DIGITAL IMAGE PROCESSING

-APPLIED MATHEMATICS CAPSTONE DESIGN-

D.I.P 응용수학과
김하현 , 최우빈

목차

- 주제 ; cats and dogs binary classification
- (1) CNN(Convolutional Neural Network , 합성 곱 신경망)을 이용한 직접 모델링 후, cats 와 dogs 의 각각 data set 을 training 한 후 accuracy 측정 – Python
- (2)기존 data set의 image 들을 fourier transform의 한 종류인 DWT(Discrete Wavelet Transform)한 image 들로 training 한 후, accuracy 측정 – DWT : Matlab 으로 RGB 변환
- (3)검증된 알고리즘을 통해, 기존 data set과 DWT data set 의 accuracy 측정 (Alex net, Google net)
- -Matlab R2017b의 toolbox

CATS AND DOGS TRAINING , TEST , AUGMENTATION SET <FILE PATH>

```
TRAIN_CATS_DIR = 'input/train/cats/'
TRAIN_DOGS_DIR = 'input/train/dogs/'
TEST_DIR = 'input/test/'
VALD_CATS_DIR = 'input/validation/cats/'
VALD_DOGS_DIR = 'input/validation/dogs/'
AUG_DIR = 'input/train/aug/'
```

Image : 128 x 128 pixels

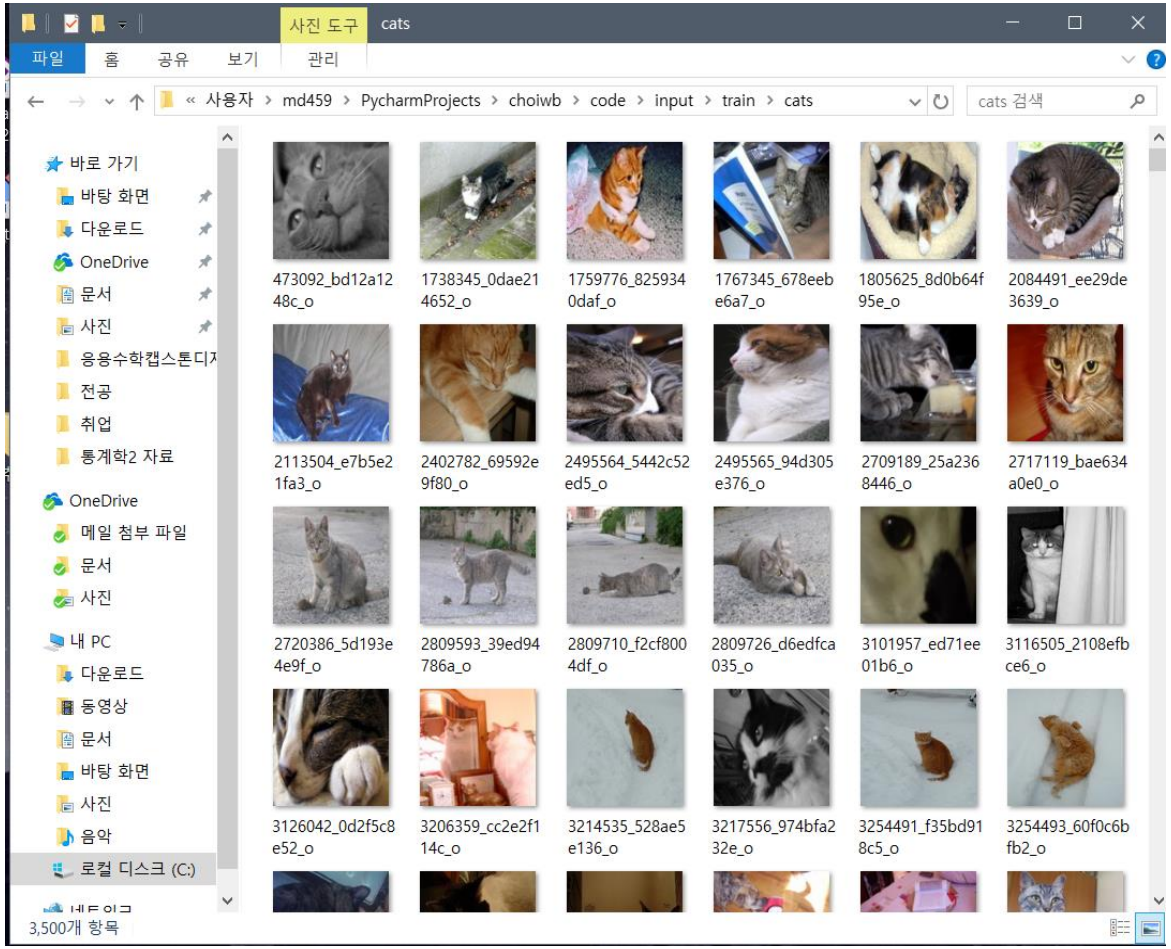


사진 도구
cats
파일
홈
공유
보기
관리
« 사용자 › md459 › PycharmProjects › choiwb › code › input › train › cats
cats 검색
바로 가기
바탕 화면
다운로드
OneDrive
문서
사진
응용수학캡스톤디자
전공
취업
통계학2 자료
OneDrive
메일 첨부 파일
문서
사진
내 PC
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
473092_bd12a1248c_o
1738345_0dae214652_o
1759776_8259340daf_o
1767345_678eebe6a7_o
1805625_8d0b64f95e_o
2084491_ee29de3639_o
2113504_e7b5e21fa3_o
2402782_69592e9f80_o
2495564_5442c52ed5_o
2495565_94d305e376_o
2709189_25a2368446_o
2717119_bae634a0e0_o
2720386_5d193e4e9f_o
2809593_39ed94786a_o
2809710_f2cf8004df_o
2809726_d6edfca035_o
3101957_ed71ee01b6_o
3116505_2108efbce6_o
3126042_0d2f5c8e52_o
3206359_cc2e2f114c_o
3214535_528ae5e136_o
3217556_974bfa232e_o
3254491_f35bd918c5_o
3254493_60f0c6bfb2_o
3,500개 항목

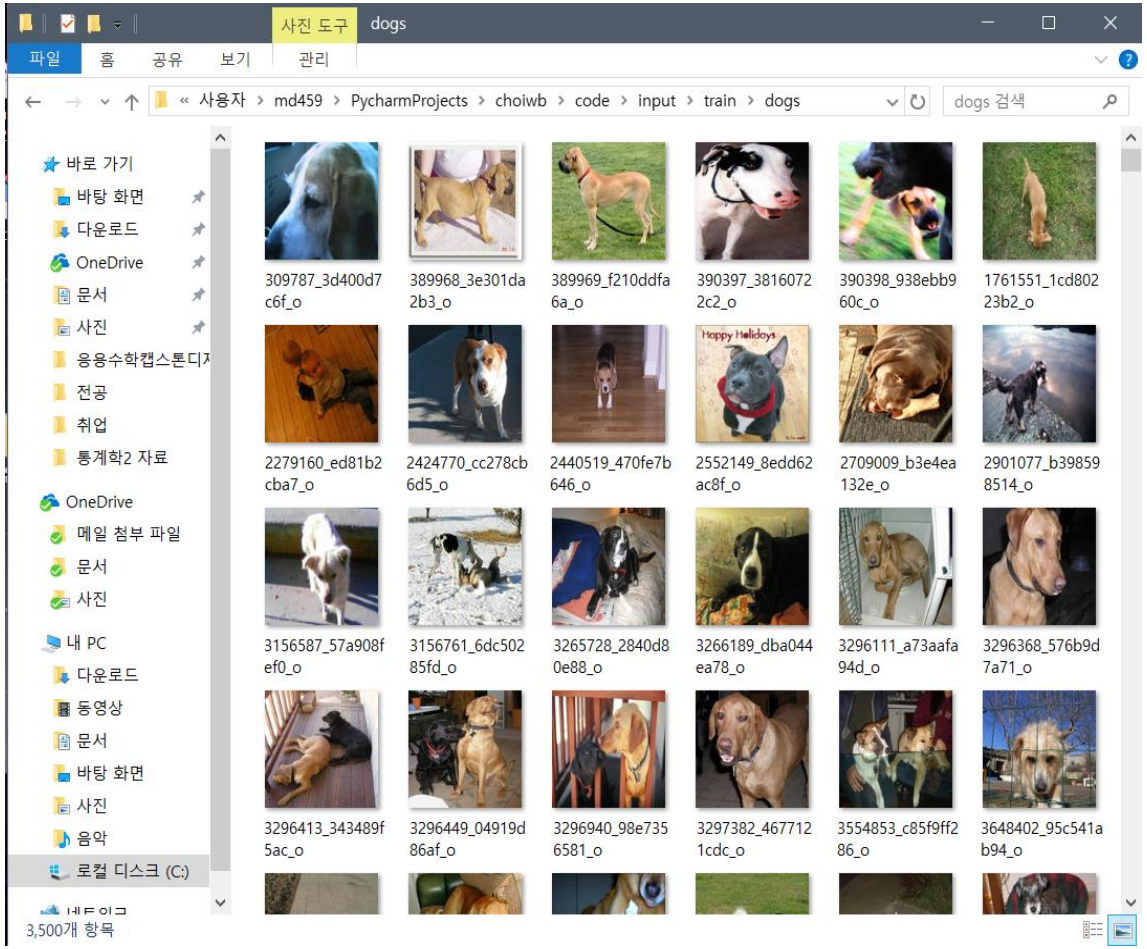
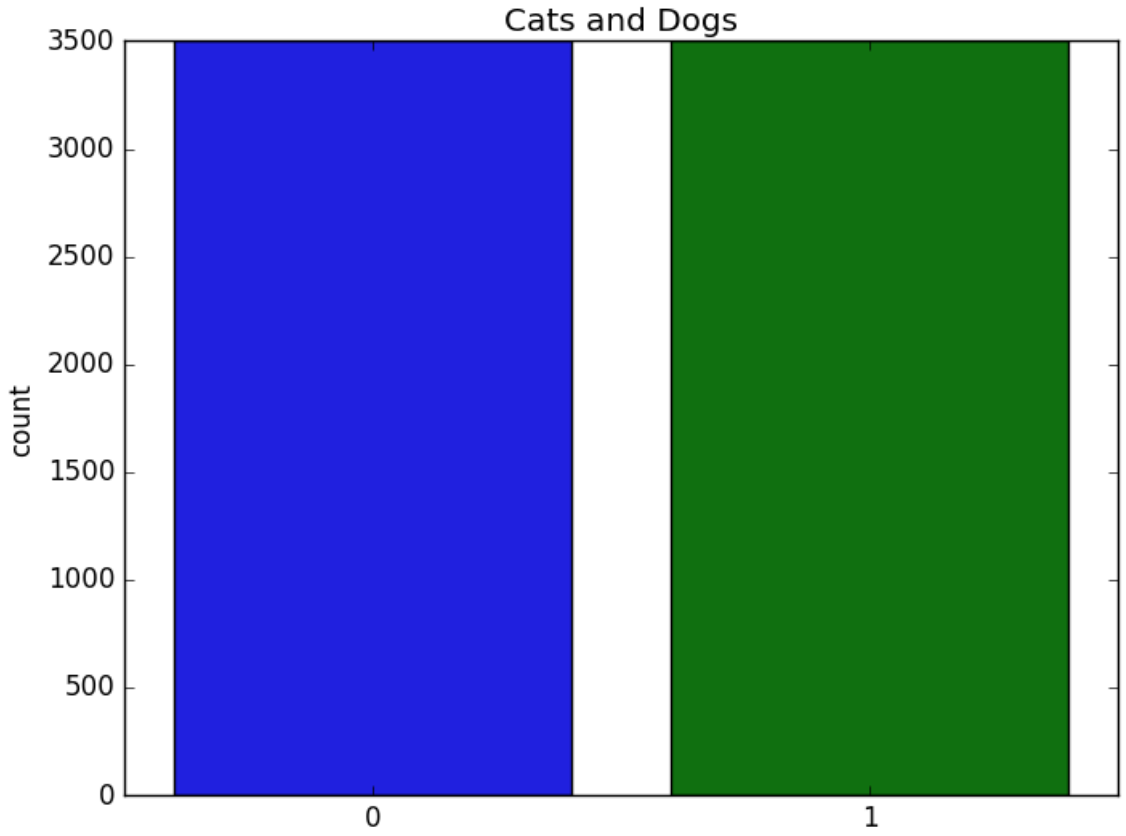


