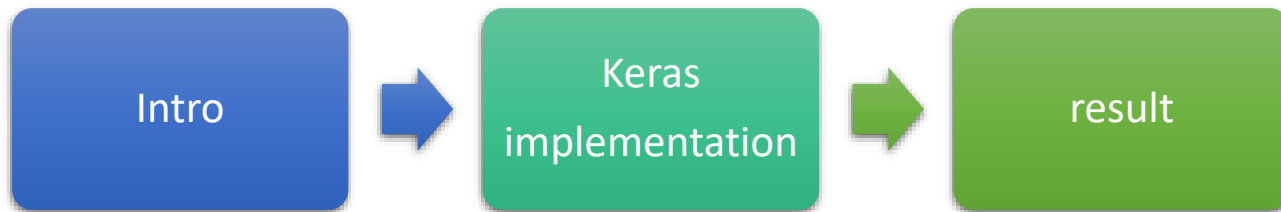


# Deep learning 3

전현호

# Contents



# Intro

## ■ 일정

1. Tensorflow 설치 과정 정리
2. Linear regression 해결을 위한 코드 구성
3. mnist 해결을 위한 NN, CNN 구성 (99.3%)

} 완료

4. TF-learn을 통한 2종류 이미지 분류, mnist
5. Keras를 통한 mnist

} 발표

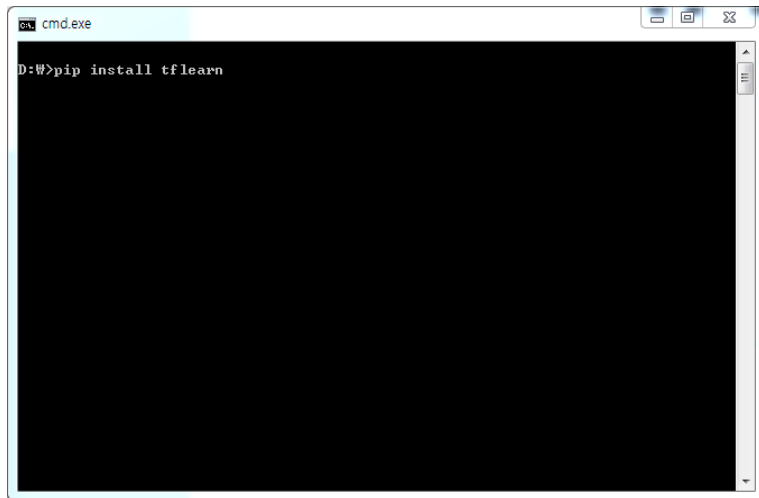
6. Keras기반 imagenet 학습 및 결과 확인
7. 논문 스터디를 통한 Detection 가능 네트워크 구조 생성 -> 동영상에 적용
8. GAN or 강화학습 or 자율 주행 시뮬레이션

} 예정



# TF-learn

- 설치 : pip install tflearn
- 사용 : import tflearn



```
cmd.exe
D:\>pip install tflearn
```

