Tensorflow를 이용한
Deep Learning의 기초

제2강: 첫 번째 신경망

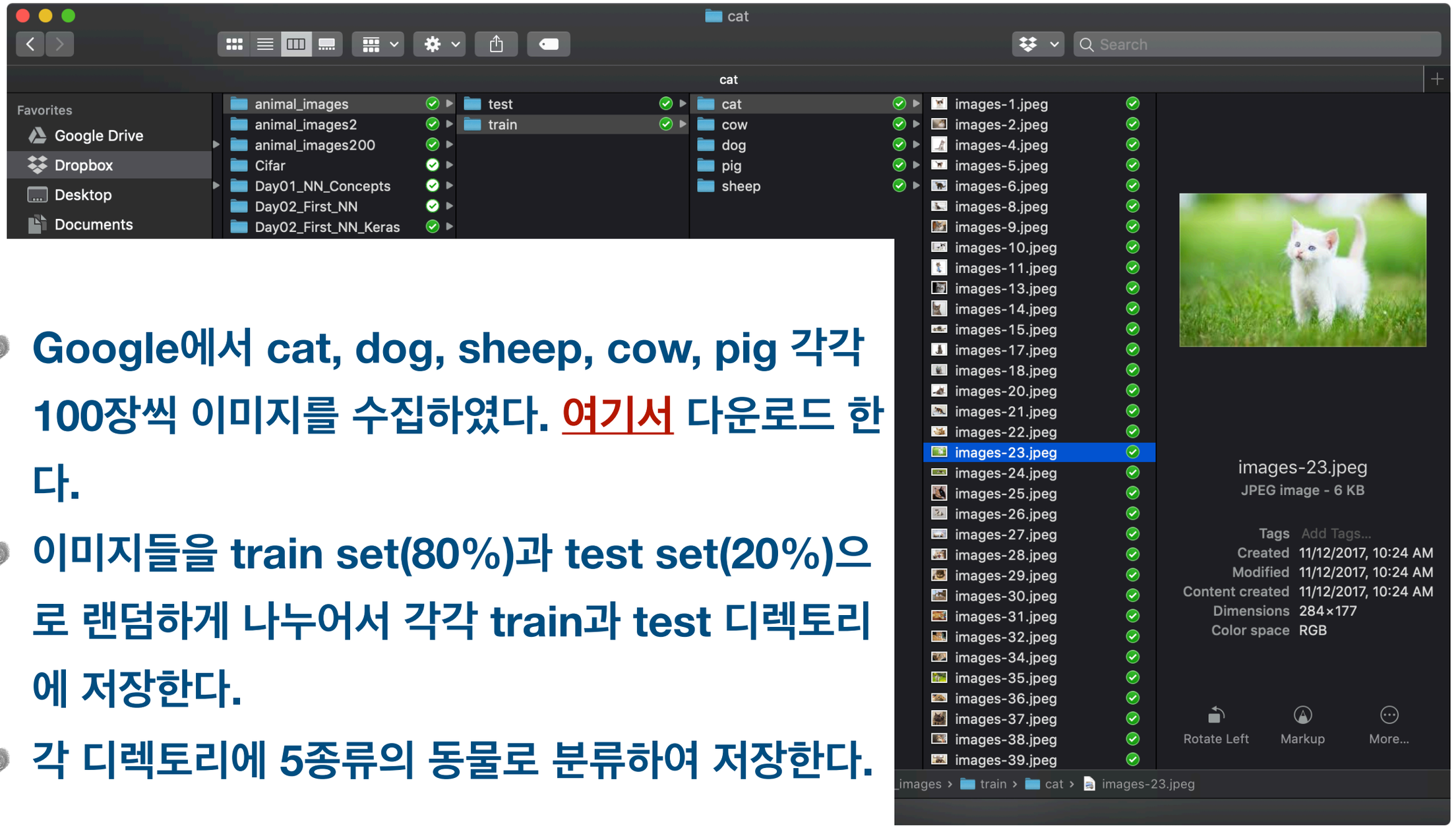
부경대학교 IT융합응용공학과
권 오 흠

학습 목표

- 간단한 image classification 문제를 다룬다.
- Python에서 이미지 파일을 읽고 다루는 방법들을 살펴본다.
- Tensorflow를 이용하여 기본적인 구조의 신경망을 구성해본다.
  - 학습 데이터 전체를 메모리에 로드할 수 있을 정도의 규모에서 사용하는 방법
  - Fully (densely) connected layer로만 구성된 신경망

