Dogs vs Cats

Dogs vs Cats

- Dogs vs Cats dataset
  - https://www.kaggle.com/c/dogs-vs-cats/data
  - 25000 images (12500 cats and 12500 dogs)
  - Create dataset
    - Training: 1000 samples for each class
    - Validation: 500 samples for each class
    - Test: 500 samples for each class

Training a convnet from scratch on a small dataset

Prepare dataset

```
import os, shutil

original_db_dir = './train'

base_dir = './cats_and_dogs_small'
os.mkdir(base_dir)
train_dir = os.path.join(base_dir, 'train')
os.mkdir(train_dir)
validation_dir = os.path.join(base_dir, 'validation')
os.mkdir(validation_dir)
test_dir = os.path.join(base_dir, 'test')
os.mkdir(test_dir)

train_cats_dir = os.path.join(train_dir, 'cats')
os.mkdir(train_cats_dir)
train_dogs_dir = os.path.join(train_dir, 'dogs')
os.mkdir(train_dogs_dir)

validation_cats_dir = os.path.join(validation_dir, 'cats')
os.mkdir(validation_cats_dir)
validation_dogs_dir = os.path.join(validation_dir, 'dogs')
os.mkdir(validation_dogs_dir)

test_cats_dir = os.path.join(test_dir, 'cats')
os.mkdir(test_cats_dir)
test_dogs_dir = os.path.join(test_dir, 'dogs')
os.mkdir(test_dogs_dir)
```

Training a convnet from scratch on a small dataset

Prepare dataset

```
frames = ['cat.{}.jpg'.format(i) for i in range(1000)]
for fname in frames:
  src = os.path.join(original_db_dir, fname)
  dst = os.path.join(train_cats_dir, fname)
  shutil.copyfile(src,dst)

frames = ['cat.{}.jpg'.format(i) for i in range(1000,1500)]
for fname in frames:
  src = os.path.join(original_db_dir, fname)
  dst = os.path.join(validation_cats_dir, fname)
  shutil.copyfile(src,dst)

frames = ['cat.{}.jpg'.format(i) for i in range(1500,2000)]
for fname in frames:
  src = os.path.join(original_db_dir, fname)
  dst = os.path.join(test_cats_dir, fname)
  shutil.copyfile(src,dst)
```

Training a convnet from scratch on a small dataset

Prepare dataset

```
frames = ['dog.{}.jpg'.format(i) for i in range(1000)]
for fname in frames:
  src = os.path.join(original_db_dir, fname)
  dst = os.path.join(train_dogs_dir, fname)
  shutil.copyfile(src,dst)

frames = ['dog.{}.jpg'.format(i) for i in range(1000,1500)]
for fname in frames:
  src = os.path.join(original_db_dir, fname)
  dst = os.path.join(validation_dogs_dir, fname)
  shutil.copyfile(src,dst)

frames = ['dog.{}.jpg'.format(i) for i in range(1500,2000)]
for fname in frames:
  src = os.path.join(original_db_dir, fname)
  dst = os.path.join(test_dogs_dir, fname)
  shutil.copyfile(src,dst)
```

Training a convnet from scratch on a small dataset

Create Neural Network

```
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras import models

model = models.Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()

from keras import optimizers

model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])
```

Training a convnet from scratch on a small dataset

Create Neural Network

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

Training a convnet from scratch on a small dataset

Create Neural Network

문제 유형	마지막 층의 활성화 함수	손실 함수
이진 분류	sigmoid	binary_crossentropy
단일 레이블 다중분류	softmax	categorical_crossentropy
다중 레이블 다중분류	sigmoid	binary_crossentropy
임의 값에 대한 회귀	없음	mse
0과 1사이 값에 대한 회귀	sigmoid	mse or binary_crossentropy

```
from keras import optimizers

model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])
```

Training a convnet from scratch on a small dataset

Data Preprocessing

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150,150),
    batch_size=20,
    class_mode='binary'
)
validation_generator = test_datagen.flow_from_directory(
    validation_dir,
    target_size=(150,150),
    batch_size=20,
    class_mode='binary'
)
```

- ImageDataGenerator 클래스는 디스크에 있는 이미지 파일을 읽어 전처리 된 배치 텐서로 자동으로 바꾸어 주는 파이썬 제너레이터를 만들어 줌

Training a convnet from scratch on a small dataset

Data Preprocessing