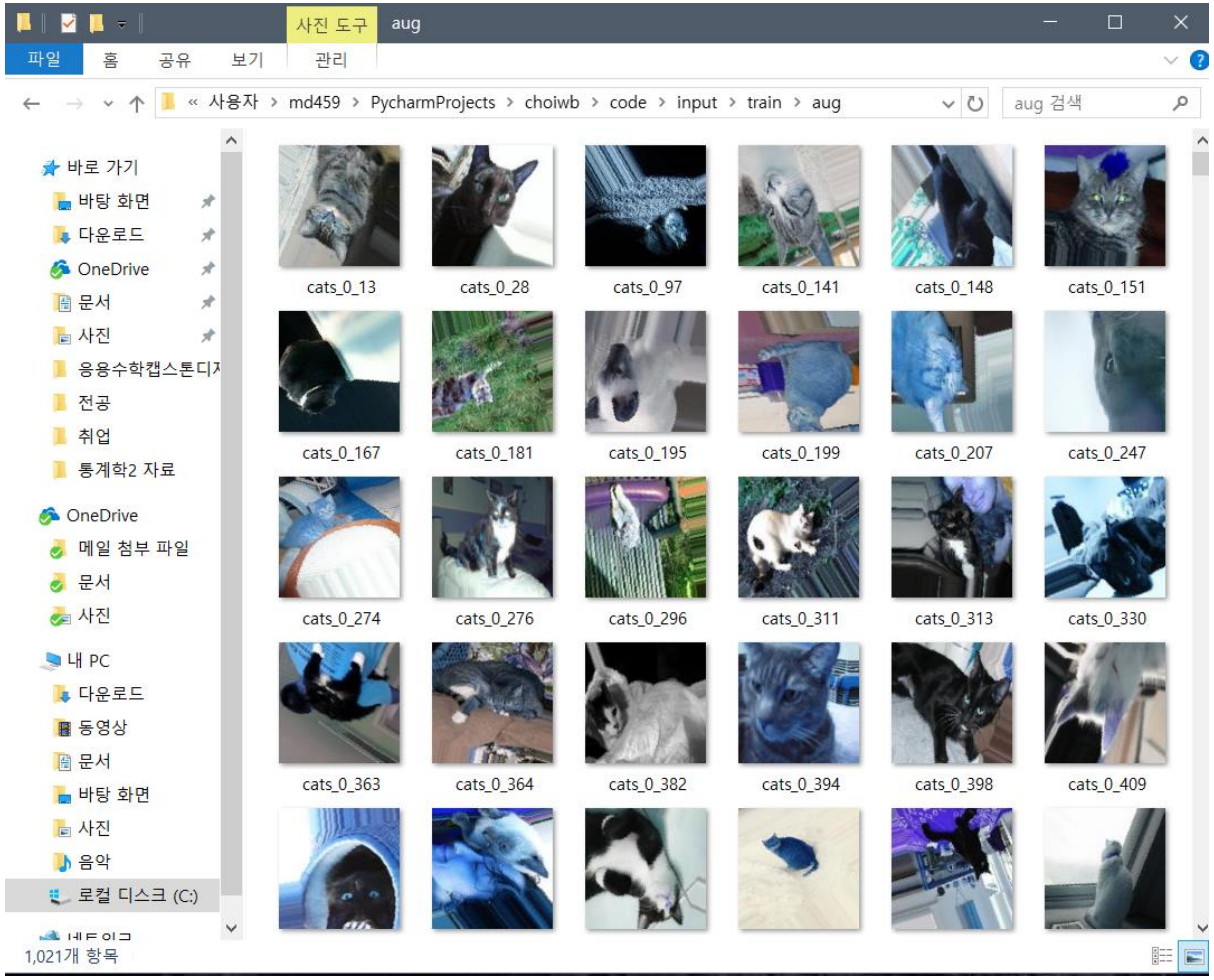
사진 도구
dogs
파일
홈
공유
보기
관리
« 사용자 › md459 › PycharmProjects › choiwb › code › input › train › dogs
dogs 검색
바로 가기
바탕 화면
다운로드
OneDrive
문서
사진
응용수학캡스톤디자
전공
취업
통계학2 자료
OneDrive
메일 첨부 파일
문서
사진
내 PC
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
309787_3d400d7c6f_o
389968_3e301da2b3_o
389969_f210ddfa6a_o
390397_38160722c2_o
390398_938ebb960c_o
1761551_1cd80223b2_o
2279160_ed81b2cba7_o
2424770_cc278cb6d5_o
2440519_470fe7b646_o
2552149_8edd62ac8f_o
2709009_b3e4ea132e_o
2901077_b398598514_o
3156587_57a908fef0_o
3156761_6dc50285fd_o
3265728_2840d80e88_o
3266189_dba044ea78_o
3296111_a73aafa94d_o
3296368_576b9d7a71_o
3296413_343489f5ac_o
3296449_04919d86af_o
3296940_98e7356581_o
3297382_4677121cdc_o
3554853_c85f9ff286_o
3648402_95c541ab94_o
3,500개 항목

TRAINING SET (CATS:; 3500 , DOGS : 3500),



AUGMENTATION 과정

- round(len(train) / 50)

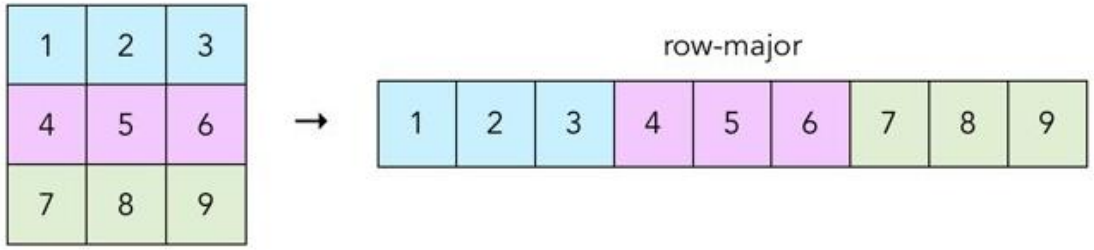


TRANSPOSE(대각화) : [COUNT , ROW , COLUMN , CHANNEL] -> [COUNT , CHANNEL , ROW , COLUMN]

teosorflow : [count , row , column . channell default(기본값) 시작 과정 (CPU)

row major order(행 우선 순위)

속도 때문 !



NVIDIA cuDNN : [count , channel , row , columnl default(기본값) 모델링 과정 (GPU)

- gray scale : 1 channel
- RGB scale : 3 channel

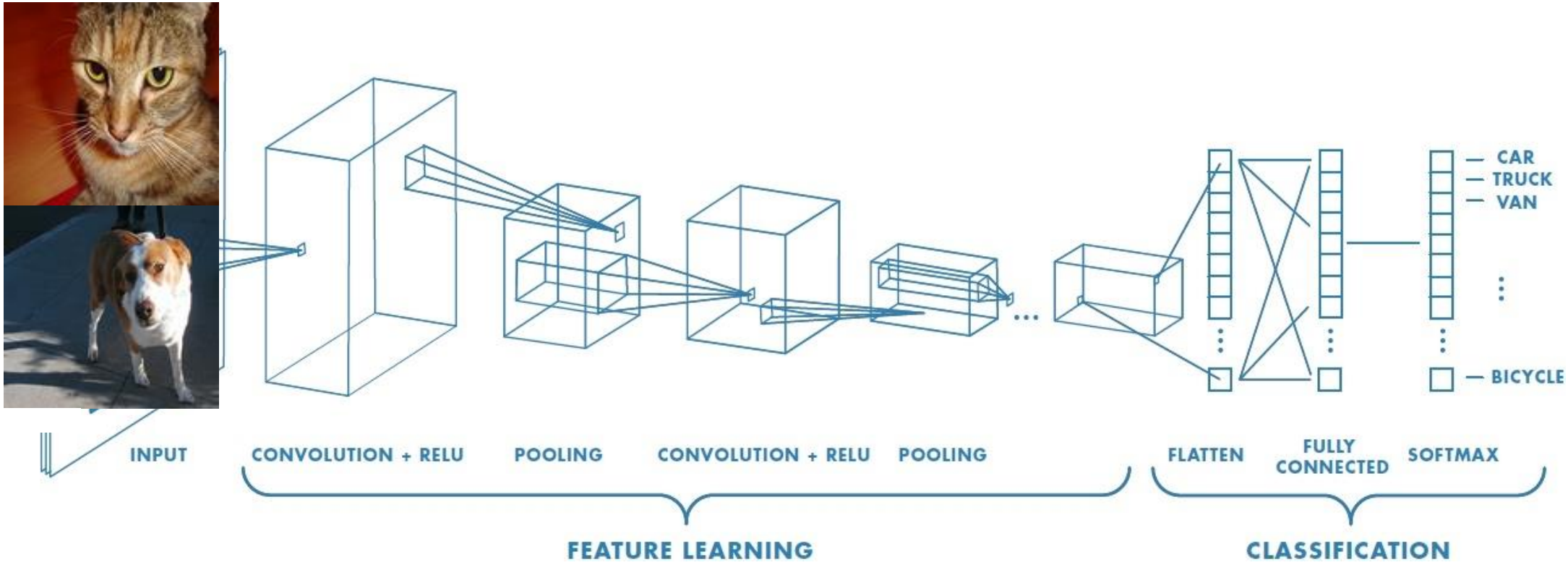
- Train set 형태 : (7000, 3, 128, 128)

augmentation 적용

- Train set 형태 : (7890, 3, 128, 128)
- Validation set 형태 : (1964, 3, 128, 128)

Validation_split : 20% Train on 6312 samples, validate on 1578 samples

CNN(CONVOLUTIONAL NEURAL NETWORK) 모델링 과정



- CNN 모델링 과정

Input layer(128 X 128 픽셀) : 16 neurons, 4 x 4 , ReLU , max_norm = 2

Max pooling layer 3 x 3

Hidden layer : 32 neurons 4 x 4 , ReLU , max_norm = 2

Max pooling layer 3 x 3

Hidden layer 64 neurons 4 x 4 , ReLU , max_norm = 2

Max pooling layer 3 x 3

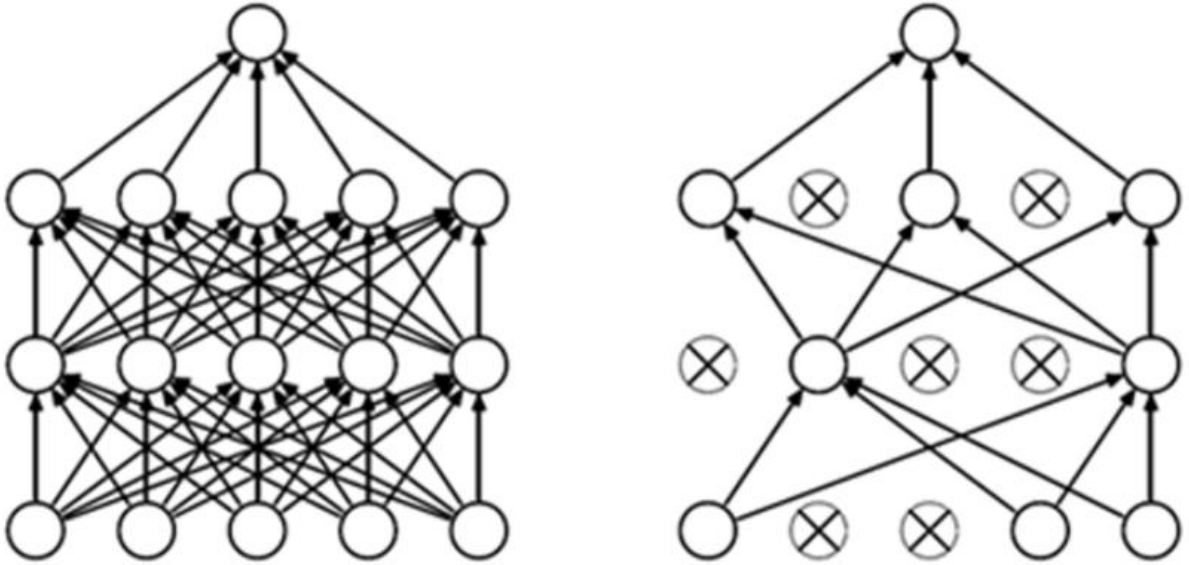
- Model flatten
- Output Layer 128 neurons , Sigmoid , Dropout : 0.3

