

Temporal Convolutional Networks

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

위 자료는 <https://medium.com/unit8-machine-learning-publication/temporal-convolutional-networks-and-forecasting-5ce1b6e97ce4> 논문 리뷰를 참고하여 제가 공부하면서 정리하기 위하여 ppt 버전으로 만든 것입니다. 따라서, 보다 자세한 설명은 위 링크에서 찾을 수 있으며, ppt 에도 미디엄 게시글의 figures, images 가 많이 들어가있습니다.

Index

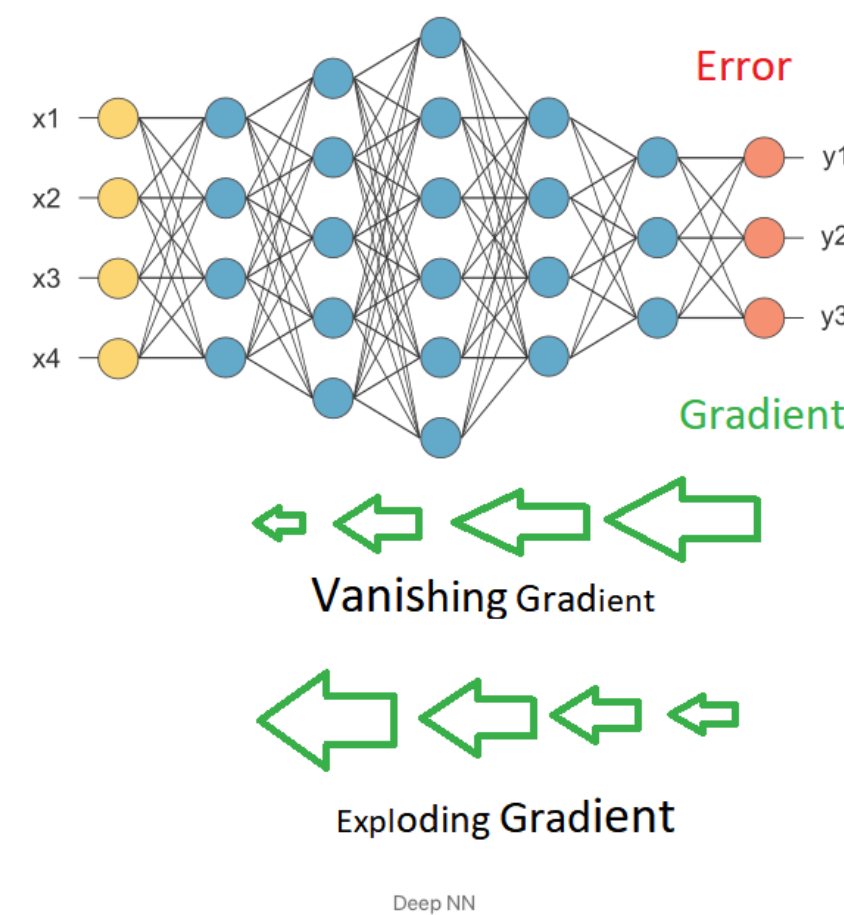
- 동기
- 모델 핵심 포인트 1) Casual Convolution
- 모델 핵심 포인트 2) Dilation
- 모델 핵심 포인트 3) Residual Blocks
- 전체 모델 구조
- 성능 및 결론

Motivation

RNN,LSTM 의 단점

deep learning auto regressive model

- gradient vanishing / gradient exploding (어느 모델이나 있겠지만 특히 rnn,lstm 계열의 고질적인 문제)
- Lacking memory retention



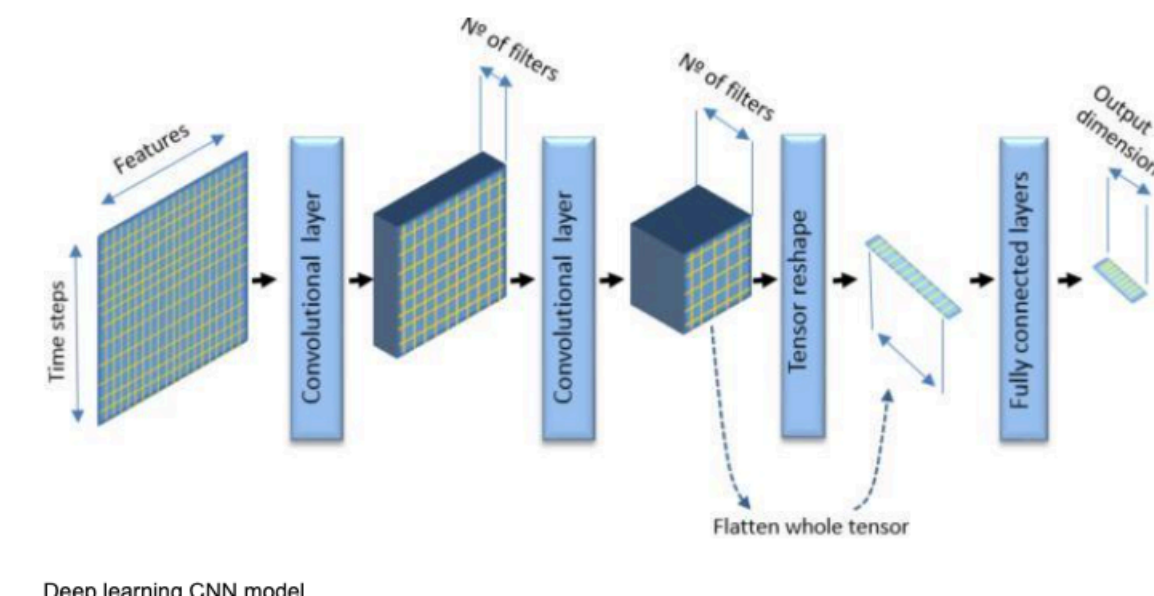
CNN 의 장단점

장점

- parallel computation for outputs

단점

- 단점이라기 보다는 CNN 계열이다 보니까 forecasting 에는 past information 만을 이용해야 하는데 cnn kernel 은 그렇게 작동하지 않음



=> CNN 모델을 forecasting 에 맞게 활용해보자 !


TCN

casual convolution

1D Convolutional Network

- 1D 이므로 conv direction 0 | 1-direction (stride is always set to 1)
- cf. 2D conv 이면 conv direction 0 | 2- direction

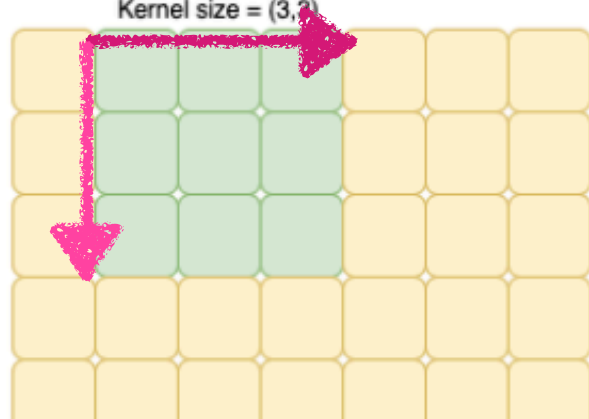
1D CNN



Kernel size = 3

Input shape = 2D
Batch = None
Width = Time axis = 7
Feature maps / Channels = 1

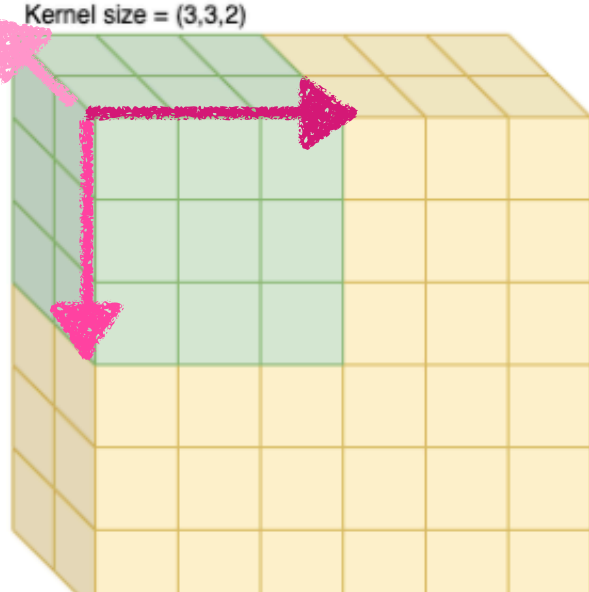
2D CNN



Kernel size = (3, 3)

Input shape = 3D
Height = 5
Width = 7
Feature maps / Channels = 1

3D CNN



Kernel size = (3, 3, 2)

