인공지능

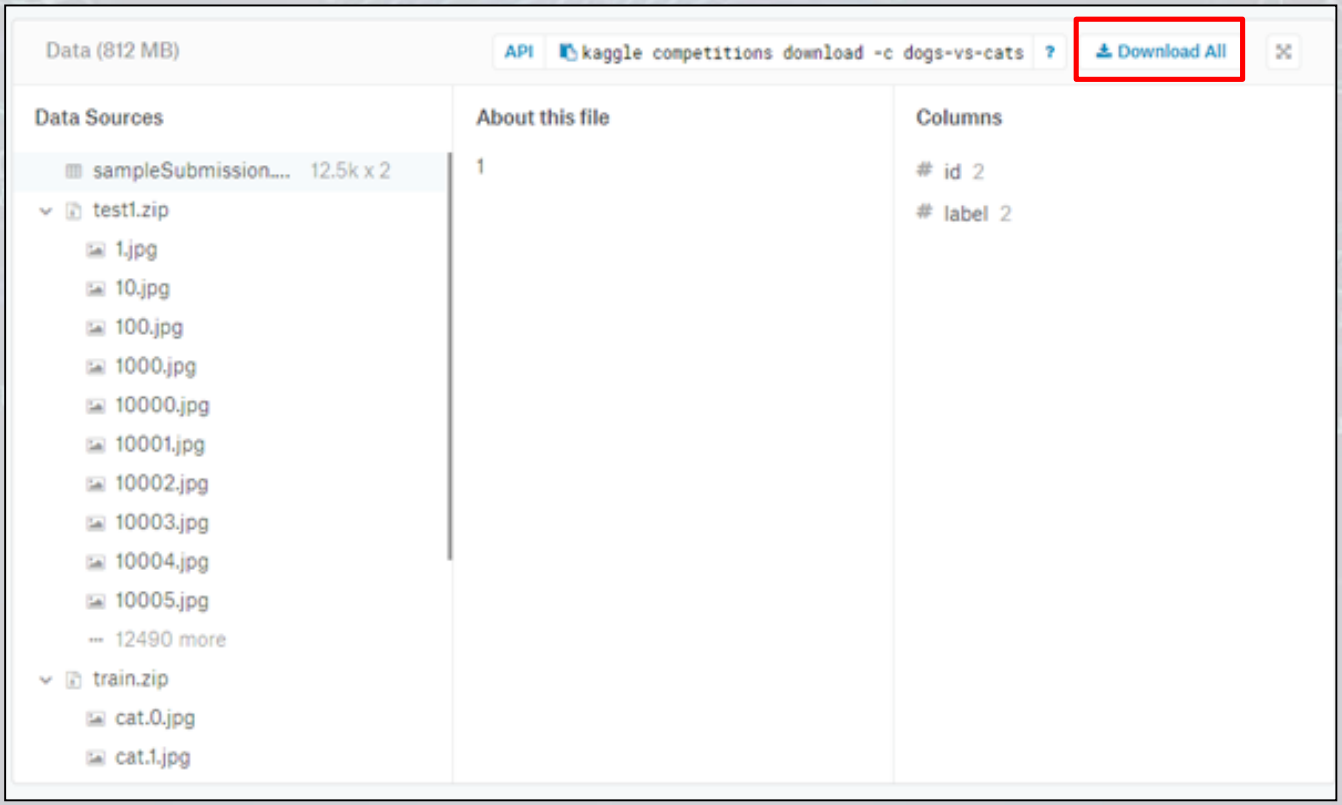
딥러닝 중급_CNN 응용

강성관 (silicon1@hanmail.net)

CNN 응용

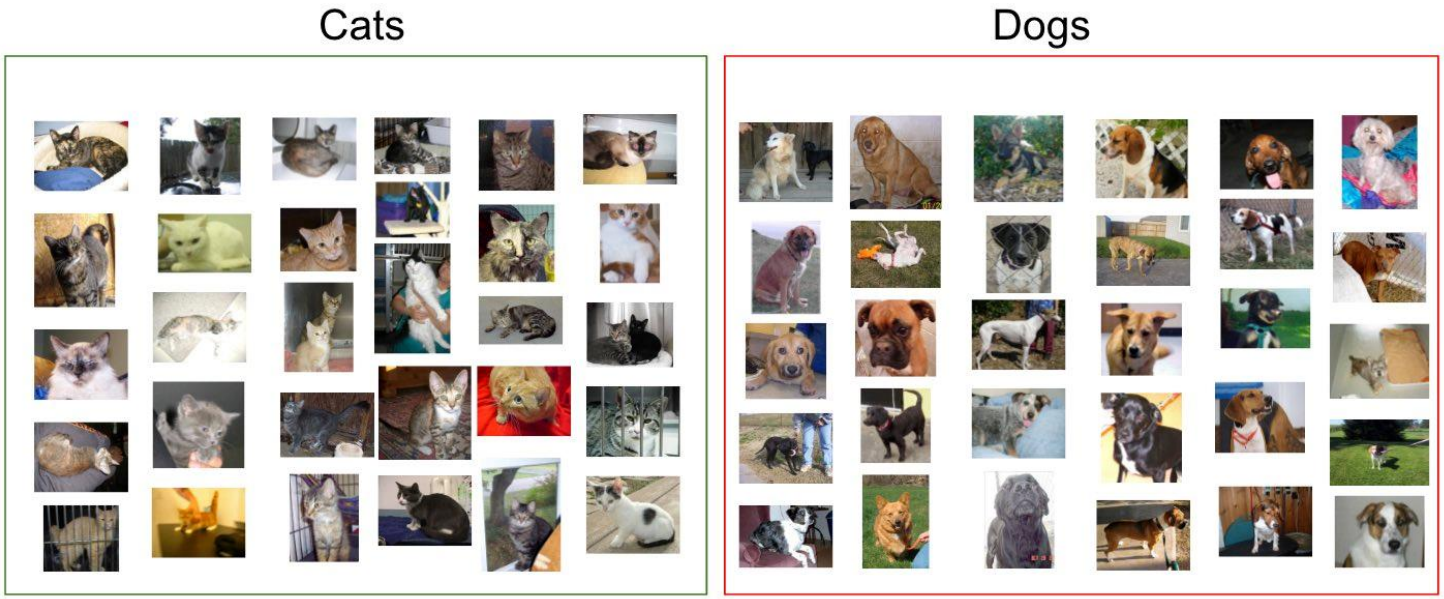
CNN – 개와 고양이 예측

- 데이터셋 다운로드 : https://www.kaggle.com/c/dogs-vs-cats/data



CNN – 개와 고양이 예측

- 압축을 풀면 훈련셋(train.zip)과 테스트셋(test.zip)이 있으며 train.zip에는 25,000개의 크기가 다른 개와 고양이 이미지 데이터가 포함되어 있음.
- 클래스(train, test)마다 12,500개의 이미지가 포함된 543MB 크기의 데이터 셋



Sample of cats & dogs images from Kaggle Dataset

CNN – 개와 고양이 예측

- 라이브러리 설치

  - PIL 이미지를 사용하기 위한 Pillow 라이브러리 설치

```
activate env01
pip install pillow
```

CNN – 개와 고양이 예측

- 이미지 데이터 분리

  - 각 폴더의 이미지 수 출력

```
65  print("훈련용 고양이 이미지 수 : ", len(os.listdir(train_cats_dir)))
66  print("검증용 고양이 이미지 수 : ", len(os.listdir(validation_cats_dir)))
67  print("테스트용 고양이 이미지 수 : ", len(os.listdir(test_cats_dir)))
68  print("훈련용 개 이미지 수 : ", len(os.listdir(train_dogs_dir)))
69  print("검증용 개 이미지 수 : ", len(os.listdir(validation_dogs_dir)))
70  print("테스트용 개 이미지 수 : ", len(os.listdir(test_dogs_dir)))
```

```
훈련용 고양이 이미지 수 :  1000
검증용 고양이 이미지 수 :  500
테스트용 고양이 이미지 수 :  500
훈련용 개 이미지 수 :  1000
검증용 개 이미지 수 :  500
테스트용 개 이미지 수 :  500
```

CNN – 개와 고양이 예측

- 신경망 구성

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513

Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0

CNN – 개와 고양이 예측

- 데이터 전처리
  - 사진 파일 로딩 → JPG 파일을 RGB 파일로 디코딩 → 부동 소수점 타입의 텐서로 변환 → 픽셀값을 0~255에서 0.0~1.0 범위로 스케일 조정
  - **keras.preprocessing.image** 라이브러리의 **ImageDataGenerator** 클래스 활용
  - **ImageDataGenerator** : 폴더에 있는 이미지 파일을 전처리된 배치 텐서로 자동으로 바꾸어주는 파이썬 제너레이터를 만들어 줌.