- **max_norm :** 정규화(**normalization**)의 형태, 모든 뉴런에 대한 **weight vector**의 크기에 **upper bound**를 적용하고 **gradient descent**를 사용

- **flatten layer :** 2차원 행렬 형태를 1차원 벡터 형태로 변환

- **dropout :**

overfitting 피하기 위함



(a) Standard Neural Net

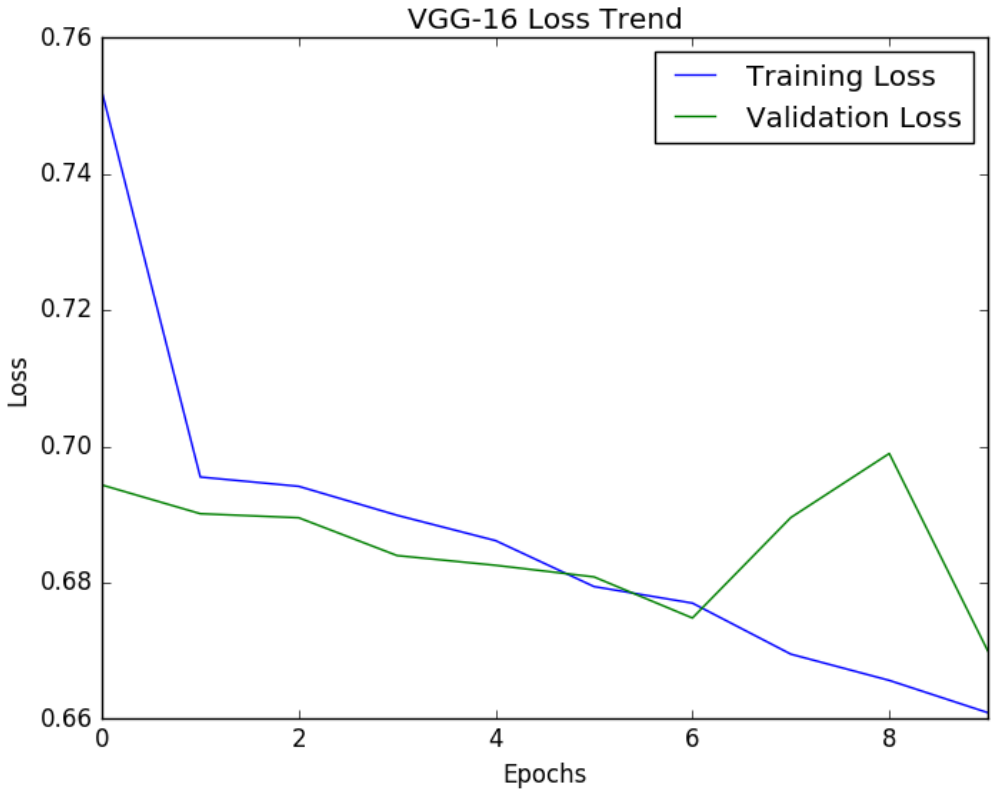
(b) After applying dropout.

모델 연산과정

- Epoch 1/10
- 80s - loss: 0.7521 - acc: 0.5021 - val_loss: 0.6943 - val_acc: 0.5006
- Epoch 2/10
- 89s - loss: 0.6955 - acc: 0.5120 - val_loss: 0.6901 - val_acc: 0.5406
- Epoch 3/10
- 84s - loss: 0.6941 - acc: 0.5119 - val_loss: 0.6895 - val_acc: 0.5393
- Epoch 4/10
- 82s - loss: 0.6899 - acc: 0.5368 - val_loss: 0.6840 - val_acc: 0.5507
- Epoch 5/10
- 74s - loss: 0.6862 - acc: 0.5433 - val_loss: 0.6825 - val_acc: 0.5634

- Epoch 6/10
- 80s - loss: 0.6794 - acc: 0.5700 - val_loss: 0.6808 - val_acc: 0.5653
- Epoch 7/10
- 74s - loss: 0.6770 - acc: 0.5762 - val_loss: 0.6748 - val_acc: 0.5659
- Epoch 8/10
- 78s - loss: 0.6695 - acc: 0.5848 - val_loss: 0.6896 - val_acc: 0.5406
- Epoch 9/10
- 74s - loss: 0.6656 - acc: 0.6008 - val_loss: 0.6990 - val_acc: 0.5640
- Epoch 10/10
- 79s - loss: 0.6609 - acc: 0.6006 - val_loss: 0.6700 - val_acc: 0.5875
- Baseline Accuracy: 60.08% , 연산시간: 987.7676200866699 , 약16분 28초

LOSS TREND


VGG-16 Loss Trend
Training Loss
Validation Loss
Loss
Epochs
0.76
0.74
0.72
0.70
0.68
0.66
0
2
4
6
8

<개선 방향>

1. Train set과 augmentation set 데이터 증가
2. Googlenet 알고리즘 적용
3. Model parameter / hyper parameter 조정

*Hyperparameter : Bayesian statistics에서 prior distribution의 parameter이다. → Posterior distribution만으로 추정을 하는 것보다 prior distribution을 알 때, 훨씬 더 잘 추정

*Neural Network에서 hyperparameter

신경망 학습을 통해서 tuning 또는 optimization 해야 하는 주 변수가 아니라 사람들이 prior 설정을 하거나,

외부 모델 메커니즘을 통해 자동으로 설정이 되는 변수

Neural Network 에서 Hyperparameter이란 (= free parameter)

Ex) learning rate , cost function(cross entropy , least square) ,
mini batch size , training 회수, hidden layer 개수,
weights initialization

-cross entropy

$$H(p, m) = -\sum_i p(xi)\log(m(xi))$$

p , m : 확률 분포

기대 값과 예측 값의 차이가 클수록 결과가 크게 나온다.

결과는 항상 양수 이다.

-cross entropy cost function

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

y : 기대 값

a : 신경망에서 출력된 값

n : 훈련 데이터의 개수

중간결과에서 개선점

- Training set 증가 (7000 → 15724)
- Batch Normalization 적용
- neuron 개수 증가 (GPU 연산)

[input – hidden1 – hidden2 – output]

[16 – 32 – 64 - 128] → [64 – 128 – 256 – 512]

BATCH NORMALIZATION

Gradient Vanishing / Gradient Exploding 이 일어나지 않도록 하는 아이디어 중의 하나. 지금까지는 이 문제를 Activation 함수의 변화 (ReLU 등), Careful Initialization, small learning rate 등으로 해결하였지만, 이런 간접적인 방법보다는 training 하는 과정 자체를 전체적으로 안정화 하여

학습 속도를 가속시킬 수 있는 근본적인 방법.

→ Layer Normalization !

*** Internal Covariate Shift : Network의 각 층이나 Activation 마다 input의 distribution이 달라지는 현상.

↓

각 층의 input의 distribution을 평균 0, 표준편차 1인 input으로 normalize 시키는 방법 고안.

LAYER NORMALIZATION

- State of the art한 DNN 학습시간 단축.
- 중간 층의 출력을 정규화 함으로써 실현.

***Covariate Shift(공 변량 변화)

: 입출력 규칙(주어진 입력에 대하여 출력의 생성규칙)은 training할 때와, test 할 때 다르지 않지만, 입력(공 변량)의 분포가 training 할 때와, test 할 때 다른 상황.

