2}(I + A)\tilde{D}^{-1/2}$$

$X \in R^{n \times f}$

$A \in R^{n \times n}$





STEP B_2

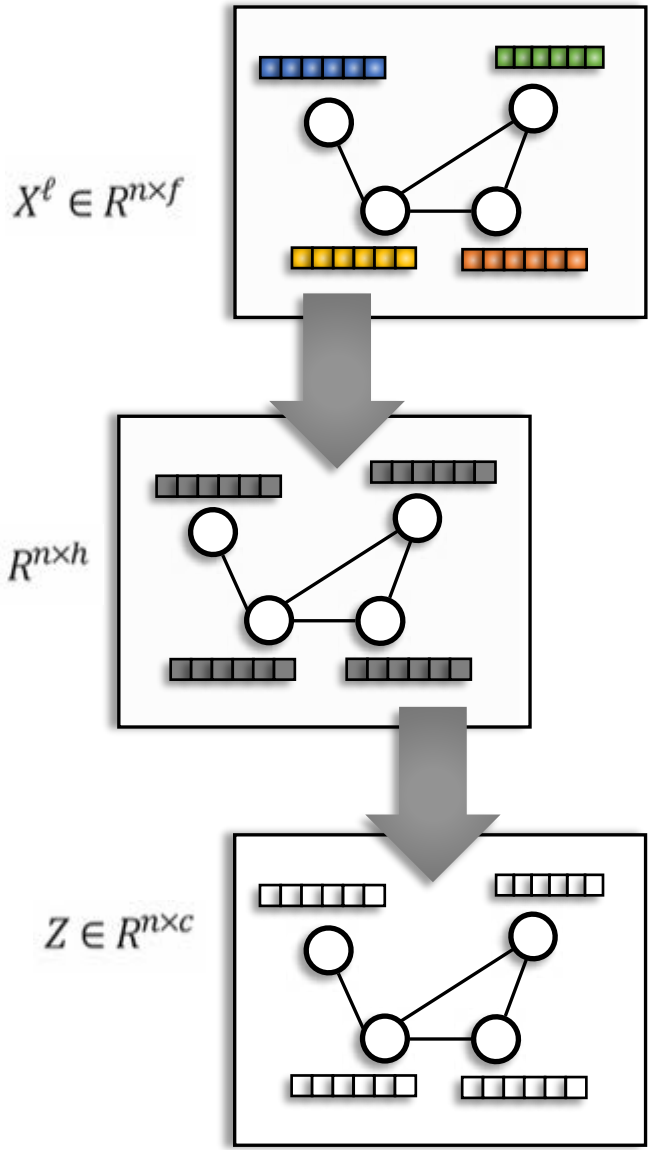
- Layer를 통과하면서 Skip Connection 함수 추가

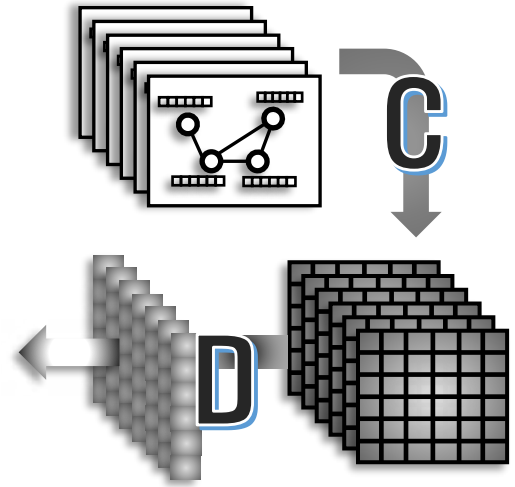
$$H_{i,\,sc} = X_i + H_i$$

$$H_{i,\,sc} = z_i \odot X_i + (1 - z_i) \odot H_i$$

- Node 간 관련성까지 학습할 수 있도록 Attention 함수 추가

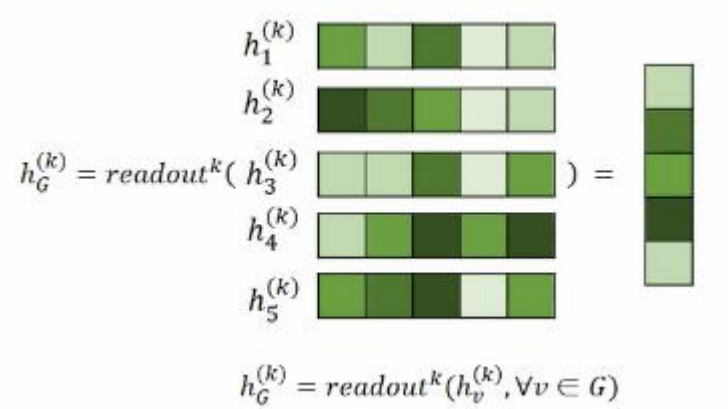
$$\alpha_{ij} = \frac{exp(LeakyRELU(\vec{a}^T[\mathbf{W}\vec{h}_i|\mathbf{W}\vec{h}_j]))}{\Sigma_{k\in N_i} exp(LeakyRELU(\vec{a}^T[\mathbf{W}\vec{h}_i|\mathbf{W}\vec{h}_k]))} \longrightarrow \vec{h}_i' = \sigma(\Sigma_{j\in N_i}\alpha_{ij}\mathbf{W}\vec{h}_j)$$