Input shape = 4D
Height = 6
Width = 6
Depth = 2
Feature maps / Channels = 1

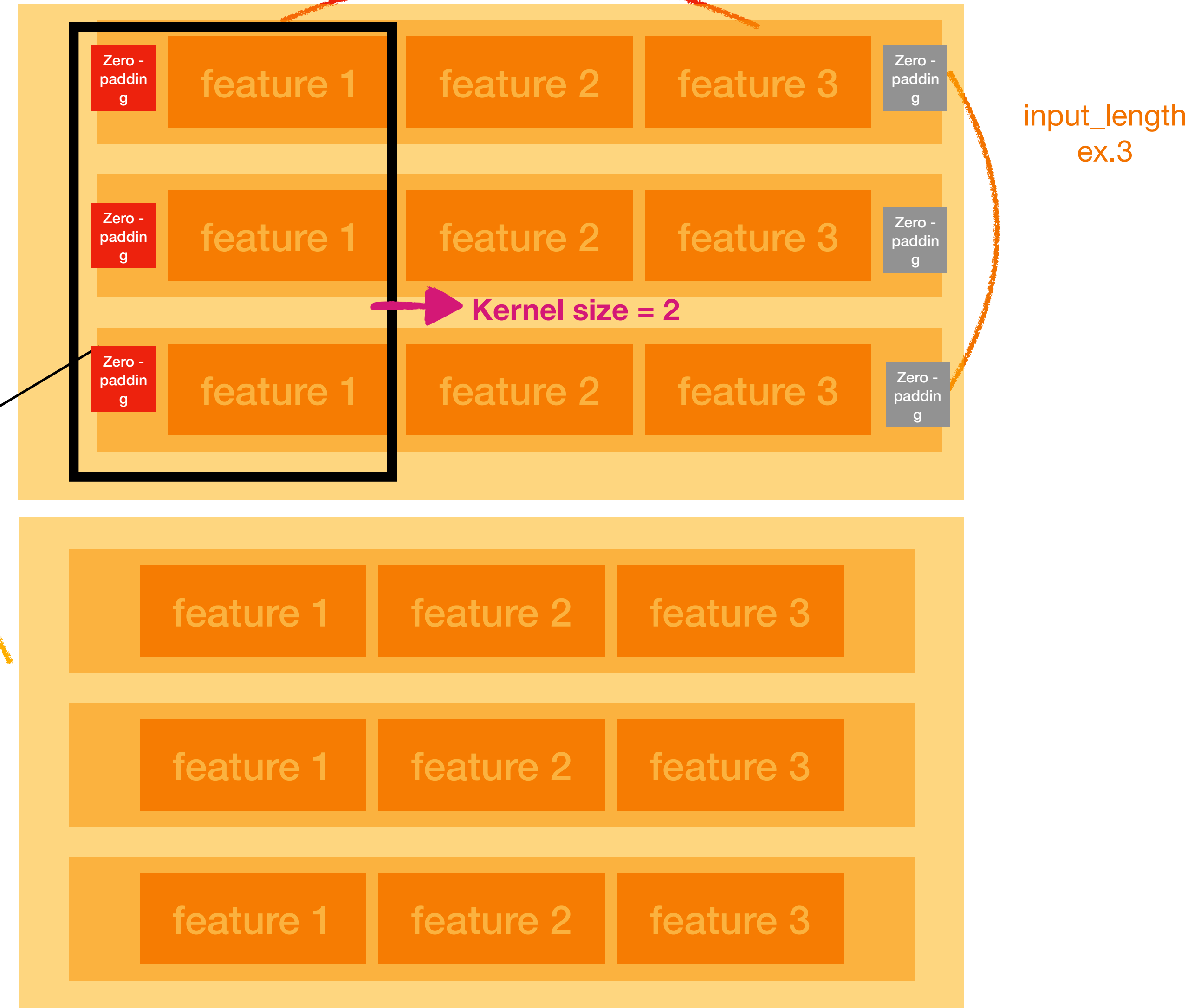
These images only have a single channel here,
but can have multiple such as R, G, B

batch_size
ex.2



Input shape :
(batch_size, input_length, input_size)

Multivariate case) input_size ex.3



casual Convolutional Network

- Casual 이란 ? (No spoilers , please ~!)

i번째 output 값을 예측하기 위해서 0~i-1 까지 **이전** 데이터만 활용하기 !

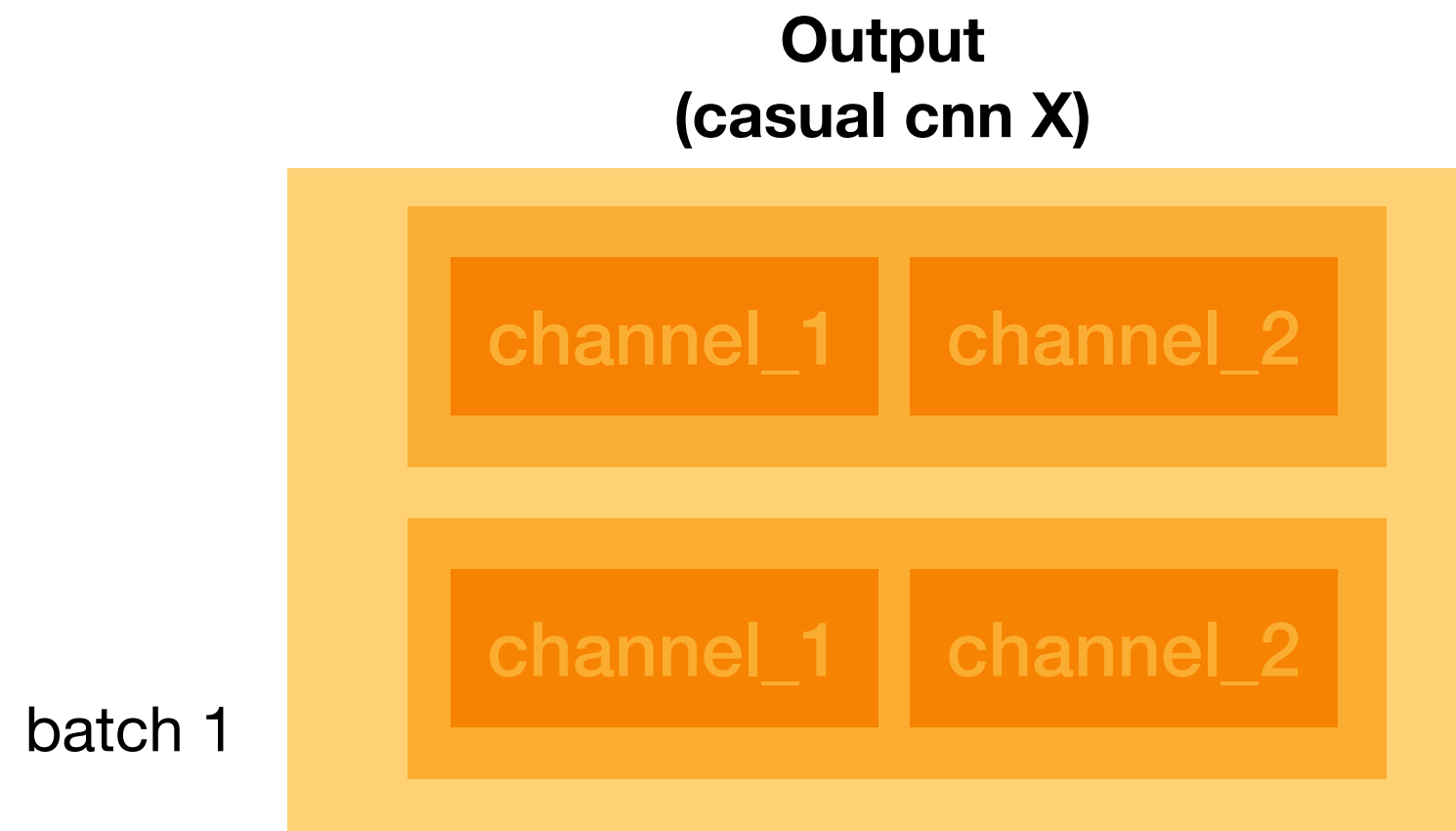
- Casual cnn 을 만들기 위해서는 ?

1) input_length = output_length

=> zero-padding

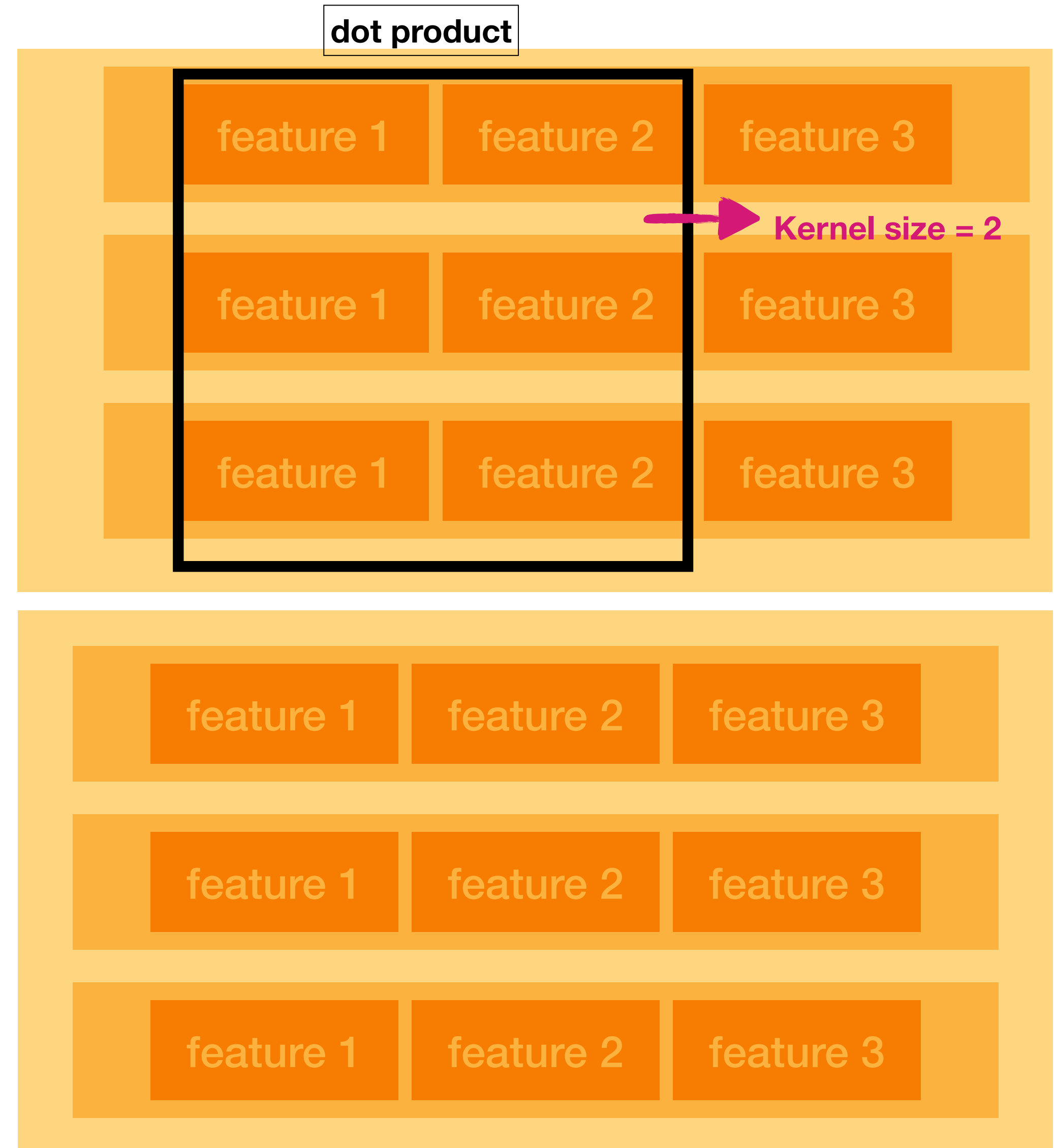
2) No spoilers !

