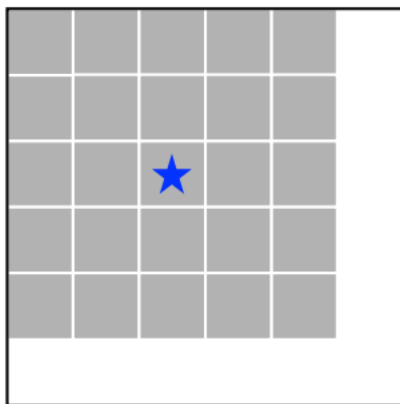


# Conformer: Convolution- augmented Transformer for Speech Recognition

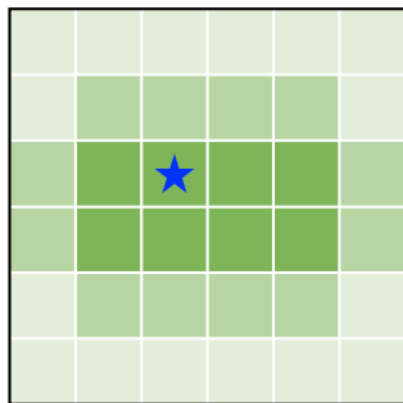
# CNN vs Transformer(MHSA)

CNN



(c) large kernel

Attention



(a) global attention

Speech 영역에서 **Transformer model**은 다른 도메인과 마찬가지로 **Long range dependency**와 **training efficiency**로 성공을 거뒀다.

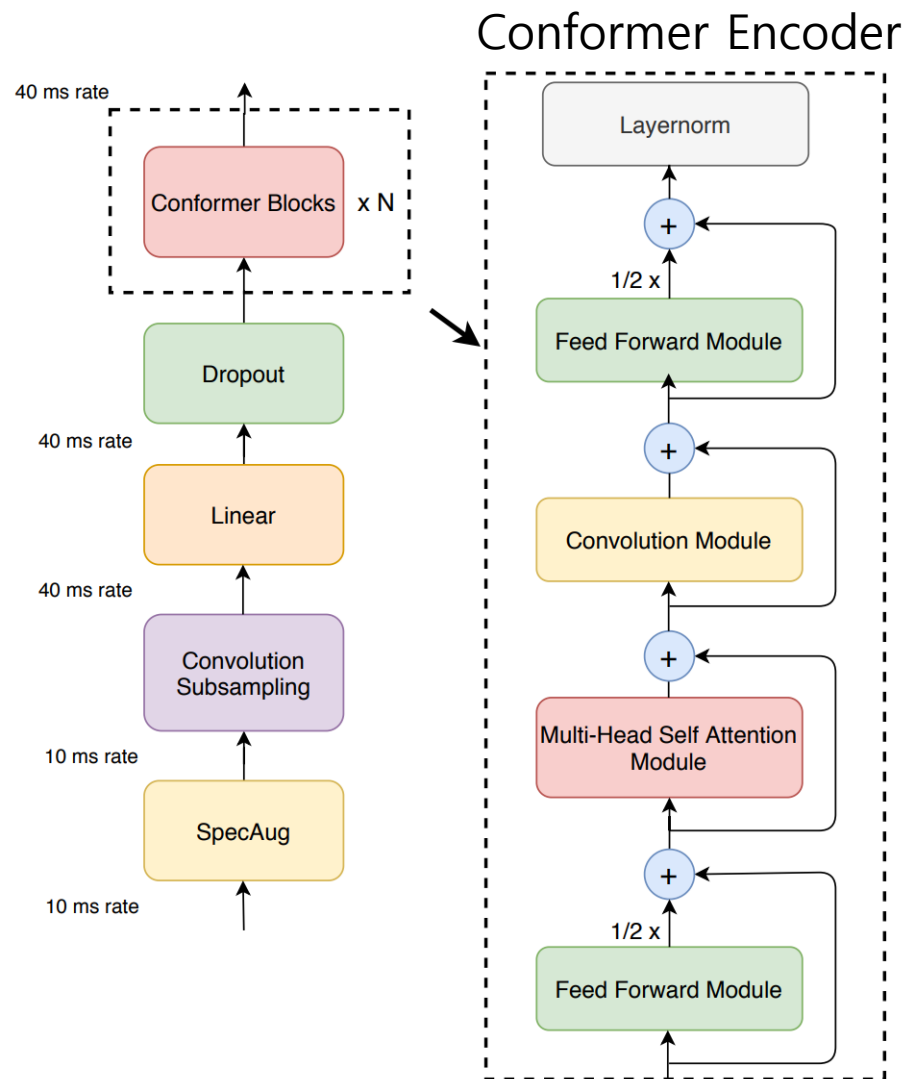
**CNN**역시 **local context**를 점진적으로 capture하기 때문에 ASR에서 성공을 거뒀다.

그러나 위의 두 모델은 각각의 단점이 있다.

**Transformer**는 **long range global context extraction**에 좋지만, **fine grained local feature patterns extraction**에는 성능이 좋지 않다.

**CNN**은 **local information extraction** 성능이 좋고, kernel을 사용하기 때문에 **translation equivariance**를 유지하며 **edge**나 **shape**을 뽑아낼 수 있지만, **global information**을 capture하기 위해서는 더 깊은 Layer나 parameter가 필요해진다.

# Conformer

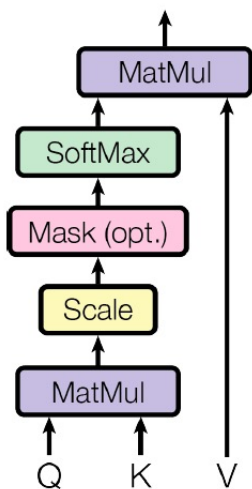


- CNN과 Transformer는 결국 각각의 문제를 가지고 있다.
- 우리는 global context와 local context의 상호작용이 parameter efficient를 위해 중요할 것이라는 가설을 세웠다.
- 그래서 이것을 해결하기 위해 CNN과 self-attention을 섞어버렸다.

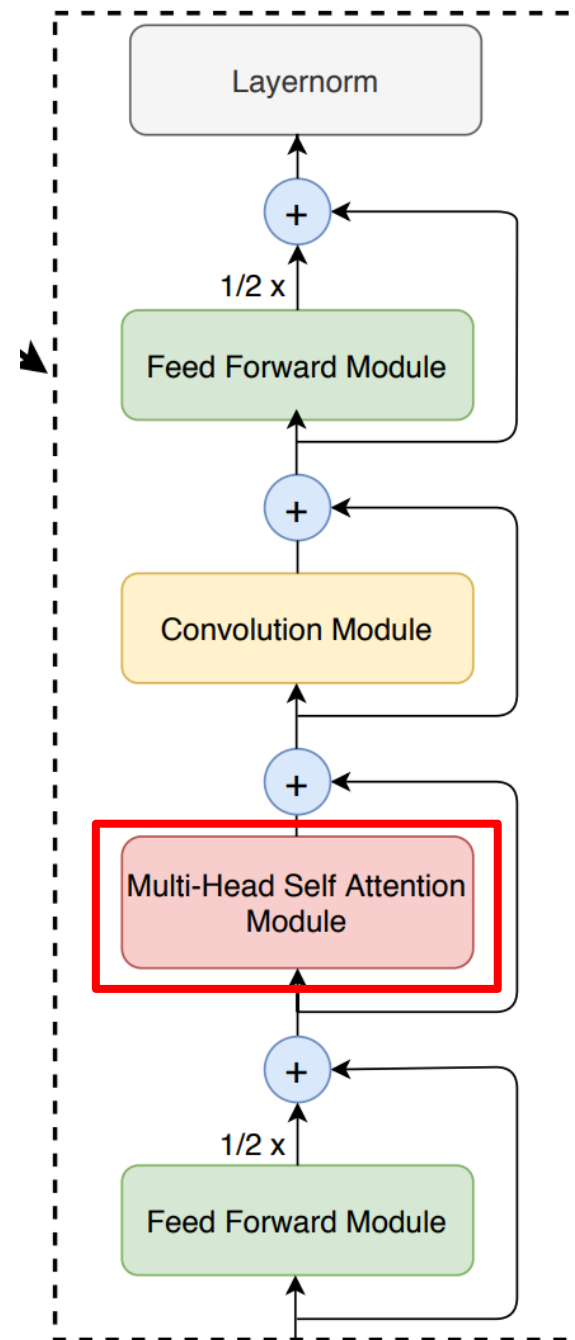
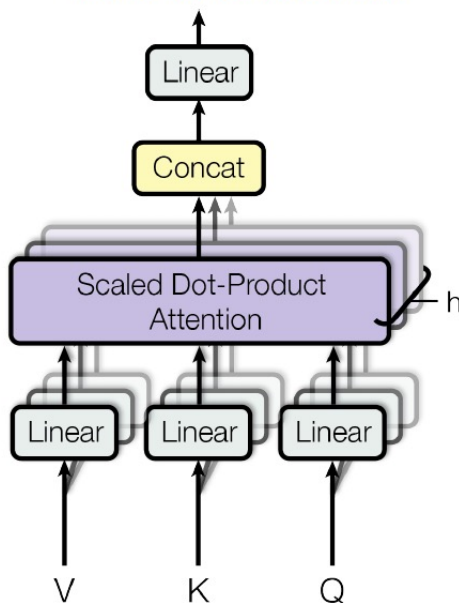
# Conformer Encoder