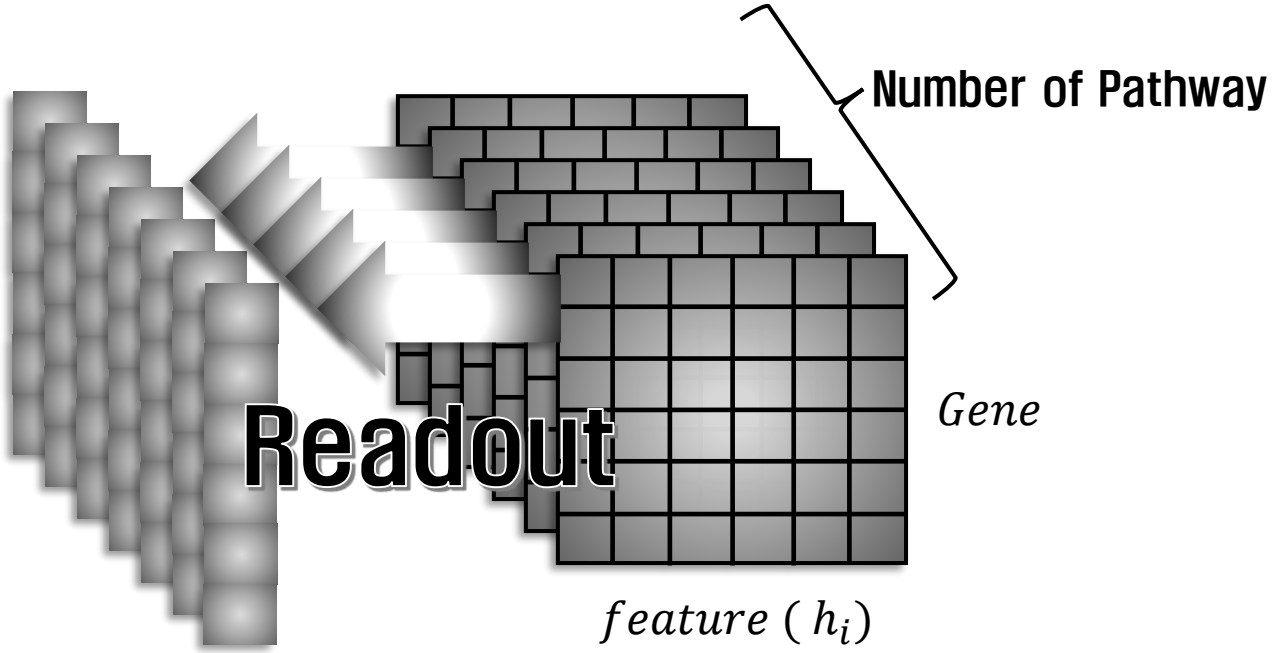




STEP D_1

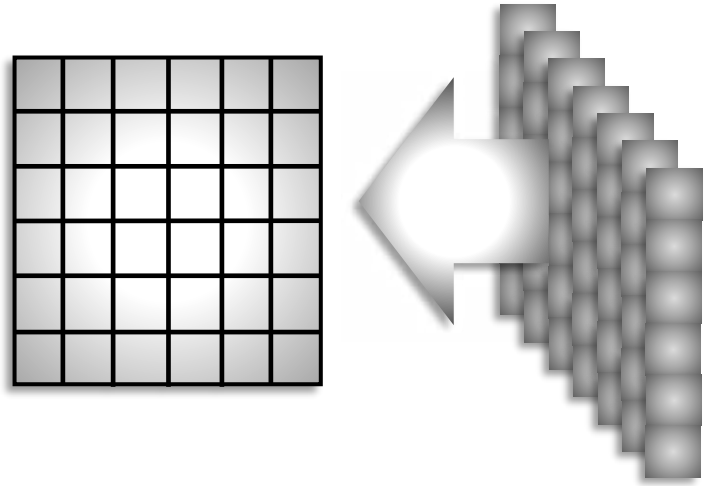
- Readout Layer : Permutation Invariance