```
>>> for data_batch, labels_batch in train_generator:
...     print ('배치 데이터 크기:',  data_batch.shape)
...     print ('배치 레이블 크기:',  labels_batch.shape)
...     break
배치 데이터 크기: (20, 150, 150, 3)
배치 레이블 크기: (20,)
```

Training a convnet from scratch on a small dataset

Data Preprocessing

```
history = model.fit_generator(    # 배치 제너레이터를 사용하여 모델 훈련
    train_generator,
    steps_per_epoch=100,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=50
)

model.save('cats_and_dogs_small_1.h5')   # 모델 저장
```

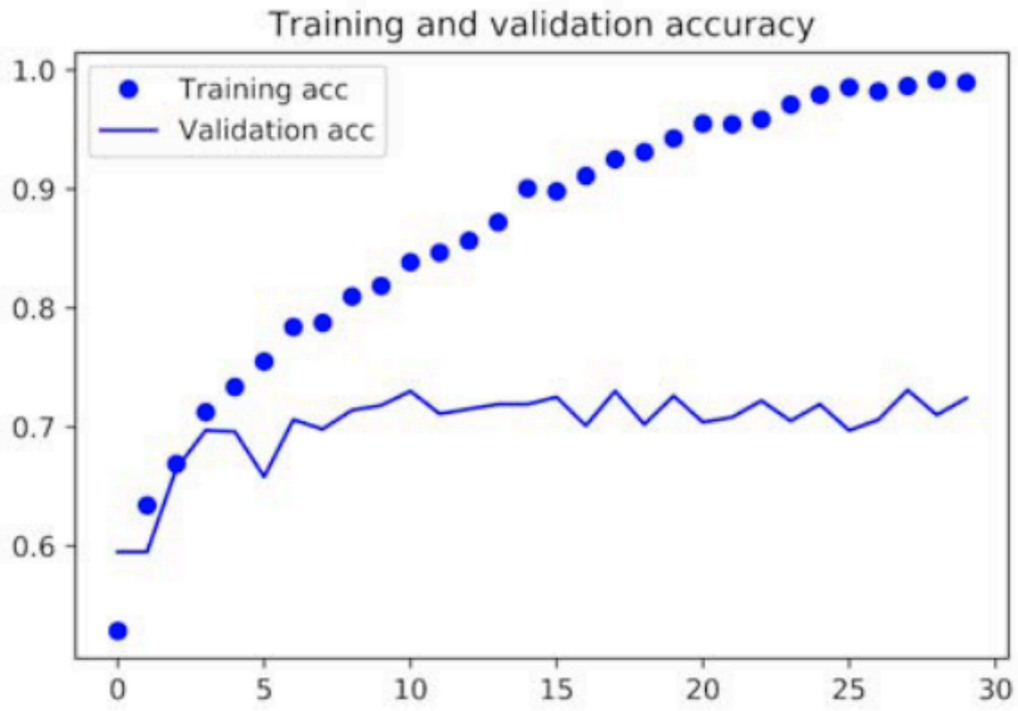
Training a convnet from scratch on a small dataset

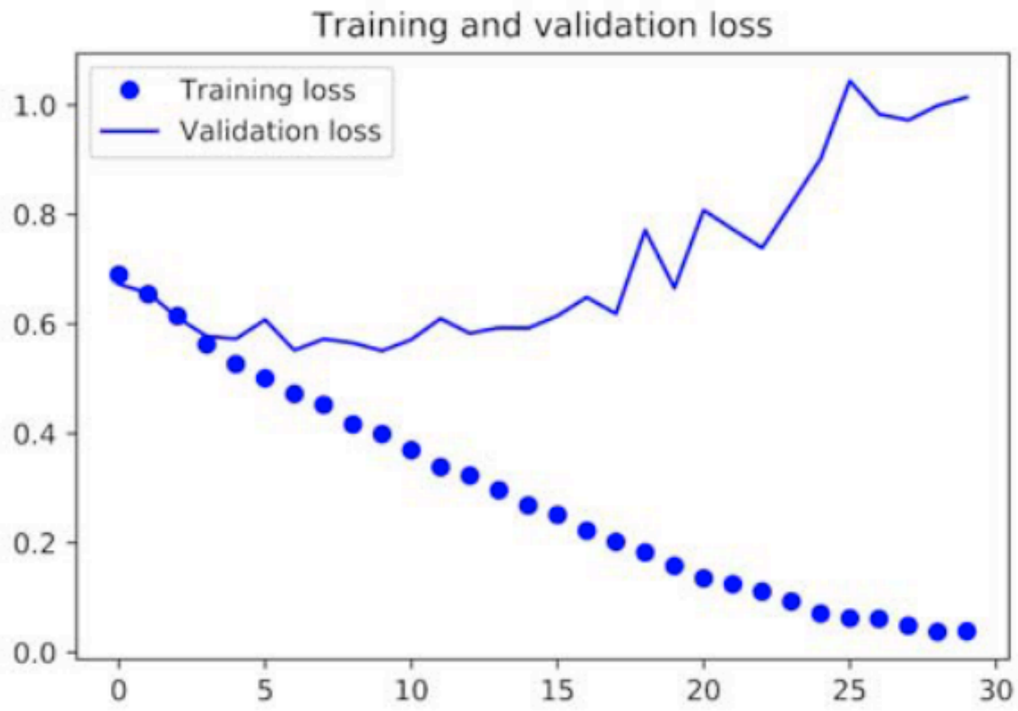
Data Preprocessing

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

Training a convnet from scratch on a small dataset

Data Preprocessing



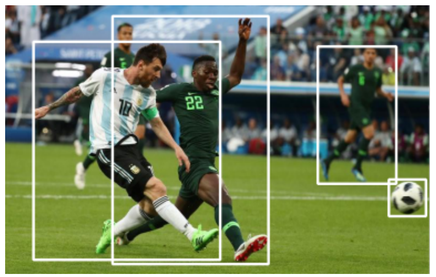


Training a convnet from scratch on a small dataset

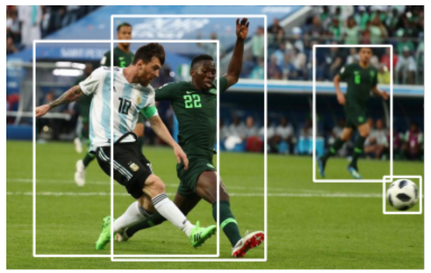
Data Augmentation

- 성능을 향상시키기 위해 원래 데이터에 랜덤변환을 적용하여 부풀림

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
```




