AI School 6기 6주차

파이썬 기초 – 클래스2

CNN 기초2

CNN 모델을 활용한 객체 분류

AI School 6기 6주차

파이썬 기초 – 클래스2

파이썬의 클래스

클래스

객체(인스턴스)

파이썬의 클래스

- 클래스는 객체의 구조와 행동을 정의
- 객체의 클래스는 초기화를 통해 제어
- 클래스는 복잡한 문제를 다루기 쉽도록 만들어줌

airtravel.py

```
class Flight:
    pass
```

flight_test.py

```
from Python_basic.airtravel import Flight
#생성한 클래스를 import

f = Flight() #클래스 객체 생성 및 변수에 할당
print(type(f))
```

파이썬의 클래스

- 메소드란 클래스 내의 함수
- self: 파이썬 메소드의 첫번째 파라미터명
- 인스턴스 메소드란 객체에서 호출되어질수 있는 함수

airtravel.py

```
class Flight:
    def number (self): #메소드 작성
        return 'KE081'
```

flight_test.py

```
from Python_basic.airtravel import Flight
#생성한 클래스를 import

f = Flight() #클래스 객체 생성 및 변수에 할당, 생성자
print(f.number())
```

파이썬의 클래스

- 생성자와 초기화

airtravel.py

```
class Flight:
    def __init__(self):
        print('init')
        super().__init__()

    def __new__(cls):
        print('new')
        return super().__new__(cls)

    def number(self): #메소드 작성
            return 'KE081'
```

flight_test.py

```
from Python_basic.airtravel import Flight

f = Flight()
```

파이썬의 클래스

- 초기화

airtravel.py

```
class Flight:
    def __init__(self, number):
        self._number = number

    def number(self):
            return self._number
```

flight_test.py

```
from Python_basic.airtravel import Flight

f = Flight('KE082')

print(f.number())
print(f._number)
```

파이썬의 클래스

- 인스턴스 속성(변수)

airtravel.py

```
class Flight:
    def __init__(self, number, passenger_num):
        self.__number = number
        self._passenger_num = passenger_num

    def number(self): #메소드 작성
            return self.__number

    def add_passenger(self, num):
        self._passenger_num += num
```

flight_test.py

```
f1 = Flight('KE082', 0) #클래스 객체 생성 및 변수에 할당
f2 = Flight('KE081', 0)
f1.add_passenger(2)
f2.add_passenger(3)
print(f1._passenger_num)
print(f2._passenger_num)
```

파이썬의 클래스

- 클래스 속성

airtravel.py

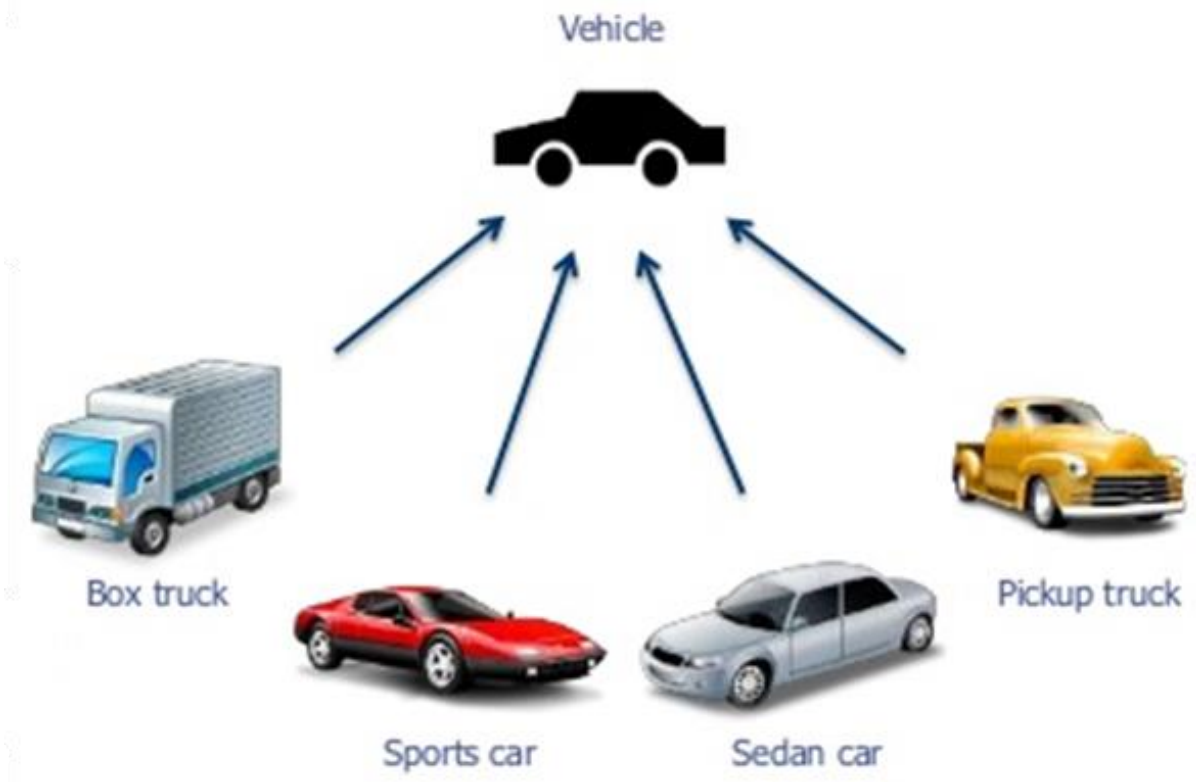
```
class Flight:
    nation = 'Korea'
.
.
.
```

flight_test.py

```
f1 = Flight('KE082', 0) #클래스 객체 생성 및 변수에 할당
f2 = Flight('KE081', 0)
print(f1.nation)
print(f2.nation)
```

클래스의 상속

- 어떤 클래스를 만들 때 다른 클래스의 기능을 물려받음
- 물려받은 기능을 유지한채로 다른 기능을 추가할 때 사용
- 물려주는 클래스 (Parent class, Super class)
- 물려받는 클래스 (Child class, Sub class)



클래스의 상속

- 자식 클래스는 이름 옆에 부모클래스의 이름을 입력
- 부모 클래스의 메소드와 속성 모두 사용 가능
- 부모 클래스의 메소드 사용 예제

person.py

```
class Person:
    def greeting(self):
        print('안녕하세요.')

class Student(Person):
    def study(self):
        print('공부하기')
```

person_test.py

```
from Python_basic.person import Student

kim = Student()
kim.greeting()
kim.study()
```

클래스의 상속

- 부모 클래스의 속성

person.py

```
class Person:
    def __init__(self):
        self.num_arm = 2
        print("Person_init")
            ...
class Student(Person):
    def __init__(self, semester):
        super().__init__()
        print("Student_init")
        self.semester =semester
            ...
```

person_test.py

```
from Python_basic.person import Student

kim = Student(2)
print(kim.num_arm)
print(kim.semester)
```

클래스의 상속

- 자식 클래스에 __init__이 없으면 super() 생략가능

person.py

```
class Person:
    def __init__(self):
        self.num_arm = 2
        print("Person_init")
            ...
class Student(Person):
    pass
```

person_test.py

```
from Python_basic.person import Student

kim = Student()
print(kim.num_arm)
```

클래스의 상속

- 메소드 오버라이딩 (Overriding)

person.py

```
class Person:
    def __init__(self):
        self.num_arm = 2
    def greeting(self):
        print('안녕하세요.')
class Student(Person):
    ...
    def greeting(self):
        super().greeting()
        print(f'석사과정 {self.semester}학기생입니다.')
```

person_test.py

```
from Python_basic.person import Student

kim = Student(2)
kim.greeting()
```

클래스의 상속

- 다중 상속: 여러 부모 클래스로 부터 상속을 받는 것

person.py

```
class Person:
    def __init__(self):
        self.num_arm = 2
    def greeting(self):
        print('안녕하세요.')

class University:
    def credit_show(self):
        print("A")

class Student(Person, University):
            ….
```

person_test.py

```
from Python_basic.person import Student
kim = Student(2)
kim.greeting()
kim.credit_show()
```

클래스의 상속

- 추상 클래스

person.py

```
from abc import *

class StudentBase(metaclass=ABCMeta):
    @abstractmethod
    def study(self):
        pass
    @abstractmethod
    def go_to_school(self):
        pass
class Student(StudentBase):
    def study(self):
        print('공부하기')
    def go_to_school(self):
        print('학교가기')
```

person_test.py

```
from Python_basic.person import Student
kim = Student()
kim.go_to_school()
kim.study()
```

연습문제 (지난 주)

- fourcal.py에서 사칙연산을 수행하는 Calculator 클래스를 만드세요.
- def __init__(self, num1, num2)
- def add(self)
  .
  .
  return result
- def sub(self)
- def mul(self)
- def div(self)

- calculator_test.py에서 객체 생성 후 4가지 메소드 사용 결과 출력

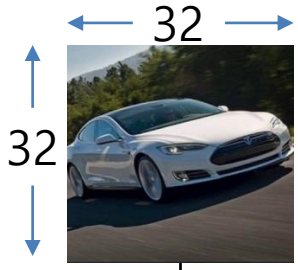
연습문제

- Calculator 클래스를 상속 받는 ScienCalculator를 만드세요.
- 제곱을 계산하는 def pow(self) 함수를 추가하세요. $num1^{num2}$
- 메소드 오버라이딩을 이용해 div함수가 분모가 0일 때 0을 return하도록 재정의 하세요.
- Calculator_test.py에서 ScienCalculator 객체 생성 후 5가지 메소드 사용 결과 출력
- dha8102@naver.com

AI School 6기 6주차

CNN 모델을 활용한 객체 분류

ResNet for CIFAR-10



1
# of filter: 16

3x3 conv, 16

2n
# of filter: 16

3x3 conv, 16
3x3 conv, 16
3x3 conv, 16
3x3 conv, 16
3x3 conv, 16
3x3 conv, 16

2n
# of filter: 32

3x3 conv, 32, /2
3x3 conv, 32
3x3 conv, 32
3x3 conv, 32
3x3 conv, 32
3x3 conv, 32

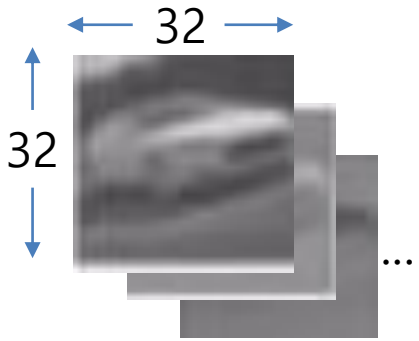
2n
# of filter: 64

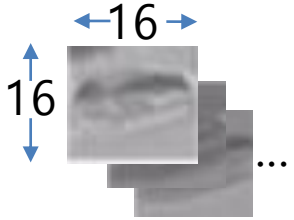
3x3 conv, 64, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64

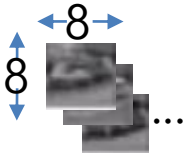
avg pool

1

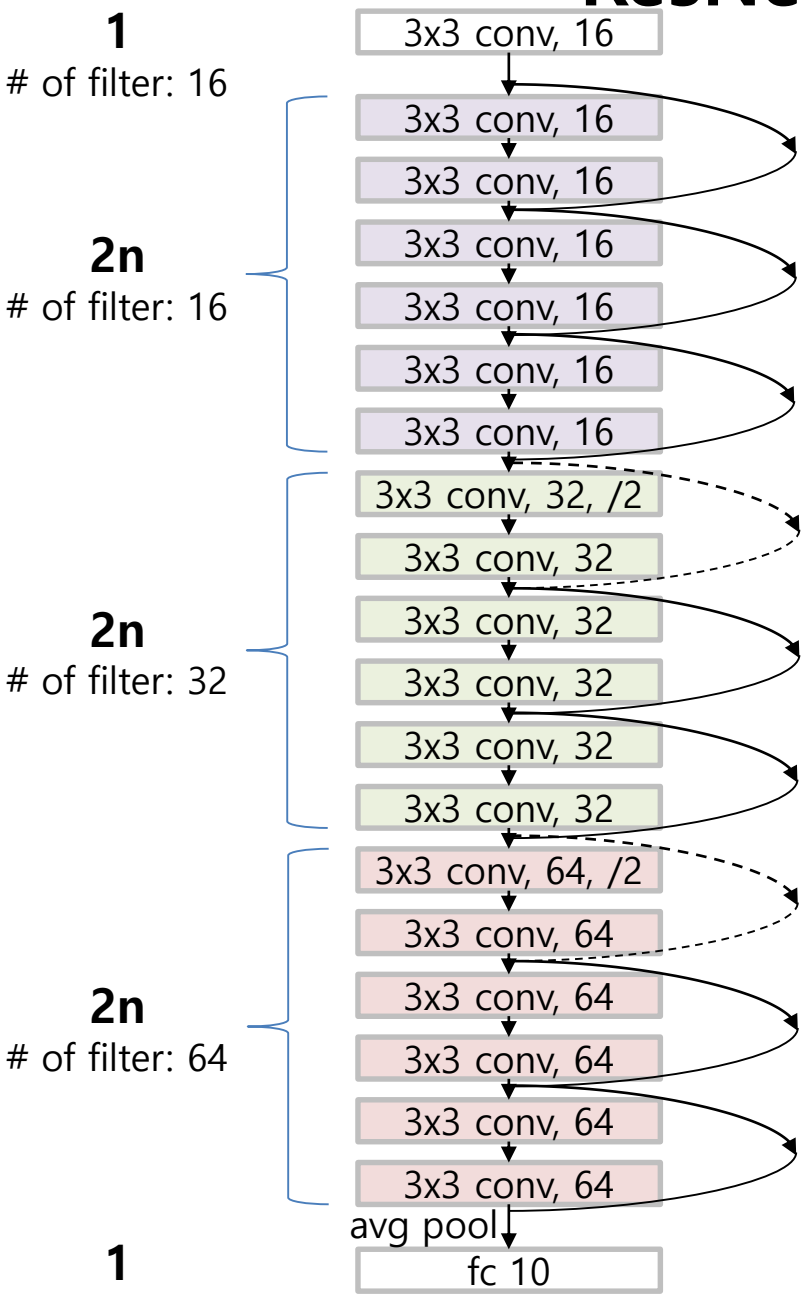
fc 10







ResNet for CIFAR-10



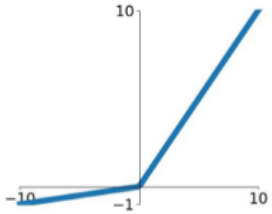
	method			error (%)
n=3	ResNet	20	0.27M	8.75
n=5	ResNet	32	0.46M	7.51
n=7	ResNet	44	0.66M	7.17
n=9	ResNet	56	0.85M	6.97
n=18	ResNet	110	1.7M	**6.43** (6.61±0.16)
n=200	ResNet	1202	19.4M	7.93