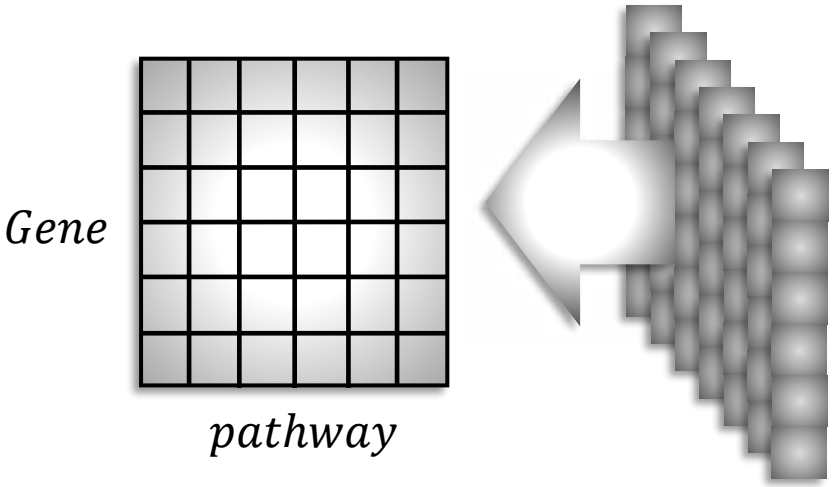




Node-wise summation

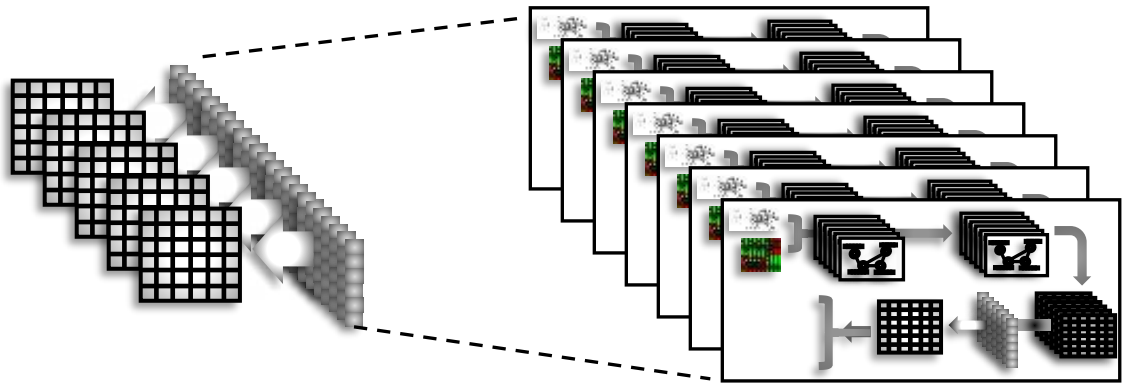
$$z_G = \tau\left(\sum_{i \in G} MLP\left(H_i^{(L)}\right)\right)$$