Reading Image Files
이미지 파일 읽기

사용할 이미지 데이터의 준비



- Google에서 cat, dog, sheep, cow, pig 각각 100장씩 이미지를 수집하였다. 여기서 다운로드 한다.
- 이미지들을 train set(80%)과 test set(20%)으로 랜덤하게 나누어서 각각 train과 test 디렉토리에 저장한다.
- 각 디렉토리에 5종류의 동물로 분류하여 저장한다.

> ★ **Note:** The preferred way to feed data into a tensorflow program is using the **`tf.data`** API.

There are four methods of getting data into a TensorFlow program:

강좌의 후반부에 다룰 예정이지만
여기까지 진도를 나갈 수 있을지 잘 모르겠음

- `tf.data` API: Easily construct a complex input pipeline. (preferred method)
- Feeding: Python code provides the data when running each step.

강좌의 전반부에는 이 방법을 사용

- `QueueRunner`: a queue-based input pipeline reads the data from files at the beginning of a TensorFlow graph.
- Preloaded data: a constant or variable in the TensorFlow graph holds all the data (for small data sets).

Libraries for Reading and Preprocessing Images

- **Handling Images**
  - **PIL (Python Image Library)**: discontinued
  - **Pillow**: PIL fork, actively maintained
  - **OpenCV**: primarily for vision
  - **Tensorflow** 자체도 image handling library를 제공
- **Matplotlib**
  - for various **plotting**

code01.py

https://docs.opencv.org/3.1.0/dc/d2e/tutorial_py_image_display.html

- opencv를 이용하여 이미지를 프로그램 내로 load하고 간단한 조작을 하는 것을 연습한다.
- 프로그램 내에서 이미지는 numpy array로 저장된다. numpy array의 기본적인 사용법을 python list와 비교해서 어느정도 습득해야 한다.

```
import numpy as np
import os
import cv2

img_path = '../animal_images/cat/images-1.jpeg'

img = cv2.imread(img_path)
```

numpy array가 가진 기본 속성들(dtype, shape 등)을 확인해 본다.

img는 numpy array이다.
numpy array는 배열의 개념이 없는 Python에서 다차원 배열을 구현해주는 라이브러리이다.
PyCharm Debugger로 확인해보자.

```
print(img)
img_list = img.tolist()
img_array = np.array(img_list)
```

numpy array와 python list간의 상호변환 방법을 알아본다.

- cv2.imread() 함수의 2번째 매개변수로 다음 중 하나를 지정할 수 있다.
  - cv2.IMREAD_COLOR: default flag
  - cv2.IMREAD_GRAYSCALE: grayscale mode
  - cv2.IMREAD_UNCHANGED: include alpha channel
- 모드를 바꿔가면서 테스트해본다.

code01.py

https://docs.opencv.org/3.1.0/dc/d2e/tutorial_py_image_display.html

- 이미지를 화면에 display하고, resize하고, 이미지 파일로 저장하고, color space를 변환하는 일을 해본다.

```
cv2.imshow('Test Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

← 이미지를 화면에 display한다. 아무 키나 누르면 사라진다.