## 1. Multi-Head Self-Attention Module

Scaled Dot-Product Attention

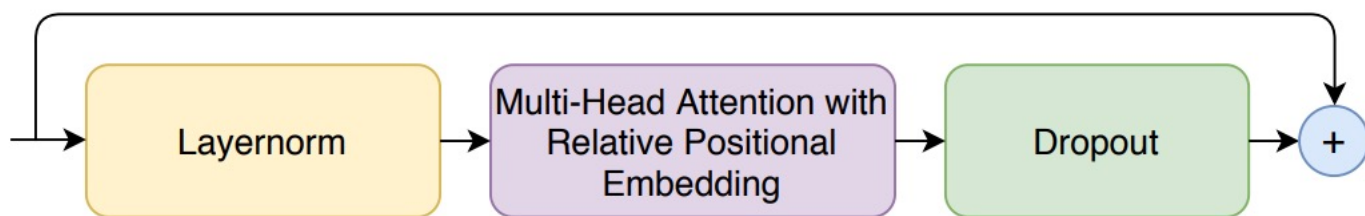


Multi-Head Attention



# Conformer Encoder

## 1. Multi-Head Self-Attention Module



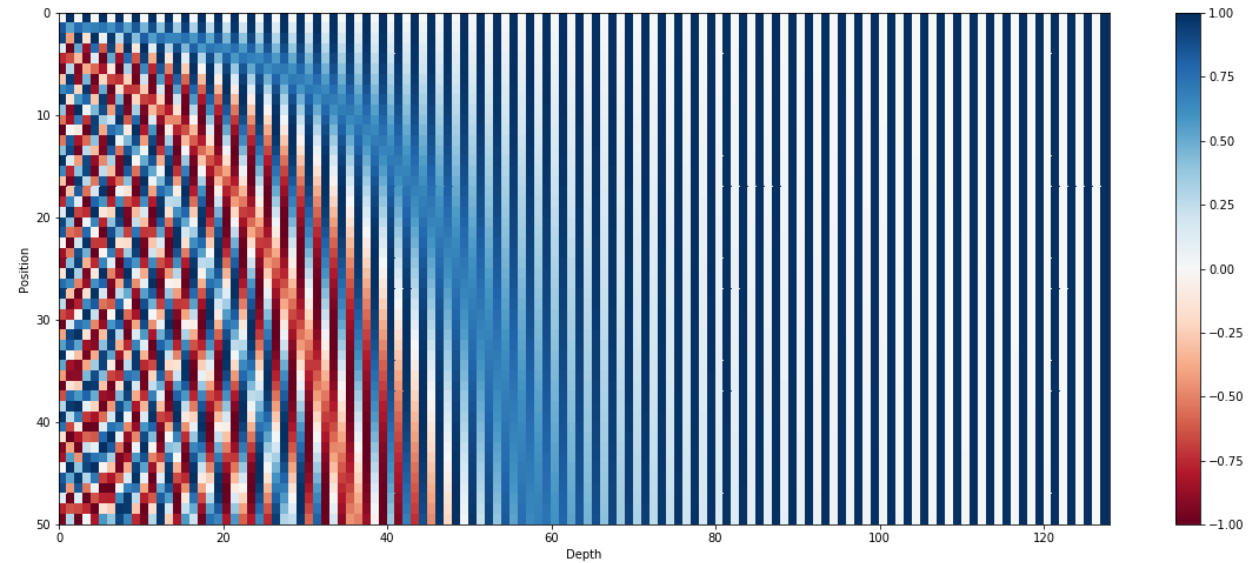
연산 전에 norm을 사용하는 pre-norm을 이용했고,  
relative positional encoding을 통해서 length가 다른 Input에도 general하게 작용할 수 있게 했다.

# Conformer Encoder

## 1-1. 기존 Positional Encoding

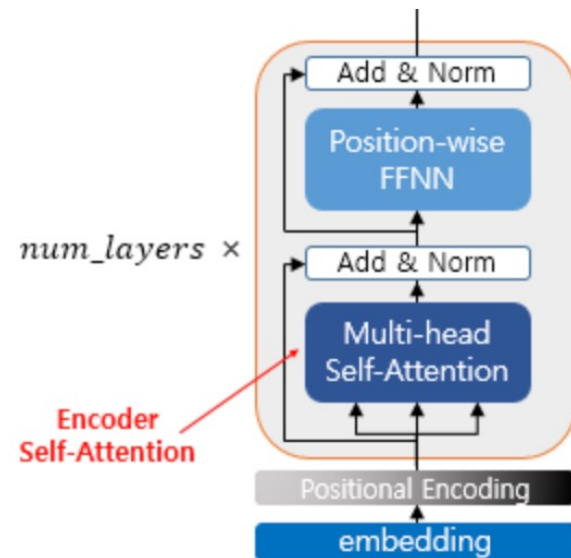
$$\vec{p}_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases}$$

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

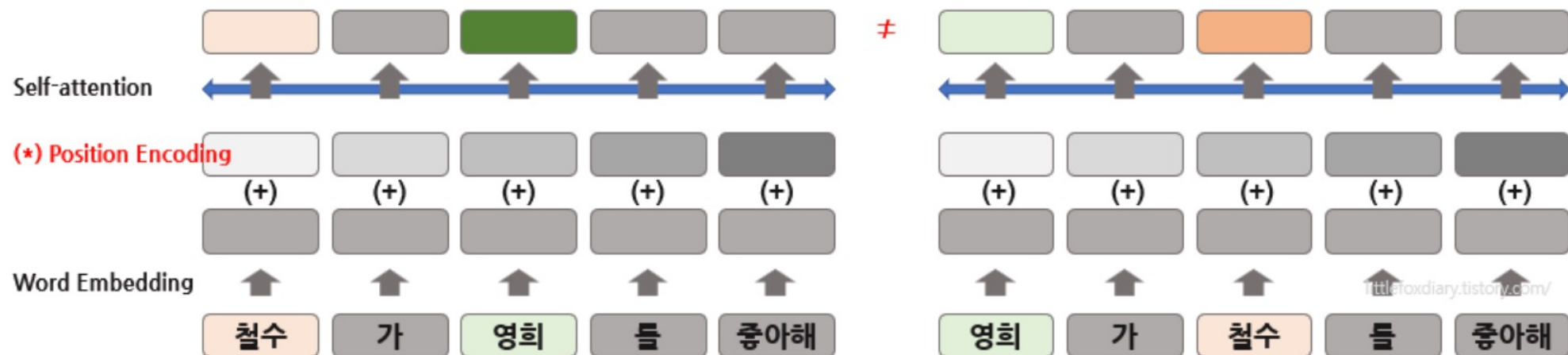


# Conformer Encoder

## 1-1. 기존 Positional Encoding



(b) **위치 인코딩**을 더해줌으로써 단어 시퀀스 정보를 줄 수 있고, 아래의 두 문장은 다른 representation을 가짐



# Conformer Encoder

## 1-2. Relative Positional Encoding

철수

가

영희

들

중아해

인풋 시퀀스 512인 문장에 대해 Self-attention 상황을 가정하면, relative position은 다음과 같이 구할 수 있다.