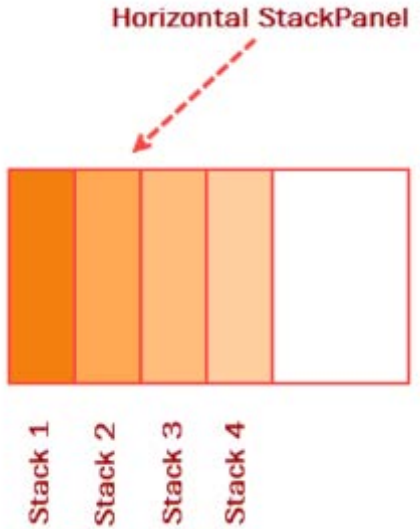


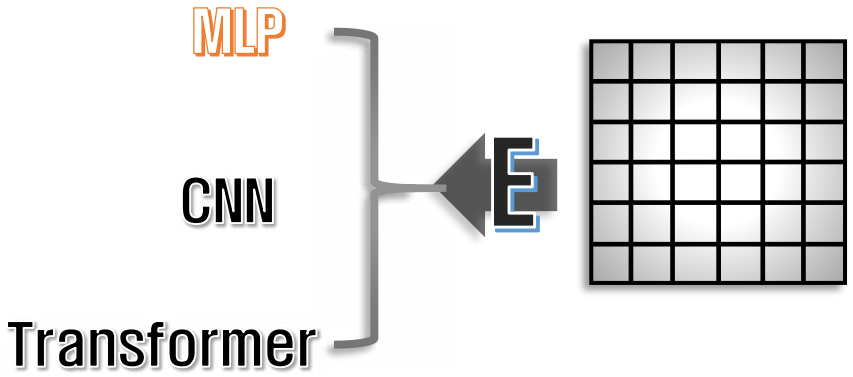


STEP D_2

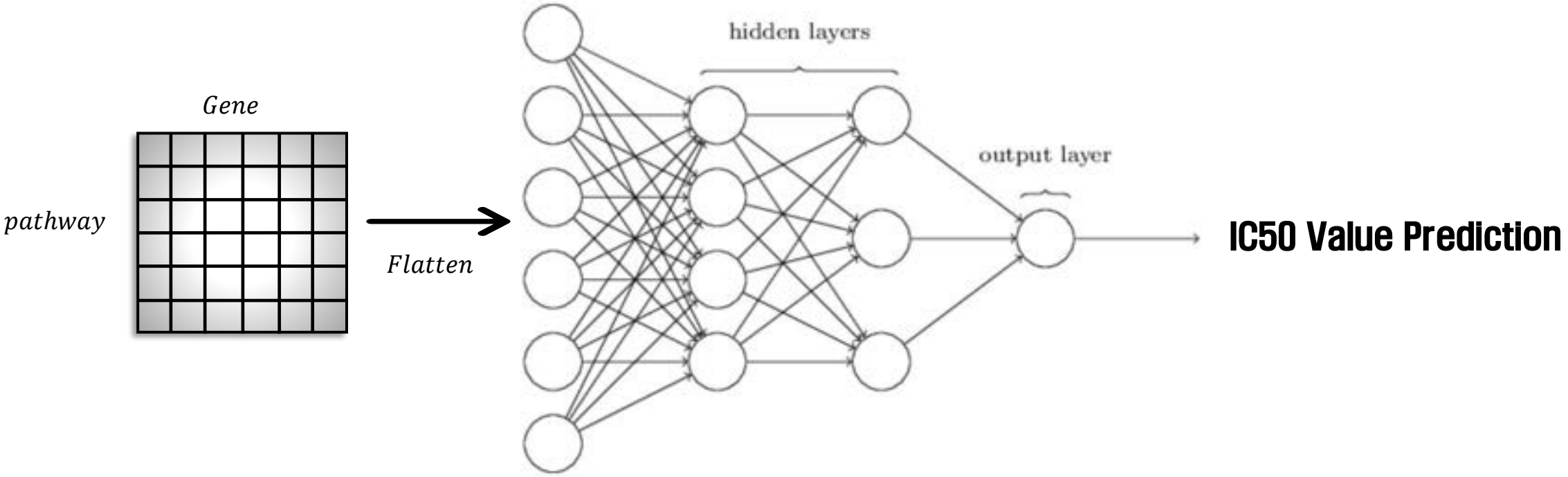
- Pathway별로 나온 Readout Layer를 통해 나온 벡터를 Horizontal Stack하여 하나의 Matrix로 변환
- 위 과정을 통해 나온 하나의 Matrix는
  IC50 값을 예측하는 딥러닝 모델의 Batch가 1인 Input Tensor Data
- Step A ~ D의 과정을 반복하는 횟수만큼 Batch size가 결정됨

