Kaggle CNN 발표

2017010715 허지혜

CNN(Convolution)

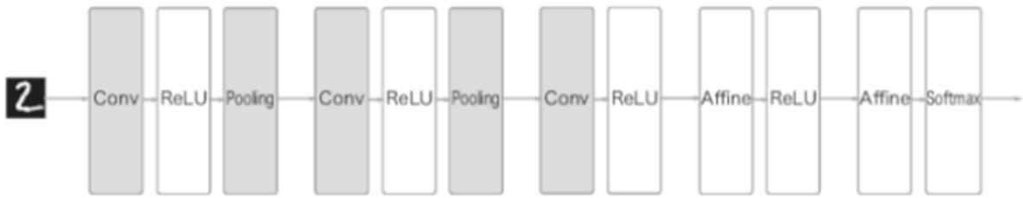
Convolutional base

합성곱층(covolutional layer)과 풀링층(pooling)으로 구성

<기존 계층>

classfy

<합성곱 계층>

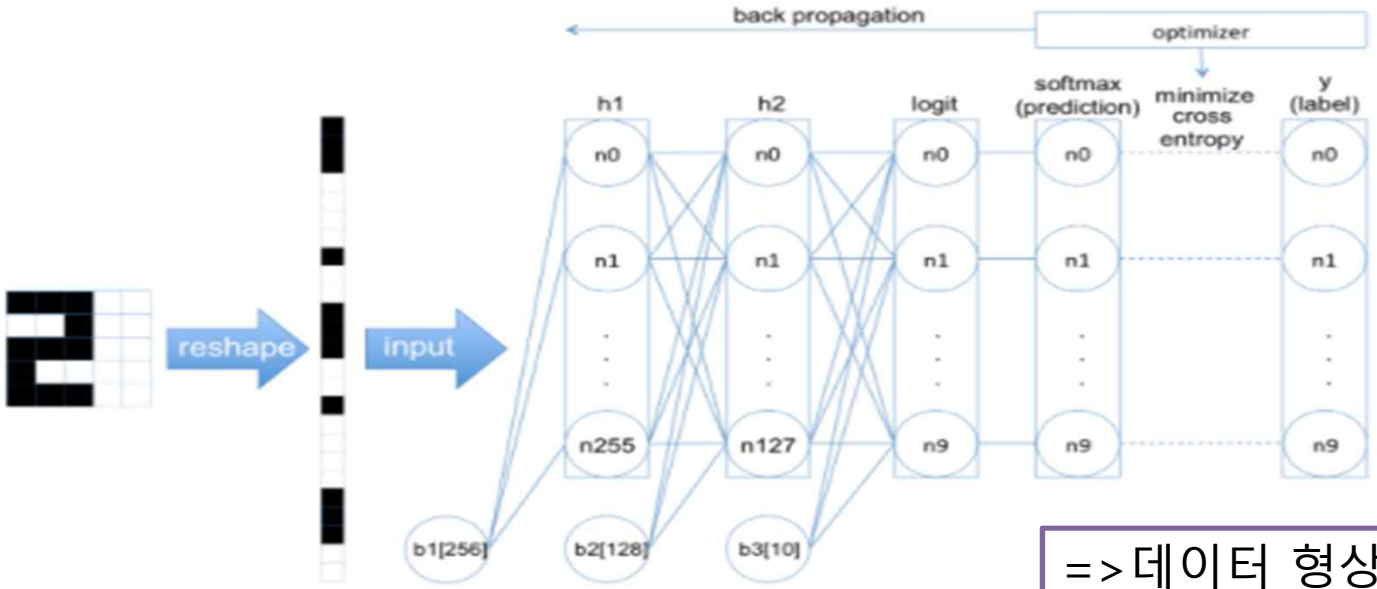


완전연결계층(FC)

모든 계층의 뉴런이 이전 층의 출력 노드와 하나도 빠짐없이 연결되어있는 층

In [2]:
```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/mlp
```

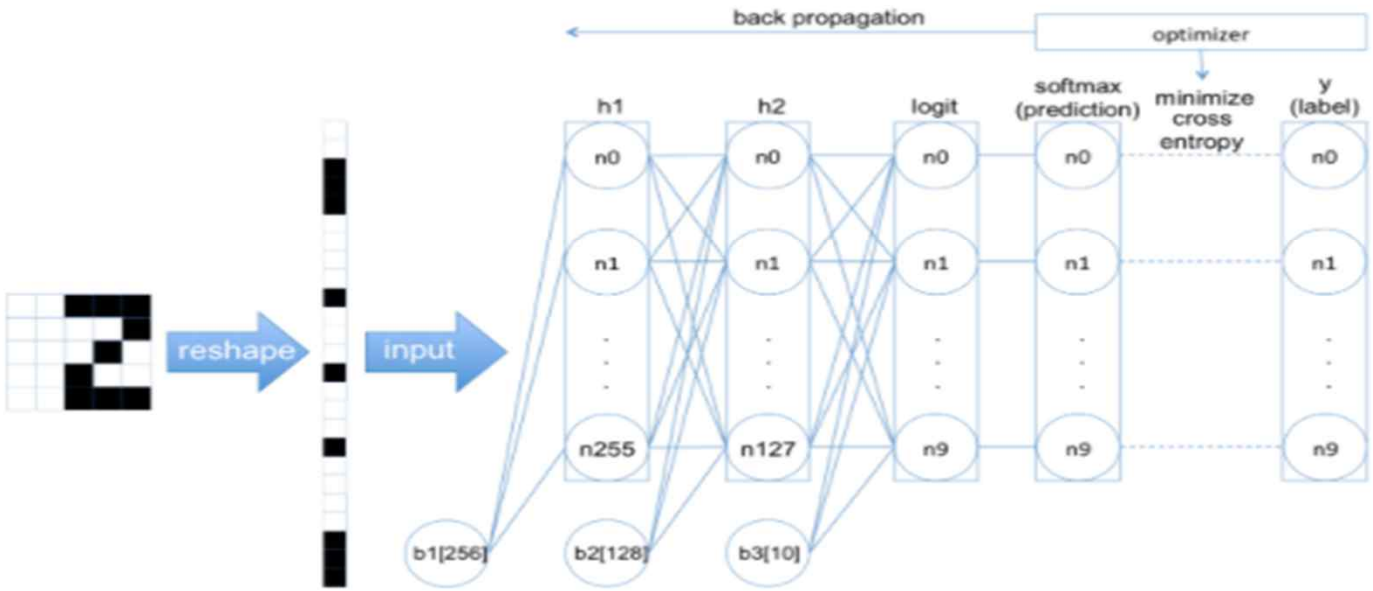
Out [2]:



=>데이터 형상 무시

In [3]:
```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/mlp
```

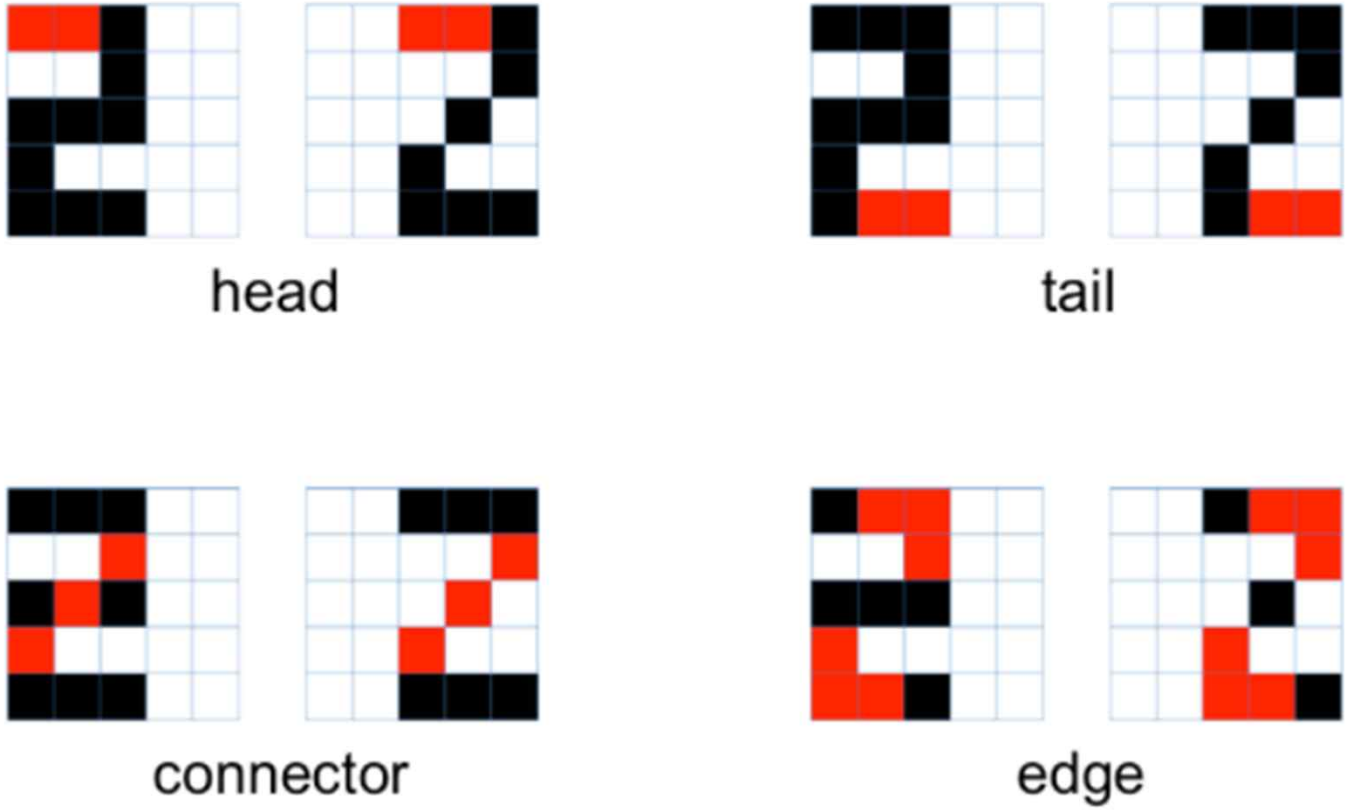
Out [3]:



합성곱층

```
In [4]: Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/cnn
```

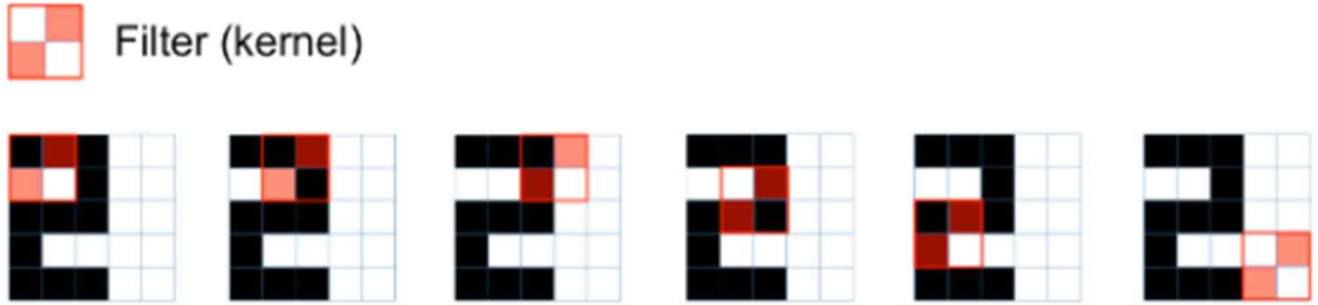
Out [4]:



CNN은 3차원의 이미지를 그대로 입력층 에 받으며
출력 또한 3차원 데이터로 출력하여 다음 계층으로
전달하게 때문에 형상을 가지는 데이터를 제대로 학습할 수 있다

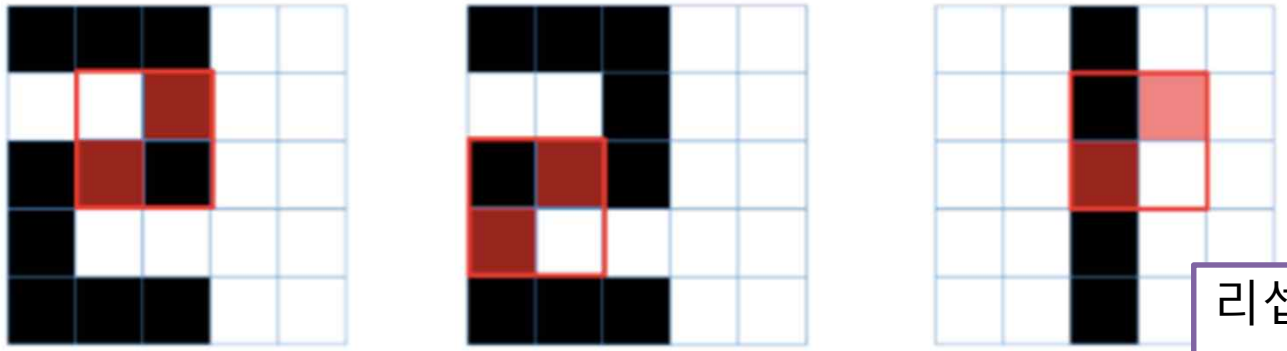
```
In [5]: Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/str
```

Out [5]:




```
In [6]: Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/fil
```