CNN – 개와 고양이 예측

- 데이터 전처리

```
3   from keras.preprocessing.image import ImageDataGenerator
4
5   train_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩train"
6   validation_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩validation"
7
8   train_datagen = ImageDataGenerator(rescale = 1./255)
9   test_datagen = ImageDataGenerator(rescale= 1./255)
10
11  train_generator = train_datagen.flow_from_directory(train_dir, target_size = (150, 150),
12      batch_size=20, class_mode = "binary")
13
14  validation_generator = test_datagen.flow_from_directory(validation_dir, target_size = (150, 150),
15      batch_size=20, class_mode = "binary")
```

- [8-9] 이미지의 스케일을 조정 (0~255 → 0.0~1.0)
- [11-15] (데이터폴더, 변경할 데이터크기, 배치크기, 모드)
  - 이미지 크기를 150x150으로 변경
  - 20개의 샘플을 하나의 배치로 설정
  - 모드는 binary_crossentropy를 사용하기 때문에 이진 레이블이 필요

CNN – 개와 고양이 예측

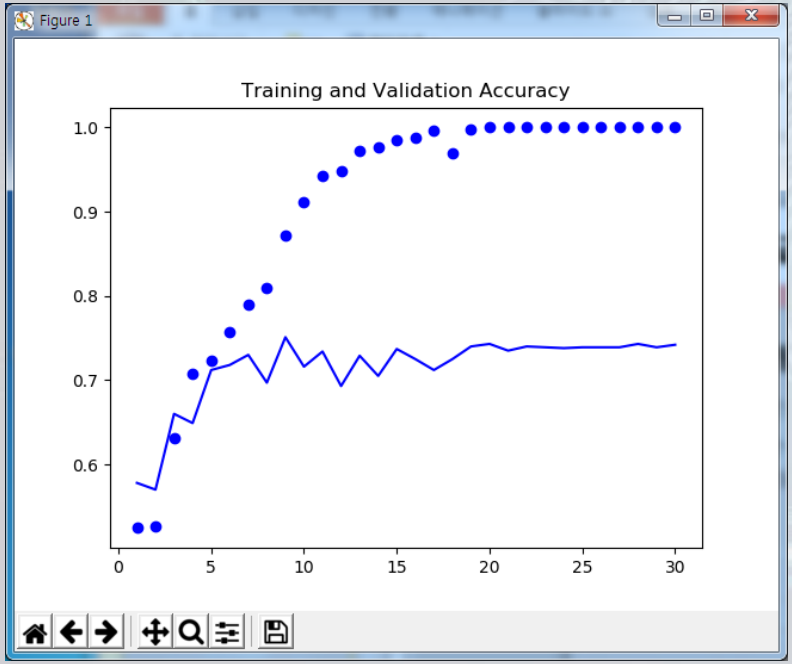
- 훈련하기

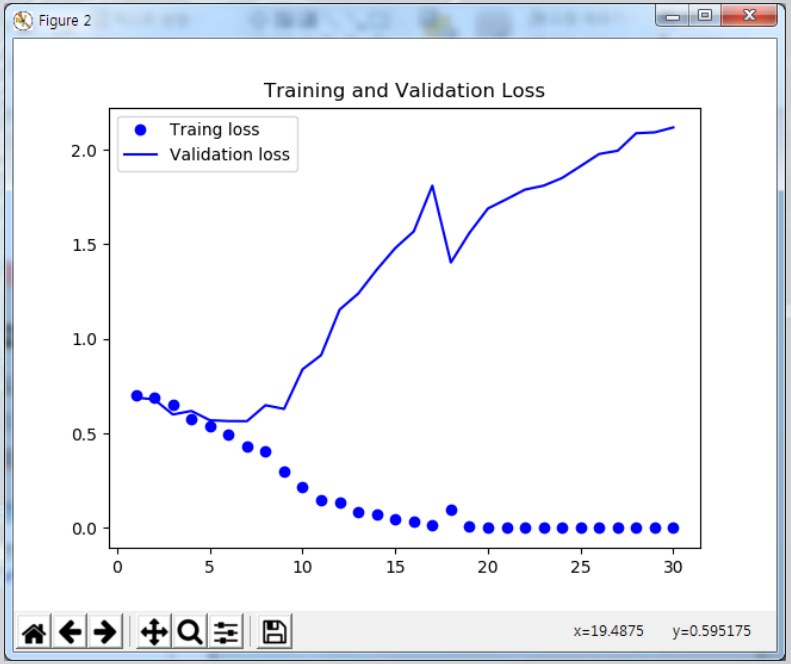
```
31  history = model.fit_generator(train_generator, steps_per_epoch=100, epochs=30,
32      verbose=0, validation_data=validation_generator, validation_steps=50)
33
34  model.save("dogs_vs_cats_small_01.h5")
```

- [31] fit_generator()는 fit() 함수와 동일한 기능을 하고 데이터 제너레이터를 사용할 수 있음
- 제너레이터에 의해 데이터가 끊임없이 생성되므로 제너레이터로부터 얼마나 많은 샘플을 뽑을 것인지 설정해야 함 → steps_per_epoch (에포크 당 스텝 수 또는 경사하강법 단계 수) 설정
- 20개의 샘플이 하나의 배치이므로 2000개의 샘플을 모두 처리하려면 100개의 배치를 뽑아야 함
- 검증 데이터도 동일한 동작을 수행 (50개의 스탭 수)
- [34] 훈련이 끝나면 모델을 저장한다

CNN – 개와 고양이 예측

- 훈련 결과 출력 (정확도, 손실오차)
  - 훈련정확도는 시간이 지남에 따라 증가해서 100%에 도달하지만 검증 정확도는 70~72% 수준
  - 검증 손실은 5번 에포크에서 최소가 된 후 개선이 없고, 훈련 손실은 0%까지 계속 감소





CNN – 개와 고양이 예측

- 학습된 모델로 개와 고양이 예측하기

```
1  from keras.preprocessing.image import ImageDataGenerator
2  from keras.models import load_model
3  import numpy as np
4
5  test_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩test"
6  test_datagen = ImageDataGenerator(rescale= 1./255)
7
8  test_generator = test_datagen.flow_from_directory(
9          test_dir, target_size=(150, 150), batch_size=20, class_mode='binary')
10
11 model = load_model("dogs_vs_cats_small_02.h5")
12
13 Y_prediction = model.predict_generator(test_generator, steps=5)
14 prediction = np.where(Y_prediction > .5, "개", "고양이")
15
16 for i in range(20):
17   print(test_generator.filenames[i])
18   print(prediction[i])
```

CNN – 개와 고양이 예측

- 데이터 증식
  - 데이터가 200개로 적기 때문에 과대 적합 가장 중요한 문제 → Dropout이나 L2 규제 등을 사용하여 감소
  - 이미지를 다루는 경우 → 데이터 증식 방법 사용
  - **데이터 증식** : 기존 훈련 샘플로 부터 더 많은 훈련 데이터를 생성하는 방법으로 원 이미지를 여러 가지 변환을 적용하여 샘플을 늘리는 방법 → 같은 모델이 2번 만나는 것을 방지하는 것이 목표

CNN – 개와 고양이 예측

- 데이터 증식

```
dagagen = ImageDataGenerator (
    rotation_range = 40,
    width_shift_range = 0.2,
    height_shift_range = 0,2,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    vertical_flip = True,
    fill_mode = "nearrest"
)
```