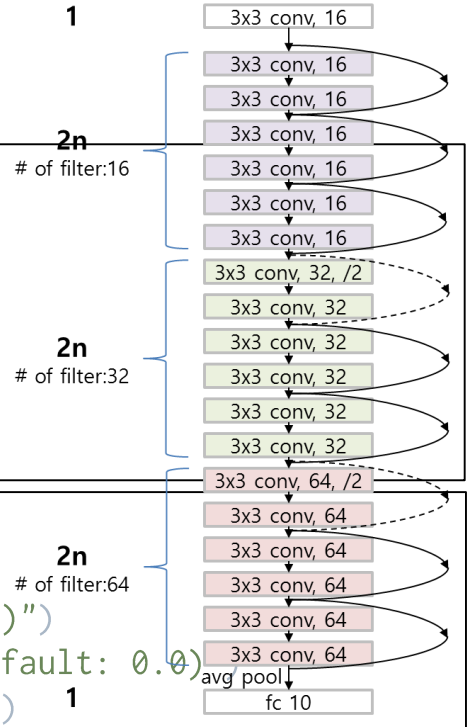
(6n +2)

Hyperparameters

- resnet_train.py

```
import tensorflow as tf
import os
import time
import datetime
import CIFAR.data_helpers as dh
from CIFAR.resnet import ResNet
from tensorflow.keras.datasets.cifar10 import load_data
```






```
# Model Hyperparameters
tf.flags.DEFINE_float("lr", 0.1, "learning rate (default=0.1)")
tf.flags.DEFINE_float("lr_decay", 0.1, "learning rate decay rate(default=0.1)")
tf.flags.DEFINE_float("l2_reg_lambda", 0.0001, "L2 regularization lambda (default: 0.0)")
tf.flags.DEFINE_float("relu_leakiness", 0.1, "relu leakiness (default: 0.1)")
tf.flags.DEFINE_integer("num_residual_units", 3, "The number of residual_units (default: 5)")
tf.flags.DEFINE_integer("num_classes", 10, "The number of classes (default: 10)")
# Training parameters
tf.flags.DEFINE_integer("batch_size", 128, "Batch Size (default: 64)")
tf.flags.DEFINE_integer("num_epochs", 100, "Number of training epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 100, "Evaluate model on dev set after this many steps
(default: 100)")
tf.flags.DEFINE_integer("checkpoint_every", 100, "Save model after this many steps (default: 100)")
tf.flags.DEFINE_integer("num_checkpoints", 3, "Number of checkpoints to store (default: 5)")
# Misc Parameters
tf.flags.DEFINE_boolean("allow_soft_placement", True, "Allow device soft device placement")
tf.flags.DEFINE_boolean("log_device_placement", False, "Log placement of ops on devices")
FLAGS = tf.flags.FLAGS
```

0보다 작을 때 기울기

6n +2

Unit 개수, 즉 n의 크기

Data loading & preprocessing

- resnet_train.py

```
(x_train_val, y_train_val), (x_test, y_test) = load_data()
x_train, y_train, x_test, y_test, x_val, y_val = dh.shuffle_data(x_train_val, y_train_val, x_test,
y_test)
```

- data_helpers.py

```
import numpy as np
import random

def shuffle_data(x_train_val, y_train_val, x_test, y_test):
    shuffle_indices = np.random.permutation(np.arange(len(y_train_val)))
    shuffled_x = np.asarray(x_train_val[shuffle_indices])
    shuffled_y = y_train_val[shuffle_indices]
    val_sample_index = -1 * int(0.1 * float(len(y_train_val)))  # 5000
    x_train, x_val = shuffled_x[:val_sample_index], shuffled_x[val_sample_index:]
    y_train, y_val = shuffled_y[:val_sample_index], shuffled_y[val_sample_index:]
    x_test = np.asarray(x_test)
    y_train_one_hot = np.eye(10)[y_train]  # [9, 8] -> [[0, 0, 0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 0,
0, 0, 0, 1, 0]
    y_train_one_hot = np.squeeze(y_train_one_hot, axis=1)  # (45000, 10)
    y_test_one_hot = np.eye(10)[y_test]
    y_test_one_hot = np.squeeze(y_test_one_hot, axis=1)
    y_val_one_hot = np.eye(10)[y_val]
    y_val_one_hot = np.squeeze(y_val_one_hot, axis=1)
    return x_train, y_train_one_hot, x_test, y_test_one_hot, x_val, y_val_one_hot
```

index	label
0	airplane (0)
1	automobile (1)
2	bird (2)
3	cat (3)
4	deer (4)
5	dog (5)
6	frog (6)
7	horse (7)
8	ship (8)
9	truck (9)
...	...
...	...

original label data

label	index									
	0	1	2	3	4	5	6	7	8	9
airplane	1	0	0	0	0	0	0	0	0	0
automobile	0	1	0	0	0	0	0	0	0	0
bird	0	0	1	0	0	0	0	0	0	0
cat	0	0	0	1	0	0	0	0	0	0
deer	0	0	0	0	1	0	0	0	0	0
dog	0	0	0	0	0	1	0	0	0	0
frog	0	0	0	0	0	0	1	0	0	0
horse	0	0	0	0	0	0	0	1	0	0
ship	0	0	0	0	0	0	0	0	1	0
truck	0	0	0	0	0	0	0	0	0	1

one-hot-encoded label data

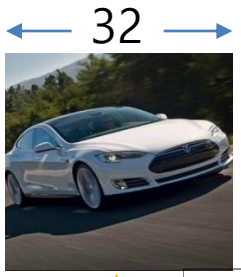
ResNet class & input

- resnet_train.py

```
with tf.Graph().as_default():
    session_conf = tf.ConfigProto(
      allow_soft_placement=FLAGS.allow_soft_placement,
      log_device_placement=FLAGS.log_device_placement)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        resnet = ResNet(FLAGS)
```

- resnet.py

```
import tensorflow as tf
import numpy as np
from tensorflow.python.training import moving_averages

class ResNet:
    def __init__(self, config):

        self._num_residual_units = config.num_residual_units
        self._batch_size = config.batch_size
        self._relu_leakiness = config.relu_leakiness
        self._num_classes = config.num_classes
        self._l2_reg_lambda = config.l2_reg_lambda

        self.X = tf.placeholder(tf.float32, [None, 32, 32, 3], name="X")
        self.Y = tf.placeholder(tf.float32, [None, self._num_classes], name="Y")
        self.extra_train_ops = []
```



label	index									
	0	1	2	3	4	5	6	7	8	9
airplane	1	0	0	0	0	0	0	0	0	0
automobile	0	1	0	0	0	0	0	0	0	0
bird	0	0	1	0	0	0	0	0	0	0
cat	0	0	0	1	0	0	0	0	0	0
deer	0	0	0	0	1	0	0	0	0	0
dog	0	0	0	0	0	1	0	0	0	0
frog	0	0	0	0	0	0	1	0	0	0
horse	0	0	0	0	0	0	0	1	0	0
ship	0	0	0	0	0	0	0	0	1	0
truck	0	0	0	0	0	0	0	0	0	1

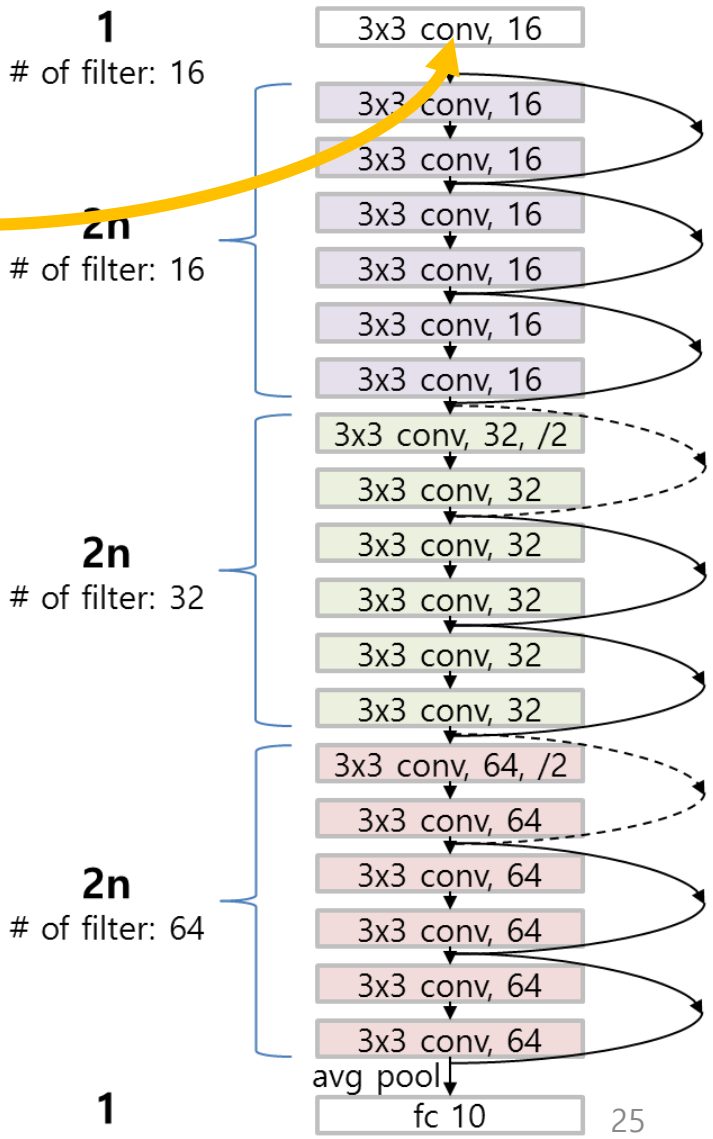
Initial convolutional layer

- resnet.py

층별 filter 개수 (output channel)

```
filters = [16, 16, 32, 64]
activate_before_residual = [True, False, False]

with tf.variable_scope('init'):
  x = self._conv('init_conv', self.X, 3, 3, filters[0],
strides=[1, 1, 1, 1])
```