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
```

# Kaggle data

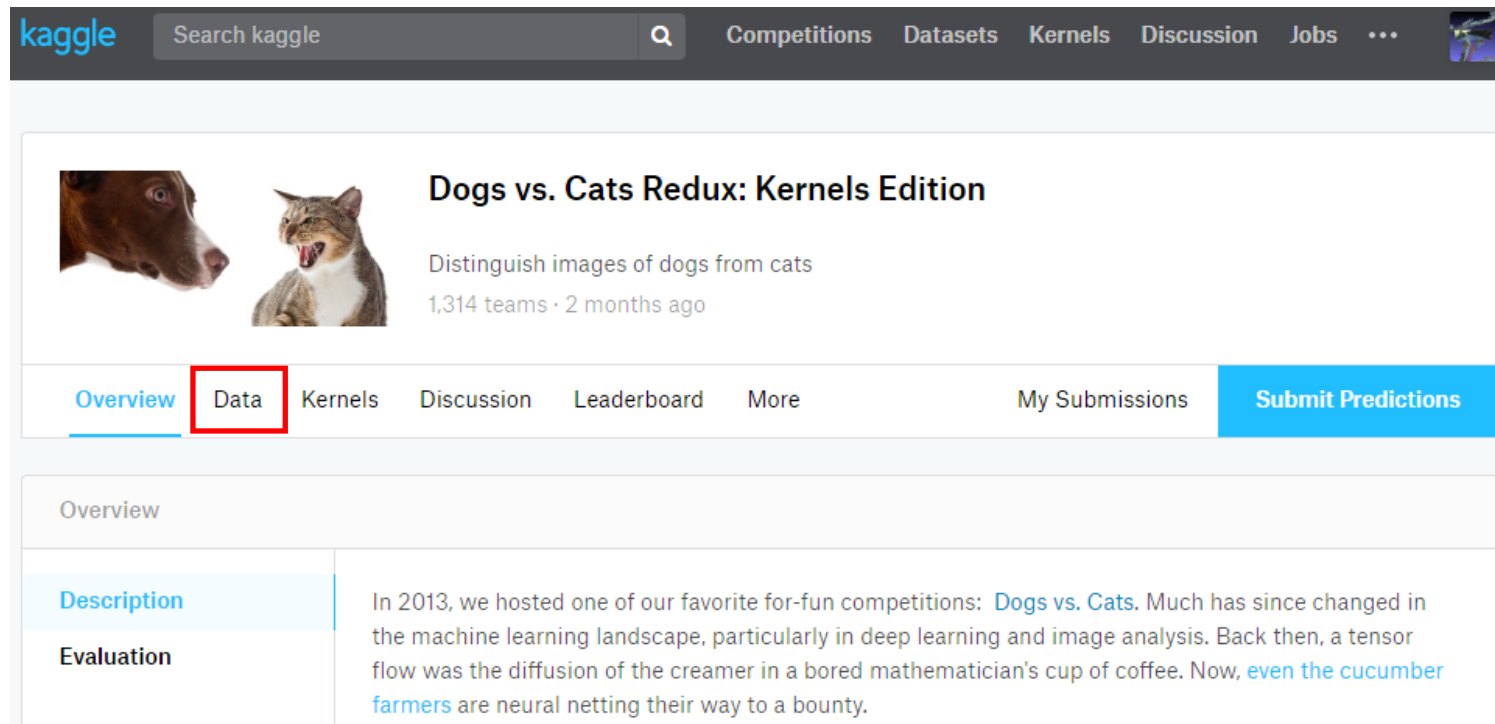
- www.kaggle.com

The screenshot shows the Kaggle website interface. The top navigation bar includes 'kaggle', a search bar, and links for 'Competitions', 'Datasets', 'Kernels', 'Discussion', and 'Jobs'. The 'Datasets' link is highlighted with a red box. The main content area features a heatmap titled 'Speed Camera Violations by Year and Month(2014-2016)'. The heatmap shows violation counts for each month from January to December across the years 2014, 2015, and 2016. A color scale on the right indicates the total number of violations, ranging from 70,000 (dark purple) to 120,000 (yellow). Below the heatmap, there are three user activity cards: one by VivekMangipudi, one by HubertLin, and one by Grzegorz Sionkowski. On the right side, there are sections for 'Hyunho Jeon' profile, 'Novice' checklist, 'My Competitions', 'Recommended Datasets', and 'Recommended Kernels'.

Month	2014	2015	2016
Dec	102088	80617	76056
Nov	100381	85119	91282
Oct	105822	111201	92592
Sep	112615	107583	90704
Aug	87668	87650	81566
Jul	122282	103360	92850
Jun		106043	92884
May		119459	104091
Apr		102961	93495
Mar		107024	96922
Feb		67021	81597
Jan		79422	83096

# Dogs vs. Cats

- [www.kaggle.com](http://www.kaggle.com)



The screenshot shows the Kaggle website interface. At the top, there is a navigation bar with the Kaggle logo, a search bar, and links for Competitions, Datasets, Kernels, Discussion, and Jobs. Below the navigation bar, the main content area features a competition card for 'Dogs vs. Cats Redux: Kernels Edition'. The card includes images of a dog and a cat, the competition title, a description 'Distinguish images of dogs from cats', and statistics '1,314 teams · 2 months ago'. Below the card, there is a horizontal menu with tabs for Overview, Data (highlighted with a red box), Kernels, Discussion, Leaderboard, and More. To the right of this menu are links for 'My Submissions' and a blue 'Submit Predictions' button. Below the menu, the 'Overview' section is visible, with a 'Description' tab selected. The description text reads: 'In 2013, we hosted one of our favorite for-fun competitions: [Dogs vs. Cats](#). Much has since changed in the machine learning landscape, particularly in deep learning and image analysis. Back then, a tensor flow was the diffusion of the creamer in a bored mathematician's cup of coffee. Now, [even the cucumber farmers](#) are neural netting their way to a bounty.'

# Dogs vs. Cats

- www.kaggle.com

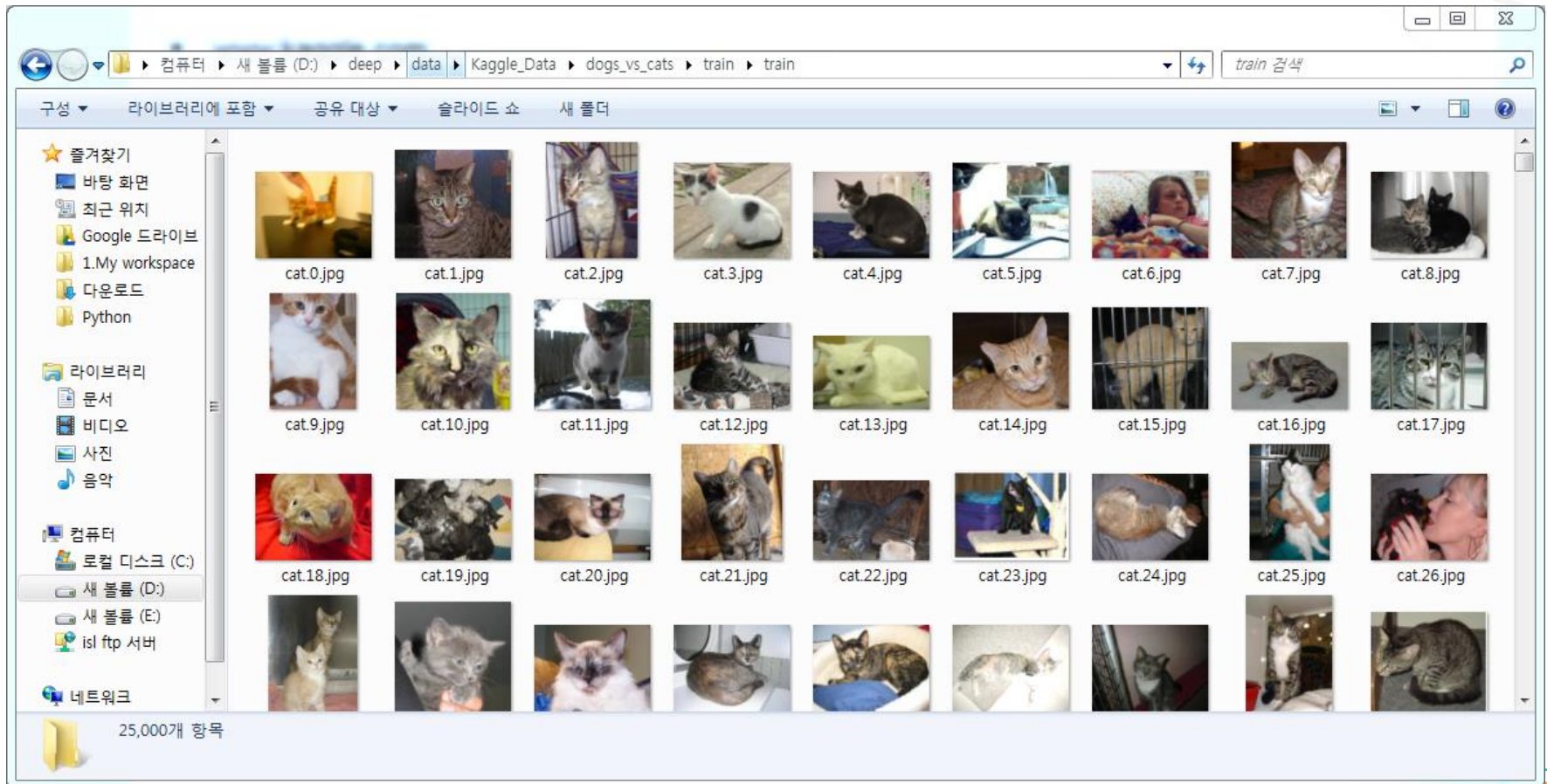
The screenshot shows the Kaggle website interface for the 'Dogs vs. Cats Redux: Kernels Edition' competition. The navigation bar at the top includes the Kaggle logo, a search bar, and links for Competitions, Datasets, Kernels, Discussion, Jobs, and a menu icon. The competition header features a dog and a cat image, the title 'Dogs vs. Cats Redux: Kernels Edition', the description 'Distinguish images of dogs from cats', and the text '1,314 teams · 2 months ago'. Below the header is a navigation menu with tabs for Overview, Data (selected), Kernels, Discussion, Leaderboard, More, My Submissions, and a prominent blue 'Submit Predictions' button. The 'Training Data' section is highlighted with a red border and contains a table of files.

Training Data	
<b>3 files</b>	<b>test.zip</b> <a href="#">Download File</a>
sample_submission.cs...	File size 271.3 MB
test.zip	
train.zip	



# Dogs vs. Cats

- 각각 12,500장의 학습 데이터 (총 25,000장)
- 12,500장의 테스트 데이터



# Dogs vs. Cats