첫 번째 토큰을 기준으로 마지막 토큰은 511만큼 뒤에 있음

```
tensor([[ 0,  1,  2,  3,  4, ..., 507, 508, 509, 510, 511],
        [-1,  0,  1,  2,  3, ..., 506, 507, 508, 509, 510],
        [-2, -1,  0,  1,  2, ..., 505, 506, 507, 508, 509],
        [-3, -2, -1,  0,  1, ..., 504, 505, 506, 507, 508],
        [-4, -3, -2, -1,  0, ..., 503, 504, 505, 506, 507],
        ...,
        [-507, -506, -505, -504, -503, ...,  0,  1,  2,  3,  4],
        [-508, -507, -506, -505, -504, ..., -1,  0,  1,  2,  3],
        [-509, -508, -507, -506, -505, ..., -2, -1,  0,  1,  2],
        [-510, -509, -508, -507, -506, ..., -3, -2, -1,  0,  1],
        [-511, -510, -509, -508, -507, ..., -4, -3, -2, -1,  0]])
```

마지막 토큰을 기준으로 첫 번째 토큰은 511만큼 앞에 있음

자기 자신과의 거리는 0



# Conformer Encoder

## 1-2. Relative Positional Encoding

철수

가

영희

들

중아해

이제 이 relative position을 \_relative\_position\_bucket에 대입하면 **상대적인 거리에 따른 버킷** 값을 얻을 수 있다.

기준 토큰과 가까운 거리의 토큰은 각기 다른 버킷 할당

tensor([[ 0, 17, 18, 19, 20, ..., 31, 31, 31, 31, 31],  
[ 1, 0, 17, 18, 19, ..., 31, 31, 31, 31, 31],  
[ 2, 1, 0, 17, 18, ..., 31, 31, 31, 31, 31],  
[ 3, 2, 1, 0, 17, ..., 31, 31, 31, 31, 31],  
[ 4, 3, 2, 1, 0, ..., 31, 31, 31, 31, 31],  
...,  
[15, 15, 15, 15, 15, ..., 0, 17, 18, 19, 20],  
[15, 15, 15, 15, 15, ..., 1, 0, 17, 18, 19],  
[15, 15, 15, 15, 15, ..., 2, 1, 0, 17, 18],  
[15, 15, 15, 15, 15, ..., 3, 2, 1, 0, 17],  
[15, 15, 15, 15, 15, ..., 4, 3, 2, 1, 0]])

멀리 떨어진 토큰의 위치정보는 덜 민감하게 반영

Bidirectional이기 때문에 기준 토큰 앞/뒤로 다른 버킷 할당

# Conformer Encoder

## 1-2. Relative Positional Encoding

기존 Self-Attention  
Positional Encoding은 사전에 더해짐

$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V)$$

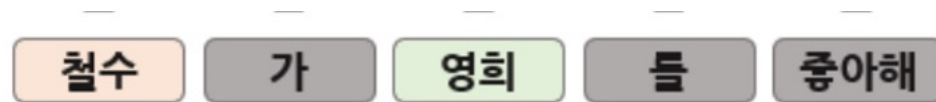
input and output

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^n \exp e_{ik}}$$

softmax function

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}}$$

attention score



Relative positional encoding은  
Attention 계산에 직접 encoding 반영  
(a<sub>ij</sub>)

$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V + a_{ij}^V)$$

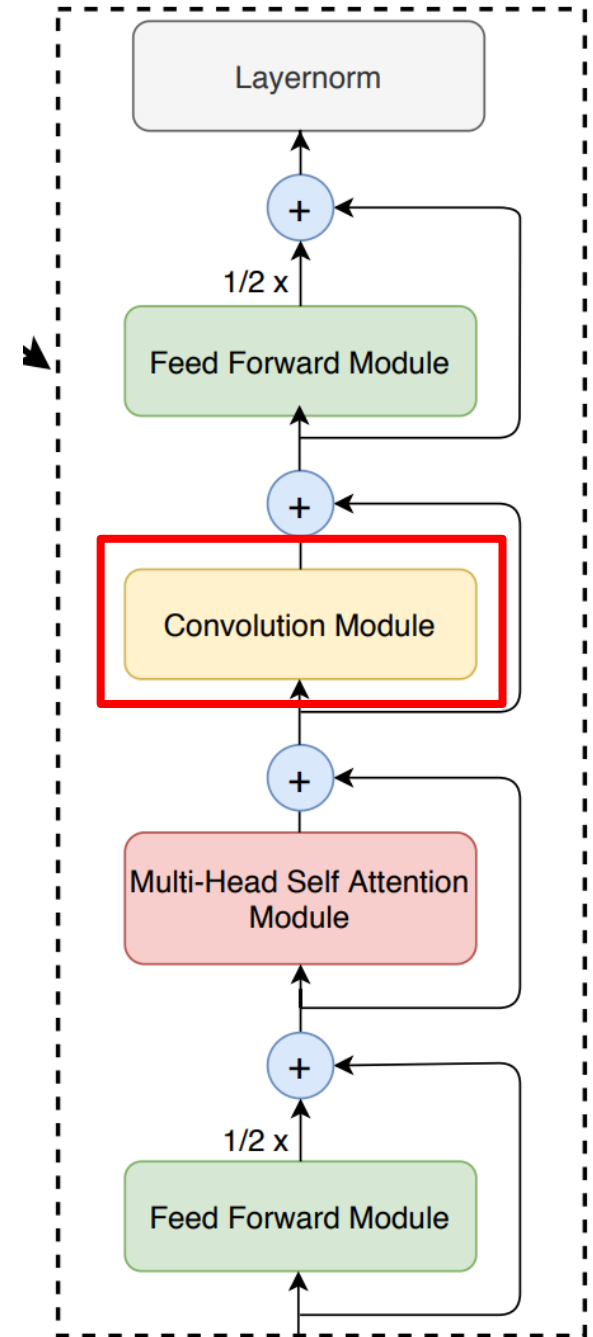
new input and output

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$

new attention score

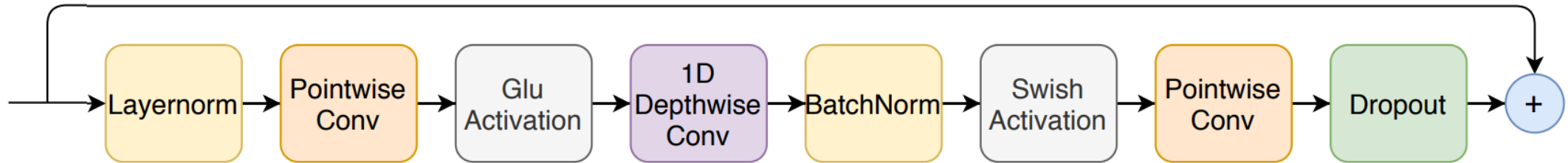
# Conformer Encoder

## 2. Convolution Module



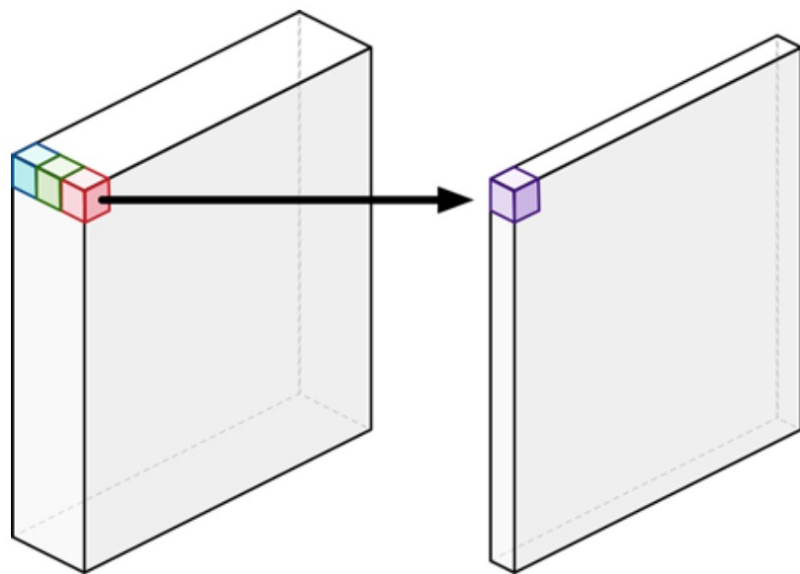
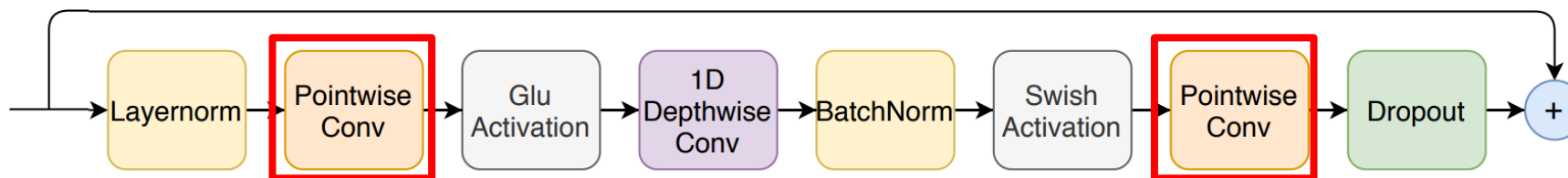
# Conformer Encoder

