

# Cats and Dogs Project

– CNN 모델 구축 및 문제점 파악 –

박성호 (neowizard2018@gmail.com)

# Cats and Dogs dataset



[https://storage.googleapis.com/mledu-datasets/cats\\_and\\_dogs\\_filtered.zip](https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip)

- Cats and Dogs 데이터셋은 CNN 아키텍처를 구축하고 평가하기 위한 일종의 Hello World 라고 할 수 있음.

## [project] CNN 아키텍처를 이용해서 Cats and Dogs 분석 및 문제점 파악

Colab 에서 wget 을 이용해서 /content 에 직접 다운 받음

```
!wget https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip
```

---

### [Project]

압축 파일을 풀어

[1] 디렉토리 구조 및 데이터 개수 확인

[2] train 디렉토리 이미지를 cv2.imread() 이용해서 입력이미지의 평균 shape 알아본 후에, 이미지 사이즈는 일괄적으로 (224, 224) 크기로 resize 시킴

[3] 학습데이터 x\_train, y\_train 생성

[4] test 디렉토리 이미지를 cv2.imread() 이용해서 입력이미지의 평균 shape 알아본 후에, 이미지 사이즈는 일괄적으로 (224, 224) 크기로 resize 시킴

[5] 테스트 데이터 x\_test, y\_test 생성

## [project] CNN 아키텍처를 이용해서 Cats and Dogs 분석 및 문제점 파악

[5] validation data 를 random shuffle 한 후에, validation data로 부터

x\_validation : x\_test = 5 : 5 비율로 test 데이터 생성

[6] CNN 아키텍처 및 손실함수 설계 (input\_shape=(224, 224, 3))

[7] Adam(), Adam(1e-4), Adam(2e-5) 학습율에 대해서 epochs=30,

batch\_size=16 으로 각각 학습하고 accuracy / loss trend 파악한 후

[8] 문제점 파악 및 해결방안 논의

# CIAFR10 Project

- 다양한 이미지 예측 -

박성호 (neowizard2018@gmail.com)

## [project\_4] cifar 10 분류 가능한 실제 이미지 예측

[1] project\_1 에서 저장된 80% 이상의 정확도 가진 CNN 모델 load

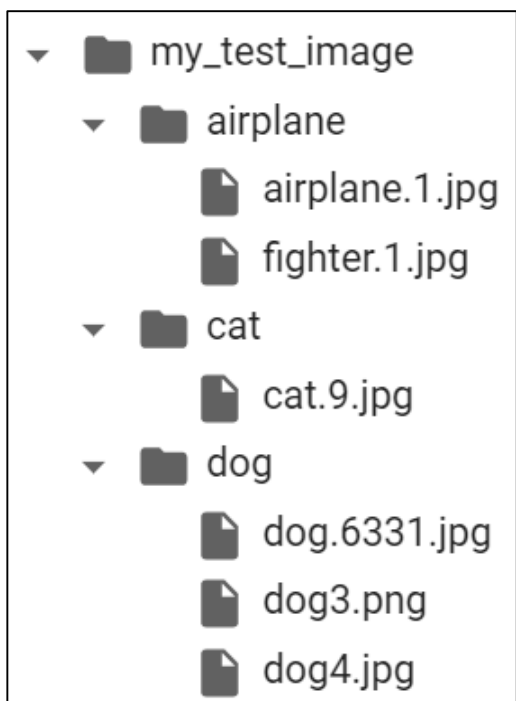
- 모델 로드 방법 예시

```
cnn = tensorflow.keras.models.load_model('./cifar10_accuracy_80.h5')  
cnn.summary()
```

---

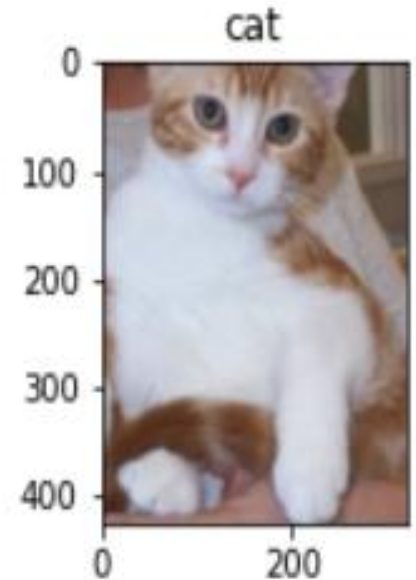
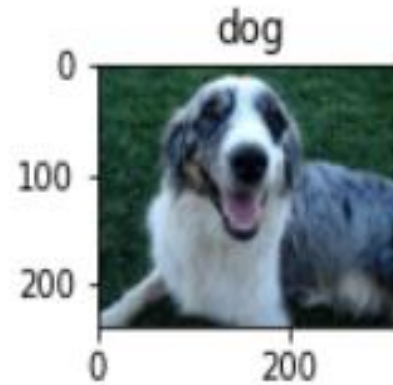
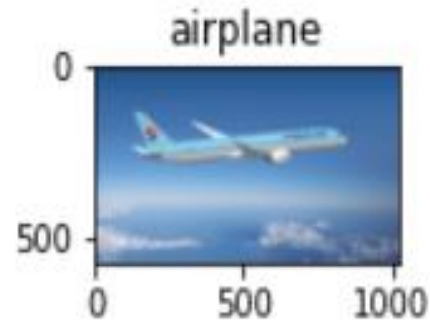
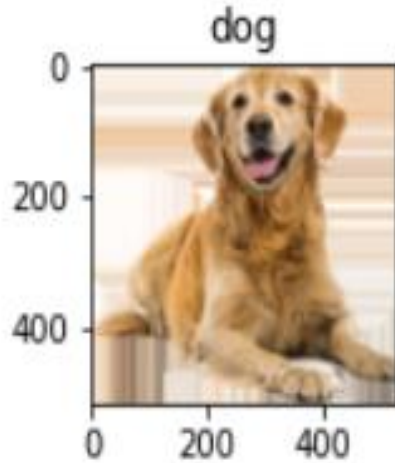
[2] 테스트 이미지 데이터 구한 후 불러오기 (정답이 디렉토리 이름이 되도록 설정함)

- 테스트에 사용될 이미지는 my\_test\_image 디렉토리의 하위 디렉토리를 만들어 저장함. 즉 정답이 dog 이미지 my\_test\_image/dog/ 디렉토리에 모두 넣고, cat 이미지 my\_test\_image/cat/ 디렉토리에 모두 넣음
- glob.glob(...) 이용하여 이미지를 읽고 정답을 자동 추출
- OpenCV 라이브러리의 imread(...) 이용하여 이미지 로드