- rotation_range : 회전 각도 범위
- width_shift_range / height_shift_range : 수평, 수직 이동 비율 범위
- shear_range : 전단 변환할 각도 범위
- zoom_range : 확대 비율 범위
- horizontal_flip / vertical_flip : 수평/수직 대칭으로 뒤집기 여부
- fill_mode : 이미지 변환에 따라 새롭게 생성된 픽셀의 채우기 방법 (nearest, bicubic 등)

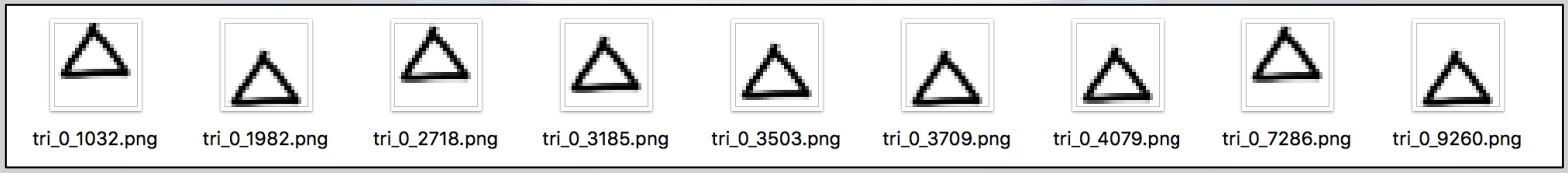
CNN – 개와 고양이 예측

- 데이터 증식

- rotation_range = 90

- width_shift_range = 0.1

- height_shift_range = 0.1



CNN – 개와 고양이 예측

- 데이터 증식

- shear_range = 0.5

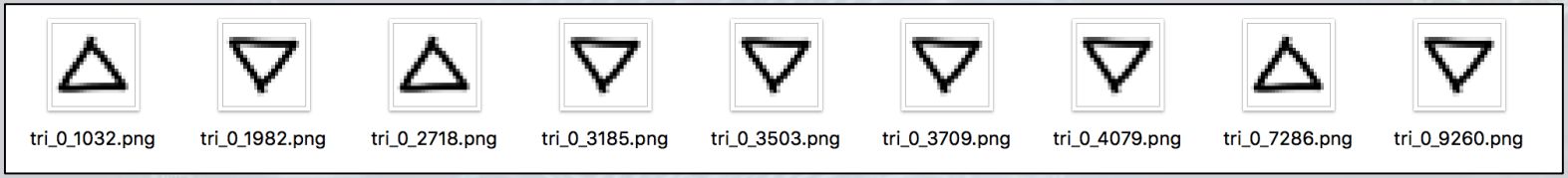
- zoom_range = 0.3

- horizontal_flip = True

CNN – 개와 고양이 예측

- 데이터 증식

- vertical_flip = True



CNN – 개와 고양이 예측

- 데이터 증식

```
1  import os
2  import matplotlib.pyplot as plt
3  from keras.preprocessing import image
4  from keras.preprocessing.image import ImageDataGenerator
5
6  train_cats_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩train₩₩cats"
7
8  fnames = sorted([os.path.join(train_cats_dir, fname)
9                   for fname in os.listdir(train_cats_dir)])
10
11 img_path = fnames[3]
12
13 img = image.load_img(img_path, target_size = (150, 150))
14 x = image.img_to_array(img)
15 x = x.reshape((1,) + x.shape)
16
```

CNN – 개와 고양이 예측

- 데이터 증식

```
17  train_datagen = ImageDataGenerator(
18        rescale = 1./255,
19        rotation_range = 40,
20        width_shift_range = 0.2,
21        height_shift_range = 0.2,
22        shear_range = 0.2,
23        zoom_range = 0.2,
24        horizontal_flip = True)
25
26  i = 0
27  for batch in train_datagen.flow(x, batch_size = 1):
28      plt.figure(i)
29      imgplot = plt.imshow(image.array_to_img(batch[0]))
30      i += 1
31      if i % 4 == 0:
32          break
33  plt.show()
```

CNN – 개와 고양이 예측

- 데이터 증식

```
 8  fnames = sorted([os.path.join(train_cats_dir, fname)
 9                   for fname in os.listdir(train_cats_dir)])
10
11  img_path = fnames[3]
12
13  img = image.load_img(img_path, target_size = (150, 150))
14
15  x = image.img_to_array(img)
```

- [8-9] train_cats_dir 폴더의 파일명을 가져와서 train_cats_dir + 파일명 문자열을 만들고 정렬
- [11] 증식에 사용할 이미지를 선택
- [13] 이미를 읽어서 150x150 크기로 변경
- [15] 이미지를 (150, 150, 3) 크기의 넘파이 배열로 변환

CNN – 개와 고양이 예측

- 데이터 증식

```
17  train_datagen = ImageDataGenerator(
...  ...
27  for batch in train_datagen.flow(x, batch_size = 1):
28      plt.figure(i)
29      imgplot = plt.imshow(image.array_to_img(batch[0]))
30      i += 1
31      if i % 4 == 0:
32          break
```

- [17] 데이터 증식 처리
- [27] 랜덤하게 변환된 이미지 배치를 생성
- [29] 넘파이 배열을 이미지로 변환
- [31-32] 4개의 이미지까지만 실행

CNN – 개와 고양이 예측

- 데이터 증식

CNN – 개와 고양이 예측

- 데이터 증식과 Dropout

```
1  from keras import layers
2  from keras import models
3  from keras.preprocessing.image import ImageDataGenerator
4
5  train_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩train"
6  validation_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩validation"
7
8  train_datagen = ImageDataGenerator(
9      rescale = 1./255,
10     rotation_range = 40,
11     width_shift_range = 0.2,
12     height_shift_range = 0.2,
13     shear_range = 0.2,
14     zoom_range = 0.2,
15     horizontal_flip = True)
16 #검증 데이터는 증식을 하면 안되므로 그대로 사용
17 test_datagen = ImageDataGenerator(rescale= 1./255)
```