=> zero-padding on the left side



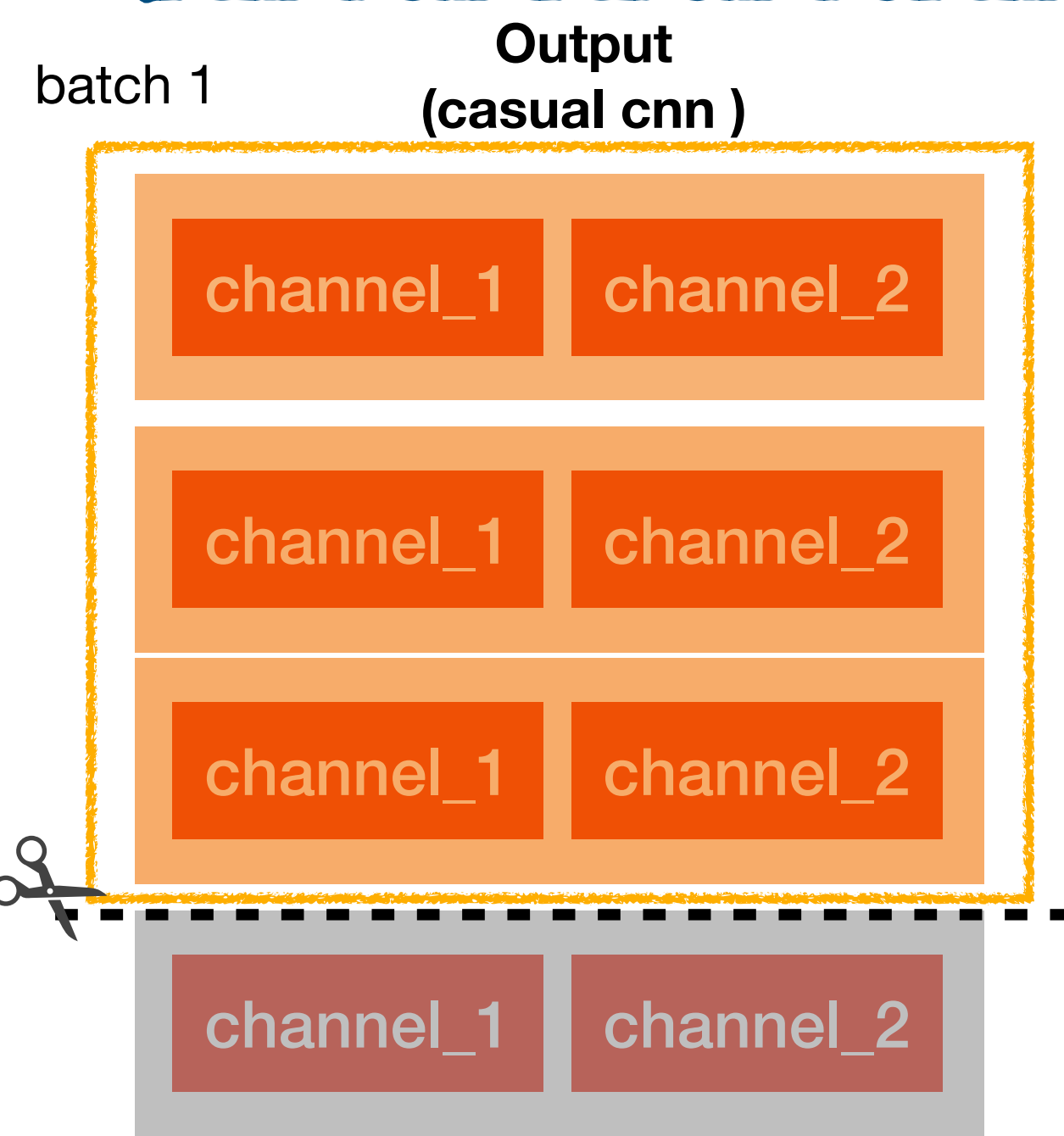
output shape :
(batch_size,output_length,output_size)

zero-padding 이 없으면 input_length != output_length 😬



casual Convolutional Network

- Casual 이란 ? (No spoilers , please ~!)
i번째 output 값을 예측하기 위해서 0~i-1 까지 **이전** 데이터만 활용하기 !
- Casual cnn 을 만들기 위해서는 ?
 - 1) input_length = output_length
=> zero-padding
 - 2) No spoilers !
=> zero-padding on the left side

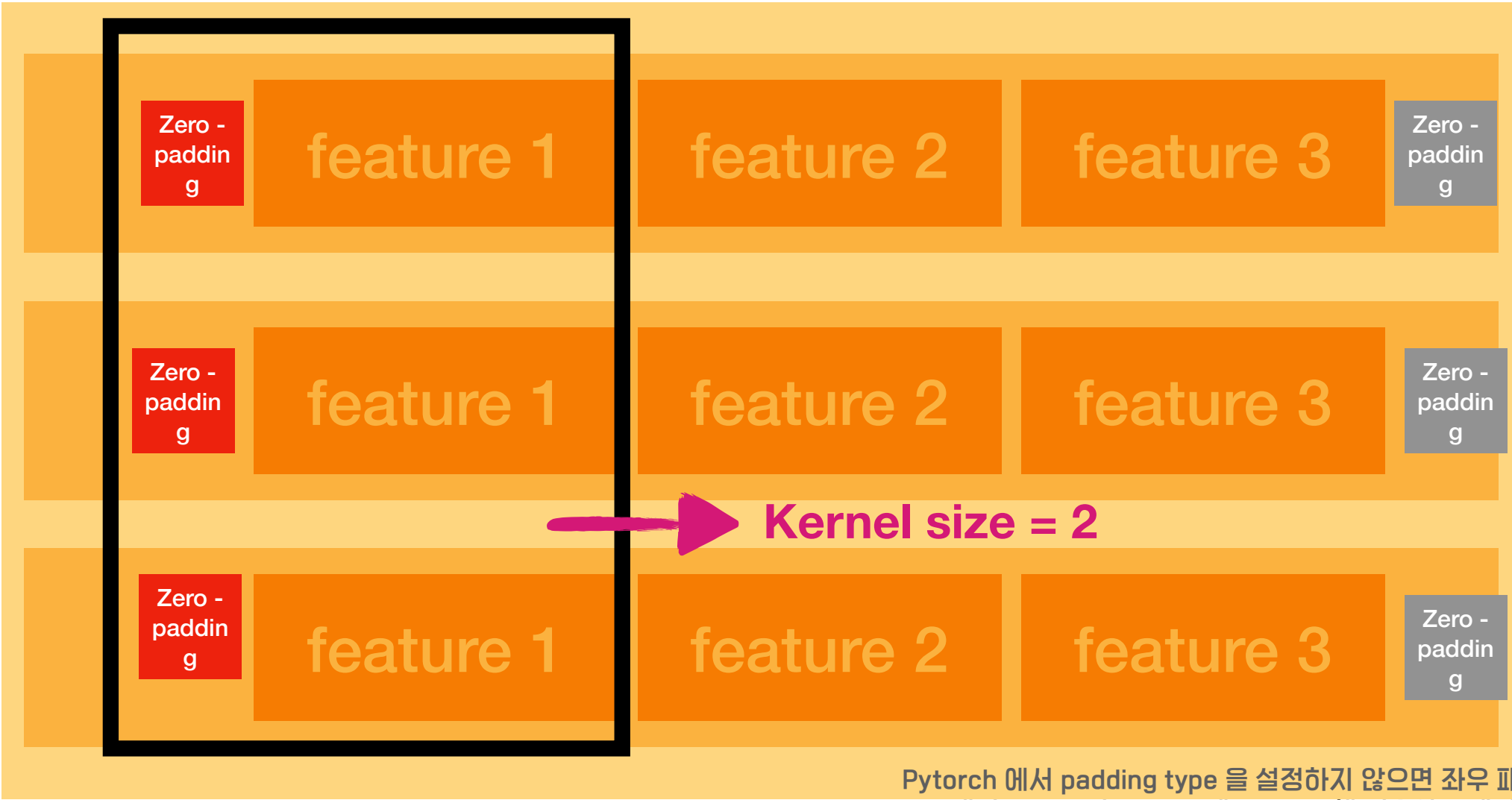
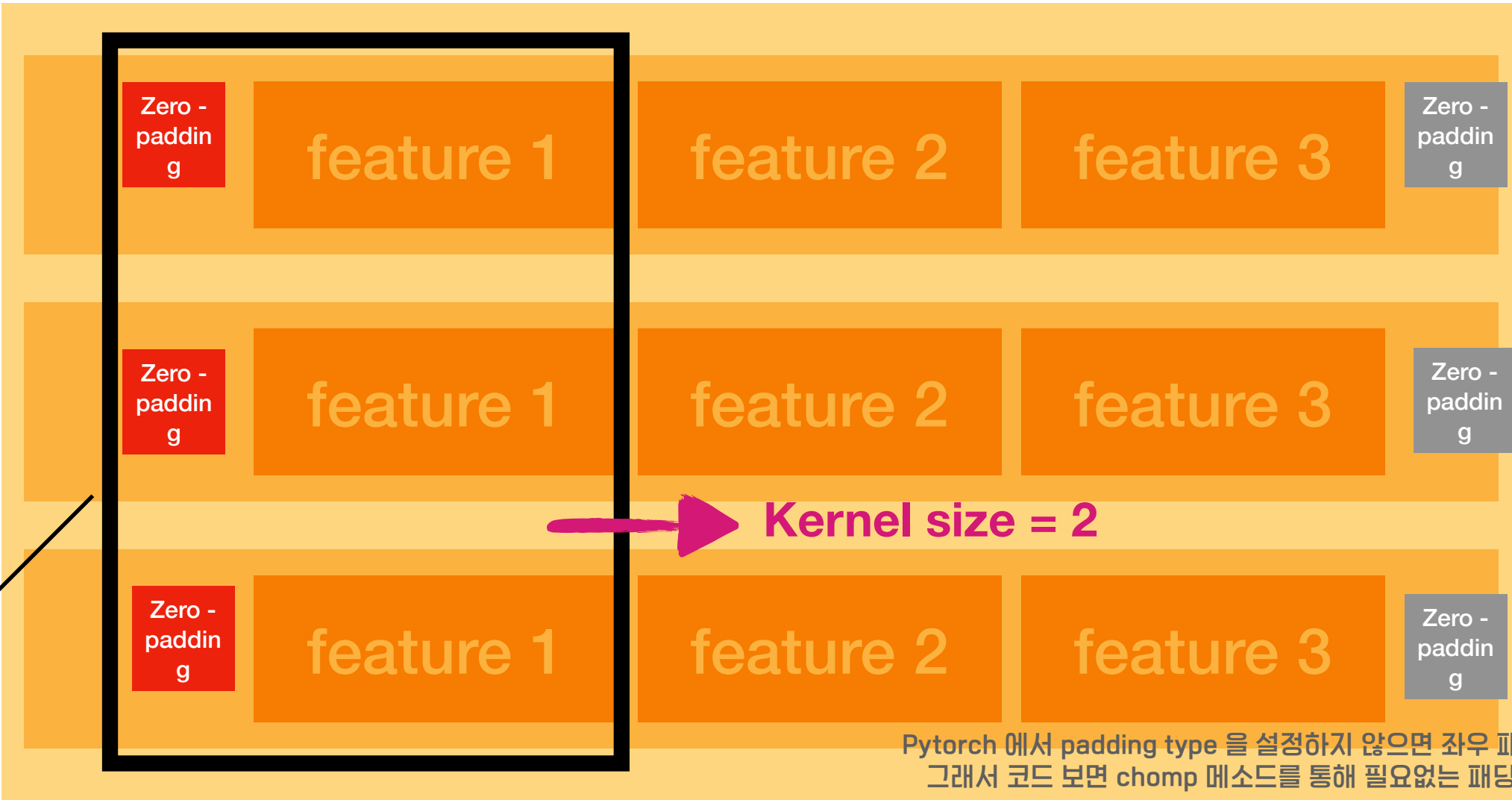


