■ 텐서 플로우란 ?

텐서 플로우(TensorFlow) 는 기계학습과 딥러닝을 위해 구글에서 만든 오픈 소스 라이브러리이다.

★ 텐서 플로우의 장점
↓ ↓
다차원 배열의 흐름 (4차원 배열의 연산(계산)을 빠르게 할 수 있게금 구현이되어짐)

1. 코드가 간결해진다.
2. 신경망 구현에 필요한 모든 함수들이 다 내장 되어있다.
3. 속도가 빠르다. (코딩도 빨라지고 실행 빠르다)
4. GPU 를 사용할 수 있다.

한국시간으로 2016년 11월 29에 TensorFlow v0.12.0 RC0 이 업데이트 되었고 2016년 11월 29일 나온 버젼의 핵심 변경사항은 window 에서 GPU 버젼의 텐써 플로우를 지원한다는 것이었다. 예전에는 Ubuntu 에서만 가능하던 GPU 버젼도 윈도우에서 설치가 가능하게 되었다.

★ 텐써 플로우 코드의 구조

모델을 생성하는 부분
- 오퍼레이션
- 변수

모델을 실행하는 부분
- 세션

★ 텐서 플로우 용어 설명

1. 오퍼레이션 (Operation)

그래프 상의 노드는 오퍼레이션(줄임말 op) 로 불린다. 오퍼레이션은 하나 이상의 텐서를 받을 수 있다. 오퍼레이션은 계산을 수행하고, 결과를 하나 이상의 텐서로 반환 할 수 있다.

2. 텐써 (Tensor)

내부적으로 모든 데이터는 텐써를 통해 표현된다. 텐써는 일종의 다차원 배열인데, 그래프 내의 오퍼레이션간에 텐써가 전달 된다.

3. 세션 (Session)

그래프를 실행하기 위해서는 세션 객체가 필요하다. 세션은 오퍼레이션의 실행환경을 캡슐화 한것이다.

모델을 생성하는 부분 <-- 그래프를 그리는 부분
- 오퍼레이션
- 변수

모델을 실행하는 부분 <-- 만들어진 그래프에 데이터를
- 세션 주입하는 부분

4. 변수 (Variable)

변수는 그래프의 실행시, 파라미터를 저장하고 갱신하는데 사용된다.
메모리상에서 텐서를 저장하는 버퍼 역활을 한다.

■ 신경망에 입력할 데이터들

1. mnist
   ↓
2. cifar10
   ↓
3. 개/고양이
   ↓
4. 정상폐/폐결절
   ↓
5. 이파리 사진 (상품의 표지의 기스 여부 확인)

예제1.

```
import  tensorflow  as  tf

sess = tf.Session()  # 그래프를 실행할 세션을 구성한다.

hello = tf.constant('Hello, Tensorflow')
```

변수를 정의하는 영역

변수를 실행하는 영역
위에서 변수를 정의했지만, 실행은 정의한 시점에서 실행되는
것은 아니다.

session 객체와 run 메소드를 사용할때 계산이 되어 실행된다.

```
print ( sess.run(hello))
print (str(sess.run(hello), encoding="utf-8")
```

b'Hello , Tensorflow'

Hello, Tensorflow

※ 파이썬 3버젼은 문자열 unicode 가 기본이므로 str 에서 encoding 처리를 해줘야 binary 타입을 unicode 타입으로 반환한다.

■ 텐써 플루우 기본 실습 두번째 예제

```
import  tensorflow  as  tf # tensorflow 모듈을 가져와서 tf 호출

x = tf.constant(35, name='x') # x라는 상수값을 만들고
                          # 숫자 35를 지정
y = tf.Variable( x +  5, name='y') # y 라는 변수를 만들고
                             # 방정식 x+5 로 정의함
model = tf.global_variables_initializer()

# global_variables_initializer() 로 변수를 초기화 하겠다.

# 그래프를 그리는 영역(빌딩 building 영역)
# ------------------------------------------------
# 그래프를 실행하는 영역

sess = tf.Session()  # 그래프를 실행할 세션을 구성한다.
sess.run(model) # 변수를 초기화 하겠다고 정의한 model 를
             # 실행하겠다.

print (sess.run(y))
```

```
40
```

■ 텐써 플로우 세번째 예제

문제 1 ~ 3

```
import tensorflow as tf

a = tf.constant(10)
b = tf.constant(32)
c = tf.add(a,b)
print(c)

Tensor("Add_1:0", shape=(), dtype=int32)
```

■ 파이썬 기본문법과 텐써 플로우 기본 문법을 비교

문제 4 ~ 6

1 에서 5까지의 숫자를 출력한다.

- 파이썬 기본 문법

```
x = 0
for i in range(5):
    x = x + 1
    print (x)
```

- 텐써 플로우

```
import tensorflow as tf

x = tf.Variable(0)

model = tf.global_variables_initializer()
# 이전에 생성했던 변수를 초기화 하는게 아니라
# 변수를 생성했으면 무조건 초기화를 해줘야 실행이 된다.

sess = tf.Session()
sess.run(model)

for i in range(5):
    x = x + 1
    print (sess.run(x))

sess.close()
```

■ numpy와 tensorflow문법 비교

문제 7 ~ 15

■ mnist데이터로 단층 신경망 구현하기

문제 16 ~ 31

■ Tensorflow로 구현하는 비용함수

문제 32

1. 최소 제곱 오차함수 (mean square error)
   loss = tf.square(y_predict, y_label)

2. 교차 엔트로피 함수 (cross entropy error)
   loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)

■ 경사 감소법을 Tensorflow로 구현하는 방법

문제 33 ~ 35

```
##############################################

# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)

        SGD :
         미니배치만큼 랜덤으로 데이터를 추출해서 확률적으로 경사를 감소하여
         global minima 로 찾아가는 방법

        단점 :  Local minima 에 잘 빠진다.


# optimizer = tf.train.AdagradOptimizer(learning_rate=0.01)

        러닝 레이트가 학습되면서 자동 조절되는 경사감소법


# optimizer = tf.train.MomentumOptimizer(learning_rate=0.01)

        관성을 이용해서 local minima 에 안빠지게 하는 경사감소법


# optimizer = tf.train.AdamOptimizer(learning_rate=0.01)

        Adagrade 의 장점 +  Momentum 의 장점

##############################################
```

■ 4. 텐서 플로우로 다층 신경망 구성

* Underfitting을 막을 수 있는 방법
    1. 가중치 초기화
    2. 배치 정규화

* Overfitting을 막을 수 있는 방법
    1. 드롭아웃

★ 텐서 플로우에서 가중치 초기화 하는 방법
문제 36 ~ 38
    1. Xavier —▶ 1 / √n
    2. He —▶ √2/n

```
################################################
# 가중치 초기화 방법
# W = tf.Variable(tf.random_uniform([784,10], -1, 1))
# W = tf.get_variable(name="W", shape=[784, 10], initializer=tf.contrib.layers.xavier_initializer())
# xavier 초기값
# W = tf.get_variable(name='W', shape=[784, 10], initializer=tf.contrib.layers.variance_scaling_initializer())
# he 초기값
# b = tf.Variable(tf.zeros([10]))
################################################
```

★ 텐서플로우로 배치정규화 구현
문제 39
    배치정규화 ?
        신경망 학습시 가중치의 값의 데이터가 골고루 분산될 수 있도록 하는 것을 강제하는 장치

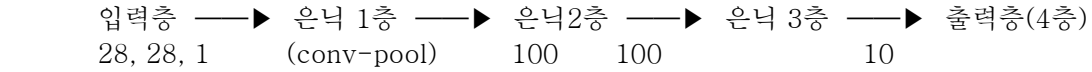
    구현 코드 ?
        batch_z1 = tf.contrib.layers.batch_norm(z1,True)

★ 훈련하는 신경망에 테스트를 하는 코드를 추가
문제 40 ~ 41

★ 오버피팅이 발생하지 않도록 drop out을 적용하는 방법
문제 42 ~ 43

■ 텐서 플로우로 CNN 구현하기

문제 44 ~ 44



■ cifar10 데이터를 신경망에 로드하는 데이터 전처리 코드 작성

문제 45 ~ 49

* cifar10 데이터 소개

cifar10은 총 60000개의 데이터 셋으로 이루어져 있으며 그 중 50000개 훈련데이터 이고 10000개가 테스트 데이터이다.

class는 비행기부터 트럭까지 10개로 구성되어 있다.

1. 비행기
2. 자동차
3. 새
4. 고양이
5. 사슴
6. 개
7. 개구리
8. 말
9. 양
10. 트럭

★ cifar10 이미지 데이터를 신경망으로 로드하는 함수 생성

1. 데이터를 신경망으로 로드하는 데이터 전처리 코드
   인터넷으로 찾기 어려우므로 자신이 직접 짜야한다.
   - 훈련 이미지 데이터를 numpy array 숫자로 변환
   - 훈련 이미지의 라벨을 one hot encoding 하는 방법

2. 신경망 코드 <------- 인터넷에서 쉽게 구할 수 있는 코드

★ 훈련 이미지의 라벨을 one hot encoding 하는 방법

문제 52 ~ 58

★ 이미지를 신경망에 로드하기 위해 만들어야 하는 함수 4가지

문제 59 ~ 66

1. image_load : 훈련 이미지와 테스트 이미지 로드하는 함수
2. label_load : 훈련 라벨과 테스트 라벨을 로드하는 함수
3. next_batch : 훈련 이미지 데이터를 100개씩 신경망에 입력하는 함수
4. shuffle_batch : data를 shuffle하는 함수

예 :

```
trainX=loader2.image_load(train_image)
trainY=loader2.label_load(train_label)
testX=loader2.image_load(test_image)
testY=loader2.label_load(test_label)
```

★ cifar10 신경망 구현

문제 65

1. cifar10데이터로 Vgg신경망 구현 훈련 정확도 93% 테스트 정확도 89%

■ 개 / 고양이 이미지 분류

문제 67 ~ 74

* 개고양이 분류 데이터 전처리 함수 4가지
    1. image_load
    2. label_load
    3. next_batch
    4. shuffle_batch

■ 이미지를 회전시켜 데이터를 늘리는 방법

문제 75 ~ 77

예제 1.png 고양이 사진을 open_CV.ipynb.txt를 이용해서 회전시오 !

```
import cv2 as cv
import matplotlib.pyplot as plt

img = plt.imread("C:₩₩1.jpg")
img = cv.rotate(img, cv.ROTATE_90_CLOCKWISE)
#img = cv.rotate(img, 2)
plt.imshow(img)
```

■ 개/고양이 사진에서 컴퓨터가 분류하기 어려워하는 사진 골라내는 작업

문제 89 ~ 95

" 분류가 잘 안되는 사진을 따로 골라내서 그 사진들은 사람이 분류하게 하고 분류가 잘되는 수많은
사진들은 컴퓨터가 분류하겠금 코드를 구현 "

1. 소프트 맥스 함수를 통과한 결과

개사진 --> 신경망 --> [0.2, 0.8]
↓ ↓
개 고양이

사람도 분류하기 어려운 사진 --> 신경망 -->

■ 훈련할때 98 % 이상 정확도를 보였던 모델이 테스트할 때 멍청해진 이유?