## [project\_4] cifar 10 분류 가능한 실제 이미지 예측

### [3] 테스트 이미지 확인 (정답과 함께 이미지 출력)



## [project\_4] cifar 10 분류 가능한 실제 이미지 예측

### [4] 테스트 이미지 데이터 전처리 (resize, normalization, etc)

- cifar10 이미지 사이즈에 맞게 `cv2.resize()` 이용하여 크기 변경
  - 이미지 이므로 0~1 값을 가지도록 정규화 수행
- 

### [5] `predict()` 실행을 위한 데이터 shape 변경

- 예측을 하기위해 `cnn.predict(test_data)` 실행. 이때 입력으로 주어지는 `test_data.shape = (batch_size, width, height, channel)` 형태의 4차원 텐서로 구성
- 

### [6] `predict()` 결과를 해석하고 정확도 검증

- 예측 `pred = cnn.predict(test_data)` 실행 후에, 리턴값 `pred.shape` 확인  
가장 높은 확률의 예측 값 1 개 확인 (`np.argmax()` 및 `np.max()` 활용)  
Top 3 예측 결과와 해당 확률 확인 (`np.argsort()` 활용, 검색활용해 사용법 숙지)



# CIAFR10 Project

- 구글 포토 서비스 프로토타입 구현 -

박성호 (neowizard2018@gmail.com)

## [project\_5] cifar 10 분류 가능한 실제 이미지 예측 및 저장

[1] project\_1 에서 저장된 80% 이상의 정확도 가진 CNN 모델 load

- 모델 로드 방법 예시

```
cnn = tensorflow.keras.models.load_model('./cifar10_accuracy_80.h5')  
cnn.summary()
```

---

[2] 테스트 이미지 저장 및 확인

- 테스트에 사용될 이미지를 my\_test\_image  
디렉토리에 저장해 놓는다.



## [project\_5] cifar 10 분류 가능한 실제 이미지 예측

### [3] 테스트 이미지 데이터 전처리 (resize, normalization, etc)

- cifar10 이미지 사이즈에 맞게 `cv2.resize()` 이용하여 크기 변경
  - 이미지 이므로 0~1 값을 가지도록 정규화 수행
- 

### [4] `predict()` 실행을 위한 데이터 shape 변경

- 예측을 하기위해 `cnn.predict(test_data)` 실행. 이때 입력으로 주어지는 `test_data.shape = (batch_size, width, height, channel)` 형태의 4차원 텐서로 구성
- 

### [5] `predict()` 결과를 해석하고 정확도 검증

- 예측 `pred = cnn.predict(test_data)` 실행 후에, 리턴값 `pred.shape` 확인  
가장 높은 확률의 예측 값 1 개 확인 (`np.argmax()` 및 `np.max()` 활용)  
Top 3 예측 결과와 해당 확률 확인 (`np.argsort()` 활용, 검색활용해 사용법 숙지)

## [project\_5] cifar 10 분류 가능한 실제 이미지 예측

[6] 분류 결과를 바탕으로 해당 이미지를 11개의 디렉토리에 저장함 (구글 포토 기능 구현)

- 예측 `pred = cnn.predict(test_data)` 실행 후에, 리턴값 `pred.shape` 확인

가장 높은 확률의 예측 값 1 개 확인 (`np.argmax()` 및 `np.max()` 활용).

가장 높은 예측 확률이 50% 이상 일 때, 해당 이미지를 정답 폴더에 저장. cifar 10 은

정답이 10개 이므로 정답디렉토리 10 개 + unknown 디렉토리 1 개 = 11개 디렉토리 필요

즉 50% 확률로 이미지를 cat 으로 예측했다면 `pred_result/cat/` 디렉토리에 32x32 크기

이미지가 아닌 원본미지로 저장하고, 50% 미만의 확률로 예측된 이미지는

`pred_result/unknown/` 디렉토리에 원본으로 저장함.

이미지를 저장하기 위해서 `shutil.copy()` 또는 `shutil.move()` 사용함

## [project\_5] cifar 10 분류 가능한 실제 이미지 예측



# GTSRB Project

(참고: <https://youtu.be/gT48GRKbADI> )

박성호 (neowizard2018@gmail.com)

# GTSRB dataset



<https://sid.erd.dk/public/archives/daaec0d7ce1152aea9b61d9f1e19370/published-archive.html>

- GTSRB (German Traffic Sign Recognition Benchmark)는 독일 신경정보학 연구원들이 작성한 것으로서 교통 표지판(Traffic Sign)을 예측하기 위한 데이터이며, **평균적으로 32 x 32 크기의 작은 color 이미지로서 43개 교통 표지판과 관련된 4만여 개의 이미지를 포함함**
- GTSRB는 해당 저장소에서 GTSRB\_Final\_Training\_Images.zip' 파일임

# GTSRB dataset

## Archive Files

1 to 21 of 21 rows 25 files per page

Name	Date	Size
GTSRB-Training_fixed.zip	2019-05-10 11:26:35	187490228
GTSRB_Final_Test_GT.zip	2019-05-10 11:26:35	99620
GTSRB_Final_Test_Haar.zip	2019-05-10 11:26:35	318949368
GTSRB_Final_Test_HOG.zip	2019-05-10 11:26:35	292285317
GTSRB_Final_Test_HueHist.zip	2019-05-10 11:26:35	4798306
GTSRB_Final_Test_Images.zip	2019-05-10 11:26:35	88978620
GTSRB_Final_Training_Haar.zip	2019-05-10 11:26:35	990882445
GTSRB_Final_Training_HOG.zip	2019-05-10 11:26:35	905512002
GTSRB_Final_Training_HueHist.zip	2019-05-10 11:26:35	18175283
GTSRB_Final_Training_Images.zip	2019-05-10 11:26:35	276294756
GTSRB_Online-Test-Haar-Sorted.	2019-05-10 11:26:35	319282224
GTSRB_Online-Test-Haar.zip	2019-05-10 11:26:35	319502253
GTSRB_Online-Test-HOG-Sorted.	2019-05-10 11:26:35	295572728
GTSRB_Online-Test-HOG.zip	2019-05-10 11:26:35	296231502
GTSRB_Online-Test-HueHist-Sort	2019-05-10 11:26:35	5245948
GTSRB_Online-Test-HueHist.zip	2019-05-10 11:26:35	5465543
GTSRB_Online-Test-Images-Sorte	2019-05-10 11:26:35	88234254
GTSRB_Online-Test-Images.zip	2019-05-10 12:05:37	88610452
GTSRB_Training_Features_Haar.z	2019-05-10 11:26:35	623933894
GTSRB_Training_Features_HOG.zip	2019-05-10 11:26:35	553169013
GTSRB_Training_Features_HueHist.zip	2019-05-10 11:26:35	14399826



## [project] CNN 아키텍처를 이용해서 99% 이상으로 GTSRB 인식

Colab 에서 wget 을 이용해서 /content 에 직접 다운 받거나, 또는 zip 파일을 Google Drive 에 저장한 후에 Drive 를 Mount 시켜 /content 에 저장함 .