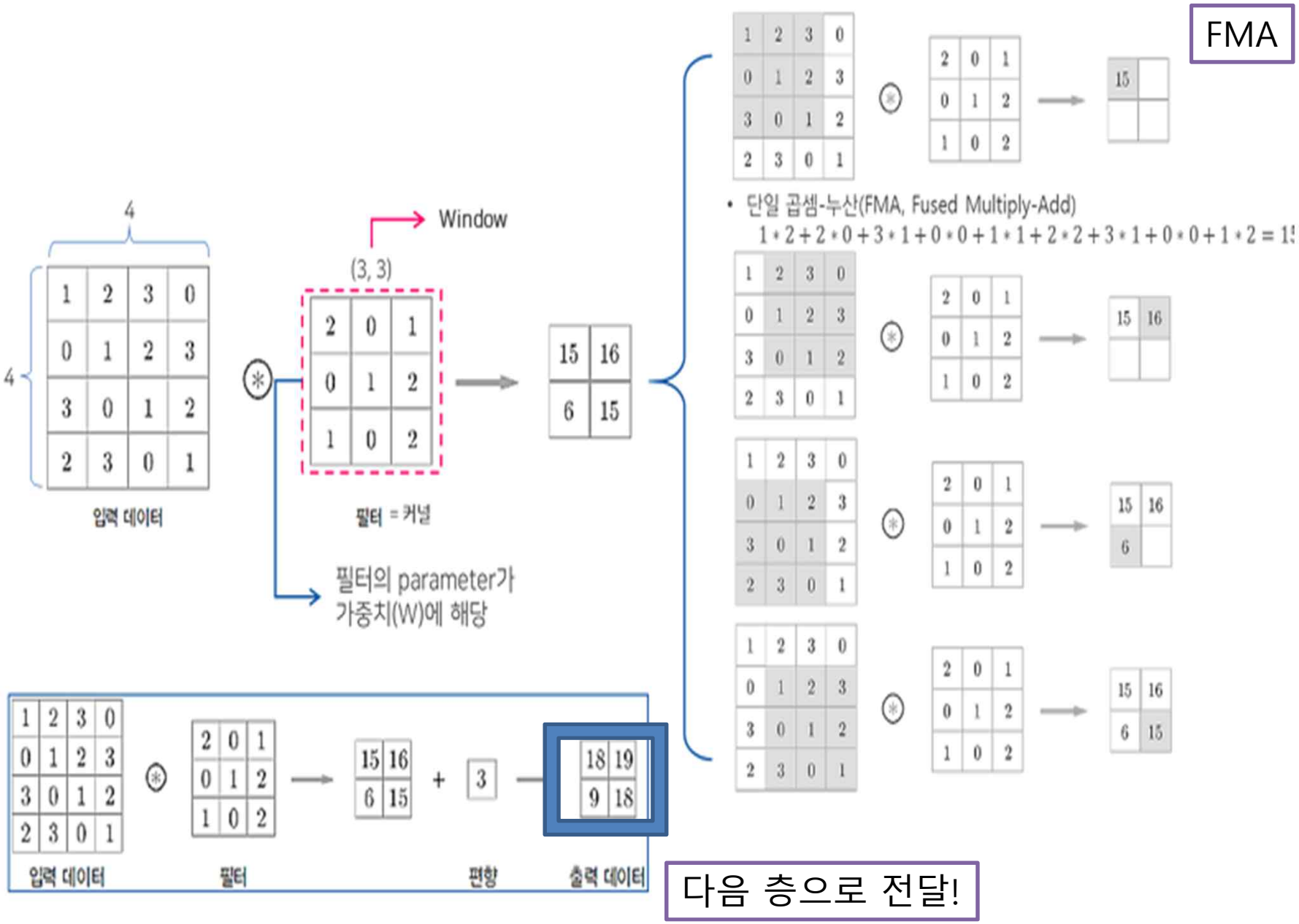
Out [6]:



필터 = 합성곱층에서 가중치 파라미터에 해당하며 학습 단계에서 적절한 특징 찾도록 학습

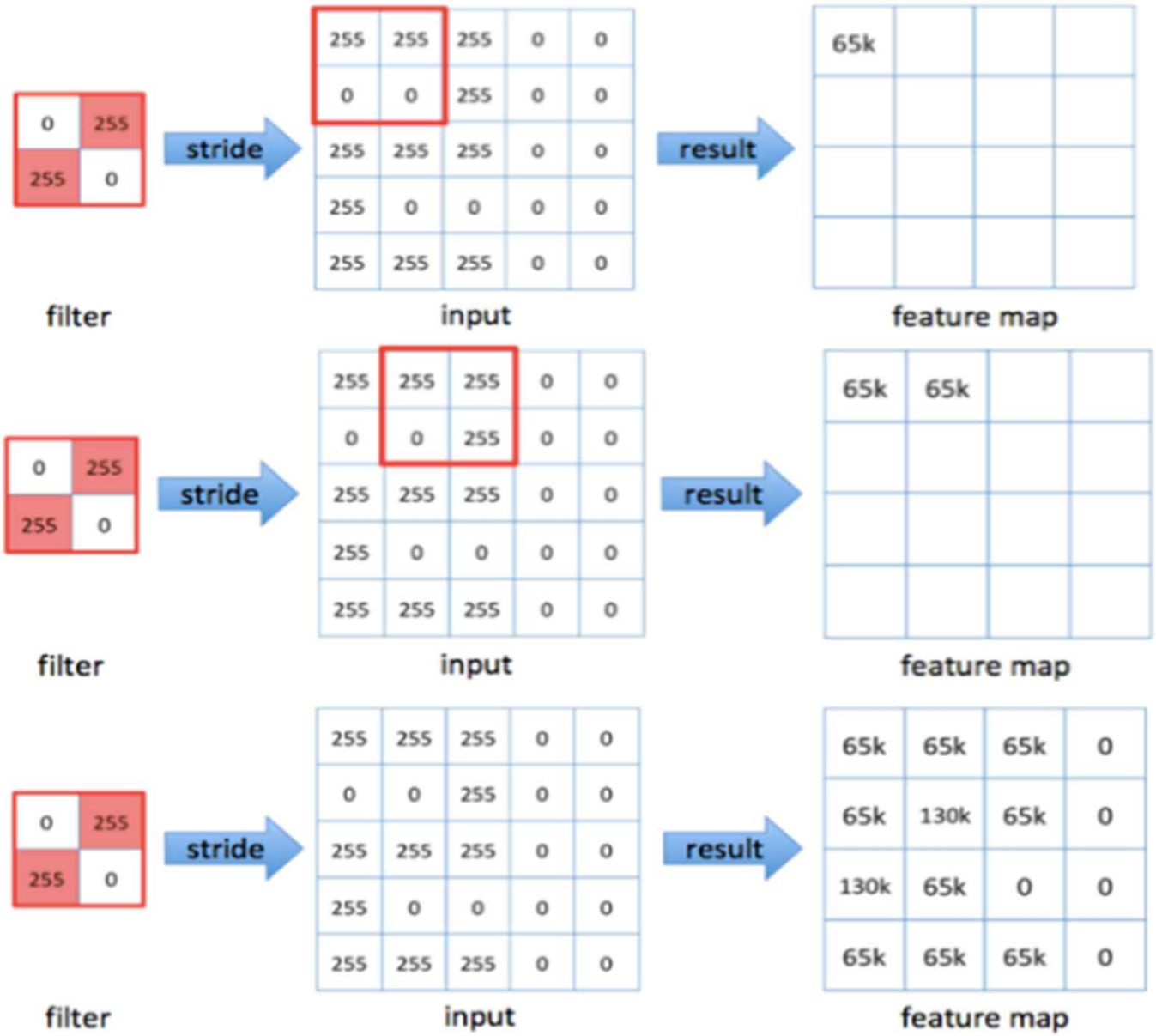
리셉티브 필터

필터를 통해 보는 공간


FMA
4
4
Window
(3, 3)
입력 데이터
필터 = 커널
필터의 parameter가
가중치(W)에 해당
• 단일 곱셈-누산(FMA, Fused Multiply-Add)
1 * 2 + 2 * 0 + 3 * 1 + 0 * 0 + 1 * 1 + 2 * 2 + 3 * 1 + 0 * 0 + 1 * 2 = 15
입력 데이터
필터
편향
출력 데이터
다음 층으로 전달!

In [9]:
```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/str
```

Out[9]:



패딩 : 합성곱 연산 전 입력데이터 주변을 특정값으로 채워 늘리는 것

In [12]:

```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/zer
```

Out[12]:

0	0	0	0	0	0	0
0	255	255	255	0	0	0
0	0	0	255	0	0	0
0	255	255	255	0	0	0
0	255	0	0	0	0	0
0	255	255	255	0	0	0
0	0	0	0	0	0	0

Zero Padding

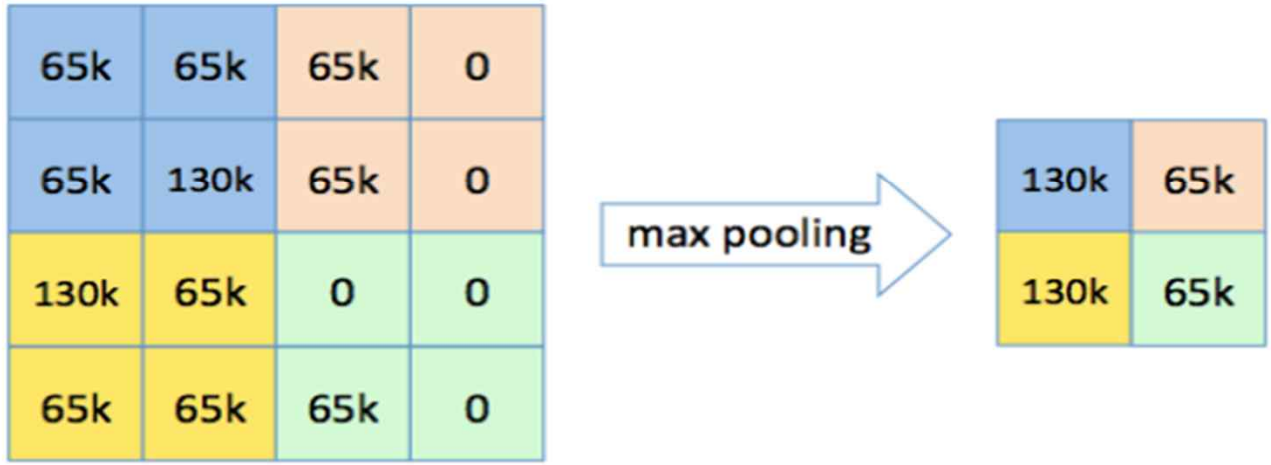
0	0	255
0	255	0
255	0	0

filter

풀링층 : 데이터의 크기 줄이기

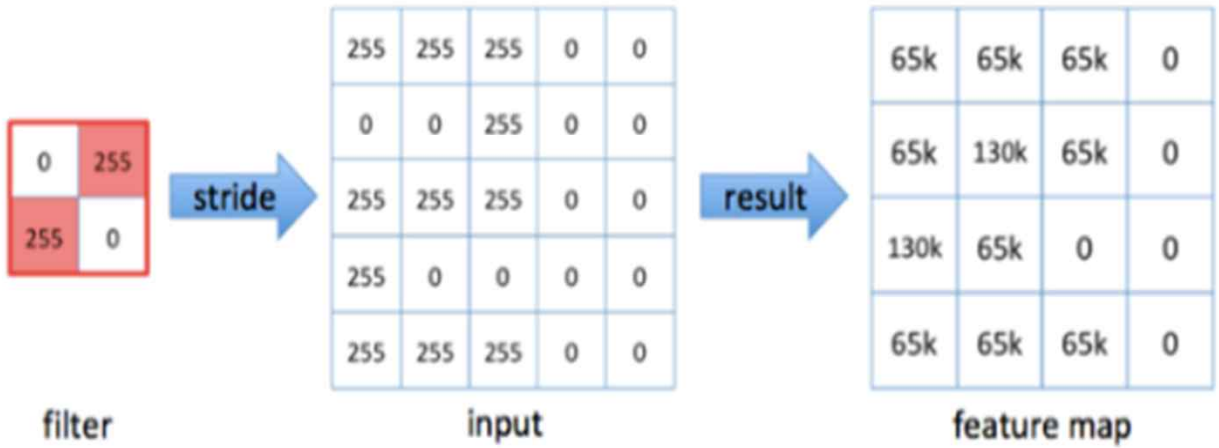
```
In [10]: Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/max
```

Out [10]:



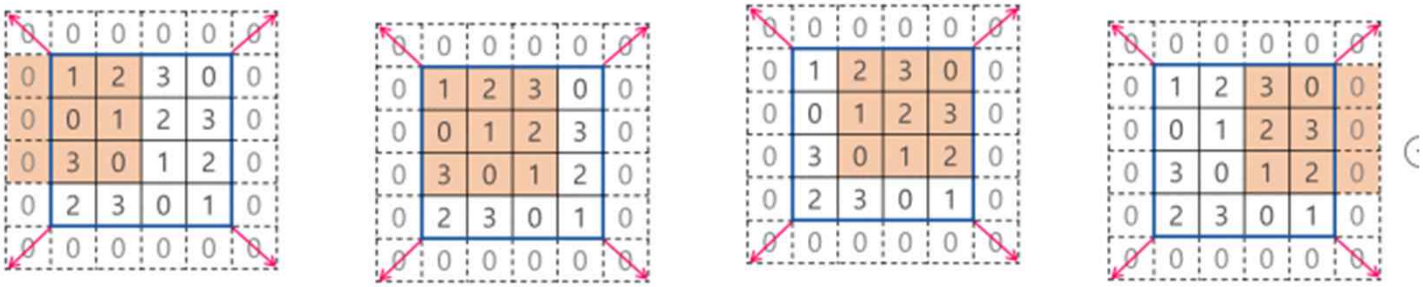

```
In [11]: Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/zer
```

Out [11]:



스트라이드 : 입력데이터에 필터를 적용할 때 이동할 간격

- 크기 조절은 풀링층 계층에서만 조절하게 할 수 있다.



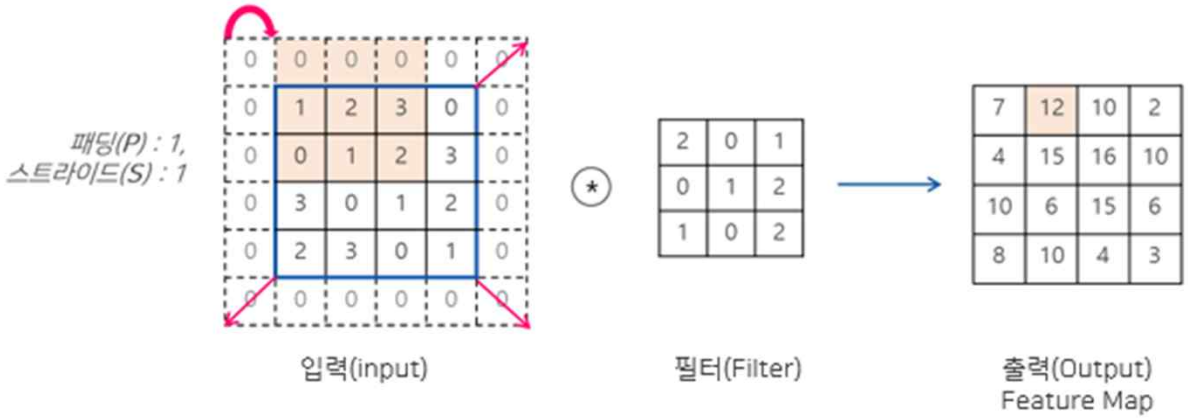
1폭짜리 ZERO-PADDING과 STRIDE값을 1로 적용한 후 합성곱 연산을 수행하는 예제