## 2. Convolution Module



# Conformer Encoder

## 2. Convolution Module



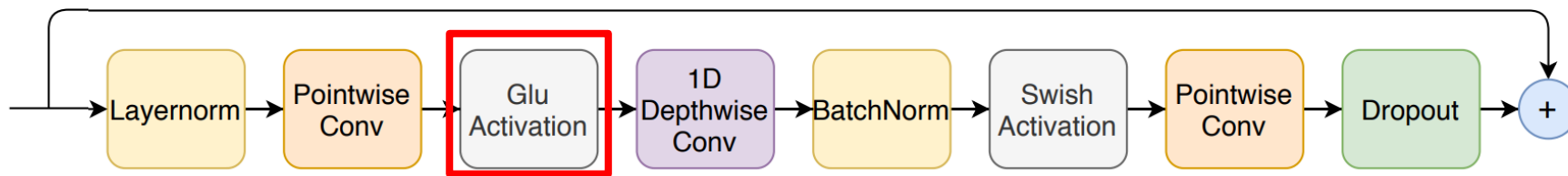
Filter 수 조절 가능

Bottleneck 구조로 연산 효율화 가능

Filter수를 줄이고 연산 후 다시 Filter 늘리기

# Conformer Encoder

## 2. Convolution Module



### Gated Linear Unit

It is used in natural language processing architectures, for example the [Gated CNN](#), because here  $b$  is the gate that control what information from  $a$  is passed up to the following layer. Intuitively, for a language modeling task, the gating mechanism allows selection of words or features that are important for predicting the next word. The GLU also has non-linear capabilities, but has a linear path for the gradient so diminishes the vanishing gradient problem.

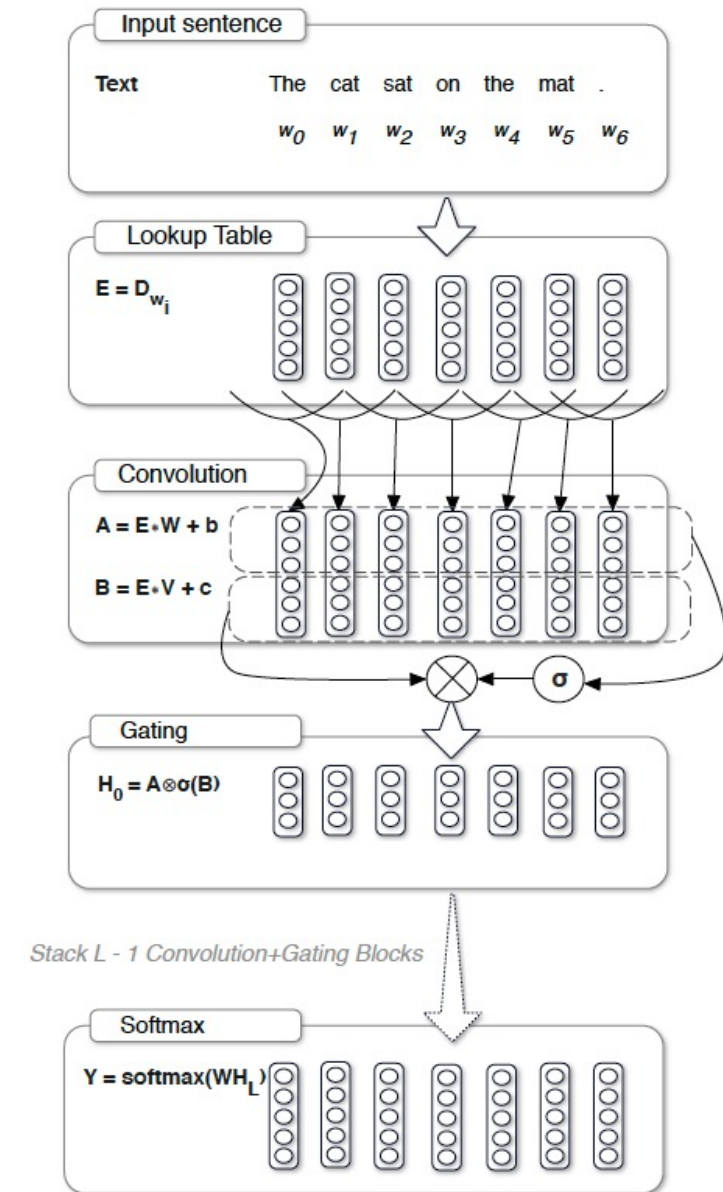
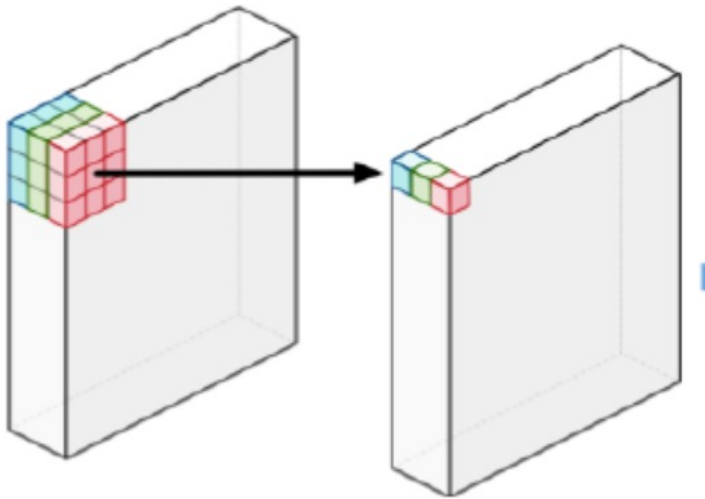
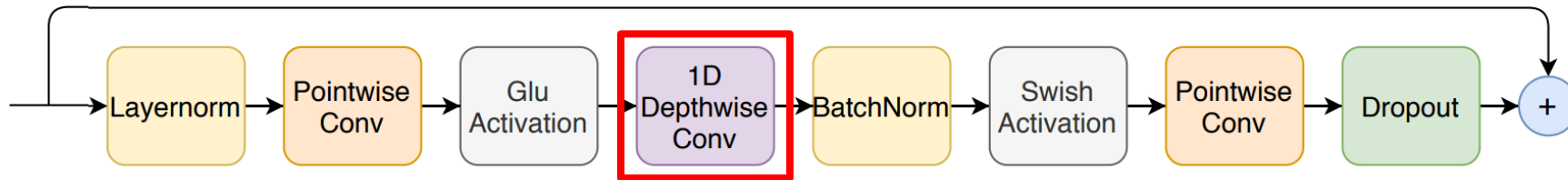


Figure 1. Architecture of the gated convolutional network for language modeling.