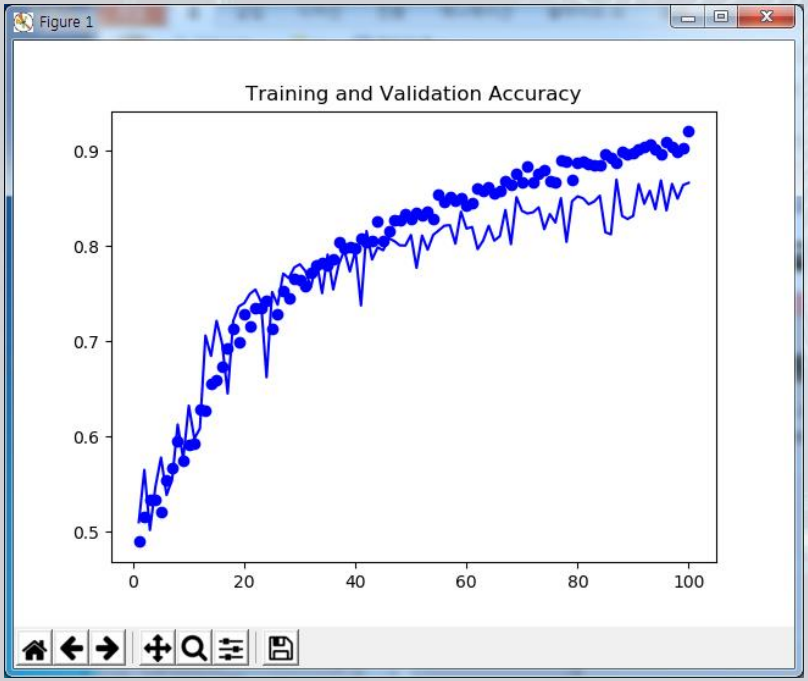
CNN – 개와 고양이 예측

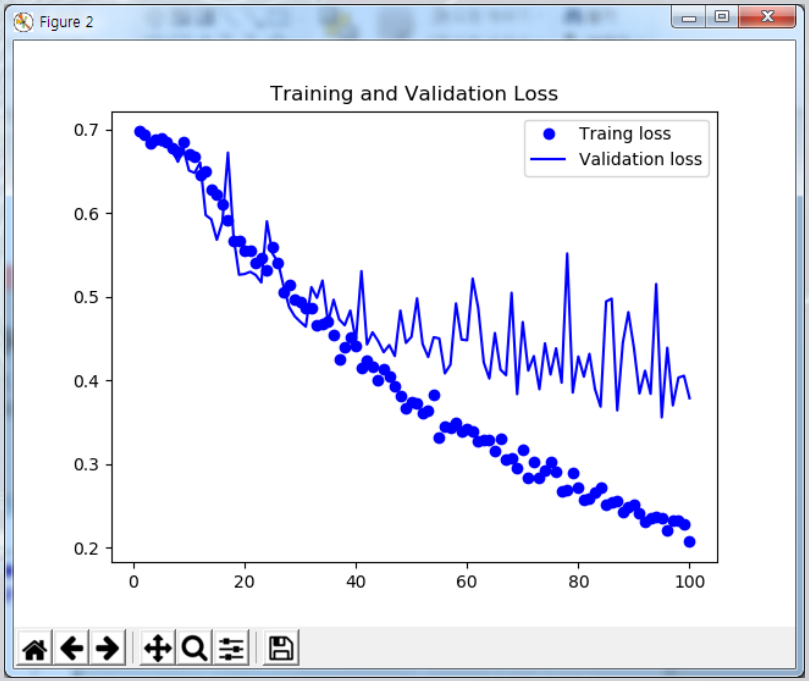
- 데이터 증식과 Dropout : 기존의 정보만을 재조합 함으로 인해 발생할 수 있는 과적합 방지를 위해 사용

```
18
19  train_generator = train_datagen.flow_from_directory(train_dir, target_size
20      = (150, 150), batch_size=32, class_mode = "binary")
21
22  validation_generator = test_datagen.flow_from_directory(validation_dir,
23      target_size = (150, 150), batch_size=32, class_mode = "binary")
...  ...
34  model.add(layers.Flatten())
35  model.add(layers.Dropout(0.5))
36  model.add(layers.Dense(512, activation="relu"))
37  model.add(layers.Dense(1, activation="sigmoid"))
38  model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["acc"])
39
40  history = model.fit_generator(train_generator, steps_per_epoch=100, epochs=100,
41       verbose=0, validation_data=validation_generator, validation_steps=50)
42
43  model.save("dogs_vs_cats_small_02.h5")
...  ...
```

CNN – 개와 고양이 예측

- 실행하기
  - 검증정확도가 기존의 70~72% 수준에서 86%까지 올라가고 검증 손실도 줄어들고 있음
  - 데이터가 적은 CNN 이므로 정확도를 높이는 것은 한계가 있음





CNN – 개와 고양이 예측

- 사전 훈련된 네트워크 사용하기
  - 사전 훈련된 네트워크 : 대규모 이미지 분류 문제를 위해 대량의 데이터셋에서 미리 훈련되어 저장된 네트워크
  - **ImageNet** : 1,400만 개의 레이블된 이미지와 1,000 개의 클래스로 구성된 네트워크
  - **VGG16** : 2014년에 개발된 ImageNet 데이터셋에 널리 사용되는 CNN 구조

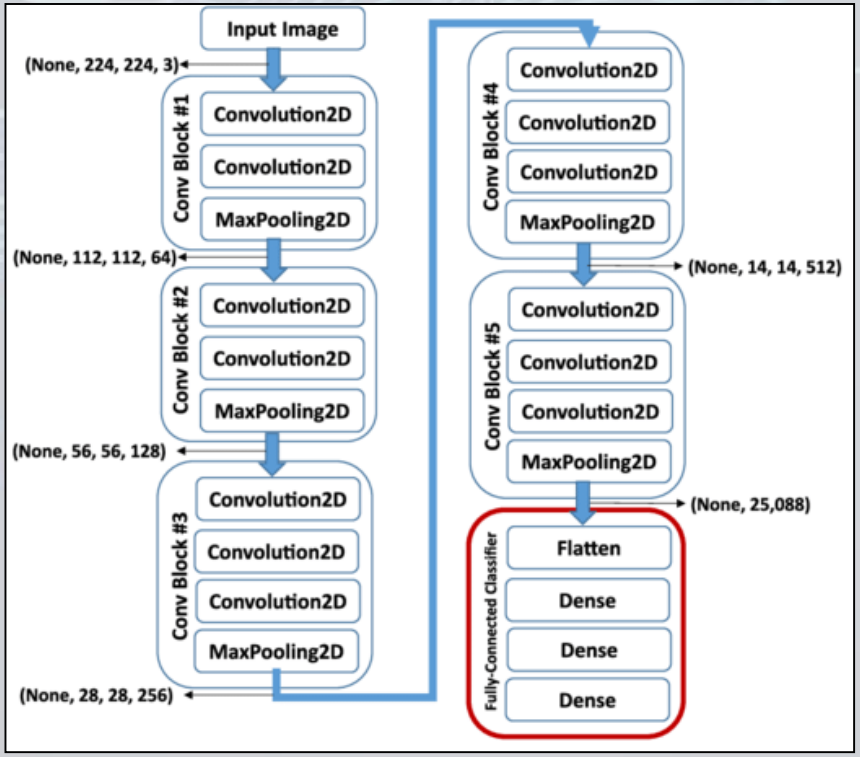
  - 사전 훈련된 네트워크를 사용하는 방법

  (1) 특성 추출 (Feature Extraction)

  (2) 미세 조정 (Fine Tuning)

CNN – 개와 고양이 예측

- 사전 훈련된 네트워크 사용하기
  - VGG16의 구조 : 컨볼루션 층 13개, MaxPooling 5개, Fully Connected Layer 4개로 구성
  - 최종적으로 특성 맵의 크기는 **(4, 4, 512)**



CNN – 개와 고양이 예측

- **특성 추출** : 사전에 학습된 네트워크의 표현을 사용하여 새로운 샘플에서 흥미로운 특성을 뽑아 내는 것
- VGG16 합성곱 기반 층 만들기

```
1  from keras.applications.vgg16 import VGG16
2
3  conv_base = VGG16(weights = "imagenet",
4                    include_top=False,
5                    input_shape=(150, 150, 3))
6
7  print(conv_base.summary())
```

- **weights** : 모델을 초기화할 가중치 체크포인트
- **include_top** : ImageNet 클래스의 완전 연결층를 사용할 것인지 여부 (False이면 별도의 완전 연결층 사용)
- **input_shape** : 네트워크에 입력할 이미지 텐서의 크기

CNN – 개와 고양이 예측

- 데이터 증식을 사용하지 않는 빠른 특성 추출 : 사전 훈련된 CNN을 가져와서 쓰고 사용자가 정의한 Fully Connected Layer만 붙여서 사용하는 방법

```
1  import os
2  import numpy as np
3  from keras.preprocessing.image import ImageDataGenerator
4  from keras.applications.vgg16 import VGG16
5
6  conv_base = VGG16(weights = "imagenet",
7                    include_top=False,
8                    input_shape=(150, 150, 3))
9
10 base_dir = "D:₩₩datasets₩₩dogs_vs_cats_small"
11 train_dir = os.path.join(base_dir, "train")
12 validation_dir = os.path.join(base_dir, "validation")
13 test_dir = os.path.join(base_dir, "test")
14
15 datagen = ImageDataGenerator(rescale=1./255)
16 batch_size = 20
17
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용하지 않는 빠른 특성 추출

```
18  def extract_feature(directory, sample_count):
19      features = np.zeros(shape=(sample_count, 4, 4, 512))
20      labels = np.zeros(shape=(sample_count))
21      generator = datagen.flow_from_directory(
22          directory,
23          target_size = (150, 150),
24          batch_size = batch_size,
25          class_mode = "binary")
26
27      i = 0
28      for inputs_batch, labels_batchs in generator:
29          features_batch = conv_base.predict(inputs_batch)
30          features[i * batch_size : (i+1) * batch_size] = features_batch
31          labels[i * batch_size : (i+1) * batch_size] = labels_batchs
32
33          i += 1
34          if i*batch_size >= sample_count:
35              break
36      return features, labels
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용하지 않는 빠른 특성 추출

```
38  train_features, train_labels = extract_feature(train_dir, 2000)
39  validation_features, validation_labels = extract_feature(validation_dir, 1000)
40  test_features, test_labels = extract_feature(test_dir, 1000)
41
42  train_features = np.reshape(train_features, (2000, 4 * 4 * 512))
43  validation_features = np.reshape(validation_features, (1000, 4 * 4 * 512))
44  test_features = np.reshape(test_features, (1000, 4 * 4 * 512))
45
46  from keras import layers
47  from keras import models
48
49  model = models.Sequential()
50  model.add(layers.Dense(256, activation="relu", input_dim=4 * 4 * 512))
51  model.add(layers.Dropout(0.5))
52  model.add(layers.Dense(1, activation="sigmoid"))
53  model.compile(loss="binary_crossentropy", optimizer=optimizers.RMSprop(lr=2e-5),
    metrics=["acc"])
54
55  history = model.fit(train_features, train_labels, epochs=20, batch_size=20,
56          verbose=0, validation_data=(validation_features, validation_labels))
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용하지 않는 빠른 특성 추출

