딥러닝을 활용한 자연어 처리(3)

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

09

APPLICATION

ARTIFICIAL INTELLIGENCE

Notice

THINKING

생각해보기

✓ Convolution을 활용한 텍스트 분류는 어떻게 할까요?

학습목표

GOALS

1 Convolution 개요와 Convolution에 대한 작동 방식에 대해 이해하고 설명할 수 있다.

2 고차원에서의 Convolution을 이해하고 설명할 수 있다.

3 1D Conv , 2D Conv에 대해 이해하고 설명할 수 있다.

4 Max-pooling와 Dense Layer에 대해 설명할 수 있다.

5 자연어 처리에서 어떻게 1D Conv을 사용하는지 파악할 수 있다.

6 IMDB 데이터셋을 활용한 자연어 처리 실습을 할 수 있다.

1 CNN 개요

2 자연어 처리에서의 CNN

3 실습

CONTENTS

학습내용

APPLICATION

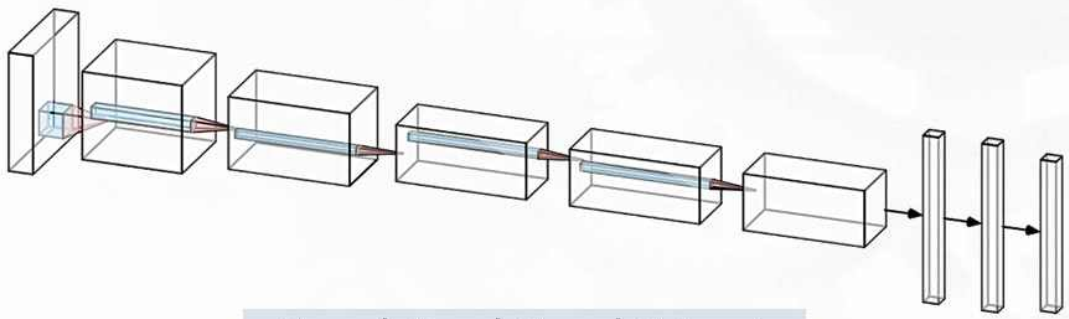
01 CNN 개요

01 Convolution 개요

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Convolution Neural Network(이하 CNN)은

"동물의 시각 신경망과 유사한 점이 많음"



Convolutional Neural Network

출처: http://alexlenail.me/NN-SVG/AlexNet.html

CNN을 구성하는 가장 중요한 요소

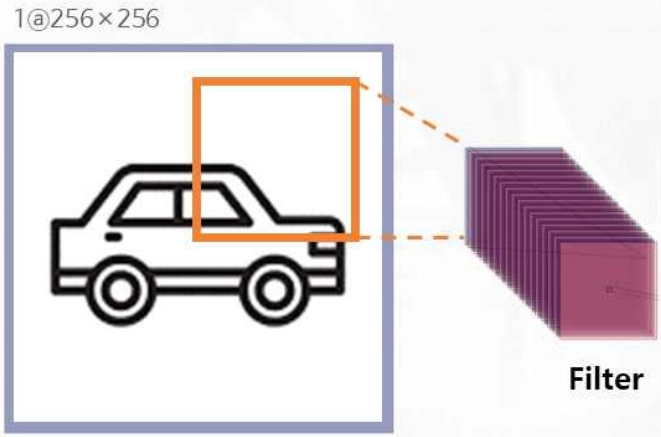
Filter
개념

Convolution
연산

01 Convolution 개요

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

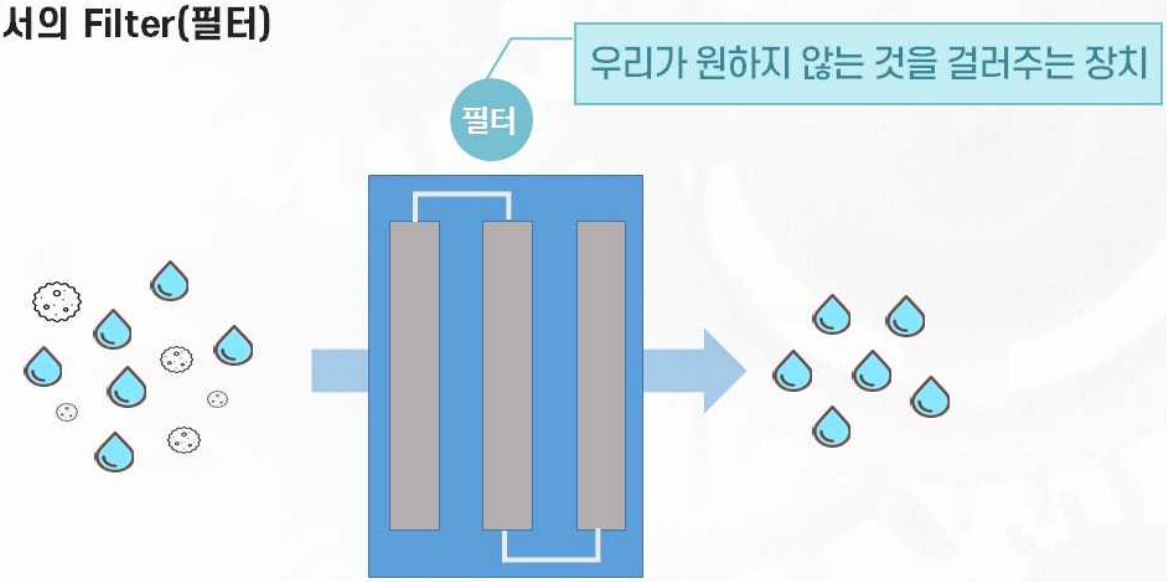
CNN을 구성하는 가장 중요한 요소



Filter와 Convolution 연산

출처: 퍼블릭에이아이(www.publicai.co.kr)

CNN에서의 Filter(필터)