Rotation




Shearing

Training a convnet from scratch on a small dataset

Data Augmentation

```
from keras.preprocessing import image

fnames = sorted([os.path.join(train_cats_dir, fname) for fname in os.listdir(train_cats_dir)])
img_path = fnames[3]     # 증식할 이미지 선택
img = image.load_img(img_path, target_size=(150,150))        #  이미지를 읽고 크기 변경
x = image.img_to_array(img)        #  (150,150,3)  크기의 numpy  배열로 변환
x = x.reshape((1,)+x.shape)        #  (1,150,150,3)  크기로 변환
i=0
for batch in datagen.flow(x, batch_size=1):
  plt.figure(i)
  imgplot = plt.imshow(image.array_to_img(batch[0]))
  i += 1
  if i%4 == 0:
    break
plt.show()
```

Training a convnet from scratch on a small dataset

Data Augmentation

Training a convnet from scratch on a small dataset

Data Augmentation

```
model = models.Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Training a convnet from scratch on a small dataset

Data Augmentation

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout (Dropout)            (None, 6272)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

Training a convnet from scratch on a small dataset

Data Augmentation

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    )
test_datagen = ImageDataGenerator(rescale=1./255)     #  검증 데이터는 증식되민 안됨

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150,150),
    batch_size=32,
    class_mode='binary'
)

validation_generator = test_datagen.flow_from_directory(
    validation_dir,
    target_size=(150,150),
    batch_size=32,
    class_mode='binary'
)
```