### [Project 1]

- ① 학습 데이터 (x\_train, y\_train), 테스트 데이터 (x\_test, y\_test) 생성하는 Data Preparation 수행
- ② 오버피팅이 거의 없는 정확도 99% 이상의 CNN 모델을 구현

참고 YouTube 강의: <https://youtu.be/gT48GRKbADI>

## [project] CNN 아키텍처를 이용해서 99% 이상으로 GTSRB 인식

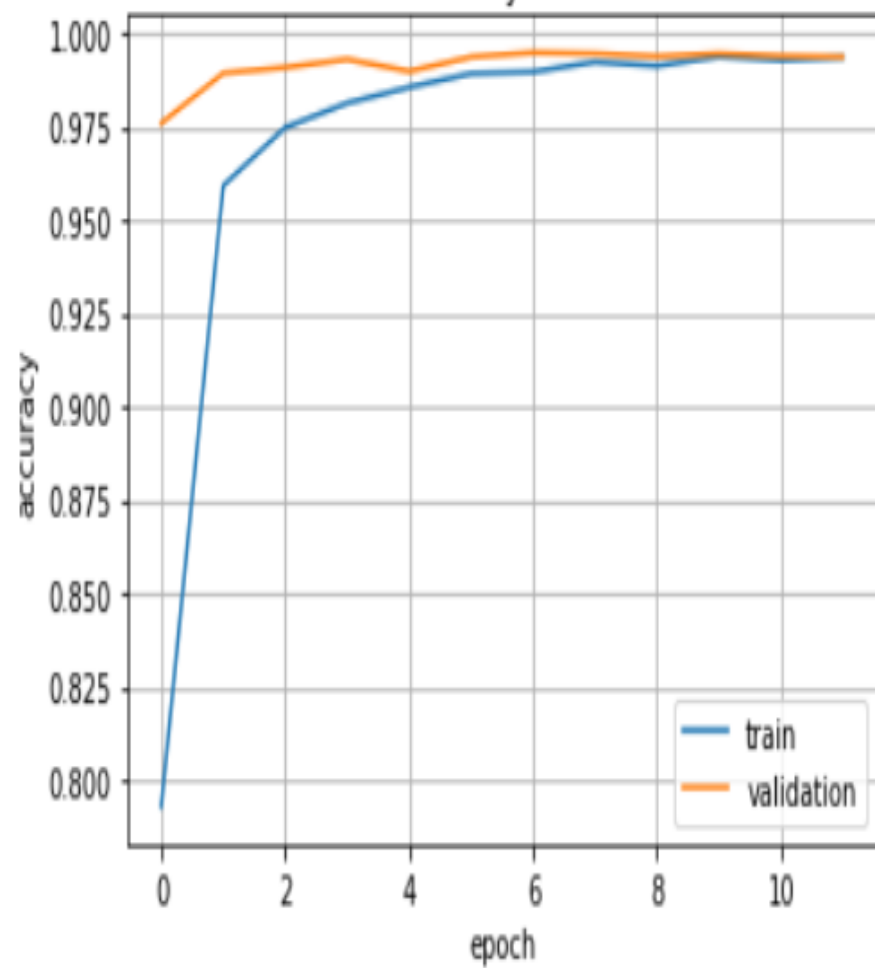
Colab 에서 wget 을 이용해서 /content 에 직접 다운 받거나, 또는 zip 파일을 Google Drive 에 저장한 후에 Drive 를 Mount 시켜 /content 에 저장함 .

### [Project 2]

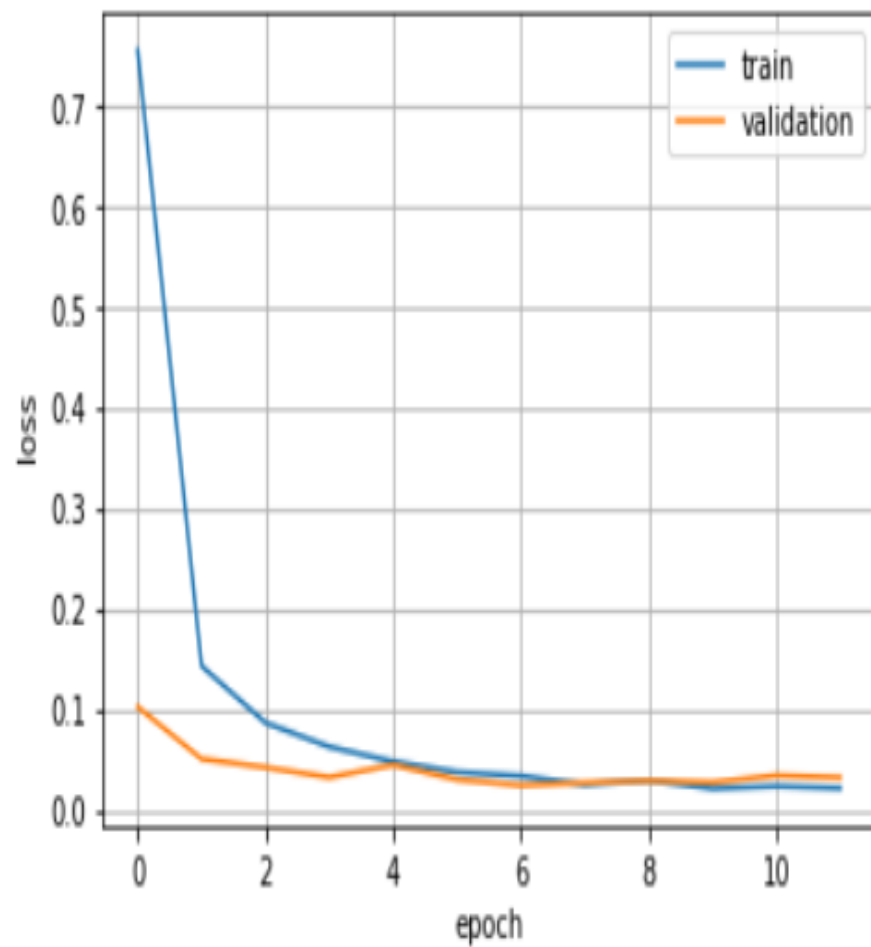
- ① 원본 이미지를 train / test 디렉토리에 별도로 저장한 후에, train / test 디렉토리를 통해서 학습과 테스트에 필요한 데이터를 생성하는 Data Preparation,
- ② 오버피팅이 거의 없는 정확도 99% 이상의 CNN 모델을 구현