# Conformer Encoder

## 2. Convolution Module

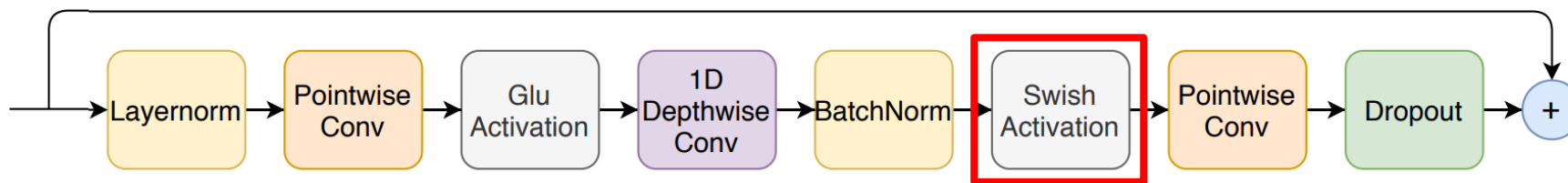


Channel별 convolution

원래 Depthwise Conv -> Pointwise Conv 순으로 쓰임

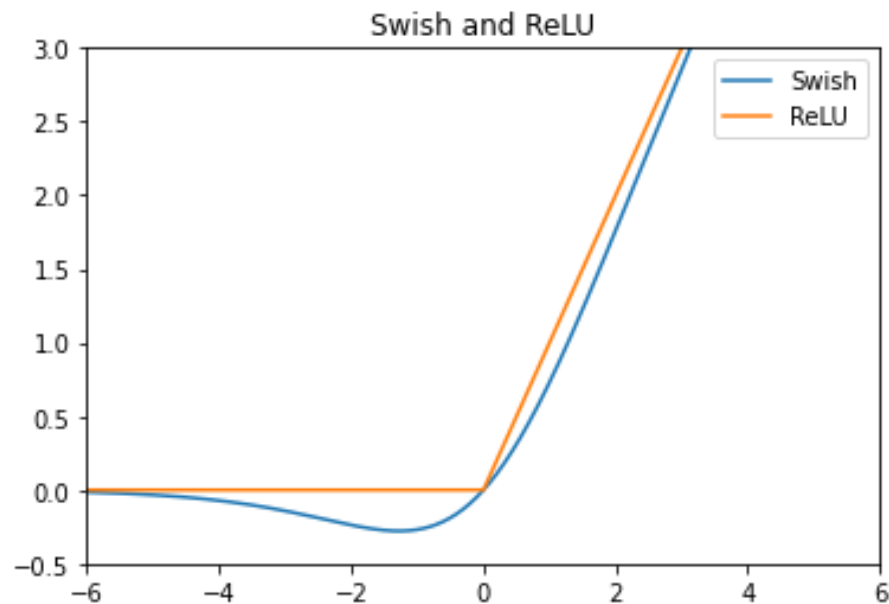
# Conformer Encoder

## 2. Convolution Module



$$f(x) = x \cdot \sigma(x),$$

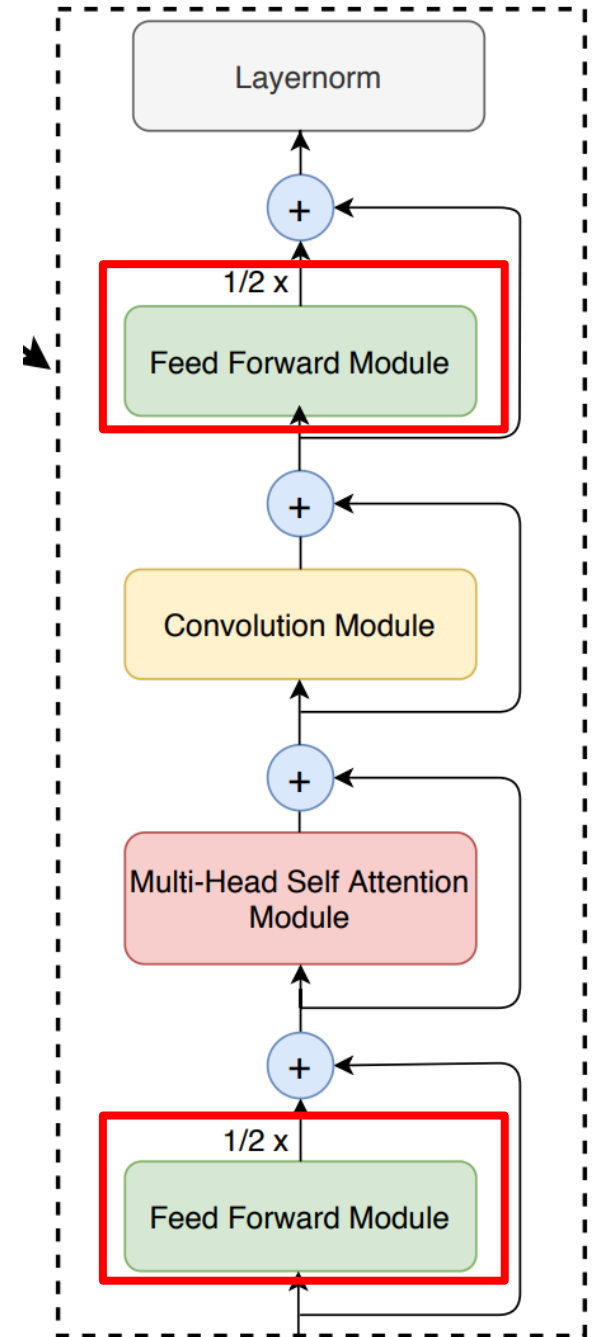
Sigmoid Activation Function에  $x$ 를 곱해준 단순한 형태  
Sigmoid( $B \cdot x$ )로  $B$ 를 학습 가능한 파라미터로 넣기도 한다.  
SiLU라고도 불림