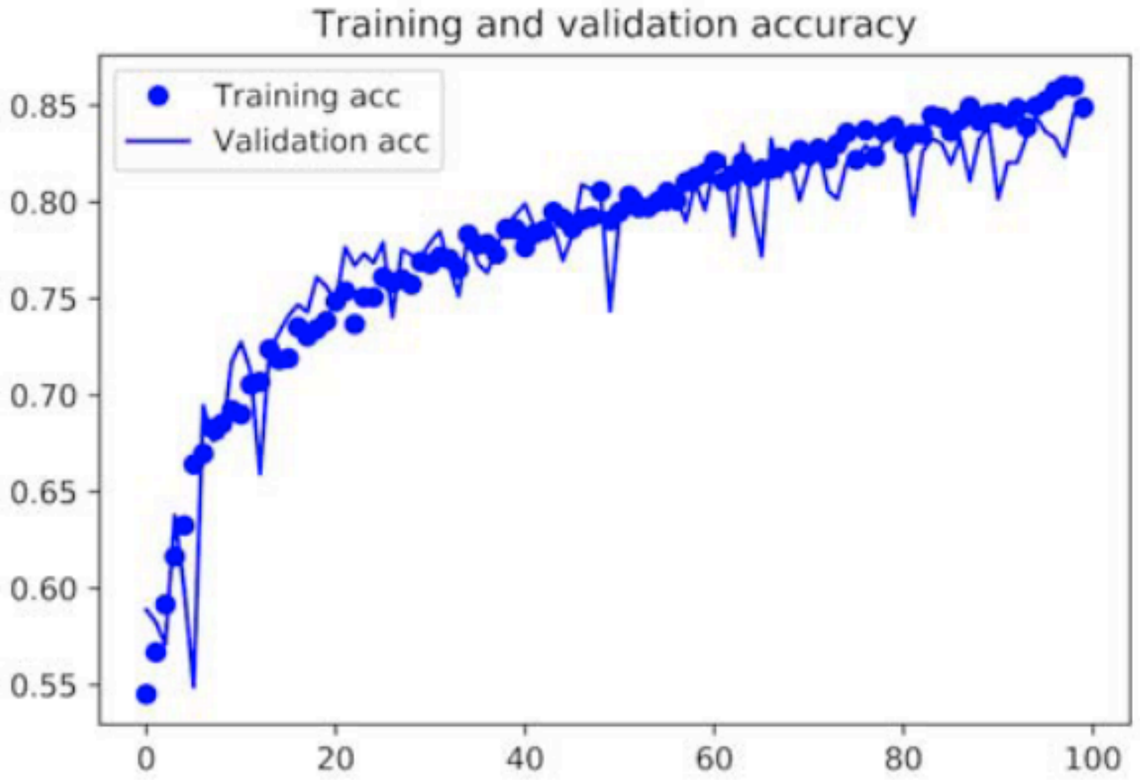
Training a convnet from scratch on a small dataset