output shape :
(batch_size, **input_length**, output_size)



will chomp this later on !

dot product



input_size and **output_size** might differ since we might not want to forecast every component of the input sequence.

Dilation

receptive field

🚩 let the value of a specific entry in the output depends on all previous entries in the input

(receptive field=input length)

- receptive field : the set of entries of the original input that affect a specific entry of the output

- # of Receptive field formula :

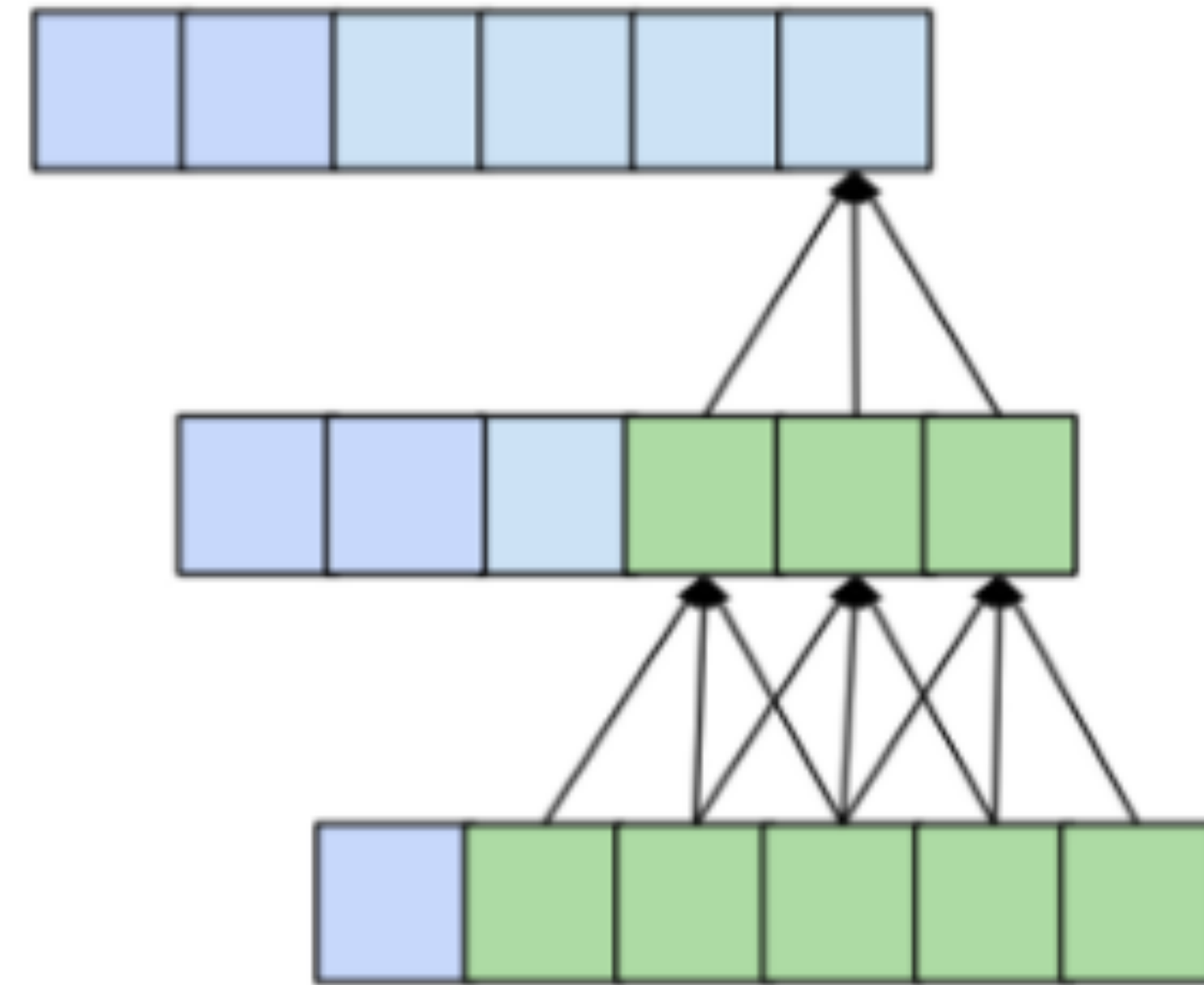
$$r = 1 + n * (k-1)$$

(n ; # of layers, k ; kernel_size)

n 이 1일 경우, 따라서 $r = \text{kernel_size}$

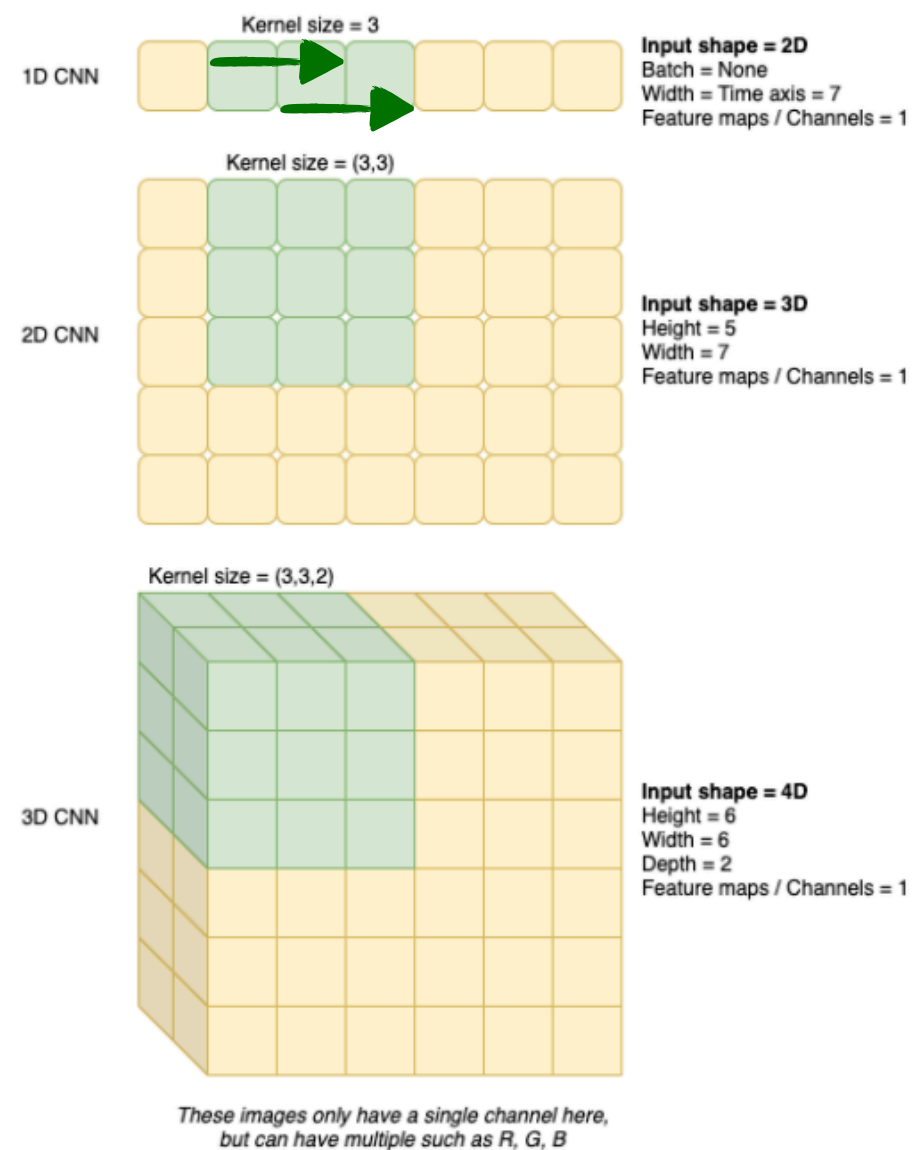
- 따라서, $n = (1 - 1) / (k - 1)$ 이 되어야 함 !