```
57  import matplotlib.pyplot as plt
58
59  acc = history.history["acc"]
60  loss =  history.history["loss"]
61  val_acc =  history.history["val_acc"]
62  val_loss =  history.history["val_loss"]
63
64  epochs = range(1, len(acc) + 1)
65
66  plt.plot(epochs, acc, "bo", label="Traing acc")
67  plt.plot(epochs, val_acc, "b", label="Validation acc")
68  plt.title("Training and Validation Accuracy")
69  plt.legend()
70
71  plt.figure()
72  plt.plot(epochs, loss, "bo", label="Traing loss")
73  plt.plot(epochs, val_loss, "b", label="Validation loss")
74  plt.title("Training and Validation Loss")
75  plt.legend()
76  plt.show()
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용하지 않는 빠른 특성 추출

```
19      features = np.zeros(shape=(sample_count, 4, 4, 512))
20      labels = np.zeros(shape=(sample_count))
...     ...
28      for inputs_batch, labels_batchs in generator:
29          features_batch = conv_base.predict(inputs_batch)
30          features[i * batch_size : (i+1) * batch_size] = features_batch
31          labels[i * batch_size : (i+1) * batch_size] = labels_batchs
32
33          i += 1
34          if i*batch_size >= sample_count:
35              break
```

- [19-20] VGG16의 최종 크기인 4 x 4 x 512의 배열을 0으로 초기화
- [29] conv_base 모델의 predict() 메소드를 호출하여 이미지에서 특성을 추출
- [30-31] 추출된 특성과 라벨을 저장
- [34] 설정한 이미지 데이터를 모두 처리하고 나면 중지

CNN – 개와 고양이 예측

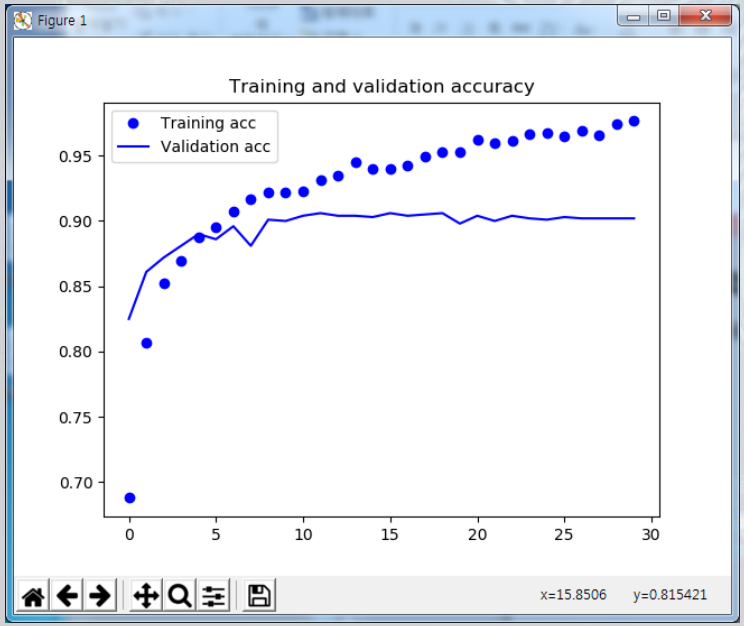
- 데이터 증식을 사용하지 않는 빠른 특성 추출

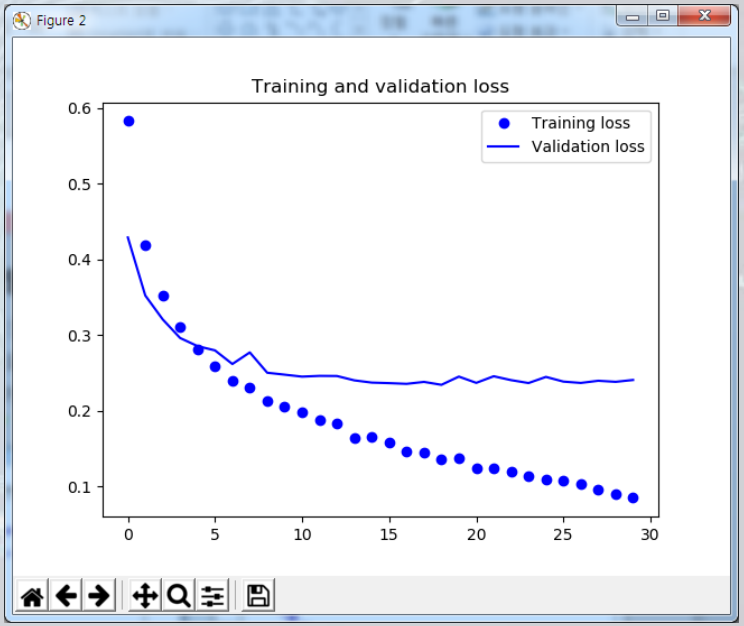
```
38  train_features, train_labels = extract_feature(train_dir, 2000)
39  validation_features, validation_labels = extract_feature(validation_dir, 1000)
40  test_features, test_labels = extract_feature(test_dir, 1000)
41
42  train_features = np.reshape(train_features, (2000, 4 * 4 * 512))
43  validation_features = np.reshape(validation_features, (1000, 4 * 4 * 512))
44  test_features = np.reshape(test_features, (1000, 4 * 4 * 512))
...  ...
50  model.add(layers.Dense(256, activation="relu", input_dim=4 * 4 * 512))
```

- [38-40] 훈련 데이터 2,000개, 검증 데이터 1,000개, 테스트 데이터 1,000개의 특성 추출
- [42-44] Fully Connected Layer에 주입하기 위해 1차원으로 변환
- [50] 입력 크기 : 4 * 4 * 512 = 8,192

CNN – 개와 고양이 예측

- 데이터 증식을 사용하지 않는 빠른 특성 추출
  - 검증 정확도가 90%에 도달 → 작은 모델보다 정확도가 높음
  - 증식을 사용하지 않았기 때문에 초기부터 과대적합 됨





CNN – 개와 고양이 예측

- 데이터 증식을 사용한 특성 추출

```
1  from keras import models
2  from keras import layers
3  from keras.applications.vgg16 import VGG16
4
5  conv_base = VGG16(weights = "imagenet",
6                    include_top=False,
7                    input_shape=(150, 150, 3))
8  conv_base.trainable = False
9
10 model = models.Sequential()
11 model.add(conv_base)
12 model.add(layers.Flatten())
13 model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
14 model.add(layers.Dropout(0.5))
15 model.add(layers.Dense(1, activation='sigmoid'))
16
17 print(model.summary())
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용한 특성 추출

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2097408
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 2,097,665
Non-trainable params: 14,714,688
_________________________________________________________________
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용한 특성 추출

8	conv_base.trainable = False

- [8] 가중치가 업데이트되는 것을 방지 (Fully Connected Layer로부터 너무 큰 오차 신호가 업데이트되어 사전 훈련된 층들이 망가지는 것을 방지하기 위해 동결) → Dense 층의 가중치만 업데이트
- 변경 사항을 적용하려면 모델을 먼저 컴파일해야 함.

CNN – 개와 고양이 예측

- 데이터 증식을 사용한 특성 추출