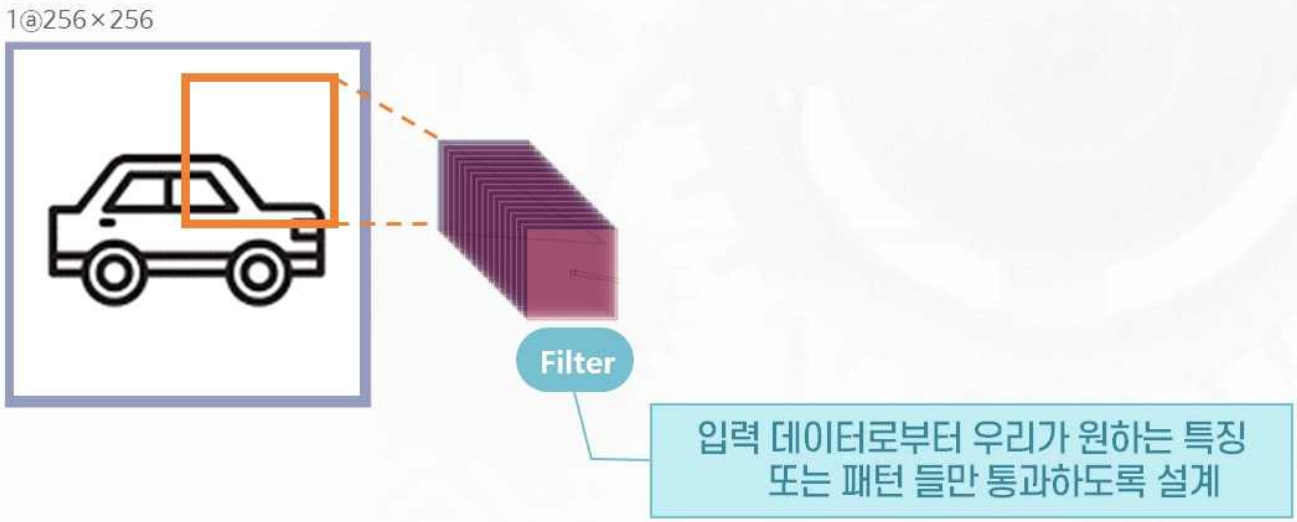
일반적인 필터의 개념

출처: 퍼블릭에이아이(www.publicai.co.kr)

01 Convolution 개요

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

CNN에서의 Filter(필터)



CNN에서의 filter 개념

출처: 퍼블릭에이아이(www.publicai.co.kr)

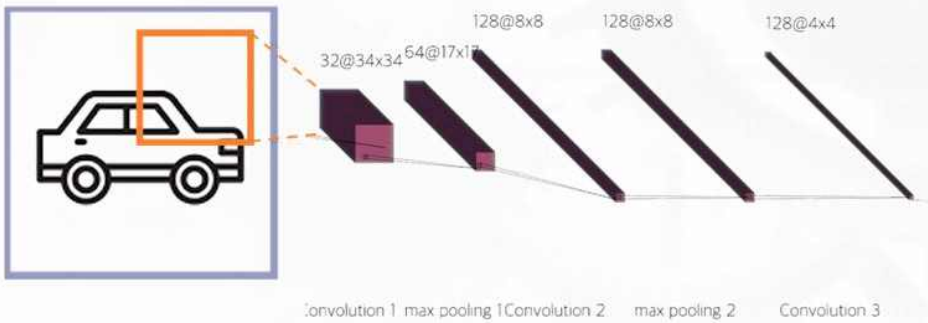
CNN에서의 Convolution 연산(합성곱 연산)

- 합성곱 연산과 필터의 역할

출처: 퍼블릭에이아이(www.publicai.co.kr)

인공신경망과 동물이 사물을 인식하는 방법

- 실제로 CNN 인공신경망의 경우 동물이 사물을 인식하는 방법과 같이 겹겹이 층으로 쌓여 동작함

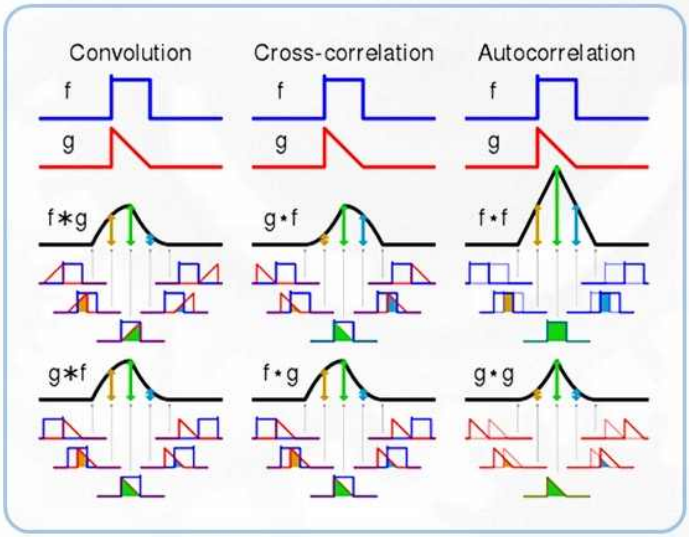


Convolutional Neural Network

출처: 퍼블릭에이아이(www.publicai.co.kr)

CNN에서의 Convolution 연산(합성곱 연산)

- 합성곱(合成-, convolution, 콘벌루션)
  : 하나의 함수와 또 다른 함수를 반전
  이동한 값을 곱한 다음, 구간에 대해
  적분하여 새로운 함수를 구하는 수학
  연산자

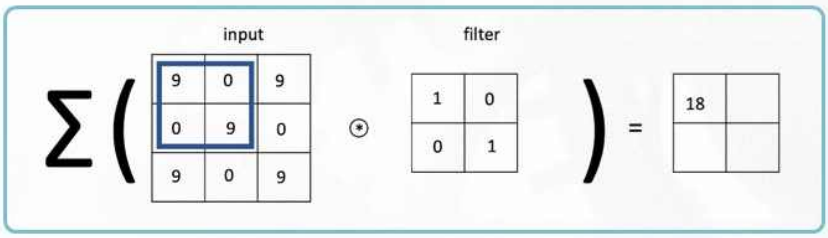


출처: 퍼블릭에이아이(www.publicai.co.kr)

02 Convolution 에 대한 작동 방식

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

CNN에서의 Convolution 연산(합성곱 연산)



Window : 입력 데이터 내 filter와 FMA 연산이 되는 부분

FMA : Fused Multiply Add (단일 곱셈 누산기)

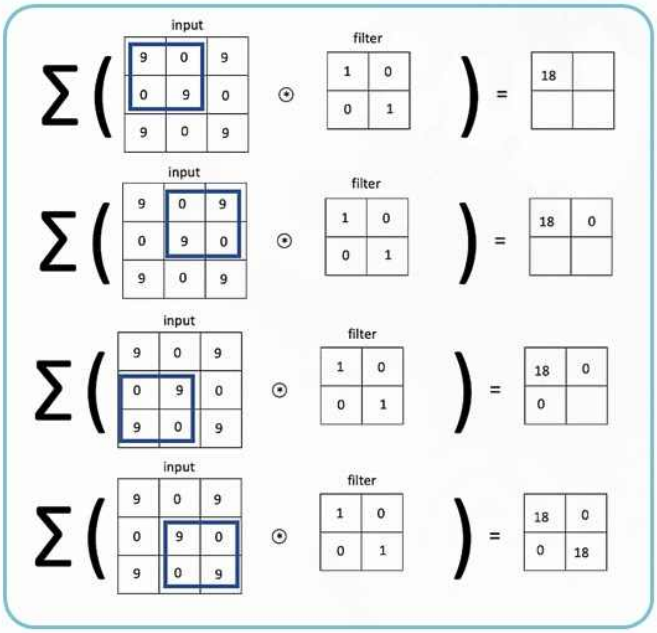
① Element-wise multiply

② Summation

1x1 + 1x0 + 0x1 = 1

출처: 퍼블릭에이아이(www.publicai.co.kr)