위 공식을 따르면 원하는 receptive field 를 만들기 위해서는
인풋 time step 에 비례해서 layer 수가 기하급수적으로 증가하는 문제가 발생 ! 😬

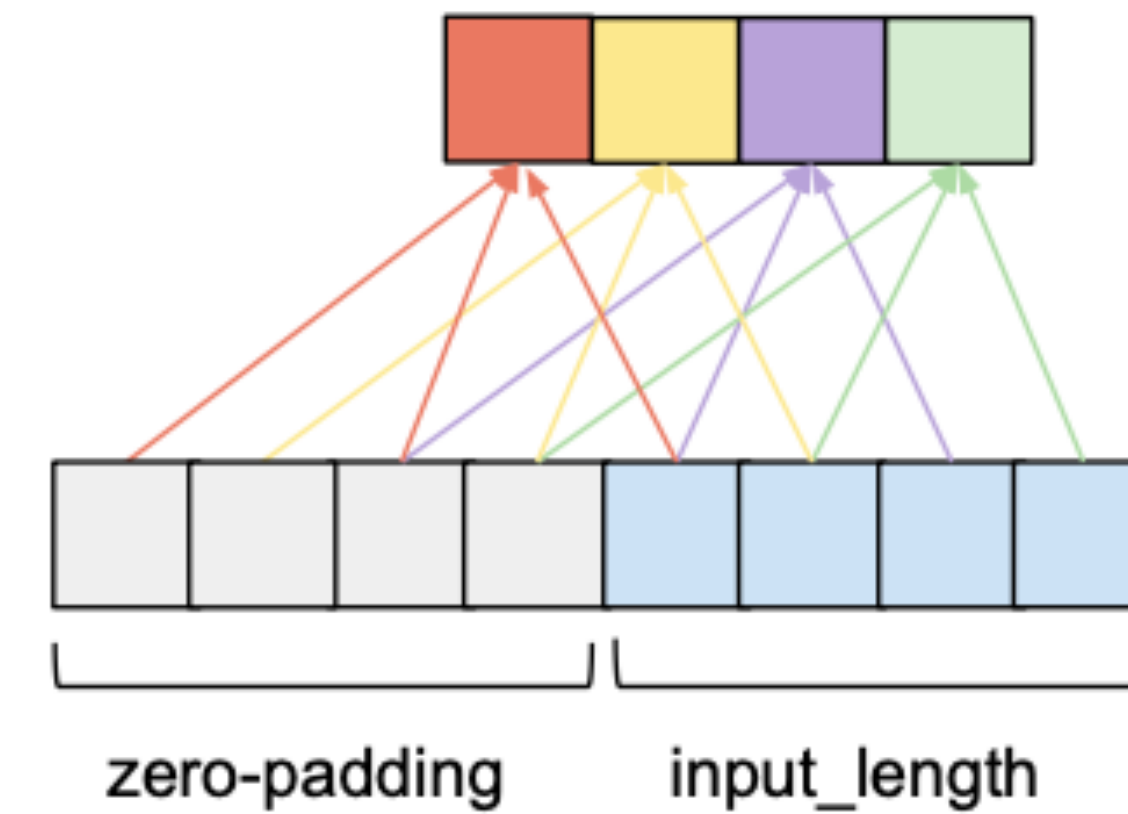


Dilation

- dilation : the distance between the elements of the input sequence that are used to compute one entry of the output sequence



전통적인 CNN에서는 dilation 이 1 !



2-dilated layer, input_length 4 ,kernel_size 3

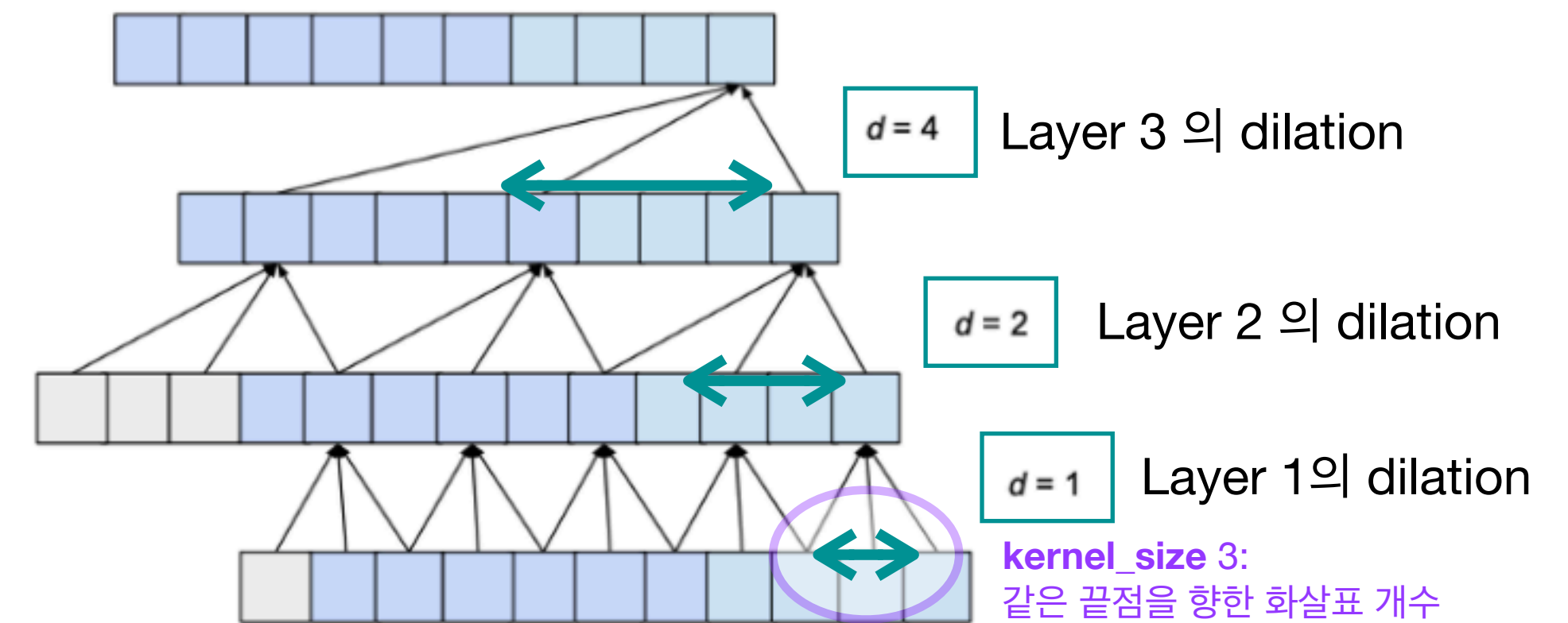
dilation skill (?) 을 쓰면 nn 이 너무 깊어지지 않아도 충분한 receptive_field 를 반영할 수 있지 않을까 ! 🤔

Dilation

- remind ! $r = 1 + n * (k-1)$
- ☆☆ let's increase the value of d exponentially as we move up through the layers !
 $d = b^{**i}$,
 where i is # of layers below the current layer