최초 convolutional layer

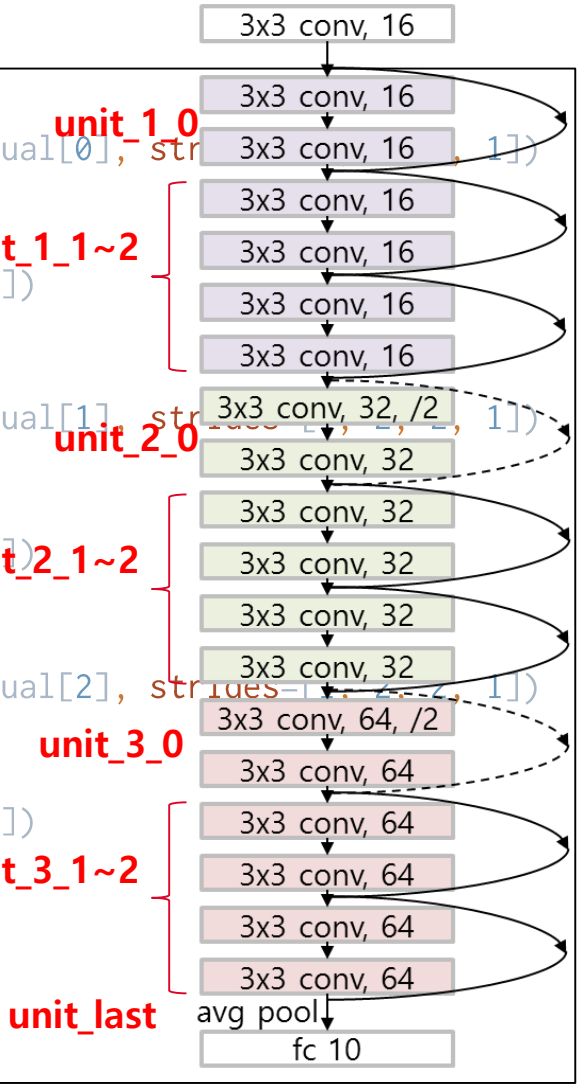


Residual units

- resnet.py

n = 3일 때

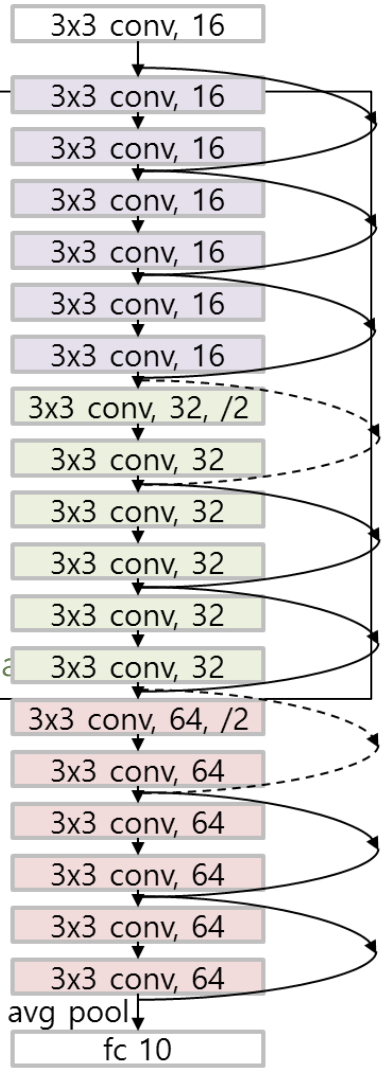
```
with tf.variable_scope('unit_1_0'):
  x = self._residual(x, filters[0], filters[1], activate_before_residual[0], str
for i in range(1, self._num_residual_units):
  with tf.variable_scope('unit_1_%d' % i):
    x = self._residual(x, filters[1], filters[1], strides=[1, 1, 1, 1])

with tf.variable_scope('unit_2_0'):
  x = self._residual(x, filters[1], filters[2], activate_before_residual[1], str
for i in range(1, self._num_residual_units):
  with tf.variable_scope('unit_2_%d' % i):
    x = self._residual(x, filters[2], filters[2], strides=[1, 1, 1, 1])

with tf.variable_scope('unit_3_0'):
  x = self._residual(x, filters[2], filters[3], activate_before_residual[2], str
for i in range(1, self._num_residual_units):
  with tf.variable_scope('unit_3_%d' % i):
    x = self._residual(x, filters[3], filters[3], strides=[1, 1, 1, 1])

with tf.variable_scope('unit_last'):
  x = self._batch_norm('final_bn', x)
  x = self._relu(x, self._relu_leakiness)
  x = self._global_avg_pool(x)
```



Fully connected layer, weight decay

- resnet.py

```
with tf.variable_scope('logit'):
  logits = self._fully_connected(x, self._num_classes)
  self.predictions = tf.nn.softmax(logits)
  self.predictions = tf.argmax(self.predictions, 1, name="predictions")

with tf.variable_scope('loss'):
  xent = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=self.Y)
  self.loss = tf.reduce_mean(xent, name='xent')
  self.loss += self._decay()

with tf.name_scope("accuracy"):
    correct_predictions = tf.equal(self.predictions, tf.argmax(self.Y, 1))
    self.accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"), name="a
```

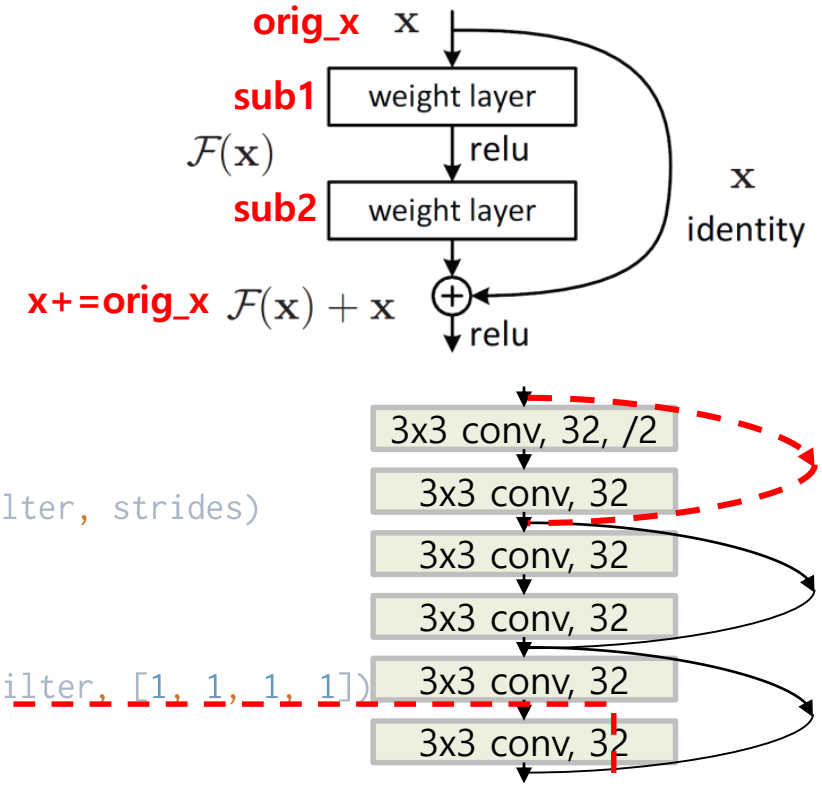


10개의 label 중 하나로 분류하기 위한
Fully connected layer

Residual

- resnet.py

```
def _residual(self, x, in_filter, out_filter, activate_before_residual=False, strides=[1, 1, 1, 1]):
    if activate_before_residual:
        with tf.variable_scope('common_activation'):
            x = self._batch_norm('init_bn', x)
            x = self._relu(x, self._relu_leakiness)
            orig_x = x
    else:
        with tf.variable_scope('residual_activation'):
            orig_x = x
            x = self._batch_norm('init_bn', x)
            x = self._relu(x, self._relu_leakiness)
    with tf.variable_scope('sub1'):
        x = self._conv('conv1', x, 3, in_filter, out_filter, strides)
    with tf.variable_scope('sub2'):
        x = self._batch_norm('bn2', x)
        x = self._relu(x, self._relu_leakiness)
        x = self._conv('conv2', x, 3, out_filter, out_filter, [1, 1, 1, 1])
    with tf.variable_scope('sub_add'):
        if in_filter != out_filter:
            orig_x = tf.nn.avg_pool(orig_x, strides, strides, 'VALID')
            orig_x = tf.pad(orig_x, [[0, 0], [0, 0], [0, 0], [(out_filter - in_filter) // 2, (out_filter - in_filter) // 2]])
        x += orig_x
    tf.logging.debug('image after unit %s', x.get_shape())
    return x
```



채널 크기가 변경됨에 따라
동일 한 크기의 feature map간에
Skip connection이 이루어지지 않을 때

Batch normalization

- resnet.py

```
def _relu(self, x, leakiness=0.0):
    return tf.where(tf.less(x, 0.0), leakiness * x, x, name='leaky_relu')
```

```
def _batch_norm(self, name, x):
    with tf.variable_scope(name):
        params_shape = [x.get_shape()[-1]]
        beta = tf.get_variable('beta', params_shape, tf.float32,
initializer=tf.constant_initializer(0.0, tf.float32))
        gamma = tf.get_variable('gamma', params_shape, tf.float32,
initializer=tf.constant_initializer(1.0, tf.float32))
        mean, variance = tf.nn.moments(x, [0, 1, 2], name='moments')
        moving_mean = tf.get_variable('moving_mean', params_shape, tf.float32,
            initializer=tf.constant_initializer(0.0, tf.float32), trainable=False)
        moving_variance = tf.get_variable('moving_variance', params_shape, tf.float32,
            initializer=tf.constant_initializer(1.0, tf.float32), trainable=False)

        self.extra_train_ops.append(moving_averages.assign_moving_
        self.extra_train_ops.append(moving_averages.assign_moving_
0.9))

        y = tf.nn.batch_normalization(x, mean, variance, beta, gam
        y.set_shape(x.get_shape())
        return y
```

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Fully connected layer, weight decay

- resnet.py

```
def _fully_connected(self, x, out_dim):
    dim = tf.reduce_prod(x.get_shape()[1:]).eval()
    x = tf.reshape(x, [-1, dim])
    w = tf.get_variable('DW', [dim, out_dim],
        initializer=tf.uniform_unit_scaling_initializer(factor=1.0))
    b = tf.get_variable('biases', [out_dim], initializer=tf.constant_initializer())
    return tf.nn.xw_plus_b(x, w, b)
```

```
def _global_avg_pool(self, x):
    assert x.get_shape().ndims == 4
    return tf.reduce_mean(x, [1, 2])
```

```
def _decay(self):
    """L2 weight decay loss."""
    costs = []
    for var in tf.trainable_variables():
        if var.op.name.find(r'DW') > 0:
            costs.append(tf.nn.l2_loss(var))

    return tf.multiply(self._l2_reg_lambda, tf.add_n(costs))
```

Optimizer

- resnet_train.py

```
with tf.Graph().as_default():
    session_conf = tf.ConfigProto(
      allow_soft_placement=FLAGS.allow_soft_placement,
      log_device_placement=FLAGS.log_device_placement)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        resnet = ResNet(FLAGS)

        # Define Training procedure
        global_step = tf.Variable(0, name="global_step", trainable=False)
        decayed_lr = tf.train.exponential_decay(FLAGS.lr, global_step, 24000, FLAGS.lr_decay,
staircase=True)
        optimizer = tf.train.MomentumOptimizer(learning_rate=decayed_lr, momentum=0.9)
        grads_and_vars = optimizer.compute_gradients(resnet.loss)
        train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)
        train_ops = [train_op] + resnet.extra_train_ops
        train_ops = tf.group(*train_ops)

        # Output directory for models and summaries
        timestamp = str(int(time.time()))
        out_dir = os.path.abspath(os.path.join(os.path.curdir, "runs", timestamp))
        print("Writing to {}\n".format(out_dir))
```

Summary, checkpoint

- resnet_train.py

```
# Summaries for loss and accuracy
loss_summary = tf.summary.scalar("loss", resnet.loss)
acc_summary = tf.summary.scalar("accuracy", resnet.accuracy)
# Train Summaries
train_summary_op = tf.summary.merge([loss_summary, acc_summary])
train_summary_dir = os.path.join(out_dir, "summaries", "train")
train_summary_writer = tf.summary.FileWriter(train_summary_dir, sess.graph)
# Dev summaries
dev_summary_op = tf.summary.merge([loss_summary, acc_summary])
dev_summary_dir = os.path.join(out_dir, "summaries", "dev")
dev_summary_writer = tf.summary.FileWriter(dev_summary_dir, sess.graph)

# Checkpoint directory. Tensorflow assumes this directory already exists so we need to create it
checkpoint_dir = os.path.abspath(os.path.join(out_dir, "checkpoints"))
checkpoint_prefix = os.path.join(checkpoint_dir, "model")
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)
saver = tf.train.Saver(tf.global_variables(), max_to_keep=FLAGS.num_checkpoints)
```

Train & dev step

- resnet_train.py

```
sess.run(tf.global_variables_initializer())
def train_step(x_batch, y_batch):
    feed_dict = {
      resnet.X: x_batch,
      resnet.Y: y_batch
    }
    _, step, lr, summaries, loss, accuracy = sess.run(
        [train_ops, global_step, decayed_lr, train_summary_op, resnet.loss, resnet.accuracy],
        feed_dict)
    time_str = datetime.datetime.now().isoformat()
    print("{}: step {}, lr {}, loss {:g}, acc {:g}".format(time_str, step, lr, loss, accuracy))
    train_summary_writer.add_summary(summaries, step)
def dev_step(x_batch, y_batch, writer=None):
feed_dict = {
      resnet.X: x_batch,
      resnet.Y: y_batch
    }
    step, summaries, loss, accuracy = sess.run(
        [global_step, dev_summary_op, resnet.loss, resnet.accuracy],
        feed_dict)
    time_str = datetime.datetime.now().isoformat()
    print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
    if writer:
        writer.add_summary(summaries, step)
    return accuracy
```