CNN에서의 Convolution 연산(합성곱 연산)



출처: 퍼블릭에이아이(www.publicai.co.kr)

→ Window가 움직이는 크기도 지정할 수 있음

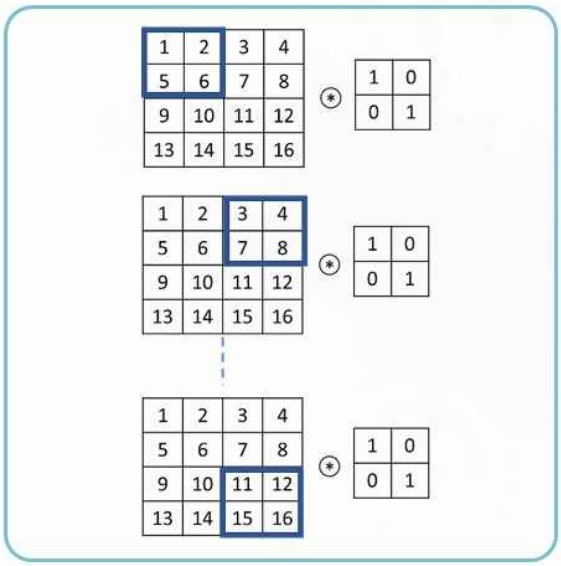
→ 왼쪽 그림에는 Stride 가 1,1로 지정되어 있어 오른쪽으로 한 칸 그리고 아래쪽으로 한 칸 이동하지만 Stride을 2, 2로 지정하면 2칸씩 건너 뛸 수도 있음

02 Convolution 에 대한 작동 방식

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

CNN에서의 Convolution 연산(합성곱 연산)

- Stride 가 2,2 인 Convolution Layer



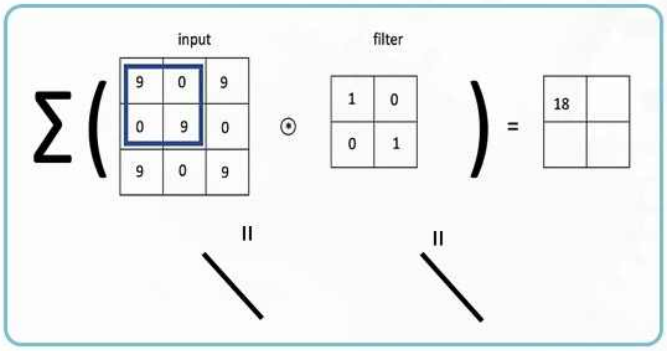
출처: 퍼블릭에이아이(www.publicai.co.kr)

→ Window가 움직이는 크기도 지정할 수 있음

→ 왼쪽 그림에는 Stride 가 1,1로 지정되어 있어 오른쪽으로 한 칸 그리고 아래쪽으로 한 칸 이동하지만 Stride을 2, 2로 지정하면 2칸씩 건너 뛸 수도 있음

CNN에서의 Convolution 연산(합성곱 연산)

- 이미지의 특징을 추출하는 방법



출처: 퍼블릭에이아이(www.publicai.co.kr)

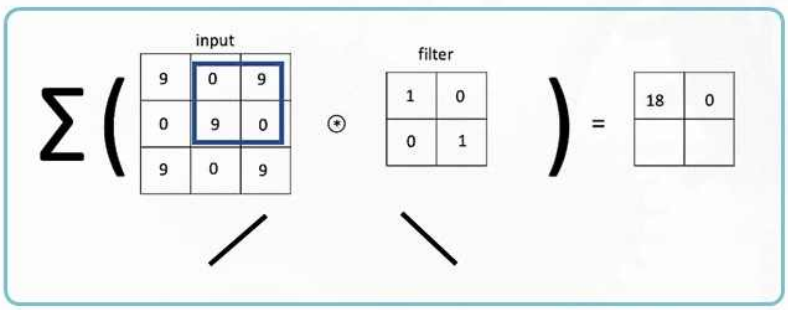
→ Filter와 window와 유사하면 출력값은 크게 되고 Filter와 window가 유사하지 않으면 상대적으로 적은 값을 가지게 됨

02 Convolution 에 대한 작동 방식

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

CNN에서의 Convolution 연산(합성곱 연산)

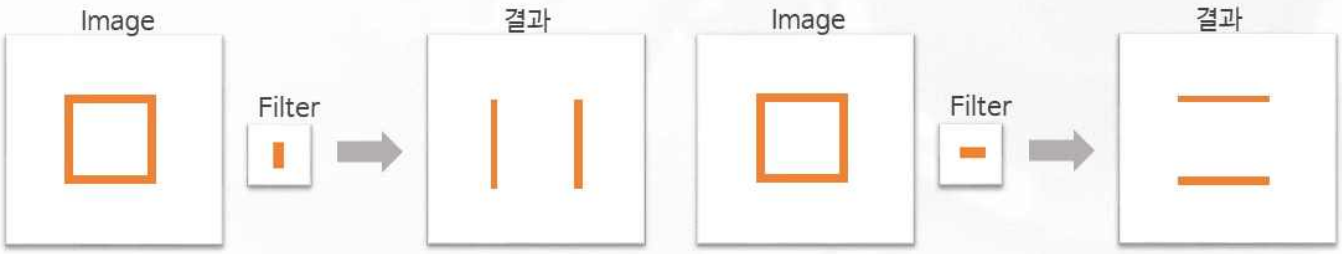
- 이동한 Window내의 특징과 필터의 특징



출처: 퍼블릭에이아이(www.publicai.co.kr)

CNN에서의 Convolution 연산(합성곱 연산)