출력 크기 계산

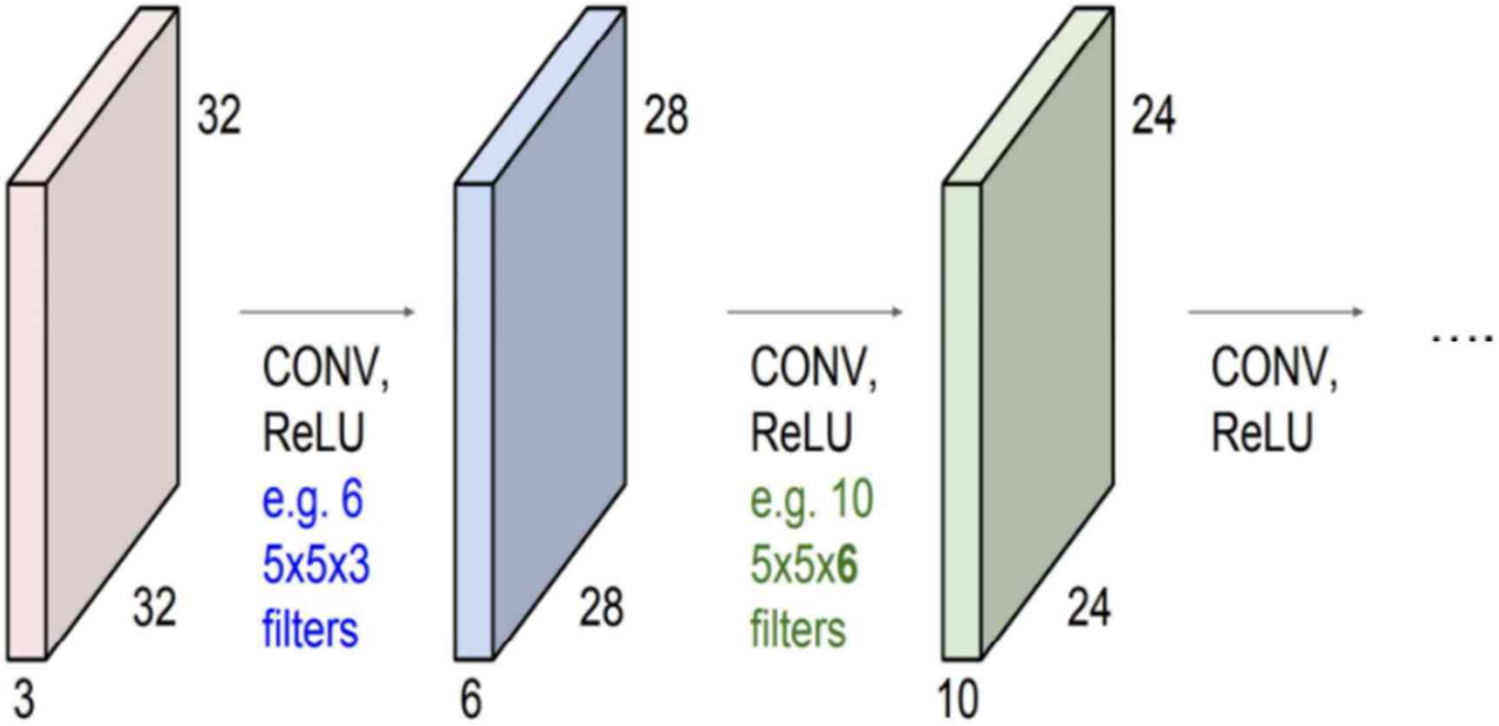
$$(OH, OW) = \left(\frac{H + 2P - FH}{S} + 1, \frac{W + 2P - FW}{S} + 1\right)$$

- (H, W) : 입력 크기 *(input size)*
- (FH, FW) : 필터 크기 *(filter/kernel size)*
- S : 스트라이드 *(stride)*
- P : 패딩 *(padding)*
- (OH, OW) : 출력 크기 *(output size)*

4 4 3 3 1 1



(OH, OW) = ((4 + 2 * 1 – 3)/1 + 1 , (4 + 2 * 1 – 3)/1 + 1)
= (4 , 4)

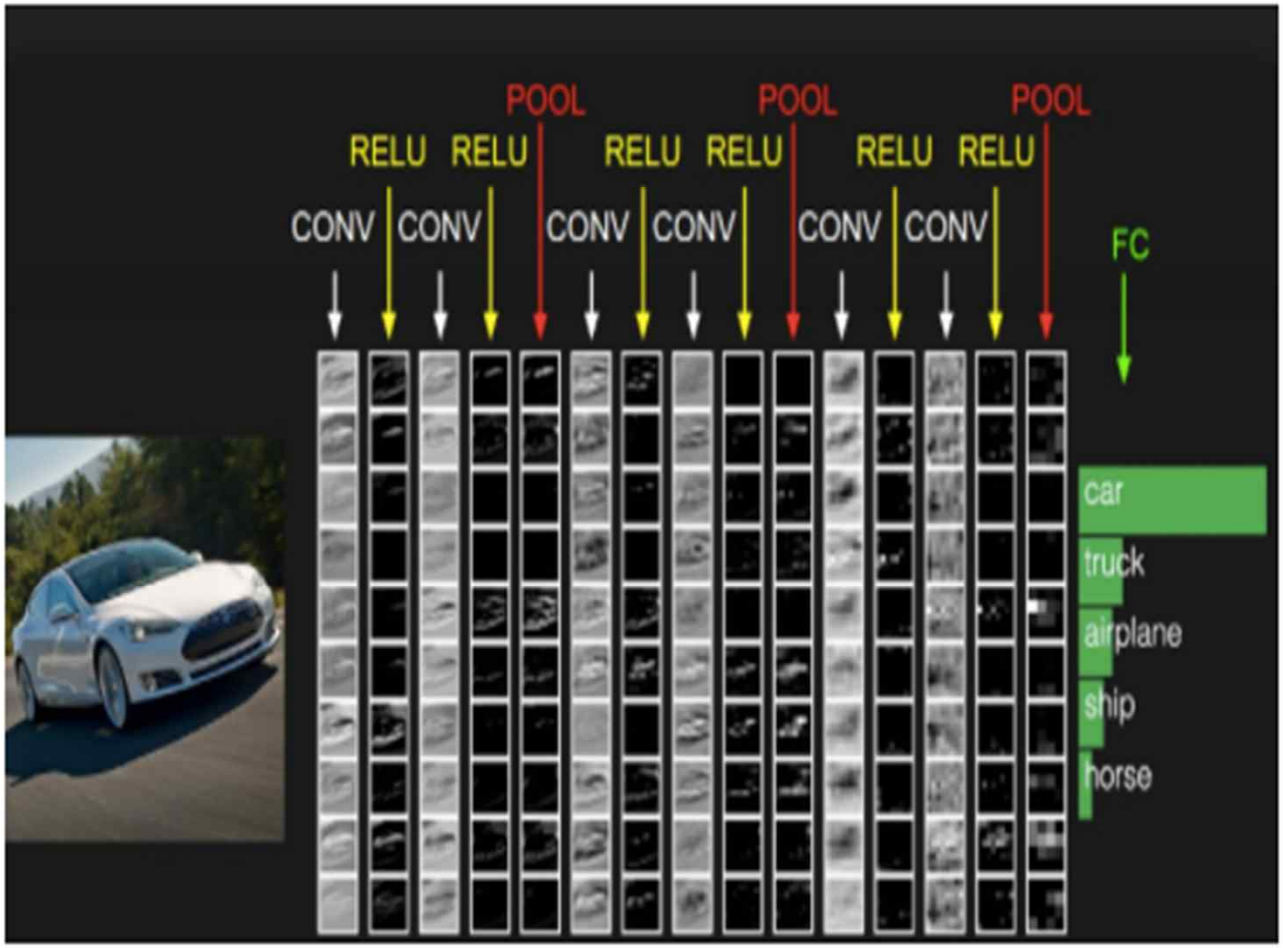


활성화 함수(Activation Function)

필터 들을 통해서 특징 맵이 추출되면 **이 특징 맵에 활성화 함수를 적용**하여 값을 활성화 **있다, 없다를 구분**하기 위해 활성화 함수를 사용하는데 **sigmoid function**와 **ReLU function**가 있다.

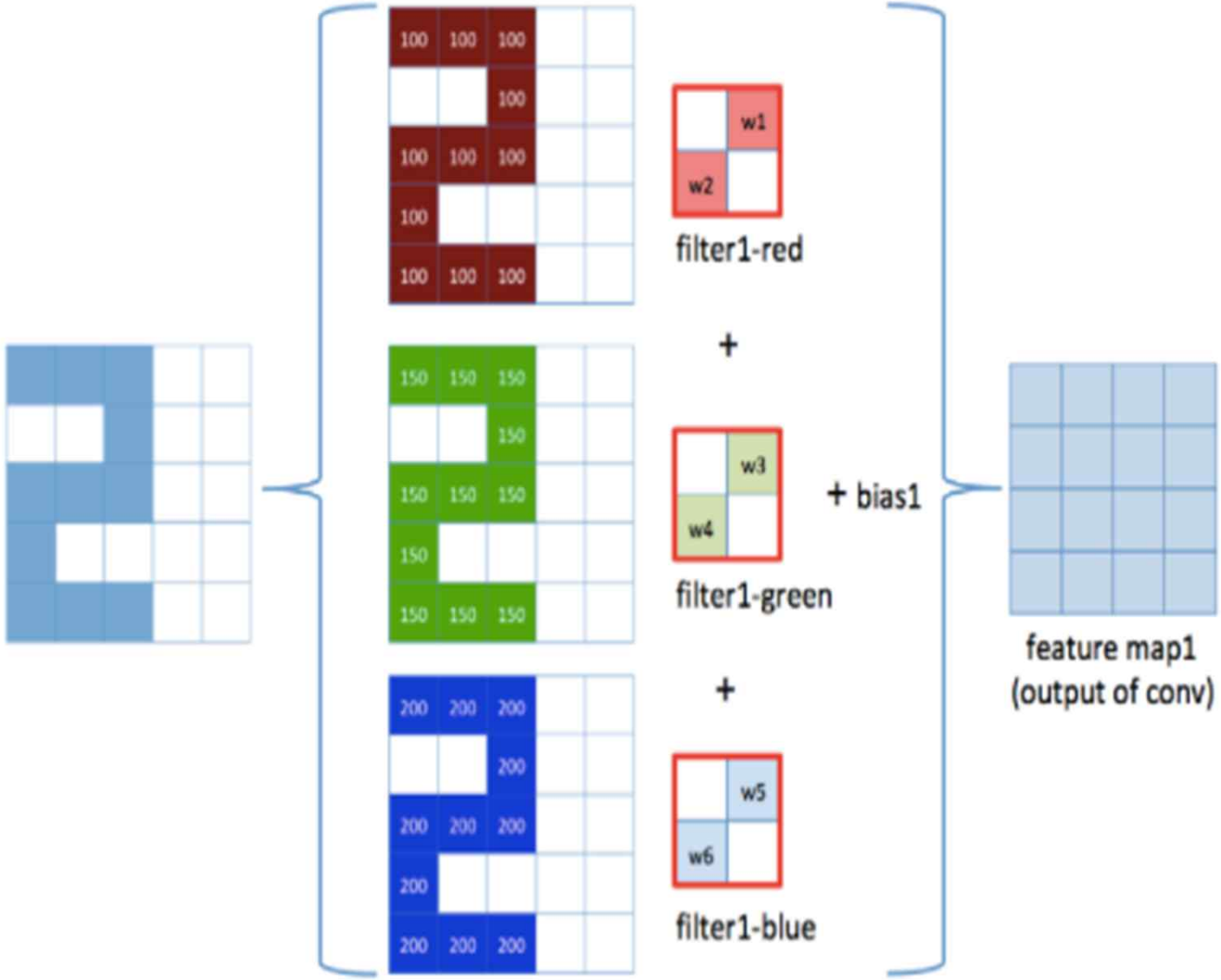
In [8]:
```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/cnn
```

Out[8]:



In [14]:
```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/rgb
```

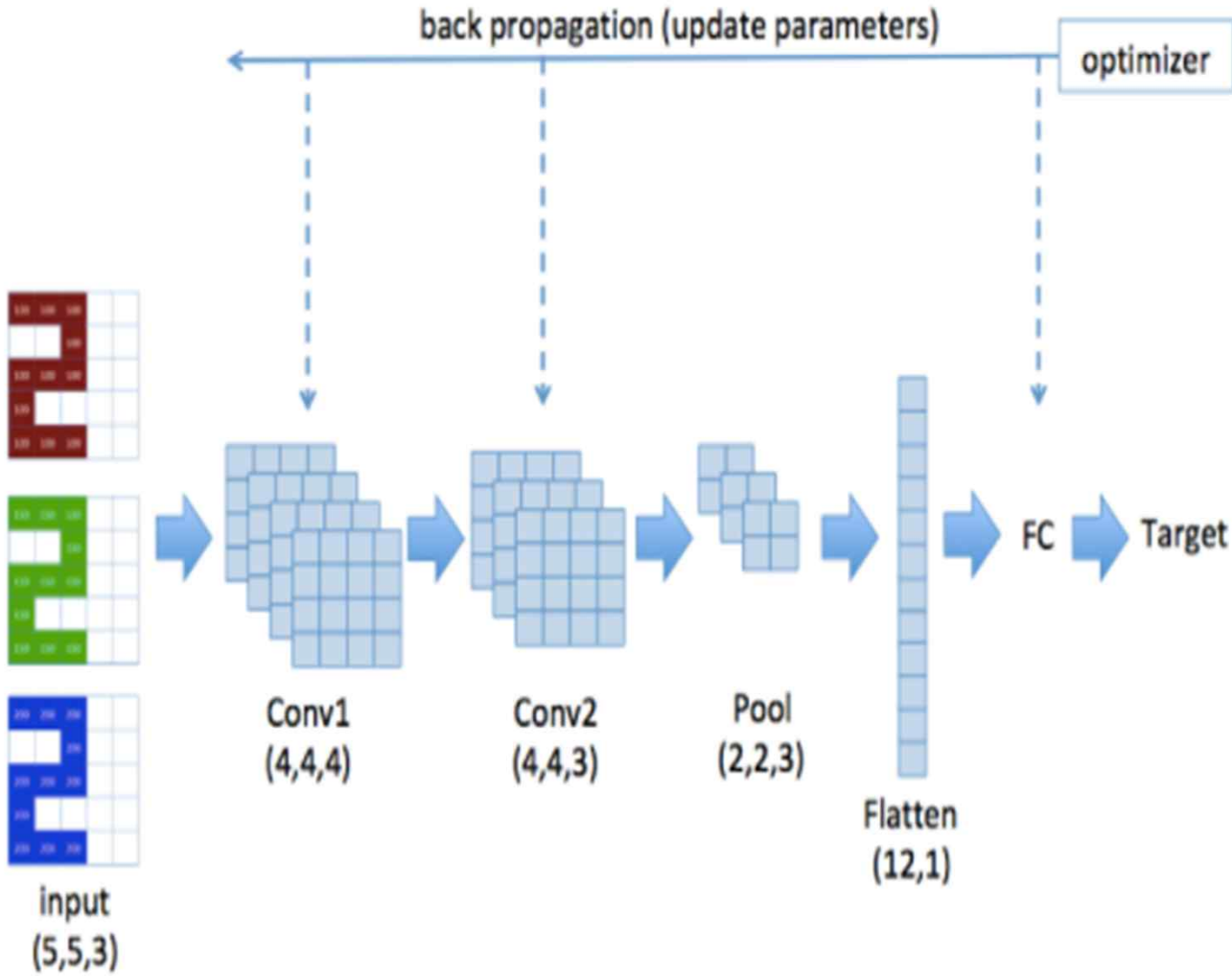
Out[14]:



In [16]:

```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/cnn
```

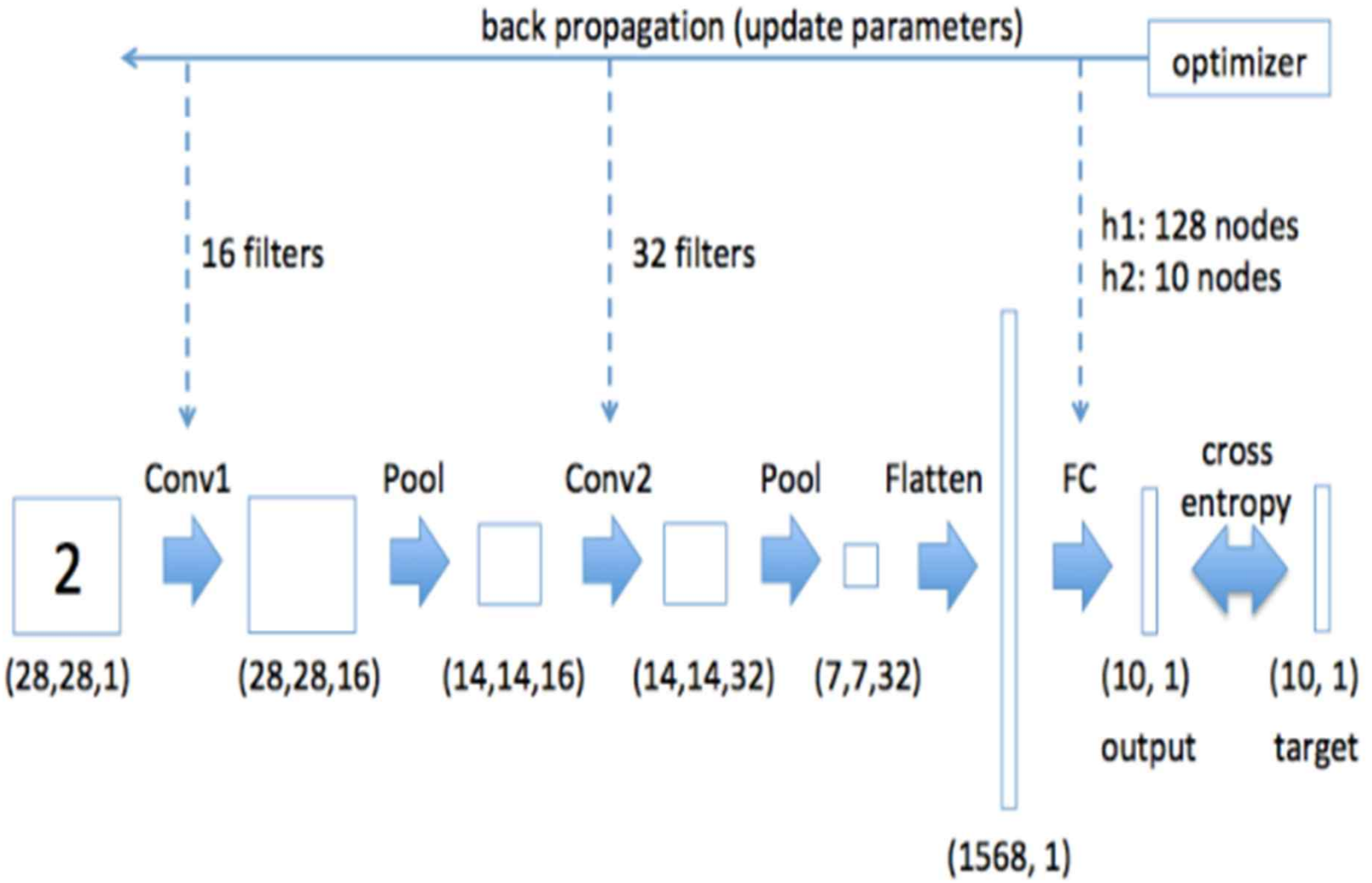
Out[16]:



In [18]:

```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/pra
```

Out[18]:



In [40]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 3s 0us/step
```

In [41]:
```
x_val   = x_train[50000:60000]
x_train = x_train[0:50000]
y_val   = y_train[50000:60000]
y_train = y_train[0:50000]
```

In [42]:
```
print("train data has " + str(x_train.shape[0]) + " samples")
print("every train data is " + str(x_train.shape[1])
      + " * " + str(x_train.shape[2]) + " image")
```

```
train data has 50000 samples
every train data is 28 * 28 image
```

In [43]:
```
print("validation data has " + str(x_val.shape[0]) + " samples")
print("every train data is " + str(x_val.shape[1])
      + " * " + str(x_train.shape[2]) + " image")
```

```
validation data has 10000 samples
every train data is 28 * 28 image
```

In [44]:
```
# sample to show gray scale values
print(x_train[0][8])
```

```
[  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
   0   0   0   0   0   0   0   0   0   0]
```

In [46]:
```
# sample to show labels for first train data to 10th train data
print(y_train[0:11])
```

```
[5 0 4 1 9 2 1 3 1 4 3]
```

reshape

In [47]:
```
import numpy as np
x_train = np.reshape(x_train, (50000,28,28,1))
x_val = np.reshape(x_val, (10000,28,28,1))
x_test = np.reshape(x_test, (10000,28,28,1))

print(x_train.shape)
print(x_test.shape)
```

```
(50000, 28, 28, 1)
(10000, 28, 28, 1)
```

Normalize data

In [49]:
```
x_train = x_train.astype('float32')
x_val = x_val.astype('float32')
x_test = x_test.astype('float32')

gray_scale = 255
x_train /= gray_scale
x_val /= gray_scale
x_test /= gray_scale
```

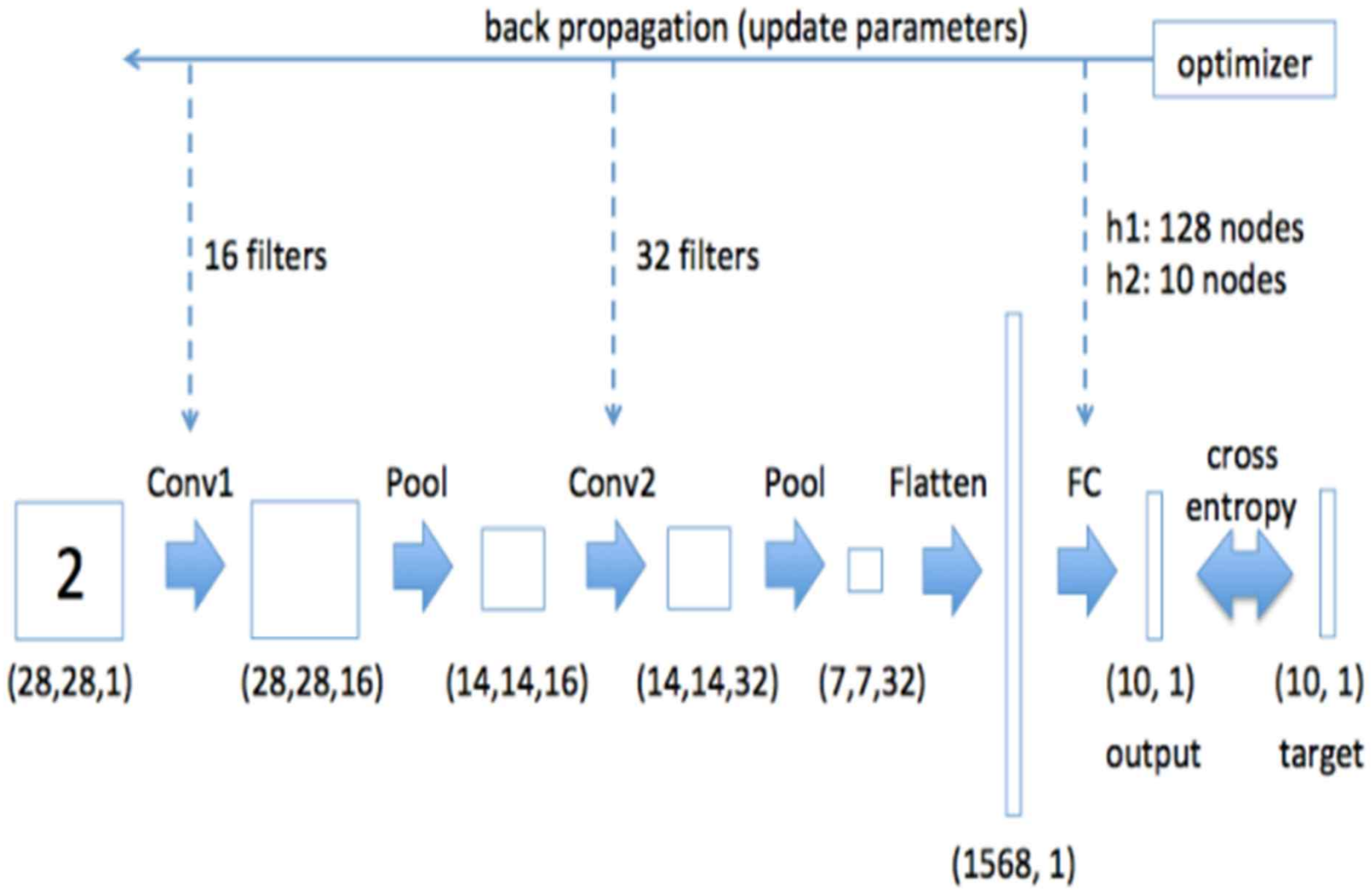
원핫인코딩

In [50]:
```
num_classes = 10
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_val = tf.keras.utils.to_categorical(y_val, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
```

In [18]:

```
Image(url= "https://raw.githubusercontent.com/minsuk-heo/deeplearning/master/img/pra
```

Out[18]:



코드- conv + pool

In [26]:
```
import tensorflow.compat.v1 as tf
tf.compat.v1.disable_eager_execution()
```

In [27]:
```
x = tf.placeholder(tf.float32, shape=[None, 28, 28, 1])
y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

In [28]:
```
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)
```

In [29]:
```
def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')
```

코드 – conv + pool

In [30]:
```
W_conv1 = weight_variable([5, 5, 1, 16])
b_conv1 = bias_variable([16])
```

```
WARNING:tensorflow:From C:₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩resour
ce_variable_ops.py:1666: calling BaseResourceVariable.__init__ (from tensorflow.pyt
hon.ops.resource_variable_ops) with constraint is deprecated and will be removed in
a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

In [31]:
```
h_conv1 = tf.nn.relu(conv2d(x, W_conv1) + b_conv1)
```

In [32]:
```
h_pool1 = max_pool_2x2(h_conv1)
```

In [33]:
```
W_conv2 = weight_variable([5, 5, 16, 32])
b_conv2 = bias_variable([32])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

코드 - FC

FC

In [34]:
```
W_fc1 = weight_variable([7 * 7 * 32, 128])
b_fc1 = bias_variable([128])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*32])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

In [35]:
```

W_fc2 = weight_variable([128, 10])
b_fc2 = bias_variable([10])

y_conv = tf.matmul(h_fc1, W_fc2) + b_fc2
```

In [36]:
```
cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits_v2(labels=y_, logits=y_conv))
```

In [37]:
```
train_step = tf.train.AdamOptimizer(0.001).minimize(cross_entropy)
```

In [38]:
```
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

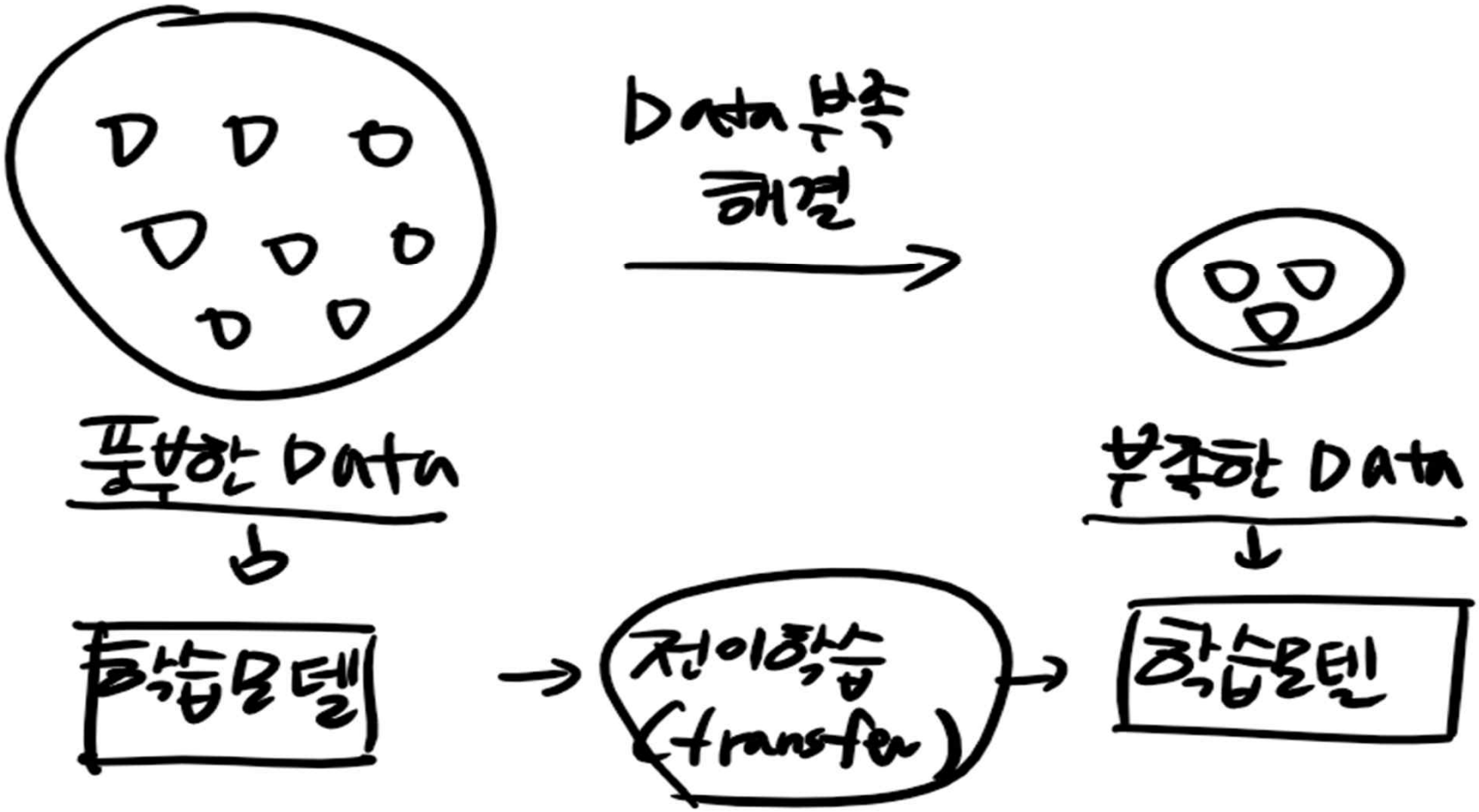
Train and Test

In [52]:

```
# initialize
init = tf.global_variables_initializer()

# train hyperparameters
epoch_cnt = 3
batch_size = 500
iteration = len(x_train) // batch_size

# Start training
with tf.Session() as sess:
    tf.set_random_seed(777)
    # Run the initializer
    sess.run(init)
    for epoch in range(epoch_cnt):
        avg_loss = 0.
        start = 0; end = batch_size

        for i in range(iteration):
            if i%10 == 0:
                train_accuracy = accuracy.eval(feed_dict={x:x_train[start: end], y_:
                print("step "+ str(i) + ": training accuracy: "+str(train_accuracy))
            train_step.run(feed_dict={x:x_train[start: end], y_: y_train[start: end]
            start += batch_size; end += batch_size

        # Validate model
        val_accuracy = accuracy.eval(feed_dict={x:x_val, y_: y_val})
        print("validation accuracy: "+str(val_accuracy))

    test_accuracy = accuracy.eval(feed_dict={x:x_test, y_: y_test})
    print("test accuracy: "+str(test_accuracy))
```

```
step 0: training accuracy: 0.1
step 10: training accuracy: 0.704
step 20: training accuracy: 0.82
step 30: training accuracy: 0.856
step 40: training accuracy: 0.878
step 50: training accuracy: 0.904
step 60: training accuracy: 0.908
step 70: training accuracy: 0.948
step 80: training accuracy: 0.938
step 90: training accuracy: 0.93
validation accuracy: 0.9496
step 0: training accuracy: 0.938
step 10: training accuracy: 0.958
step 20: training accuracy: 0.946
step 30: training accuracy: 0.956
step 40: training accuracy: 0.962
step 50: training accuracy: 0.958
step 60: training accuracy: 0.97
step 70: training accuracy: 0.962
step 80: training accuracy: 0.97
step 90: training accuracy: 0.962
validation accuracy: 0.971
step 0: training accuracy: 0.97
step 10: training accuracy: 0.972
step 20: training accuracy: 0.962
step 30: training accuracy: 0.964
step 40: training accuracy: 0.97
step 50: training accuracy: 0.976
step 60: training accuracy: 0.984
step 70: training accuracy: 0.974
step 80: training accuracy: 0.978
step 90: training accuracy: 0.966
validation accuracy: 0.9768
test accuracy: 0.9774
```