```
In [1]: import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
```

```
In [2]: TRAIN_DIR = 'D:/deep/data/Kaggle_Data/dogs_vs_cats/train/train'
TEST_DIR = 'D:/deep/data/Kaggle_Data/dogs_vs_cats/test/test'
IMG_SIZE = 50
LR = 1e-3

MODEL_NAME = 'dogsvscats-{}-{}.model'.format(LR, '5conv')
```

```
In [3]: def label_img(img):
word_label = img.split('.')[ -3]
if word_label == 'cat': return [1,0]
elif word_label == 'dog': return [0,1]
```

```
In [4]: def create_train_data():
training_data = []
for img in tqdm(os.listdir(TRAIN_DIR)):
label = label_img(img)
path = os.path.join(TRAIN_DIR, img)
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
training_data.append([np.array(img), np.array(label)])
shuffle(training_data)
np.save('train_data.npy', training_data)
return training_data
```

# Dogs vs. Cats

```
In [5]: def process_test_data():
testing_data = []
for img in tqdm(os.listdir(TEST_DIR)):
    path = os.path.join(TEST_DIR, img)
    img_num = img.split('.')[0]
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    testing_data.append([np.array(img), img_num])

shuffle(testing_data)
np.save('test_data.npy', testing_data)
return testing_data
```

```
In [6]: #train_data = create_train_data()
# If you have already created the dataset:
train_data = np.load('train_data.npy')
```

# Dogs vs. Cats

```
In [7]: import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

# Dogs vs. Cats

- 비교

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

TF-learn  
5conv net

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot = True)
```

```
n_nodes_hl1 = 500
n_nodes_hl2 = 500
n_nodes_hl3 = 500
```

```
n_classes = 10
batch_size = 100
```

```
x = tf.placeholder('float', [None, 784])
y = tf.placeholder('float')
```

```
hidden_1_layer = {'weights':tf.Variable(tf.random_normal([784, n_nodes_hl1])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}
```

```
hidden_2_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_nodes_hl2])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl2]))}
```

```
hidden_3_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl2, n_nodes_hl3])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl3]))}
```

```
output_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl3, n_classes])),
                 'biases':tf.Variable(tf.random_normal([n_classes]))}
```

```
l1 = tf.add(tf.matmul(x,hidden_1_layer['weights']), hidden_1_layer['biases'])
l1 = tf.nn.relu(l1)
```

```
l2 = tf.add(tf.matmul(l1,hidden_2_layer['weights']), hidden_2_layer['biases'])
l2 = tf.nn.relu(l2)
```

```
l3 = tf.add(tf.matmul(l2,hidden_3_layer['weights']), hidden_3_layer['biases'])
l3 = tf.nn.relu(l3)
```

```
output = tf.matmul(l3,output_layer['weights']) + output_layer['biases']
```

```
prediction = output
cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(prediction,y) )
optimizer = tf.train.AdamOptimizer().minimize(cost)
```