문제 98

```
training = tf.placeholder(tf.bool, name='training' )

batch_z4 = tf.contrib.layers.batch_norm(y4, scale=True, is_training=training) # 배치정규화
```

맨 아래 코드 :

```
print( sess.run(y_hat, feed_dict={x:test_txs, keep_prob:1.0, training:False}) )
```

■ 딥러닝 마무리 목표

1. 모델 저장과 모델 불러오는 방법
2. 배치 정규화를 훈련때는 켜고 테스트때 끄는 방법
3. vgg9 모델을 가지고 개/고양이 사진을 분류:
   정확도 : 훈련 100%, 테스트 : 90%
   " 포트폴리오를 빛내기 위한 Tip:
   컴퓨터가 분류하기 어려운 사진들은 사람이 분류할 수 있도록 별도의 폴더로 사진을 옮기는 코드 구현 "

■ 수아랩에서 이미지 데이터 전처리

문제 99 ~ 101

1. 훈련하는 신경망에 입력할 이미지 중에 아웃라이어가 섞여있으면 아무리 학습해도 정확도가 더 개선이 안된다. 아웃라이어 데이터를 제거하는 작업
   사람이 작업한다. 아르바이트 생, 하루에 5만장 정도 직접 본다.
   ↓
아웃라이어 데이터를 제거하는 자동화 코드 구현 (최근에)
↓ [0.51, 0.49] <-- 애매한 사진 사람사진(라벨 -> 0) ---> [0.2, 0.8]
개 ↑ 테스트 데이터에서 못맞춘 사진들을
따로 분석한다.
별도의 폴더에 들어간 사진이 전혀 다른 사진이다.
즉 사람 사진이다 라면 아예 삭제해 버린다.

개 사진인데 확률이 0.2 이하로 나와서 신경망이 계속 못맞추면
그 사진과 비슷한 사진을 더 모아서 훈련 데이터에 입력한다.
(회전, 명암조절등을 해서 비슷한 사진의 데이터를 늘린다)

수업 --> 폐사진 문제. 이파리

■ 정상 폐사진 vs 폐결절 사진 분류

문제 102 ~ 104

■ 문제모음

문제1. 위에서 만든 텐써 그래프를 실행하시오 !

```
sess = tf.Session()

print (sess.run(c))
```

문제2. 아래의 모델(그래프) 를 실행하시오 !

```
import  tensorflow  as  tf

a = tf.add(1,2)
b = tf.multiply(a,3)
c = tf.add(b,5)
d = tf.multiply(c,6)
e = tf.multiply(d,5)
f = tf.div(e,6)
g = tf.add(f,d)
h = tf.multiply(g,f)

sess = tf.Session()
print (sess.run(h))
```

10780

문제3. 위의 예제를 with 절 사용해서 구현하시오 !

```
import  tensorflow  as  tf

a = tf.add(1,2)
b = tf.multiply(a,3)
c = tf.add(b,5)
d = tf.multiply(c,6)
e = tf.multiply(d,5)
f = tf.div(e,6)
g = tf.add(f,d)
h = tf.multiply(g,f)

sess = tf.Session()
print (sess.run(h))
sess.close()  <---  세션을 닫는 구문을 작성해줘야한다.
```

답:

```
with  tf.Session()  as  sess:
   print ( sess.run(h) )
```

※ with 절을 사용하면 close() 를 안써도 된다.

문제4. 위의 코드를 이용해서 구구단 2단을 출력하시오 !

tf.multiply 를 활용하세요 !

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
:
2 x 9 = 18

답:

```
import  tensorflow  as  tf

x = tf.Variable(2)
y = tf.Variable(1)

model = tf.global_variables_initializer()

sess = tf.Session()

sess.run(model)

for i in  range(9):
    z = tf.multiply(x,y)
    print ( sess.run(x)," x ", sess.run(y), " = " , sess.run(z))
    y = y +  1
```

-- 준하 코드

```
import tensorflow as tf

sess = tf.Session()
for i in range(1,10):
    y = i
    x = tf.multiply(2, y, name='x')
    print('2 *',i,'='+ str(sess.run(x)))

sess.close()
```

문제5. 구구단 2단에서 9단까지 출력하시오 !

```
import  tensorflow  as  tf

x = tf.Variable(0)
y = tf.Variable(0)
z = tf.multiply(x,y)

model = tf.global_variables_initializer()

sess = tf.Session()
```

```
sess.run(model)

for i in  range(2,10):
    for j in  range(1,10):
        print (i," x ", j," = ",sess.run(z,feed_dict={x:i,y:j}))
```

문제6.(점심시간 문제) 아래의 텐써 그래프를 실행하시오 !
숫자는 알아서 feed 하세요 ~

```
import  tensorflow  as  tf

a = tf.placeholder("float")
b = tf.placeholder("float")

y = tf.multiply(a,b)
z = tf.add(y,y)
```

문제 7. zero와 숫자 1을 채워넣는 배열을 생성하시오 !

1. numpy

```
import numpy as np

a = np.zeros((2, 2))
b = np.ones((2, 2))
print(a)
print(b)

[[0. 0.]
 [0. 0.]]
[[1. 1.]
 [1. 1.]]
```

2. tensorflow

```
import tensorflow as tf

a = tf.zeros((2, 2))
b = tf.ones((2, 2))

sess = tf.Session()
print(sess.run(a))
print(sess.run(b))

sess.close()

[[0. 0.]
 [0. 0.]]
[[1. 1.]
```

```
[1. 1.]]
```

문제 8. 아래의 numpy문법을 tensorflow로 구현하시오 !

1. numpy

```
import numpy as np

a = np.array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0])
print(np.argmax(a, axis = 0))

3
```

2. tensorflow

```
import numpy as np
import tensorflow as tf

a = np.array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0])
b = tf.argmax(a, axis=0)

sess = tf.Session()
print(sess.run(b))
sess.close()

3
```

문제 9. 아래의 numpy문법을 텐서 플로우로 구현하시오 !

1. numpy

```
import tensorflow as tf
import numpy as np

a = np.array([[[1, 2, 3],
               [2, 1, 4],
               [5, 2, 1],
               [6, 3, 2]],
              [[5, 1, 3],
               [1, 3, 4],
               [4, 2, 6],
               [3, 9, 3]],
              [[4, 5, 6],
               [7, 4, 3],
               [2, 1, 5],
               [4, 3, 1]]])

print ( np.sum( a, axis = 0) )

[[10  8 12]
 [10  8 11]
 [11  5 12]
 [13 15  6]]
```

2. tensorflow

```
import tensorflow as tf
import numpy as np

a = np.array([[[1, 2, 3],
               [2, 1, 4],
               [5, 2, 1],
               [6, 3, 2]],
              [[5, 1, 3],
               [1, 3, 4],
               [4, 2, 6],
               [3, 9, 3]],
              [[4, 5, 6],
               [7, 4, 3],
               [2, 1, 5],
               [4, 3, 1]]])
b = tf.reduce_sum(a, reduction_indices = [0])

sess = tf.Session()
print(sess.run(b))
sess.close()
```

```
[[10  8 12]
 [10  8 11]
 [11  5 12]
 [13 15  6]]
```

문제 10. 아래의 numnpy문법을 tensorflow로 구현하시오 !

1. numpy

```
import numpy as np

a = np.array([i for i in range(144)])
b = a.reshape(12,12)
print(b.shape)
```

```
(12, 12)
```

2. tensorflow

```
import tensorflow as tf
import numpy as np

a = np.array([i for i in range(144)])
b = tf.reshape(a, (12, 12))

sess = tf.Session()
print(sess.run(b))
print(b.get_shape())

sess.close()
```

```
[[  0   1   2   3   4   5   6   7   8   9  10  11]
 [ 12  13  14  15  16  17  18  19  20  21  22  23]
 [ 24  25  26  27  28  29  30  31  32  33  34  35]
 [ 36  37  38  39  40  41  42  43  44  45  46  47]
```

```
 [ 48  49  50  51  52  53  54  55  56  57  58  59]
 [ 60  61  62  63  64  65  66  67  68  69  70  71]
 [ 72  73  74  75  76  77  78  79  80  81  82  83]
 [ 84  85  86  87  88  89  90  91  92  93  94  95]
 [ 96  97  98  99 100 101 102 103 104 105 106 107]
 [108 109 110 111 112 113 114 115 116 117 118 119]
 [120 121 122 123 124 125 126 127 128 129 130 131]
 [132 133 134 135 136 137 138 139 140 141 142 143]]
(12, 12)
```

문제 11. (텐서 플로우로 구현한 단층 신경망 이해에 중요 문법) 아래의 numpy배열의 열단위 sum을 출력하시오 !

1. numpy

```
import numpy as np
import tensorflow as tf

x = np.arange(6).reshape(2, 3)
print(np.sum(x, axis=0))
```

```
[3 5 7]
```

2. tensor

```
import numpy as np
import tensorflow as tf

x = np.arange(6)
y = tf.reshape(x, (2, 3))
z = tf.reduce_sum(y, reduction_indices = [0])

sess = tf.Session()
print(sess.run(z))

sess.close()
```

```
[3 5 7]
```

문제 12. (텐서 플로우로 구현한 단층 신경망 이해에 중요 문법) 아래의 두 행렬의 합을 tensorflow로 구현하시오 !

```
 0 0 0 + 1 1 1  =  1 1 1
 0 0 0   1 1 1     1 1 1
```

```
import tensorflow as tf

a = tf.zeros([2, 3])
b = tf.ones([2, 3])
result = tf.add(a, b)

sess = tf.Session()

print(sess.run(a))
print(sess.run(b))
print(sess.run(result))
sess.close()
```

```
[[0. 0. 0.]
```

```
 [0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]]
[[1. 1. 1.]
 [1. 1. 1.]]
```

문제 13. 아래의 행렬의 내적을 tensoflow로 구현하시오 !

```
2 2 2 ◎ 3 3
2 2 2   3 3 = ?
        3 3
```

```
import tensorflow as tf
import numpy as np

x = tf.placeholder("float", [2, 3])
y = tf.placeholder("float", [3, 2])
result = tf.matmul(x, y)

sess = tf.Session()

print(sess.run(result, feed_dict = {x : [[2,2,2],[2,2,2,]],
                                    y : [[3,3],[3,3],[3,3]]}))
sess.close()
```

```
[[18. 18.]
 [18. 18.]]
```

```
※ 설명 : x = tf.constant(10)                  # 숫자 10 상수를 선언
         x = tf.Variable(0)                    # x 라는 변수를 만드는데 0 으로 값을 초기화
         x = tf.placeholder("float")           # x 라는 실수형 데이터를 담을 변수만 선언
                                               # (값을 초기화하지 않았음)
         x = tf.placeholder("float", [2, 3])   # x 라는 실수형 데이터를 행렬로 담을 변수를 선언
```