```
img = cv2.resize(img, (200, 200), interpolation=cv2.INTER_CUBIC)  # resize
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # cv2 load images as BGR, convert it to RGB
```

```
cv2.imwrite('converted.jpg', img)
```

← 이미지를 파일로 저장한다.

data visualization package

data plotting module

```
from matplotlib import pyplot as plt

plt.imshow(img)
plt.show()
```

matplotlib 패키지가 설치되어 있지 않으면 이 부분에서 오류가 난다.
커서를 이 부분에 위치시키고 Alt-Enter를 누르면 “install package matplotlib”라는 메뉴가 생기는데 이것을 실행한다.
혹은 Anaconda Prompt를 실행하고 다음과 같이 설치해도 된다.

```
$ activate tfenv3.5
$ conda install matplotlib
```

```
img_reshaped = img.reshape(100, 400, 3)

plt.imshow(img_reshaped)
plt.show()

flattened_image = img.ravel()
```

```
img_f32 = np.float32(img)

normalized_img32 = img_f32/255.

zero_centered_img = (img_f32 - np.mean(img_f32))/np.std(img_f32)

print(zero_centered_img)
```

1. https://www.youtube.com/watch?v=rN0TREj8G7U
2. https://www.youtube.com/watch?v=a8aDcLk4vRc&list=PLeo1K3hjS3uset9zlVzJWqplaWBiacTEU&index=2
3. https://www.youtube.com/watch?v=_d_Ka-ks2a0&list=PLeo1K3hjS3uset9zlVzJWqplaWBiacTEU&index=3
4. https://www.youtube.com/watch?v=XawR6CjAYV4&list=PLeo1K3hjS3uset9zlVzJWqplaWBiacTEU&index=4

Preparing Training Data
학습 데이터 준비하기

HandleInput.py

Load All Images into Memory

HandleInput.py에서는 준비한 모든 이미지들을 프로그램으로 읽어와서 저장하는 일을 해본다.

```
import numpy as np
import os
import cv2

IMG_HEIGHT = 40
IMG_WIDTH = 60
NUM_CHANNEL = 3
NUM_CLASS = 5

IMAGE_DIR_BASE = '../animal_images'
```

이미지 크기는 40*60으로 resize해서 사용할 계획이다.
NUM_CLASS는 분류할 동물 종류의 가지 수이다.

```
def load_image(addr):
    img = cv2.imread(addr)
    img = cv2.resize(img, (IMG_WIDTH, IMG_HEIGHT), interpolation=cv2.INTER_CUBIC)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    img = img.astype(np.float32)

    img = (img - np.mean(img))/np.std(img)
    return img
```

unit8 타입의 데이터를 float32 타입으로 변환하는 다른 방법
(preferred method)

set_name은 문자열이며 'train' 혹은 'test'이다.

os.path.join으로 경로명을 만든다.

os.listdir은 지정된 디렉토리에 저장된 모든 파일 혹은 서브디렉토리명의 리스트를 반환한다..

알파벳 순으로 정렬한다.

```
def load_data_set(set_name):

    data_set_dir = os.path.join(IMAGE_DIR_BASE, set_name)
    image_dir_list = os.listdir(data_set_dir)
    image_dir_list.sort()

    features = []    # empty Python list
    labels = []      # 0 cat, 1 cow, 2 dog, 3 pig, 4 sheep

    for cls_index, dir_name in enumerate(image_dir_list):
        image_list = os.listdir(os.path.join(data_set_dir, dir_name))
        for file_name in image_list:
            if 'png' in file_name or 'jpg' in file_name or 'jpeg' in file_name:
                image = load_image(os.path.join(data_set_dir, dir_name, file_name))
                features.append(image)
                labels.append(cls_index)
```

이미지와 라벨들을 랜덤하게 셔플링한다. 각각 별개의 배열에 저장된 이미지와 라벨을 함께 셔플링하기 위한 한 가지 방법이다.