Batch, Training

- resnet_train.py

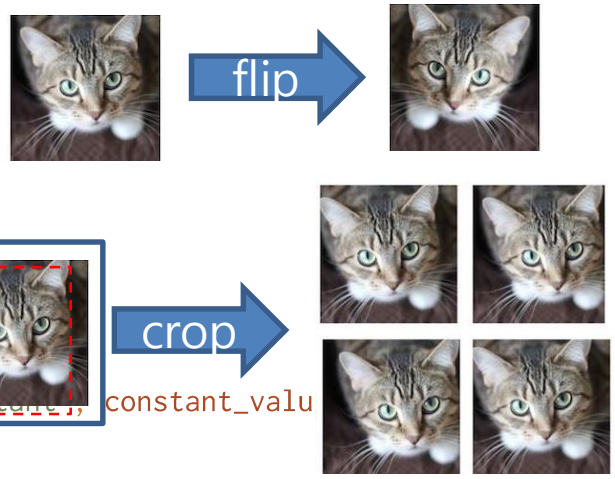
```
batches = dh.batch_iter(x_train, y_train, FLAGS.batch_size, FLAGS.num_epochs)
max = 0
for batch in batches:
    x_batch, y_batch = zip(*batch)
    train_step(x_batch, y_batch)
    current_step = tf.train.global_step(sess, global_step)
    if current_step % FLAGS.evaluate_every == 0:
        print("\nEvaluation:")
        accuracy = dev_step(x_val, y_val, writer=dev_summary_writer)
        print("")
        if accuracy > max:
            max = accuracy
            path = saver.save(sess, checkpoint_prefix, global_step=current_step)
            print("Saved model checkpoint to {}\n".format(path))
```

Batch

- data_helpers.py

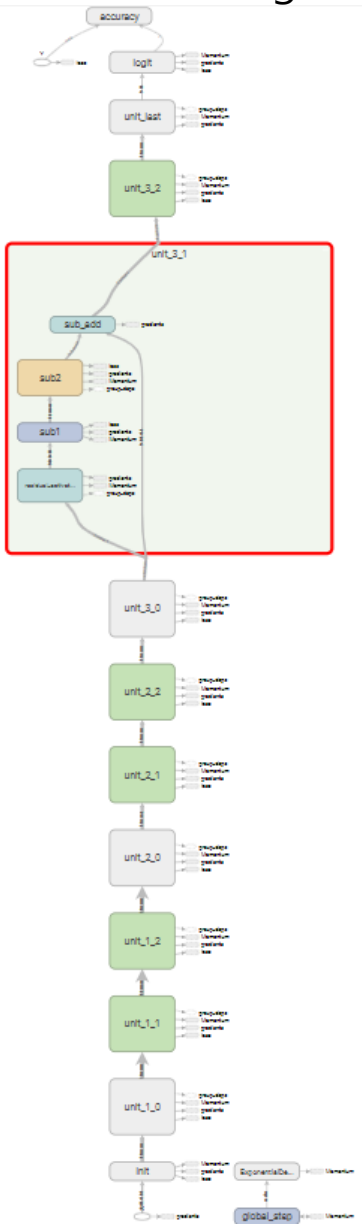
```
def batch_iter(x, y , batch_size, num_epochs, shuffle=True):
    num_batches_per_epoch = int((len(x) - 1) / batch_size) + 1
    for epoch in range(num_epochs):
        shuffle_indices = np.random.permutation(np.arange(len(x)))
        shuffled_x = x[shuffle_indices]
        shuffled_y = y[shuffle_indices]
        for batch_num in range(num_batches_per_epoch):
            start_index = batch_num * batch_size
            end_index = min((batch_num + 1) * batch_size, len(y))
            yield list(zip(data_augmentation(shuffled_x[start_index:end_index], 4),
shuffled_y[start_index:end_index]))
def data_augmentation (x_batch, padding=None):
    for i in range(len(x_batch)):
        if bool(random.getrandbits(1)):
            x_batch[i] = np.fliplr(x_batch[i])
    oshape = np.shape(x_batch[0])
    if padding:
        oshape = (oshape[0] + 2 * padding, oshape[1] + 2 * padding)
    new_batch = []
    npad = ((padding, padding), (padding, padding), (0, 0))
    for i in range(len(x_batch)):
        new_batch.append(x_batch[i])
        if padding:
            new_batch[i] = np.lib.pad(x_batch[i], pad_width=npad, mode='constant', constant_valu
        nh = random.randint(0, oshape[0] - 32)
        nw = random.randint(0, oshape[1] - 32)
        new_batch[i] = new_batch[i][nh:nh + 32, nw:nw + 32]
    return new_batch
```

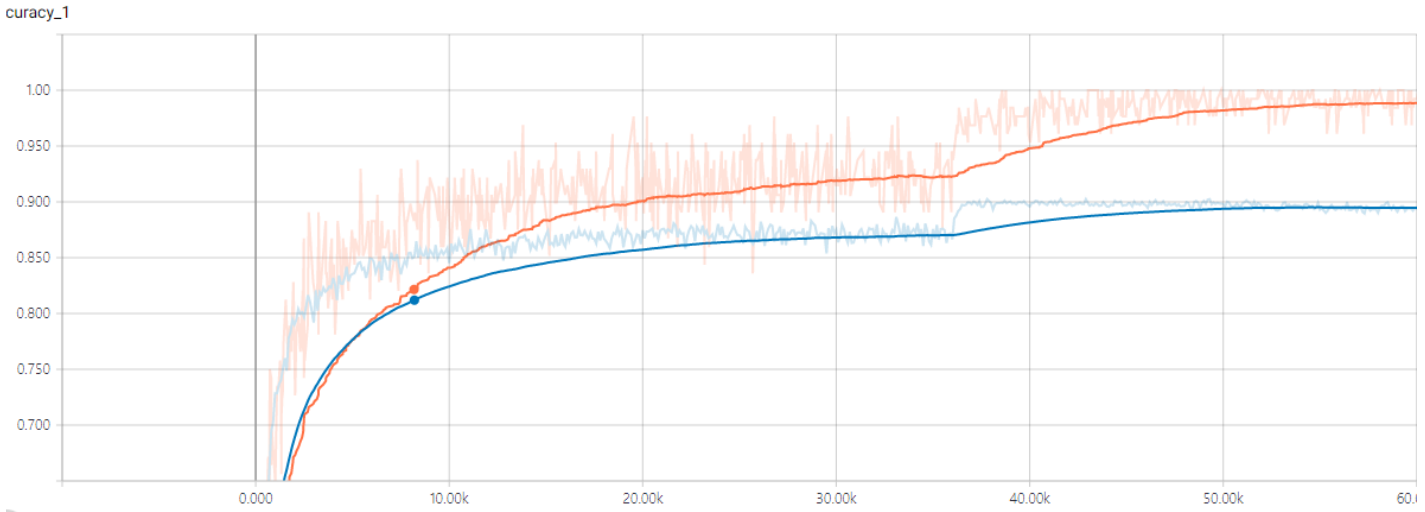
학습 데이터를 늘리기 위해
좌우 대칭 및 사진의 일부영역 잘라내기



Tensorboard

tensorboard --logdir=C:₩Users₩82102₩PycharmProjects₩aischool₩CIFAR₩runs₩1580556863





AI School 6기 6주차

CNN 모델을 활용한 영화리뷰 평점 예측

CNN for text classification

- CNN for sentence classification [Kim et al., 2014]

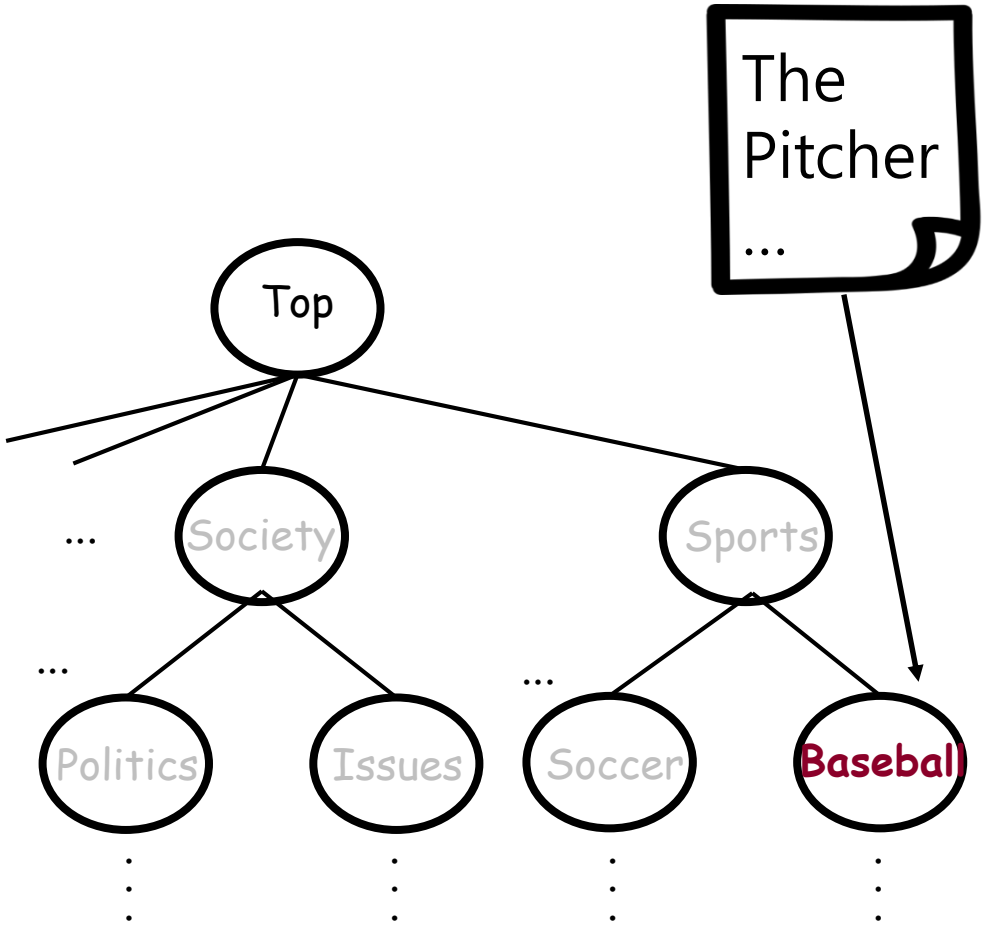
Five stars!

Positive
Negative

Economy
Sports
Weather

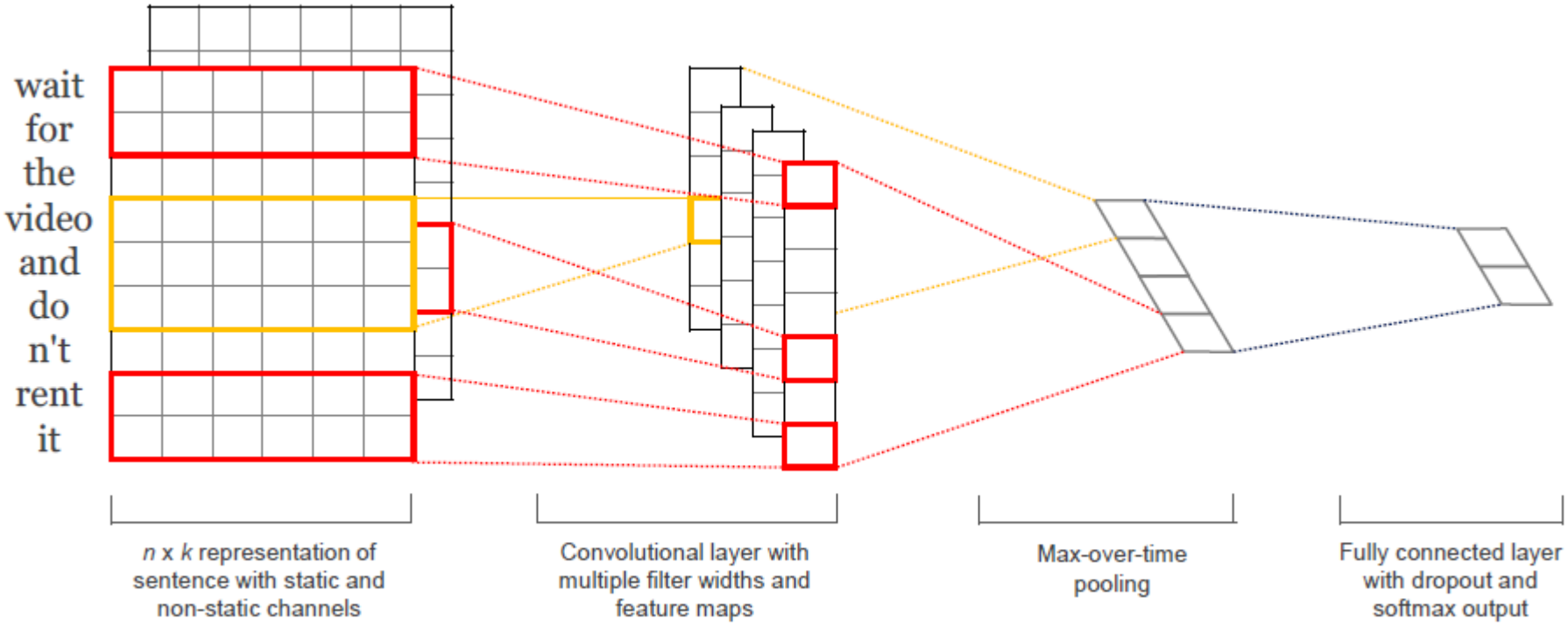
...

The Pitcher
...