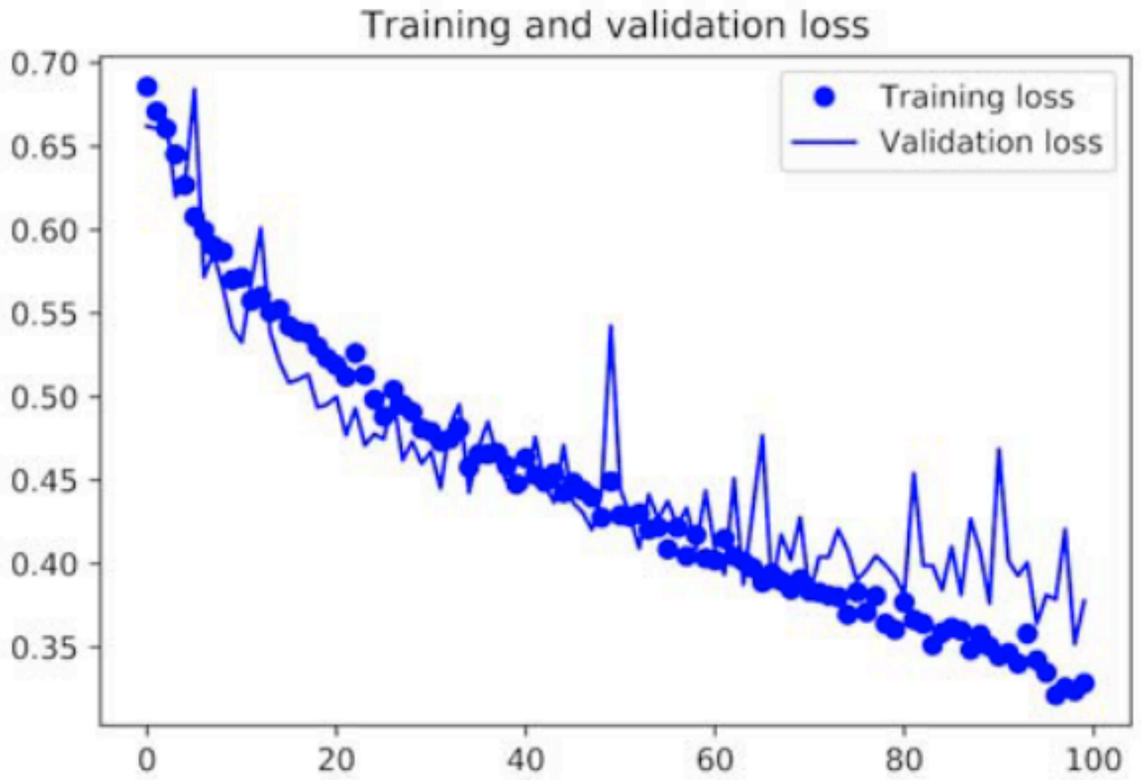
Data Augmentation

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=100,
    epochs=300,
    validation_data=validation_generator,
    validation_steps=50
)