참고 YouTube 강의: <https://youtu.be/gT48GRKbADI>

Accuracy Trend



Loss Trend



# ImageDataGenerator

- `flow_from_directory(..)` –

박성호 (neowizard2018@gmail.com)

# Image Data Augmentation ( 참고: <https://youtu.be/RuZ7SI-Yfec> )

- 원본 이미지에 적절한 변형을 가해서 새로운 데이터를 만들어 내는 방식으로, 다양한 데이터를 통해 딥러닝 성능향상 기대할 수 있음



원본 이미지

원본 이미지에 변형을 가해서 새롭게 생성된 이미지

- 텐서플로는 이미지 데이터 보강을 위한 ImageDataGenerator class 제공하며, ImageDataGenerator 에서는 입력 파라미터를 통해서 rescale, rotation\_range, width\_shift\_range, height\_shift\_range, shear\_range, horizontal\_flip, vertical\_flip 등의 다양한 변화를 줄 수 있음 (다양한 입력 파라미터는 TensorFlow API 문서 참고)
- ImageDataGenerator를 사용할 경우 flow(), flow\_from\_directory() 등의 함수를 통해 이미지 데이터를 보강할 수 있음.

# ImageDataGenerator flow\_from\_directory() 예제

## [1] ImageDataGenerator 생성

```
import tensorflow as tf

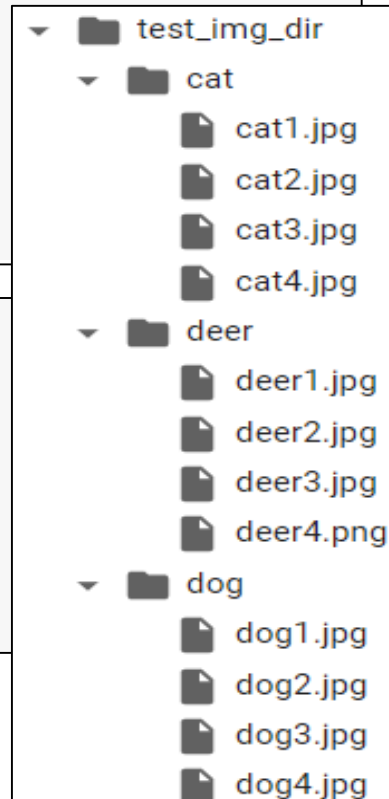
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
gen = ImageDataGenerator(rescale=1./255,
                        rotation_range=10,
                        shear_range=0.2,
                        horizontal_flip=True)
```

```
import zipfile

with zipfile.ZipFile('./test_img_dir.zip') as target_file:
    target_file.extractall('test_img_dir/')
```



# ImageDataGenerator flow\_from\_directory() 예제

## [2] flow\_from\_directory() 적용

```
data_gen = gen.flow_from_directory(directory='./test_img_dir/',  
                                   batch_size=3,  
                                   shuffle=True,  
                                   target_size=(100, 100),  
                                   class_mode='categorical')
```

디렉토리 이름에 맞춰 자동으로 labelling

test\_img\_dir

- cat : cat1.jpg / cat2.jpg
- deer : deer1.jpg / deer2.jpg
- dog : dog1.jpg / dog2.jpg

```
print(data_gen.class_indices)  
print(data_gen.num_classes)  
print(data_gen.class_mode)
```

class\_mode는 정답을 나타내는 방식

'binary', 'categorical', 'sparse'

```
Found 12 images belonging to 3 classes.  
{'cat': 0, 'deer': 1, 'dog': 2}  
3  
categorical
```

# ImageDataGenerator flow\_from\_directory() 예제

## [3] next() 실행 및 변형 이미지 출력

```
img, label = data_gen.next()

for i in range(len(label)):
    print('label => ', label[i])

plt.figure(figsize=(8,8))
for i in range(len(img)):

    plt.subplot(1, len(img), i+1)
    plt.xticks([])
    plt.yticks([])
    plt.title(str(label[i]))
    plt.imshow(img[i])

plt.show()
```

```
label => [0. 1. 0.]
label => [1. 0. 0.]
label => [0. 0. 1.]
```

정답을 십진수로 나타내기 위해서는 ?

[0. 1. 0.]



[1. 0. 0.]



[0. 0. 1.]