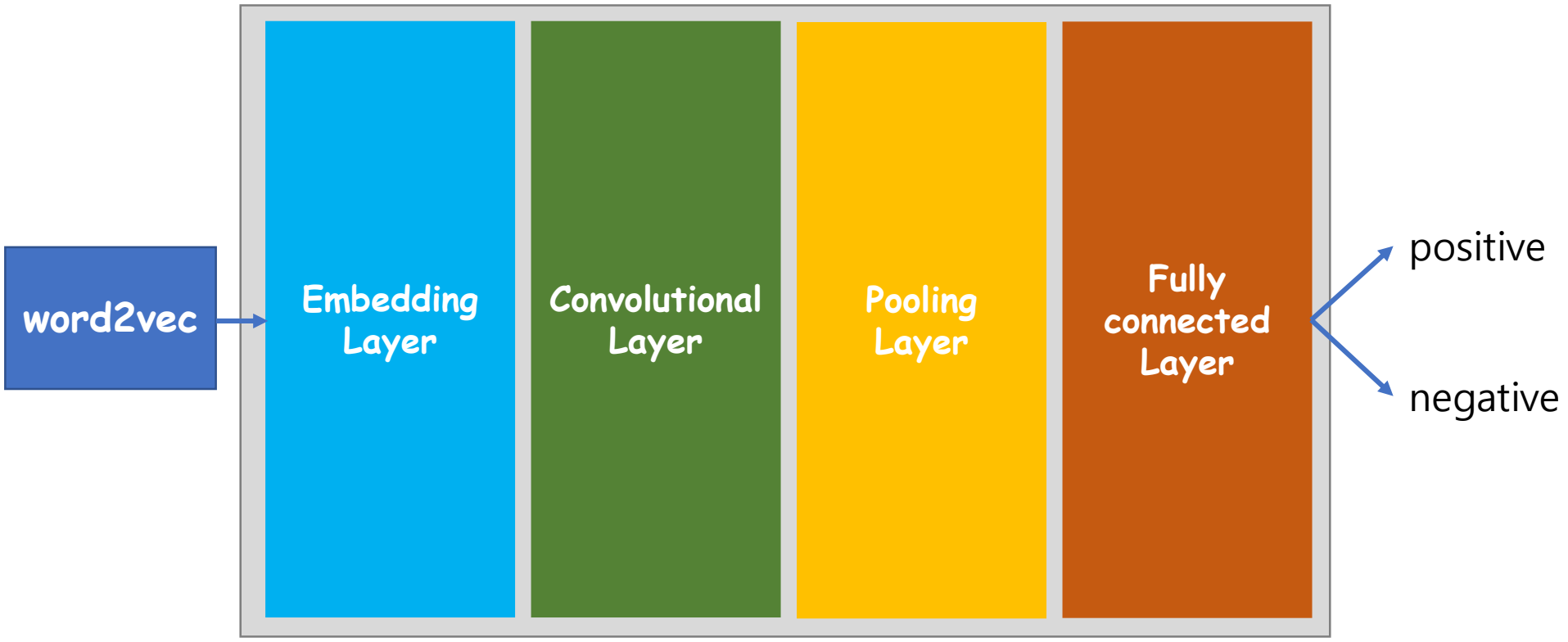
CNN for text classification

- Model overview



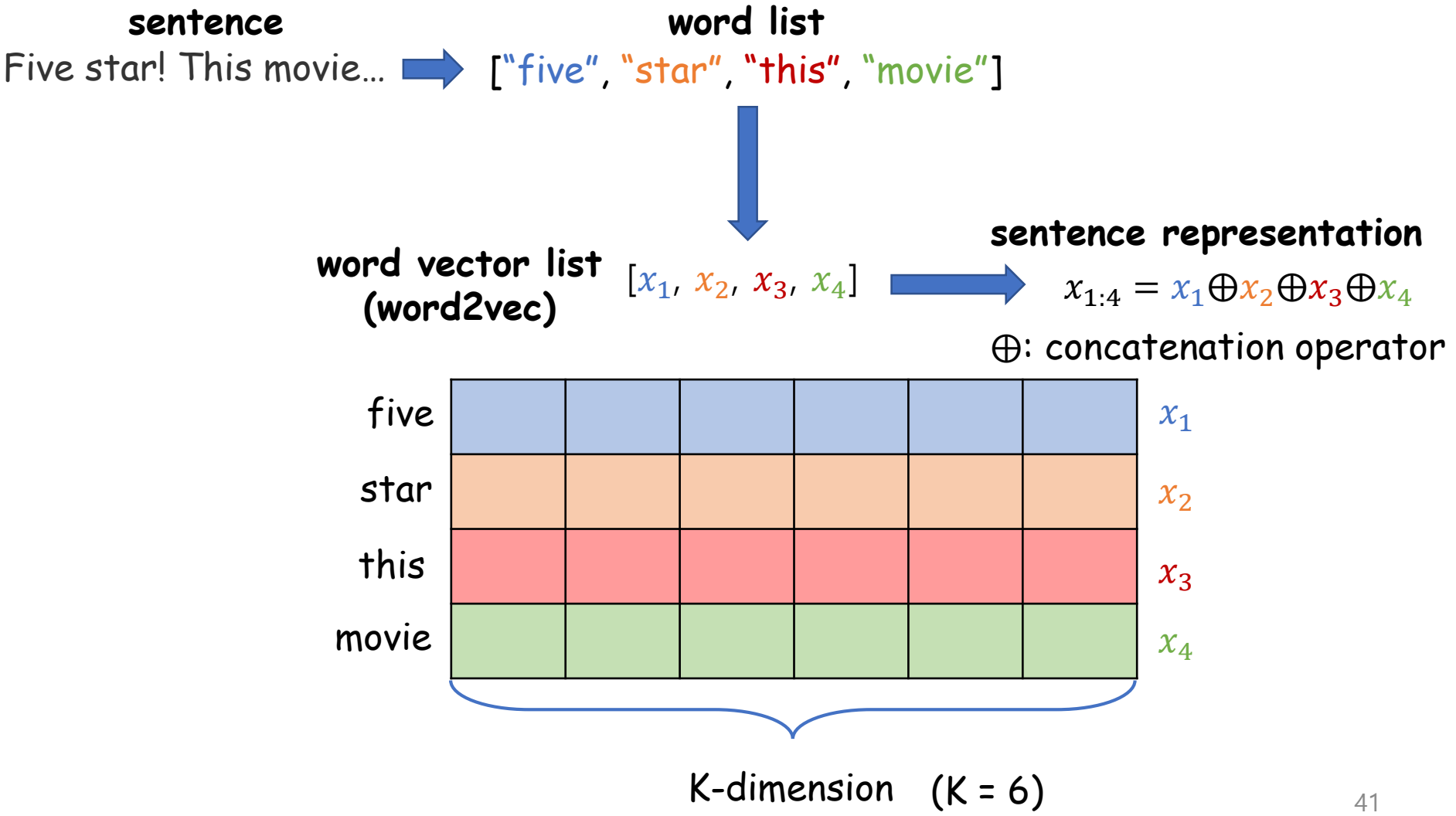
CNN for text classification

- Model overview



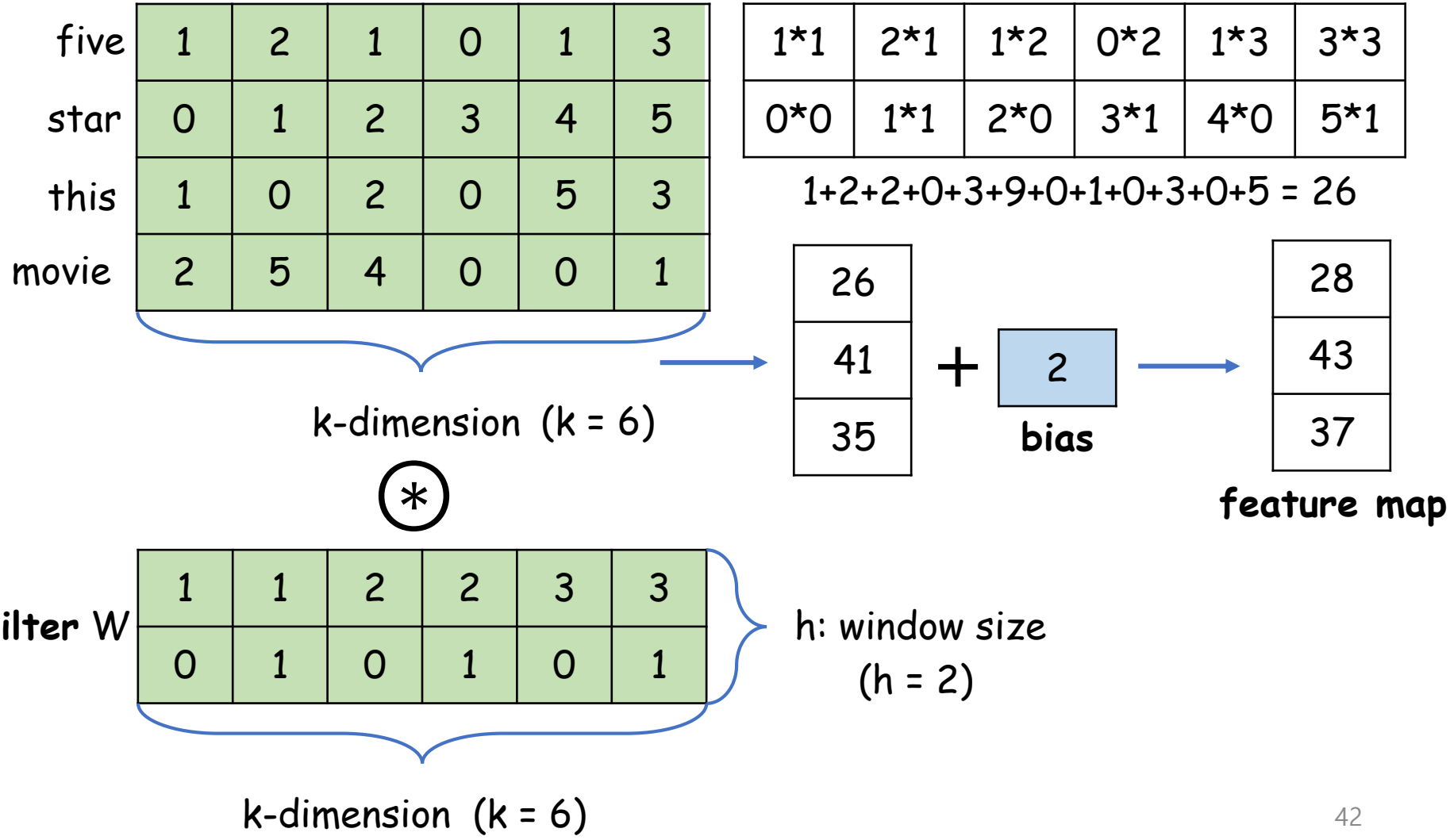
CNN for text classification

- Word2vec & Embedding layer



CNN for text classification

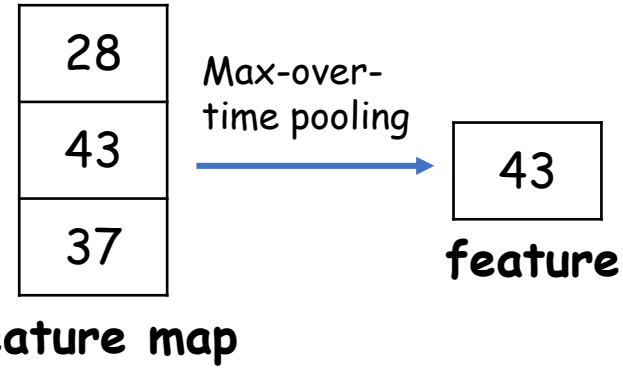
- Convolutional layer



CNN for text classification

- Pooling layer

five	1	2	1	0	1	3
star	0	1	2	3	4	5
this	1	0	2	0	5	3
movie	2	5	4	0	0	1

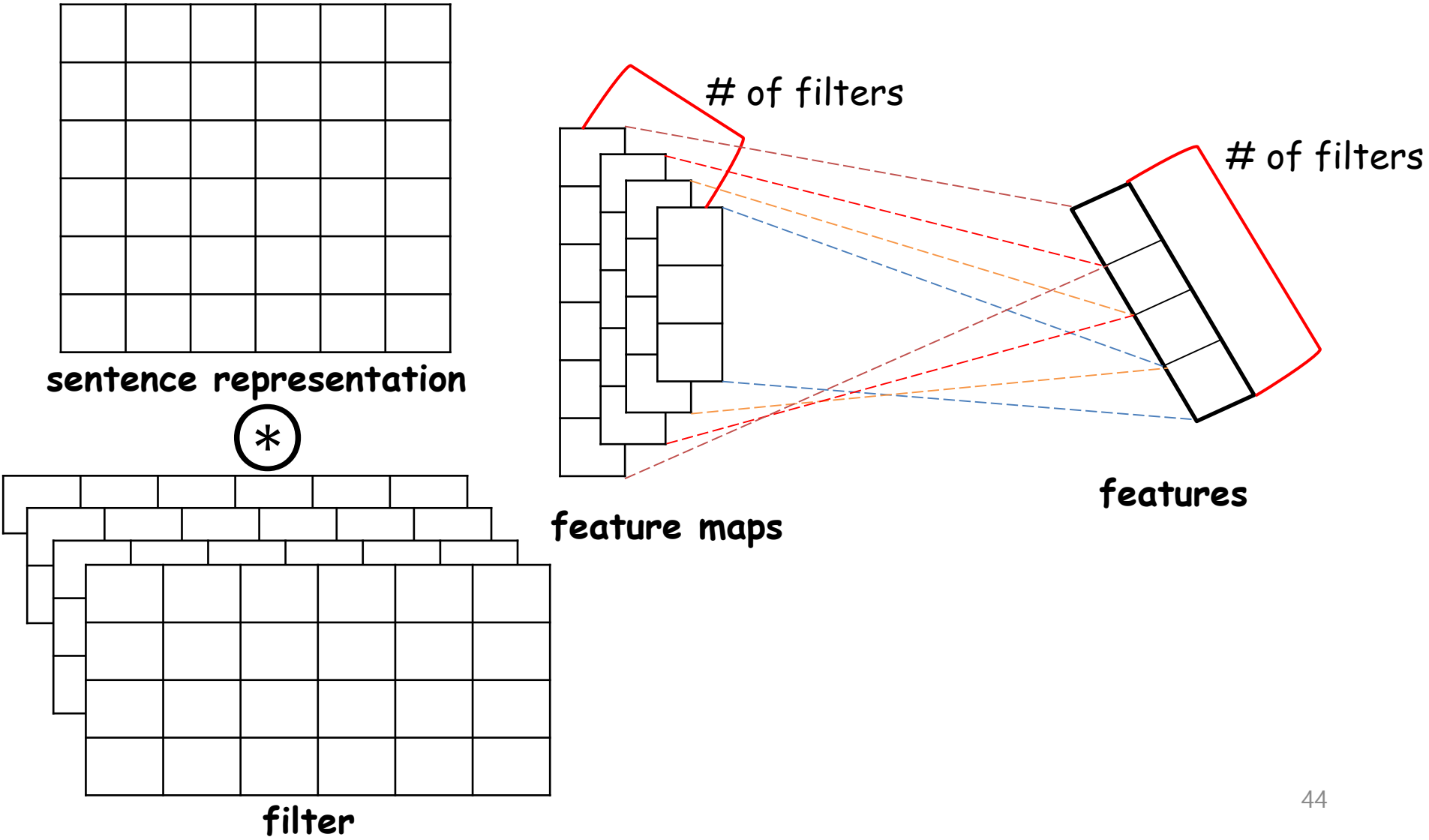


filter W	1	1	2	2	3	3
	0	1	0	1	0	1

One filter -> **One** feature
Multiple filter -> **Multiple** feature

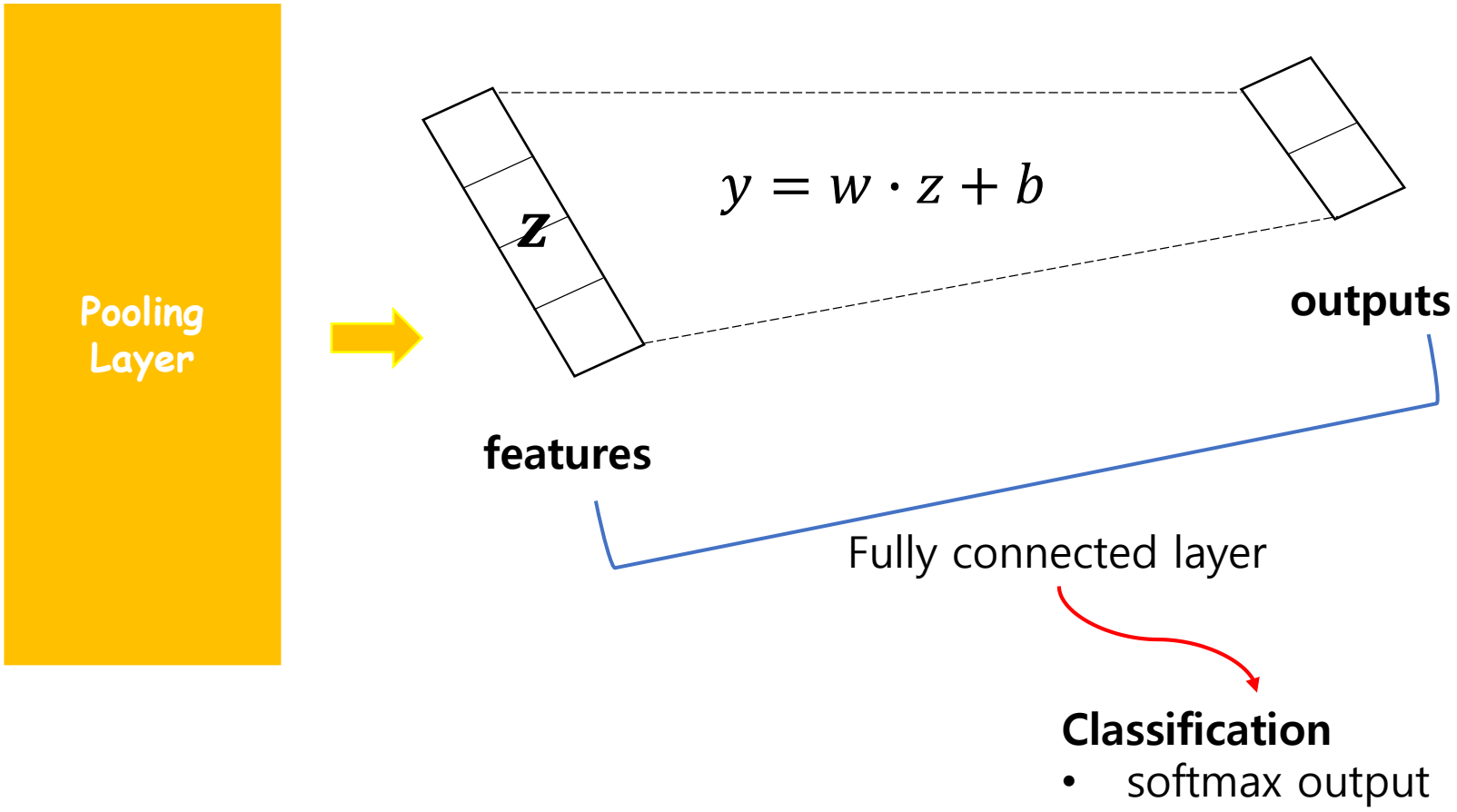
CNN for text classification

- Pooling layer



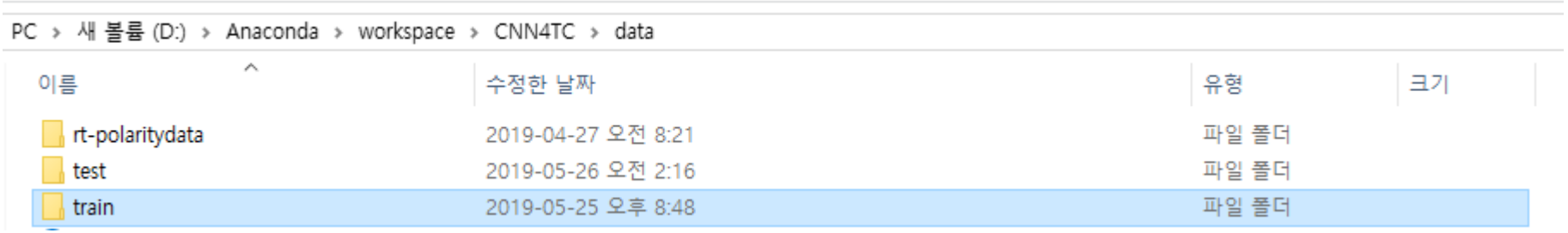
CNN for text classification

- Fully connected layer



CNN 분류기 학습

- 학습 데이터 다운로드



CNN 분류기 학습

- Import

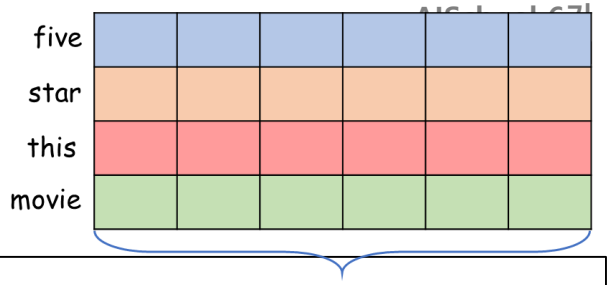
train.py

```
import tensorflow as tf
import numpy as np
import os
import time
import datetime
import CNN4TC.data_helpers as dh
from CNN4TC.text_cnn import TextCNN
from tensorflow.contrib import learn
```

data_helpers.py

```
import numpy as np
import re
import glob
```

Hyperparameters



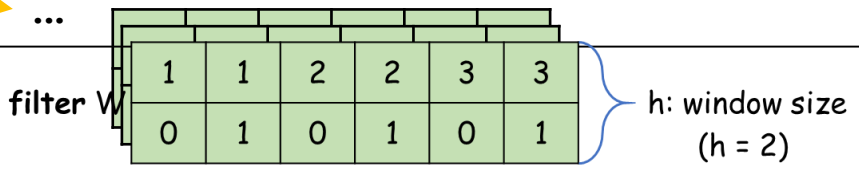
- train.py

```
# Data loading params
tf.flags.DEFINE_float("dev_sample_percentage", .1, "Percentage of the training data to use for validation")
tf.flags.DEFINE_string("imdb_pos_data_file", "./data/train/pos/*", "Data source for the positive data.")
tf.flags.DEFINE_string("imdb_neg_data_file", "./data/train/neg/*", "Data source for the negative data.")
# Model Hyperparameters
tf.flags.DEFINE_integer("embedding_dim", 128, "Dimensionality of word embedding (default: 128)")
tf.flags.DEFINE_string("filter_sizes", "2,3,4", "Comma-separated filter sizes (N-gram) (default: '3,4,5')")
tf.flags.DEFINE_integer("num_filters", 128, "Number of filters per filter size (default: 128)")
tf.flags.DEFINE_float("lr", 5e-4, "learning rate (default: 0.001)")
tf.flags.DEFINE_float("dropout_keep_prob", 0.6, "Dropout keep probability (default: 0.5)")