문제 14. (tensorflow의 cast함수의 이해) 아래의 배열의 True를 1로 변경하고 False를 0으로 변경시키시오 !

```
import tensorflow as tf

correct_prediction = [ True, False , True  ,True  ,True  ,True  ,True,  True  ,True
  ,True  ,True  ,True
  ,True  ,True  ,True, False , True  ,True, False , True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True,
  True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True ,False , True  ,True  ,True  ,True  ,True
  ,True  ,True, False , True, False , True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
 ,False , True  ,True  ,True]

sess = tf.Session()

a = tf.cast(correct_prediction,"float")
```

```
print(sess.run(a))
sess.close()
```

```
[1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 1.]
```

문제 15. 위의 출력된 결과에서 전체 갯수 중에 1이 몇개나 되는지 정확도를 출력하시오 1
전체 다 더해서 전체 갯수로 나눈 값을 아래와 같이 출력하시오 !

```
import tensorflow as tf

correct_prediction = [ True, False , True  ,True  ,True  ,True  ,True,  True  ,True
  ,True  ,True  ,True
  ,True  ,True  ,True, False , True  ,True, False , True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True,
  True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True ,False , True  ,True  ,True  ,True  ,True
  ,True  ,True, False , True, False , True  ,True  ,True  ,True  ,True  ,True  ,True
  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True  ,True
 ,False , True  ,True  ,True]

sess = tf.Session()

a = tf.cast(correct_prediction,"float")
b = tf.reduce_mean(a)

print(sess.run(b))
sess.close()
```

```
0.93
```

문제 16. Tensorflow에 기본적으로 내장되어 있는 mnist데이터를 가져오시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
batch_xs, batch_ys = mnist.train.next_batch(100)

print(batch_xs.shape)
print(batch_ys.shape)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 784)
```

(100, 10)

문제 17. 위의 mnist 데이터 중에 train데이터의 라벨을 one hot encoding하지 말고 숫자로 100개의 라벨을 가져오시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/")
batch_xs, batch_ys = mnist.train.next_batch(100)

print(batch_xs.shape)
print(batch_ys)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 784)
[4 3 2 5 0 2 0 2 8 8 5 3 1 6 0 6 9 9 6 5 0 2 8 9 5 3 3 1 5 9 8 8 3 0 6 5 9
 8 4 2 3 2 8 9 1 7 2 8 5 1 4 2 5 4 6 0 9 2 0 3 2 7 8 3 6 5 6 5 0 3 7 7 4 1
 2 7 8 4 4 9 1 8 5 3 5 8 4 0 5 7 4 7 1 9 1 9 9 3 7 0]
```

문제 18. 이번에는 test데이터와 test 데이터의 라벨을 100개를 가져오는데 shape만 출력하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)
batch_xs, batch_ys = mnist.test.next_batch(100)

print(batch_xs.shape)
print(batch_ys.shape)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 784)
(100, 10)
```

문제 19. 숫자 2로 채워진 행렬 2x3행렬을 텐서 플로우로 출력하시오 !

```
import tensorflow as tf

a = tf.placeholder("float",[2, 3])

sess = tf.Session()
print(sess.run(a, feed_dict = {a : [[2, 2, 2], [2, 2, 2]]}))
```

```
[[2. 2. 2.]
 [2. 2. 2.]]
```

----아래와 같이 숫자 2를 None으로 바꿔서 실행해 보시오 !

```
import tensorflow as tf

a = tf.placeholder("float",[None, 3]) # None : 입력되는 행의 갯수가 몇개든 상관 없다는 뜻

sess = tf.Session()
print(sess.run(a, feed_dict = {a : [[2, 2, 2], [2, 2, 2], [2, 2, 2]]}))
```

```
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

문제 20. Mnist 데이터 784(28x28)개에 맞게 x 변수를 placeholder로 선언하고 배치로 입력될 데이터의 갯수는 몇개이든 상관없게 None으로 변수를 만들고 Mnist데이터를 x변수에 100개를 담고 출력해 보시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
batch_xs, batch_ys = mnist.test.next_batch(100)

sess = tf.Session()
print(sess.run(x, feed_dict = {x : batch_xs}).shape)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 784)
```

문제 21. 위의 코드를 수정해서 훈련데이터 100개 뿐만 아니라 훈련데이터 라벨 100개도 출력되게끔 코드를 추가하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y = tf.placeholder("float", [None, 10])
batch_xs, batch_ys = mnist.test.next_batch(100)

sess = tf.Session()
print(sess.run(x, feed_dict = {x : batch_xs}).shape)
print(sess.run(y, feed_dict = {y : batch_ys}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 784)
(100, 10)
```

문제 22. (텐서 플로우로 가중치를 랜점으로 생성하는 방법) 2x3행렬로 -1에서 1사이의 난수를 생성하는 변수 W를 생성하고 안의 내용을 확인하시오 !

```
import tensorflow as tf

W = tf.Variable(tf.random_uniform([2,3], -1, 1))

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(W))
sess.close()
```

```
[[-0.42522287 -0.67010736  0.5873139 ]
 [ 0.4402938   0.93929124 -0.42909193]]
```

문제 23. 이번에는 mnist데이터에 맞게 100x784와 내적할 가중치 행렬 784x50으로 W를 생성하시오 !

```
import tensorflow as tf

W = tf.Variable(tf.random_uniform([784,50], -1, 1))
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(W))
sess.close()
```

문제 24. 위에서 만든 입력값 (문제21) 100x784와 지금 만든 가중치 785x50행렬과 내적을 한 결과를 출력하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
W = tf.Variable(tf.random_uniform([784,50], -1, 1))
batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()
z = tf.matmul(x, W)
```

```
sess = tf.Session()
sess.run(init)
print(sess.run(z, feed_dict = {x : batch_xs}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 50)
```

문제 25. 1x50으로 bias를 생성하는데 변수 b로 생성하고 숫자를 다 1로 채우시오 !

```
import tensorflow as tf

b = tf.Variable(tf.ones([50]))
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(b))
sess.close()
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1.]
```

문제 26. 문제 24번에서 구한 두 행렬의 내적과 지금 방금 생성한 바이어스의 합을 출력하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
W = tf.Variable(tf.random_uniform([784,50], -1, 1))
b = tf.Variable(tf.ones([50]))
y = tf.matmul(x, W) +  b

batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(y, feed_dict = {x : batch_xs}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 50)
```

문제 27. 문제 26번에서 구한 가중의 합인 y값을 시그모이드 함수에 입력해서 출력한 결과를 출력하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
W = tf.Variable(tf.random_uniform([784,50], -1, 1))
b = tf.Variable(tf.ones([50]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.sigmoid(y)
batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(y_hat, feed_dict = {x : batch_xs}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 50)
```

문제 28. 위의 활성화 함수를 Relu로 변경하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
W = tf.Variable(tf.random_uniform([784,50], -1, 1))
b = tf.Variable(tf.ones([50]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.relu(y)
batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(y_hat, feed_dict = {x : batch_xs}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 50)
```

문제 29. 이번에는 Relu가 아니라 가중의 합인 y_hat을 softmax 함수를 통과시킨 결과가 어떻게 되는지 확인하시오 !
(10개짜리 확률벡터 100개 출력이 예상됨)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(y_hat, feed_dict = {x : batch_xs}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100, 10)
```

문제 30. 텐서 플로우의 argmax 함수를 이용해서 위에서 출력된 100x10확률벡터들의 최대값의 인덱스 번호를
100개 출력하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(y_predict, feed_dict = {x : batch_xs}))
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
[4 2 1 2 2 3 2 1 1 2 2 4 4 2 2 8 2 2 4 2 1 1 2 2 4 4 2 4 2 2 1 1 4 4 1 2 2
 1 2 6 2 7 1 4 2 2 8 4 2 2 8 4 2 2 1 7 2 4 2 2 1 4 1 4 2 4 1 6 2 6 2 8 1 4
 2 6 4 4 1 2 2 1 2 2 2 1 4 2 1 2 4 2 2 1 9 2 4 1 2 2]
```

문제 31. 위의 코드에 라벨을 가져오는 코드를 추가해서 정확도를 출력하시오 !
(위의 예상 숫자 100개와 실제 숫자 100개를 비교)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
y_label = tf.argmax(y_onehot, axis = 1)
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
correction_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))

batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(accuracy, feed_dict = {x : batch_xs, y_onehot: batch_ys}))
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.1
```

문제 32. 문제 31번 코드에 오차함수인 교차 엔트로피 함수를 추가하고 loss를 출력하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
# y_label = tf.argmax(y_onehot, axis = 1)
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
# correction_prediction = tf.equal(y_predict, y_label)
# accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)
```

```
batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
print(sess.run(loss, feed_dict = {x : batch_xs, y_onehot: batch_ys}).shape)
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
(100,)
```

문제 33. 지금까지 만든 코드에 Adam 경사감소법 코드를 추가해서 학습이 되게 하시오 1

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
y_label = tf.argmax(y_onehot, axis = 1)
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
correction_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)

batch_xs, batch_ys = mnist.test.next_batch(100)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
sess.run(train, feed_dict = {x : batch_xs, y_onehot: batch_ys})
print(sess.run(accuracy, feed_dict = {x : batch_xs, y_onehot : batch_ys}))
sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz

0.09
```

문제 34. 위의 코드에 for loop문을 이용해서 1에폭 돌게 구성하시오 !