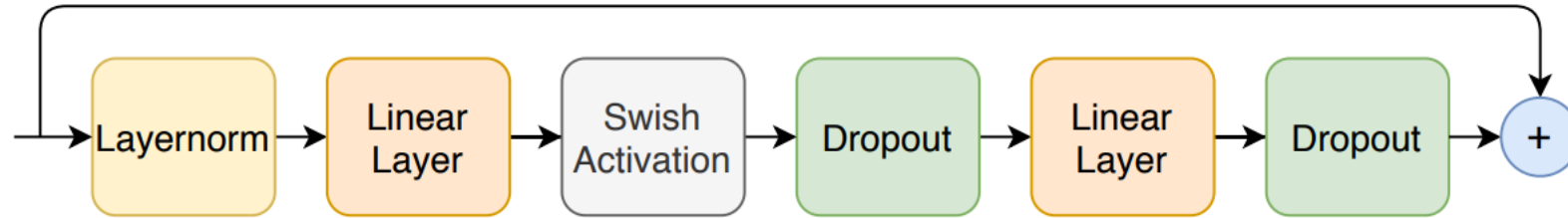
# Conformer Encoder

## 3. FeedForward Module



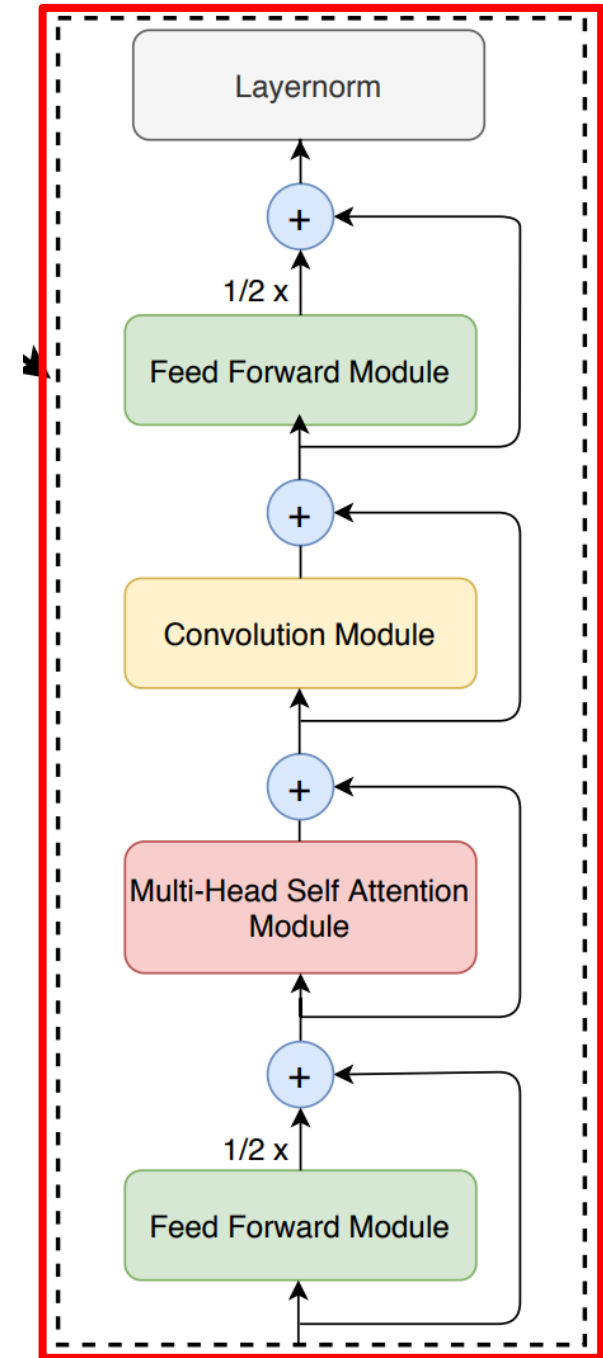
# Conformer Encoder

## 3. FeedForward Module



# Conformer Encoder

## 4. Encoder Block



# Conformer Encoder

## 4. Encoder Block

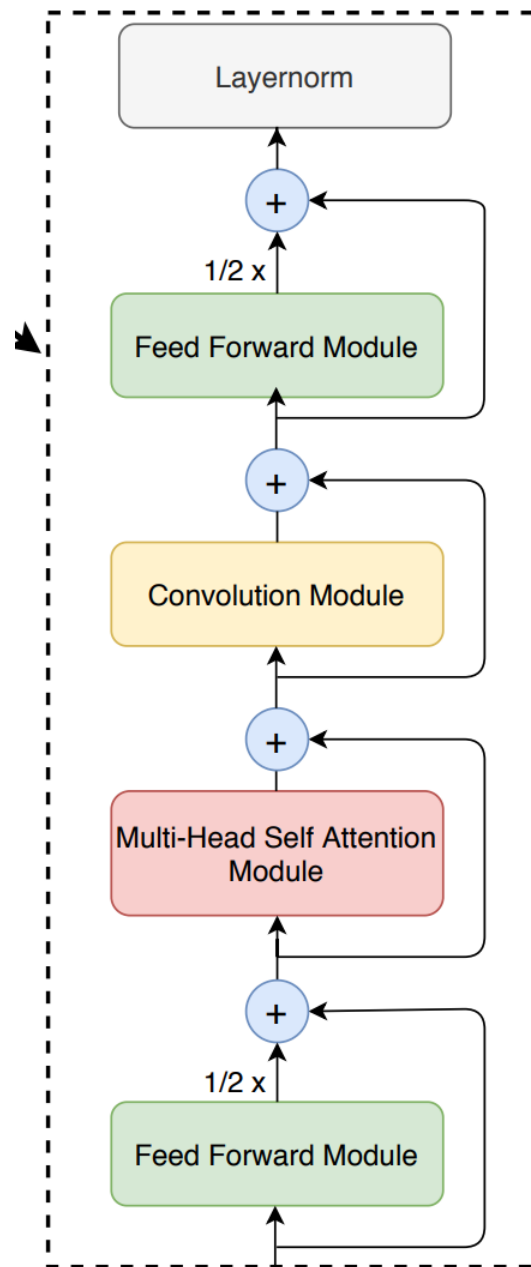
Macaron-Net에서 영감을 받아 sandwich structure를 고안  
FFNN을 2-half-step FFNN으로 변경