INTERNAL COVARIATE SHIFT

<CNN에서의 각 층 마다 입력분포>

(1)입력 층 으로의 입력분포

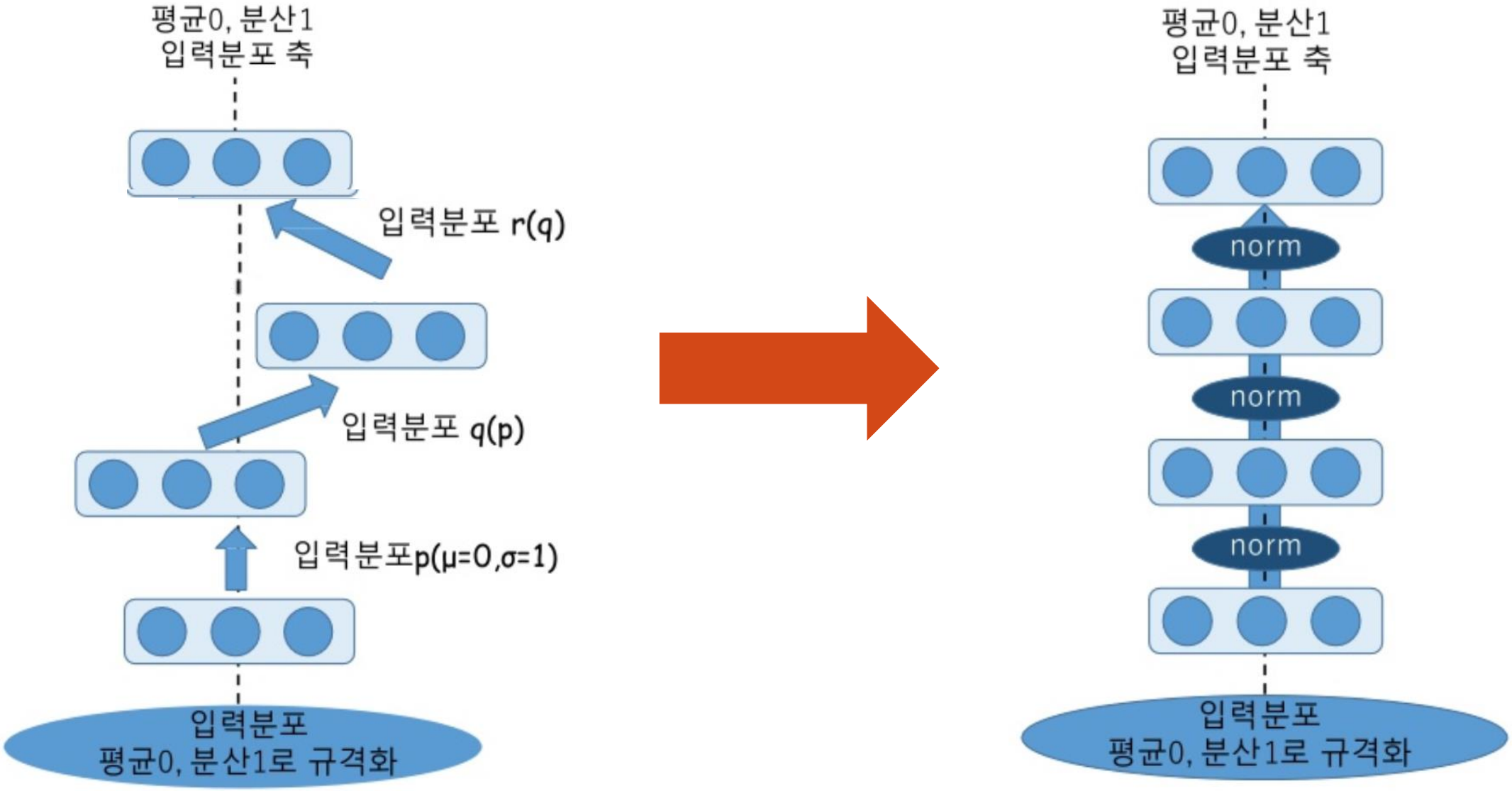
평균 0 , 분산 1로 선형변환

(2)중간 층 으로의 입력분포

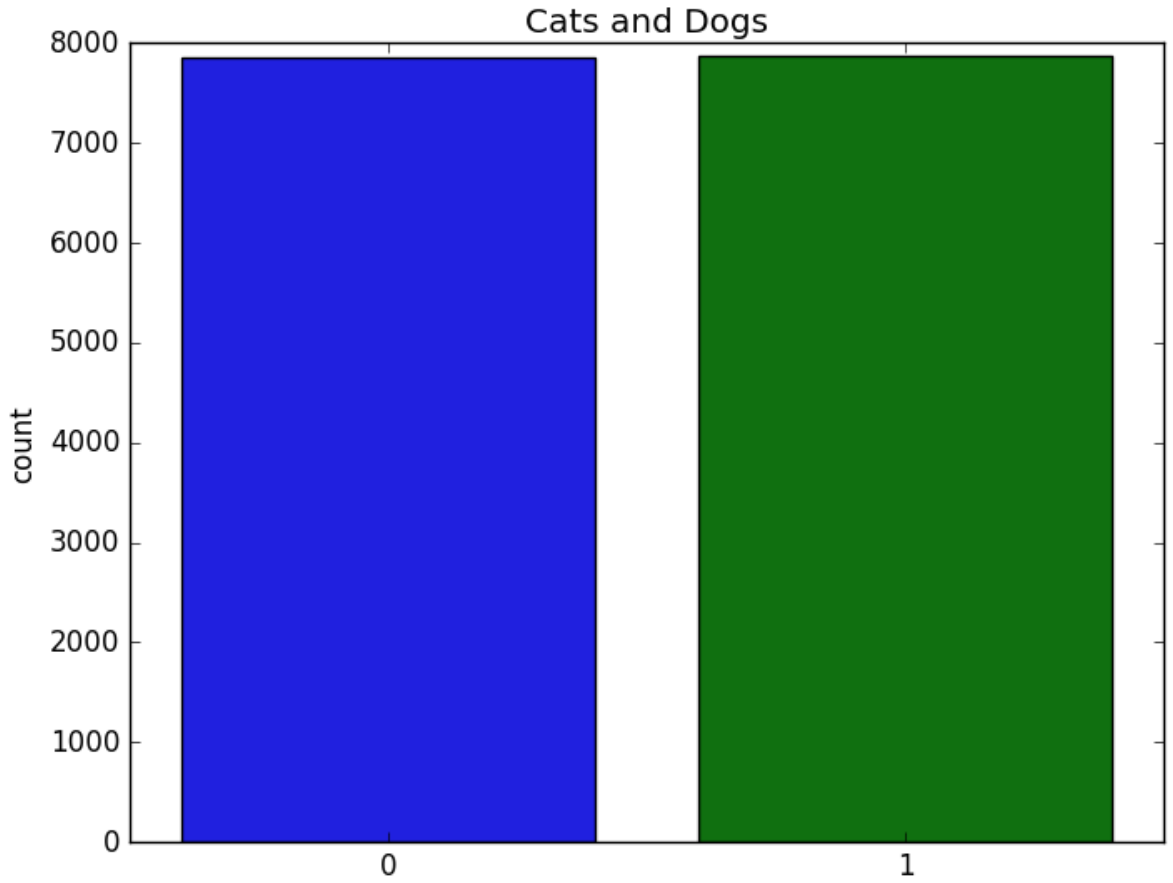
-정해진 입력분포를 유지하지 못함!

학습할 때 마다, 중간층으로의
입력분포가 변화
(Internal Covariate Shift)

NORMALIZATION PROCESS



TRAINING SET (CATS:; 7853 , DOGS : 7871),



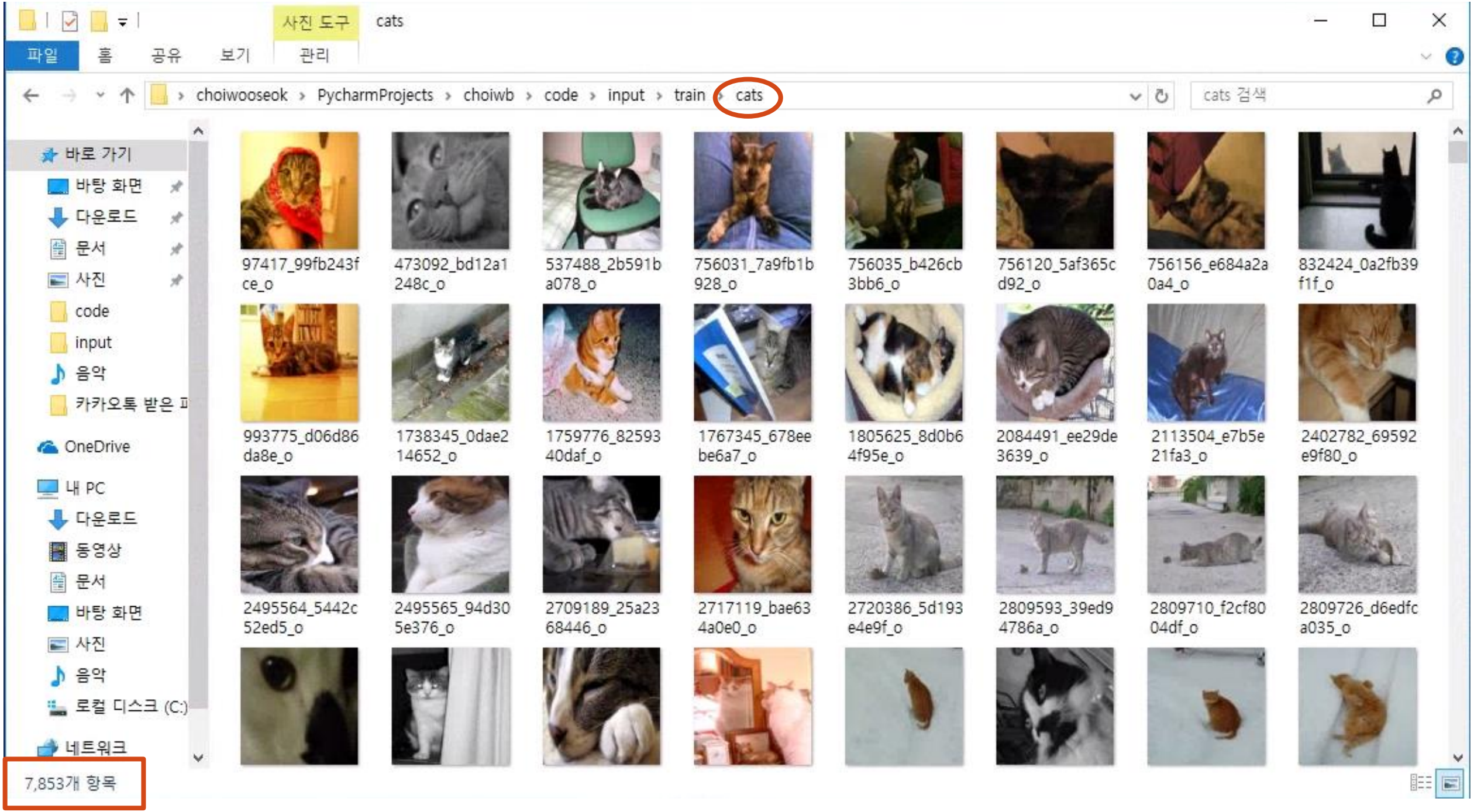


사진 도구
cats
파일
홈
공유
보기
관리
choiwooseok › PycharmProjects › choiwb › code › input › train › cats
cats 검색
바로 가기
바탕 화면
다운로드
문서
사진
code
input
음악
카카오톡 받은 파
OneDrive
내 PC
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
네트워크
97417_99fb243fce_o
473092_bd12a1248c_o
537488_2b591ba078_o
756031_7a9fb1b928_o
756035_b426cb3bb6_o
756120_5af365cd92_o
756156_e684a2a0a4_o
832424_0a2fb39f1f_o
993775_d06d86da8e_o
1738345_0dae214652_o
1759776_8259340daf_o
1767345_678eebe6a7_o
1805625_8d0b64f95e_o
2084491_ee29de3639_o
2113504_e7b5e21fa3_o
2402782_69592e9f80_o
2495564_5442c52ed5_o
2495565_94d305e376_o
2709189_25a2368446_o
2717119_bae634a0e0_o
2720386_5d193e4e9f_o
2809593_39ed94786a_o
2809710_f2cf8004df_o
2809726_d6edfca035_o
7,853개 항목

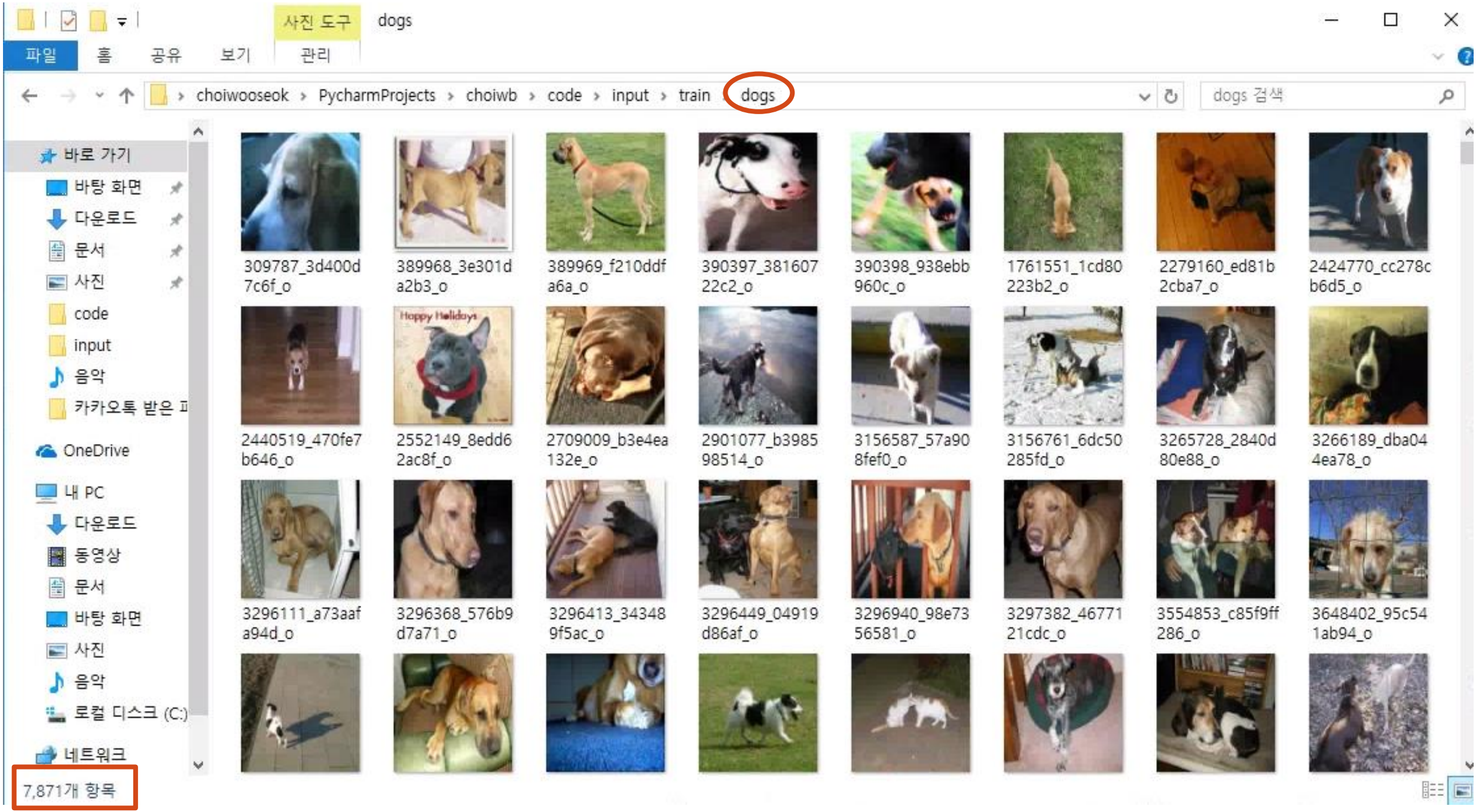
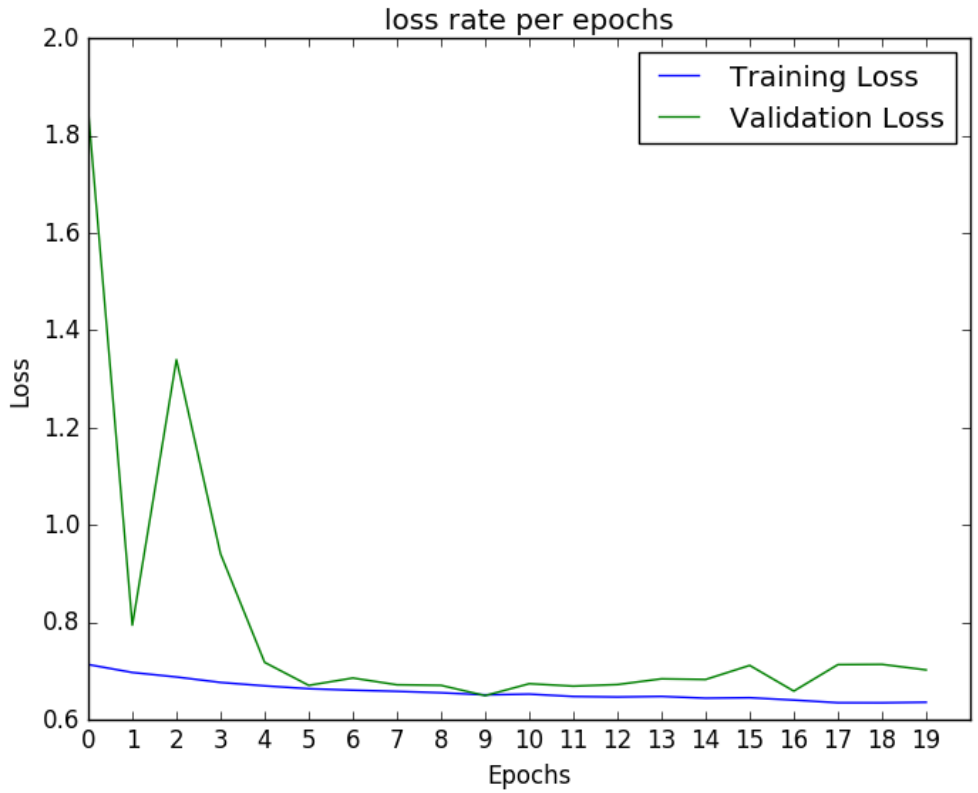