```
idxs = np.arange(0, len(features))
np.random.shuffle(idxs)
features = np.array(features)
labels = np.array(labels)
shuf_features = features[idxs]
shuf_labels = labels[idxs]

return shuf_features, shuf_labels
```

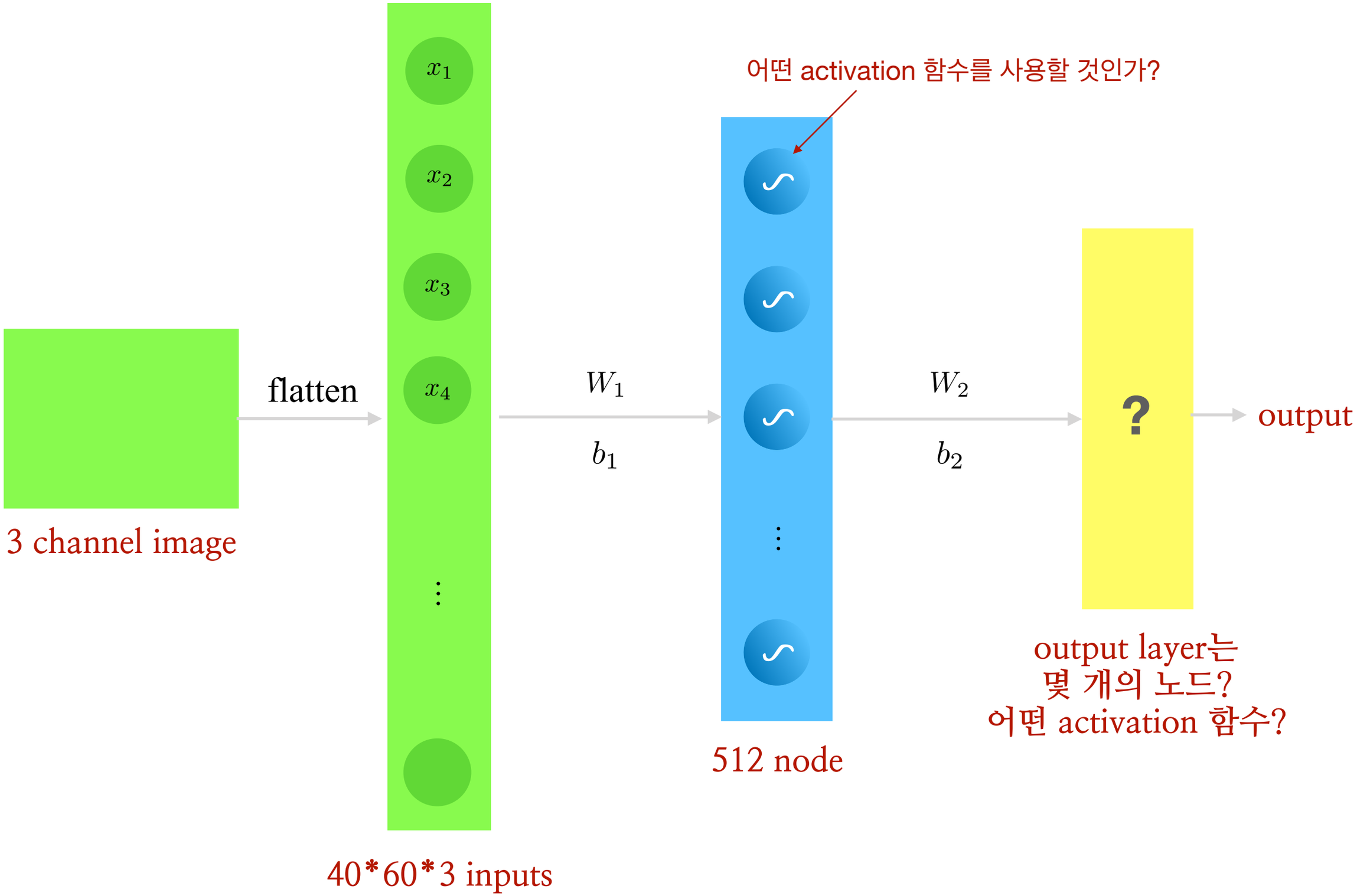
```
def load_all_data():
    train_images, train_labels = load_data_set('train')
    test_images, test_labels = load_data_set('test')
    return train_images, train_labels, test_images, test_labels


if __name__ == '__main__':
    train_images, train_labels, test_images, test_labels = load_all_data()
    print(train_images.shape)
```

이미지 분류 신경망 구현

https://www.tensorflow.org/tutorials/keras/basic_classification