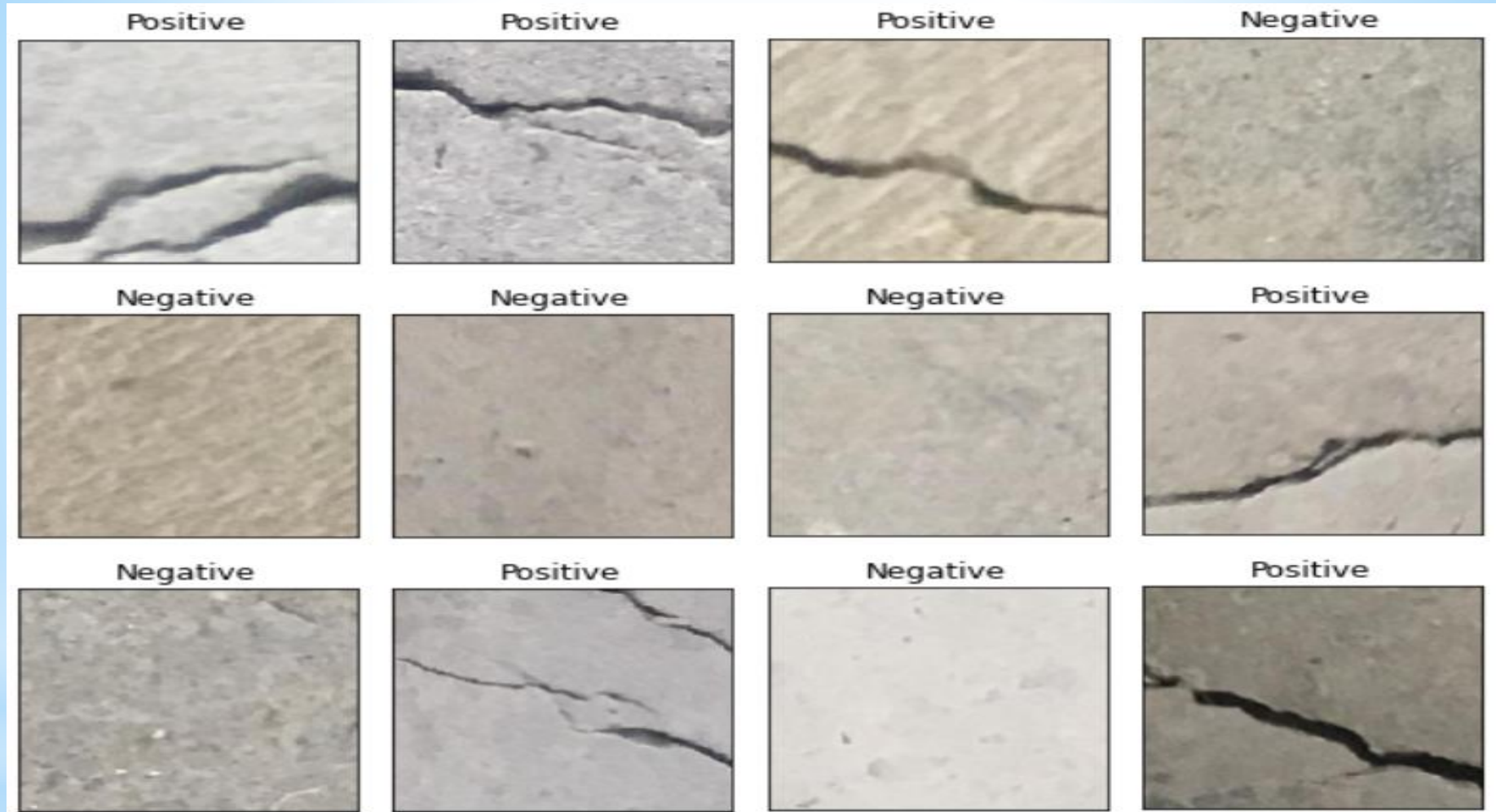


# Surface Crack Detection Project

(참고 강의: [https://youtu.be/p\\_3iJ035Q\\_Q](https://youtu.be/p_3iJ035Q_Q) )

박성호 (neowizard2018@gmail.com)

## Training Data – Kaggle Surface Crack Detection



Surface Crack Detection은 콘크리트 표면 결함(concrete surface crack)을 발견하고 예측하기 위한 Kaggle의 공개 데이터로서, 평균적으로  $227 \times 227$  크기를 가지는 color 이미지이며, crack 없는 Negative 데이터 2만개와 crack 발생한 Positive 데이터 2만개, 총 4만 개의 이미지 데이터

# Kaggle Surface Crack Detection

<https://www.kaggle.com/arunrk7/surface-crack-detection>

Dataset

Surface Crack Detection

Concrete surface sample images for Surface Crack Detection

Arun Pandian R • updated 2 years ago (Version 1)

Data

Code (46)

Discussion (4)

Activity


Metadata

Download (245 MB)

New Notebook

Your Dataset download has started.  
Show your appreciation with an upvote

143



Usability 9.4

License

Data files © Original Authors

Tags

business, computer science, classification, urban planning

Description

Surface Crack Detection Dataset

Context

Concrete surface cracks are major defect in civil structures. Building Inspection which is done for the evaluation of rigidity and tensile strength of the building. Crack detection plays a major role in the building inspection, finding the cracks and determining the building health.

Local PC 다운된 archive.zip 파일을 surface\_crack.zip 이름으로 변경 하여 Google Drive 에 저장함

## [project] CNN 아키텍처를 이용해서 99% 이상으로 정확도 달성

Google Drive 를 Mount 시켜 /content 에 surface\_crack.zip 저장함 .

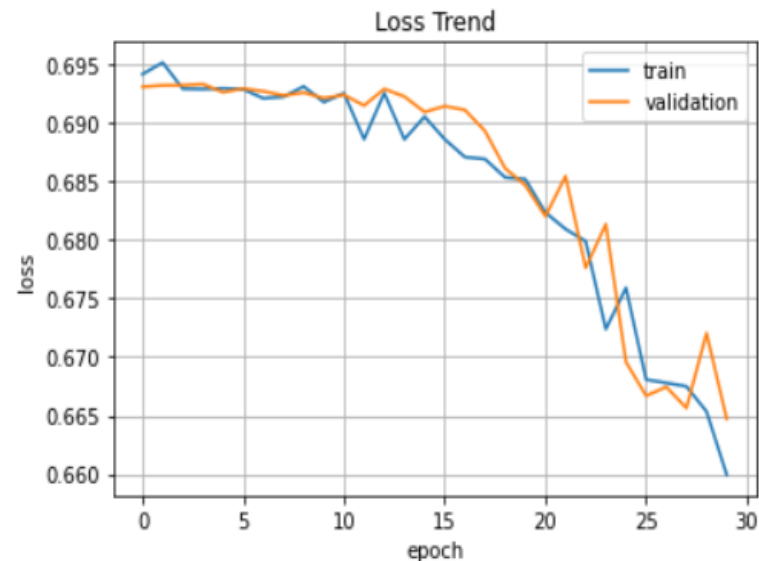
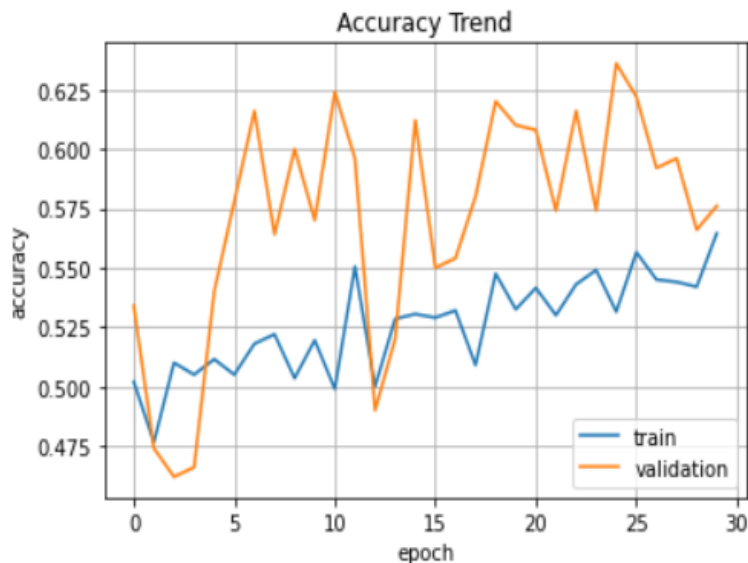
---

train / test 디렉토리를 생성하고, 원본 이미지를 train / test 디렉토리에 8 : 2 비율로 저장한 후에, train / test 디렉토리를 통해서 학습과 테스트에 필요한 데이터를 생성한 후에 학습 시켜, 오버피팅이 거의 없는 정확도 99% 이상의 CNN 모델을 구현