- 합성곱 연산과 필터의 역할

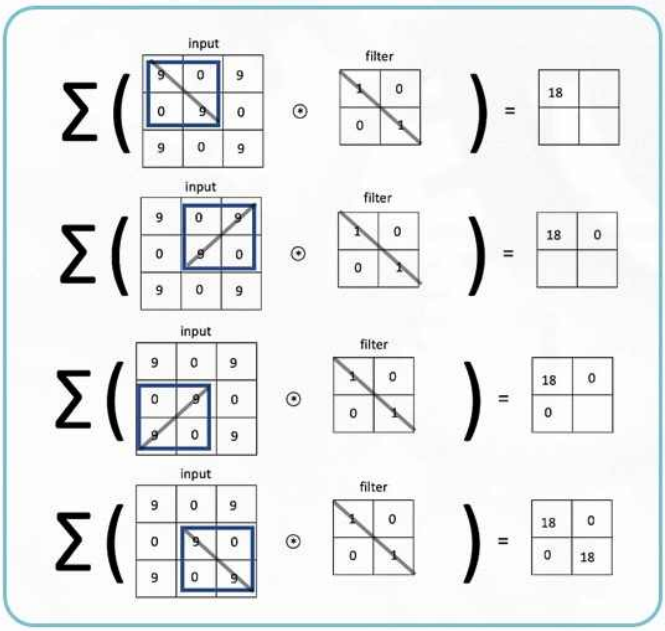


출처: 퍼블릭에이아이(www.publicai.co.kr)

02 Convolution 에 대한 작동 방식

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

CNN에서의 Convolution 연산(합성곱 연산)



출처: 퍼블릭에이아이(www.publicai.co.kr)

02 Convolution 에 대한 작동 방식

Tensorflow로 따라 해보기

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.layers import Conv2D, Input, Dense
from tensorflow.keras.models import Model
```

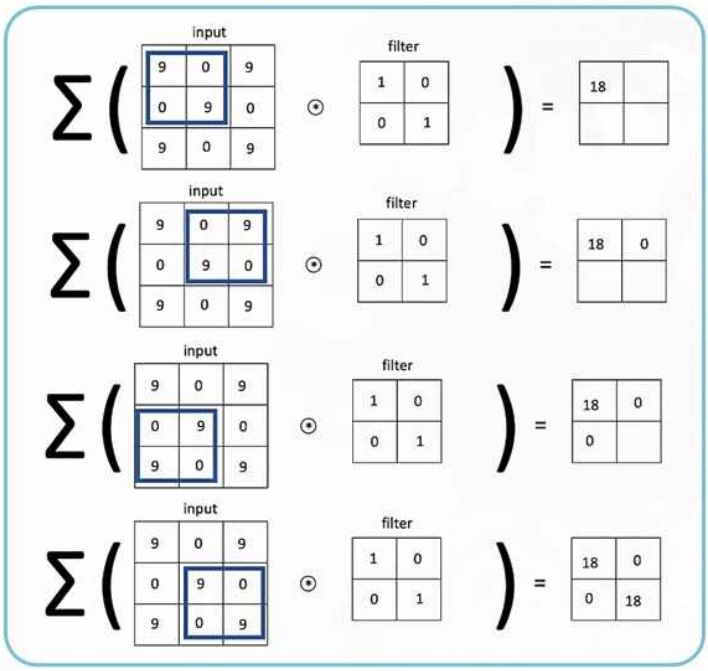
```
# 입력 백터를 설정합니다.
input_array = np.array(
    [[9, 0, 9],
     [0, 9, 0],
     [9, 0, 9]]
)
input_array = input_array.reshape([1,3,3,1])

# convolution 에 사용할 filter 입니다.
filter_ = np.array([1, 0, 0, 1]).reshape([2, 2, 1, 1])
```

```
input_tensor = Input(shape=[3,3,1])
layer = Conv2D(filters=1, kernel_size=2, use_bias=False)(input_tensor)

model = Model(input_tensor, layer)
model.layers[1].set_weights([filter_])
```

```
model.predict(input_array)
```

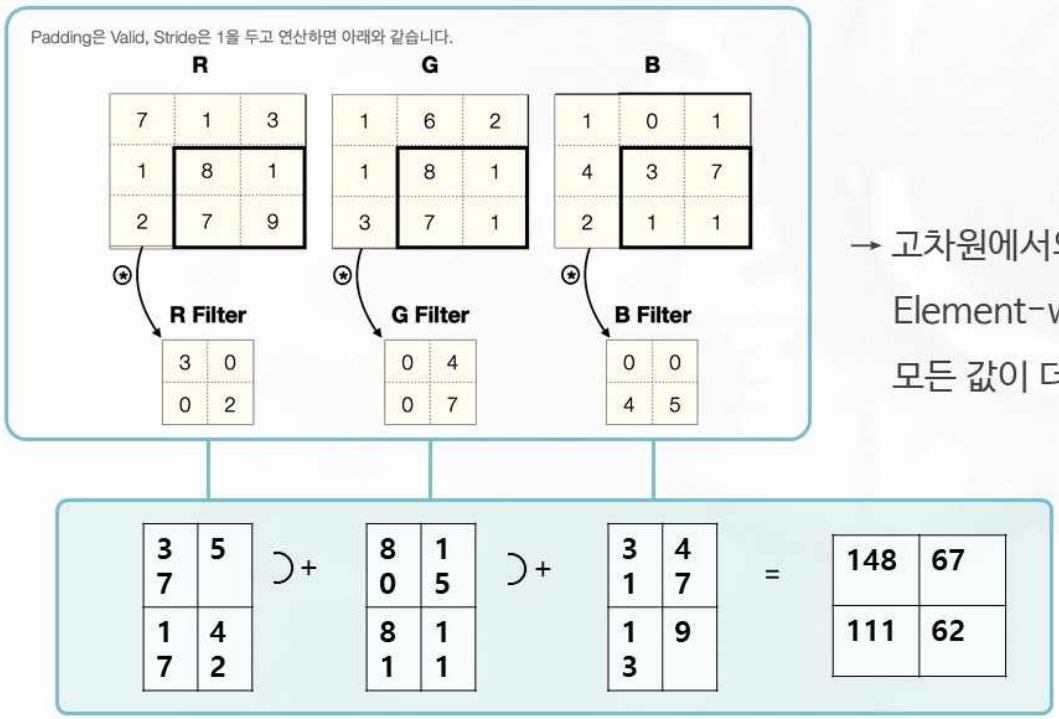
```
array([[[[18.],
         [ 0.]],