```
import tensorflow as tf
```

```
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
y_label = tf.argmax(y_onehot, axis = 1)
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
correction_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(1, 601):
    batch_xs, batch_ys = mnist.test.next_batch(100)
    sess.run(train, feed_dict = {x : batch_xs, y_onehot: batch_ys})
    print(sess.run(accuracy, feed_dict = {x : batch_xs, y_onehot : batch_ys}))
sess.close()

Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.96
```

문제 35. 위의 코드를 수정해서 아래와 같이 결과가 출력되게 하시오 ! (오늘의 마지막 문제)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
y_label = tf.argmax(y_onehot, axis = 1)
W = tf.Variable(tf.random_uniform([784,10], -1, 1))
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
correction_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
```

```
sess = tf.Session()
sess.run(init)
for i in range(1, 4):
    for j in range(1, 601):
        batch_xs, batch_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : batch_xs, y_onehot: batch_ys})
    print('%d 에폭 정확도'%i, sess.run(accuracy, feed_dict = {x : batch_xs, y_onehot : batch_ys}))

sess.close()
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
1 에폭 정확도 0.95
2 에폭 정확도 0.98
3 에폭 정확도 0.96
```

문제 36. 문제 35번까지 만든 단층 신경망에 가중치 초기화를 he를 달아서 학습 시키시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
y_label = tf.argmax(y_onehot, axis = 1)
# W = tf.Variable(tf.random_uniform([784,10], -1, 1))
# xavier 초기값
# W = tf.get_variable(name="W", shape=[784, 10], initializer=tf.contrib.layers.xavier_initializer())
# he 초기값
W = tf.get_variable(name='W', shape=[784, 10], ₩
                    initializer=tf.contrib.layers.variance_scaling_initializer())
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
correction_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(1, 4):
    for j in range(1, 601):
        batch_xs, batch_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : batch_xs, y_onehot: batch_ys})
    print('%d 에폭 정확도'%i, sess.run(accuracy, feed_dict = {x : batch_xs, y_onehot : batch_ys}))

sess.close()
```

* 텐서 플로우에서 가중치 초기화 할때 주의사항 !!
주피터 노트북과 스파이더는 오류가 나므로 tf.reset_default_graph()를 맨 위에다가 적어줘야 한다.

텐서 플로우는 그래프를 메모리에 올려서 실행하게 되는데 파이참은 코드를 매번 실행할때마다 메모리를 지워주는데 스파이더나 주피터는 대화형이라서 실행할때 마다 메모리에 그래프가 누적이 되어서 맨위에 tf.reset_default_graph()를 맨 위에다가 적어줘야 한다.

문제 37. learning rate를 0.05로 해서 다시 학습 시키시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
y_onehot = tf.placeholder("float", [None, 10])
y_label = tf.argmax(y_onehot, axis = 1)
W = tf.get_variable(name='W', shape=[784, 10], ₩
                    initializer=tf.contrib.layers.variance_scaling_initializer())
b = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W) +  b
y_hat = tf.nn.softmax(y)
y_predict = tf.argmax(y_hat, axis = 1)
correction_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(y_onehot*tf.log(y_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.05)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(1, 20):
    for j in range(1, 601):
        batch_xs, batch_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : batch_xs, y_onehot: batch_ys})
    print('%d 에폭 정확도'%i, sess.run(accuracy, feed_dict = {x : batch_xs, y_onehot : batch_ys}))

sess.close()
```

문제 38. 위의 단층 신경망을 다층(2층) 신경망으로 변환해서 돌리시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
```

```
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
W1 = tf.get_variable(name='W1', shape=[784, 50], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b1 = tf.Variable(tf.ones([50]))
W2 = tf.get_variable(name='W2', shape=[50, 10], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b2 = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W1) +  b1
y_hat = tf.nn.relu(y)
z = tf.matmul(y_hat, W2) +  b2
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)

correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.0008)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(1, 20):
    for j in range(1, 601):
        batch_xs, batch_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : batch_xs, z_onehot: batch_ys})
    print('%d 에폭 정확도'%i, sess.run(accuracy, feed_dict = {x : batch_xs, z_onehot : batch_ys}))

sess.close()
```

문제 39. 문제 38번에서 완성한 3층 신경망에 배치 정규화 코드를 추가해서 돌리고 정확도를 확인하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
W1 = tf.get_variable(name='W1', shape=[784, 50], ₩
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([50]))
W2 = tf.get_variable(name='W2', shape=[50, 10], ₩
initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W1) +  b1
batch_y = tf.contrib.layers.batch_norm(y,True)
y_hat = tf.nn.relu(y)
z = tf.matmul(y_hat, W2) +  b2
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)
```

```
correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.0005)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(1, 15):
    for j in range(1, 601):
        batch_xs, batch_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : batch_xs, z_onehot: batch_ys})
    print('%d 에폭 정확도'%i, sess.run(accuracy, feed_dict = {x : batch_xs, z_onehot : batch_ys}))

sess.close()
```

문제 40. 지금 현재까지의 코드는 신경망을 훈련만 시키는 코드였는데 테스트 데이터도 신경망에 입력해서 오버피팅이 발생하는지 확인할 수 있도록 훈련 데이터의 정확도와 테스트 데이터의 정확도를 같이 출력할 수 있도록 코드를 작성하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)

W1 = tf.get_variable(name='W1', shape=[784, 50], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([50]))
W2 = tf.get_variable(name='W2', shape=[50, 10], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W1) +  b1
batch_y = tf.contrib.layers.batch_norm(y,True)
y_hat = tf.nn.relu(y)
z = tf.matmul(y_hat, W2) +  b2
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)

correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.0005)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(1, 15):
    for j in range(1, 601):
```

```
            train_xs, train_ys = mnist.train.next_batch(100)
            test_xs, test_ys = mnist.test.next_batch(100)
            sess.run(train, feed_dict = {x : train_xs, z_onehot: train_ys})
        print('train %d 에폭 정확도'%i, ₩
              sess.run(accuracy, feed_dict = {x : train_xs, z_onehot : train_ys}))
        print('test  %d 에폭 정확도'%i, ₩
              sess.run(accuracy, feed_dict = {x : test_xs, z_onehot : test_ys}))
        print('========================')

sess.close()
```

문제 41. 위의 결과를 그래프로 시각화 하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)

W1 = tf.get_variable(name='W1', shape=[784, 50], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([50]))
W2 = tf.get_variable(name='W2', shape=[50, 10], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W1) +  b1
batch_y = tf.contrib.layers.batch_norm(y,True)
y_hat = tf.nn.relu(y)
z = tf.matmul(y_hat, W2) +  b2
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)

correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
train_acc_list = []
test_acc_list = []

sess = tf.Session()
sess.run(init)
for i in range(1, 15):
    for j in range(1, 601):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : train_xs, z_onehot: train_ys})
    train_acc = sess.run(accuracy, feed_dict = {x : train_xs, z_onehot : train_ys})
```

```
    test_acc = sess.run(accuracy, feed_dict = {x : test_xs, z_onehot : test_ys})
    print('train %d 에폭 정확도'%i, train_acc)
    train_acc_list.append(train_acc)
    print('test  %d 에폭 정확도'%i, test_acc)
    test_acc_list.append(test_acc)
    print('========================')

sess.close()

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

문제 42. 책 132페이지를 참고해서 훈련을 다 시키고 나서 테스트 데이터를 입려갛도록 코드를
작성하시오 ! (점심시간 문제)
드롭아웃은 훈련할때는 50%의 노드로 학습하고 테스트 할때는 100%의 노드로 테스트하게 하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
keep_prob = tf.placeholder("float")

W1 = tf.get_variable(name='W1', shape=[784, 100], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([100]))
W2 = tf.get_variable(name='W2', shape=[100, 10], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([10]))
y = tf.matmul(x, W1) +  b1
batch_y = tf.contrib.layers.batch_norm(y,True)
y_hat = tf.nn.relu(y)
y_drop = tf.nn.dropout(y_hat, keep_prob)
z = tf.matmul(y_drop, W2) +  b2
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)

correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.0005)
train = optimizer.minimize(loss)
```

```
init = tf.global_variables_initializer()
train_acc_list = []
test_acc_list = []

sess = tf.Session()
sess.run(init)
for i in range(1, 15):
    for j in range(1, 601):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : train_xs, z_onehot: train_ys, keep_prob : 0.5})
    train_acc = sess.run(accuracy, feed_dict = {x : train_xs, z_onehot : train_ys, keep_prob : 1.0})
    print('train %d 에폭 정확도'%i, train_acc)
    train_acc_list.append(train_acc)
    test_acc = sess.run(accuracy, feed_dict = {x : test_xs, z_onehot : test_ys, keep_prob : 1.0})
    print('test  %d 에폭 정확도'%i, test_acc)
    test_acc_list.append(test_acc)
    print('========================')

sess.close()

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

문제 43. 2층 신경망 ——▶ 3층 신경망으로 변경하고 정확도를 확인하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x = tf.placeholder("float", [None,784])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
keep_prob = tf.placeholder("float")

W1 = tf.get_variable(name='W1', shape=[784, 100], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b1 = tf.Variable(tf.ones([100]))
W2 = tf.get_variable(name='W2', shape=[100, 100], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
b2 = tf.Variable(tf.ones([100]))

W3 = tf.get_variable(name='W3', shape=[100, 10], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
```

```
b3 = tf.Variable(tf.ones([10]))

y = tf.matmul(x, W1) +  b1
batch_y = tf.contrib.layers.batch_norm(y,True)
y_hat = tf.nn.relu(y)
y_drop = tf.nn.dropout(y_hat, keep_prob)
y1 = tf.matmul(y_drop, W2) +  b2
y1_hat = tf.nn.relu(y1)
y1_drop = tf.nn.dropout(y1_hat, keep_prob)
z = tf.matmul(y1_drop, W3) +  b3
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)

correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.0005)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
train_acc_list = []
test_acc_list = []

sess = tf.Session()
sess.run(init)
for i in range(1, 15):
    for j in range(1, 601):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x : train_xs, z_onehot: train_ys, keep_prob : 0.9})
    train_acc = sess.run(accuracy, feed_dict = {x : train_xs, z_onehot : train_ys, keep_prob : 1.0})
    print('train %d 에폭 정확도'%i, train_acc)
    train_acc_list.append(train_acc)
    test_acc = sess.run(accuracy, feed_dict = {x : test_xs, z_onehot : test_ys, keep_prob : 1.0})
    print('test  %d 에폭 정확도'%i, test_acc)
    test_acc_list.append(test_acc)
    print('==========================')

sess.close()

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

문제 44. 텐서플로우로 CNN구현하기

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
```

```
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)

tf.reset_default_graph()
x_load = tf.placeholder("float", [None, 784])

z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
keep_prob = tf.placeholder("float")

W = tf.get_variable(name='W', shape=[3, 3, 1, 32], ₩
                    initializer=tf.contrib.layers.variance_scaling_initializer())
b = tf.Variable(tf.ones([28, 28, 32]))
W1 = tf.get_variable(name='W1', shape=[6272, 100], ₩
                    initializer=tf.contrib.layers.variance_scaling_initializer())
b1 = tf.Variable(tf.ones([100]))
W2 = tf.get_variable(name='W2', shape=[100, 100], ₩
                    initializer=tf.contrib.layers.variance_scaling_initializer())
b2 = tf.Variable(tf.ones([100]))
W3 = tf.get_variable(name='W3', shape=[100, 10], ₩
                    initializer=tf.contrib.layers.variance_scaling_initializer())
b3 = tf.Variable(tf.ones([10]))
x = tf.reshape(x_load, [-1 ,28, 28, 1])
x1 = tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding = 'SAME')
x1_hat = tf.nn.relu(x1) +  b
x1_pool = tf.nn.max_pool(x1_hat, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')
x1_res = tf.reshape(x1_pool, [-1, 6272])
y = tf.matmul(x1_res, W1) +  b1
batch_y = tf.contrib.layers.batch_norm(y,True)
y_hat = tf.nn.relu(y)
y_drop = tf.nn.dropout(y_hat, keep_prob)
y1 = tf.matmul(y_drop, W2) +  b2
y1_hat = tf.nn.relu(y1)
y1_drop = tf.nn.dropout(y1_hat, keep_prob)
z = tf.matmul(y1_drop, W3) +  b3
z_hat = tf.nn.softmax(z)
z_predict = tf.argmax(z_hat, axis = 1)

correction_prediction = tf.equal(z_predict, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.0005)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
train_acc_list = []
test_acc_list = []

sess = tf.Session()
sess.run(init)

train_xs, train_ys = mnist.train.next_batch(100)
test_xs, test_ys = mnist.test.next_batch(100)
train_acc = sess.run(accuracy, ₩
                    feed_dict = {x_load : train_xs, z_onehot : train_ys, keep_prob : 1.0})
print('train   초기 정확도', train_acc)
train_acc_list.append(train_acc)
test_acc = sess.run(accuracy, ₩
                   feed_dict = {x_load : test_xs, z_onehot : test_ys, keep_prob : 1.0})
```

```
print('test    초기 정확도', test_acc)
test_acc_list.append(test_acc)
print('=========================')
for i in range(1, 16):
    for j in range(1, 601):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x_load : train_xs, z_onehot: train_ys, keep_prob : 0.9})
    train_acc = sess.run(accuracy, ₩
                         feed_dict = {x_load : train_xs, z_onehot : train_ys, keep_prob : 1.0})
    print('train %d 에폭 정확도'%i, train_acc)
    train_acc_list.append(train_acc)
    test_acc = sess.run(accuracy, ₩
                        feed_dict = {x_load : test_xs, z_onehot : test_ys, keep_prob : 1.0})
    print('test  %d 에폭 정확도'%i, test_acc)
    test_acc_list.append(test_acc)
    print('=========================')

sess.close()

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

문제 44. 문제 43번 4층 신경망에 conv-poling 층을 하나 더 추가해서 5층으로 변경을하고 gpu에서 돌리시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)


# 입력층
tf.reset_default_graph()
x_load = tf.placeholder("float", [None, 784])
x = tf.reshape(x_load, [-1 ,28, 28, 1])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
keep_prob = tf.placeholder("float")

# 은닉 1층
W1 = tf.get_variable(name='W1', shape=[3, 3, 1, 32], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b1 = tf.Variable(tf.ones([28, 28, 32]))

x = tf.nn.conv2d(x, W1, strides=[1, 1, 1, 1], padding = 'SAME')
x = tf.nn.relu(x) +  b1
```

```
x = tf.nn.max_pool(x, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')
# x = tf.reshape(x, [-1, 6272])



# 은닉 2층
W2 = tf.get_variable(name='W2', shape=[3, 3, 32, 64], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b2 = tf.Variable(tf.ones([14, 14, 64]))

x1 = tf.nn.conv2d(x, W2, strides=[1, 1, 1, 1], padding = 'SAME')
x1 = tf.nn.relu(x1) + b2
x1 = tf.nn.max_pool(x1, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')
x1 = tf.reshape(x1, [-1, 3136])



# 은닉 3층
W3 = tf.get_variable(name='W3', shape=[3136, 100], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b3 = tf.Variable(tf.ones([100]))

y = tf.matmul(x1, W3) + b3
y = tf.contrib.layers.batch_norm(y,True)
y = tf.nn.relu(y)
y = tf.nn.dropout(y, keep_prob)



# 은닉 3층
W4 = tf.get_variable(name='W4', shape=[100, 100], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b4 = tf.Variable(tf.ones([100]))

y1 = tf.matmul(y, W4) + b4
y1 = tf.nn.relu(y1)
y1 = tf.nn.dropout(y1, keep_prob)



# 출력층(4층)
W5 = tf.get_variable(name='W5', shape=[100, 10], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b5 = tf.Variable(tf.ones([10]))

z = tf.matmul(y1, W5) + b5
z_hat = tf.nn.softmax(z)
z = tf.argmax(z_hat, axis = 1)



correction_prediction = tf.equal(z, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
train_acc_list = []
test_acc_list = []
```

```
# 세션 활성화
sess = tf.Session()
sess.run(init)

train_xs, train_ys = mnist.train.next_batch(100)
test_xs, test_ys = mnist.test.next_batch(100)
train_acc = sess.run(accuracy, ₩
                  feed_dict = {x_load : train_xs, z_onehot : train_ys, keep_prob : 1.0})
print('train   초기 정확도', train_acc)
train_acc_list.append(train_acc)
test_acc = sess.run(accuracy, ₩
                 feed_dict = {x_load : test_xs, z_onehot : test_ys, keep_prob : 1.0})
print('test    초기 정확도', test_acc)
test_acc_list.append(test_acc)
print('=========================')
for i in range(1, 16):
    for j in range(1, 601):
        train_xs, train_ys = mnist.train.next_batch(100)
        test_xs, test_ys = mnist.test.next_batch(100)
        sess.run(train, feed_dict = {x_load : train_xs, z_onehot: train_ys, keep_prob : 0.9})
    train_acc = sess.run(accuracy, ₩
                     feed_dict = {x_load : train_xs, z_onehot : train_ys, keep_prob : 1.0})
    print('train %d 에폭 정확도'%i, train_acc)
    train_acc_list.append(train_acc)
    test_acc = sess.run(accuracy, ₩
                    feed_dict = {x_load : test_xs, z_onehot : test_ys, keep_prob : 1.0})
    print('test  %d 에폭 정확도'%i, test_acc)
    test_acc_list.append(test_acc)
    print('=========================')

sess.close()



# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

문제 45. c:₩a₩cifar10₩test 폴더를 만들고 test이미지 100개를 이 폴더에 따로 복사하고 복사한 이미지를 아래와 같이 불러오는 함수를 생성하시오 !