사진 도구
dogs
파일
홈
공유
보기
관리
choiwooseok › PycharmProjects › choiwb › code › input › train › dogs
dogs 검색
바로 가기
바탕 화면
다운로드
문서
사진
code
input
음악
카카오톡 받은 파
OneDrive
내 PC
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
네트워크
7,871개 항목
309787_3d400d7c6f_o
389968_3e301da2b3_o
389969_f210ddfa6a_o
390397_38160722c2_o
390398_938ebb960c_o
1761551_1cd80223b2_o
2279160_ed81b2cba7_o
2424770_cc278cb6d5_o
2440519_470fe7b646_o
2552149_8edd62ac8f_o
2709009_b3e4ea132e_o
2901077_b398598514_o
3156587_57a908fef0_o
3156761_6dc50285fd_o
3265728_2840d80e88_o
3266189_dba044ea78_o
3296111_a73aafa94d_o
3296368_576b9d7a71_o
3296413_343489f5ac_o
3296449_04919d86af_o
3296940_98e7356581_o
3297382_4677121cdc_o
3554853_c85f9ff286_o
3648402_95c541ab94_o

MODELING

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 3, 128, 64)	131136
batch_normalization_1 (Batch	(None, 3, 128, 64)	256
activation_1 (Activation)	(None, 3, 128, 64)	0
conv2d_2 (Conv2D)	(None, 3, 128, 64)	65600
batch_normalization_2 (Batch	(None, 3, 128, 64)	256
activation_2 (Activation)	(None, 3, 128, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 3, 42, 21)	0
conv2d_3 (Conv2D)	(None, 3, 42, 128)	43136
batch_normalization_3 (Batch	(None, 3, 42, 128)	512
activation_3 (Activation)	(None, 3, 42, 128)	0
conv2d_4 (Conv2D)	(None, 3, 42, 128)	262272
batch_normalization_4 (Batch	(None, 3, 42, 128)	512
activation_4 (Activation)	(None, 3, 42, 128)	0
max_pooling2d_2 (MaxPooling2	(None, 3, 14, 42)	0
conv2d_5 (Conv2D)	(None, 3, 14, 256)	172288
batch_normalization_5 (Batch	(None, 3, 14, 256)	1024
activation_5 (Activation)	(None, 3, 14, 256)	0
conv2d_6 (Conv2D)	(None, 3, 14, 256)	1048832
batch_normalization_6 (Batch	(None, 3, 14, 256)	1024
activation_6 (Activation)	(None, 3, 14, 256)	0
max_pooling2d_3 (MaxPooling2	(None, 3, 4, 85)	0
conv2d_7 (Conv2D)	(None, 3, 4, 512)	696832
batch_normalization_7 (Batch	(None, 3, 4, 512)	2048
activation_7 (Activation)	(None, 3, 4, 512)	0
conv2d_8 (Conv2D)	(None, 3, 4, 512)	4194816
batch_normalization_8 (Batch	(None, 3, 4, 512)	2048
activation_8 (Activation)	(None, 3, 4, 512)	0
max_pooling2d_4 (MaxPooling2	(None, 3, 1, 170)	0
flatten_1 (Flatten)	(None, 510)	0
dense_1 (Dense)	(None, 512)	261632
batch_normalization_9 (Batch	(None, 512)	2048
activation_9 (Activation)	(None, 512)	0

```
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
_________________________________________________________________
batch_normalization_11 (Batc (None, 1)                 4
_________________________________________________________________
activation_11 (Activation)   (None, 1)                 0
=================================================================
Total params: 7,151,493
Trainable params: 7,145,603
Non-trainable params: 5,890
_________________________________________________________________
```



하드웨어 사양 , 연산시간 , ACCURACY

- 프로세서 : Intel(R) core TM i5 – 6500 CPU @ 3.20GHz
- RAM : 8.00 GB
- GPU : NVIDIA GeForce GTX 1060 3 GB

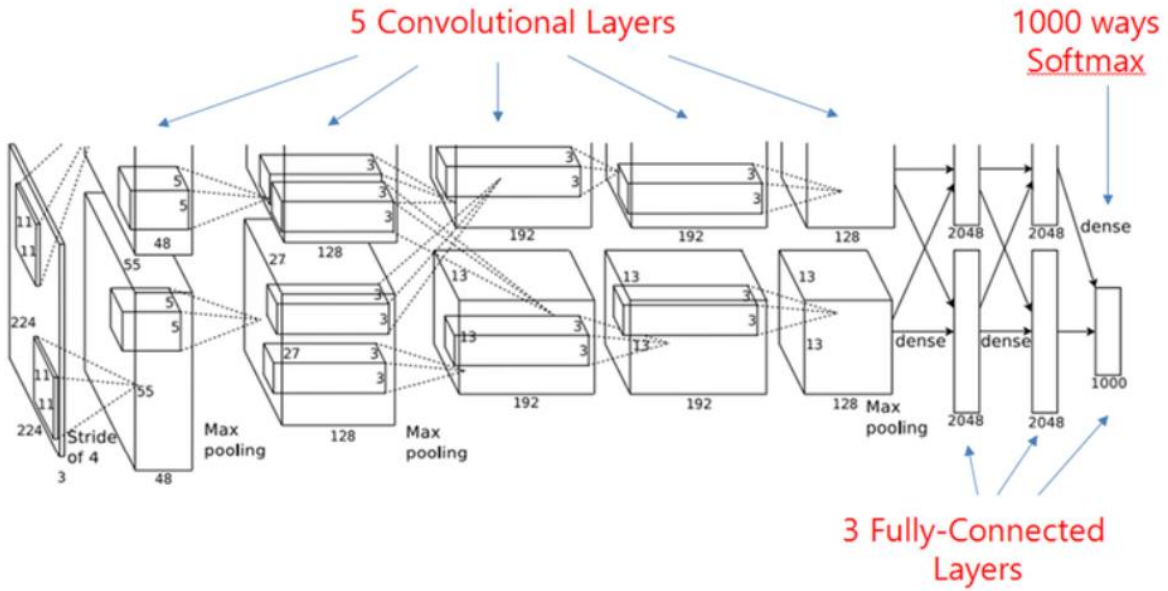
- 연산시간 : 약 28분 4초
- Accuracy : 70.67%

```
Accuracy: 70.67%
[INFO] Saving the model and weights...
Saved model to disk
연산시간: 1684.4383420944214
```

ALEX NET , GOOGLE NET

- Alex net 이란?
- ImageNet 영상 데이터 베이스를 기반으로 한, ILSVRC(ImageNet Large Scale Visual Recognition Challenge) – 2012 우승 한 CNN 알고리즘