        [[ 0.],
         [18.]]]], dtype=float32)
```


03 고차원에서의 Convolution



출처: 퍼블릭에이아이(www.publicai.co.kr)

→ 2차원 입력 영상 Convolution에 이어서 3차원 이상의 입력 데이터에 대해서도 Convolution을 어떻게 수행해야 하는지 학습함



→ 고차원에서의 Convolution도 Element-wise multiply 후에 모든 값이 더해짐

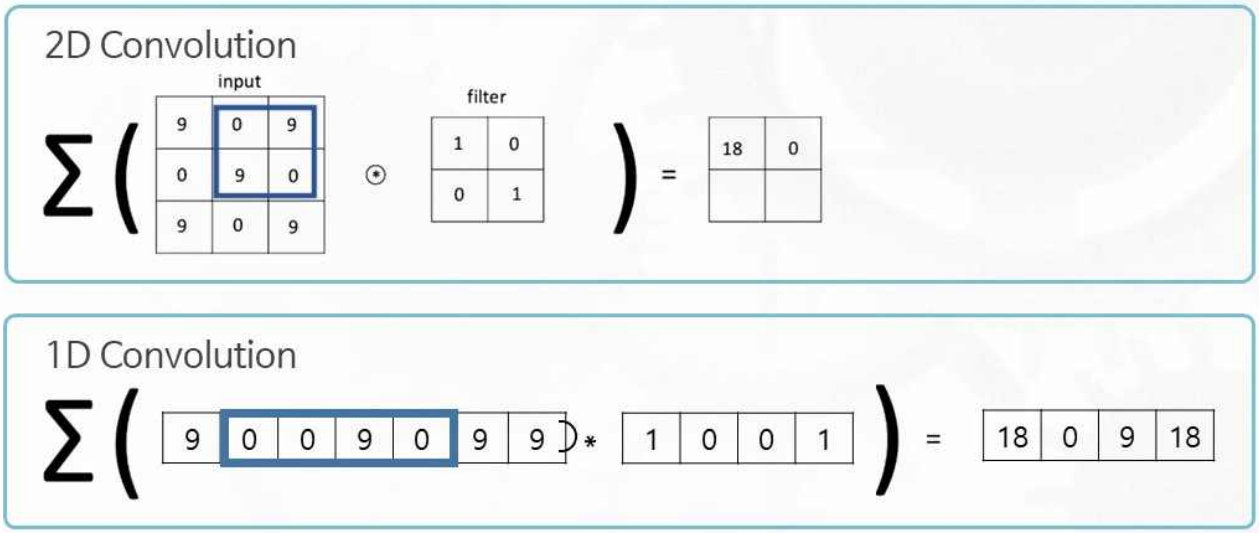
출처: 퍼블릭에이아이(www.publicai.co.kr)

Tensoflow Convolution Function

```
[59]  1 input_array = np.array(
      2     [[[7, 1, 1],
      3       [1, 0, 2],
      4       [3, 2, 1]],
      5
      6      [[1, 1, 4],
      7       [8, 8, 3],
      8       [1, 1, 7]],
      9
     10      [[2, 3, 2],
     11       [7, 7, 1],
     12       [9, 1, 1]]]
     13 )
     14 input_array.shape
     15
     16 # 텐서플로우에 입력 가능하도록 앞에 차원 하나를 추가합니다.
     17 input_array = input_array.reshape([1,3,3,3])
     18
     19 # 필터도 3차원으로 만듭니다.
     20 filter_ = np.array(
     21     [[3, 0, 0],
     22      [0, 4, 0],
     23      [0, 0, 4],
     24      [2, 7, 5]]
     25 )
     26 filter_ = filter_.reshape([2,2,3,1])

[61]  1 input_tensor = Input(shape=[3,3,3])
      2 layer = Conv2D(filters=1, kernel_size=2, use_bias=False)(input_tensor)
      3
      4 model = Model(input_tensor, layer)
      5 model.layers[1].set_weights([filter_])

      1 model.predict(input_array)

WARNING:tensorflow:8 out of the last 9 calls to <function Model.make_predict_
array([[[[124.],
         [ 67.]],

        [[111.],
         [ 62.]]]], dtype=float32)
```

1D Convolution 연산 vs 2D Convolution 연산

- 1D Convolution 과 2D Convolution의 차이

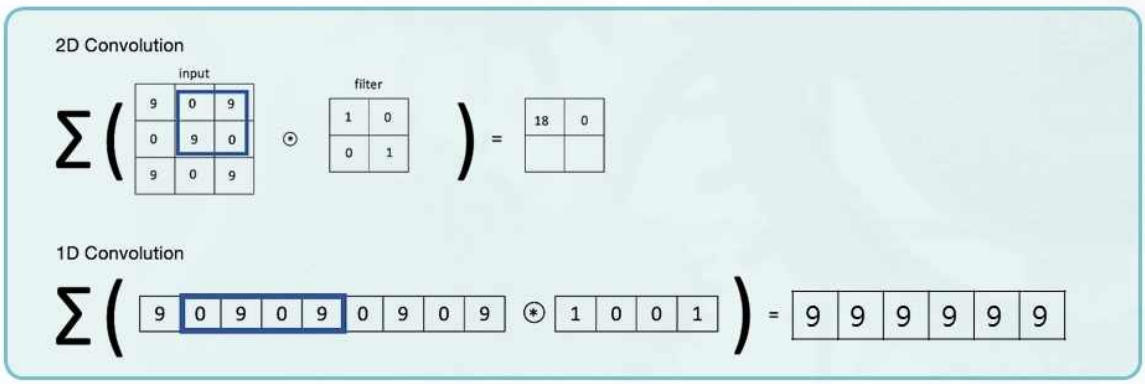


출처: 퍼블릭에이아이(www.publicai.co.kr)

04 1D Conv , 2D Conv

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

1D Convolution 연산 vs 2D Convolution 연산



→ 1D, 2D Convolution의 원리는 동일하지만 차이점은
Filter와 Filter가 적용될 Window가 1D 와 2D라는 것임

→ 2D Convolution 은 2D 특성을 추출하고
1D Convolution은 1D 특성을 추출함

출처: 퍼블릭에이아이(www.publicai.co.kr)

Tensorflow 1D Convolution 코드

1D Convolution Layer

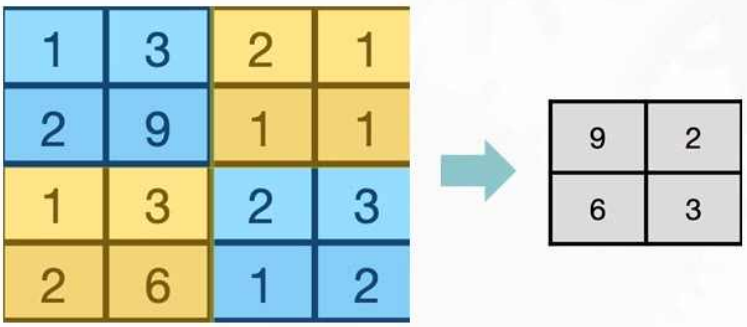
```
[67]  1 input_array = np.array(
      2     [9,0,9,0,9,0,9,0,9]
      3 )
      4 input_array = input_array.reshape([1,9,1])
      5
      6 filter_ = np.array(
      7     [1, 0, 0, 1]
      8 )
      9
     10 filter_ = filter_.reshape(4,1,1)
     11
```