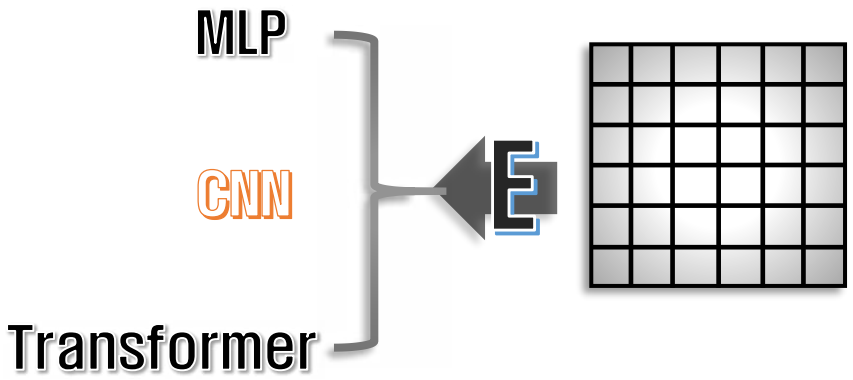


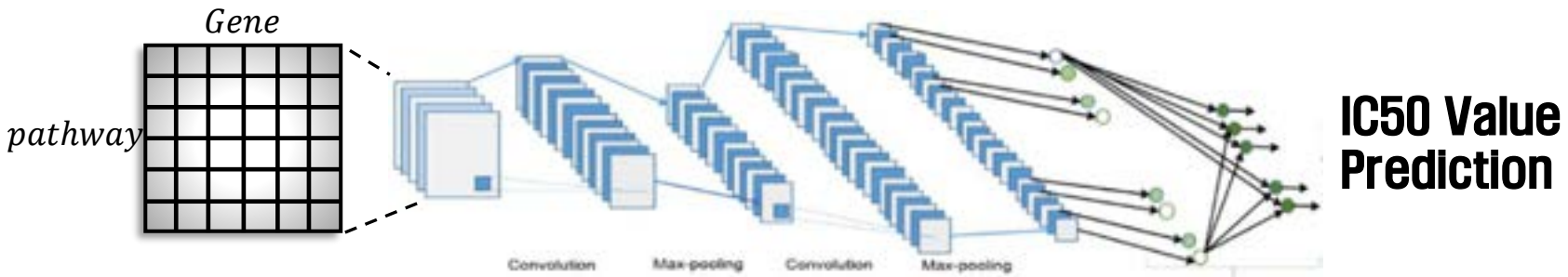



MLP
CNN
Transformer
E

STEP E_MLP


Gene
pathway
Flatten
hidden layers
output layer
IC50 Value Prediction





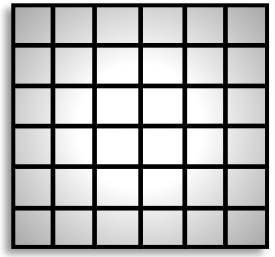
STEP E_CNN

- Vanillia CNN, ResNet 사용
- Pooling size는 [NxN]이 아닌 [Nx1]을 사용

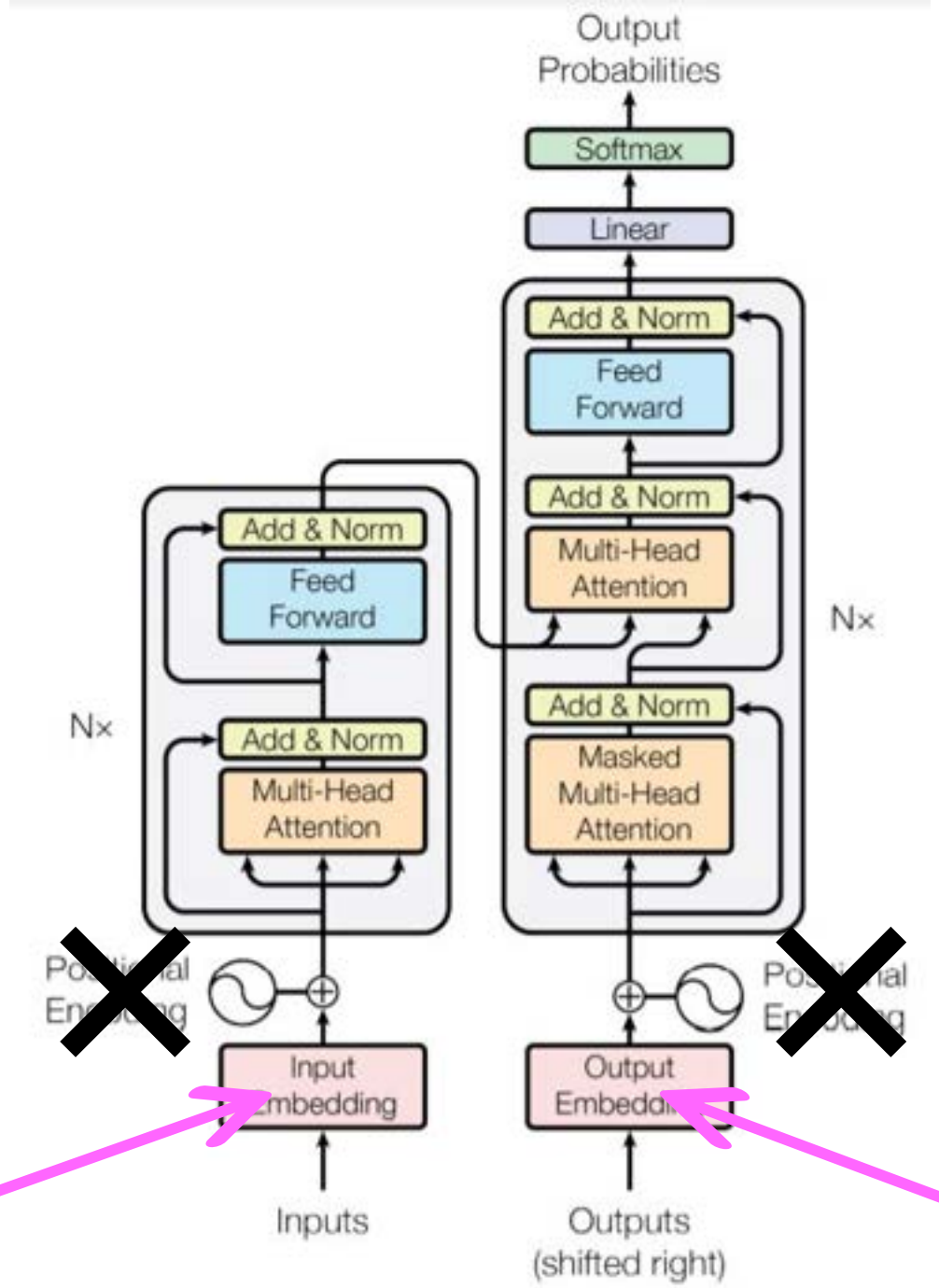
MLP

CNN

Transformer



STEP E_Transformer_1

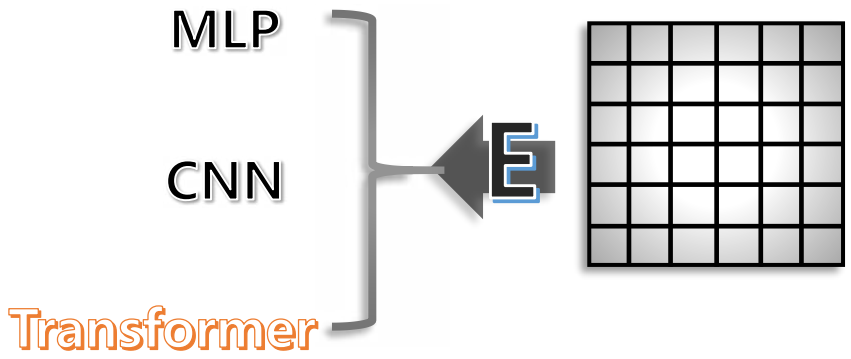


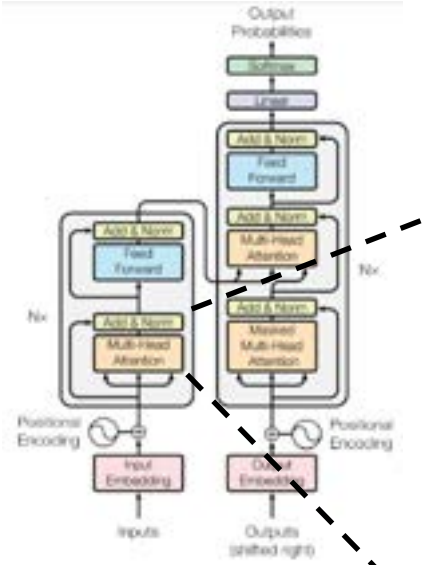