$$r = 1 + \sum_{i=0}^{n-1} (b^{**i}) * (k-1)$$

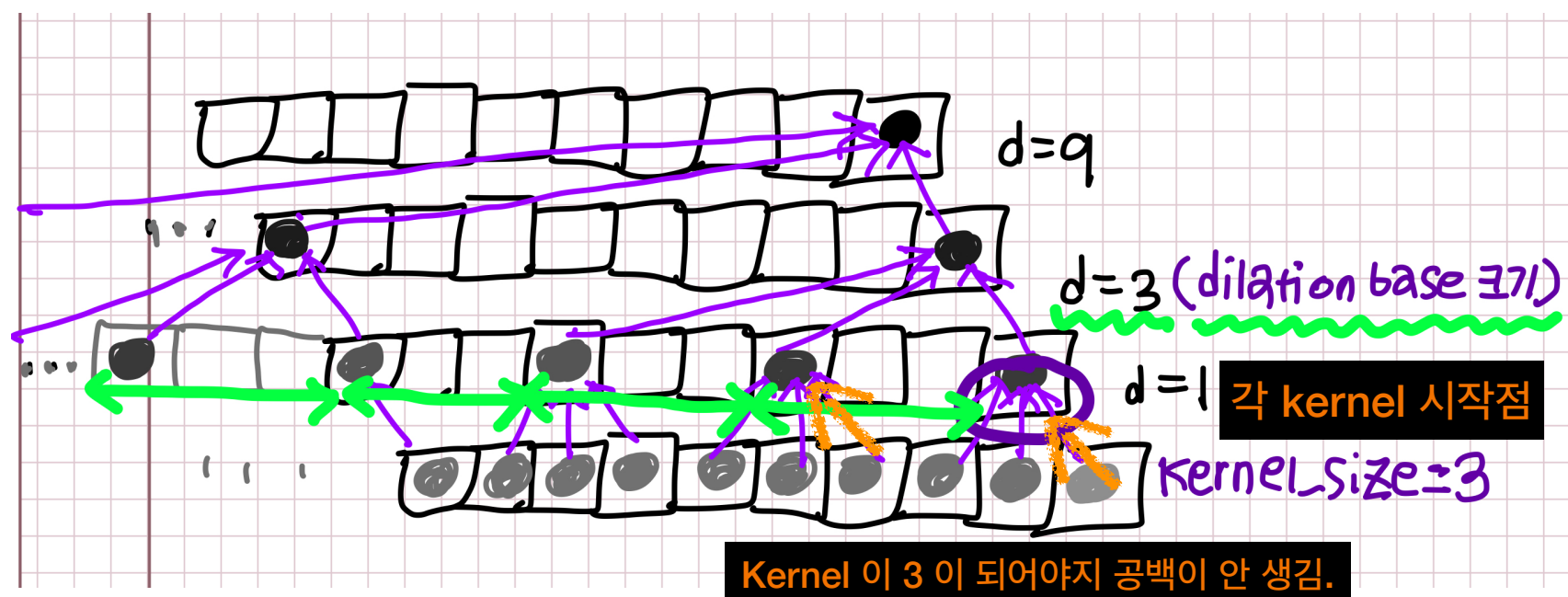
각 layer마다 dilation 정도가 다르기 때문에



input_length 10 ,kernel_size 3, dilation_base 2

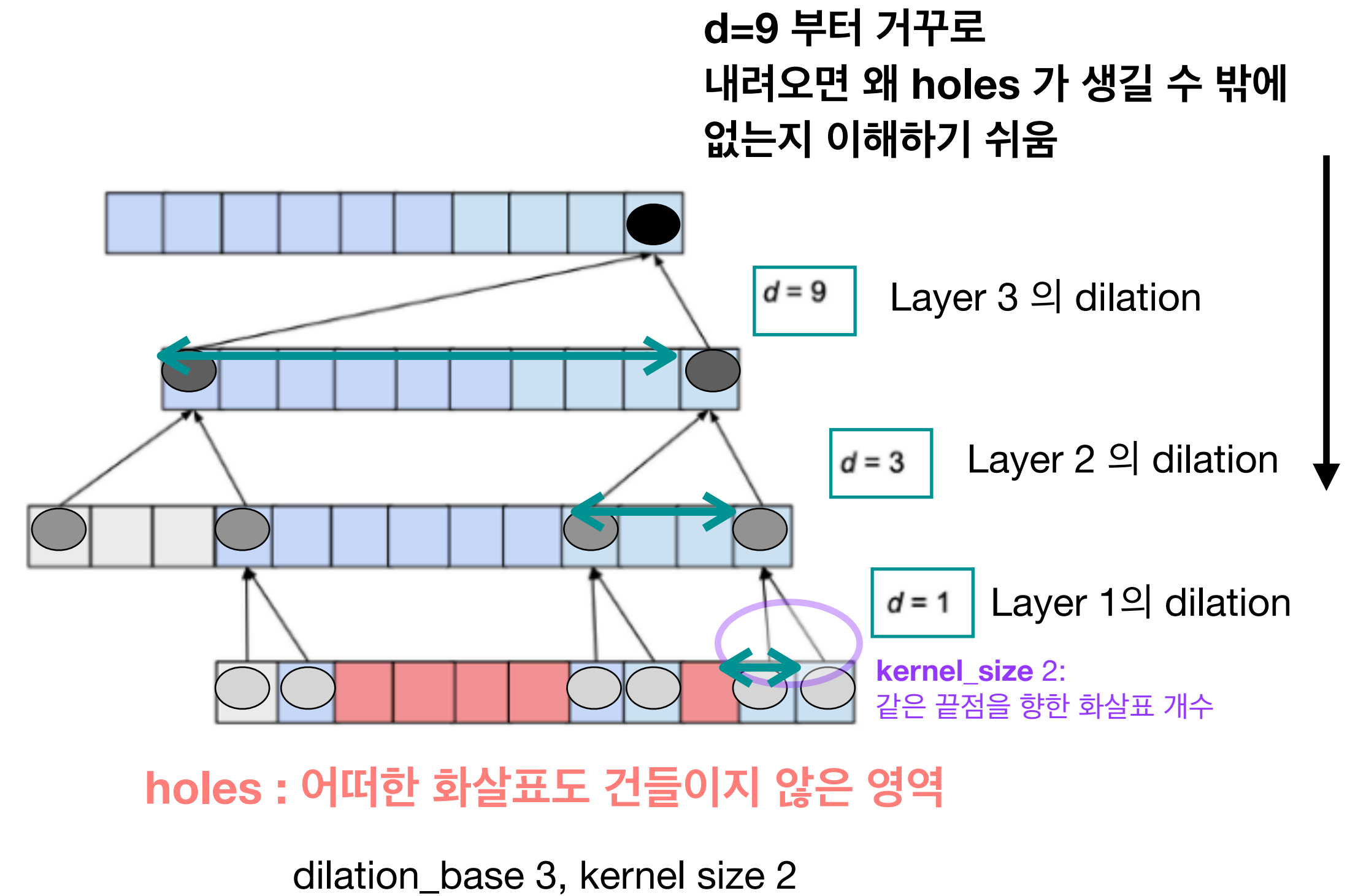
Dilation

- holes : entries in the input sequence that the output value is not dependent on
- holes 현상을 해결하기 위해선 ? **kernel size** 를 늘리거나, dilation base 를 더 작게 하거나
- kernel size 를 늘려서 해결할꺼면 $k \geq b$ 이어야 함 !



이유) 결국 layer 2 의 dilation 은 $b \times 1 = b$ 이므로 dilation base 의 크기가 됨 !
 즉 dilation=1 인 layer 1 에서 hole 이 안 생기려면 kernel size $\geq b$ 여야지
 촌촌해집

VS



정리

- $r \geq \text{input_length}$

Remind

$$r = 1 + \sum_{i=0}^{n-1} (b^{**i}) * (k-1)$$

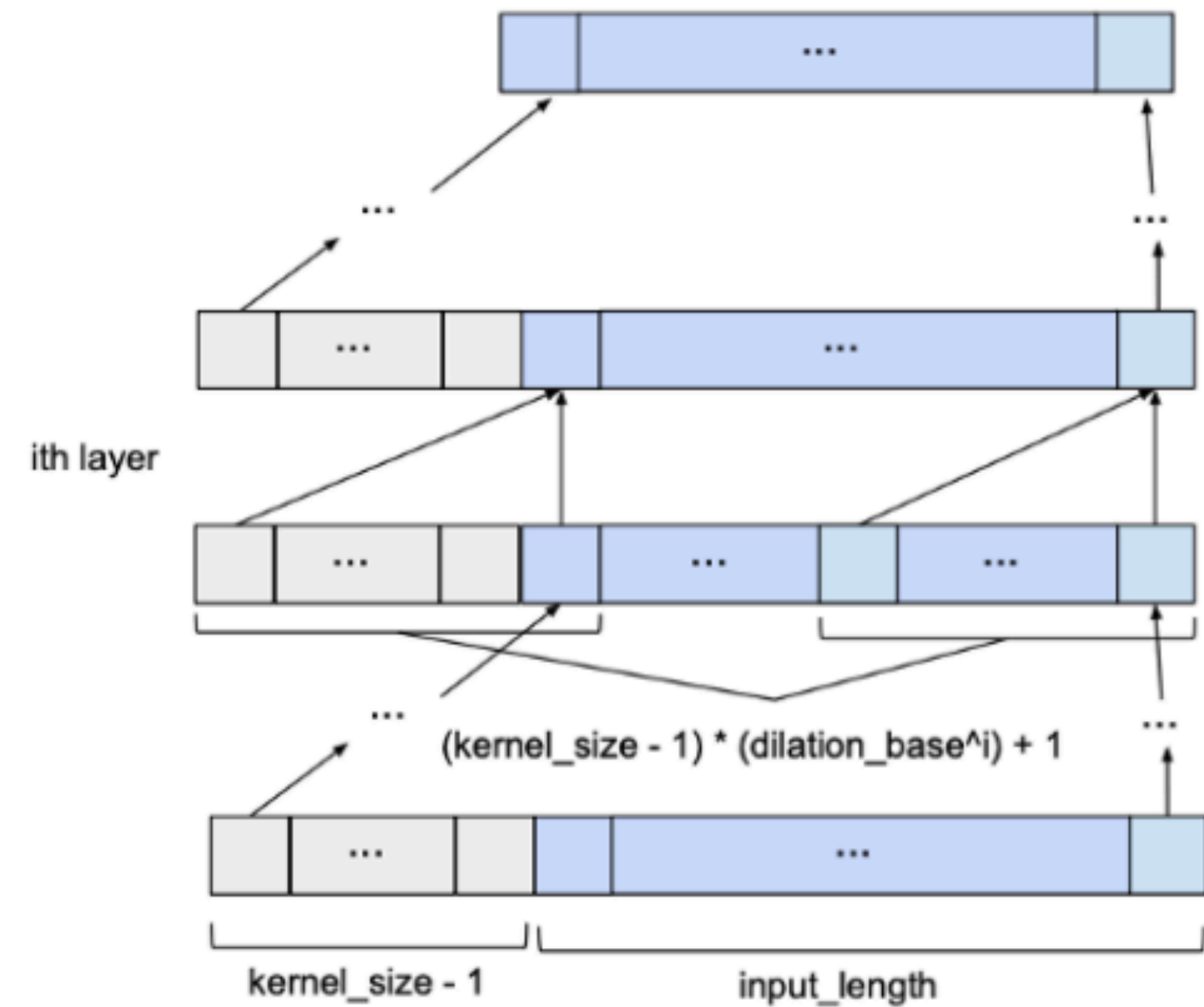
따라서

$$n = \left\lceil \log_b \left(\frac{(l-1) \cdot (b-1)}{(k-1)} + 1 \right) \right\rceil$$