model.save('cats_and_dogs_small_2.h5')
```

Training a convnet from scratch on a small dataset

Data Augmentation



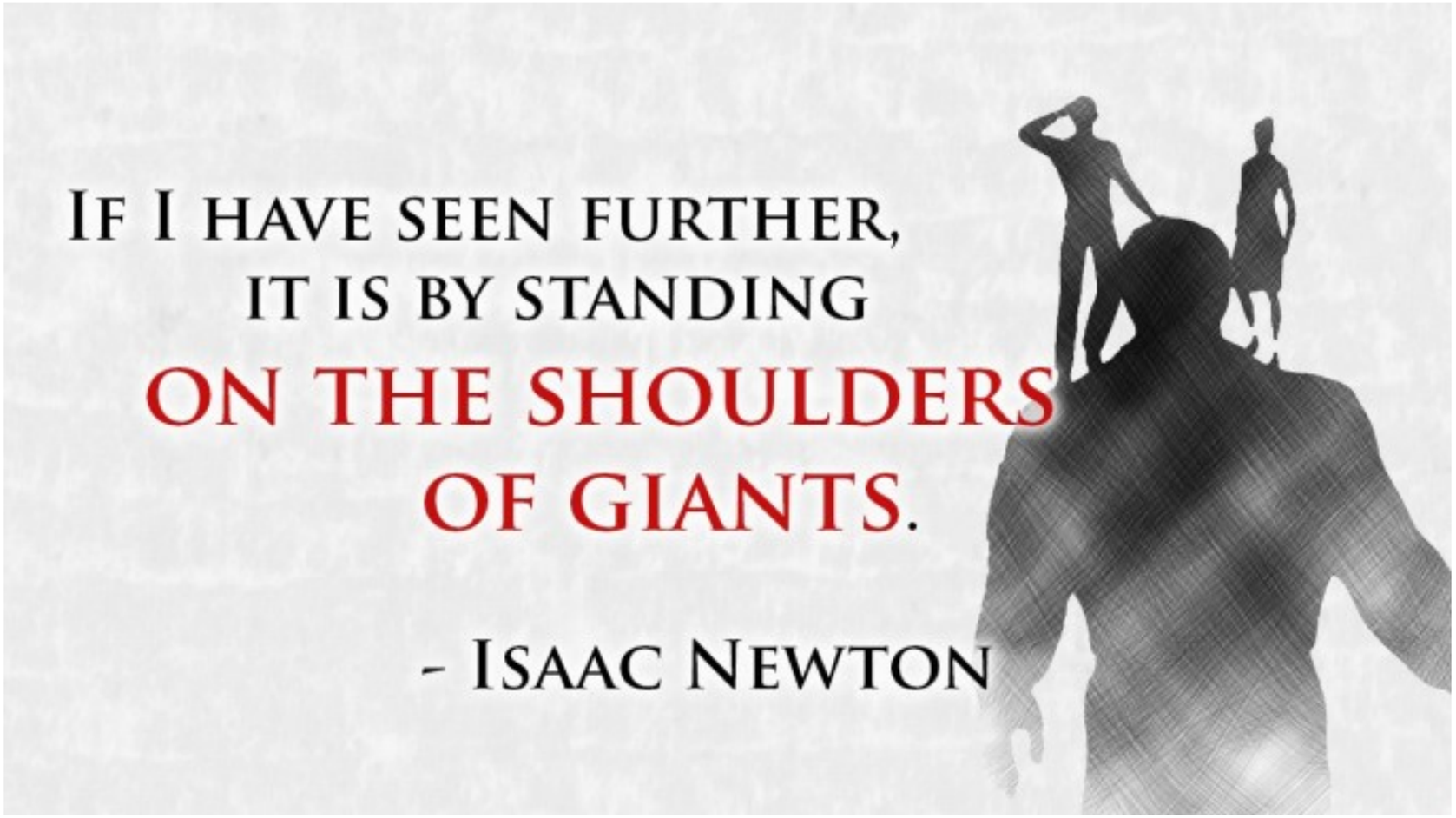


Transfer Learning

Don't reinvent the wheel

Transfer Learning

Standing on the shoulders of giants



Transfer Learning

- 작은 이미지 데이터셋에 딥러닝을 적용하는 일반적이고 효과적인 방법은 사전 훈련된 네트워크를 사용
- 사전 훈련된 네트워크(pretrained network) - 일반적으로 대규모 이미지 분류 문제를 위해 대량의 데이터셋에서 미리 훈련되어 저장된 네트워크
- ImageNet - 1400만개의 레이블된 이미지와 1000개의 클래스로 이루어 진 데이터셋
- VGG16 - Karen Simonyan & Andrew Zisserman이 2014년에 개발
- VGG16, ResNet, Inception, Inception-ResNet, Xception, ...

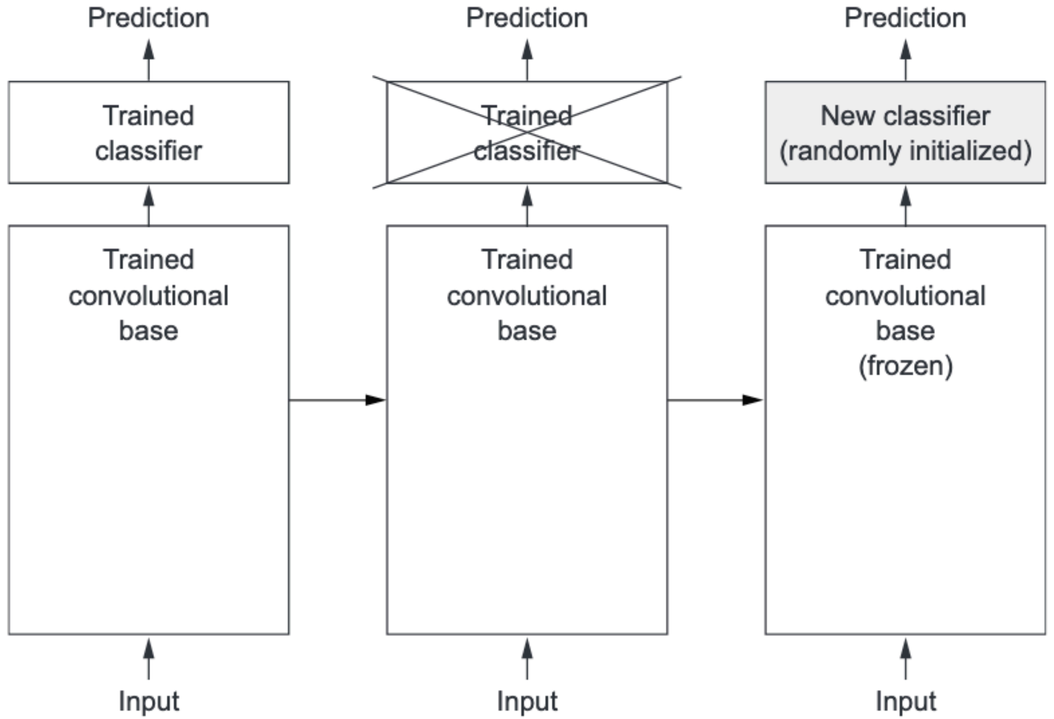
Transfer Learning

- 사전 훈련된 네트워크를 사용하는 방법
  - 특성 추출(Feature Extraction)
  - 미세 조정(Fine Tuning)

Transfer Learning

Feature Extraction

- 사전에 학습된 네트워크의 표현을 사용하여 새로운 샘플에서 흥미로운 특성을 추출