```
[69]  1 input_tensor = Input(shape=[9, 1])
      2 layer = Conv1D(filters=1, kernel_size=4, use_bias=False)(input_tensor)
      3
      4 model = Model(input_tensor, layer)
      5 model.layers[1].set_weights([filter_])
```

```
[70]  1 model.predict(input_array)
```

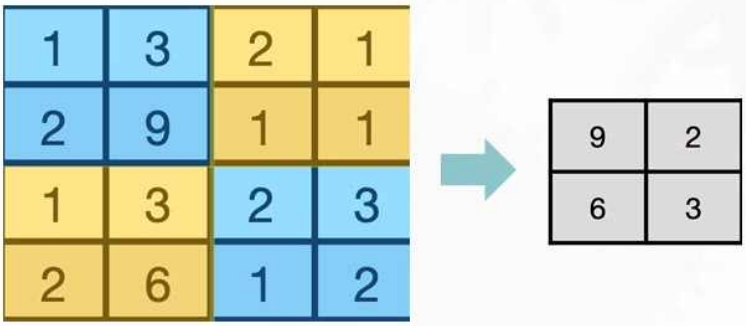
```
WARNING:tensorflow:9 out of the last 10 calls to <function Model.make_predict_func
array([[[9.],
        [9.],
        [9.],
        [9.],
        [9.],
        [9.]]], dtype=float32)
```


Pooling 연산



- Conovolution을 수행 시 입력데이터의 크기가 거의 줄어 들지 않음
- 이로 인해 연산량이 증가하게 되고 학습 속도가 느려지게 됨
- 연산량을 줄이고 학습 속도를 빠르게 하기 위해 Pooling을 수행함

Pooling 연산



- Max Pooling은 Window 내 가장 큰 값을 반환함
- 일반적으로 Window을 겹치지 않게 함으로서 입력 데이터를 반으로 줄임

05 MaxPooling

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Pooling 연산

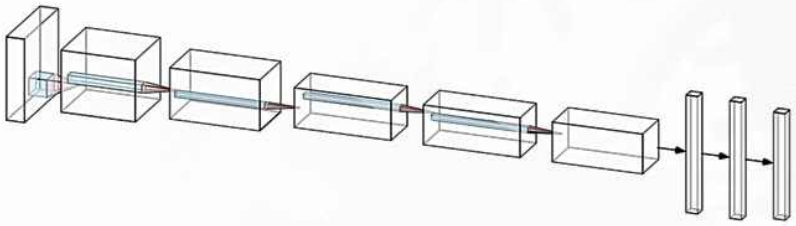
Max Pooling

```
[76]  1 input_array = np.array(
      2     [[1, 3, 2, 1],
      3      [2, 9, 1, 1],
      4      [1, 3, 2, 3],
      5      [2, 6, 1, 2]]
      6 )
      7
      8 input_array = input_array.reshape(1,4,4,1)
```

```
[80]  1 input_tensor = Input(shape=[4, 4, 1])
      2 layer = MaxPool2D()(input_tensor)
      3 model = Model(input_tensor, layer)
      4 model.predict(input_array)
      5
```

```
WARNING:tensorflow:10 out of the last 11 calls to <fu
array([[[[9.],
         [2.]],

        [[6.],
         [3.]]]], dtype=float32)
```

Fully Connected Layer



출처: 퍼블릭에이아이(www.publicai.co.kr)

- 입력 데이터가 Convolution layer을 지난 다음에는 Fully Connnected Layer(DenseLayer)을 지나게 함으로서 최종적으로 우리가 원하는 형태로 변환함

06 Dense Layer

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

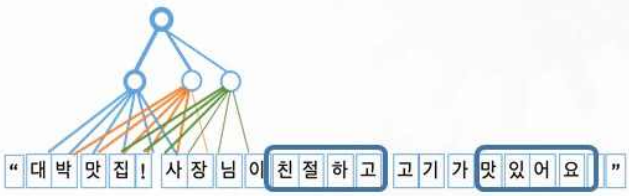
Dense Layer

```
input_tensor = Input(shape=[16, 16, 1])

# Convolution
layer = Conv2D(filters=1, kernel_size=2, use_bias=False)(input_tensor)
layer = Conv2D(filters=1, kernel_size=2, use_bias=False)(layer)
layer = MaxPool2D()(layer)

# Convolution
layer = Conv2D(filters=1, kernel_size=2, use_bias=False)(input_tensor)
layer = Conv2D(filters=1, kernel_size=2, use_bias=False)(layer)
layer = MaxPool2D()(layer)