tf.flags.DEFINE_float("l2_reg_lambda", 0.000000001, "L2 regularization lambda (default: 0.0)")
# Training parameters
tf.flags.DEFINE_integer("batch_size", 64, "Batch Size (default: 64)")
tf.flags.DEFINE_integer("num_epochs", 20, "Number of training epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 100, "Evaluate model on dev set after this many steps (default:
100)")
tf.flags.DEFINE_integer("checkpoint_every", 100, "Save model after this many steps (iterations) (default: 100)")
tf.flags.DEFINE_integer("num_checkpoints", 3, "Number of checkpoints to store (default: 5)")
# Misc Parameters
tf.flags.DEFINE_boolean("allow_soft_placement", True, "Allow device soft device placement")
tf.flags.DEFINE_boolean("log_device_placement", False, "Log placement of ops on devices")
FLAGS = tf.flags.FLAGS
```

단어를 표현하는 벡터의 크기

Filter의 높이 or Window size or N-gram

Filter 종류별 개수



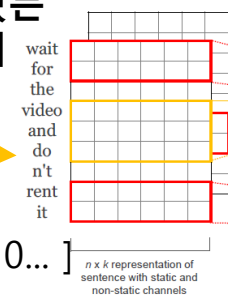
Data loading & preprocessing

- train.py

```
# Load data
print("Loading data...")
x_text, y = dh.load_imdb_data_and_labels(FLAGS.imdb_pos_data_file, FLAGS.imdb_neg_data_file)

# Build vocabulary
max_document_length = max([len(x.split(" ")) for x in x_text])
print("max_document_length: ", max_document_length) #298
vocab_processor = learn.preprocessing.VocabularyProcessor(max_document_length)
x = np.array(list(vocab_processor.fit_transform(x_text)))
np.random.seed(10)
shuffle_indices = np.random.permutation(np.arange(len(y)))
x_shuffled = x[shuffle_indices]
y_shuffled = y[shuffle_indices]
# Split train/test set
dev_sample_index = -1 * int(FLAGS.dev_sample_percentage * float(len(y)))
x_train, x_dev = x_shuffled[:dev_sample_index], x_shuffled[dev_sample_index:]
y_train, y_dev = y_shuffled[:dev_sample_index], y_shuffled[dev_sample_index:]

del x, y, x_shuffled, y_shuffled

print("Vocabulary Size: {:d}".format(len(vocab_processor.vocabulary_)))
print("Train/Dev split: {:d}/{:d}".format(len(y_train), len(y_dev)))
```

모델이 처리할 수 있는
문장의 최대 길이

five star this movie is good--> [8 379 3 47574 2 45 0 0 0 0 0 0 0 0 0 0...]