```
import tensorflow as tf
from tensorflow import keras
from HandleInput import *
import numpy as np
from matplotlib import pyplot as plt

class_names = ['cat', 'cow', 'dog', 'pig', 'sheep']
train_features, train_labels, test_features, test_labels = load_all_data()

plt.figure()
plt.imshow(train_features[0])
plt.colorbar()
plt.grid(False)
plt.show()

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_features[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])

plt.show()
```

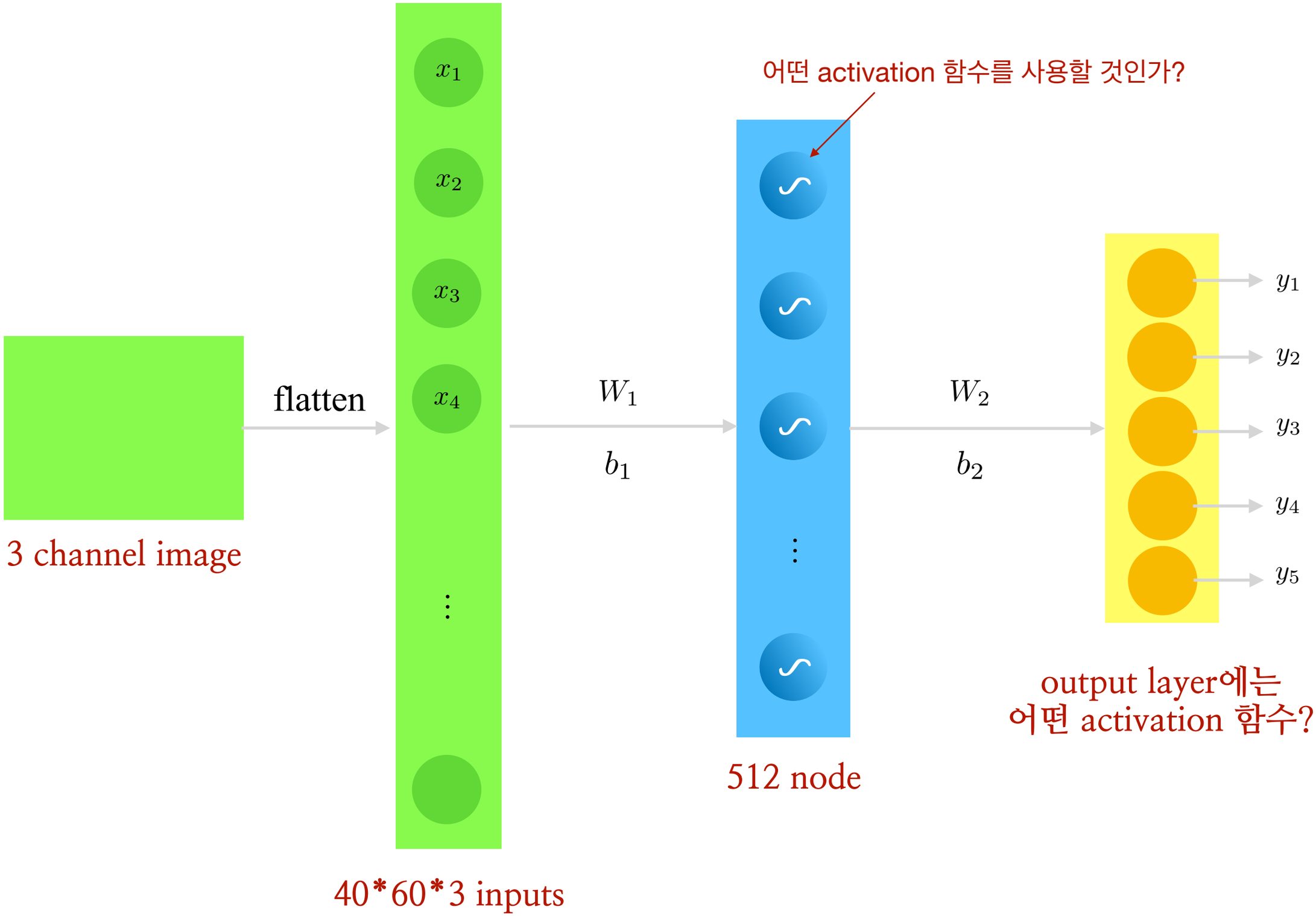
네트워크 설계


어떤 activation 함수를 사용할 것인가?
x_1
x_2
x_3
x_4
flatten
W_1
b_1
W_2
b_2
?
output
3 channel image
40*60*3 inputs
512 node
output layer는
몇 개의 노드?
어떤 activation 함수?

One-Hot Encoding