# Flatten
layer = Flatten()(layer)

# Dense Layrer
layer = Dense(units=128)(layer)
layer = Dense(units=128)(layer)
layer = Dense(units=1)(layer)

# Model
model = Model(input_tensor, layer)
```

APPLICATION

02 자연어 처리에서의 CNN

01 자연어 처리에서의 1D Convolution Neural Network

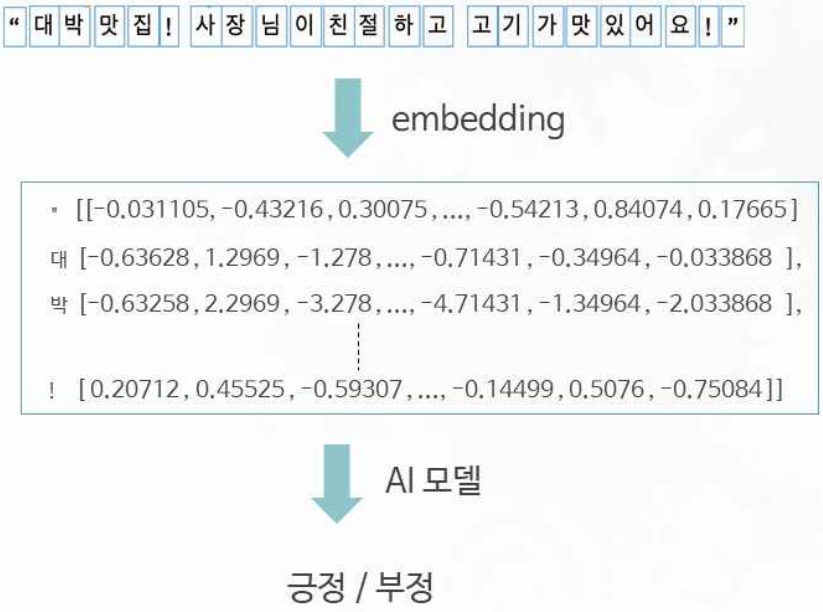


특정 패틴이 해당 문장이
긍정적인걸 파악 할 수 있게 해줌

- 일반적으로 자연어는 1D Sequence 의 데이터임
- 추가적인 전가 필처리요하지만 일반적으로 우리는 자연어에서 목적 함수에 영향을 미치는 패턴을 1D Convolution을활용해 찾을 수 있음

출처: 퍼블릭에이아이(www.publicai.co.kr)

01 자연어 처리에서의 1D Convolution Neural Network



- 자연어를 Embedding 한 후 적절한 AI 모형에 넣어 분류 또는 회귀 모델을 만들 수 있음
- 해당 과정에서는 Conv1D를 활용해 긍정과 부정의 리뷰를 분류 해보도록 함

출처: 퍼블릭에이아이(www.publicai.co.kr)

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

02 IMDB 데이터셋을 활용한 자연어 처리 실습

IMDB Dataset 설명 및 프로젝트 설명

IMDB Dataset은

> **약 25,000 개 영어 단어로 구성되어 있는 영어 리뷰 데이터 셋임**

IMDB Dataset 설명 및 프로젝트 설명

IMDB 감정 분석 프로젝트는

DB 데이터와 Convolution Neural Network을 활용해

리뷰가 긍정(Positive)인지 부정(Negative) 인지 파악

02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

IMDB 감정 분류 데이터셋 다운로드

- Keras 을 활용한 IMDB 데이터셋 다운로드

```
from tensorflow.keras.datasets import imdb
```

```
(x_train, y_train), (x_test, y_test) = imdb.load_data()
```

```
Signature:
imdb.load_data(
    path='imdb.npz',
    num_words=None,
    skip_top=0,
    maxlen=None,
    seed=113,
    start_char=1,
    oov_char=2,
    index_from=3,
    **kwargs,
)
Docstring:
Loads the IMDB dataset.

Arguments:
    path: where to cache the data (relative to `~/.keras/dataset`).
    num_words: max number of words to include. Words are ranked
        by how often they occur (in the training set) and only
        the most frequent words are kept
    skip_top: skip the top N most frequently occurring words
        (which may not be informative).

Returns:
    Tuple of Numpy arrays: `(x_train, y_train), (x_test, y_test)`.

Raises:
    ValueError: in case `maxlen` is so low
        that no input sequence could be kept.

Note that the 'out of vocabulary' character is only used for
words that were present in the training set but are not included
because they're not making the `num_words` cut here.
Words that were not seen in the training set but are in the test set
have simply been skipped.
File:      ~/anaconda3/envs/python3/lib/python3.6/site-packages/tensorflow_core/python/keras/datasets/imdb.py
Type:      function
```

Keras에서 제공해주는 Dataset은

일반적으로 (x_train,y_train), (x_test,y_test)

이 형태로 제공됨

IMDB 감정 분류 데이터셋 다운로드

IMDB movie review sentiment classification dataset

`load_data` function

```
tf.keras.datasets.imdb.load_data(
    path="imdb.npz",
    num_words=None,
    skip_top=0,
    maxlen=None,
    seed=113,
    start_char=1,
    oov_char=2,
    index_from=3,
    **kwargs
)
```

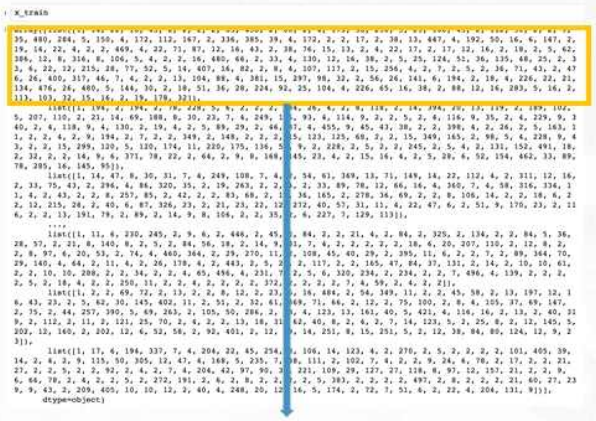
Loads the IMDB dataset.

This is a dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative). Reviews have been preprocessed, and each review is encoded as a list of word indexes (integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data. This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".

As a convention, "0" does not stand for a specific word, but instead is used to encode any unknown word.

Arguments

- **path**: where to cache the data (relative to `~/.keras/dataset`).
- **num_words**: integer or None. Words are ranked by how often they occur (in the training set) and only the `num_words` most frequent words are kept. Any less frequent word will appear as `oov_char` value in the sequence data. If None, all words are kept. Defaults to None, so all words are kept.
- **skip_top**: skip the top N most frequently occurring words (which may not be informative). These words will appear as `oov_char` value in the dataset. Defaults to 0, so no words are skipped.
- **maxlen**: int or None. Maximum sequence length. Any longer sequence will be truncated. Defaults to None, which means no truncation.
- **seed**: int. Seed for reproducible data shuffling.
- **start_char**: int. The start of a sequence will be marked with this character. Defaults to 1 because 0 is usually the padding character.
- **oov_char**: int. The out-of-vocabulary character. Words that were cut out because of the `num_words` or `skip_top` limits will be replaced with this character.
- **index_from**: int. Index actual words with this index and higher.
- ****kwargs**: Used for backwards compatibility.



기존의 단어를 index로 변환해 놓음

* Index 순서는 해당데이터에서 자주 나타난 순위

02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

IMDB 감정 분류 데이터셋 다운로드

```
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=500)
```

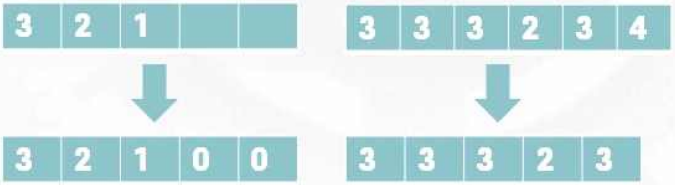
출현 빈도수가 가장 높은 500개의 단어만 사용함

Data EDA

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
# 문장의 길이를 300 으로 맞춥니다. 문장이 길면 자르고 문장이 짧으면 padding 을 수행합니다.
x_train_padded = pad_sequences(x_train, maxlen=300, padding='post')
x_test_padded = pad_sequences(x_test, maxlen=300, padding='post')
```