- $k \geq b$

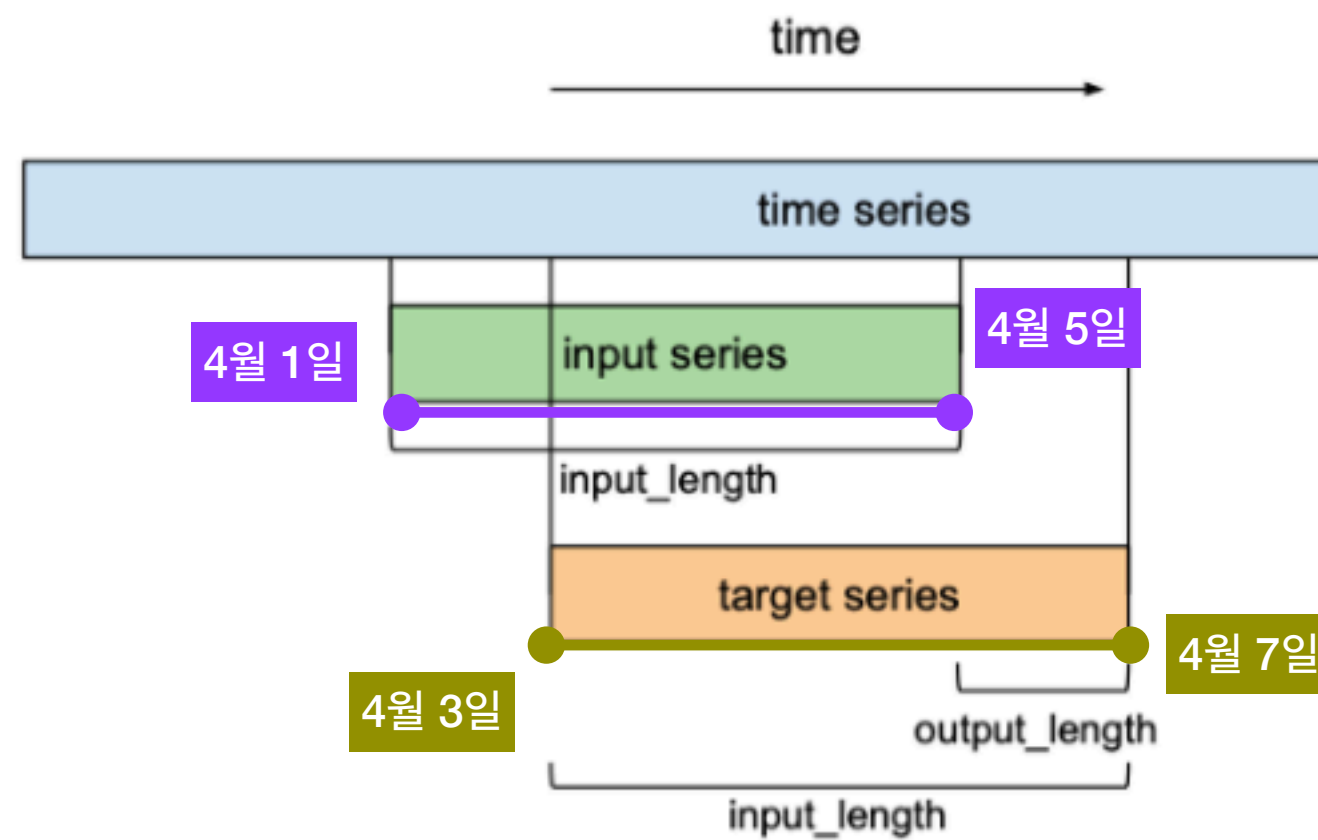
- 각 layer 마다 필요한 zero padding 수

$$p = b^i \cdot (k - 1)$$



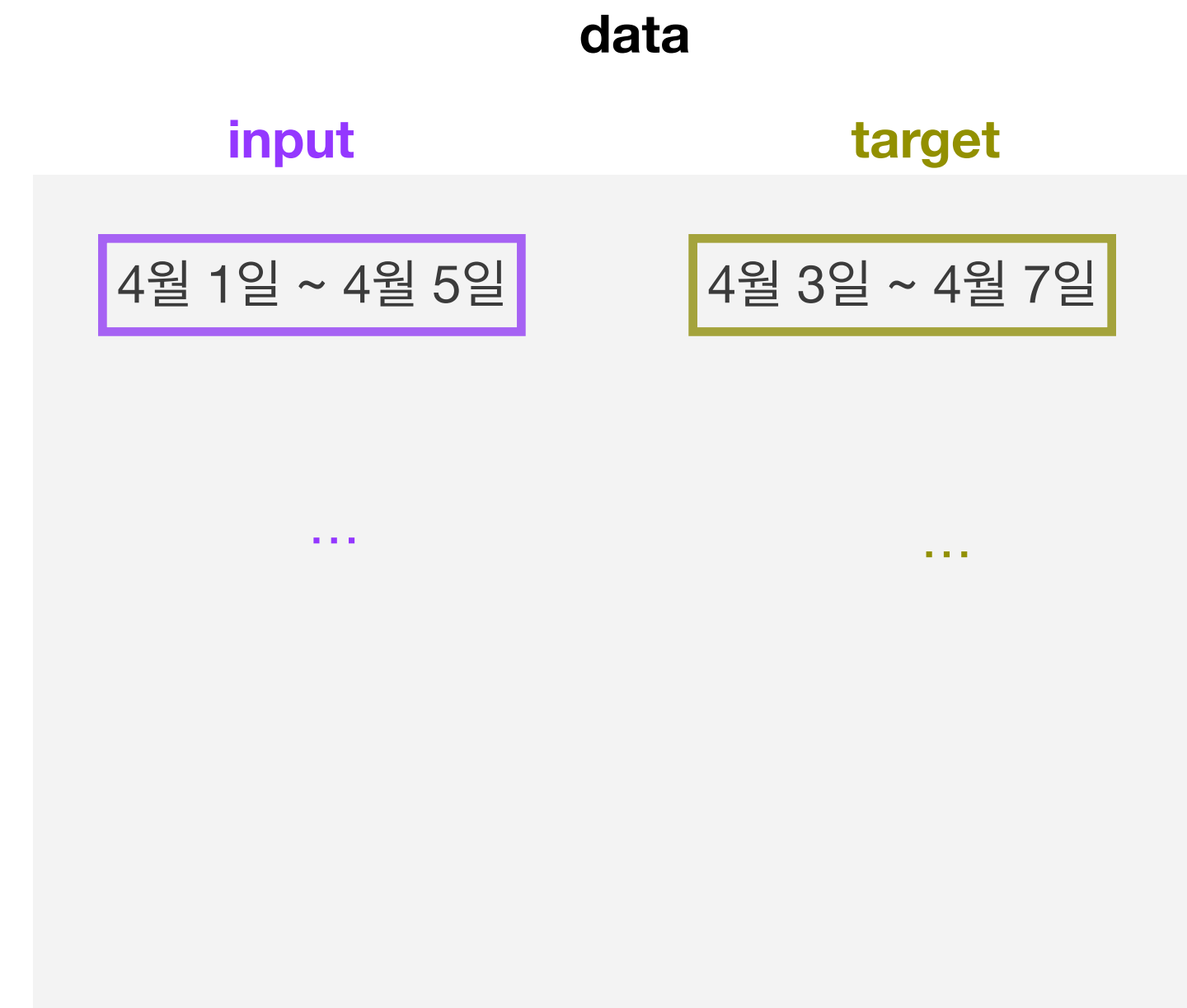
정리

얼마의 timesteps 를 관측한 후 예측하는데 (;input_length) 얼마나 겹치게 할 것인지 (;output_length)