- 이미지 classification 문제에서는 label과 네트워크의 출력을 one-hot encoding으로 표현하는 것이 일반적이다.
- 클래스의 개수가 k개일 때 각각의 이미지에 대해서 라벨과 출력을 k-차원 벡터로 표현한다. 해당 이미지가 소속된 클래스에 해당하는 하나의 값만 1이고 나머지는 모두 0이다.
  - 우리의 예에서 클래스의 개수는 5개이고, 순서는 cat, cow, dog, pig, sheep라고 하자.
  - cat은 [1, 0, 0, 0, 0]으로, cow는 [0, 1, 0, 0, 0], dog는 [0, 0, 1, 0, 0], pig는 [0, 0, 0, 1, 0], 그리고 sheep는 [0, 0, 0, 0, 1]로 표현한다.
- 그냥 0, 1, 2, 3, 4와 같이 하나의 스칼라 값으로 표현하는 것이 적절하지 않은 이유는?
- 우리의 경우 label은 one-hot-encoding으로 표현되어 있지 않다.

네트워크 설계


어떤 activation 함수를 사용할 것인가?
x_1
x_2
x_3
x_4
flatten
W_1
b_1
W_2
b_2
y_1
y_2
y_3
y_4
y_5
3 channel image
40*60*3 inputs
512 node
output layer에는
어떤 activation 함수?

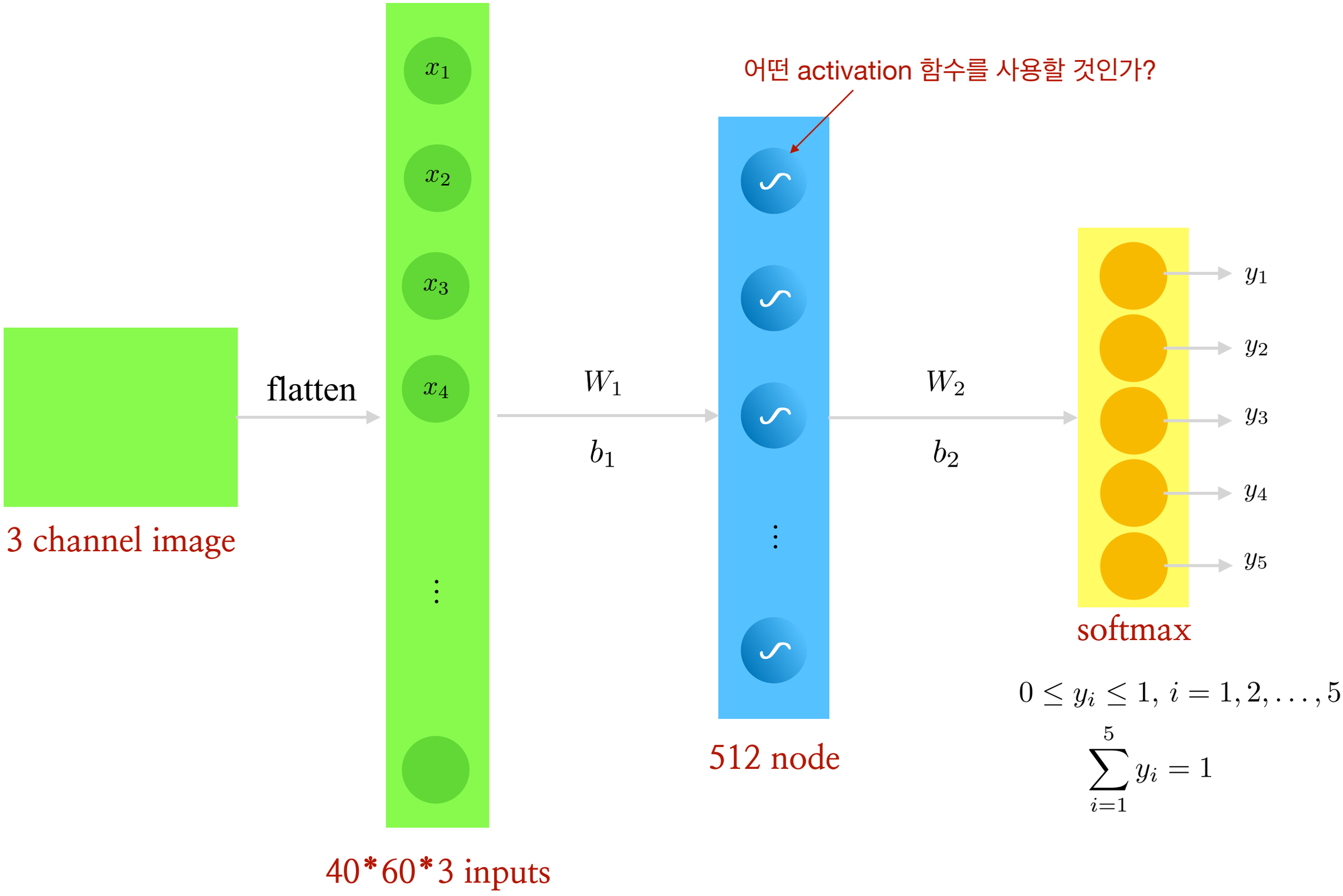
- classification 문제에서 출력 벡터 (우리의 경우 5차원 벡터)는 해당 이미지가 각 클래스에 속할 확률을 표현하면 자연스럽다.
- 임의의 K차원 벡터 $z = [z_1, z_2, \ldots, z_K]$를 크기 순서를 유지하면서 각 원소가 0 ~ 1범위에 들어가고 합이 1이 되도록 변환해주는 함수가 softmax 함수이다.

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

- softmax함수는 classification 문제의 출력층에서 가장 흔히 사용되는 activation 함수이다.

네트워크 설계



$$0 \leq y_i \leq 1,\ i = 1, 2, \ldots, 5$$

$$\sum_{i=1}^{5} y_i = 1$$

Your first neural network

- Relu (Rectified Linear Unit)은 대표적인 activation 함수이며 다음과 같이 정의된다:

$$relu(x) = \max(0, x)$$

- Relu는 sigmoid보다 vanishing gradient 문제에 대해서 효과적이다.

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(IMG_HEIGHT, IMG_WIDTH, NUM_CHANNEL)),
    keras.layers.Dense(512, activation=tf.nn.relu),
    keras.layers.Dense(NUM_CLASS, activation=tf.nn.softmax)
])
```