$$\tilde{x}_i = x_i + \frac{1}{2}\text{FFN}(x_i)$$

$$x'_i = \tilde{x}_i + \text{MHSA}(\tilde{x}_i)$$

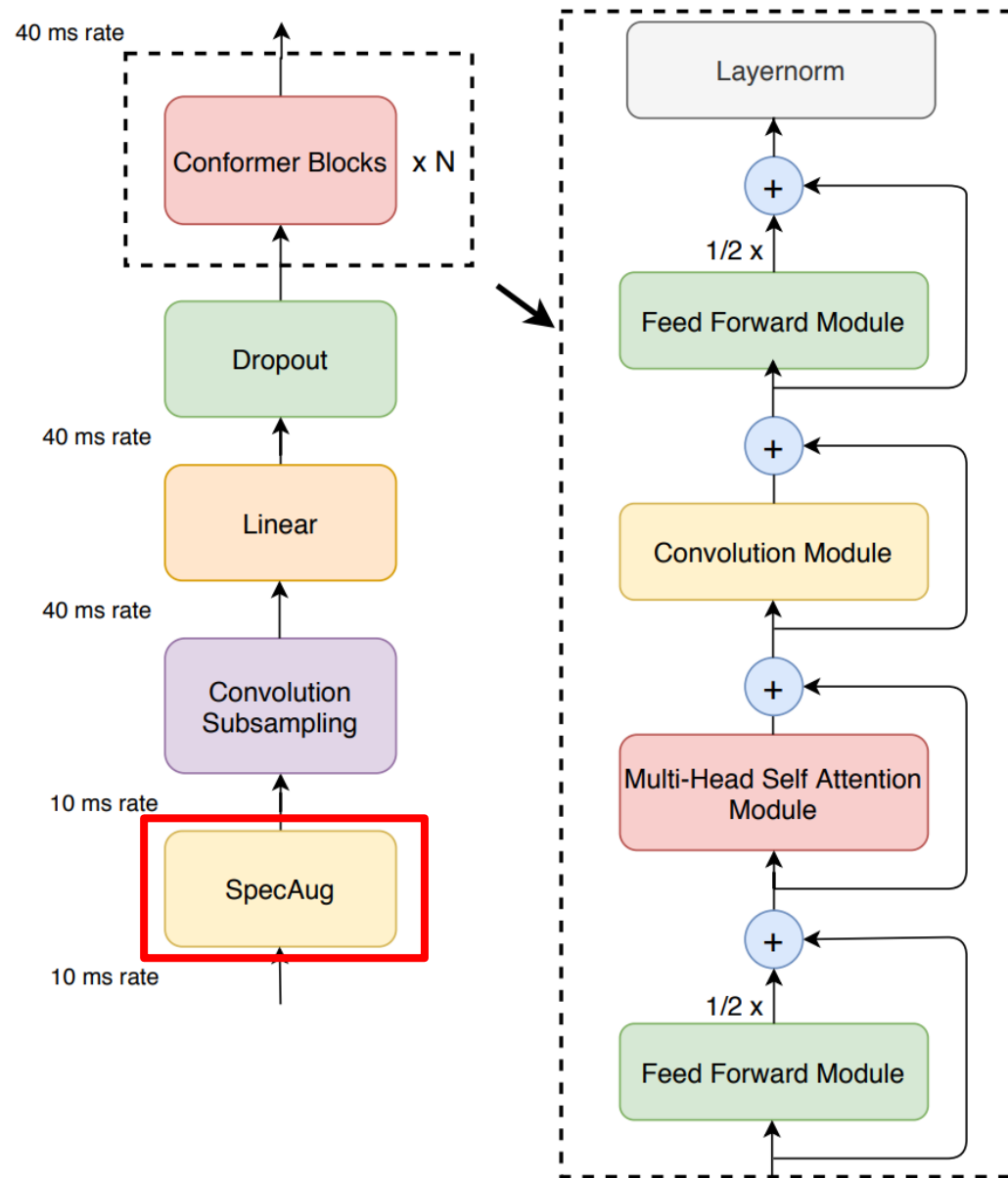
$$x''_i = x'_i + \text{Conv}(x'_i)$$

$$y_i = \text{Layernorm}(x''_i + \frac{1}{2}\text{FFN}(x''_i))$$



# Conformer

## 5. SpecAug

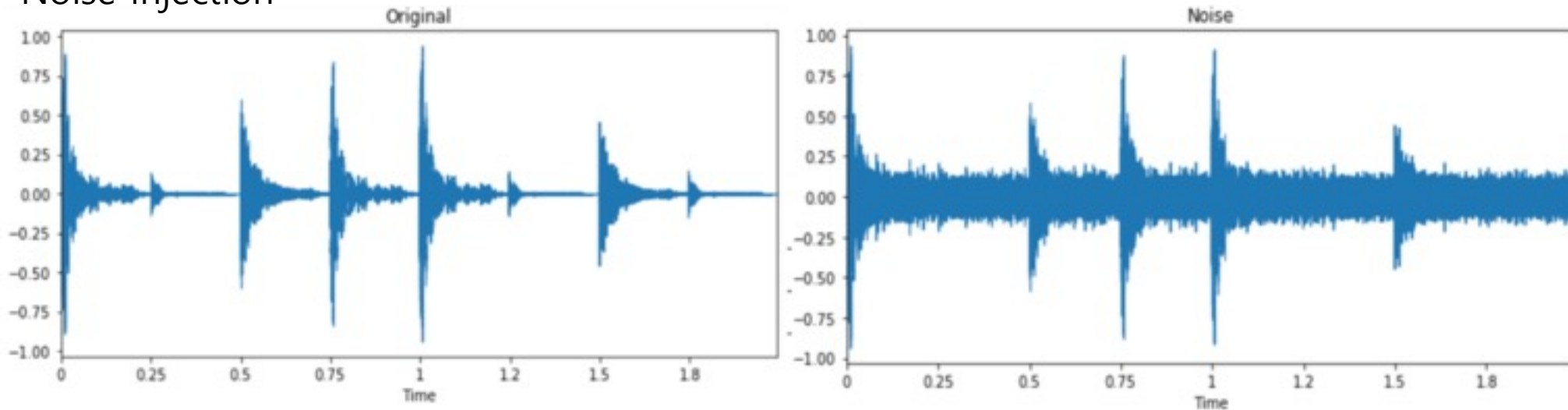


# Conformer

## 5. SpecAugment

기본적인(옛날에 쓰던) Speech Augmentation

Noise Injection



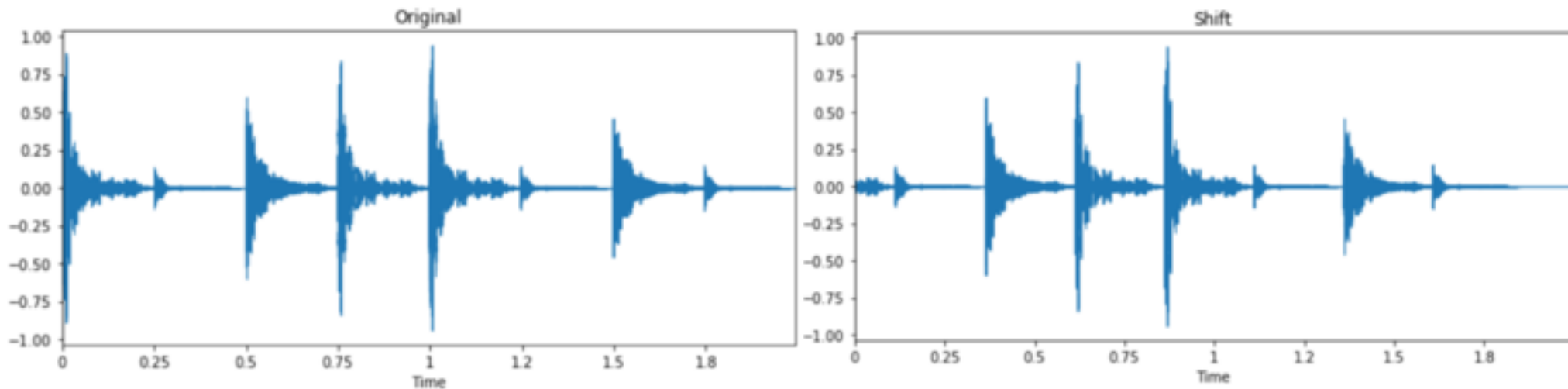
Raw audio에 임의의 난수를 더하여 Noise 추가

# Conformer

## 5. SpecAugment

기본적인(옛날에 쓰던) Speech Augmentation

Shifting Time



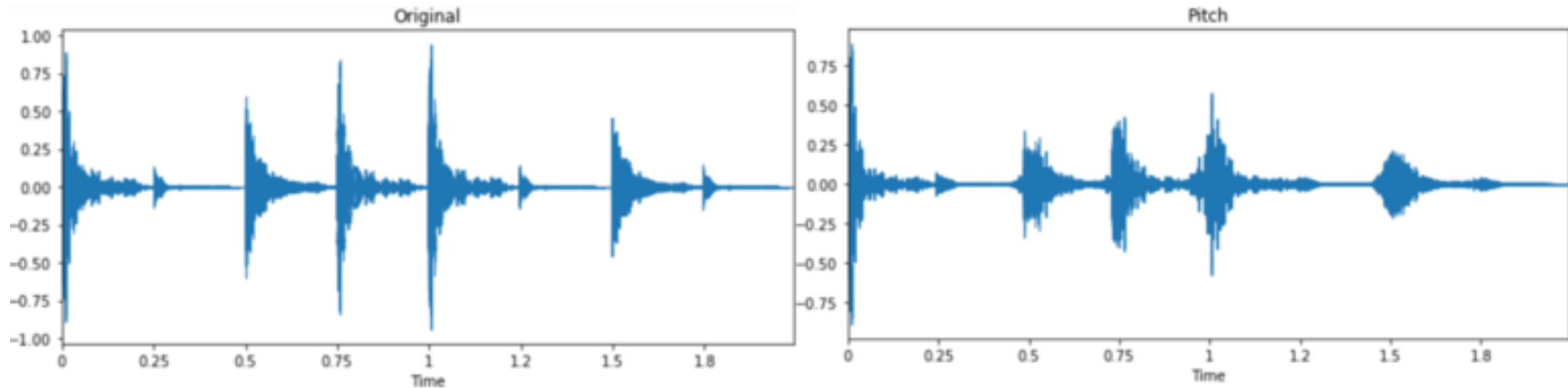
Raw audio를 한 방향으로 밀고 빈 공간 0으로

# Conformer

## 5. SpecAugment

기본적인(옛날에 쓰던) Speech Augmentation

Changing Pitch



Raw audio의 Pitch(음높이 or 주파수)를 조정

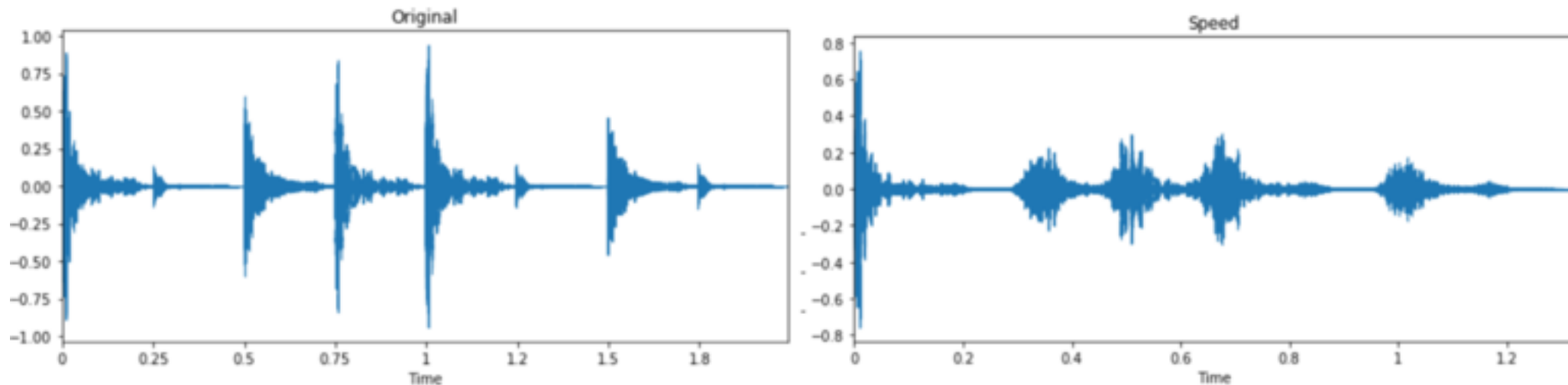


# Conformer

## 5. SpecAugment

기본적인(옛날에 쓰던) Speech Augmentation

Changing Speed



Raw audio의 속도를 바꿔줌

# Conformer

## 5. SpecAugment

본 논문에서는 굳이 Raw Audio를 바꿀 필요가 없다고 주장

MFCC / Log Mel-spectrogram을 사용하는데, 여기서 augmentation을 하자!