```
1  from keras import models
2  from keras import layers
3  from keras.applications.vgg16 import VGG16
4  import os
5
6  conv_base = VGG16(weights = "imagenet",
7                    include_top=False,
8                    input_shape=(150, 150, 3))
9  conv_base.trainable = False
10
11 base_dir = "D:₩₩datasets₩₩dogs_vs_cats_small"
12 train_dir = os.path.join(base_dir, "train")
13 validation_dir = os.path.join(base_dir, "validation")
14 test_dir = os.path.join(base_dir, "test")
15
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용한 특성 추출

```
16  model = models.Sequential()
17  model.add(conv_base)
18  model.add(layers.Flatten())
19  model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
20  model.add(layers.Dropout(0.5))
21  model.add(layers.Dense(1, activation='sigmoid'))
22
```

CNN – 개와 고양이 예측

- 데이터 증식을 사용한 특성 추출

```
23 from keras.preprocessing.image import ImageDataGenerator
24 from keras import optimizers
25
26 train_datagen = ImageDataGenerator(
27         rescale=1./255,
28         rotation_range=20,
29         width_shift_range=0.1,
30         height_shift_range=0.1,
31         shear_range=0.1,
32         zoom_range=0.1,
33         horizontal_flip=True,
34         fill_mode='nearest')
35
36 test_datagen = ImageDataGenerator(rescale=1./255)
37
```

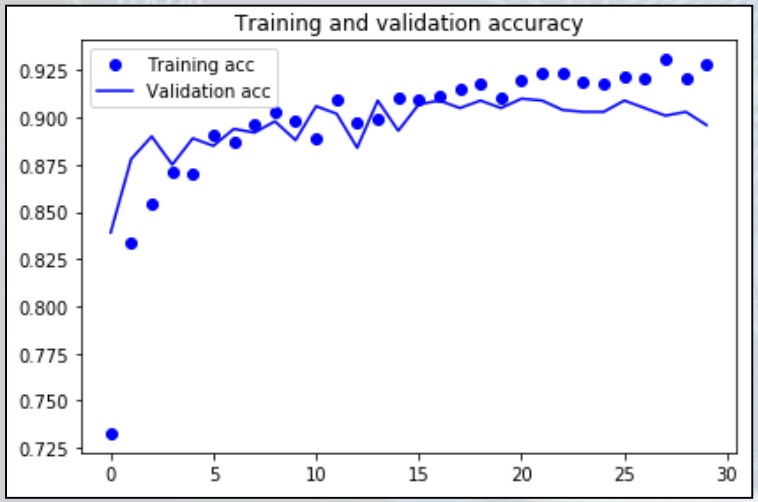
CNN – 개와 고양이 예측

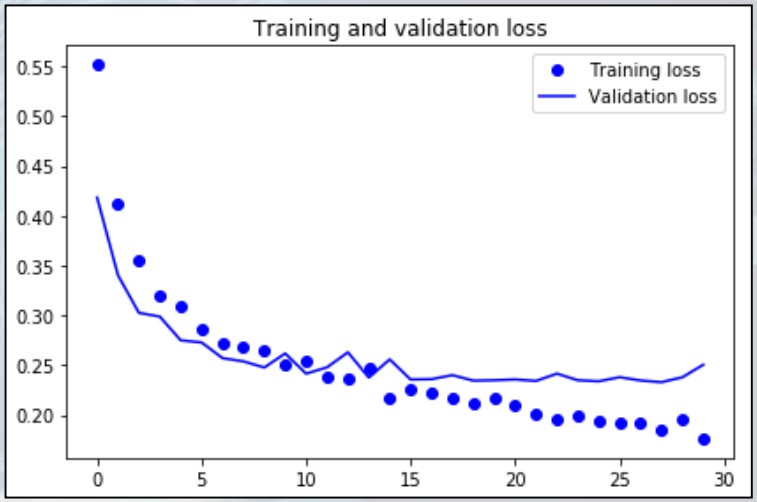
- 데이터 증식을 사용한 특성 추출

```
38  train_generator = train_datagen.flow_from_directory(
39          train_dir,
40          target_size=(150, 150),
41          batch_size=20,
42          class_mode='binary')
43
44  validation_generator = test_datagen.flow_from_directory(
45          validation_dir,
46          target_size=(150, 150),
47          batch_size=20,
48          class_mode='binary')
 9
50  model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=2e-5),
    metrics=['acc'])
51
52  history = model.fit_generator(train_generator, steps_per_epoch=100, epochs=30,
53        validation_data=validation_generator, validation_steps=50, verbose=2)
54  ...
```

CNN – 개와 고양이 예측

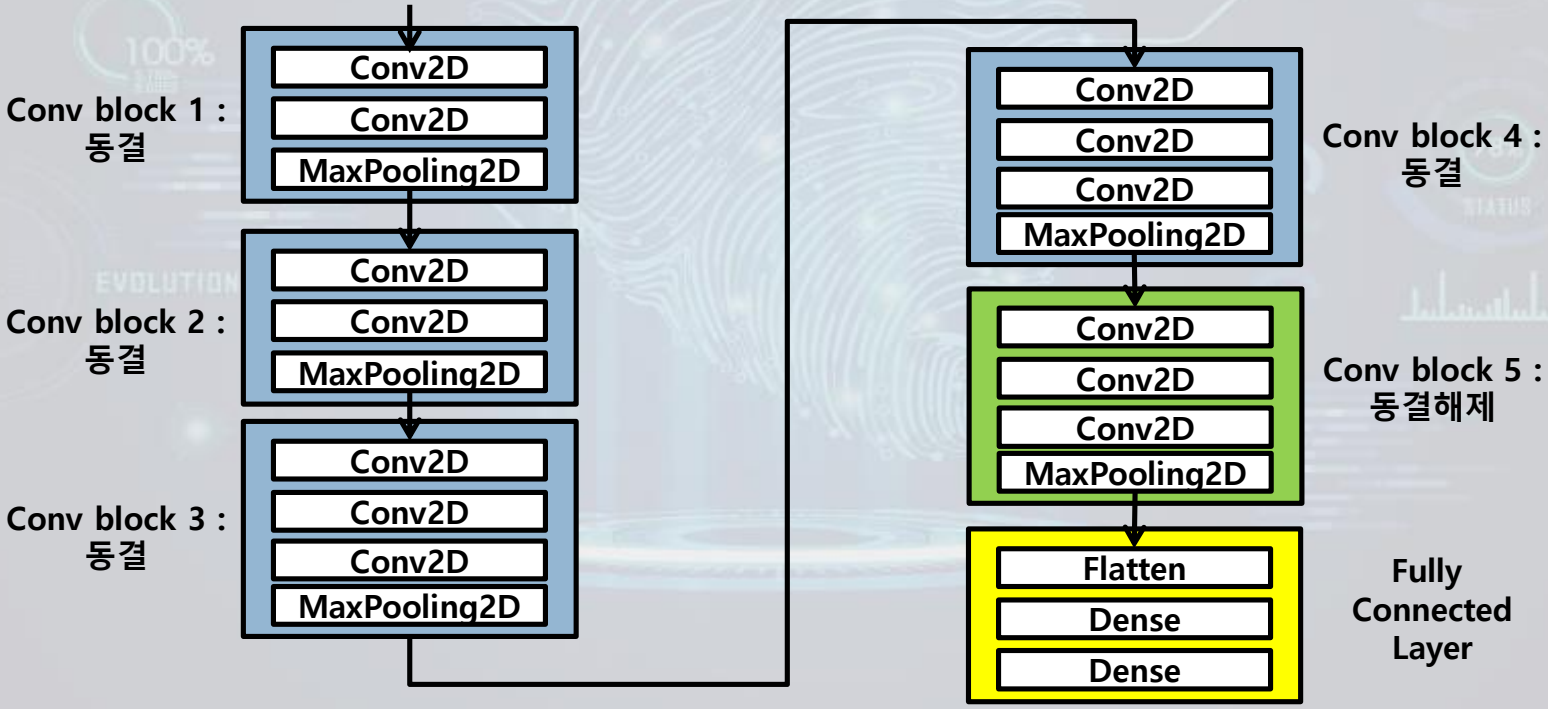
- 데이터 증식을 사용한 특성 추출
  - 정확도는 유사하지만 훈련 데이터의 과적합은 감소