- 이런 특성을 사용하여 새로운 분류기를 처음부터 훈련



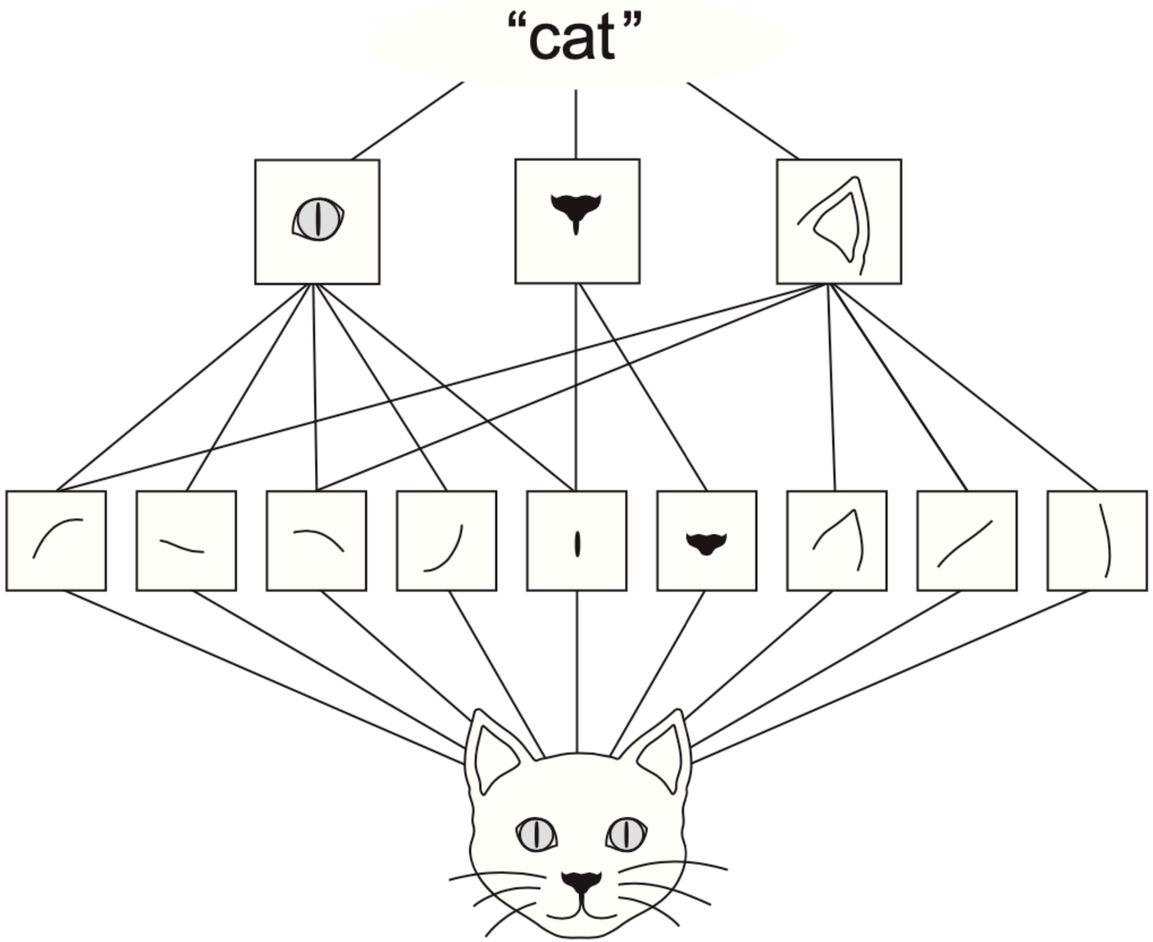
Transfer Learning

Feature Extraction

- 합성곱층(convolutional base)에 의해 학습된 표현이 더 일반적이어서 재사용이 가능
- Convnet의 특성맵은 사진에 대한 일반적인 콘셉트의 존재여부를 기록한 맵 → 주어진 컴퓨터 비젼 문제에 상관없이 유용하게 사용 가능
- 분류기에서 학습한 표현은 모델이 훈련된 클래스 집합에 특화 → 전체 사진에 어떤 클래스가 존재할 확률에 관한 정보만 가지고 있음
- Convnet에서 추출한 표현의 일반성(그리고 재사용성) 수준은 모델에 있는 층의 깊이에 달려 있음
  - 모델의 하위층은 (에지, 색깔, 질감 등) 지역적이고 매우 일반적 특성맵을 추출
  - 상위층은 ('강아지 눈'이나 '고양이 귀'처럼) 좀 더 추상적인 개념을 추출 → 새로운 데이터셋이 원본 모델이 훈련한 데이터셋과 많이 다르면 전체 convnet을 사용하는것 보다 모델의 하위 몇개 층만 특성추출에 사용하는 것이 좋음

Transfer Learning

Feature Extraction



Transfer Learning

Feature Extraction

- keras.applications 모델에서 사용 가능한 이미지분류 모델
  - Xception
  - Inception V3
  - ResNet50
  - VGG16
  - VGG19
  - MobileNet

Transfer Learning

Feature Extraction