```
hm_epochs = 10
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

Tensorflow  
3 hidden layer

+Session ~~~

# Dogs vs. Cats

```
In [8]: if os.path.exists('{} .meta'.format(MODEL_NAME)):  
        model.load(MODEL_NAME)  
        print('model loaded!')
```

model loaded!

```
In [9]: train = train_data[:-500]  
        test = train_data[-500:]
```

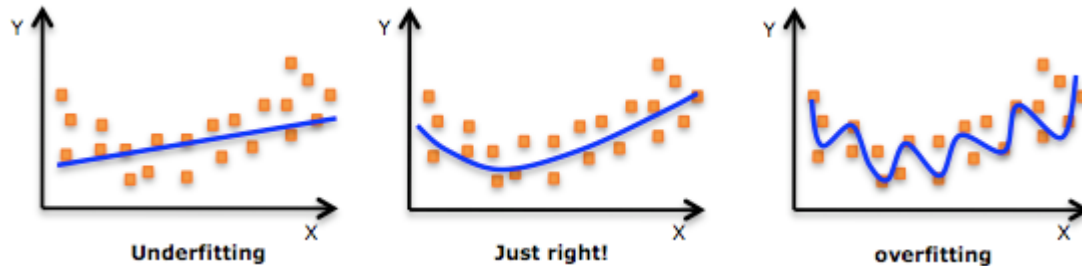
```
In [10]: print(train.shape)  
X = np.array([i[0] for i in train]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)  
Y = [i[1] for i in train]  
  
test_x = np.array([i[0] for i in test]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)  
test_y = [i[1] for i in test]
```

(24500, 2)

Validation set 생성

# Dogs vs. Cats

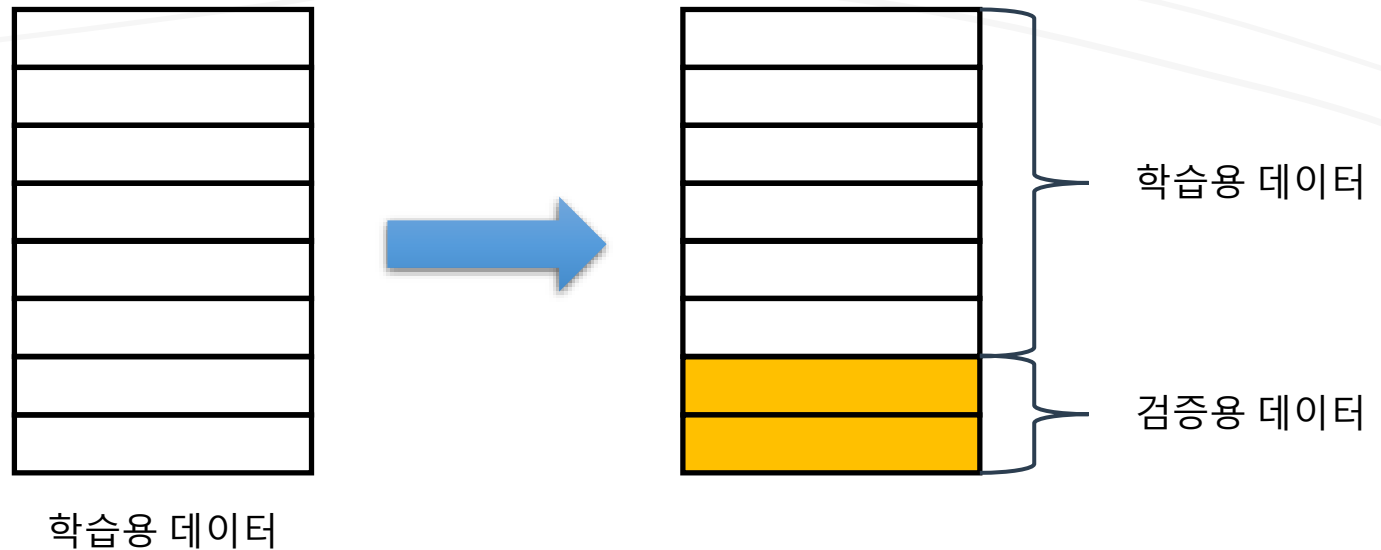
- Validation



- 간단한 문제에서는 과적합(overfitting)을 확인하기 쉬움
- 하지만 실제 해결하고자 하는 문제는 차원이 높아 확인하기 어려움
- 검증(Validation) 기법 고안 – 모든 데이터를 학습에 사용하지 않고 일부를 떼어 성능 검증에 사용함

# Dogs vs. Cats

- Validation



- 검증이 도입된 학습 절차

1. 학습 데이터를 학습용 데이터와 검증용 데이터로 나눔. 보통 8:2 비율 사용
2. 학습용 데이터로 모델을 학습
3. 검증용 데이터로 모델의 성능을 평가
  - a. 성능이 만족스러운 경우, 학습 종료
  - b. 성능이 떨어질 경우, 모델의 구조 등을 수정해 다시 학습