→ 2개의 GPU를 기반으로 한 병렬 구조



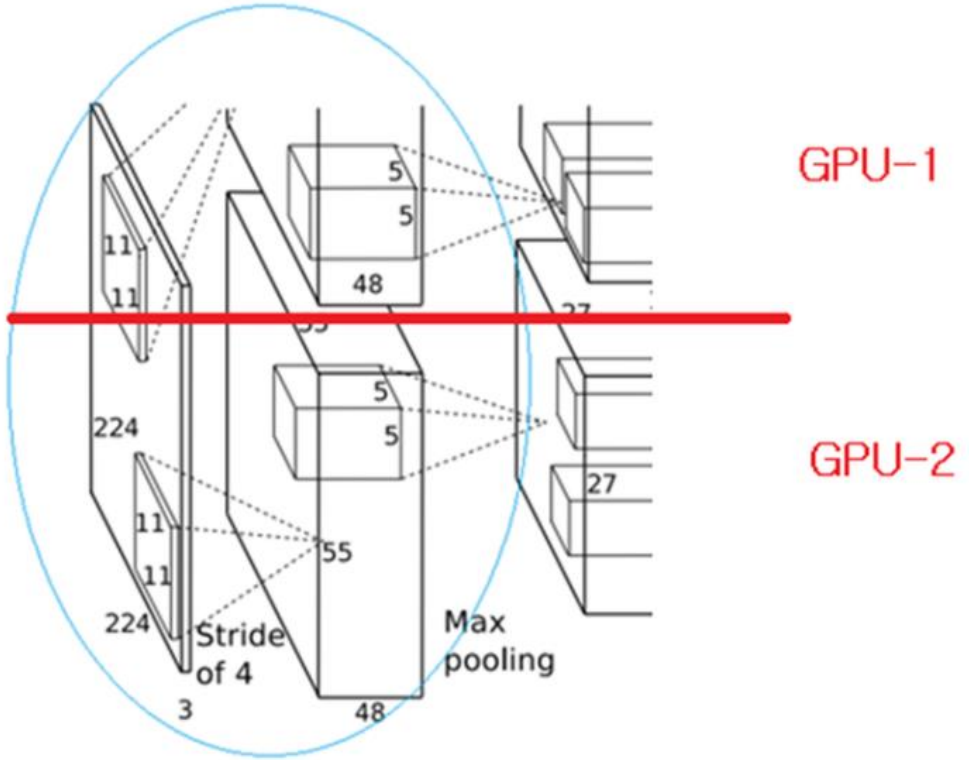
Alex net 구조

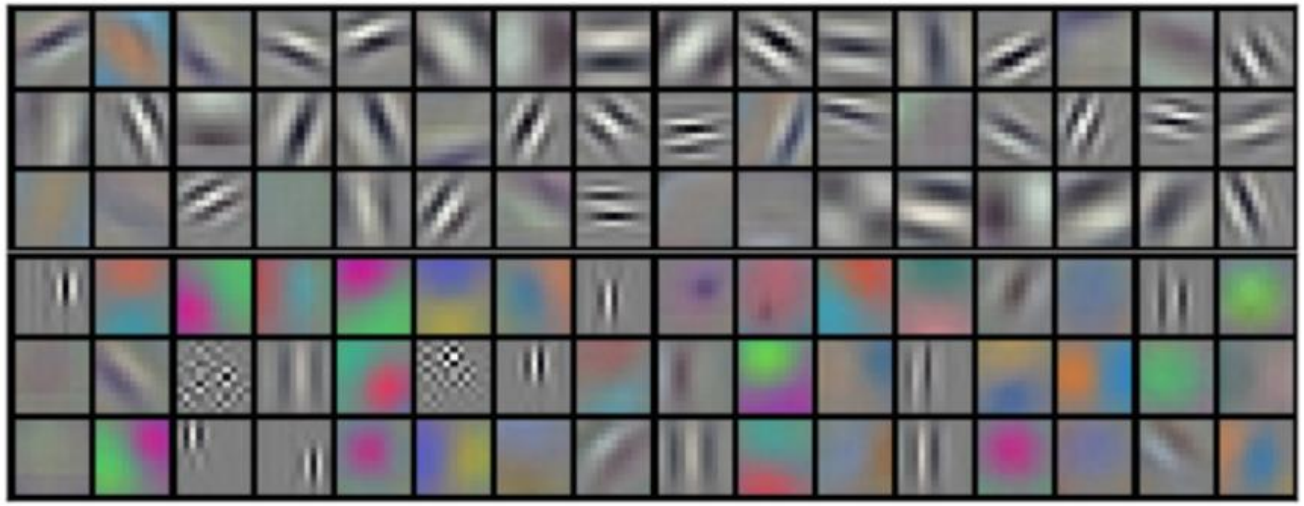
- GPU-1 : color과 상관 없는 정보를 추출 하기 위

Kernel 이 학습

- GPU-2 : 주로 color과 관련된 정보를 추출하기

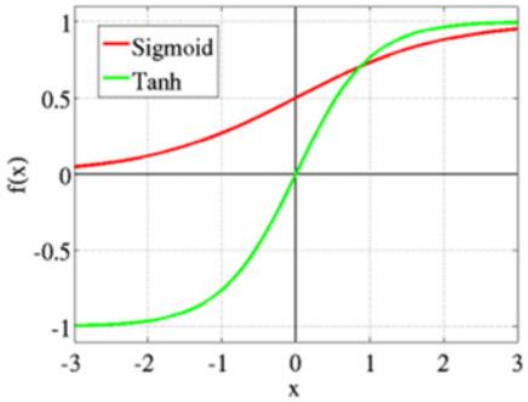
Kernel이 학습

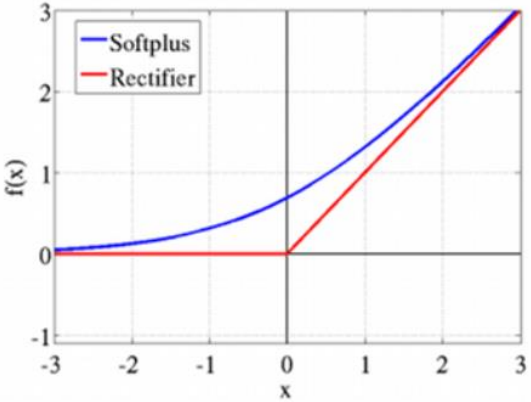




- **Alex net** 성능 향상

(1) **ReLU : x = 0**에서 미분이 안되지만, **Sigmoid**나 **Tanh**보다 학습 속도 탁월, **back propagation** 결과도 단순



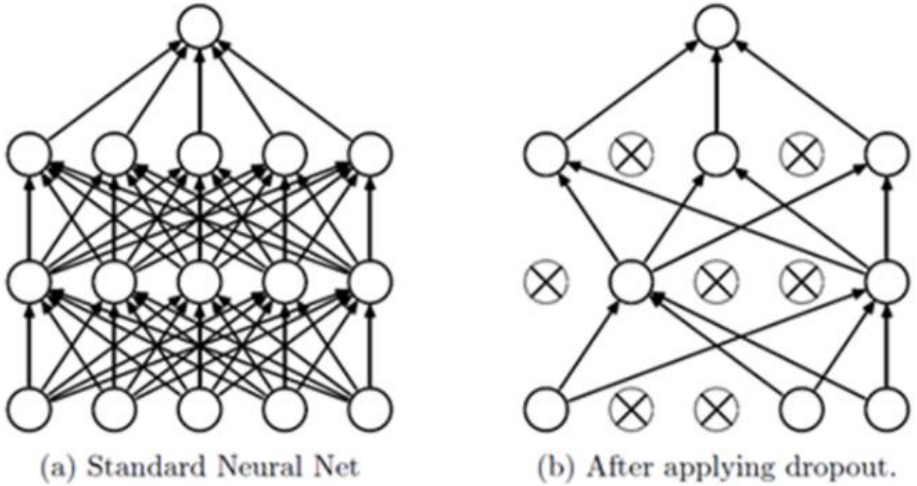


(2) **Max pooling : pooling**란 **convolution**에서 **Feature map** 영상의 크기를 줄이기 위한 용도이며, **max pooling**는 **Window**에서 최대값을 갖는 픽셀을 선택

(**Alex net**에서는 **3 x 3**)

(3) **Dropout : overfitting** 해결

일정한 **mini batch** 구간 동안 생략된 망에 대한 학습을 끝내면 다시 무작위로 다른 뉴런들을 생략하면서 반복적인 학습 수행



(a) Standard Neural Net

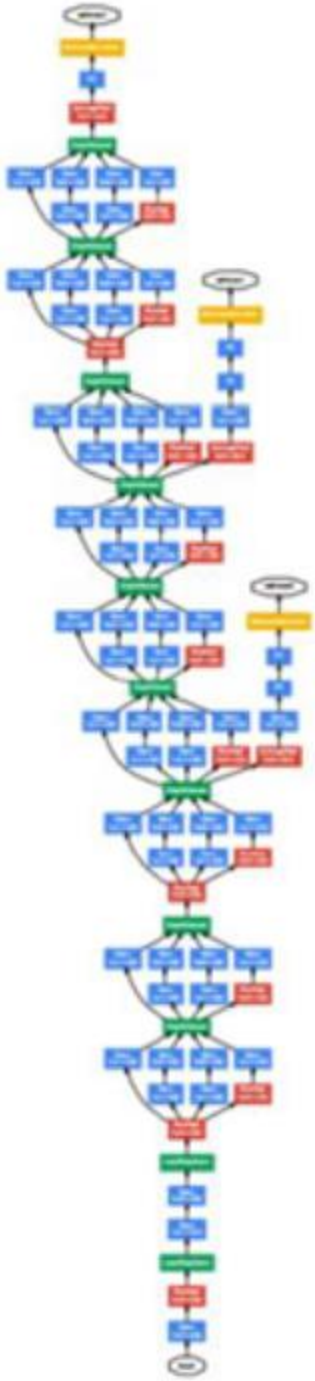
(b) After applying dropout.

- Google net 이란?
- ILSVRC 2014 우승한 CNN 알고리즘

- 핵심 : Networks are deeper and deeper !!!
- 2013년도 까지는 10 layers 미만 이었지만, Google net은 22 layers

- ** 망 증가 시 free parameter의 수 , 연산 량 증가 → overfitting 문제 발생
- 따라서, 문제 해결을 위한 Inception model 적용

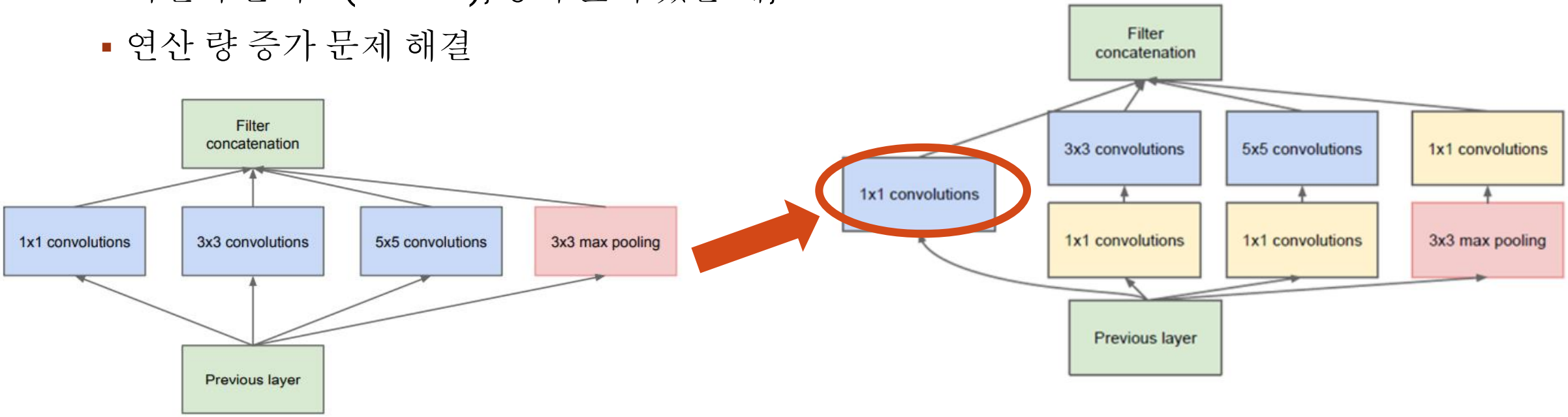
C. Szegedy et al, "Going Deeper with Convolutions" (CVPR 2015)



INCEPTION MODEL

- 같은 **layer**에 서로 다른 크기를 갖는 **convolution filter**을 적용하여, 다른 **scale**의 **feature**을 획득

- **1 x 1 convolution**을 사용하여 차원(**dimension**)을
- 적절히 줄이고(**reduce**), 망이 깊어 졌을 때,
- 연산 량 증가 문제 해결



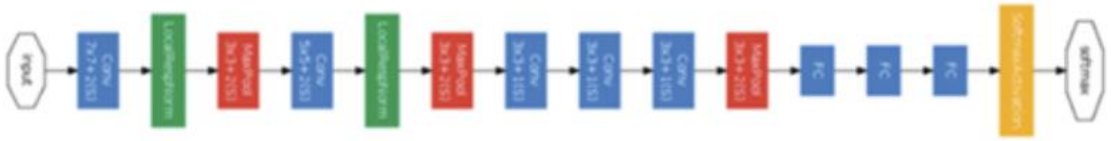
22 LAYER

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

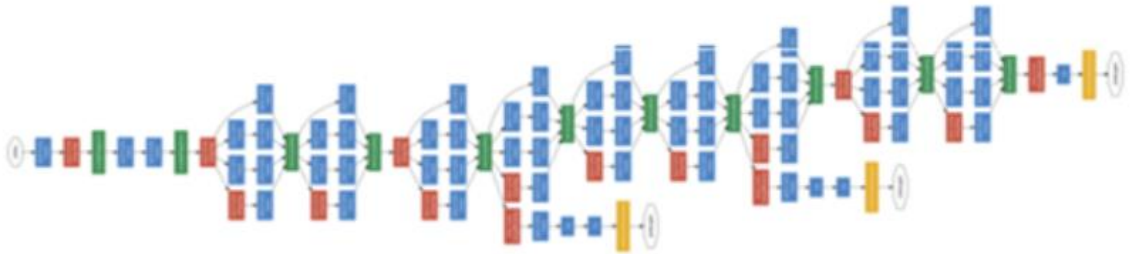
Table 1: GoogLeNet incarnation of the Inception architecture.