IMDB data loading

- data_helpers.py

```
def load_imdb_data_and_labels(pos_file, neg_file):
    # Load data from files
    pos_list = glob.glob(pos_file) #load file list
    pos_final = []# sentence list

    for pos in pos_list:
        x_text = list(open(pos, "r", encoding='UTF8').readlines())
        x_text = [clean_str(sent) for sent in x_text]
        pos_final = pos_final + x_text

    neg_list = glob.glob(neg_file)
    neg_final = []

    for neg in neg_list:
        x_text = list(open(neg, "r", encoding='UTF8').readlines())
        x_text = [clean_str(sent) for sent in x_text]
        neg_final = neg_final + x_text

    positive_labels = [[0, 1] for _ in pos_final] #[[0,1], [0,1], [0,1], [0,1], [0,1], [0,1],...]
    negative_labels = [[1, 0] for _ in neg_final] #[[1,0], [1,0], [1,0], [1,0], [1,0], [1,0],...]

    y = np.concatenate([positive_labels, negative_labels], 0) ##[[0,1], [0,1], [0,1], [0,1], [0,1], [0,1],...[1,0], [1,0]
    x_final = pos_final + neg_final
    return [x_final, y]
```

```
list = [[1,2,3], [3,6,9], [2,4,6]]
matrix = numpy.array(list)
```

```
matrix = numpy.zeros((3,3))
```

```
matrix =
numpy.random.rand(3,3)
```

```
matrix.shape
```

TextCNN class & input

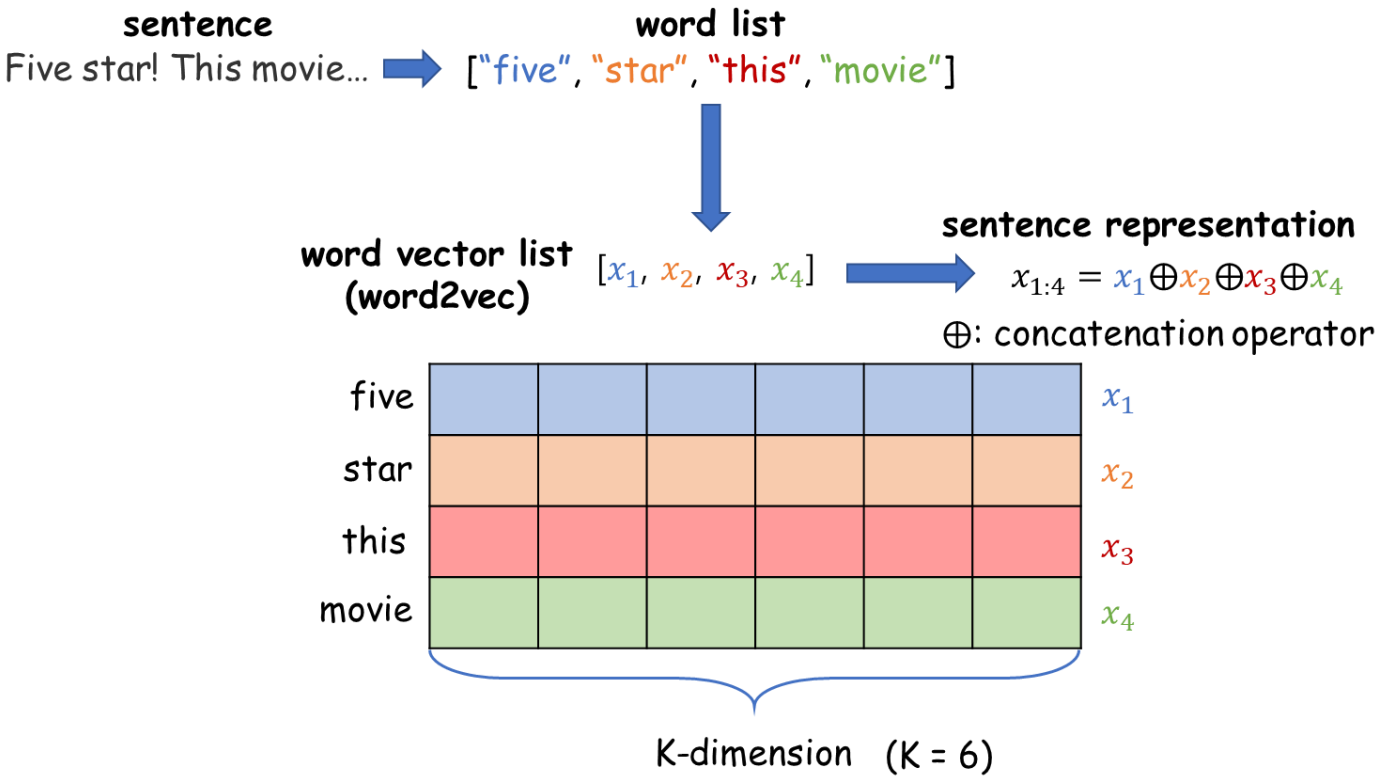
- text_cnn.py

```
import tensorflow as tf


class TextCNN(object):
    def __init__(self, sequence_length, num_classes, vocab_size, embedding_size, filter_sizes, num_filters,
l2_reg_lambda=0.0):

        # Placeholders for input, output and dropout
        self.input_x = tf.placeholder(tf.int32, [None, sequence_length], name="input_x")
        self.input_y = tf.placeholder(tf.float32, [None, num_classes], name="input_y")#pos: [0 1] neg: [1 0]
        self.dropout_keep_prob = tf.placeholder(tf.float32, name="dropout_keep_prob") #training: 0.5 #test:
1.0



        # Keeping track of l2 regularization loss (optional)
        l2_loss = tf.constant(0.0)
```

Embedding layer

- text_cnn.py

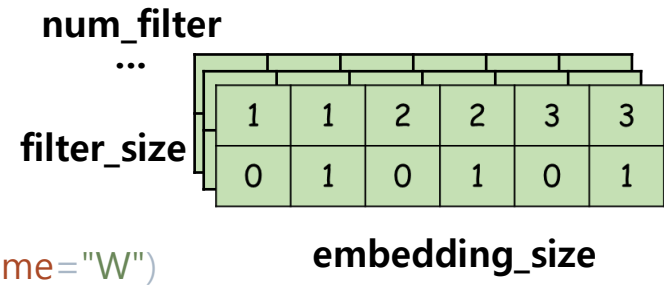
```
# Embedding layer
with tf.device('/gpu:0'), tf.name_scope("embedding"):
    self.W = tf.Variable(tf.random_uniform([vocab_size, embedding_size], -1.0, 1.0), name="W")
    self.embedded_chars = tf.nn.embedding_lookup(self.W, self.input_x) #[batch, embedding_dim_,
sequenth_length]
    self.embedded_chars_expanded = tf.expand_dims(self.embedded_chars, -1)#[batch, embedding_dim_,
sequenth_length, 1]
```

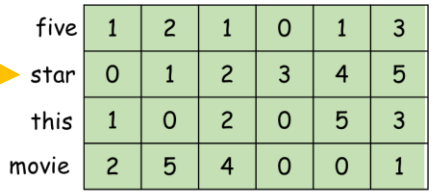


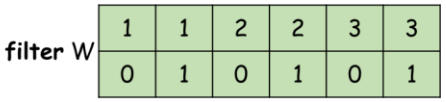
Convolutional layer

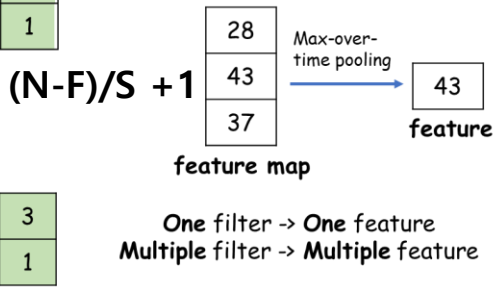
- text_cnn.py

```
# Create a convolution + maxpool layer for each filter size
pooled_outputs = []
for i, filter_size in enumerate(filter_sizes):
    with tf.name_scope("conv-maxpool-%s" % filter_size):
        # Convolution Layer
        filter_shape = [filter_size, embedding_size, 1, num_filters]
        W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W")
        b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b")
        conv = tf.nn.conv2d(
            self.embedded_chars_expanded,
            W,
            strides=[1, 1, 1, 1],
            padding="VALID",
            name="conv")
        # Apply nonlinearity
        h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu")
        # Maxpooling over the outputs
        pooled = tf.nn.max_pool(
            h,
            ksize=[1, sequence_length - filter_size + 1, 1, 1], #mnist [1,2,2,1]
            strides=[1, 1, 1, 1],
            padding='VALID',
            name="pool")
        pooled_outputs.append(pooled)
```





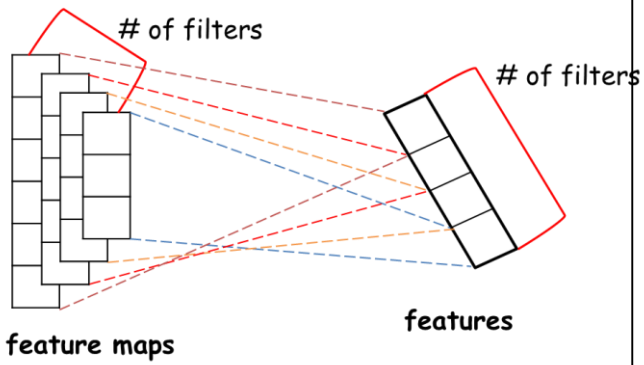


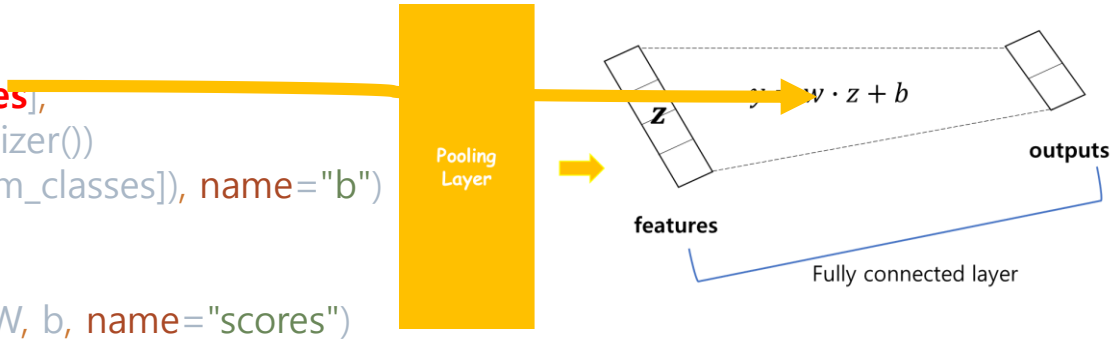


Fully connected layer

- text_cnn.py

```
# Combine all the pooled features
num_filters_total = num_filters * len(filter_sizes)
self.h_pool = tf.concat(pooled_outputs, 3)
self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total])

with tf.name_scope("dropout"):
    self.h_drop = tf.nn.dropout(self.h_pool_flat, self.dropout_keep_prob)

with tf.name_scope("output"):
    W = tf.get_variable(
        "W",
        shape=[num_filters_total, num_classes],
        initializer=tf.contrib.layers.xavier_initializer())
    b = tf.Variable(tf.constant(0.1, shape=[num_classes]), name="b")
    l2_loss += tf.nn.l2_loss(W)
    l2_loss += tf.nn.l2_loss(b)
    self.scores = tf.nn.xw_plus_b(self.h_drop, W, b, name="scores")
    self.predictions = tf.argmax(self.scores, 1, name="predictions")

with tf.name_scope("loss"):
    losses = tf.nn.softmax_cross_entropy_with_logits(logits=self.scores, labels=self.input_y)
    self.loss = tf.reduce_mean(losses) + l2_reg_lambda * l2_loss

with tf.name_scope("accuracy"):
    correct_predictions = tf.equal(self.predictions, tf.argmax(self.input_y, 1))
    self.accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"), name="accuracy")
```





Optimizer

- train.py

```
with tf.Graph().as_default():
    session_conf = tf.ConfigProto(
      allow_soft_placement=FLAGS.allow_soft_placement,
      log_device_placement=FLAGS.log_device_placement)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        cnn = TextCNN(
            sequence_length=x_train.shape[1],
            num_classes=y_train.shape[1],
            vocab_size=len(vocab_processor.vocabulary_),
            embedding_size=FLAGS.embedding_dim,
            filter_sizes=list(map(int, FLAGS.filter_sizes.split(","))),
            num_filters=FLAGS.num_filters,
            l2_reg_lambda=FLAGS.l2_reg_lambda)

        # Define Training procedure
        global_step = tf.Variable(0, name="global_step", trainable=False)
        optimizer = tf.train.AdamOptimizer(FLAGS.lr)
        #optimizer = tf.train.AdagradOptimizer(FLAGS.lr)
        grads_and_vars = optimizer.compute_gradients(cnn.loss)
        train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)
```