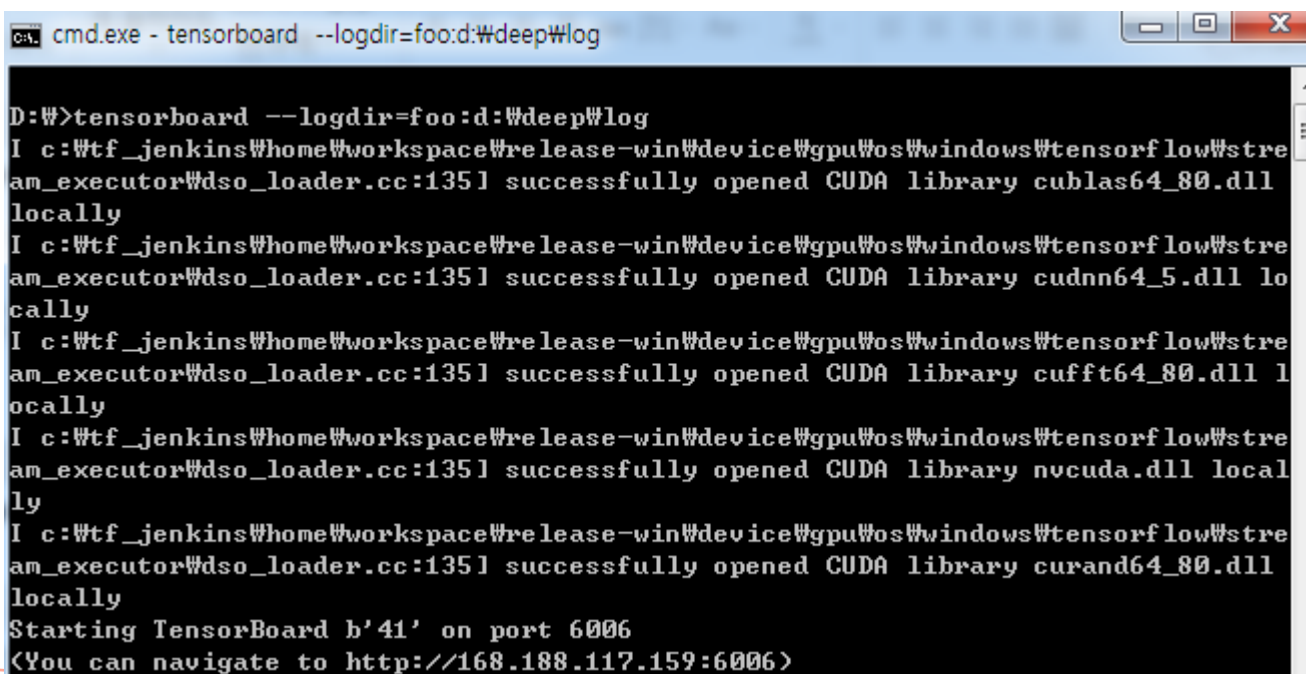
# Dogs vs. Cats

```
In [11]: model.fit({'input': X}, {'targets': Y}, n_epoch=30, validation_set=({'input': test_x}, {'targets': test_y}),  
            snapshot_step=500, show_metric=True, run_id=MODEL_NAME)  
#tensorboard --logdir=foo:d:\deep\log
```

학습

```
Training Step: 11489 | total loss: 0.07476 | time: 6.873s  
| Adam | epoch: 030 | loss: 0.07476 - acc: 0.9727 -- iter: 24448/24500  
Training Step: 11490 | total loss: 0.07033 | time: 7.907s  
| Adam | epoch: 030 | loss: 0.07033 - acc: 0.9738 | val_loss: 1.51778 - val_acc: 0.7540 -- iter: 24500/24500  
--
```

```
In [12]: model.save(MODEL_NAME)      모델 저장
```



```
cmd.exe - tensorboard --logdir=foo:d:\deep\log  
D:\>tensorboard --logdir=foo:d:\deep\log  
I c:\Wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\wds_loader.cc:135] successfully opened CUDA library cublas64_80.dll locally  
I c:\Wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\wds_loader.cc:135] successfully opened CUDA library cudnn64_5.dll locally  
I c:\Wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\wds_loader.cc:135] successfully opened CUDA library cufft64_80.dll locally  
I c:\Wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\wds_loader.cc:135] successfully opened CUDA library nvcuda.dll locally  
I c:\Wtf_jenkins\home\workspace\release-win\device\gpu\os\windows\tensorflow\stream_executor\wds_loader.cc:135] successfully opened CUDA library curand64_80.dll locally  
Starting TensorBoard b'41' on port 6006  
<You can navigate to http://168.188.117.159:6006>
```

Tensorboard 실행

# Dogs vs. Cats

## Tensorboard

- 학습 과정에서 curses가 없다는 경고 문구가 나올 경우 tensorboard에 데이터가 보이지 않는 문제가 있음
  - pip install curses로 해결

The screenshot shows the TensorBoard web interface. At the top, there's a navigation bar with tabs for SCALARS, IMAGES, AUDIO, GRAPHS, DISTRIBUTIONS, HISTOGRAMS, and EMBEDDINGS. Below this, there's a search bar with the text "Write a regex to create a tag group" and a close button. There are three checkboxes: "Split on underscores", "Data download links", and "Data download links". A dropdown menu for "Tooltip sorting method" is set to "default". Below that is a "Smoothing" slider set to 0.6. Under "Horizontal Axis", there are three buttons: "STEP" (selected), "RELATIVE", and "WALL". The "Runs" section has a filter bar with the text "Write a regex to filter runs". Below the filter, there are five runs listed, each with a colored circle and a checkmark:

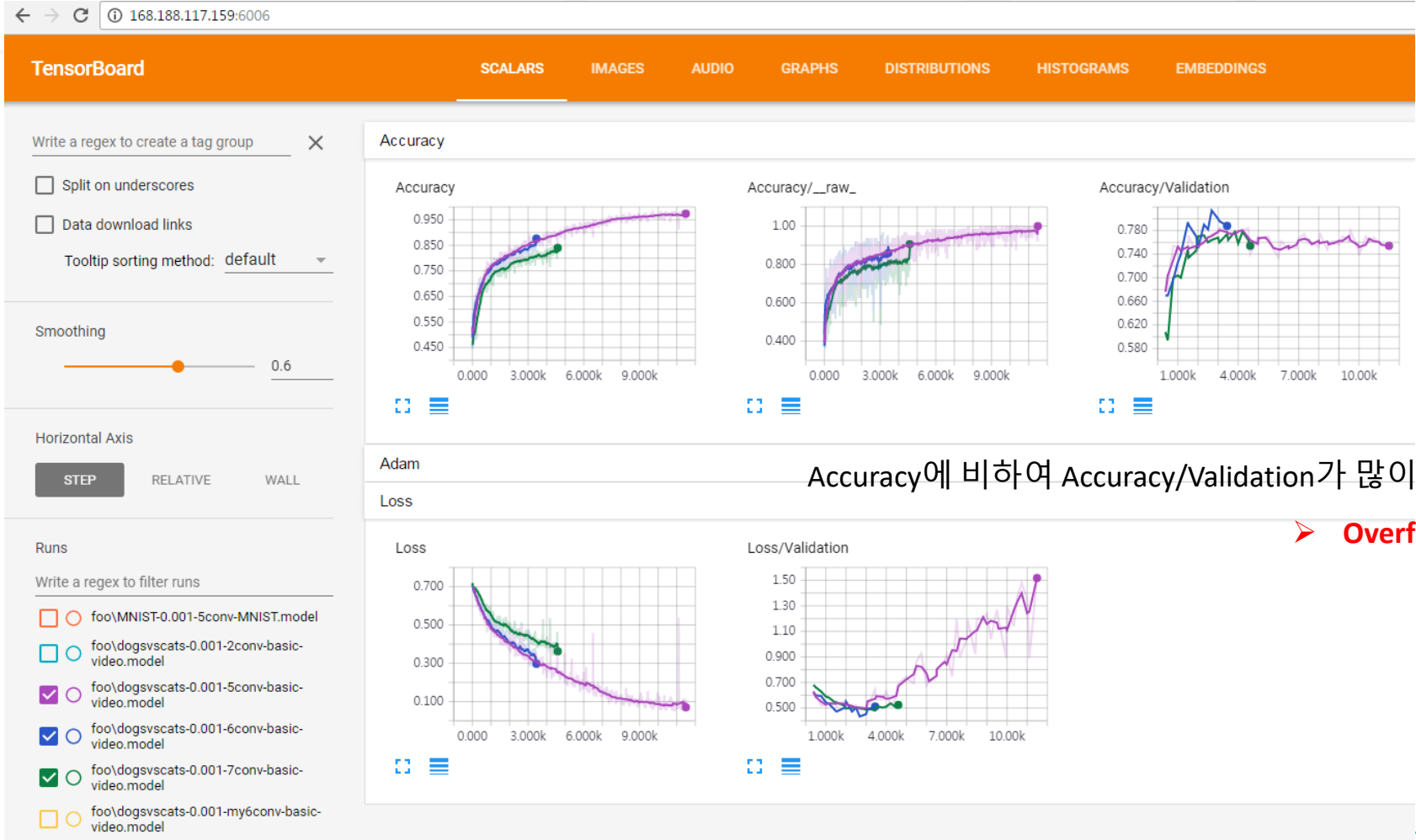
- foo\MNIST-0.001-5conv-MNIST.model
- foo\dogsvscats-0.001-2conv-basic-video.model
- foo\dogsvscats-0.001-5conv-basic-video.model
- foo\dogsvscats-0.001-6conv-basic-video.model
- foo\dogsvscats-0.001-7conv-basic-video.model

On the right side of the screenshot, there's a list of metrics: Accuracy, Adam, and Loss. A bracket groups these three metrics with the text "확인 가능한 데이터, 주로 accuracy와 loss 확인".

저장된  
학습 모델

# Dogs vs. Cats

## TensorBoard



# Dogs vs. Cats

```
In [13]: import matplotlib.pyplot as plt

# if you dont have this file yet
# test_data = process_test_data()
# if you already have it
test_data = np.load('test_data.npy')

fig = plt.figure()

for num, data, in enumerate(test_data[:12]):
    # cat : [1, 0]
    # dog : [0, 1]

    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(3, 4, num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE, IMG_SIZE, 1)

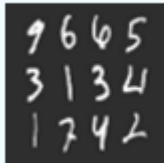
    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 1: str_label = 'Dog'
    else: str_label = 'Cat'

    y.imshow(orig, cmap = 'gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```



# MNIST : TF-learn & Keras



## Digit Recognizer

Learn computer vision fundamentals with the famous MNIST data

Getting Started · 3 years to go · Entered

9665407401 Digit Recognizer  
3134727121 Learn computer vision fundamentals with the famous MNIST data  
1742351244 1,594 teams · 3 years to go

Overview Data Kernels Discussion **Leaderboard** More My Submissions [Submit Predictions](#)

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
MNIST-0.001-keras-3conv-MNIST.mo...	11 days ago	2 seconds	0 seconds	0.99614

**Complete**

[Jump to your position on the leaderboard](#) ▾

[Public Leaderboard](#) [Private Leaderboard](#)

This leaderboard is calculated with approximately 25% of the test data.  
The final results will be based on the other 75%, so the final standings may be different. [Raw Data](#) [Refresh](#)

#	Δ1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	Jakob Peterlin			1.00000	5	2mo
2	—	linbo_jacas			1.00000	1	2mo
3	—	Tanaka			1.00000	1	1mo
4	—	XueHao Xiang			1.00000	8	1mo

# MNIST - Keras