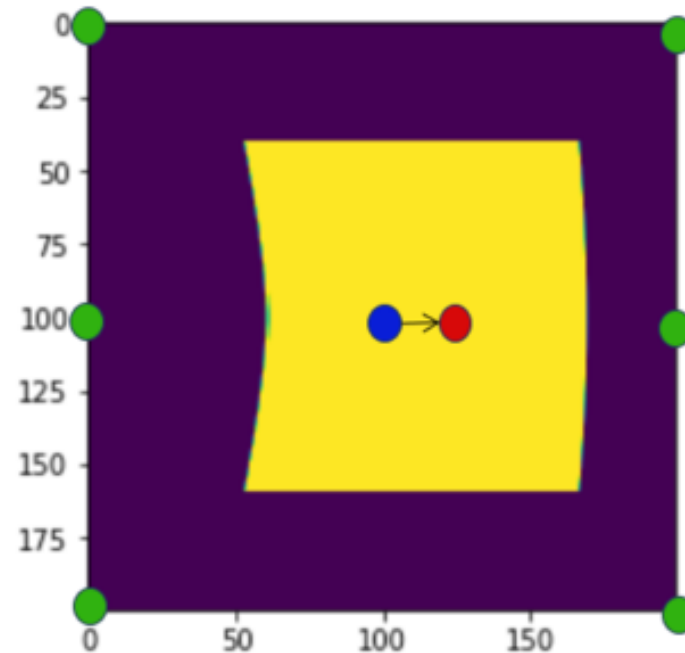
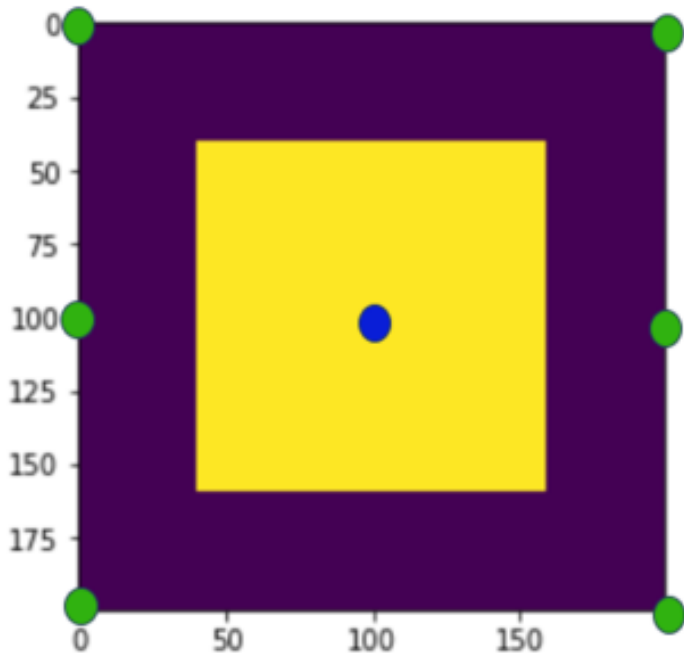
MFCC / Log Mel-spectrogram을 이미지와 비슷하게 생각하고 Augmentation기법 3가지를 제시

위의 방법으로 SOTA달성

# Conformer

## 5. SpecAugment

### (1) Time Warping

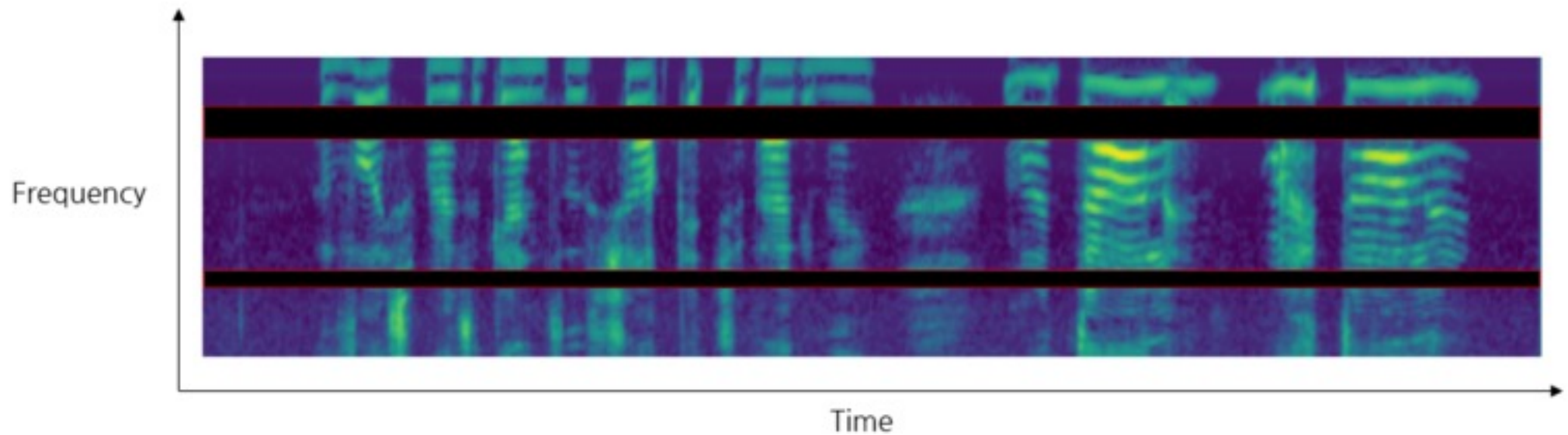


MFCC / log mel-spectrogram 를 이미지처럼 보고 Warping을 수행

# Conformer

## 5. SpecAugment

### (2) Frequency Masking

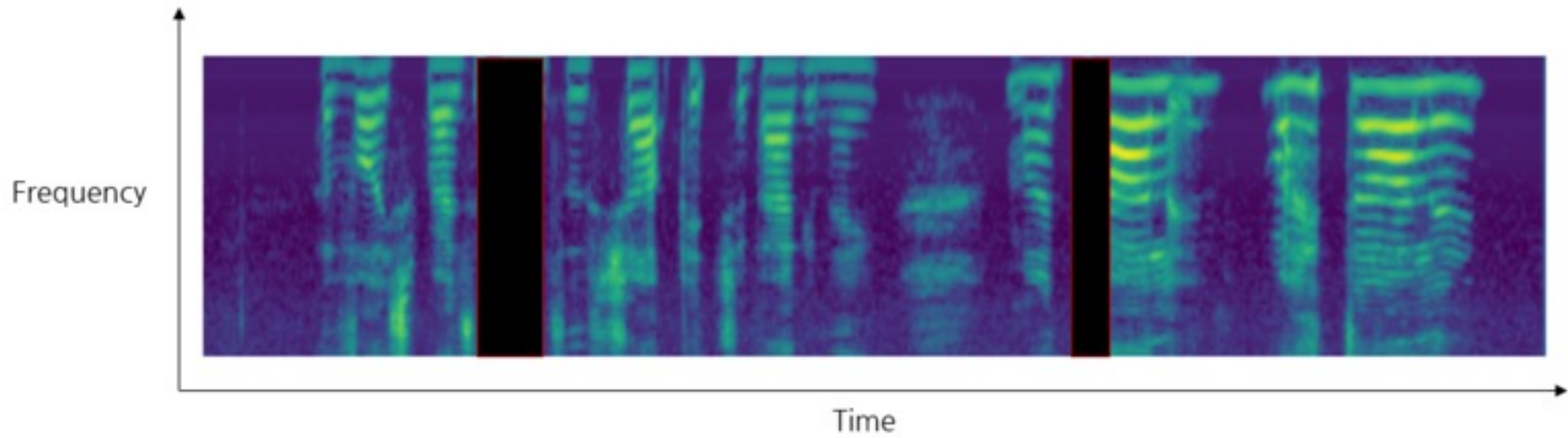


MFCC / log mel-spectrogram의 Frequency축을 그냥 간단하게 mask해버림

# Conformer

## 5. SpecAugment

### (3) Time Masking



MFCC / log mel-spectrogram의 Time축을 그냥 간단하게 mask해버림

# Conformer