```
import os

test_image = "c:₩₩a₩₩cifar10₩₩test"

def image_load(path):
    file_list = os.listdir(path)
```

```
    return file_list

print(image_load(test_image))
```

['1.png', '10.png', '100.png', '11.png', '12.png', ... '98.png', '99.png']

문제 46. 위의 함수를 수정해서 아래와 같이 숫자만 출력되게 하시오 !

```
import os

test_image = "c:₩₩a₩₩cifar10₩₩test"

def image_load(path):
    file_list = os.listdir(path)
    for i in range(len(file_list)):
        file_list[i] = int(file_list[i][:-4])
    file_list.sort()
    return file_list

print(image_load(test_image))
```

[1, 10, 100, 11, 12, 13, ... 95, 96, 97, 98, 99]

문제 47. 아래의 결과를 정렬해서 출력되게 하시오 !

```
import os

test_image = "c:₩₩a₩₩cifar10₩₩test"

def image_load(path):
    file_list = os.listdir(path)
    for i in range(len(file_list)):
        file_list[i] = int(file_list[i][:-4])
    file_list.sort()
    return file_list

print(image_load(test_image))
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, ... 95, 96, 97, 98, 99, 100]

문제 48. 문제 47번에 나온 결과에 png를 붙여서 아래와 같이 결과가 출력되게 하시오 !

```
import os

test_image = "c:₩₩a₩₩cifar10₩₩test"

def image_load(path):
    file_list = os.listdir(path)
    for i in range(len(file_list)):
        file_list[i] = int(file_list[i][:-4])
```

```
    file_list.sort()
    for i in range(len(file_list)):
        file_list[i] = str(file_list[i]) +  ".png"
    return file_list

print(image_load(test_image))
```

['1.png', '2.png', '3.png', '4.png', '5.png', ... '98.png', '99.png', '100.png']

문제 49. 이미지 이름 앞에 절대경로가 아래처럼 붙게 하시오 !

```
import os

test_image = "c:₩₩a₩₩cifar10₩₩test"

def image_load(path):
    file_list = os.listdir(path)
    for i in range(len(file_list)):
        file_list[i] = int(file_list[i][:-4])
    file_list.sort()
    for i in range(len(file_list)):
        file_list[i] = path +  "₩₩" +  str(file_list[i]) +  ".png"
    return file_list

print(image_load(test_image))
```

['c:₩₩a₩₩cifar10₩₩test₩₩1.png', ... 'c:₩₩a₩₩cifar10₩₩test₩₩100.png']

문제 52. test_label.csv 파일을 J:₩₩a₩₩cifar10 밑에 복사하고 결과가 아래와 같이 출력될 수 있도록 함수를 생성하시오

```
test_image = "c:₩₩a₩₩cifar10₩₩test"
test_label = "c:₩₩a₩₩cifar10₩₩test_label.csv"

import os
import numpy as np
import csv

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    return labellist
print( label_load(test_label) )
```

문제 53. 위의 숫자 list 를 numpy 배열로 변환하시오 !

```
test_image = "c:₩₩a₩₩cifar10₩₩test"
test_label = "c:₩₩a₩₩cifar10₩₩test_label.csv"

import os
import numpy as np
import csv

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    labellist = np.array(labellist)
    return labellist

print( label_load(test_label) )
```

문제 54. 위의 결과가 문자가 아니라 숫자로 출력되게 변환하시오 !

```
test_image = "c:₩₩a₩₩cifar10₩₩test"
test_label = "c:₩₩a₩₩cifar10₩₩test_label.csv"

import os
import numpy as np
import csv

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    labellist = np.array(labellist).astype(int)
    return labellist

print( label_load(test_label) )
```

문제 55. 아래의 결과를 출력하시오 !

```
import numpy as np

print(np.eye(10)[4])
```

문제 56. 문제 54번에서 가져온 숫자리스트를 가지고 아래와 같이 one hot encoding된 결과를 출력하시오 !

```
test_image = "c:₩₩a₩₩cifar10₩₩test"
test_label = "c:₩₩a₩₩cifar10₩₩test_label.csv"

import os
import numpy as np
import csv

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    labellist = np.array(labellist).astype(int)
    labellist = np.eye(10)[labellist]
    return labellist

print( label_load(test_label).shape )
```

문제 57. 위의 차원은 3차원인데 우리는 2차원으로 줄여야 한다. 왜냐하면 cnn코드에서 라벨이 입력될 때는 아래처럼 2차원이기 때문이다.

```
test_image = "c:₩₩a₩₩cifar10₩₩test"
test_label = "c:₩₩a₩₩cifar10₩₩test_label.csv"

import os
import numpy as np
import csv

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    labellist = np.array(labellist).astype(int)
    labellist = np.eye(10)[labellist]
    return np.squeeze(labellist)

print( label_load(test_label).shape )
```

문제 58. 지금까지 만든 두가지 함수 image_load,label_load를 loader2.py라는 파이썬 코드에 저장하고 아래와 같이 loader2.py를 import 한후에 cifar10전체 데이터를 로드하는 코드를 구현하시오 !

```
import  loader2
import time
```

```
train_image = 'c:₩₩a₩₩cifar10₩₩train₩₩'
train_label = 'c:₩₩a₩₩cifar10₩₩train_label.csv'
test_image = 'c:₩₩a₩₩cifar10₩₩test₩₩'
test_label = 'c:₩₩a₩₩cifar10₩₩test_label.csv'

print("LOADING DATA")
start = time.time()
trainX = loader2.image_load(train_image)
print(trainX.shape) # (50000, 32, 32,3)
trainY = loader2.label_load(train_label)
print(trainY.shape) # (50000, 10)
testX = loader2.image_load(test_image)
print(testX.shape) # (10000,32, 32, 3)
testY = loader2.label_load(test_label)
print(testY.shape) # (10000, 10)
```