```
In [1]: import cv2                # working with, mainly resizing, images
import numpy as np          # dealing with arrays
import os                  # dealing with directories
from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.
from tqdm import tqdm      # a percentage bar for tasks
import pandas as pd        # data processing, CSV file I/O (e.g. pd.read_csv)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import keras
from keras.utils import np_utils
from keras.models import Sequential
from keras.models import load_model

from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler

import tensorflow as tf
#1
```

Using TensorFlow backend.

# MNIST - Keras

```
In [2]: # get data
train_data = np.array(pd.read_csv('D:/deep/data/Kaggle_Data/MNIST/train/train.csv'))
test_data = np.array(pd.read_csv('D:/deep/data/Kaggle_Data/MNIST/test/test.csv'))

x_train, x_val, y_train, y_val = train_test_split(train_data[:,1:], train_data[:,0], test_size=0.1)

x_train = x_train.astype("float32")/255.0
x_val = x_val.astype("float32")/255.0
y_train = np_utils.to_categorical(y_train)
y_val = np_utils.to_categorical(y_val)

n_train = x_train.shape[0]
n_val = x_val.shape[0]
x_train = x_train.reshape(n_train, 28, 28, 1)
x_val = x_val.reshape(n_val, 28, 28, 1)
n_classes = y_train.shape[1]
```

# MNIST - Keras

```
In [3]: LR = 1e-3  
MODEL_NAME = 'MNIST-{}-{}.model'.format(LR, 'keras-conv-MNIST-0502-padding')  
#2
```

```
In [4]: model = Sequential()  
  
model.add(Conv2D(filters = 32, kernel_size = (3, 3), padding = 'same', activation='relu',  
                input_shape = (28, 28, 1)))  
model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation='relu'))  
model.add(BatchNormalization())  
model.add(Dropout(0.3))  
model.add(MaxPool2D(strides=(2,2)))  
model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'same', activation='relu'))  
model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation='relu'))  
model.add(Conv2D(filters = 128, kernel_size = (3, 3), activation='relu'))  
model.add(Conv2D(filters = 128, kernel_size = (3, 3), activation='relu'))  
model.add(BatchNormalization())  
model.add(Dropout(0.3))  
model.add(MaxPool2D(strides=(2,2)))  
  
model.add(Flatten())  
#model.add(Dense(512, activation='relu'))  
#model.add(Dropout(0.25))  
#model.add(Dense(1024, activation='relu'))  
#model.add(Dropout(0.5))  
model.add(Dense(10, activation='softmax'))
```



# MNIST - Keras

```
In [5]: datagen = ImageDataGenerator(zoom_range = 0.1,  
                                     height_shift_range = 0.1,  
                                     width_shift_range = 0.1,  
                                     rotation_range = 10)
```

```
In [6]: model.compile(loss='categorical_crossentropy', optimizer = Adam(lr=3e-5), metrics = ["accuracy"])  
if os.path.exists('{} .h5'.format(MODEL_NAME)):  
    model = load_model('{} .h5'.format(MODEL_NAME))  
    print('model loaded!')  
#6
```

```
In [ ]: hist = model.fit_generator(datagen.flow(x_train, y_train, batch_size = 16),  
                                   steps_per_epoch = 500, #increase this when not on Kaggle kernel  
                                   epochs = 100, #increase this when not on Kaggle kernel  
                                   verbose = 2, #verbose=1 outputs ETA, but doesn't work well in the cloud  
                                   validation_data = (x_val[:400,:], y_val[:400,:]), #To evaluate faster  
                                   callbacks = [annealer])
```

Epoch 1/100

10s - loss: 0.5677 - acc: 0.8127 - val\_loss: 0.2417 - val\_acc: 0.9375

Epoch 2/100

10s - loss: 0.2338 - acc: 0.9223 - val\_loss: 0.1808 - val\_acc: 0.9475

Epoch 3/100

10s - loss: 0.1999 - acc: 0.9376 - val\_loss: 0.2587 - val\_acc: 0.9275

Epoch 4/100

10s - loss: 0.1489 - acc: 0.9548 - val\_loss: 0.1999 - val\_acc: 0.9595

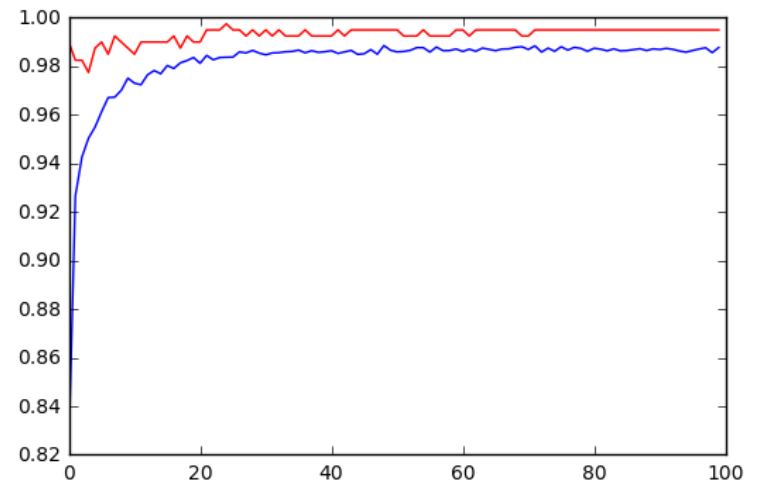
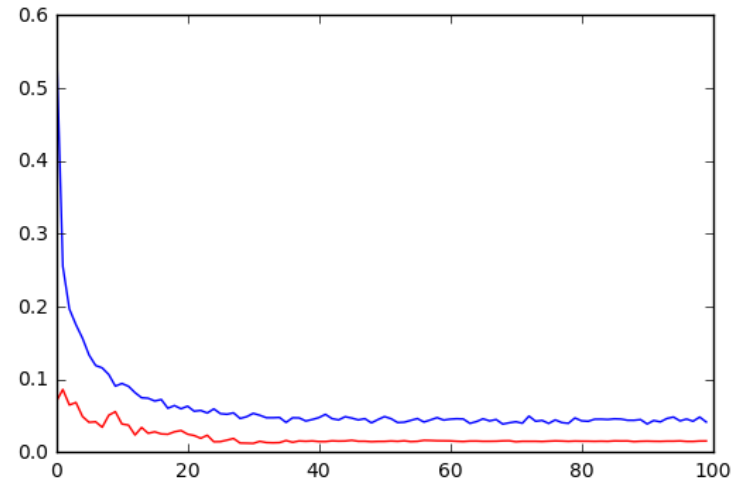
# MNIST - Keras