output layer에서는 activation 함수로 softmax를 적용한다.

activation 함수로는 sigmoid대신 relu를 적용한다.

Your first neural network

AdamOptimizer는 learning rate를 adaptive하게 조절해주는 기능이 추가된 gradient descent 알고리즘이라고 생각하면 된다.

```
model.compile(optimizer=tf.train.AdamOptimizer(),

              loss='sparse_categorical_crossentropy',

              metrics=['accuracy'])
```

label이 one-hot-encode되어 있지 않은 상황에서 cross entropy loss 함수를 사용할 경우 이렇게 한다.

Used to monitor the training and testing steps. The following example uses accuracy, the fraction of the images that are correctly classified.

- Cross entropy 함수는 다음과 같이 정의된다. ($\bar{y}$ 를 one-hot-encoded label, y를 출력이라고 하자. y는 각 클래스에 속할 확률을 나타내는 벡터의 형태를 가진다. 즉, softmax의 출력이다.)

$$\text{cross\_entropy}(y, \bar{y}) = -\sum \bar{y} \log y$$

- Cross entropy 함수는 MSE보다 learning 속도 측면에서 효율적이다.
- 만약 label이 one-hot-encoded되어 있다면 loss = 'categorical_crossentropy'로 하면 된다.
- 그냥 squared error sum을 사용하려면 loss = 'mse'로 하면 된다. 단, label은 먼저 one-hot-encoding되어야 한다.

트레이닝 된 weight를 저장할 파일 이름이다.

```
checkpoint_path = "training/cp-{epoch:04d}.ckpt"

# Create checkpoint callback
cp_callback = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1,
                                                 period=5)
```

5 epoch 마다 weight를 저장하는 일을 할 callback 함수이다.

AdamOptimizer를 이용하여 Gradient Descent 기법으로 학습을 수행한다.

Train 데이터와 label를 주면 알아서 적절한 크기의 batch로 나누어서 stochastic gradient descent 기법을 적용한다.

트레이닝은 100 epoch 동안 진행한다.

```
model.fit(train_features, train_labels, epochs=100,
          validation_data=(test_features,test_labels),
          callbacks=[cp_callback])
```

트레이닝 과정에 실행할 callback 함수를 이렇게 제공한다.

매 epoch가 종료될 때 마다 validation data로 테스트를 수행한다. 여기에서는 그냥 test data를 validation data로 사용하였는데 이건 정석적인 방법은 아니다.

```
# model.load_weights('./training/cp-0010.ckpt')
```


신경망을 트레이닝 하는 대신 이미 트레이닝되어 저장된 weight를 load하여 사용하려면 이렇게 한다.

Test data로 학습된 신경망의 성능을 테스트한다. 우리의 예에서는 Test data로 validation 을 했으므로 트레인 과정에서 마지막 validation과 동일한 결과를 얻게 될 것이다.


```
test_loss, test_acc = model.evaluate(test_features, test_labels)

print('Test accuracy:', test_acc)
```

Train 데이터도 아니고 test data도 아닌 다른 새로운 데이터로 학습된 신경망을 이용해
어떤 동물인지 예측하는 것이다. 이 예에서는 그냥 test data를 다시 사용하였다.


```
predictions = model.predict(test_features)

print(predictions[0])
print(np.argmax(predictions[0]))
print(test_labels[0])
```

결과를 보기 편하게 출력하기 위한 보조 코드들

```
def plot_image(i, predictions_array, true_label, img):
    predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    plt.imshow(img, cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'

    plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                         100 * np.max(predictions_array),
                                         class_names[true_label]), color=color)
```

```
def plot_value_array(i, predictions_array, true_label):
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
thisplot = plt.bar(range(5), predictions_array, color="#777777")
plt.ylim([0, 1])
predicted_label = np.argmax(predictions_array)

thisplot[predicted_label].set_color('red')
thisplot[true_label].set_color('blue')
```

결과를 보기 편하게 출력하기 위한 보조 코드들

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_features)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```

```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_features)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```

```
# Plot the first X test images, their predicted label, and the true label
# Color correct predictions in blue, incorrect predictions in red
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions, test_labels, test_features)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions, test_labels)
```

```
# Grab an image from the test dataset
img = test_features[0]
print(img.shape)

# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))
print(img.shape)
```

```
predictions_single = model.predict(img)
print(predictions_single)

plot_value_array(0, predictions_single, test_labels)
_ = plt.xticks(range(5), class_names, rotation=45)
plt.show()
print(np.argmax(predictions_single[0]))
```