# Cats and Dogs Project Review

박성호 (neowizard2018@gmail.com)

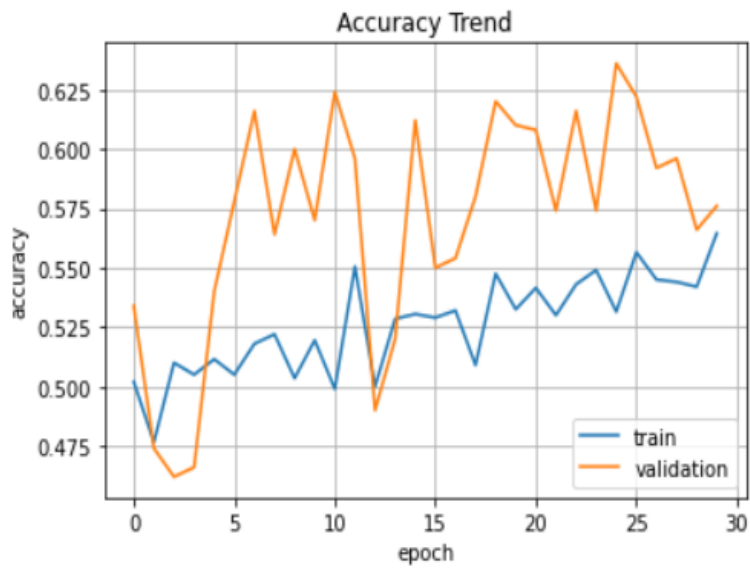
# Review – CNN Cats and Dog Project



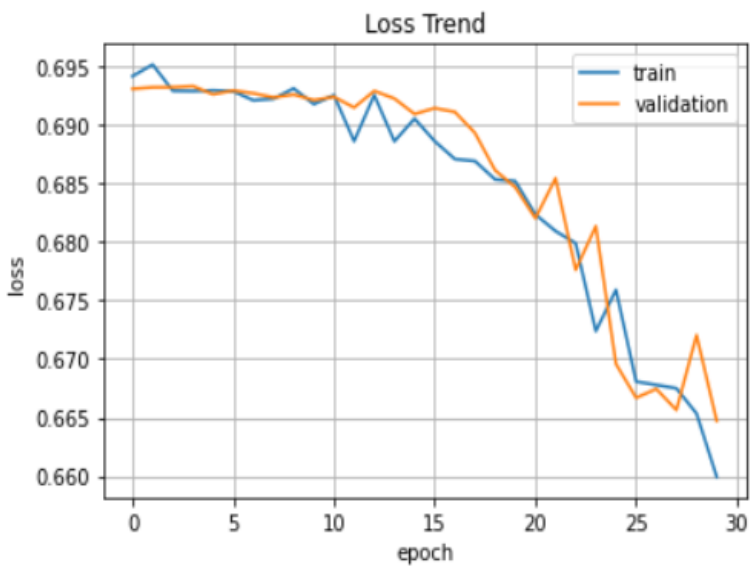
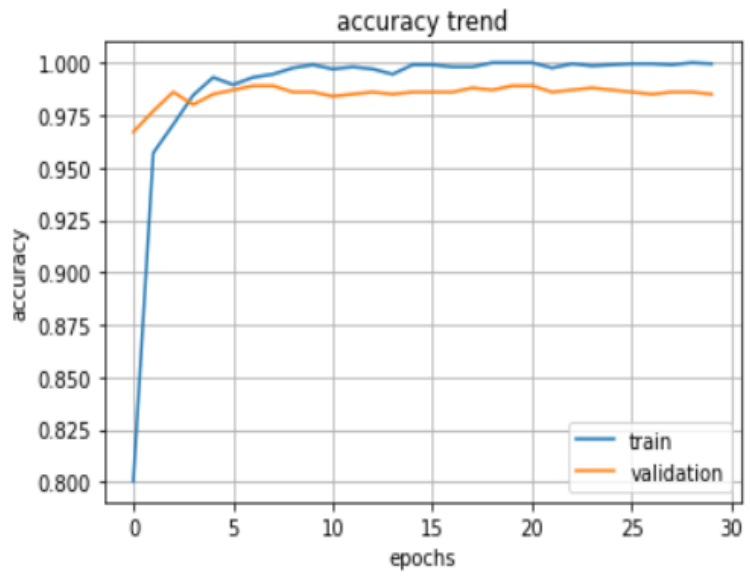
[문제점]

- (1) 정확도가 낮고 불규칙함. (60~70%)
- (2) 데이터 개수가 절대적으로 적음 (학습데이터 약 1000개)
- (3) CNN 아키텍처를 얼마나 Deep 하게 설계해야 하는지 가능하기 어려움
- (4) 매번 디렉토리 만들고 학습 데이터와 테스트 데이터 만드는 반복 작업 필수

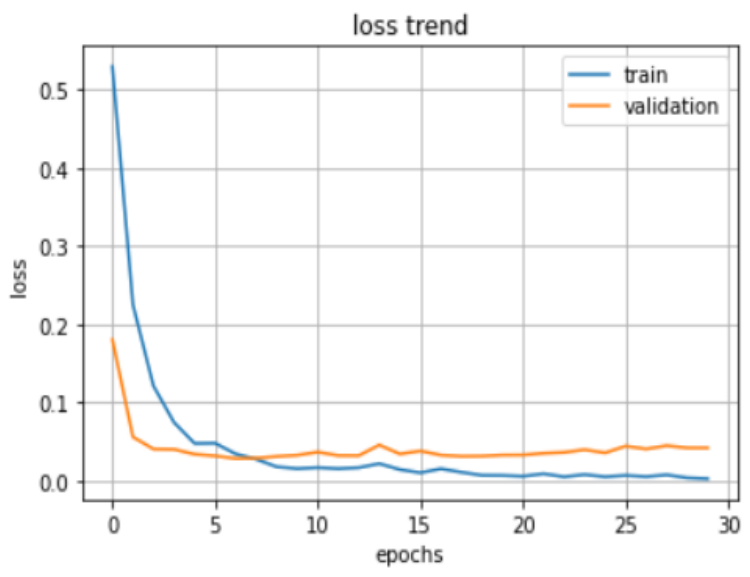
# 성능향상(정확도 ↑ · 오버피팅 ↓) 가능 ?