```
In [12]: model.save('{}_h5'.format(MODEL_NAME))
```

```
In [12]: model.evaluate(x_val, y_val, verbose=0)
```

```
Out [12]: [0.022180345102481493, 0.99404761904761907]
```

```
In [13]: import matplotlib.pyplot as plt  
plt.plot(hist.history['loss'], color='b')  
plt.plot(hist.history['val_loss'], color='r')  
plt.show()  
plt.plot(hist.history['acc'], color='b')  
plt.plot(hist.history['val_acc'], color='r')  
plt.show()
```



# MNIST - Keras

```
In [12]: x_test = test_data.astype("float32")/255.0
n_test = x_test.shape[0]
x_test = x_test.reshape(n_test, 28, 28, 1)

In [13]: y_hat = model.predict(x_test, batch_size=64)

In [14]: y_pred = np.argmax(y_hat,axis=1)

In [15]: with open('{} .csv'.format(MODEL_NAME), 'w') as f :
    f.write('ImageId,Label\n')
    for i in range(0, n_test) :
        f.write("".join([str(i+1), ',', str(y_pred[i]), '\n']))
```

Competitions Novice



Unranked



Digit Recognizer

3 years to go - Top 5%

83<sup>rd</sup>  
of 1662

83



Hyunho Jeon



0.99614

5

11d

Your Best Entry

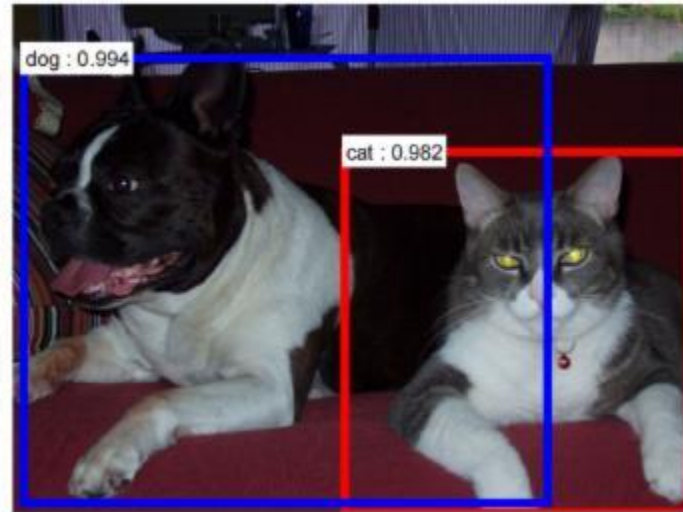
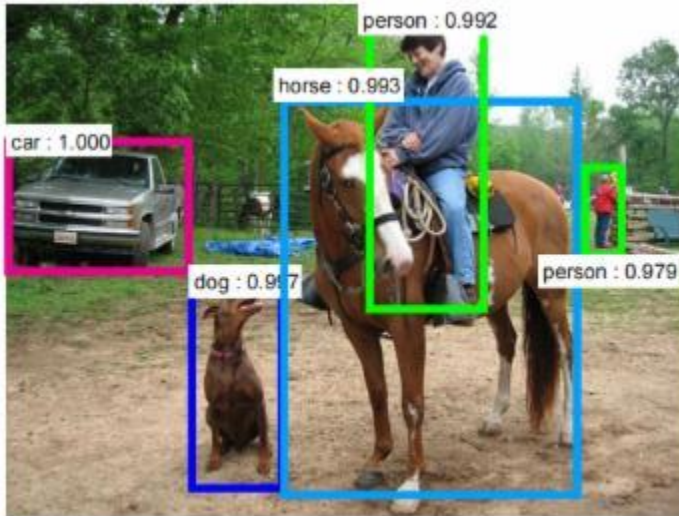
Your submission scored 0.99614, which is an improvement of your previous score of 0.99600. Great job!



Tweet this!

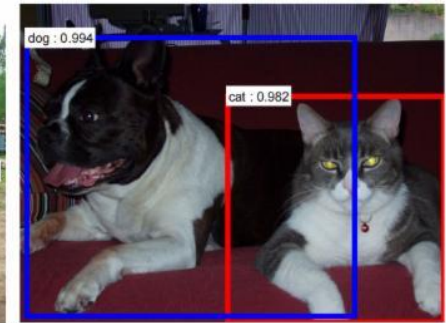
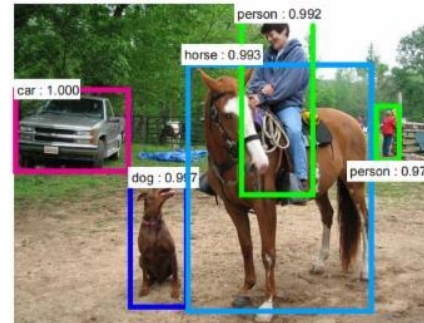
# Conclusion

- Object detection + RAFSet ME



# Conclusion

- Future work : Imagenet classification and Object detection + RAFSet ME



- Future work...? : GAN or Self-driving car simulation



(b) Handbag images (input) & **Generated** shoe images (output)



(c) Shoe images (input) & **Generated** handbag images (output)



# Q & A