Summaries

- train.py

```
# Keep track of gradient values and sparsity (optional)
grad_summaries = []
for g, v in grads_and_vars:
    if g is not None:
        grad_hist_summary = tf.summary.histogram("{}/grad/hist".format(v.name), g)
        sparsity_summary = tf.summary.scalar("{}/grad/sparsity".format(v.name), tf.nn.zero_fraction(g))
        grad_summaries.append(grad_hist_summary)
        grad_summaries.append(sparsity_summary)
grad_summaries_merged = tf.summary.merge(grad_summaries)

# Output directory for models and summaries
timestamp = str(int(time.time()))
out_dir = os.path.abspath(os.path.join(os.path.curdir, "runs", timestamp))
print("Writing to {}₩n".format(out_dir))
# Summaries for loss and accuracy
loss_summary = tf.summary.scalar("loss", cnn.loss)
acc_summary = tf.summary.scalar("accuracy", cnn.accuracy)
# Train Summaries
train_summary_op = tf.summary.merge([loss_summary, acc_summary, grad_summaries_merged])
train_summary_dir = os.path.join(out_dir, "summaries", "train")
train_summary_writer = tf.summary.FileWriter(train_summary_dir, sess.graph)
# Dev summaries
dev_summary_op = tf.summary.merge([loss_summary, acc_summary])
dev_summary_dir = os.path.join(out_dir, "summaries", "dev")
dev_summary_writer = tf.summary.FileWriter(dev_summary_dir, sess.graph)
```

Checkpoint, Save vocabulary

- train.py

```
# Checkpoint directory. Tensorflow assumes this directory already exists so we need to create it
checkpoint_dir = os.path.abspath(os.path.join(out_dir, "checkpoints"))
checkpoint_prefix = os.path.join(checkpoint_dir, "model")
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)
saver = tf.train.Saver(tf.global_variables(), max_to_keep=FLAGS.num_checkpoints)

# Write vocabulary
vocab_processor.save(os.path.join(out_dir, "vocab"))

# Initialize all variables
sess.run(tf.global_variables_initializer())
```

Train & dev step

- train.py

```
def train_step(x_batch, y_batch):
    feed_dict = {
      cnn.input_x: x_batch,
      cnn.input_y: y_batch,
      cnn.dropout_keep_prob: FLAGS.dropout_keep_prob
    }
    _, step, summaries, loss, accuracy = sess.run(
        [train_op, global_step, train_summary_op, cnn.loss, cnn.accuracy],
        feed_dict)
    time_str = datetime.datetime.now().isoformat()
    print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
    train_summary_writer.add_summary(summaries, step)
def dev_step(x_batch, y_batch, writer=None):
    feed_dict = {
      cnn.input_x: x_batch,
      cnn.input_y: y_batch,
      cnn.dropout_keep_prob: 1.0
    }
    step, summaries, loss, accuracy = sess.run(
        [global_step, dev_summary_op, cnn.loss, cnn.accuracy],
        feed_dict)
    time_str = datetime.datetime.now().isoformat()
    print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
    if writer:
        writer.add_summary(summaries, step)
    return accuracy
```

Batch, Training

- train.py

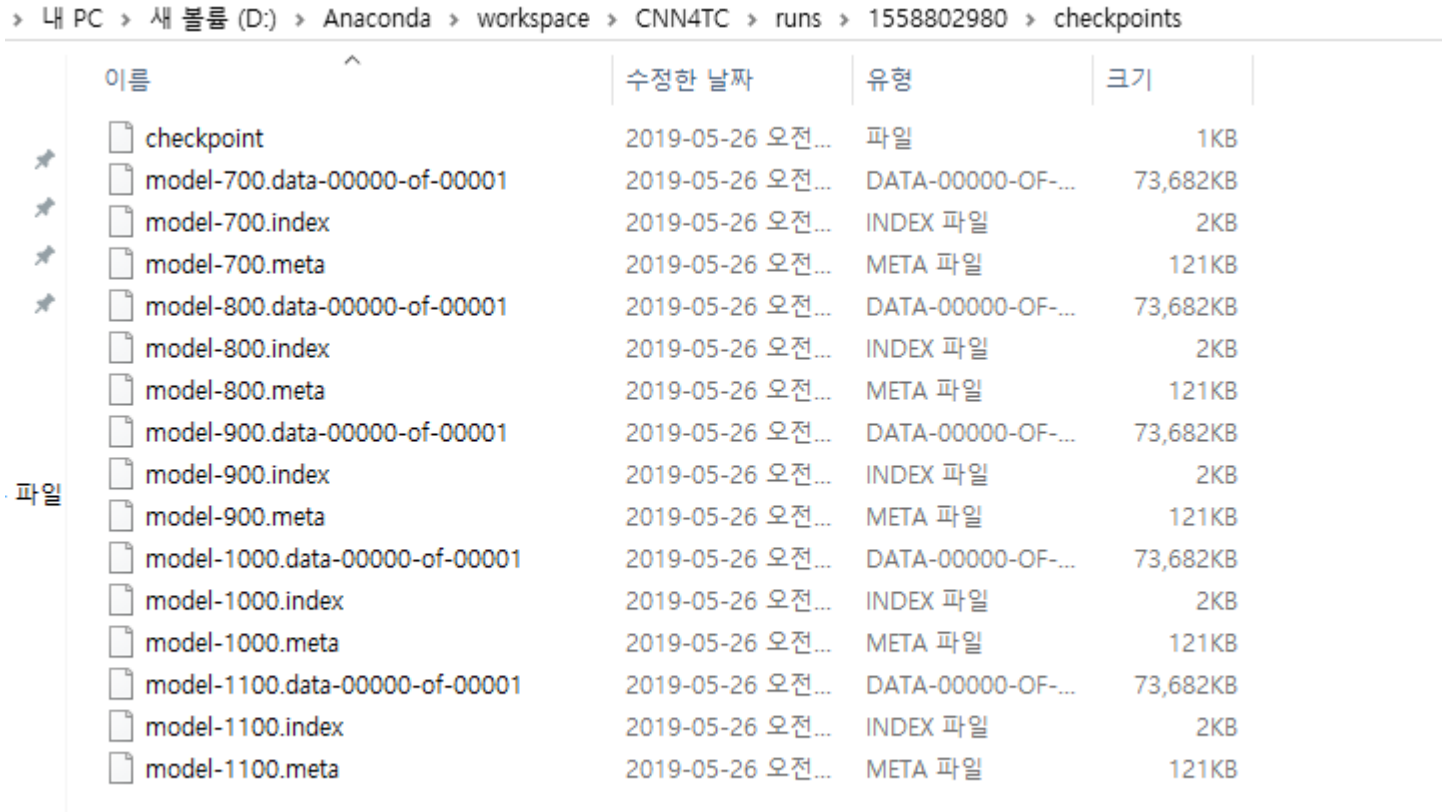
```
batches = dh.batch_iter(list(zip(x_train, y_train)), FLAGS.batch_size, FLAGS.num_epochs)
max = 0
for batch in batches:
    x_batch, y_batch = zip(*batch)
    train_step(x_batch, y_batch)
    current_step = tf.train.global_step(sess, global_step)
    if current_step % FLAGS.evaluate_every == 0:
        accuracy = dev_step(x_dev, y_dev, writer=dev_summary_writer)
        if accuracy > max:
            max = accuracy
            path = saver.save(sess, checkpoint_prefix, global_step=current_step)
            print("Saved model checkpoint to {}₩n".format(path))
```

- data_helpers.py

```
def batch_iter(data, batch_size, num_epochs, shuffle=True):
    data = np.array(data)
    data_size = len(data)
    num_batches_per_epoch = int((len(data)-1)/batch_size) + 1
    for epoch in range(num_epochs):
        if shuffle:
            shuffle_indices = np.random.permutation(np.arange(data_size))
            shuffled_data = data[shuffle_indices]
        else: shuffled_data = data
        for batch_num in range(num_batches_per_epoch):
            start_index = batch_num * batch_size
            end_index = min((batch_num + 1) * batch_size, data_size)
            yield shuffled_data[start_index:end_index]
```

CNN 분류기 학습

- ./run/생성시간/checkpoints- 학습 모델 확인




```
Evaluation:
2019-05-26T02:22:50.357046: step 1200, loss 0.37375, acc 0.838
```

CNN 분류기 평가

- eval.py

```
tf.flags.DEFINE_string("imdb_pos_data_file", "./data/test/pos/*", "Data source for the positive data.")
tf.flags.DEFINE_string("imdb_neg_data_file", "./data/test/neg/*", "Data source for the negative data.")
tf.flags.DEFINE_string("checkpoint_dir", "./runs/1522115047/checkpoints", "Checkpoint directory from training run")
tf.flags.DEFINE_boolean("eval_test", True, "Evaluate on all test data")
x_raw, y_test = dh.load_imdb_data_and_labels(FLAGS.imdb_pos_data_file, FLAGS.imdb_neg_data_file)
```

CNN 분류기 평가 (실제 imdb data)

- Imdb_eval.py

```
tf.flags.DEFINE_string("real_imdb_x_data_file", "./data/review.txt", "Data source for the positive data.")
tf.flags.DEFINE_string("real_imdb_t__data_file", "./data/score.txt", "Data source for the negative data.")
tf.flags.DEFINE_string("checkpoint_dir", "./runs/1558802980/checkpoints", "Checkpoint directory from training run")

x_raw, y_test = dh.load_real_imdb_data_and_labels(FLAGS.real_imdb_x_data_file, FLAGS.real_imdb_t_data_file)
```

CNN 분류기 평가 (실제 imdb data)

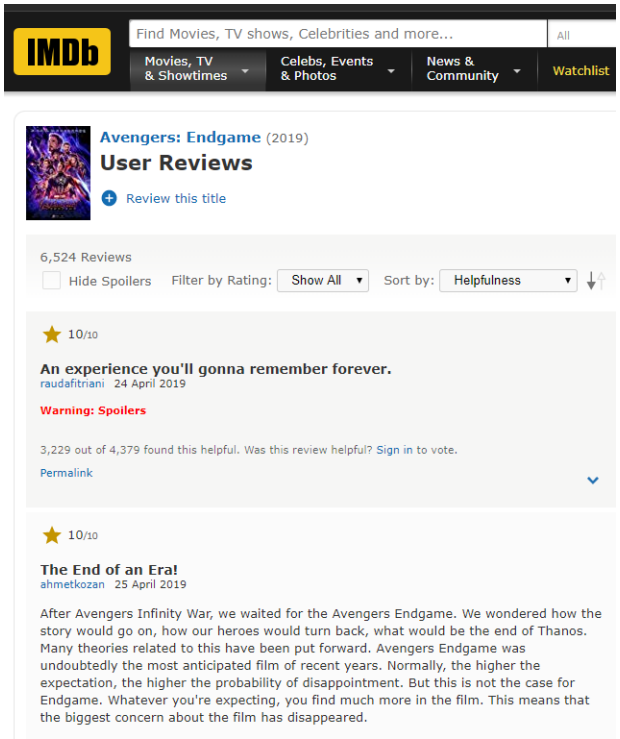
- data_helpers – 함수 추가

```
def load_real_imdb_data_and_labels(text_data_file, score_data_file):
    text_list = list(open(text_data_file, "r", encoding='utf-8').readlines())
    text_list = [s.strip() for s in text_list]
    score_list = list(open(score_data_file, "r", encoding='utf-8').readlines())
    score_list = [s.strip() for s in score_list]

    x_text = [clean_str(sent) for sent in text_list]
    print(score_list)
    y = []
    for score in score_list:
        if int(score) > 5:
            y.append([0, 1])
        else:
            y.append([1, 0])
    print(y)
    return [x_text, y]
```

CNN 분류기 평가 (실제 imdb data)

- Imdb_eval.py – 분류 결과 확인

```
predictions_human_readable = np.column_stack((np.array(x_raw), y_test,
all_predictions))
out_path = os.path.join(FLAGS.checkpoint_dir, "..", "prediction_imdb.csv")
```



an experience you 'll gonna remember forever there is no way that i could describe my emotions for this n	1	1
the end of an era ! after avengers infinity war , we waited for the avengers endgame we wondered how th	1	1
the ending made all 22 movies worth it if you 're going to watch this movie , avoid any spoilers , even spo	1	0
overall an epic conclusion but inferior to infinity war 's pace and energy ₩(spoiler free ₩) this is probably	1	1
amazing , but the more i dwell on it the worse it becomes i have to say , my first reaction walking out of t	1	1
the writers got carried away , the directors over reached and the studio sacrificed the integrity of their proc	0	1
overhyped infinity war was way better unpopular opinion let me start by saying that endgame is not a bac	1	0
dug themselves a hole they could n't get out of first of , infinity war was the greatest mcu movie ever mad	1	0
so much potential lost after infinity war i had huge expectations unfortunately i think it does n't follow the	1	0
absolute perfection end game ₩(score 10 10 ₩) absolute perfection end game ! ! good acting performanc	1	1
good but a mess in the story department a nice production for those only interested in action sequences b	0	0
emotional but bit messy so it all ends i have loved most of the mcu , some have been too predictable but	1	1
the best marvel superhero movie ever and also the epic conclusion to the 21 movies in the mcu avengers	1	1
all the epicness and efforts of 20 movies ruined with single snap are you moviegoer ₩? waited to long for	0	0
perfection unbelievable for marvel to pull this tour de force of a film is unreal 21 films 21 films culminatin	1	1

Q&A