ALEX NET 과 GOOGLE NET의 파라미터(PARAMS)의 수와 연산량(FLOP'S) 비교

	params	*FLOPs*
Krizhevsky, Suskever and Hinton (2012)	60M	2B



Szegedy et al (2014)	5M	1.5B



ALEX NET , GOOGLE NET APPLICATION

- Alex net 검증 (validation set : cats 976 images , dogs 976 images)

23x1 Layer array with layers:

1	'input'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'fc7'	Fully Connected	4096 fully connected layer
20	'relu7'	ReLU	ReLU
21	'fc8'	Fully Connected	1000 fully connected layer
22	'prob'	Softmax	softmax
23	'classificationLayer'	Classification Output	crossentropyex with 'n01440764', 'n01443537', and 998 other classes

ans =

2×2 table

Label	Count
cats	976
dogs	976

confMat =

0.9268	0.0732
0.1098	0.8902

%	cats	dogs
cats	92.68	7.32
dogs	10.98	89.02

- **Google net 검증 (validation set : cats 77 images , dogs 77 images)**

mini batch size : 10
cross validation : 3
validation split : 0.7 , 0.3

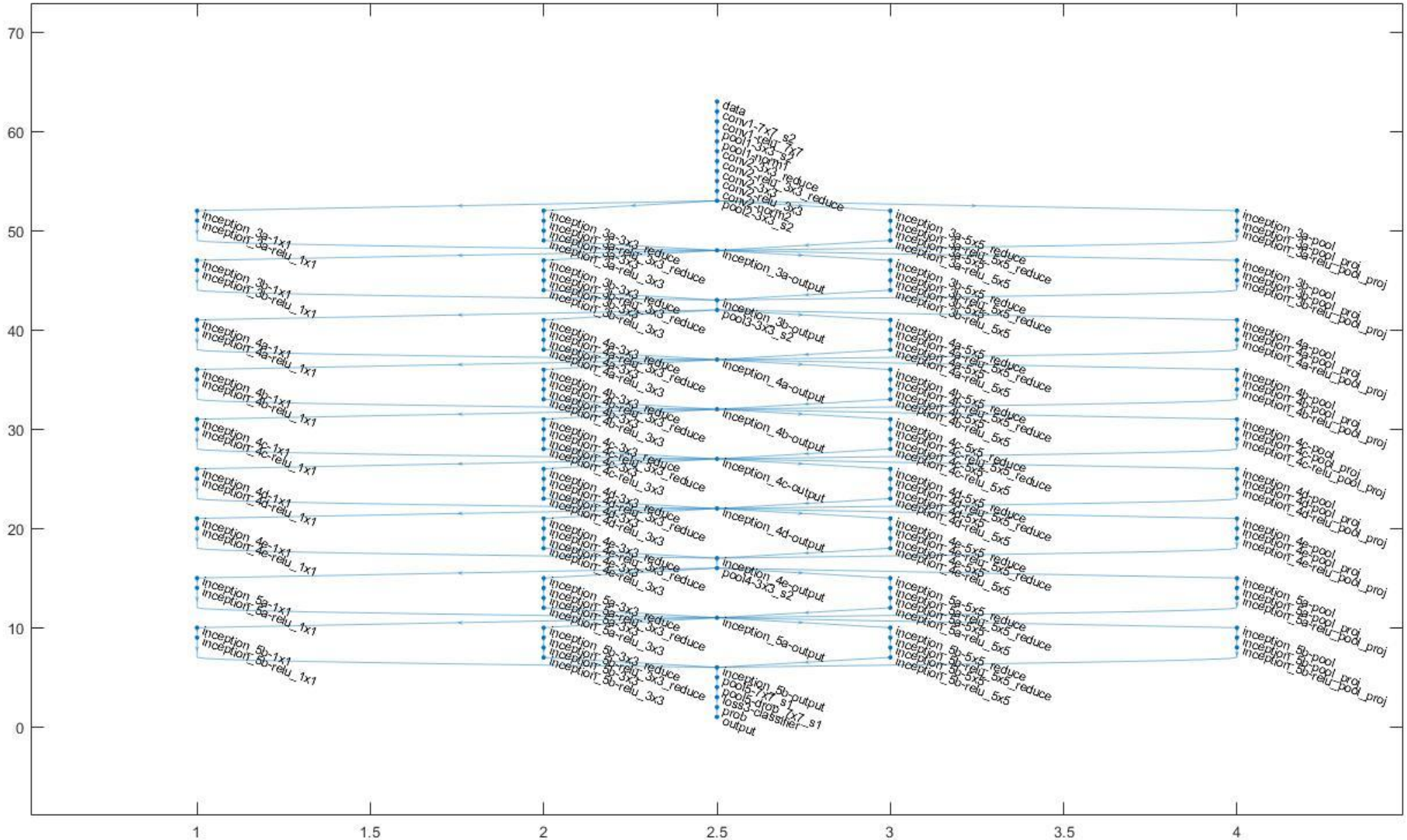
Layers: [144×1 nnet.cnn.layer.Layer]
Connections: [170×2 table]

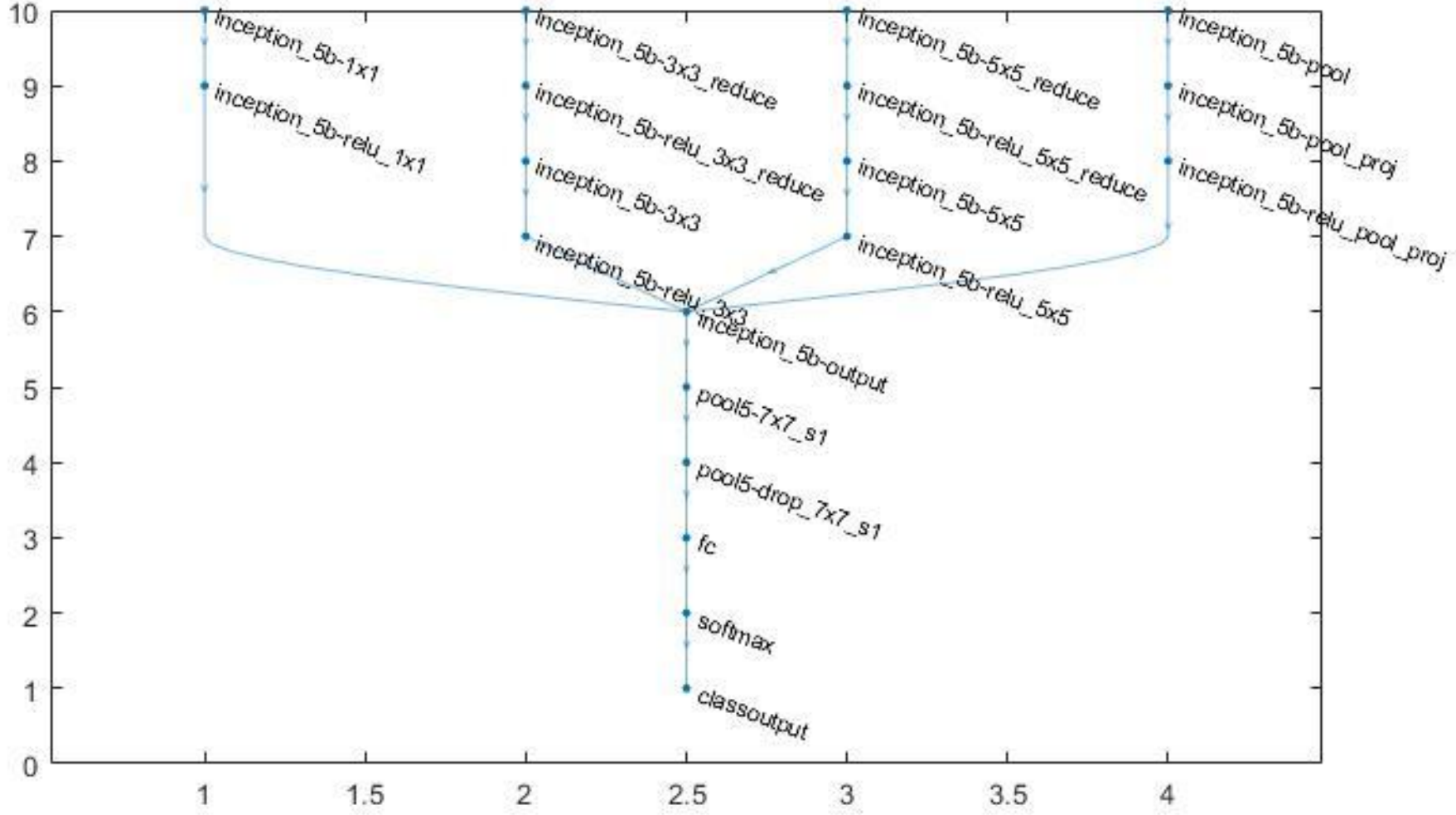
Training on single GPU.
Initializing image normalization.