CNN – 개와 고양이 예측

- **미세 조정** : 특성 추출에 사용했던 동결 모델의 **상위 층 몇 개를 동결에서 해제**하고 모델에 새로 추가한 층 (여기에서는 Fully Connected Layer)과 함께 훈련하는 것
- **주어진 문제에 조금 더 밀접하게 재사용 모델의 표현**을 일부 조정 → 미세 조정



CNN – 개와 고양이 예측

- 미세 조정 단계
  (1) 사전에 훈련된 기반 네트워크 위에 새로운 네트워크를 추가
  (2) 기반 네트워크를 동결
  (3) 새로 추가한 네트워크를 훈련
  (4) 기반 네트워크에서 일부 층의 동결을 해제
  (5) 동결을 해제한 층과 새로 추가한 층을 함께 훈련

- 미세 조정 시 주의점
  (1) 상위 층은 하위 층보다 좀 더 특화된 특성이므로 상위층을 미세 조정하는 것이 유리
  (2) 훈련해야 할 파라미터가 많을수록 과대적합 가능성 높음
  → 최상위층 2-3개의 층만 미세 조정하는 것을 권장

CNN – 개와 고양이 예측

- 미세 조정

```
1  from keras import models
2  from keras import layers
3  from keras.applications.vgg16 import VGG16
4  import os
5
6  conv_base = VGG16(weights = "imagenet",
7                    include_top=False,
8                    input_shape=(150, 150, 3))
9  conv_base.trainable = False
10
11 set_trainable = False
12 for layer in conv_base.layers:
13     if layer.name == 'block5_conv1':
14         set_trainable = True
15     if set_trainable:
16         layer.trainable = True
17     else:
18         layer.trainable = False
```

CNN – 개와 고양이 예측

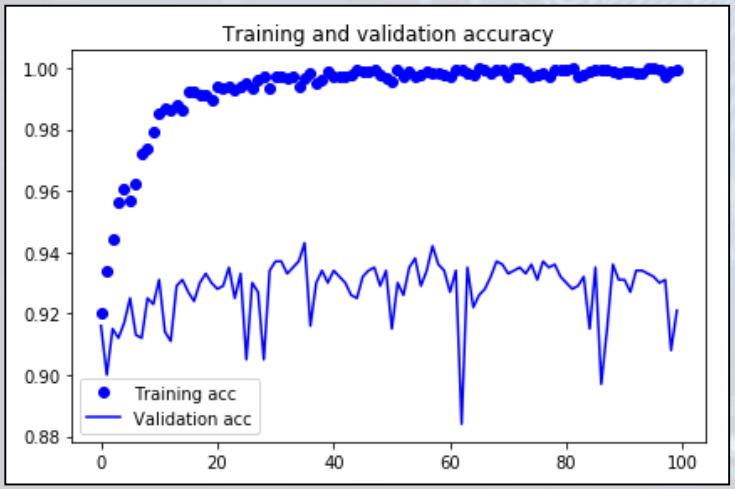
- 미세 조정

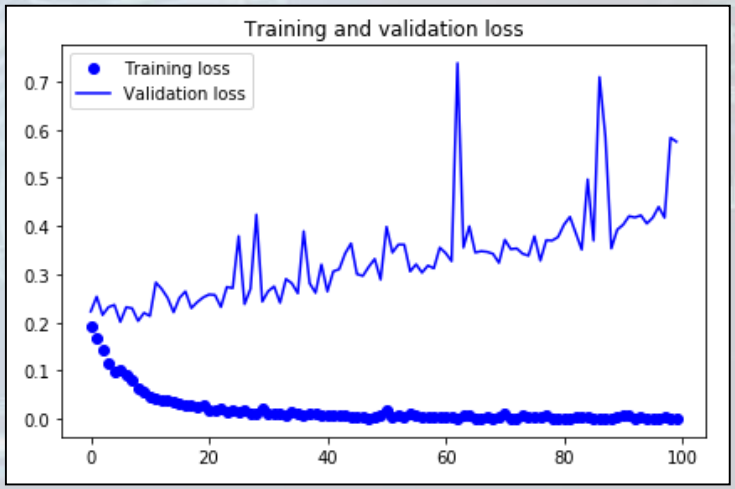
```
59  model.compile(loss='binary_crossentropy',
60        optimizer=optimizers.RMSprop(lr=1e-5), metrics=['acc'])
61
62  history = model.fit_generator(train_generator, steps_per_epoch=100,
63      epochs=100, validation_data=validation_generator,
        validation_steps=50, verbose=2)
64
65  model.save('cats_and_dogs_small_4.h5')
...  ...
```

- 학습률을 낮춘 RMSProp 옵티마이저를 사용
- 학습률을 낮추는 이유는 미세 조정하는 세 개의 층에서 학습된 표현을 조금씩 수정하기 위해서
- 변경량이 너무 크면 학습된 표현에 나쁜 영향을 끼칠 수 있음

CNN – 개와 고양이 예측

- 미세 조정





CNN – 개와 고양이 예측

- 모델 평가

```
1  from keras.preprocessing.image import ImageDataGenerator
2  from keras.models import load_model
3
4  test_dir = "D:₩₩datasets₩₩dogs_vs_cats_small₩₩test"
5  test_datagen = ImageDataGenerator(rescale= 1./255)
6
7  test_generator = test_datagen.flow_from_directory(
8          test_dir,
9          target_size=(150, 150),
10         batch_size=20,
11         class_mode='binary')
12
13 model = load_model("dogs_vs_cats_small_01.h5")
14
15 test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
16 print('test acc:', test_acc)
```

CNN – 개와 고양이 예측

- 모델 평가

```
test acc: 0.7130000007152557
```

CNN – 개와 고양이 예측

- 학습 과정 시각화

(1) CNN 중간층의 출력 (중간층에 있는 활성화)을 시각화 : 연속된 CNN 층이 입력을 어떻게 변형시키는지 파악

(2) CNN 필터를 시작화 : CNN 필터가 찾으려는 시각적인 패턴과 개념 이해

(3) 클래스 활성화 대한 히트맵을 이미지에 시각화 : 이미지의 어느 부분이 주어진 클래스에 속하는데 기여했는지 파악

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기

```
1  from keras.models import load_model
2
3  model = load_model('dogs_vs_cats_small_02.h5')
4  model.summary()
```

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513

Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기 (이미지를 4차원 텐서로 변환)

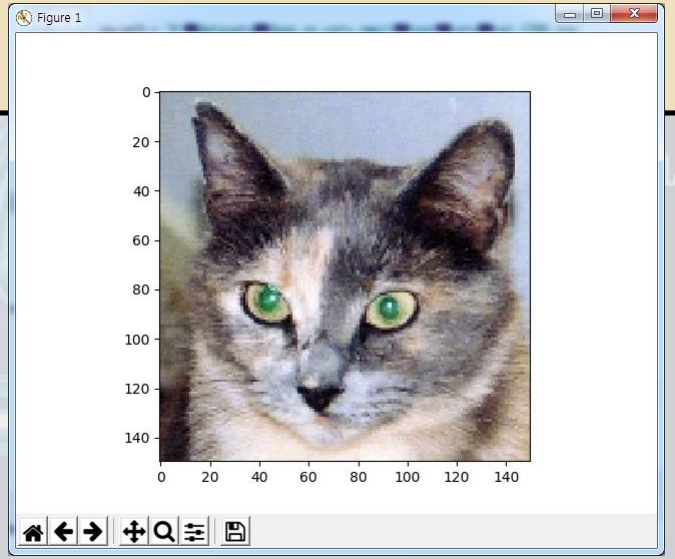
```
1   from keras.models import load_model
2
3   model = load_model('dogs_vs_cats_small_02.h5')
4
5   img_path = 'D:₩₩datasets₩₩dogs_vs_cats_small₩₩test₩₩cats₩₩cat.1700.jpg'
6
7   from keras.preprocessing import image
8   import numpy as np
9
10  img = image.load_img(img_path, target_size=(150, 150))
11  img_tensor = image.img_to_array(img)
12  img_tensor = np.expand_dims(img_tensor, axis=0)
13  img_tensor /= 255.
14
15  print(img_tensor.shape)
16
```

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기 (이미지를 4차원 텐서로 변환)

```
17  from keras import models
18  import matplotlib.pyplot as plt
19
20  plt.imshow(img_tensor[0])
21  plt.show()
```

```
(1, 150, 150, 3)
```