전이학습



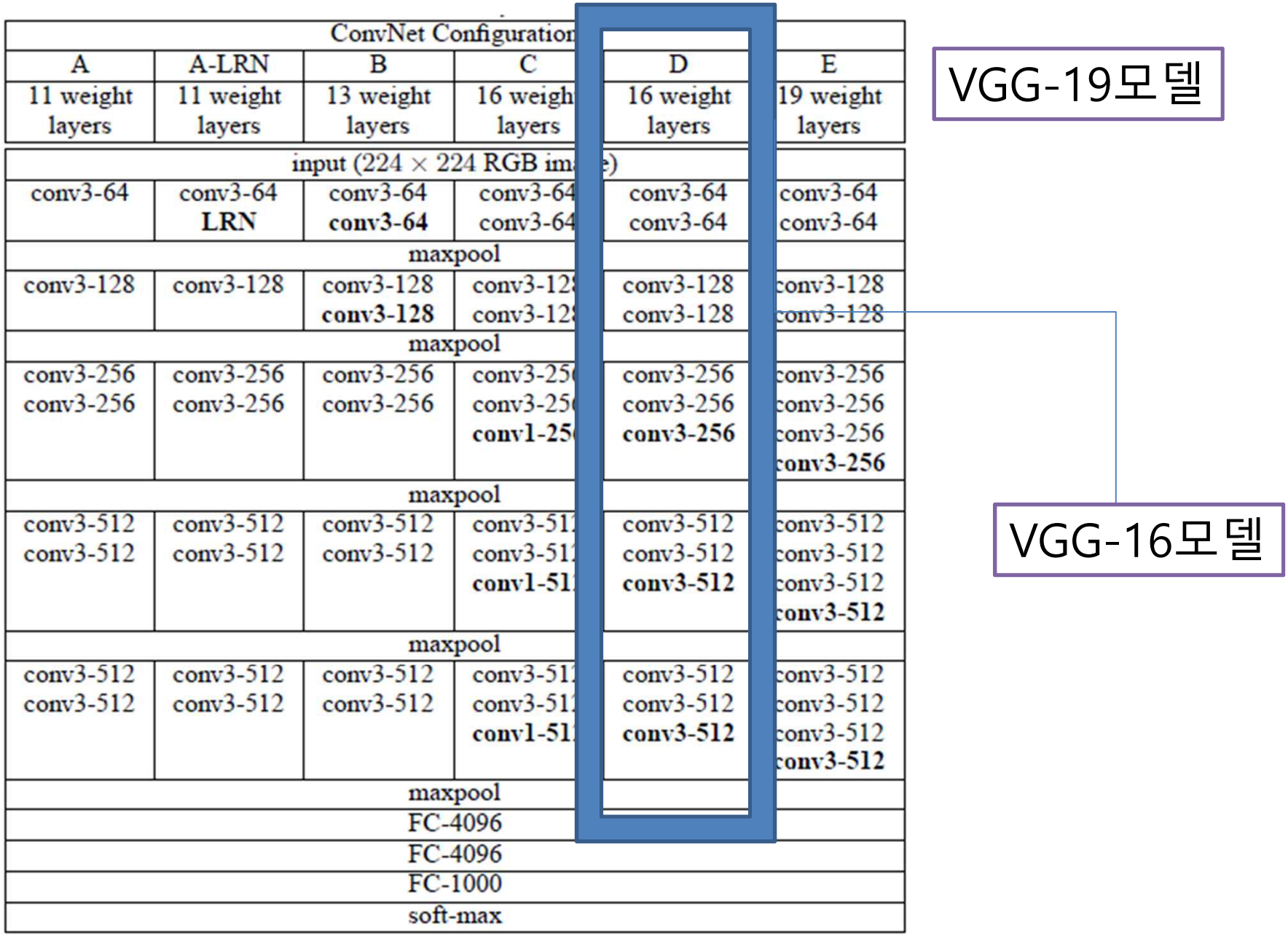
전이학습

Transfer learning

VGG, ResNet, gooGleNet 등 사전에 학습이 완료된 모델을 가지고 우리가 원하는 학습에 미세조정, 즉 작은 변화를 이용하여 학습시키는 방법

이미 학습된 weight들을 transfer하여 자신의 모델에 맞게 학습을 시키는 방법

신경망의 이러한 재학습 과정을 fine-tuning 이라고 부름



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weigh layers	16 weight layers	19 weight layers
input (224 × 224 RGB im e)					
conv3-64	conv3-64 LRN	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-12 conv3-12	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-25 conv3-25 **conv1-25**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-51 conv3-51 **conv1-51**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-51 conv3-51 **conv1-51**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

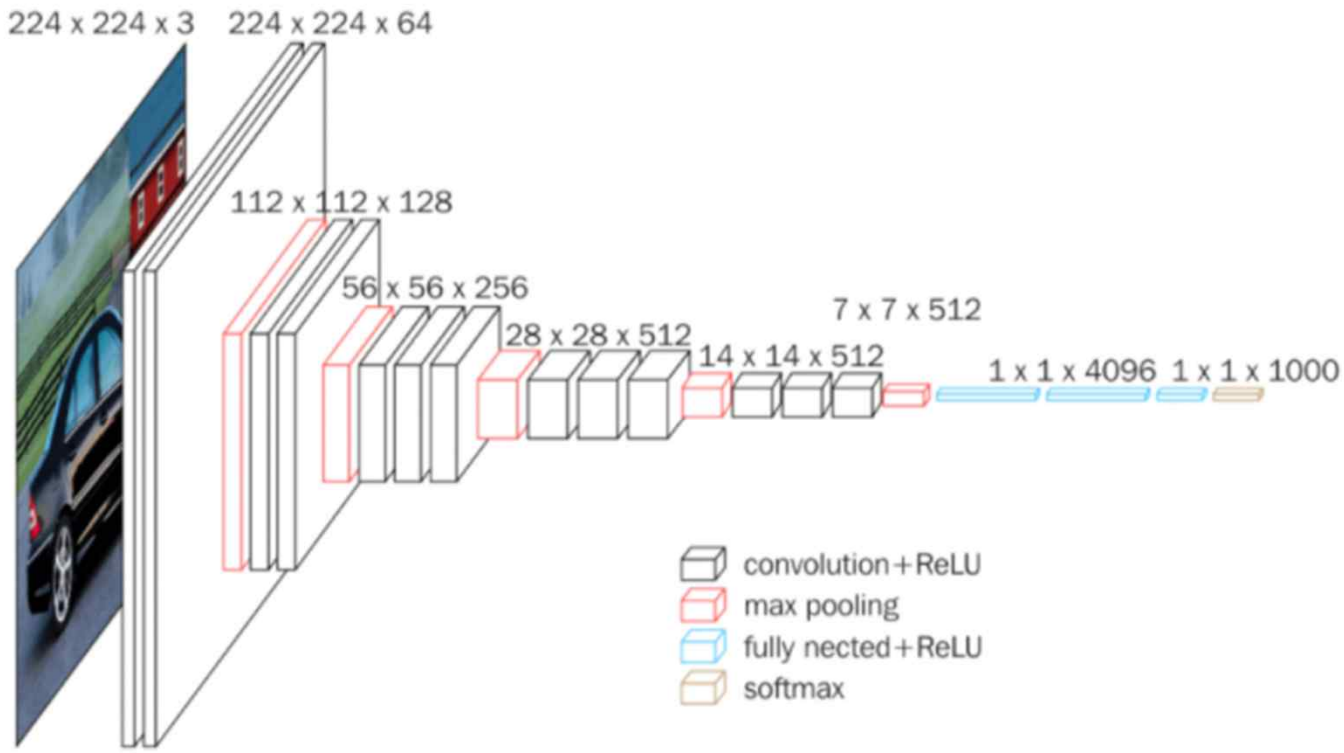
VGG-19모델

VGG-16모델

VGGNet 구조 설명 표 [출처: original 논문]

깊어질수록 성능이 좋아짐

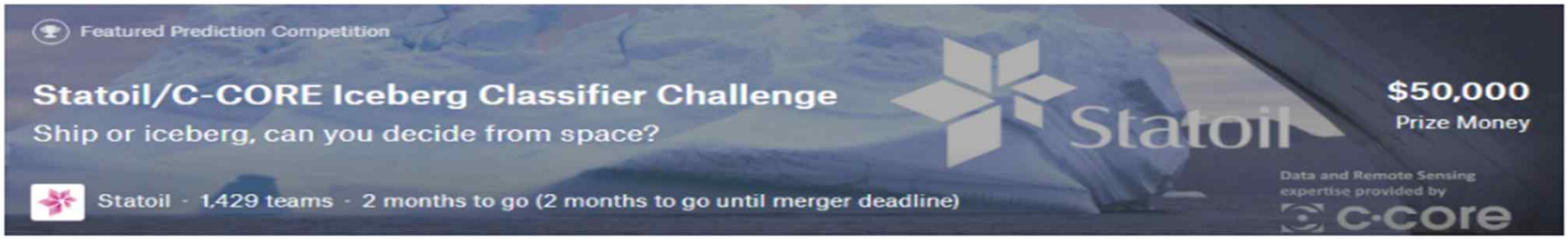
VGG16 구조 분석



VGG16 구조 [6]

출처 : https://bskyvision.com/504

배인지 빙하인지 분류하기



표류하는 빙산은 캐나다 동부 해안과 같은 지역에서 항해 및 활동에 위협이됩니다.

현재 많은 기관과 회사가 항공 정찰 및 해안 기반 지원을 사용하여 환경 조건을 모니터링하고 빙산의 위험을 평가합니다. 그러나 특히 혹독한 날씨가있는 외딴 지역에서는 이러한 방법이 가능하지 않으며 유일하게 실행 가능한 모니터링 옵션은 위성을 통하는 것입니다.

전 세계적으로 운영되는 국제 에너지 회사 인 Statoil은 C-CORE와 같은 회사와 긴밀하게 협력 해 왔습니다. C-CORE는 30 년 이상 위성 데이터를 사용해 왔으며 컴퓨터 비전 기반 감시 시스템을 구축했습니다. 운영을 안전하고 효율적으로 유지하기 위해 Statoil은 기계 학습을 사용하여 위협이되는 빙산을 가능한 한 빨리 감지하고 차별하는 방법에 대한 새로운 관점을 얻는 데 관심이 있습니다.

이 대회에서는 원격으로 감지 된 표적이 선박인지 빙산인지 자동으로 식별하는 알고리즘을 구축해야합니다. 개선 된 사항은 안전한 작업 조건을 유지하기위한 비용을 줄이는 데 도움이 될 것입니다.

24.Transfer Learning with VGG-16 CNN+AUG LB 0.1712

패키지 불러오기

sample_submission.csv	2020-09-07 오후 10:03	Microsoft Excel ...	116KB
test.json	2020-09-07 오후 10:04	JSON 파일	1,486,106...
train.json	2020-09-07 오후 10:04	JSON 파일	191,713KB

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import log_loss
from sklearn.model_selection import StratifiedKFold, StratifiedShuffleSplit
from os.path import join as opj
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pylab
plt.rcParams['figure.figsize'] = 10, 10
%matplotlib inline
```

In [4]:
```
train = pd.read_json('C:/Users/HOME/Desktop/수DA쟁이/배, 빙하 분류하기/Data/train.js
target_train=train['is_iceberg']
test = pd.read_json("C:/Users/HOME/Desktop/수DA쟁이/배, 빙하 분류하기/Data/test.json
```

데이터 확인

```
train.head()
```

Train_target

	id	band_1	band_2	inc_angle	is_iceberg
0	dfd5f913	[-27.878360999999998, -27.15416, -28.668615, -...	[-27.154118, -29.537888, -31.0306, -32.190483,...	43.9239	0
1	e25388fd	[-12.242375, -14.920304999999999, -14.920363, ...	[-31.506321, -27.984554, -26.645678, -23.76760...	38.1562	0
2	58b2aaa0	[-24.603676, -24.603714, -24.871029, -23.15277...	[-24.870956, -24.092632, -20.653963, -19.41104...	45.2859	1
3	4cfc3a18	[-22.45 -23.99			
4	271f93f4	[-26.00 -23.16			

입사각

```
test.head()
```

	id	band_1	band_2	inc_angle
0	5941774d	[-15.863251, -15.201077, -17.887735, -19.17248...	[-21.629612, -21.142353, -23.908337, -28.34524...	34.966400
1	4023181e	[-26.058969497680664, -26.058969497680664, -26...	[-25.754207611083984, -25.754207611083984, -25...	32.615072
2	b20200e4	[-14.14109992980957, -15.064241409301758, -17....	[-14.74563980102539, -14.590410232543945, -14....	37.505433
3	e7f018bb	[-12.167478, -13.706167, -16.54837, -13.572674...	[-24.32222, -26.375538, -24.096739, -23.8769, ...	34.473900
4	4371c8c3	[-23.37459373474121, -26.02718162536621, -28.1...	[-25.72234344482422, -27.011577606201172, -23....	43.918874

자료형 변환