accuracy =

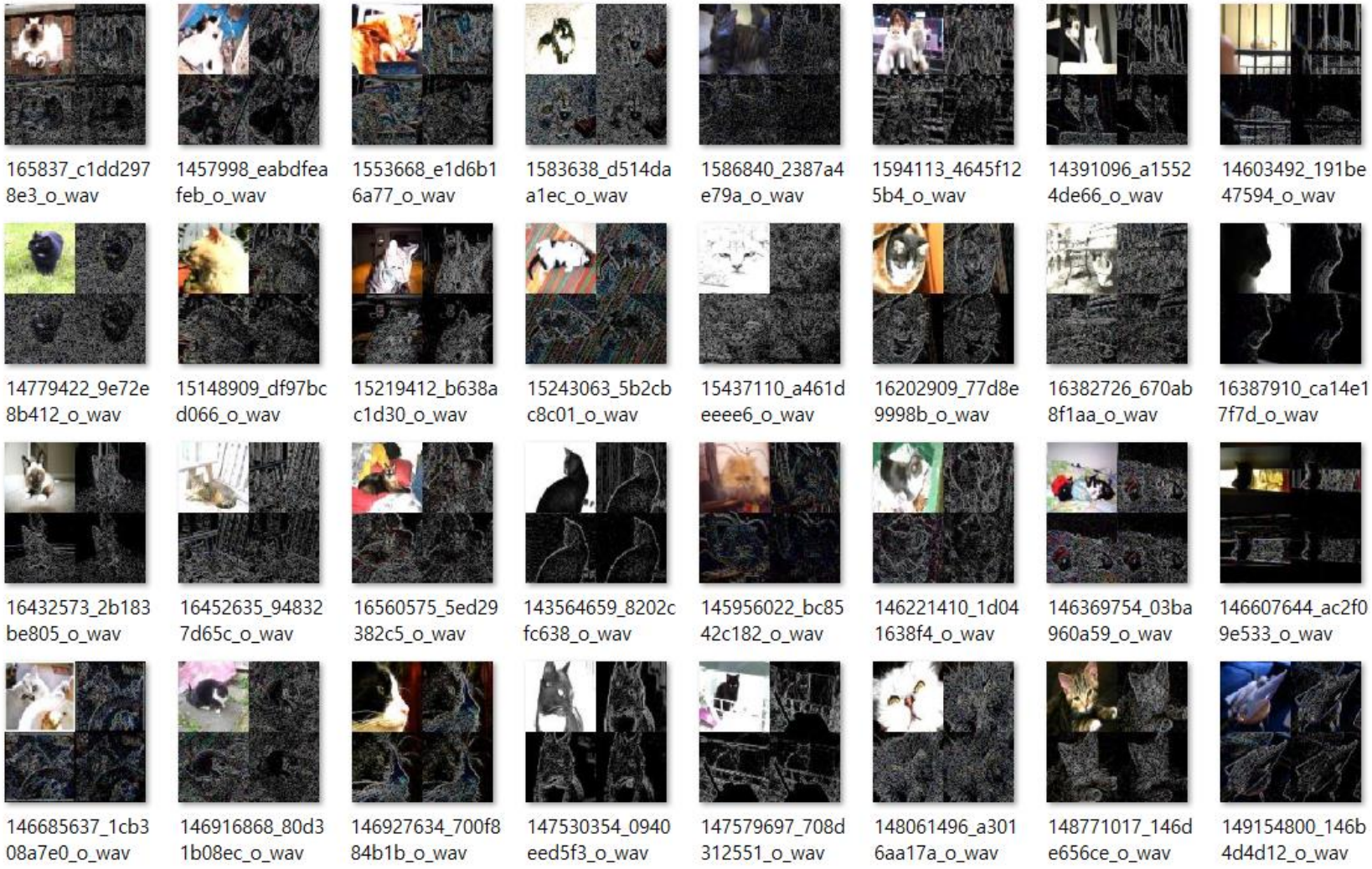
0.9348

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Validation Loss	Mini-batch Accuracy	Validation Accuracy	Base Learning Rate
1	1	0.43	0.6682	0.5906	60.00%	65.22%	1.00e-04
1	2	2.39	0.7177		40.00%		1.00e-04
1	3	3.08	0.4452	0.5106	80.00%	78.26%	1.00e-04
1	4	5.68	0.6661		60.00%		1.00e-04
1	5	6.00	0.9095		60.00%		1.00e-04
1	6	6.29	0.5651	0.2347	60.00%	93.48%	1.00e-04
1	7	8.09	0.2109		90.00%		1.00e-04
1	8	8.42	0.3843		80.00%		1.00e-04
1	9	8.75	0.1850	0.2542	100.00%	93.48%	1.00e-04
1	10	10.13	0.0413		100.00%		1.00e-04
2	11	10.43	0.3277		90.00%		1.00e-04
2	12	10.72	0.2489	0.1380	90.00%	93.48%	1.00e-04
2	13	11.82	0.1378		90.00%		1.00e-04
2	14	12.11	0.0070		100.00%		1.00e-04
2	15	12.39	0.1494	0.1026	90.00%	93.48%	1.00e-04
2	16	13.58	0.0633		100.00%		1.00e-04
2	17	13.88	0.0345		100.00%		1.00e-04
2	18	14.17	0.0098	0.0956	100.00%	93.48%	1.00e-04
2	19	15.08	0.0344		100.00%		1.00e-04
2	20	15.41	0.0010		100.00%		1.00e-04
3	21	15.71	0.0760	0.0925	100.00%	93.48%	1.00e-04
3	22	16.89	0.0149		100.00%		1.00e-04
3	23	17.16	0.0557		100.00%		1.00e-04
3	24	17.46	0.0047	0.0905	100.00%	93.48%	1.00e-04
3	25	18.46	0.1053		90.00%		1.00e-04
3	26	18.74	0.0140		100.00%		1.00e-04
3	27	19.03	0.0090	0.0889	100.00%	93.48%	1.00e-04
3	28	19.89	0.0048		100.00%		1.00e-04
3	29	20.18	0.0342		100.00%		1.00e-04
3	30	20.47	0.0009	0.0904	100.00%	93.48%	1.00e-04


70
60
50
40
30
20
10
0
1
1.5
2
2.5
3
3.5
4
data
conv1-7x7_s2
conv1-relu_7x7
pool1-3x3_s2
pool1-norm1
conv2-3x3_reduce
conv2-relu_3x3_reduce
conv2-3x3
conv2-relu_3x3
conv2-norm2
pool2-3x3_s2
inception_3a-output
pool3-3x3_s2
inception_4a-output
inception_4b-output
inception_4c-output
inception_4d-output
inception_4e-output
pool4-3x3_s2
inception_5a-output
inception_5b-output
pool5-7x7_s1
loss3-drop_7x7_s1
loss3-classifier
prob
output


inception_5b-1x1
inception_5b-relu_1x1
inception_5b-3x3_reduce
inception_5b-relu_3x3_reduce
inception_5b-3x3
inception_5b-relu_3x3
inception_5b-5x5_reduce
inception_5b-relu_5x5_reduce
inception_5b-5x5
inception_5b-relu_5x5
inception_5b-pool
inception_5b-pool_proj
inception_5b-relu_pool_proj
inception_5b-output
pool5-7x7_s1
pool5-drop_7x7_s1
fc
softmax
classoutput
10
9
8
7
6
5
4
3
2
1
0
1
1.5
2
2.5
3
3.5
4

DWT(DISCRETE WAVELET TRANSFORM) CATS ,DOGS)



1499022_5bead74eca_o_wav

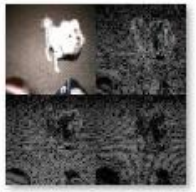
14324892_59a7fb7926_o_wav

15332687_a1d7f735f4_o_wav

15520587_dffd1cb12a_o_wav

15531917_701eaa6a9d_o_wav

15679347_7920ce9d76_o_wav


15729566_bf6027d097_o_wav

15869719_fdf750f41b_o_wav

143345262_7decd87211_o_wav

143345264_8dafe641a3_o_wav

143357497_ab41a83a73_o_wav

144600279_d85b1e8e68_o_wav

144824399_fc442d576f_o_wav

145894818_539d924acc_o_wav

146567642_81530c461e_o_wav

146809396_da3e913ad4_o_wav

146838310_e825f0be2c_o_wav

147686126_34034a5bd0_o_wav

148013534_748c5039c9_o_wav

148818579_e6f3f86992_o_wav

148818772_bf7f6fa398_o_wav

148830727_3b51a9c051_o_wav

149455812_8dd6b1f1e0_o_wav

149798019_28a8223ab9_o_wav

149894383_996aade99a_o_wav

149895119_64d97d08e2_o_wav

149964964_7df42190db_o_wav

150071395_6d7bebbd90_o_wav

150135258_50d159b2a1_o_wav

150367532_c58958bf99_o_wav

150794181_f82749b66a_o_wav

151469509_90842fa09c_o_wav

- DWT(Discrete Wavelet Transform) validation set(cats : 77 images , dogs : 77 images)

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Validation Loss	Mini-batch Accuracy	Validation Accuracy	Base Learning Rate
1	1	3.75	0.6794	0.7729	60.00%	52.17%	1.00e-04
1	2	6.18	0.7394		60.00%		1.00e-04
1	3	6.88	0.8574	0.7031	30.00%	56.52%	1.00e-04
1	4	9.55	0.8722		40.00%		1.00e-04
1	5	9.92	0.9498		50.00%		1.00e-04
1	6	10.22	0.4789	0.5751	70.00%	65.22%	1.00e-04
1	7	11.79	0.4361		80.00%		1.00e-04
1	8	12.10	0.8257		60.00%		1.00e-04
1	9	12.41	0.5685	0.5274	80.00%	71.74%	1.00e-04
1	10	13.66	0.5026		60.00%		1.00e-04
2	11	13.96	0.1524		100.00%		1.00e-04
2	12	14.25	0.3132	0.5333	80.00%	73.91%	1.00e-04
2	13	15.41	0.3985		80.00%		1.00e-04
2	14	15.70	0.4820		70.00%		1.00e-04
2	15	16.01	0.4234	0.5384	80.00%	73.91%	1.00e-04
2	16	17.25	0.5126		80.00%		1.00e-04
2	17	17.55	0.0912		100.00%		1.00e-04
2	18	17.88	0.7960	0.5496	80.00%	78.26%	1.00e-04
2	19	19.06	0.4647		90.00%		1.00e-04
2	20	19.37	0.2492		80.00%		1.00e-04
3	21	19.66	0.0324	0.5252	100.00%	80.43%	1.00e-04
3	22	20.49	0.2599		90.00%		1.00e-04
3	23	20.79	0.2644		90.00%		1.00e-04
3	24	21.09	0.2600	0.5385	90.00%	80.43%	1.00e-04
3	25	22.19	0.2810		90.00%		1.00e-04
3	26	22.48	0.2412		90.00%		1.00e-04
3	27	22.76	0.1104	0.5378	100.00%	76.09%	1.00e-04
3	28	23.84	0.2868		90.00%		1.00e-04
3	29	24.12	0.4389		80.00%		1.00e-04
3	30	24.41	0.1122	0.5462	100.00%	76.09%	1.00e-04

accuracy =

0.7609

GOOGLE NET 성능 향상을 위한 파라미터 조정

mini batch size : 32
cross validation : 5
validation split ; 0.8 , 0.2

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Validation Loss	Mini-batch Accuracy	Validation Accuracy	Base Learning Rate
1	1	0.90	0.8037	0.5993	53.13%	70.00%	1.00e-04
1	2	2.30	0.6441		62.50%		1.00e-04
1	3	3.28	0.5509		84.38%		1.00e-04
2	4	4.24	0.3990		81.25%		1.00e-04
2	5	5.23	0.3082	0.2285	93.75%	90.00%	1.00e-04
2	6	6.86	0.2100		96.88%		1.00e-04
3	7	7.93	0.2087		90.63%		1.00e-04
3	8	9.08	0.1065		100.00%		1.00e-04
3	9	10.11	0.1262		96.88%		1.00e-04

accuracy =

0.9333

mini batch size : 32
cross validation : 4
validation split : 0.75 , 0.25

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Validation Loss	Mini-batch Accuracy	Validation Accuracy	Base Learning Rate
1	1	1.08	0.7959	0.6687	43.75%	57.89%	1.00e-04
1	2	4.63	0.7294		56.25%		1.00e-04
1	3	5.63	0.5745		71.88%		1.00e-04
2	4	6.63	0.3609	0.2693	90.63%	94.74%	1.00e-04
2	5	8.26	0.2540		96.88%		1.00e-04
2	6	9.16	0.2788		90.63%		1.00e-04
3	7	10.09	0.1105		96.88%		1.00e-04
3	8	11.13	0.1820	0.1063	90.63%	94.74%	1.00e-04
3	9	12.62	0.2245		87.50%		1.00e-04

accuracy =

0.9474

Applied_Mathematics_Capstone_Design

-Matlab : 응용수학캡스톤디자인 전공수업 (Alexnet , Googlenet) : cats vs dogs binary classification

- CNN(Convolutional Neural Network) 사용

<프로젝트 진행 과정>

- (1) Python 으로 직접 모델링 (Contests 폴더 kaggle_cats_dogs 참조)
- (2) image augmentation , training image wavelet transform 하였으나 accuracy 가 평균 65%
- DWT_RGB.m : discrete wavelet transform code
- training set의 부족임을 알게되어 googlenet , alexnet 적용 ,
- At least, Matlab r2017b version, Using Computer Vision System Toolbox, Image Processing Toolbox, Neural Network Toolbox. Parallel Computing Toolbox, Statistics and Machine Learning Toolbox
- (3) Alexnet accuracy : cats : 87.04% , dogs : 83.33% , validation set : cats(77 images) , dogs(77 images)
- accuracy : cats : 92.68%, dogs : 89.02% , validation set : cats(976 images) , dogs(976 images)
- (4) Googlenet (I recommend using GPU better than CPU, because of time spending.)
- accuracy : 93.33% , validation set : cats(77 images) , dogs(77 images) -cross validation : 5, minibatch size : 32, validation : 0.2
- accuracy : 93.48% , validation set : cats(77 images) , dogs(77 images) -cross validation : 3, minibatch size : 10, validation : 0.3
- accuracy : 94.74% , validation set : cats(77 images) , dogs(77 images) -cross validation : 4, minibatch size : 32, validation : 0.25
- mini batch size : 64 -> GPU out of memory
- accuracy : 76.09% , DWT(Discrete Wavelet Transform) validation set : cats(77 images) , dogs(77 images)

RESULT

- (1) Python 과 Matlab 의 사용을 통한 Deep Learning 에서 알고리즘의 한 종류인 CNN 직접 모델링.
- (2) 검증된 알고리즘인 Alex net , Google net 을 이용한 적용.

- <추가적인 진행 방향>
- Google net 의 내부 알고리즘 구조를 완벽히 이해 한 후, binary classification 뿐만 아니라, 실생활에 더욱더
- 밀접한 문제 해결 능력 향상.

- < Github portfolio URL (MATLAB , PYTHON related code) >
- https://github.com/choiwb

참고 문헌

- Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift : ICML 2015
https://pdfs.semanticscholar.org/c1ba/ed41e4bc9401b1b2ec8ef55ba45543f7a1a3.pdf

- Classifying cats and dogs from images using deep learning
: university of Alberta
https://github.com/shrobon/Classifying-Dogs-and-Cats-using-CNN/blob/master/MM803_Project_Reportf.pdf

- https://www.slideshare.net/ssuser06e0c5/normalization-72539464

- Mathematical Analysis of Convolutional Neural Networks
https://github.com/ahmedmazariML/Mathematical-analysis-of-Convolutional-Neural-Networks/blob/master/Mathematical-analysis-of-Convolutional-Neural-Networks.pdf

- -Google net
- https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf
- -Alex net
- https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf
- -Alex net vs Google net
- On the Performance of GoogLeNet and AlexNet Applied to Sketches.pdf