정확도 증가  
→



손실값 감소  
→



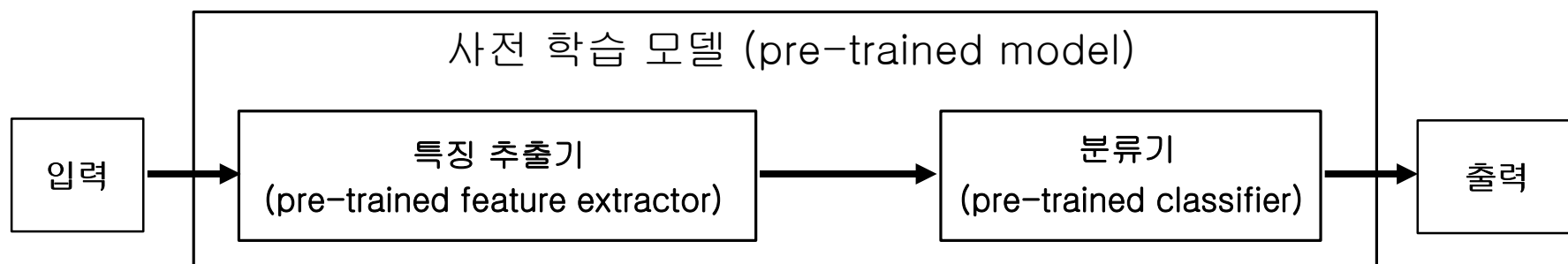
# Transfer Learning

박성호 (neowizard2018@gmail.com)



# 전이 학습 (Transfer Learning) (참고: <https://youtu.be/r4nQgQkdOqM> )

- CNN 기반의 딥러닝 모델을 훈련시키려면 많은 양의 데이터가 필요하지만 큰 데이터셋을 얻는 것은 쉽지 않음. 이러한 현실적인 어려움을 해결한 것이 전이 학습인데, **전이 학습은 ImageNet처럼 아주 큰 데이터셋을 써서 사전 학습 모델(pre-trained model)의 가중치를 가져와 우리가 분석하려는 데이터에 맞게 보정해서 사용하는 것을 의미함**

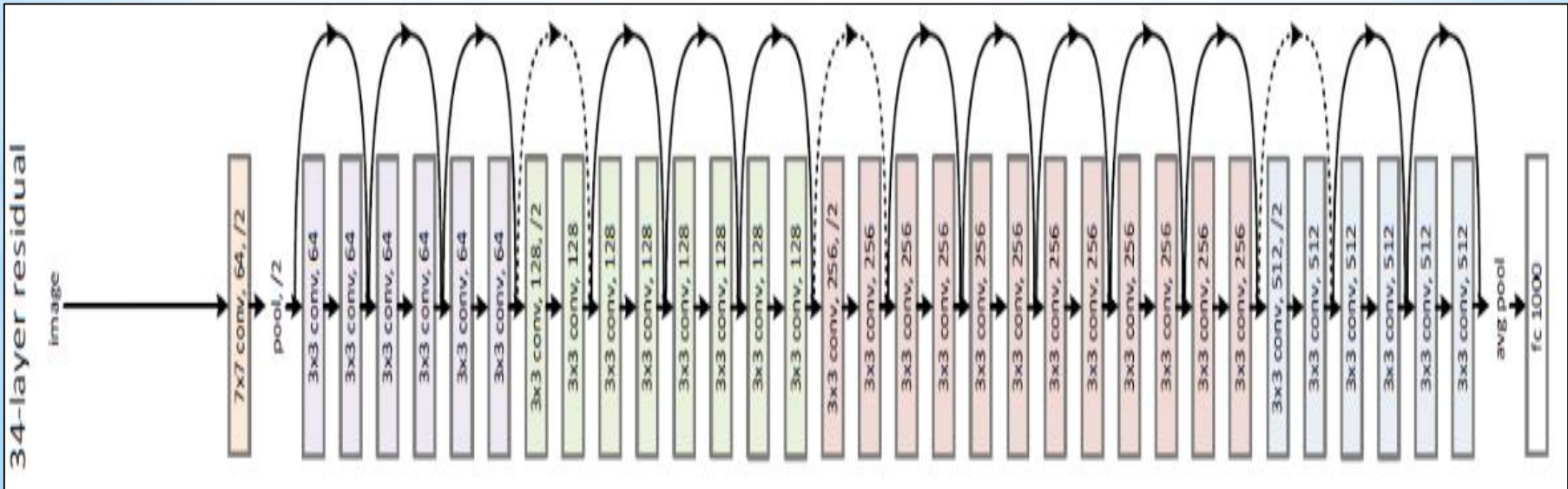
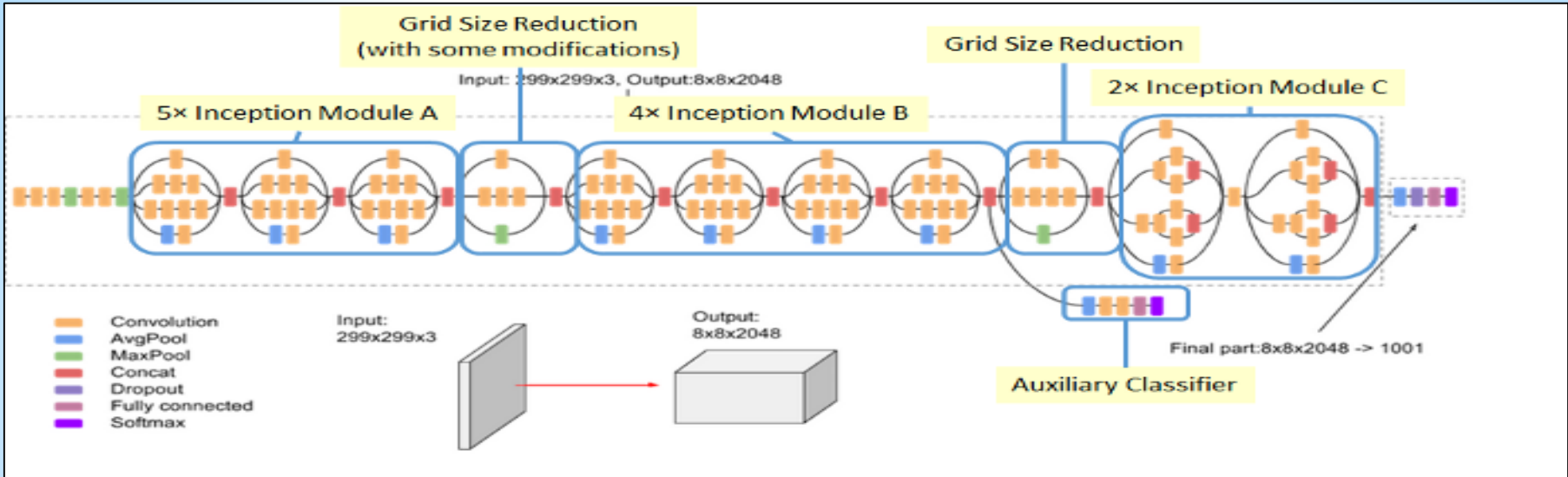