```
test['inc_angle']=pd.to_numeric(test['inc_angle'], errors='coerce')
train['inc_angle']=pd.to_numeric(train['inc_angle'], errors='coerce')
```

문자열 칼럼을 숫자로 변환하는 방법

값 중에 몇 개의 실수로 된 문자열이 아니라
문자로 된 문자열이 포함 되어있다고 할 때
문자열을 숫자로 피싱할 수 없다며 ValueError가 뜨는데
에러를 방지하기 위해서 errors = 'coerce' 옵션을 추가

결측치 채우기

```
train['inc_angle']=train['inc_angle'].fillna(method='pad')
```

```
X_angle=train['inc_angle'] # train_x값
test['inc_angle']=pd.to_numeric(test['inc_angle'], errors='coerce')
X_test_angle=test['inc_angle'] # test_x값
```

결측값을 앞 방향으로 채우기 :
fillna(method="ffill" or "pad")

결측값을 뒷 방향으로 채우기 :
fillna(method="bfill" or "backfill")

Train, test 정규화

여러 배열 합치기

```
In [15]: X_band_1=np.array([np.array(band).astype(np.float32).reshape(75, 75)
                            for band in train["band_1"]])
         X_band_2=np.array([np.array(band).astype(np.float32).reshape(75, 75)
                            for band in train["band_2"]])
         X_band_3=(X_band_1+X_band_2)/2
         #X_band_3=np.array([np.full((75, 75), angel).astype(np.float32)
         #for angel in train["inc_angle"]])
         X_train = np.concatenate([X_band_1[:, :, :, np.newaxis]
                                   , X_band_2[:, :, :, np.newaxis]
                                   , X_band_3[:, :, :, np.newaxis]], axis=-1)
```

```
In [15]: X_band_test_1=np.array([np.array(band).astype(np.float32).reshape(75, 75)
                                 for band in test["band_1"]])
         X_band_test_2=np.array([np.array(band).astype(np.float32).reshape(75, 75)
                                 for band in test["band_2"]])
         X_band_test_3=(X_band_test_1+X_band_test_2)/2
         #X_band_test_3=np.array([np.full((75, 75), angel).astype(np.float32)
         #for angel in test["inc_angle"]])
         X_test = np.concatenate([X_band_test_1[:, :, :, np.newaxis]
                                  , X_band_test_2[:, :, :, np.newaxis]
                                  , X_band_test_3[:, :, :, np.newaxis]], axis=-1)
```

Band_1 : HH로 찍은 이미지

Band_2 : HV로 찍은 이미지

다중 축에 축을 임의로 추가할 수 있는 유사인덱스 행렬의 차원이 커진다고 생각

필요한 패키지

```
# 필요한 패키지 개많네
from matplotlib import pyplot
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Input, Flatten, Acti
from keras.layers import GlobalMaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.merge import Concatenate
from keras.models import Model
from keras import initializers
from keras.optimizers import Adam
from keras.optimizers import RMSprop
from keras.layers.advanced_activations import LeakyReLU, PReLU
from keras.optimizers import SGD
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping

from keras.datasets import cifar10
from keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg16 import VGG16
from keras.applications.xception import Xception
from keras.applications.mobilenet import MobileNet
from keras.applications.vgg19 import VGG19
from keras.layers import Concatenate, Dense, LSTM, Input, concatenate
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
```

```
#Data Aug for multi-input 다중입력을 통한 이미지 확대?
from keras.preprocessing.image import ImageDataGenerator
batch_size=64
```

이미지 변환 정의

```
# horizontal_flip, rescale, shear_range,zoom_range를 설정해줌으로써 해당 데이터의
#전처리를 어떻게 할지 결정
# horizontal_flip : Boolean. Randomly flip inputs horizontally.(가로 무작위 뒤집기)
```

데이터 부풀리기 함수

```
gen = ImageDataGenerator(horizontal_flip = True, # 가로 무작위 뒤집기
                         vertical_flip = True, # 수직 방향 뒤집기
                         width_shift_range = 0., # 수평방향 범위내에서 이미지 이동
                         height_shift_range = 0., #수직방향 범위내에서 이미지 이동
                         channel_shift_range=0,
                         zoom_range = 0.2,  # 확대축소 범위 내에서 확대축소
                         rotation_range = 10) # 지정 각도 내에서 원본 이미지 회전
```

데이터를 늘리고, 변형시킴
=> 새로운 학습 데이터를 만든다.

출처 : https://tykimos.github.io/2017/06/10/CNN_Data_Augmentation/

두 생성기 병합 함수

```
#y배열과 각도 배열에대해 동일한 랜덤 시드를 가진 정확히 동일한 생성기를 사용합니다.
def gen_flow_for_two_inputs(X1, X2, y):
    genX1 = gen.flow(X1,y,  batch_size=batch_size,seed=55)
    genX2 = gen.flow(X1,X2, batch_size=batch_size,seed=55)
    while True:
            X1i = genX1.next()
            X2i = genX2.next()
            #Assert 배열은 동일하다 -
            #안심할 수 있지만 훈련속도가 느려짐
            #np.testing.assert_array_equal(X1i[0],X2i[0])
            yield [X1i[0], X2i[1]], X1i[1]
```

마지막 생성기 생성 : 가장 좋은 모델을 픽하기 위해!

```
#early stopping = epoch를 많이 돌린 후 특정 시점에서 멈추게 하기
#val_loss : validation set의 loss를 모니터링한다는 의미
#patience는 성능이 증가하지 않는 epoch를 몇 번이나 허용할 것인가를 의미
def get_callbacks(filepath, patience=2):
   # performance를 정의하고 최소화하는 es
   es = EarlyStopping('val_loss', patience=10, mode="min")
   #가장 좋은 모델 저장
   msave = ModelCheckpoint(filepath, save_best_only=True)
   return [es, msave]
```

모델 정의

```
def getVggAngleModel():
    input_2 = Input(shape=[1], name="angle")
    angle_layer = Dense(1, )(input_2)
    #사전 훈련된 VGG16모델 사용
    # 로딩할 가중치 =imagenet
    base_model = VGG16(weights='imagenet', include_top=False,
                       input_shape=X_train.shape[1:], classes=1)
    # block5_pool : 물체 특성을 뽑는 층까지만 떼서 쓰기 위한 코드
    x = base_model.get_layer('block5_pool').output


    x = GlobalMaxPooling2D()(x)
    merge_one = concatenate([x, angle_layer])
    merge_one = Dense(512, activation='relu', name='fc2')(merge_one)
    merge_one = Dropout(0.3)(merge_one)
    merge_one = Dense(512, activation='relu', name='fc3')(merge_one)
    merge_one = Dropout(0.3)(merge_one)

    predictions = Dense(1, activation='sigmoid')(merge_one)

    model = Model(input=[base_model.input, input_2], output=predictions)

    sgd = SGD(lr=1e-3, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])
    return model
```

Kfold CV 정의

```
#Using K-fold Cross Validation with Data Augmentation.
def myAngleCV(X_train, X_angle, X_test):
    K=3
    folds = list(StratifiedKFold(n_splits=K, shuffle=True, random_state=16).split(X
    y_test_pred_log = 0
    y_train_pred_log=0
    y_valid_pred_log = 0.0*target_train
    for j, (train_idx, test_idx) in enumerate(folds):
        print('\n===================FOLD=',j)
        X_train_cv = X_train[train_idx]
        y_train_cv = target_train[train_idx]
        X_holdout = X_train[test_idx]
        Y_holdout= target_train[test_idx]

        #Angle
        X_angle_cv=X_angle[train_idx]
        X_angle_hold=X_angle[test_idx]

        #define file path and get callbacks
        file_path = "%s_aug_model_weights.hdf5"%j
        callbacks = get_callbacks(filepath=file_path, patience=5)
        gen_flow = gen_flow_for_two_inputs(X_train_cv, X_angle_cv, y_train_cv)
        galaxyModel= getVggAngleModel()
        galaxyModel.fit_generator(gen_flow,
                                  steps_per_epoch=24,
                                  epochs=100,
                                  shuffle=True,
                                  verbose=1,
                                  validation_data=([X_holdout,X_angle_hold], Y_hol
                                  callbacks=callbacks)
        #Getting the Best Model
        galaxyModel.load_weights(filepath=file_path)
        #Getting Training Score
        score = galaxyModel.evaluate([X_train_cv,X_angle_cv], y_train_cv, verbose=0)
        print('Train loss:', score[0])
        print('Train accuracy:', score[1])
        #Getting Test Score
        score = galaxyModel.evaluate([X_holdout,X_angle_hold], Y_holdout, verbose=0)
        print('Test loss:', score[0])
        print('Test accuracy:', score[1])
```

Kfold CV 정의

```
                                  callbacks=callbacks)
#Getting the Best Model
galaxyModel.load_weights(filepath=file_path)
#Getting Training Score
score = galaxyModel.evaluate([X_train_cv,X_angle_cv], y_train_cv, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])
#Getting Test Score
score = galaxyModel.evaluate([X_holdout,X_angle_hold], Y_holdout, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

#Getting validation Score.
pred_valid=galaxyModel.predict([X_holdout,X_angle_hold])
y_valid_pred_log[test_idx] = pred_valid.reshape(pred_valid.shape[0])

#Getting Test Scores
temp_test=galaxyModel.predict([X_test, X_test_angle])
y_test_pred_log+=temp_test.reshape(temp_test.shape[0])

#Getting Train Scores
temp_train=galaxyModel.predict([X_train, X_angle])
y_train_pred_log+=temp_train.reshape(temp_train.shape[0])

y_test_pred_log=y_test_pred_log/K
y_train_pred_log=y_train_pred_log/K

print('₩n Train Log Loss Validation= ',log_loss(target_train, y_train_pred_log))
print(' Test Log Loss Validation= ',log_loss(target_train, y_valid_pred_log))
return y_test_pred_log
```

오류 : 이유 몰라 ...

```
#원래 다르게 오류가 나야하는데 ㅠㅠ
preds=myAngleCV(X_train, X_angle, X_test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-44-91c7d543c17f> in <module>
----> 1 preds=myAngleCV(X_train, X_angle, X_test)

NameError: name 'X_test' is not defined
```

```
Exception: URL fetch failure on https://github.com/fchollet/deep-learning-mode
ls/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5: N
one -- [Errno -2] Name or service not known
```

```
#Submission for each day.
submission = pd.DataFrame()
submission['id']=test['id']
submission['is_iceberg']=preds
submission.to_csv('sub.csv', index=False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-2e45f3cb6d9c> in <module>()
      2 submission = pd.DataFrame()
      3 submission['id']=test['id']
----> 4 submission['is_iceberg']=preds
      5 submission.to_csv('sub.csv', index=False)

NameError: name 'preds' is not defined
```

26.Keras+TF LB 0.18

무작위로 초기화

In [2]:
```
import numpy as np
np.random.seed(98643)
import tensorflow as tf
tf.random.set_seed(683)
```

이미지 지우기 종속성

In [3]:
```
from skimage.restoration import (denoise_tv_chambolle, denoise_bilateral,
                                 denoise_wavelet, estimate_sigma,
                                 denoise_tv_bregman, denoise_nl_means)
from skimage.filters import gaussian
from skimage.color import rgb2gray
```

데이터 읽기 및 시각화

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import MinMaxScaler
```

Training part

]:
```
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Input, Flatten, Glob
from keras.layers import GlobalMaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.merge import Concatenate
from keras.models import Model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from keras.preprocessing.image import ImageDataGenerator
```

데이터 이미지 형식으로 변환

```
def color_composite(data):
    rgb_arrays = []
    for i, row in data.iterrows():
        band_1 = np.array(row['band_1']).reshape(75, 75)
        band_2 = np.array(row['band_2']).reshape(75, 75)
        band_3 = band_1 / band_2

        r = (band_1 + abs(band_1.min())) / np.max((band_1 + abs(band_1.min())))
        g = (band_2 + abs(band_2.min())) / np.max((band_2 + abs(band_2.min())))
        b = (band_3 + abs(band_3.min())) / np.max((band_3 + abs(band_3.min())))

        rgb = np.dstack((r, g, b))
        rgb_arrays.append(rgb)
    return np.array(rgb_arrays)

def denoise(X, weight, multichannel):
    return np.asarray([denoise_tv_chambolle(item, weight=weight, multichannel=multi

def smooth(X, sigma):
    return np.asarray([gaussian(item, sigma=sigma) for item in X])

def grayscale(X):
    return np.asarray([rgb2gray(item) for item in X])
```

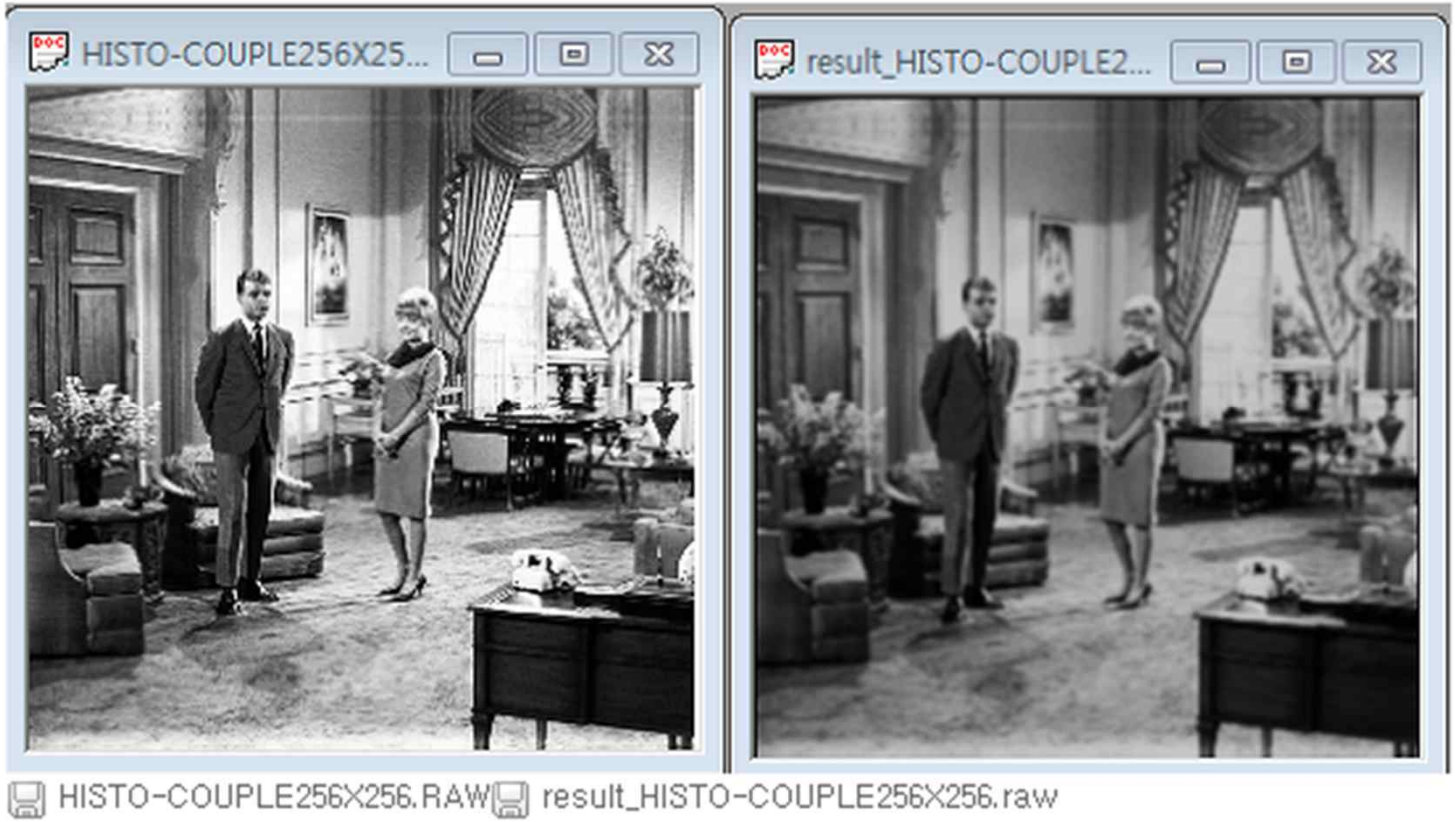
새로운 축으로 r,g,b 붙이기

이미지에서 잡음 제거

rbg 이미지를 읽고 그레이 스케일 이미지로 변환

가우시안 분포
- 정규분포 공식에서 평균값을 0으로 하여 유도한 분포

가우시안 스무딩 필터링
- 가우시안 분포를 영상처리에 적용
- - 생성된 잡음을 제거하기 위한 필터



데이터, 적절한 매개변수 선택

In [7]:
```
train = pd.read_json("C:/Users/HOME/Desktop/수DA쟁이/배, 빙하 분류하기/Data/train.js
train.inc_angle = train.inc_angle.replace('na', 0)
train.inc_angle = train.inc_angle.astype(float).fillna(0.0)
train_all = True
```