문제 59. test100폴더 밑에 10000개의 데이터중 100개만 출력하시오 !

```
import os
import numpy as np
import csv
import loader2

test_image='c:₩₩a₩₩cifar10₩₩test'
trainX=loader2.image_load(test_image)

print(trainX[0:100])
print(trainX[100:200])
```

문제 60. next_batch함수를 만들어서 아래와 같이 데이터를 입력하고 함수를 실행하면 trainX에서 100개의 데이터(numpy 배열)을 가져오게 하시오 !

```
import os
import numpy as np
import csv
import loader2

test_image='c:₩₩a₩₩cifar10₩₩test'
trainX=loader2.image_load(test_image)

def next_batch(img, start, finish):
    return img[start:finish]

print(next_batch(trainX, 0, 100).shape)
```

문제 61. 이번에는 라벨도 배치 사이즈 만큼 같이 출력될 수 있도록 next_batch함수에 코드를 추가해서 아래와 같이 출력되게 하시오 !

```
import os
```

```
import numpy as np
import csv
import loader2

def next_batch(img, label, start, finish):
    return img[start:finish], label[start:finish]

train_image = 'c:₩₩a₩₩cifar10₩₩train₩₩'
train_label = 'c:₩₩a₩₩cifar10₩₩train_label.csv'
test_image = 'c:₩₩a₩₩cifar10₩₩test₩₩'
test_label = 'c:₩₩a₩₩cifar10₩₩test_label.csv'

print("LOADING DATA")
trainX = loader2.image_load(train_image)
trainY = loader2.label_load(train_label)

x, y = next_batch(trainX, trainY, 0, 100)
print(x)
print(y)
```

문제 62. 아래의 코드를 실행해 보시오 !

```
import random
import numpy as np
print( np.arange(10))
```

```
[0 1 2 3 4 5 6 7 8 9]
```

문제 63. 위의 숫자 10개가 랜덤으로 섞여서 출력되게 하시오 !
(random.shuffle을 사용해서 구현하시오 !)

```
import random
import numpy as np
x = np.arange(10)
random.shuffle(x)
print(x)
```

```
[4 2 6 8 7 0 9 1 5 3]
```

문제 64. 위의 코드를 이용해서 shuffle_batch함수를 만들어서 입력된 데이터가 shuffle되게 하시오 !

```
import random
import numpy as np
import  loader2

train_image = 'c:₩₩a₩₩cifar10₩₩train₩₩'
train_label = 'c:₩₩a₩₩cifar10₩₩train_label.csv'
test_image = 'c:₩₩a₩₩cifar10₩₩test₩₩'
```

```
test_label = 'c:₩₩a₩₩cifar10₩₩test_label.csv'

print("LOADING DATA")
start = time.time()
testX = loader2.image_load(test_image)
testY = loader2.label_load(test_label)
print(testX)
print(testY)
def shuffle_batch(dataa, datab):
    x = np.arange(len(dataa))
    random.shuffle(x)
    data_list2 = dataa[x]
    label2 = datab[x]
    return data_list2, label2

x, y = shuffle_batch(testX, testY)
print(x)
print(y)
```

문제 66. 기존 mnist 데이터를 텐써 플로우로 만든 cnn신경망에 입력하는 코드를 가져와서 mnist 대신에 cifar10 데이터를 입력해서 학습 시키고 정확도 그래프를 볼 수 있도록 코드를 완성시키시오.

```
trainX = loader2.image_load(train_image)
print(trainX.shape) # (50000, 32, 32,3)
trainY = loader2.label_load(train_label)
print(trainY.shape) # (50000, 10)
testX = loader2.image_load(test_image)
print(testX.shape) # (10000,32, 32, 3)
testY = loader2.label_load(test_label)
print(testY.shape) # (10000, 10)
```

문제 65. 기존 mnist 데이터를 텐써 플로우로 만든 cnn신경망에 입력하는 코드를 가져와서 mnist 대신에 cifar10 데이터를 입력해서 학습 시키고 정확도 그래프를 볼 수 있도록 코드를 완성시키시오.

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np
import loader2

#데이터 로드 그리고 한번에 셔플
train_image = 'c:₩₩a₩₩cifar10₩₩train₩₩'
train_label = 'c:₩₩a₩₩cifar10₩₩train_label.csv'
test_image = 'c:₩₩a₩₩cifar10₩₩test₩₩'
test_label = 'c:₩₩a₩₩cifar10₩₩test_label.csv'

print("LOADING DATA")
trainX = loader2.image_load(train_image)
trainY = loader2.label_load(train_label)
testX = loader2.image_load(test_image)
```

```
testY = loader2.label_load(test_label)

trainX, trainY = loader2.shuffle_batch(trainX, trainY)
testX, testY = loader2.shuffle_batch(testX, testY)



# 입력층
tf.reset_default_graph()
x_load = tf.placeholder("float", [None, 32, 32, 3])
z_onehot = tf.placeholder("float", [None, 10])
z_label = tf.argmax(z_onehot, axis = 1)
keep_prob = tf.placeholder("float")



# 은닉 1층
W1 = tf.get_variable(name='W1', shape=[3, 3, 3, 32], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b1 = tf.Variable(tf.ones([32, 32, 32]))

x = tf.nn.conv2d(x_load, W1, strides=[1, 1, 1, 1], padding = 'SAME')
x = tf.nn.relu(x) + b1
x = tf.nn.max_pool(x, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')
# x = tf.reshape(x, [-1, 6272])



# 은닉 2층
W2 = tf.get_variable(name='W2', shape=[3, 3, 32, 64], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b2 = tf.Variable(tf.ones([16, 16, 64]))

x1 = tf.nn.conv2d(x, W2, strides=[1, 1, 1, 1], padding = 'SAME')
x1 = tf.nn.relu(x1) + b2
x1 = tf.nn.max_pool(x1, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'SAME')
x1 = tf.reshape(x1, [-1, 4096])



# 은닉 3층
W3 = tf.get_variable(name='W3', shape=[4096, 100], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b3 = tf.Variable(tf.ones([100]))

y = tf.matmul(x1, W3) + b3
y = tf.contrib.layers.batch_norm(y,True)
y = tf.nn.relu(y)
y = tf.nn.dropout(y, keep_prob)



# 은닉 3층
W4 = tf.get_variable(name='W4', shape=[100, 100], ₩
                     initializer=tf.contrib.layers.variance_scaling_initializer())
b4 = tf.Variable(tf.ones([100]))

y1 = tf.matmul(y, W4) + b4
y1 = tf.nn.relu(y1)
y1 = tf.nn.dropout(y1, keep_prob)
```

```
# 출력층(4층)
W5 = tf.get_variable(name='W5', shape=[100, 10], ₩
                initializer=tf.contrib.layers.variance_scaling_initializer())
b5 = tf.Variable(tf.ones([10]))

z = tf.matmul(y1, W5) +  b5
z_hat = tf.nn.softmax(z)
z = tf.argmax(z_hat, axis = 1)



correction_prediction = tf.equal(z, z_label)
accuracy = tf.reduce_mean(tf.cast(correction_prediction, "float"))
loss = - tf.reduce_sum(z_onehot*tf.log(z_hat), axis = 1)
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()



# 세션 활성화
sess = tf.Session()
sess.run(init)

print("start")
for i in range(1, 11):
    for j in range(1, 500):
        batch_range_st = (j-1)*100
        batch_range_fi = j*100
        train_xs, train_ys = loader2.next_batch(trainX, trainY, batch_range_st, batch_range_fi)
        sess.run(train, feed_dict = {x_load : train_xs, z_onehot: train_ys, keep_prob : 0.9})
    train_acc = sess.run(accuracy, ₩
                     feed_dict = {x_load : train_xs, z_onehot : train_ys, keep_prob : 1.0})
    print('train %d 에폭 정확도'%i, train_acc)
    test_xs, test_ys =loader2.next_batch(testX, testY, 0, 100)
    test_acc = sess.run(accuracy, ₩
                    feed_dict = {x_load : test_xs, z_onehot : test_ys, keep_prob : 1.0})
    print('test  %d 에폭 정확도'%i, test_acc)
    print('+ ======================+ ')

sess.close()
```

문제 67. 개 고양이 사진이 있는 폴더를 만들고 그 폴더에 개사진 100장과 고양이 사진 100장을 넣고 아래와 같이 불러오는 함수를 생성하시오 !

```
import os
import numpy as np
import cv2

test_image = "c:₩₩b₩₩catdog"

def image_load(path):
    file_list = os.listdir(path)
```

```
    for i in range(len(file_list)):
        file_list[i] = int(file_list[i][1:-6])
    file_list.sort()
    for i in range(len(file_list)):
        file_list[i] = path +  "₩₩" +  str(file_list[i]) +  ".jpeg"
    image =[]
    for i in file_list:
        img = cv2.imread(i)
        image.append(img)
    image = np.array(image)
    return file_list

print(image_load(test_image))
```

문제 68.

```
import numpy as np
import csv

test_label = "c:₩₩b₩₩cat_dog_label.csv"

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    labellist = np.array(labellist).astype(int)
    labellist = np.eye(2)[labellist]
    return np.squeeze(labellist, axis = 1)

print(label_load(test_label))
```

문제 69 위의 4개의 함수를 loader3.py로 생성해서 아래와 같이 실행되게 해보시오 !

```
import loader3


train_image = "c:₩₩b₩₩train_4000"
train_label = "c:₩₩b₩₩train_label_4000.csv"

print("LOADING DATA")
trainX = loader3.image_load(train_image)
print(trainX.shape)
trainY = loader3.label_load(train_label)
print(trainY.shape)
```

문제 70. 개 / 고양이 훈련 데이터 4000장중에 개사진 50장과 고양이 사진 50장을 따로 별도의 폴더에 옮기시오 !

훈련 데이터 4000장

고양이 사진 : 1 ~ 2000
개 사진 : 2000 ~ 4000

↓

훈련 데이터 3900장 테스트 데이터 100장

고양이 사진 1 ~ 1950 1951 ~ 2000
개 사진 2001 ~ 3950 3951 ~ 4000

※ 신경망에 데이터를 입력하기 위해 만들어야 하는 함수 4가지
1. image_load
2. label_load
3. next_batch
4. shuffle_batch

문제 71. 지난번에 cifar10 이미지 신경망 생성할 때 사용했던 next_batch 함수를 가지고 와서 개 / 고양이 사진이 100개씩 배치되도록 next_batch 함수를 만들고 실행하시오 !