- 문장을 제한된 길이로 늘리거나 줄이는 작업을 수행함
  Padding = 'post' 임으로

Maxlen = 5, padding='post'



1 : !
2 : world
3 : hello
4 : apple

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

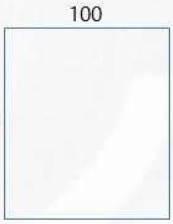
- 입력 백터의 크기를 지정함
- 입력 백터의 크기는 이전 코드의 pad_sequence 함수의 Maxlen 길이와 같음

```
# 문장의 길이를 300 으로 맞춥니다. 문장이 길면 자르고 문장이 짧으면 padding 을 수행합니다.
x_train_padded = pad_sequences(x_train, maxlen=300, padding='post')
x_test_padded = pad_sequences(x_test, maxlen=300, padding='post')
```

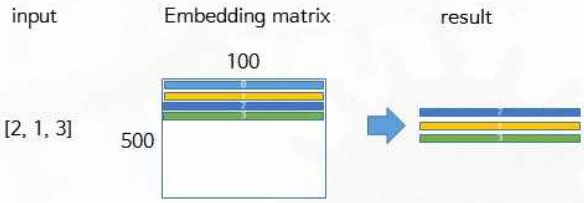
Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

- 크기가 500, 100인 Embedding matrix을 생성함



- 입력값이 들어오면 해당 index에 맞는 Vector 을 출력함



02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

→ Convolution layer을 생성함

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

→ Max pooling layer을 통해 출력 크기를 반으로 줄임

02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

→
- Convolution layer을 생성함
- 입력 크기가 반으로 준 만큼 필터의 개수를 2배 더 증가시킴
- 그리고 2개의 convolution layer을 통과한 후에는 다시 출력의 크기를 반으로 줄임

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

- Convolution layer을 생성함
- 입력 크기가 반으로 준 만큼 필터의 개수를 2배 더 증가시킴
- 그리고 2개의 convolution layer을 통과한 후에는 다시 출력의 크기를 반으로 줄임

02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

Convolution layer 출력 값을 한 줄로 피는 작업을 수행함

Model

```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```

- Activation function이 sigmoid인 Dense Layer을 추가함
- Dense Layer을 통과한 Layer은 0, 1의 사이의 확률 값을 가짐
- 우리는 0.5 이상이면 긍정, 0.5 이하면 부정으로 예측함

02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Model


```
# Model
K.clear_session()

# input define
inputs = Input(shape=(300))

# weight shape (3000, 500)
layer = Embedding(input_dim=500, output_dim=100)(inputs)

# block 1
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=64, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 2
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=128, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

# block 3
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = Conv1D(filters=256, kernel_size=5, activation='relu', kernel_initializer='he_normal')(layer)
layer = MaxPool1D()(layer)

layer = Flatten()(layer)
layer = Dense(1, activation='sigmoid')(layer)
model = Model(inputs, layer)
```


Input, output을 묶어 Keras Model로 만듦

Training

```
model.compile('adam', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x=padded_train_x, y=y_train, validation_data=(padded_test_x, y_test), epochs=15)
```

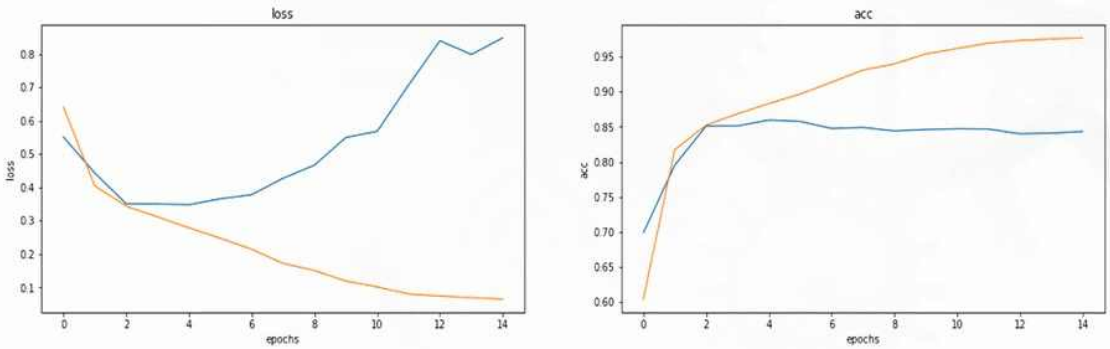
- Optimizer는 adam으로, loss function은 binary cross entropy로 평가 방법은 accuracy로 함
- 학습 방법을 지정했다면 이제 학습함

02 IMDB 데이터셋을 활용한 자연어 처리 실습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

```
# result visualization
fig, axes = plt.subplots(1,2, figsize=(20,5))
axes = np.array(axes).ravel()
axes[0].plot(history.history["val_loss"])
axes[0].plot(history.history["loss"])
axes[0].set_title('loss')
axes[0].set_xlabel('epochs')
axes[0].set_ylabel('loss')

axes[1].plot(history.history["val_acc"])
axes[1].plot(history.history["acc"])
axes[1].set_title('acc')
axes[1].set_xlabel('epochs')
axes[1].set_ylabel('acc')
```

```
Text(0, 0.5, 'acc')
```



APPLICATION

03 실습

SUMMARY

학습정리

- Convolution 개요
- Convolution에 대한 작동 방식
- 고차원에서의 Convolution
- 1D Conv, 2D Conv
- Max-pooling와 Dense Layer
- 자연어 처리에서의 1D Conv 사용
- IMDB 데이터셋을 활용한 자연어 처리 실습

EXPANSION

확장하기

1. CNN에서의 Filter는 어떤 역할을 할까요?
2. CNN에서의 Convolution 연산은 어떤 원리로 동작할까요?
3. Pooling 연산을 진행하는 이유는 무엇일까요?
4. 자연어처리에서 Conv1D layer를 사용하는 이유는 무엇일까요?
5. Keras에서 제공하는 데이터셋은 어떤 형태로 존재할까요?

참고 문헌

REFERENCE

◆ 참고 사이트
- 용어들에 대한 정의 : https://ko.wikipedia.org/wiki.
- Stanford CS224n : http://web.stanford.edu/class/cs224n/
- 퍼블릭에이아이(www.publicai.co.kr)

서체 출처 : 에스코어드림체-㈜에스코어, 나눔글꼴체-㈜네이버