- 특징 추출기는 컨볼루션층과 풀링 층의 조합으로 구성되어 있으며 ImageNet 데이터에 대해 이미 학습되어 있음
- 분류기는 완전 연결 층(Dense) 조합으로 구성되며 이미지에 대한 정답을 분류하는 역할

# 전이학습 - TensorFlow 사전 학습 모델 (pre-trained model) 종류

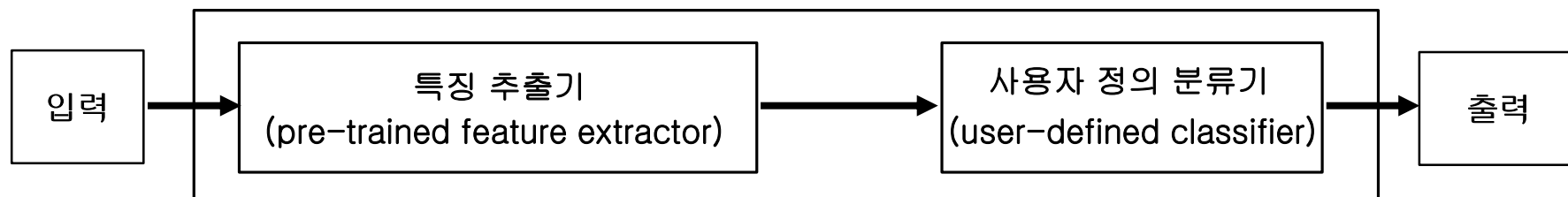
모델 이름	파일 크기	1순위 정확률	5순위 정확률	매개변수 개수	층의 개수(깊이)
Xception	88MB	0.790	0.945	22,910,480	126
VGG16	528MB	0.713	0.901	138,357,544	23
VGG19	549MB	0.713	0.900	143,667,240	26
ResNet50	98MB	0.749	0.921	25,636,712	—
ResNet101	171MB	0.764	0.928	44,707,176	—
ResNet152	232MB	0.766	0.931	60,419,944	—
ResNet50V2	98MB	0.760	0.930	25,613,800	—
ResNet101V2	171MB	0.772	0.938	44,675,560	—
ResNet152V2	232MB	0.780	0.942	60,380,648	—
InceptionV3	92MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215MB	0.803	0.953	55,873,736	572
MobileNet	16MB	0.704	0.895	4,253,864	88
MobileNetV2	14MB	0.713	0.901	3,538,984	88
DenseNet121	33MB	0.750	0.923	8,062,504	121
DenseNet169	57MB	0.762	0.932	14,307,880	169
DenseNet201	80MB	0.773	0.936	20,242,984	201
NASNetMobile	23MB	0.744	0.919	5,326,716	—
NASNetLarge	343MB	0.825	0.960	88,949,818	—

# pre-trained model (InceptionV3, MS ResNet50,...)



## 전이학습 - 파인튜닝 (fine tuning)

- 파인 튜닝은 분석하려는 새로운 데이터에 잘 맞도록 ① 사전 학습 모델의 가중치 일부를 재 학습 시키거나 또는 ② 모든 가중치를 처음부터 다시 학습시키는 방법. 즉 파인 튜닝은 새롭게 분석하려는 데이터에 맞게 모델 가중치를 조정하는 기법이며, GPU 사용이 필수임



# Transfer Learning 사용법

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.applications import VGG16, ResNet50, MobileNet, InceptionV3
```

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(240,240,3))
```

사전학습에 사용된 데이터셋

include\_top=False 사전학습 모델의 특징 추출기만 가져옴

include\_top=True 사전학습 모델의 특징추출기와 분류기 모두 가져옴

새롭게 학습시킬  
이미지 텐서 크기

# base\_model.trainable = False 설정하면, 파인튜닝 없음

```
model = Sequential()
```

```
model.add(base_model)
```

```
model.add(Flatten())
```

```
model.add(Dense(64, activation='relu'))
```

```
model.add(Dropout(0.25))
```

```
model.add(Dense(4, activation='softmax'))
```

새로운 분류기  
(user-defined classifier)

```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(2e-5),
              metrics=['accuracy'])
```

```
model.summary()
```

## [예제]

제공되는 TransferLearning\_Exercise\_1 코드는 CNN 강의에서 진행한 Cats and Dogs Project 코드에서, 모델 구축과 학습 부분을 제외한 코드임.

즉 train / test 디렉토리로 부터 학습 데이터 파일 (x\_train, y\_train), (x\_test, y\_test), (x\_val, y\_val) 을 생성하고 정규화 하는 코드만 있음

제공되는 TransferLearning\_Exercise\_1 코드에, 사전학습 모델과 사용자 정의 분류기를 추가하여 학습 한 후에, 정확도와 손실함수 추세가 CNN 프로젝트 결과와 어떻게 다른지 확인하시오

(사전 학습 모델은 MobileNet, ResNet50, InceptionV3, Xception 에 대해서 각각 코드를 구현하고, 어떤 차이점이 있는지 정확도와 손실함수 관점에서 논의하시오)

## [예제] GTSRB 데이터를 이용한 파인 튜닝

CNN 강의에서 진행한 GTSRB Project 코드에서, Transfer Learning 을 이용하여 다음 조건에 맞도록 구현하시오.

[1] train 과 test 디렉토리를 생성하고, train : test = 8 : 2 비율로 분리

[2] test data 를 validation data 로 사용

[3] 모델 아키텍처는 다음과 같이 MobileNet-Flatten-Dense(128, activation='relu')-Dropout(0.5)-Dense(43, activation='softmax') 으로 구성할 때, EarlyStopping 조건하에서 learning rate 을  $1e-2$ ,  $1e-3$ ,  $1e-4$ ,  $2e-5$  등으로 변경하면서 각각의 모델 성능을 확인 하시오