```
import loader3

test_image = "c:₩₩b₩₩test_100"
test_label = "c:₩₩b₩₩test_label_100.csv"

testX = loader3.image_load(test_image)
testY = loader3.label_load(test_label)

def next_batch(img, label, start, finish):
    return img[start:finish], label[start:finish]

print(next_batch(testX, testY, 0, 100))
```

--- 훈련 데이터 next_batch

```
import loader3

train_image = "c:₩₩b₩₩train_3900"
train_label = "c:₩₩b₩₩train_label_3900.csv"

trainX = loader3.image_load(train_image)
trainY = loader3.label_load(train_label)


def next_batch(img, label, start, finish):
    return img[start:finish], label[start:finish]

trainx, trainy = next_batch(trainX, trainY, 0, 100)

print(trainX.shape)
print(trainY.shape)
print(trainx.shape)
```

```
print(trainy.shape)

(3900, 128, 128, 3)
(3900, 2)
(100, 128, 128, 3)
(100, 2)
```

문제 72. cifar10때 사용했던 shuffle_batch 함수를 가져와서 개/고양이의 훈련 데이터가 잘 섞이는지 확인하시오 !

```
import loader3
import numpy as np

train_image = "c:₩₩b₩₩train_3900"
train_label = "c:₩₩b₩₩train_label_3900.csv"

trainX = loader3.image_load(train_image)
trainY = loader3.label_load(train_label)


def shuffle_batch(dataa, datab):
    x = np.arange(len(dataa))
    np.random.shuffle(x)
    data_list2 = dataa[x]
    label2 = datab[x]
    return data_list2, label2

trainx_shuffle, trainy_shuffle = shuffle_batch(trainX, trainY)

print(trainy_shuffle)
```

문제 73. 개/고양이 사진이 128 x 128 인데 32 x 32 로 resize 해서 별도의 폴더에 저장하시오 !

```
import os
import cv2

path = "c:₩₩b₩₩train_3900"

file_list = os.listdir(path)
file_name = sorted([int(i[:-4]) for i in file_list])
file_list = [path+ '₩₩'+ str(i)+ '.jpg' for i in file_name]
for j,i in enumerate(file_list):
    img = cv2.imread(i)
    width, height = img.shape[:2]
    resize = cv2.resize(img, (int(width / 4), int(height / 4)), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite("c:₩₩b₩₩resize₩₩train_3900₩₩" +  str(j+ 1) +  '.jpg',resize)

# -- test도 resize 해주자

path = "c:₩₩b₩₩test_100"

file_list = os.listdir(path)
file_name = sorted([int(i[:-4]) for i in file_list])
```

```
    file_list = [path+ '₩₩'+ str(i)+ '.jpg' for i in file_name]
    for j,i in enumerate(file_list):
        img = cv2.imread(i)
        width, height = img.shape[:2]
        resize = cv2.resize(img, (int(width / 4), int(height / 4)), interpolation=cv2.INTER_CUBIC)
        cv2.imwrite("c:₩₩b₩₩resize₩₩test_100₩₩" +  str(j+ 1) +  '.jpg',resize)
```

문제 74. (점심시간 문제) 32x32로 줄인 개고양이 사진을 지난번 cifar10할때 만들었던 vgg 코드에 입력해서 훈련시키고 나서 테스트 데이터를 입력해서 정확도를 확인하시오 !

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt
import numpy as np
import loader3

#데이터 로드 그리고 한번에 셔플
train_image = '/home/heaven/work_space/d/resize/train_3900'
train_label = '/home/heaven/work_space/d/resize/train_label_3900.csv'
test_image = '/home/heaven/work_space/d/resize/test_100'
test_label = '/home/heaven/work_space/d/resize/test_label_100.csv'



print("LOADING DATA")

trainX = loader3.image_load(train_image)
trainY = loader3.label_load(train_label)
testX = loader3.image_load(test_image)
testY = loader3.label_load(test_label)

print("LOADED DATA")

tf.reset_default_graph()

#입력층
x = tf.placeholder("float",[None,32,32,3])
keep_prob = tf.placeholder("float")



#conv_1
b1 = tf.Variable(tf.ones([128]))
W1 = tf.Variable(tf.random_normal([3,3,3,128],stddev = 0.01))
y1 = tf.nn.conv2d(x, W1, strides=[1,1,1,1], padding = 'SAME')
y1 = y1 +  b1
y1 = tf.contrib.layers.batch_norm(y1,scale=True)
y1 = tf.nn.leaky_relu(y1)

#conv_2
b1_2 = tf.Variable(tf.ones([128]))
W1_2 =  tf.Variable(tf.random_normal([3,3,128,128],stddev = 0.01))
y1_2 = tf.nn.conv2d(y1, W1_2, strides=[1,1,1,1], padding = 'SAME')
y1_2 = y1_2 +  b1_2
y1_2 = tf.contrib.layers.batch_norm(y1_2,scale=True)
y1_2 = tf.nn.leaky_relu(y1_2)
y1_2 = tf.nn.dropout(y1_2, keep_prob)
```

```
#maxpooling
y1_2 = tf.nn.max_pool(y1_2, ksize = [1,2,2,1], strides = [1,2,2,1], padding = 'SAME')


#conv_3
b2 = tf.Variable(tf.ones([256]))
W2 = tf.Variable(tf.random_normal([3,3,128,256],stddev = 0.01))
y2 = tf.nn.conv2d(y1_2, W2, strides=[1,1,1,1], padding = 'SAME')
y2 = y2 +  b2
y2 = tf.contrib.layers.batch_norm(y2,scale=True)
y2 = tf.nn.leaky_relu(y2)

#conv_4
b2_2 = tf.Variable(tf.ones([256]))
W2_2 = tf.Variable(tf.random_normal([3,3,256,256],stddev = 0.01))
y2_2 = tf.nn.conv2d(y2, W2_2, strides=[1,1,1,1], padding = 'SAME')
y2_2 = y2_2 +  b2_2
y2_2 = tf.contrib.layers.batch_norm(y2_2,scale=True)
y2_2 = tf.nn.leaky_relu(y2_2)
y2_2 = tf.nn.dropout(y2_2, keep_prob)

#maxpooling
y2_2 = tf.nn.max_pool(y2_2, ksize = [1,2,2,1], strides = [1,2,2,1], padding = 'SAME')


#conv_5
b3 = tf.Variable(tf.ones([512]))
W3 = tf.Variable(tf.random_normal([3,3,256,512],stddev = 0.01))
y3 = tf.nn.conv2d(y2_2, W3, strides=[1,1,1,1], padding = 'SAME')
y3 = y3 +  b3
y3 = tf.contrib.layers.batch_norm(y3,scale=True)
y3 = tf.nn.leaky_relu(y3)

#conv_6
b3_2 = tf.Variable(tf.ones([512]))
W3_2 = tf.Variable(tf.random_normal([3,3,512,512],stddev = 0.01))
y3_2 = tf.nn.conv2d(y3, W3_2, strides=[1,1,1,1], padding = 'SAME')
y3_2 = y3_2 +  b3_2
y3_2 = tf.contrib.layers.batch_norm(y3_2,scale=True)
y3_2 = tf.nn.leaky_relu(y3_2)
y3_2 = tf.nn.dropout(y3_2, keep_prob)

#maxpooling
y3_2 = tf.nn.max_pool(y3_2, ksize = [1,2,2,1], strides = [1,2,2,1], padding = 'SAME')


#Affine1
b4 = tf.Variable(tf.ones([1024]))
W4 = tf.get_variable(name='W4', shape=[4*4*512, 1024], ₩
    initializer=tf.contrib.layers.variance_scaling_initializer())
y4 = tf.reshape(y3_2, [-1, 4*4*512])
y4 = tf.matmul(y4,W4) +  b4
y4 = tf.contrib.layers.batch_norm(y4,scale=True)
y4 = tf.nn.leaky_relu(y4)


#Affine2
b5 = tf.Variable(tf.ones([1024]))
W5 = tf.get_variable(name='W5', shape=[1024, 1024],₩
```

```
    initializer=tf.contrib.layers.variance_scaling_initializer())
y5 = tf.matmul(y4,W5) +  b5
y5 = tf.contrib.layers.batch_norm(y5,scale=True)
y5 = tf.nn.leaky_relu(y5)


#드롭아웃
y5_drop = tf.nn.dropout(y5, keep_prob)


#출력층
b6 = tf.Variable(tf.ones([2]))
W6 = tf.get_variable(name='W6', shape=[1024, 2],₩
    initializer=tf.contrib.layers.variance_scaling_initializer()) # he 초기값
y6 = tf.matmul(y5_drop,W6) +  b6
y6 = tf.contrib.layers.batch_norm(y6,scale=True)
y_hat = tf.nn.softmax(y6)


#예측값
y_predict = tf.argmax(y_hat,1)


# 라벨을 저장하기 위한 변수 생성
y_onehot = tf.placeholder("float",[None,2])
y_label = tf.argmax(y_onehot, axis = 1)


# 정확도를 출력하기 위한 변수 생성
correct_prediction = tf.equal(y_predict, y_label)
accuracy = tf.reduce_mean(tf.cast(correct_prediction,"float"))


# 교차 엔트로피 오차 함수
loss = -tf.reduce_sum(y_onehot * tf.log(y_hat), axis = 1)


# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)


# Adam 경사 감소법
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)


# 학습 오퍼레이션 정의
train = optimizer.minimize(loss)


# 변수 초기화
init = tf.global_variables_initializer()
train_acc_list = []
test_acc_list = []


with tf.Session() as sess:
    sess.run(init)
    for j in range(20):
        for i in range(600):
            trainX , trainY = loader3.shuffle_batch(trainX, trainY)
```

```
            testX, testY = loader3.shuffle_batch(testX, testY)

            train_xs, train_ys = loader3.next_batch(trainX, trainY, 0, 100)
            test_xs, test_ys = loader3.next_batch(testX, testY, 0, 100)

            sess.run(train, feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 0.8})

            if i == 0:
                train_acc = sess.run(accuracy, ₩
                          feed_dict={x: train_xs, y_onehot: train_ys, keep_prob: 1.0})
                test_acc = sess.run(accuracy, ₩
                         feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0})

                train_acc_list.append(train_acc)
                test_acc_list.append(test_acc)

                print('훈련', str(j + 1) + '에폭 정확도 :', train_acc)
                print('테스트', str(j + 1) + '에폭 정확도 :', test_acc)
                print('-----------------------------------------------')

    markers = {'train': 'o', 'test': 's'}
    x = np.arange(len(train_acc_list))
    plt.plot()
    plt.plot(x, train_acc_list, label='train acc')
    plt.plot(x, test_acc_list, label='test acc', linestyle='--')
    plt.xlabel("epochs")
    plt.ylabel("accuracy")
    plt.ylim(min(min(train_acc_list),min(test_acc_list))-0.1, 1.1)
    plt.legend(loc='lower right')
    plt.show()
```

문제 75. 이미지를 회전시키는 아래의 코드를 이용해서 개사진 1950장을 회전시킨 1950장을 생성하시오 !

```
    import os
    import cv2

    path = "c:₩₩b₩₩train_3900"

    file_list = os.listdir(path)
    file_name = sorted([int(i[:-4]) for i in file_list])
    file_list = [path+ '₩₩'+ str(i)+ '.jpg' for i in file_name]

    for j,i in enumerate(file_list):
        img = cv2.imread(i)
        img = cv.rotate(img, cv.ROTATE_90_CLOCKWISE)
        width, height = img.shape[:2]
        resize = cv2.resize(img, (int(width / 4), int(height / 4)), interpolation=cv2.INTER_CUBIC)
        cv2.imwrite("c:₩₩b₩₩train_3900_rotate₩₩" + str(j+ 1) + '.jpg',resize)

    -- test데이터도 해보자

    import os
    import cv2

    path = "c:₩₩b₩₩test_100"

    file_list = os.listdir(path)
```

```
file_name = sorted([int(i[:-4]) for i in file_list])
file_list = [path+ '₩₩'+ str(i)+ '.jpg' for i in file_name]

for j,i in enumerate(file_list):
    img = cv2.imread(i)
    img = cv.rotate(img, cv.ROTATE_90_CLOCKWISE)
    width, height = img.shape[:2]
    resize = cv2.resize(img, (int(width / 4), int(height / 4)), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite("c:₩₩b₩₩test_100_rotate₩₩" +  str(j+ 1) +  '.jpg',resize)
```

문제 77. (오늘의 마지막 문제) 카톡에 첨부한 고양이 14977장, 개사진 17830장 이미지를 다운받아 훈련 데이터와 테스트 데이터로 나누고 vgg9신경망에 넣어서 학습시키시오 !