$Gene(Embedding)$

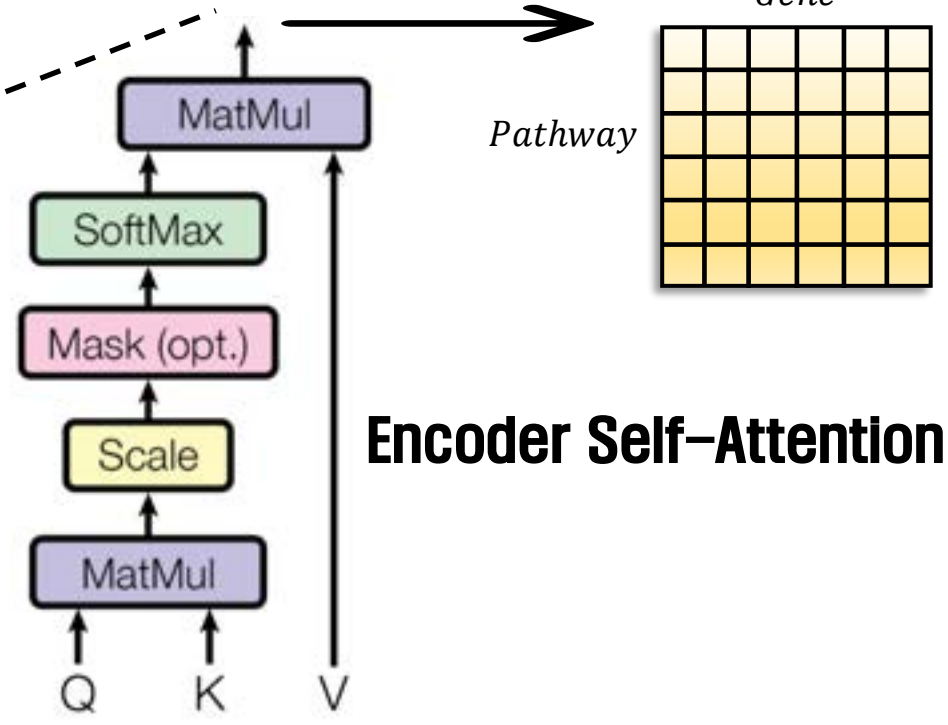
$pathway$

$pathway(Embedding)$

$gene$



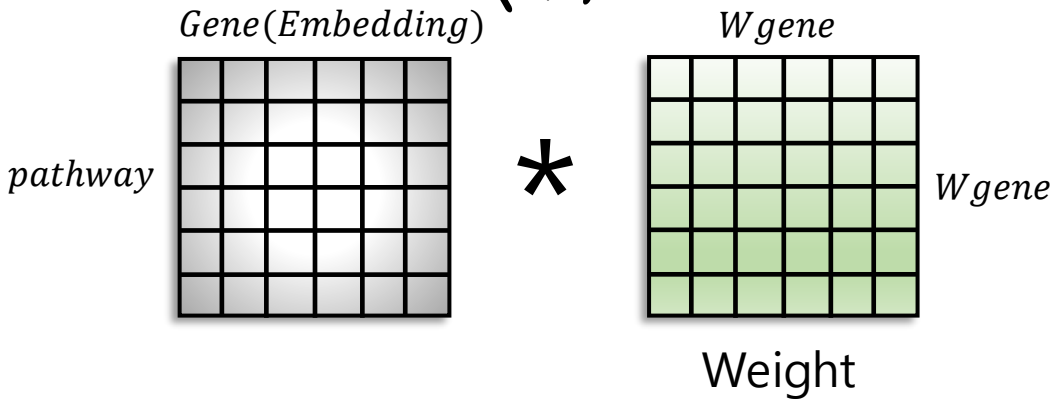




Encoder Self-Attention

STEP E_Transformer_2

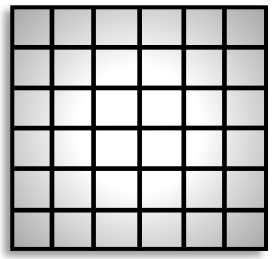
- Encoder Layer
- Layer의 개수는 1
- Key, Query, Value는 Embeding Matrix에
  (Gene수 X Gene 수) 크기의 Weight Matrix를 곱한 값으로
  모두 동일하게 설정
- Scaled Dot-Product Attention을 모두
  완료했을 때의 Attention Value의 크기는
  (Pathway 수 X Gene 수) 가 된다.