In [8]:
```
train_b = True or train_all
train_img = True or train_all
train_total = True or train_all
predict_submission = True and train_all

clean_all = False
clean_b = False or clean_all
clean_img = False or clean_all

load_all = False
load_b = False or load_all
load_img = False or load_all
```

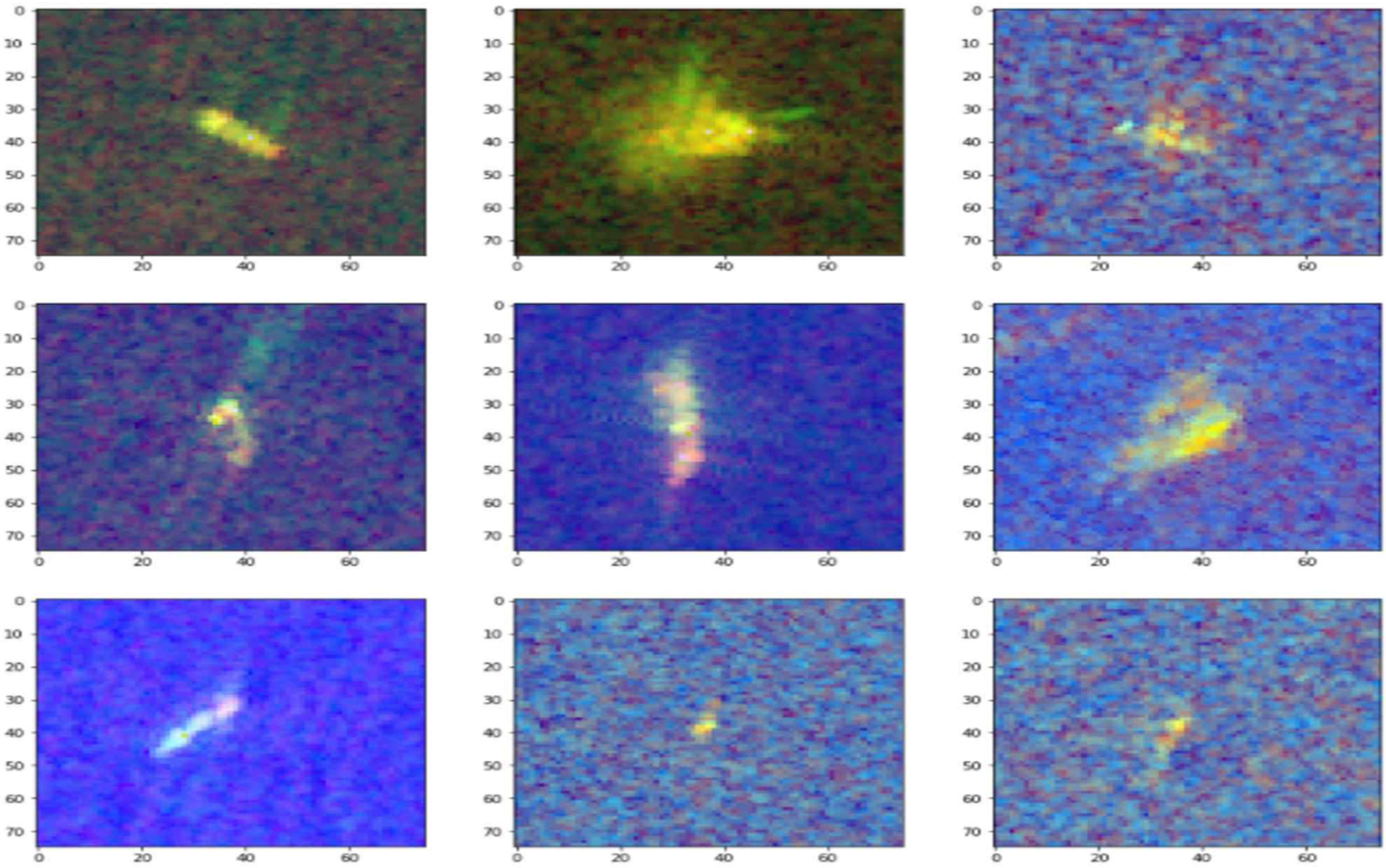
In [9]:
```
def create_dataset(frame, labeled, smooth_rgb=0.2, smooth_gray=0.5,
                   weight_rgb=0.05, weight_gray=0.05):
    band_1, band_2, images = frame['band_1'].values, frame['band_2'].values, color_c
    to_arr = lambda x: np.asarray([np.asarray(item) for item in x])
    band_1 = to_arr(band_1)
    band_2 = to_arr(band_2)
    band_3 = (band_1 + band_2) / 2
    gray_reshape = lambda x: np.asarray([item.reshape(75, 75) for item in x])
    # 평면 벡터에서 그림 형식 만들기
    band_1 = gray_reshape(band_1)
    band_2 = gray_reshape(band_2)
    band_3 = gray_reshape(band_3)
    print('Denoising and reshaping')
    if train_b and clean_b:
        # Smooth and denoise data
        band_1 = smooth(denoise(band_1, weight_gray, False), smooth_gray)
        print('Gray 1 done')
        band_2 = smooth(denoise(band_2, weight_gray, False), smooth_gray)
        print('Gray 2 done')
        band_3 = smooth(denoise(band_3, weight_gray, False), smooth_gray)
        print('Gray 3 done')
    if train_img and clean_img:
        images = smooth(denoise(images, weight_rgb, True), smooth_rgb)
    print('RGB done')
    tf_reshape = lambda x: np.asarray([item.reshape(75, 75, 1) for item in x])
    band_1 = tf_reshape(band_1)
    band_2 = tf_reshape(band_2)
    band_3 = tf_reshape(band_3)
    #images = tf_reshape(images)
    band = np.concatenate([band_1, band_2, band_3], axis=3)
    if labeled:
        y = np.array(frame["is_iceberg"])
    else:
        y = None
    return y, band, images
```

In [10]:
```
y_train, X_b, X_images = create_dataset(train, True)
```

```
Denoising and reshaping
RGB done
```

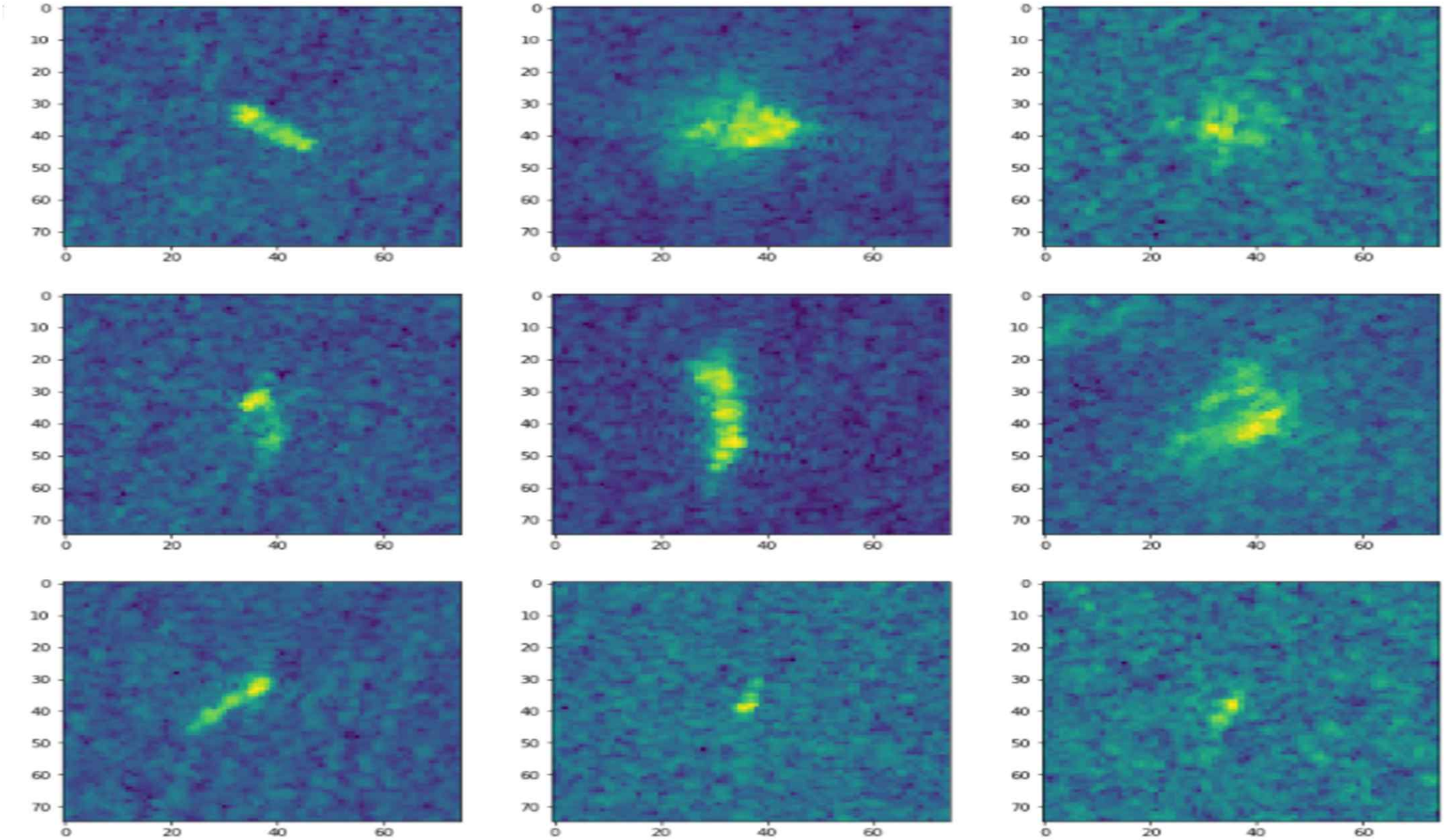
In [11]:

```
fig = plt.figure(200, figsize=(15, 15))
random_indicies = np.random.choice(range(len(X_images)), 9, False)
subset = X_images[random_indicies]
for i in range(9):
    ax = fig.add_subplot(3, 3, i + 1)
    ax.imshow(subset[i])
plt.show()
```

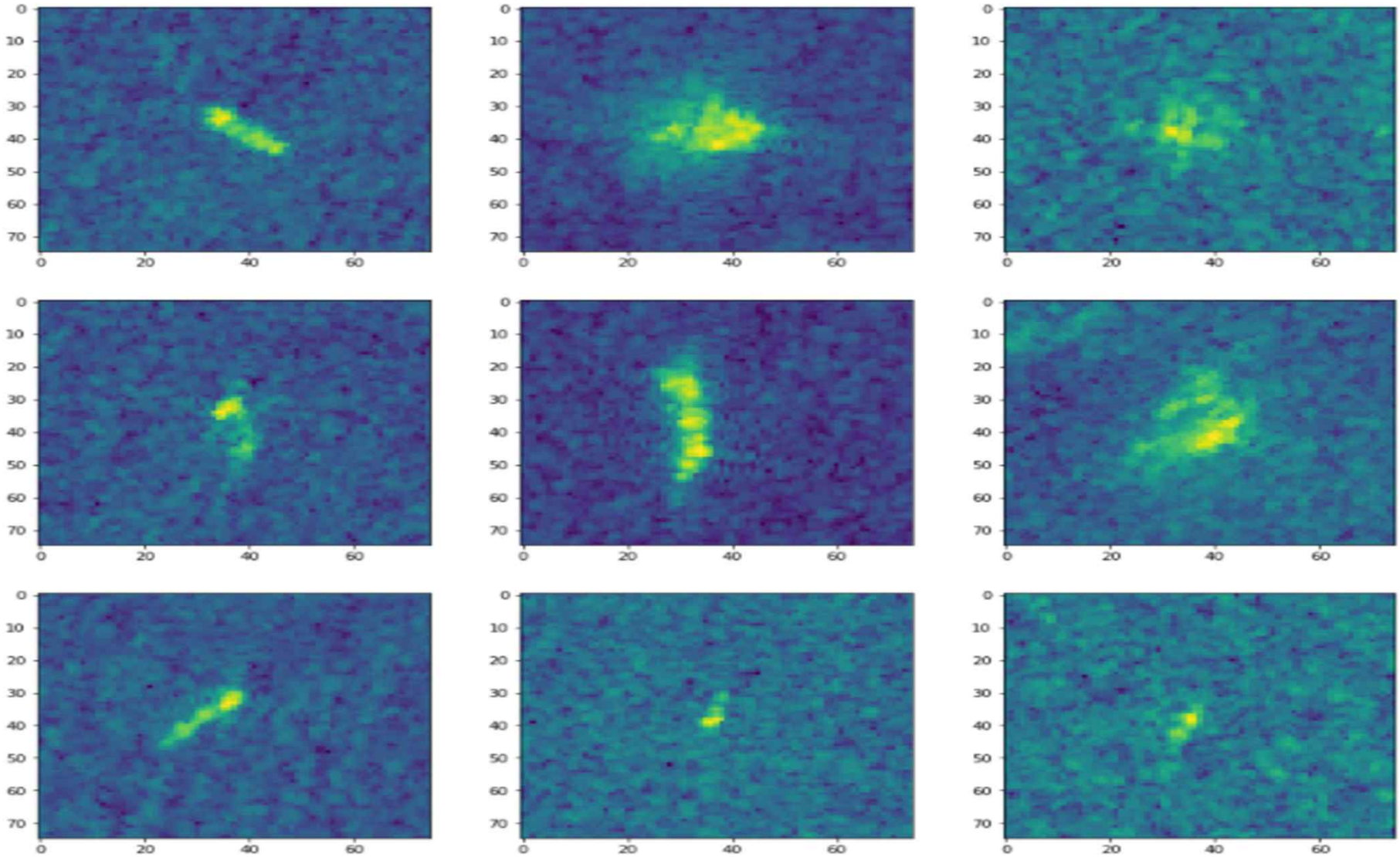


In [12]:

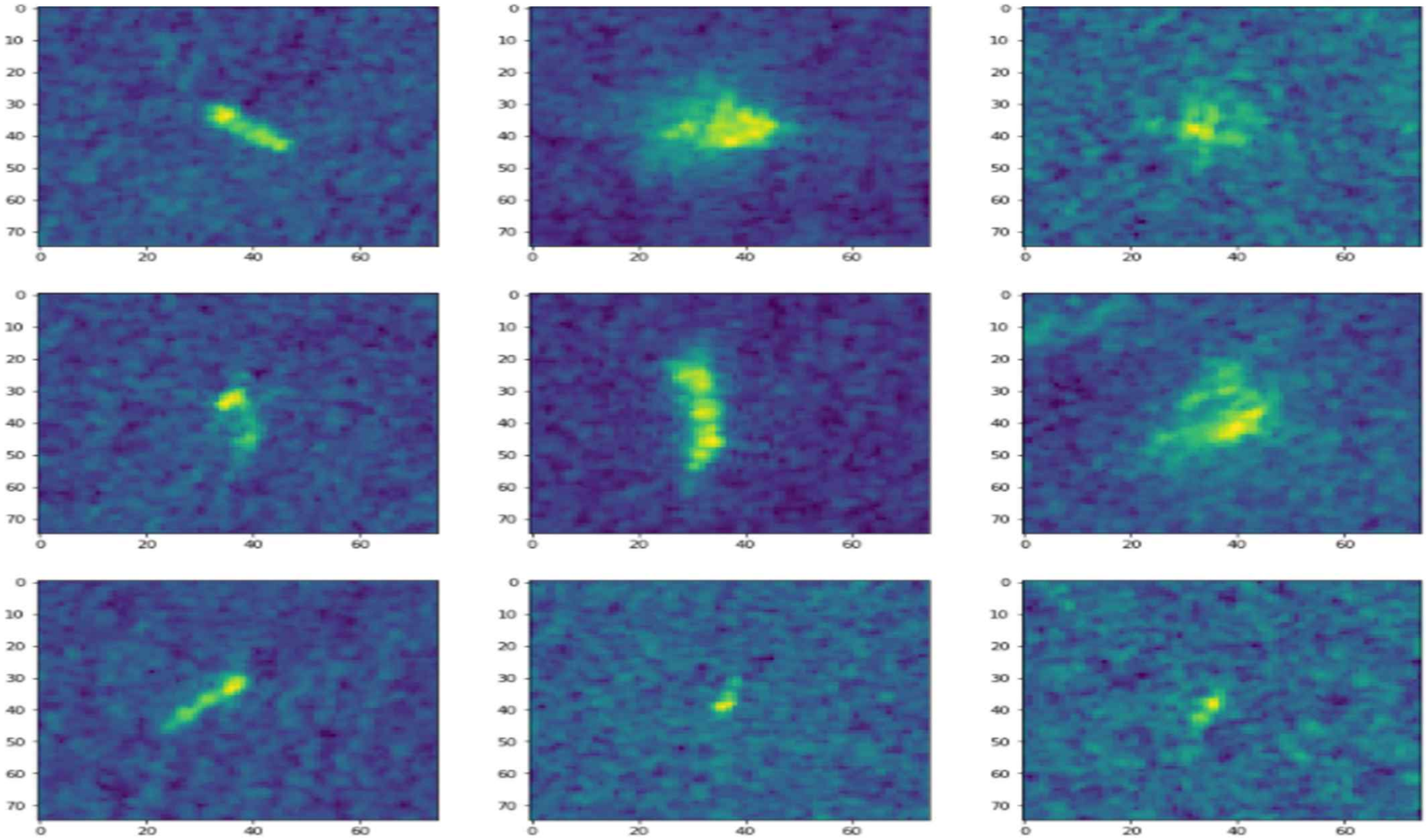
```
fig = plt.figure(202, figsize=(15, 15))
band_1_x = train['band_1'].values
subset = np.asarray(band_1_x)[random_indicies]
subset = np.asarray([np.asarray(item).reshape(75, 75) for item in subset])
for i in range(9):
    ax = fig.add_subplot(3, 3, i + 1)
    ax.imshow(subset[i])
plt.show()
```



In [13]:
```
fig = plt.figure(202, figsize=(15, 15))
subset = np.asarray(band_1_x)[random_indicies]
subset = denoise(np.asarray([np.asarray(item).reshape(75, 75) for item in subset]),
for i in range(9):
    ax = fig.add_subplot(3, 3, i + 1)
    ax.imshow(subset[i])
plt.show()
```



In [14]:
```
fig = plt.figure(202, figsize=(15, 15))
subset = np.asarray(band_1_x)[random_indicies]
subset = smooth(denoise(np.asarray(
    [np.asarray(item).reshape(75, 75) for item in subset]), 0.05, False), 0.5)
for i in range(9):
    ax = fig.add_subplot(3, 3, i + 1)
    ax.imshow(subset[i])
plt.show()
```



In [15]:

```
def get_model_notebook(lr, decay, channels, relu_type='relu'):
    # angle variable defines if we should use angle parameter or ignore it
    input_1 = Input(shape=(75, 75, channels))

    fcnn = Conv2D(32, kernel_size=(3, 3), activation=relu_type)(
        BatchNormalization()(input_1))
    fcnn = MaxPooling2D((3, 3))(fcnn)
    fcnn = Dropout(0.2)(fcnn)
    fcnn = Conv2D(64, kernel_size=(3, 3), activation=relu_type)(fcnn)
    fcnn = MaxPooling2D((2, 2), strides=(2, 2))(fcnn)
    fcnn = Dropout(0.2)(fcnn)
    fcnn = Conv2D(128, kernel_size=(3, 3), activation=relu_type)(fcnn)
    fcnn = MaxPooling2D((2, 2), strides=(2, 2))(fcnn)
    fcnn = Dropout(0.2)(fcnn)
    fcnn = Conv2D(128, kernel_size=(3, 3), activation=relu_type)(fcnn)
    fcnn = MaxPooling2D((2, 2), strides=(2, 2))(fcnn)
    fcnn = Dropout(0.2)(fcnn)
    fcnn = BatchNormalization()(fcnn)
    fcnn = Flatten()(fcnn)
    local_input = input_1
    partial_model = Model(input_1, fcnn)
    dense = Dropout(0.2)(fcnn)
    dense = Dense(256, activation=relu_type)(dense)
    dense = Dropout(0.2)(dense)
    dense = Dense(128, activation=relu_type)(dense)
    dense = Dropout(0.2)(dense)
    dense = Dense(64, activation=relu_type)(dense)
    dense = Dropout(0.2)(dense)
    # For some reason i've decided not to normalize angle data
    output = Dense(1, activation="sigmoid")(dense)
    model = Model(local_input, output)
    optimizer = Adam(lr=lr, decay=decay)
    model.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accurac
    return model, partial_model
```

합성곱층과
풀링층
여러겹 쌓임

완전연결계층

In [20]:
```
def combined_model(m_b, m_img, lr, decay):
    input_b = Input(shape=(75, 75, 3))
    input_img = Input(shape=(75, 75, 3))

    m1 = m_b(input_b)
    m2 = m_img(input_img)

    #모델을 결합하고이를 기반으로 퍼셉트론을 훈련시킵니다.
    # 반복되는 아이디어는이 작업에 XGB를 사용하는 것이지만
    #실제로는이 방법이 싫습니다
    common = Concatenate()([m1, m2])
    common = BatchNormalization()(common)
    common = Dropout(0.3)(common)
    common = Dense(1024, activation='relu')(common)
    common = Dropout(0.3)(common)
    common = Dense(512, activation='relu')(common)
    common = Dropout(0.3)(common)
    output = Dense(1, activation="sigmoid")(common)
    model = Model([input_b, input_img], output)
    optimizer = Adam(lr=lr, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=decay)
    model.compile(loss="binary_crossentropy", optimizer=optimizer,
                  metrics=["accuracy"])
    return model
```

In [21]:
```
def gen_flow_multi_inputs(I1, I2, y, batch_size):
    gen1 = ImageDataGenerator(horizontal_flip=True,
                              vertical_flip=True,
                              width_shift_range=0.,
                              height_shift_range=0.,
                              channel_shift_range=0,
                              zoom_range=0.2,
                              rotation_range=10)
    gen2 = ImageDataGenerator(horizontal_flip=True,
                              vertical_flip=True,
                              width_shift_range=0.,
                              height_shift_range=0.,
                              channel_shift_range=0,
                              zoom_range=0.2,
                              rotation_range=10)
    genI1 = gen1.flow(I1, y, batch_size=batch_size, seed=57, shuffle=False)
    genI2 = gen2.flow(I1, I2, batch_size=batch_size, seed=57, shuffle=False)
    while True:
        I1i = genI1.next()
        I2i = genI2.next()
        #print I1i[0].shape
        np.testing.assert_array_equal(I2i[0], I1i[0])
        yield [I1i[0], I2i[1]], I1i[1]
```

In [22]:
```
def train_model(model, batch_size, epochs, checkpoint_name, X_train, y_train,
                val_data, verbose=2):
    callbacks = [ModelCheckpoint(checkpoint_name, save_best_only=True,
                                 monitor='val_loss')]
    datagen = ImageDataGenerator(horizontal_flip=True,
                                 vertical_flip=True,
                                 width_shift_range=0.,
                                 height_shift_range=0.,
                                 channel_shift_range=0,
                                 zoom_range=0.2,
                                 rotation_range=10)
    x_test, y_test = val_data
    try:
        model.fit_generator(datagen.flow(X_train, y_train, batch_size=batch_size),
                            epochs=epochs,
                                 steps_per_epoch=len(X_train) / batch_size,
                                 validation_data=(x_test, y_test), verbose=1,
                                 callbacks=callbacks)
    except KeyboardInterrupt:
        if verbose > 0:
            print('Interrupted')
    if verbose > 0:
        print('Loading model')
    model.load_weights(filepath=checkpoint_name)
    return model
```

In [23]:
```
#특정 모델 훈련
def gen_model_weights(lr, decay, channels, relu, batch_size, epochs,
                      path_name, data, verbose=2):
    X_train, y_train, X_test, y_test, X_val, y_val = data
    model, partial_model = get_model_notebook(lr, decay, channels, relu)
    model = train_model(model, batch_size, epochs, path_name,
                        X_train, y_train, (X_test, y_test), verbose=verbose)

    if verbose > 0:
        loss_val, acc_val = model.evaluate(X_val, y_val,
                                verbose=0, batch_size=batch_size)

        loss_train, acc_train = model.evaluate(X_test, y_test,
                                    verbose=0, batch_size=batch_size)

        print('Val/Train Loss:', str(loss_val) + '/' + str(loss_train), #
            'Val/Train Acc:', str(acc_val) + '/' + str(acc_train))
    return model, partial_model
```

```python
# Train all 3 models
def train_models(dataset, lr, batch_size, max_epoch, verbose=2, return_model=False):
    y_train, X_b, X_images = dataset
    y_train_full, y_val,#
    X_b_full, X_b_val,#
    X_images_full, X_images_val = train_test_split(y_train, X_b, X_images, random_state=687, train_size=0.9)

    y_train, y_test, #
    X_b_train, X_b_test, #
    X_images_train, X_images_test = train_test_split(y_train_full, X_b_full, X_images_full, random_state=576, train_size=0.85

    if train_b:
        if verbose > 0:
            print('Training bandwidth network')
        data_b1 = (X_b_train, y_train, X_b_test, y_test, X_b_val, y_val)
        model_b, model_b_cut = gen_model_weights(lr, 1e-6, 3, 'relu', batch_size, max_epoch, 'model_b',
                                                 data=data_b1, verbose=verbose)

    if train_img:
        if verbose > 0:
            print('Training image network')
        data_images = (X_images_train, y_train, X_images_test, y_test, X_images_val, y_val)
        model_images, model_images_cut = gen_model_weights(lr, 1e-6, 3, 'relu', batch_size, max_epoch, 'model_img',
                                                           data_images, verbose=verbose)

    if train_total:
        common_model = combined_model(model_b_cut, model_images_cut, lr/2, 1e-7)
        common_x_train = [X_b_full, X_images_full]
        common_y_train = y_train_full
        common_x_val = [X_b_val, X_images_val]
        common_y_val = y_val
        if verbose > 0:
            print('Training common network')
        callbacks = [ModelCheckpoint('common', save_best_only=True, monitor='val_loss')]
        try:
            common_model.fit_generator(gen_flow_multi_inputs(X_b_full, X_images_full, y_train_full, batch_size),
                                       epochs=30,
                                steps_per_epoch=len(X_b_full) / batch_size,
                                validation_data=(common_x_val, common_y_val), verbose=1,
                                callbacks=callbacks)
        except KeyboardInterrupt:
            pass
        common_model.load_weights(filepath='common')
        loss_val, acc_val = common_model.evaluate(common_x_val, common_y_val,
                                          verbose=0, batch_size=batch_size)
        loss_train, acc_train = common_model.evaluate(common_x_train, common_y_train,
                                                      verbose=0, batch_size=batch_size)
        if verbose > 0:
            print('Loss:', loss_val, 'Acc:', acc_val)
    if return_model:
        return common_model
    else:
        return (loss_train, acc_train), (loss_val, acc_val)
```

결과

```
# Best parameters i got are
# epochs : 250
# learning rate : 8e-5
# batch size : 32
common_model = train_models((y_train, X_b, X_images), 7e-04, 32, 50, 1, return_mo
del=True)
```

```
l_loss: 0.1734 - val_acc: 0.9193
Epoch 27/30
46/45 [==============================] - 28s - loss: 0.1745 - acc: 0.9280 - va
l_loss: 0.1714 - val_acc: 0.9006
Epoch 28/30
46/45 [==============================] - 28s - loss: 0.1858 - acc: 0.9137 - va
l_loss: 0.1789 - val_acc: 0.9006
Epoch 29/30
46/45 [==============================] - 28s - loss: 0.1576 - acc: 0.9436 - va
l_loss: 0.1544 - val_acc: 0.9130
Epoch 30/30
46/45 [==============================] - 27s - loss: 0.1517 - acc: 0.9327 - va
l_loss: 0.1865 - val_acc: 0.9006
Loss: 0.154399480139 Acc: 0.913043478261
```

끝~!

- https://iskim3068.tistory.com/41
- https://jayharvey.tistory.com/11
- https://excelsior-cjh.tistory.com/180
- https://m.blog.naver.com/PostView.nhn?blogId=flowerdances&logNo=221189533377&proxyReferer=https:%2F%2Fwww.google.com%2F
- https://github.com/minsuk-heo