문제 89. 아래의 소프트 맥스 함수를 통과한 결과 데이터를 가지고 아래의 결과를 출력하시오 !

```
import  numpy as np

x = [[0.1,0.05,0.1,0.0,0.05,0.7,0.0,0.1,0.0,0.0],
     [0.1,0.05,0.2,0.0,0.05,0.1,0.0,0.6,0.0,0.0],
     [0.0,0.05,0.3,0.0,0.05,0.1,0.0,0.6,0.0,0.0],
     [0.0,0.05,0.4,0.0,0.05,0.0,0.0,0.5,0.0,0.0],
     [0.0,0.05,0.5,0.0,0.05,0.0,0.0,0.4,0.0,0.0],
     [0.0,0.05,0.6,0.0,0.05,0.0,0.0,0.3,0.0,0.0],
     [0.0,0.05,0.7,0.0,0.05,0.0,0.0,0.2,0.0,0.0],
     [0.0,0.1,0.8,0.0,0.1,0.0,0.0,0.2,0.0,0.0],
     [0.0,0.05,0.9,0.0,0.05,0.0,0.0,0.0,0.0,0.0]]
```

결과 :

```
[ 0.1   0.05  0.1   0.    0.05  0.7   0.    0.1   0.    0.  ] 0.7
[ 0.1   0.05  0.2   0.    0.05  0.1   0.    0.6   0.    0.  ] 0.6
[ 0.    0.05  0.3   0.    0.05  0.1   0.    0.6   0.    0.  ] 0.6
[ 0.    0.05  0.4   0.    0.05  0.    0.    0.5   0.    0.  ] 0.5
[ 0.    0.05  0.5   0.    0.05  0.    0.    0.4   0.    0.  ] 0.5
[ 0.    0.05  0.6   0.    0.05  0.    0.    0.3   0.    0.  ] 0.6
[ 0.    0.05  0.7   0.    0.05  0.    0.    0.2   0.    0.  ] 0.7
[ 0.    0.1   0.8   0.    0.1   0.    0.    0.2   0.    0. ] 0.8
[ 0.    0.05  0.9   0.    0.05  0.    0.    0.    0.    0.  ] 0.9
```

```
for i in range(len(x)):
    print(x[i],np.max(x,axis=1)[i])
```

문제 90. 아래와 같이 순서번호와 최대값이 출력되게 하시오 !

```
0 0.7
1 0.6
```

2 0.6
3 0.5
4 0.5
5 0.6
6 0.7
7 0.8
8 0.9

```
for i in range(len(x)):
    print(i,np.max(x,axis=1)[i])
```

문제 91. 0.5~0.6 사이의 확률을 갖는 이미지의 번호와 확률을
출력하시오 !

1 0.6
2 0.6
3 0.5
4 0.5
5 0.6

문제 92. 어제 완성한 개/고양이 vgg 모델을 저장하시오 !

파이썬을 이용했을때는 최종 갱신된 가중치와 바이어스를 pickle 로 내렸는데 텐써플로우는 아래의 코드를 이용해서 모델을 저장한다.

saver = tf.train.Saver() 는 변수초기화 위에 두고
모델 저장하는
saver.save(sess, 'D:₩₩b₩₩resize₩₩model₩₩model')
는 훈련을 다 시키고 tf.Session() 이 있는 곳에 배치

```
# 모델 저장

saver = tf.train.Saver()

# 변수 초기화
```

https://goodtogreate.tistory.com/entry/Saving-and-Restoring 참고

문제 93. 개/고양이가 아닌 사진들을 훈련 데이터에서 빼내고 테스트 데이터에 포함 시키시오 !

개사진 : 3297, 3299, 3302, 3300, 8362

라벨도 훈련 라벨에서 0(개) 5개 빼고 테스트 라벨에 0(개) 5개를 추가하시오 !

문제 94. GPU PC 에서 20 에폭 정도 돌린 모델을 저장하고 그 모델을 CPU PC 로 옮겨서 테스트 데이터의 정확도를 확인하시오 !

D:₩b₩resize₩model <-- 카톡으로 받은 알집 안의 4개의 파일을 둠

1. model.data-00000-of-00001 ---> training variables 를 가지고 있다.

2. checkpoint ---> binary 파일로 weights, biases, gradients 등을 저장한다.

3. model.index

4. model.meta ---> Tensorflow graph 를 저장하게 된다. 즉 all variables, operations, collections 등을 저장한다.

* 모델 불러오는 방법

```
saver = tf.train.import_meta_graph('model.meta')

# 변수 초기화
#init = tf.global_variables_initializer()

with tf.Session() as sess:
    saver.restore(sess, 'd:₩₩catdog₩₩model₩₩model')
```

문제 95. 훈련시에 배치정규화로 인해서 만들어진 최적의 배타와 감마를 계속 잘 유지 시킬수 있도록 위의 코드에 아래 코드를 추가하시오 !

```
# SGD 경사 감소법
# optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)

update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
    # Ensures that we execute the update_ops before performing the train_step
    Train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)
```

문제 98. (오늘의 마지막 문제) 테스트 데이터 100장을 테스트 할 때 소프트 맥스를 통과한 확률값이 0.4 ~ 0.6 사이인 이미지의 이미지 번호를 출력하고 사람이 봐도 개인지 고양이 인지 구분이 어려운지 확인하시오 !

문제 99. 문제 98번 코드로 테스트 데이터 (100장) 을 다시 테스트 하는데 테스트 데이터 중에서 못맞춘 사진의 번호가 어떻게 되는지 출력하시오 !

힌트 :

```
for t in test_acc_list:
    for j, i in zip(t,label):
        if np.argmax(j) != i:
            print('-----------------------')
```

답 :

```
  p = sess.run(y_predict, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0, training: False})
    l = sess.run(y_label, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0, training: False})
    w = []
    for x in range(len(p)):
        if p[x] != l[x] :
            w.append(x+ 1)
    print(w)
    print(len(w))
```

문제 100. 틀린 사진만 d:₩₩wrong_image 라는 폴더에 옮겨지게 하시오 !

힌트 : os.rename 사용

```
#모델 저장
saver = tf.train.Saver()
with tf.Session() as sess:
    test_acc_list = []

    saver.restore(sess, 'D:₩₩catdog₩₩final_weight₩₩model')
    #test_txs, test_tys = loader3.next_batch(test_img, test_lb,0,3)

    test_txs = testX
    test_label = testY

    test_acc_list.append(sess.run(accuracy, feed_dict={x: test_txs, y_onehot: test_label, ₩
                                                       keep_prob: 1.0, training:False}))

    print("정확도:",test_acc_list)

    l = sess.run(y_label, feed_dict={x: test_txs, y_onehot: test_label, keep_prob: 1.0, training: False})
    p = sess.run(y_predict, feed_dict={x:test_txs, keep_prob:1.0, training:False}).round(4)

    w = []
    for x in range(len(p)):
        if p[x] != l[x] :
            w.append(x+ 1)


    #w = [ x+ 1 for x in range(len(p)) if p[x] != l[x] ]
    print("불일치 사진:",w)
    print("불일치 갯수:",len(w))



import cv2
path = 'D:₩₩b₩₩JH_catdog2₩₩test2₩₩'
```

```
for i in w:
    image = cv2.imread('{}{}.jpeg'.format(path, i) )
    cv2.imwrite('D:₩₩b₩₩wrong_img₩₩{}.jpeg'.format(i), image)
print('complite')
```

문제 101. 옮긴 사진 다시 원본에 넣어주고 다시수행하는데 못맞춘 이미지 중에 확률이 0.2 이하인 것만 wrong_img 폴더에 옮겨지게 하시오 !

```
# 애매한 사진 필터링하기
p = sess.run(y_predict, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0, training: False})
l = sess.run(y_label, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0, training: False})
h = sess.run(y_hat, feed_dict={x: test_xs, y_onehot: test_ys, keep_prob: 1.0, training: False})

for x in range(len(p)):
    if p[x] != l[x] and np.min(h[x])<=0.2:
        wrong_image_list.append(x+ 1)
```

문제 102. 폐사진 이미지의 사이즈를 128 x 128 로 일괄 변경하시오 !

PlastiliqImageResizerInstall.exe 프로그램 사용

문제 103. (점심시간 문제) 폐사진을 로드해서 numpy array 숫자 리스트로 변환하는 loader4.py 를 loader3.py 를 가지고 수정해서 만드시오 !

1~ 6470 장을 train 폴더에 넣고
6471 ~ 7470 장을 test 폴더에 넣고
라벨을 lung_train_label.csv 와 lung_test_label.csv
를 생성해서 아래의 코드가 실행되게 하시오 !

```
test_image = 'D:₩₩d₩₩test2₩₩'
test_label = 'D:₩₩d₩₩lung_test_label.csv'

train_image = 'D:₩₩d₩₩train2₩₩'
train_label = 'D:₩₩d₩₩lung_train_label.csv'


print("LOADING DATA")
# 폐사진 데이터 로드

trainX = loader4.image_load(train_image)
trainY = loader4.label_load(train_label)
testX = loader4.image_load(test_image)
testY = loader4.label_load(test_label)
```

normal 은 1 patient 는 0으로 고쳐서

문제 104. 개고양이 vgg신경망에 폐사진을 입력해서 20에폭 수행하고 훈련 정확도와 테스트 정확도를 확인하시오 !