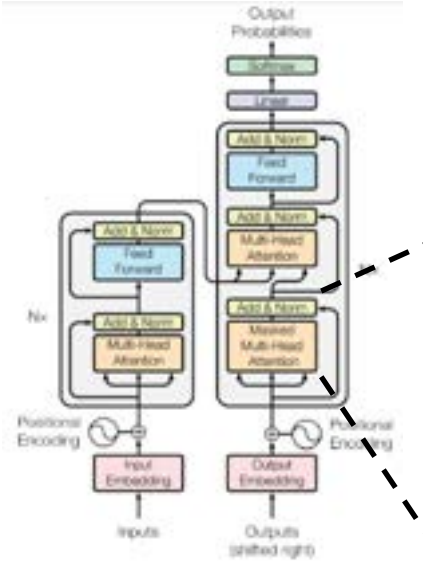


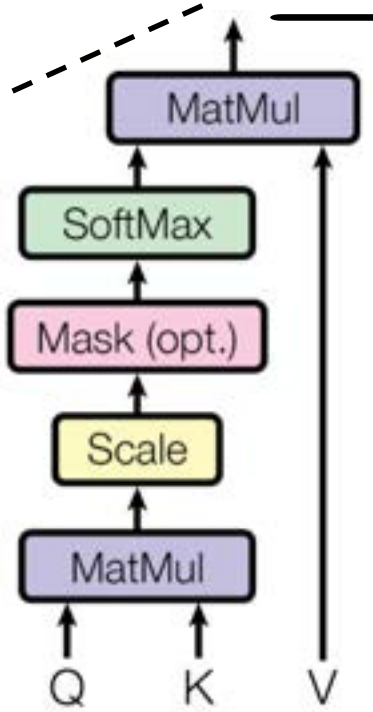
MLP

CNN

Transformer

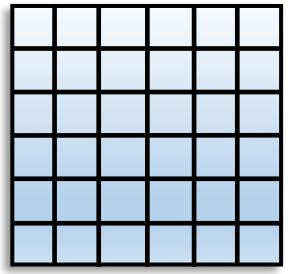






Pathway

Gene



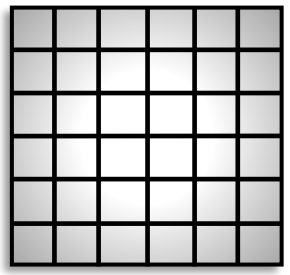
Decoder Self-Attention

STEP E_Transformer_3

- Decoder Layer
- Layer의 개수는 1
- Key, Query, Value는 Embeding Matrix에
  (Pathway수 X Pathway 수) 크기의 Weight Matrix를 곱한 값
  으로 모두 동일하게 설정
- Scaled Dot-Product Attention을 모두
  완료했을 때의 Attention Value의 크기는
  (Gene 수 X Pathway 수) 가 된다.

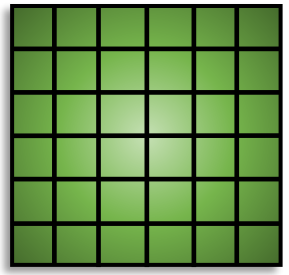
Pathway(Embedding)