ex. input_length = 5, output_length = 3

4월 1일 ~ 5일 관측 (input series), 4월 3일 ~ 7일 예측



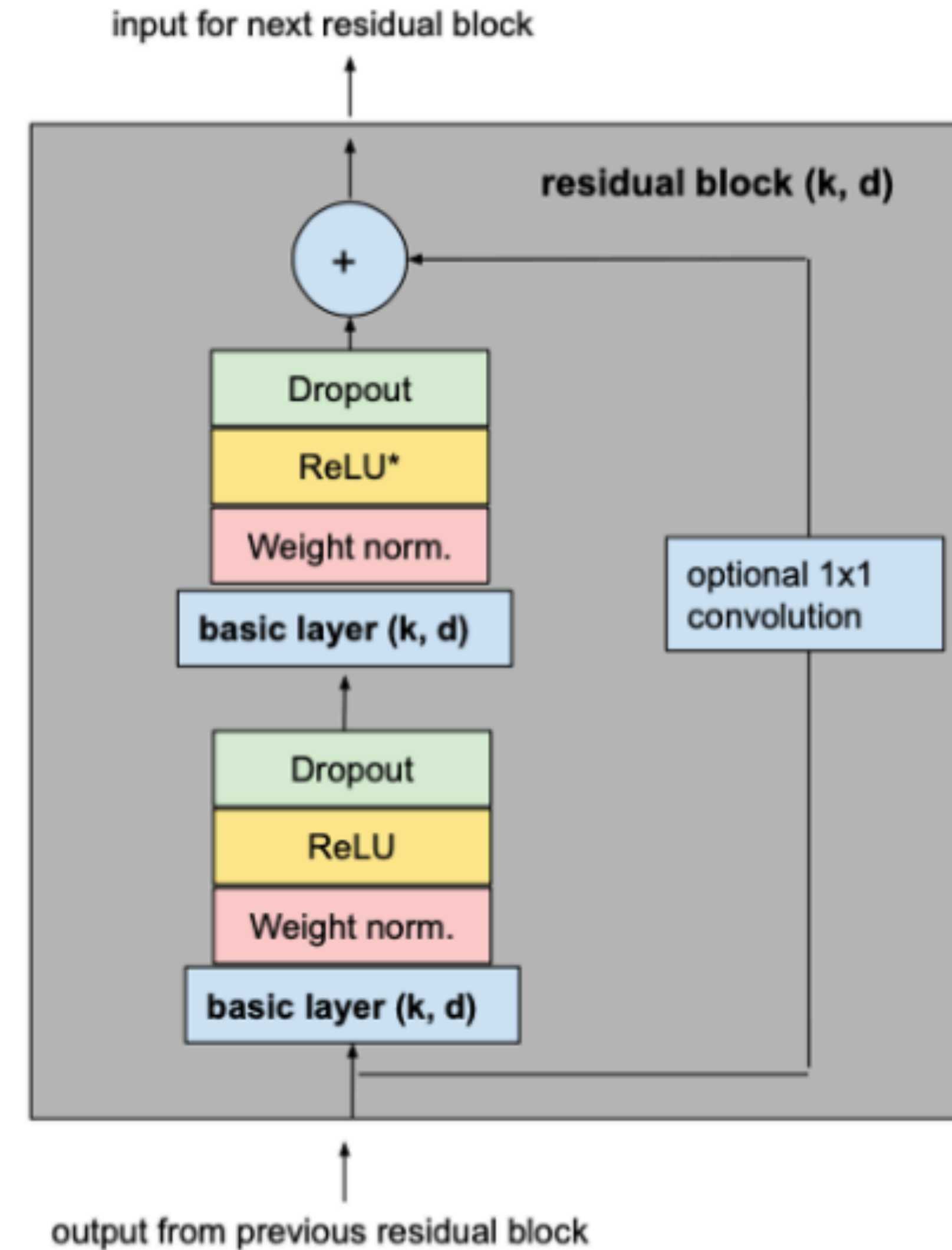
Residual Blocks

Residual Blocks

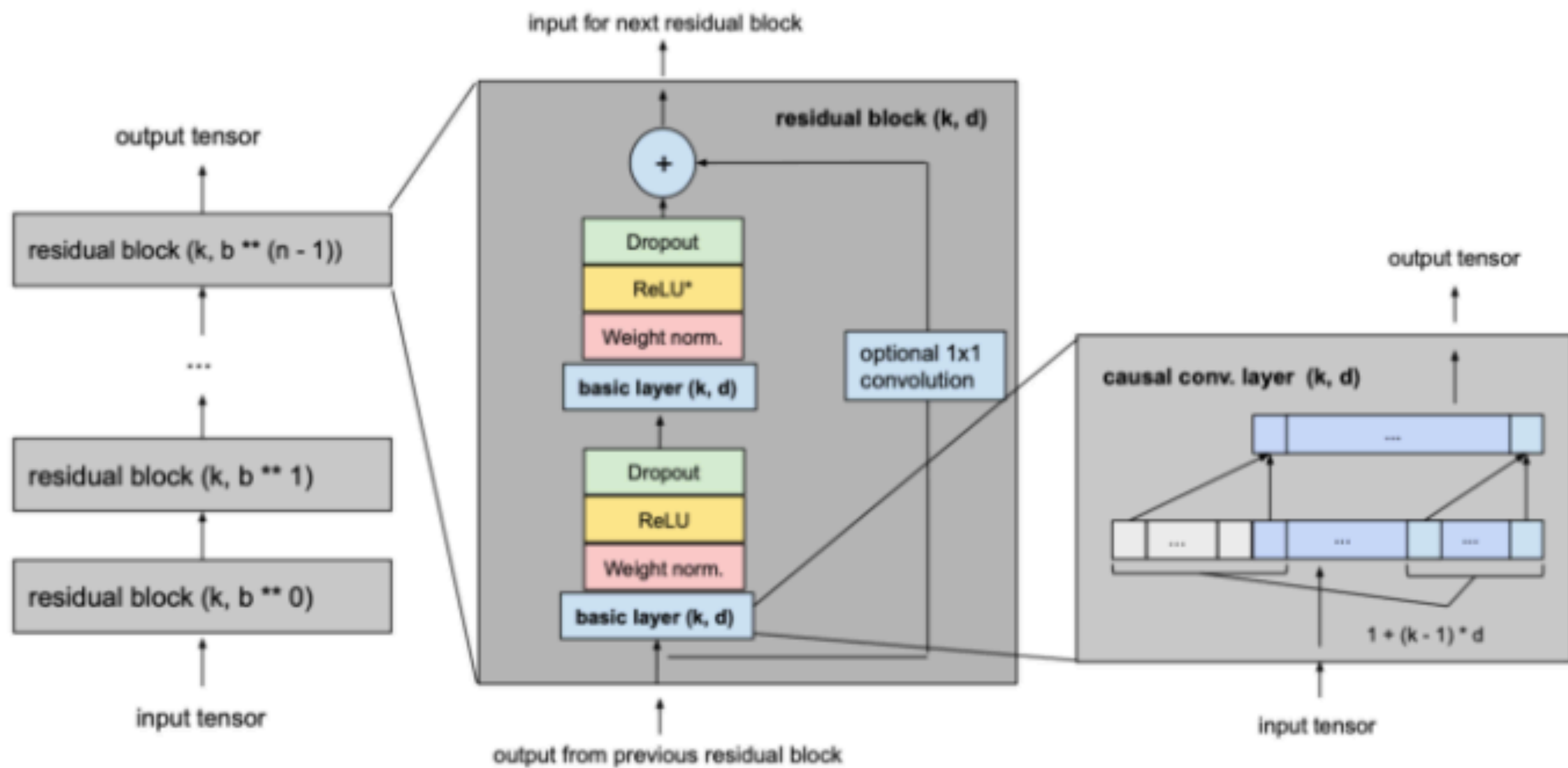
- 2개의 TCN 이 하나의 residual block 을 구성.
따라서 # of receptive fields formula 도 변경 !

$$r = 1 + \sum_{i=0}^{n-1} 2 \cdot (k-1) \cdot b^i = 1 + 2 \cdot (k-1) \cdot \frac{b^n - 1}{b - 1} \quad n = \left\lceil \log_b \left(\frac{(l-1) \cdot (b-1)}{(k-1) \cdot 2} + 1 \right) \right\rceil$$

하나의 residual block 이 하나의 layer 처럼 취급되어서 쌓아짐



전체 모델 구조



성능 및 결론

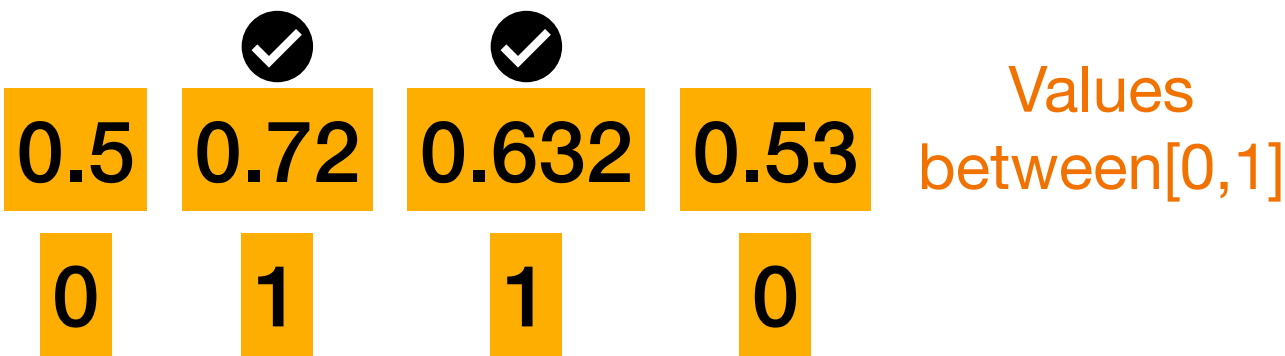
성능 평가

데이터 셋 설명

- PTB : 구문 주석 말뭉치인 트리뱅크 중 가장 유명한 데이터셋으로 400어절 규모로 이루어져 있으며, 주로 월 스트리트 저널의 문장들로 이루어져 있음
- Wikitext-103 : 위키피디아에서 1억개 이상의 토큰을 추출한 데이터 셋으로 PTB 보다 110배 더 큰 데이터 셋
- Text8 : text8 또한 위키피디아에서 추출한 데이터 셋으로 PTB 보다 약 20배 더 큰 데이터 셋

성능 평가

Task 설명



The adding problem

Sum the two random values whose second dimensions are marked by 1



First 10 numbers chosen between 1,2..8

Copy memory

Output the same length that is zero everywhere except the last 10 values after the delimiter
Repeat the 10 values it encountered at the start of the input,

(1)	<i>Context:</i> “Yes, I thought I was going to lose the baby.” “I was scared too,” he stated, sincerity flooding his eyes. “You were ?” “Yes, of course. Why do you even ask?” “This baby wasn’t exactly planned for.” <i>Target sentence:</i> “Do you honestly think that I would want you to have a ____ ?” <i>Target word:</i> miscarriage
(2)	<i>Context:</i> “Why?” “I would have thought you’d find him rather dry,” she said. “I don’t know about that,” said <u>Gabriel</u> . “He was a great craftsman,” said Heather. “That he was,” said Flannery. <i>Target sentence:</i> “And Polish, to boot,” said _____. <i>Target word:</i> Gabriel

LAMBDA

with an average of 4.6 sentences as context, and 1 target sentence the last word of which is to be predicted.

Language Modeling Task : PennTreebank, Wikitext-103, text8

성능 평가

Sequence Modeling Task	Model Size (\approx)	Models			
		LSTM	GRU	RNN	TCN
Seq. MNIST (accuracy ^h)	70K	87.2	96.2	21.5	99.0
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	97.2
Adding problem $T=600$ (loss ^ℓ)	70K	0.164	5.3e-5	0.177	5.8e-5
Copy memory $T=1000$ (loss)	16K	0.0204	0.0197	0.0202	3.5e-5
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	8.10
Music Nottingham (loss)	1M	3.29	3.46	4.05	3.07
Word-level PTB (perplexity ^ℓ)	13M	78.93	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	45.19
Word-level LAMBADA (perplexity)	-	4186	-	14725	1279
Char-level PTB (bpc ^ℓ)	3M	1.36	1.37	1.48	1.31
Char-level text8 (bpc)	5M	1.50	1.53	1.69	1.45

느낀 점

- CNN 이 sequential task 에 개입될 여지가 없다고 생각했는데 결과가 놀라웠음
- TCN 모델 구조는 복잡하지 않으면서도 RNN 계열의 단점을 극복할 수 있게 CNN 을 변형했다는 점에서 매우 훌륭하다고 생각
- TCN 계산에 수학이 관여한 것을 보고 나도 수학을 좀 더 자유자재로 연구에 쓰려면 더욱 분발해야겠다는 생각이 듦
- 앞으로 forecasting 모델 트렌드는 어떤 모델이 RNN + LSTM 의 단점 (한계) 를 뛰어넘을 수 있을까 인 것 같음 !