CNN – 개와 고양이 예측

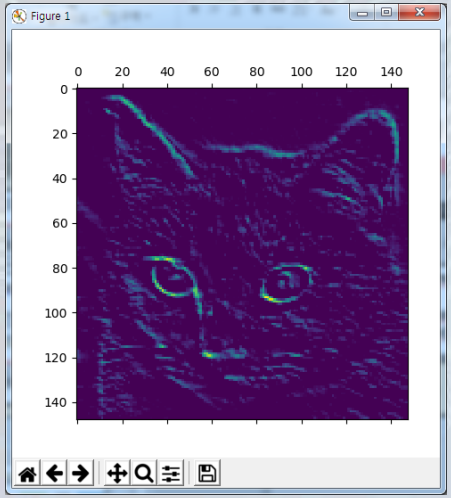
- 중간층의 활성화 시각화하기

```
23  layer_outputs = [layer.output for layer in model.layers[:8]]
24  activation_model = models.Model(inputs=model.input, outputs=layer_outputs)
25  activations = activation_model.predict(img_tensor)
26
27  first_layer_activation = activations[0]
28  print(first_layer_activation.shape)
29
30  plt.matshow(first_layer_activation[0, :, :, 22], cmap='viridis')
31  plt.show()
```

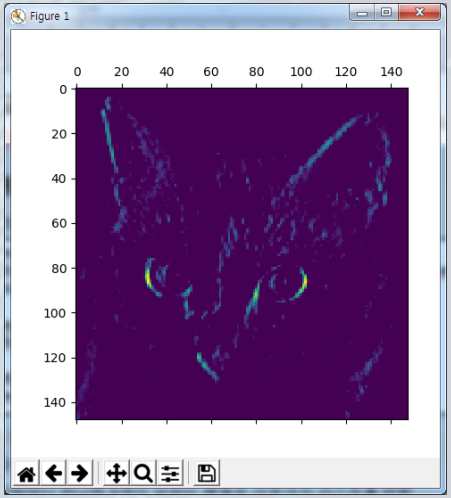
- [23] 상위 8개 층의 출력을 추출
- [24] 입력에 대해 8개 층의 출력을 반환하는 모델을 만듬
- [25] 층의 활성화마다 하나씩 8개의 넘파이 배열로 이루어진 리스트를 반환
- [30] 22번째 채널의 특성맵을 시각화하기

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기



22번째



25번째

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기 (모든 채널 시각화)

```
27  # 상위 8 개 층의 이름을 그래프 제목으로 사용
28  layer_names = []
29  for layer in model.layers[:8]:
30      layer_names.append(layer.name)
31
32  # 한 줄에 표시하는 이미지 수
33  images_per_row = 16
34
35  # 특성 맵을 그림
36  for layer_name, layer_activation in zip(layer_names, activations):
37      # 특성 맵에 있는 특성의 수
38      n_features = layer_activation.shape[-1]
39      # 특성 맵의 크기 (1, size, size, n_features)
40      size = layer_activation.shape[1]
41      # 활성화 채널을 위한 그리드 크기를 구함
42      n_cols = n_features // images_per_row
43      display_grid = np.zeros((size * n_cols, images_per_row * size))
44
```

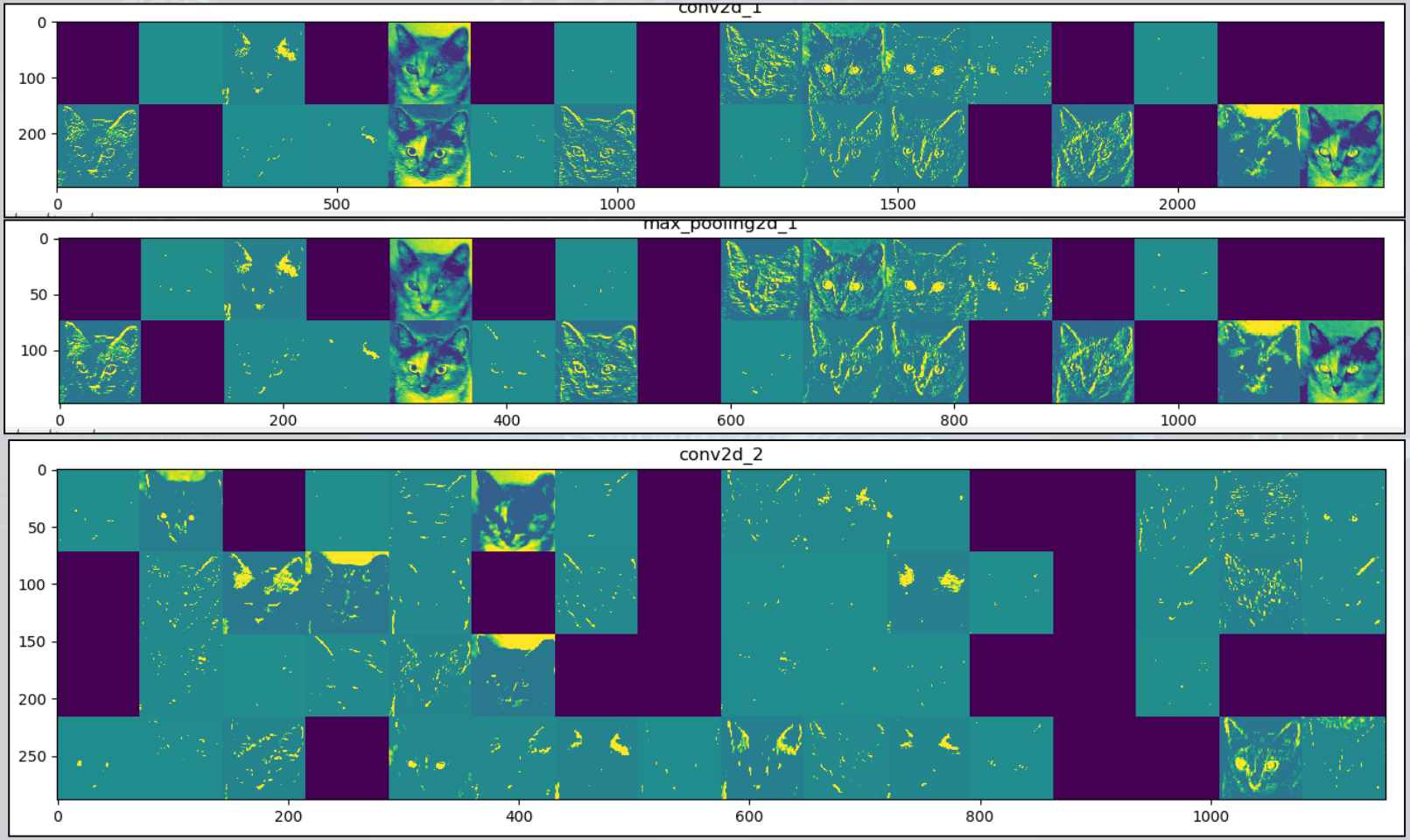
CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기 (모든 채널 시각화)

```
45      # 각 활성화를 하나의 큰 그리드에 채움
46      for col in range(n_cols):
47          for row in range(images_per_row):
48              channel_image = layer_activation[0, :, :, col * images_per_row + row]
49
50              # 그래프로 나타내기 좋게 특성을 처리
51              channel_image -= channel_image.mean()
52              channel_image /= channel_image.std()
53              channel_image *= 64
54              channel_image += 128
55              channel_image = np.clip(channel_image, 0, 255).astype('uint8')
56              display_grid[col * size : (col + 1) * size,
57                           row * size : (row + 1) * size] = channel_image
58
59      # 그리드를 출력
60      scale = 1. / size
61      plt.figure(figsize=(scale * display_grid.shape[1],
62                          scale * display_grid.shape[0]))
63      plt.title(layer_name)
64      plt.grid(False)
65      plt.imshow(display_grid, aspect='auto', cmap='viridis')
66
67  plt.show()
```

CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기 (모든 채널 시각화)



CNN – 개와 고양이 예측

- 중간층의 활성화 시각화하기 (모든 채널 시각화) – 상위층으로 갈수록 특징만 남음