Gene



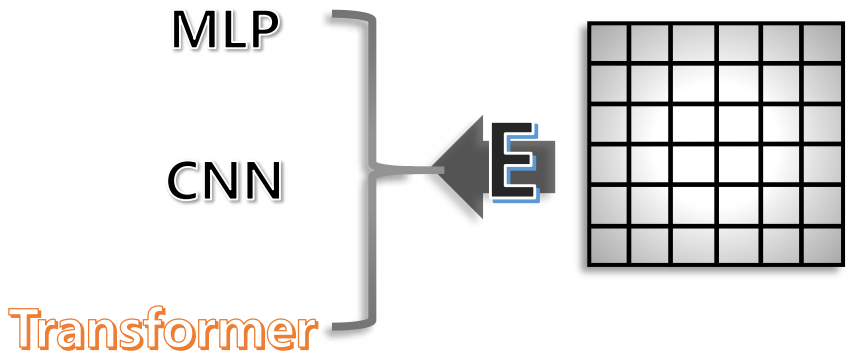
*

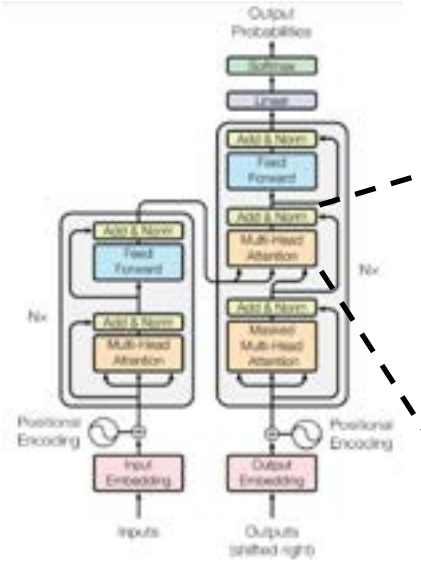
Wpathway

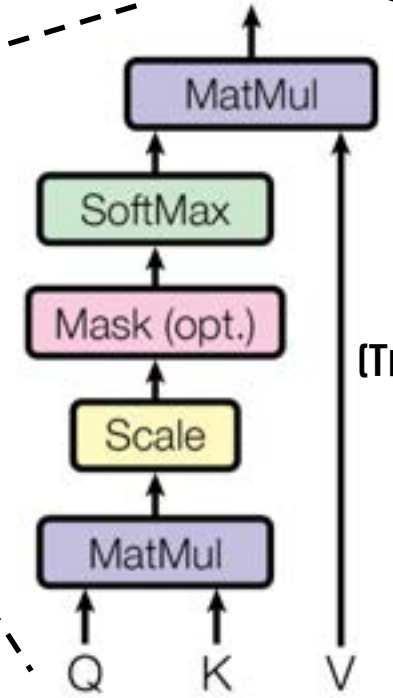


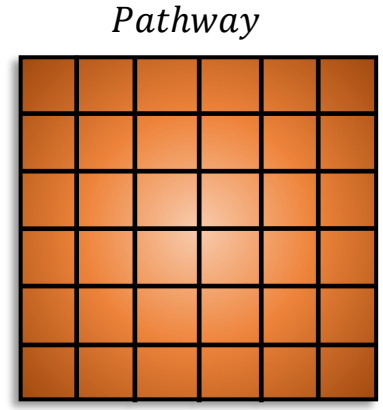
Wpathway

Weight





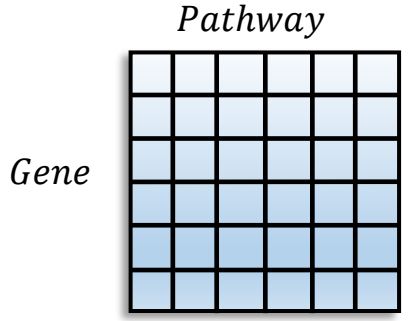




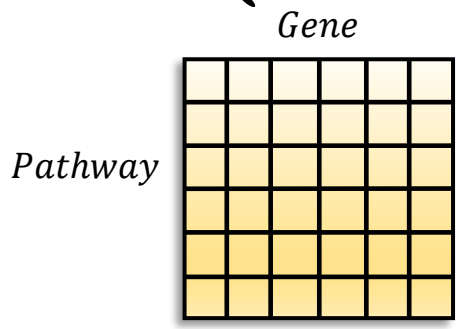
Encoder_Decoder Attention

STEP E_Transformer_5

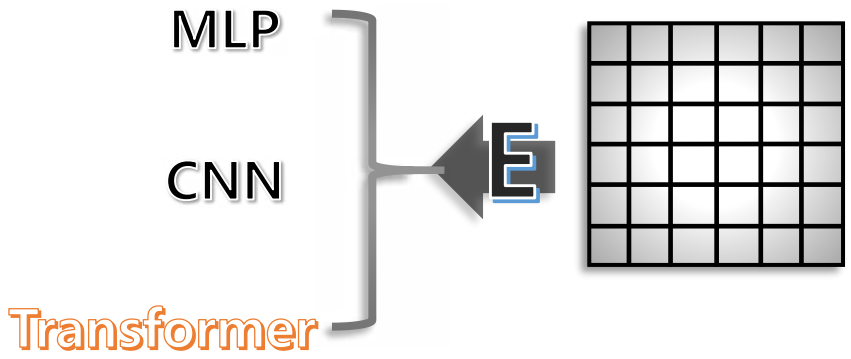
- Encoder_Decoder Layer
- Layer의 개수는 1
- Key, Value 값을 Encoder Layer에서 나온 Attention값, Query 값은 Decoder Layer에서 나온 Attention값으로 설정
- 최종 Output 값은 (Batch_size X Gene 수 X Pathway 수)의 크기를 가지는 텐서

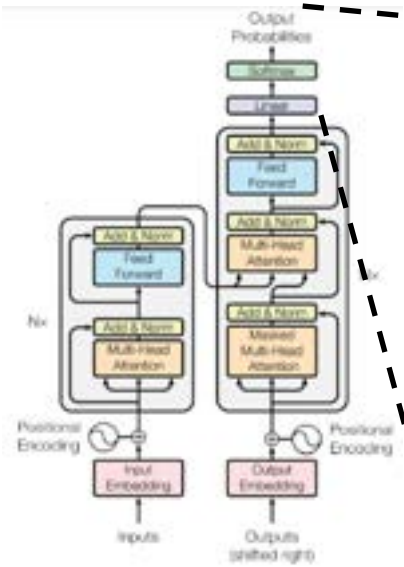


Decoder Attention Result



Encoder Attention Result





STEP E_Transformer_6

- Fully_Connected_Layer
- Encoder_Decoder Attention을 통해 나온 Ouput 텐서를 Flatten하여 MLP모델을 통해 최종 IC50 값을 예측