```
from keras.applications.vgg16 import VGG16

conv_base = VGG16(
    weights='imagenet',
    include_top=False,
    input_shape=(150,150,3)
)

conv_base.summary()
```

Transfer Learning

Feature Extraction

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Transfer Learning

Feature Extraction

- 이 지점에서 두가지 방식이 가능
  - 새로운 데이터셋에서 convnet을 실행하고 출력을 numpy배열로 디스크에 저장 → 이 데이터를 독립된 완전연결분류기에 입력으로 사용. 모든 입력 이미지에 대해 convnet을 한번만 실행하면 되므로 빠르고 비용이 적게 듬. 하지만 이 기법에선 data augmentation을 사용할 수 없음
  - 준비한 모델(conv_base)위에 Dense층을 쌓아 확장 → 입력 데이터에서 end-to-end로 전체 모델을 실행. 모델에 노출된 모든 입력 이미지가 매번 convnet을 통과하므로 data augmentation 사용 가능. 하지만 그렇기 때문에 첫번째 보다 훨씬 많은 비용이 듬

Transfer Learning

Fast Feature Extraction without Data Augmentation

```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = './datasets/cats_and_dogs_small'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

datagen = ImageDataGenerator(rescale=1./255)
batch_size = 20
```

Transfer Learning

Fast Feature Extraction without Data Augmentation

```
def extract_features(dir, sample_count):
  features = np.zeros(shape=(sample_count, 4, 4, 512))
  labels = np.zeros(shape=(sample_count))
  generator = datagen.flow_from_directory(
      dir,
      target_size = (150,150),
      batch_size = batch_size,
      class_mode = ‘binary'
  )
  i = 0
  for inputs_batch, labels_batch in generator:
    features_batch = conv_base.predict(inputs_batch)
    features[i*batch_size: (i+1)*batch_size] = features_batch
    labels[i*batch_size: (i+1)*batch_size] = labels_batch
    i += 1
    if i*batch_size >= sample_count:
      break
  return features, labels

train_features, train_labels = extract_features(train_dir, 2000)
validation_features, validation_labels = extract_features(validation_dir, 1000)
test_features, test_labels = extract_features(test_dir, 1000)
```

Transfer Learning

Fast Feature Extraction without Data Augmentation

```
train_features = np.reshape(train_features, (2000, 4*4*512))
validation_features = np.reshape(validation_features, (1000, 4*4*512))
test_features = np.reshape(test_features, (1000, 4*4*512))
```

Transfer Learning

Fast Feature Extraction without Data Augmentation

```
from keras.layers import Dense, Dropout
from keras import models
from keras import optimizers

model = models.Sequential()
model.add(Dense(256, activation='relu', input_dim=4*4*512))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=optimizers.RMSprop(lr=2e-5),
              loss='binary_crossentropy',
              metrics=['acc']
              )
history = model.fit(
    train_features,
    train_labels,
    epochs=30,
    batch_size=20,
    validation_data=(validation_features, validation_labels)
)
model.save('cats_and_dogs_small_3.h5')
```

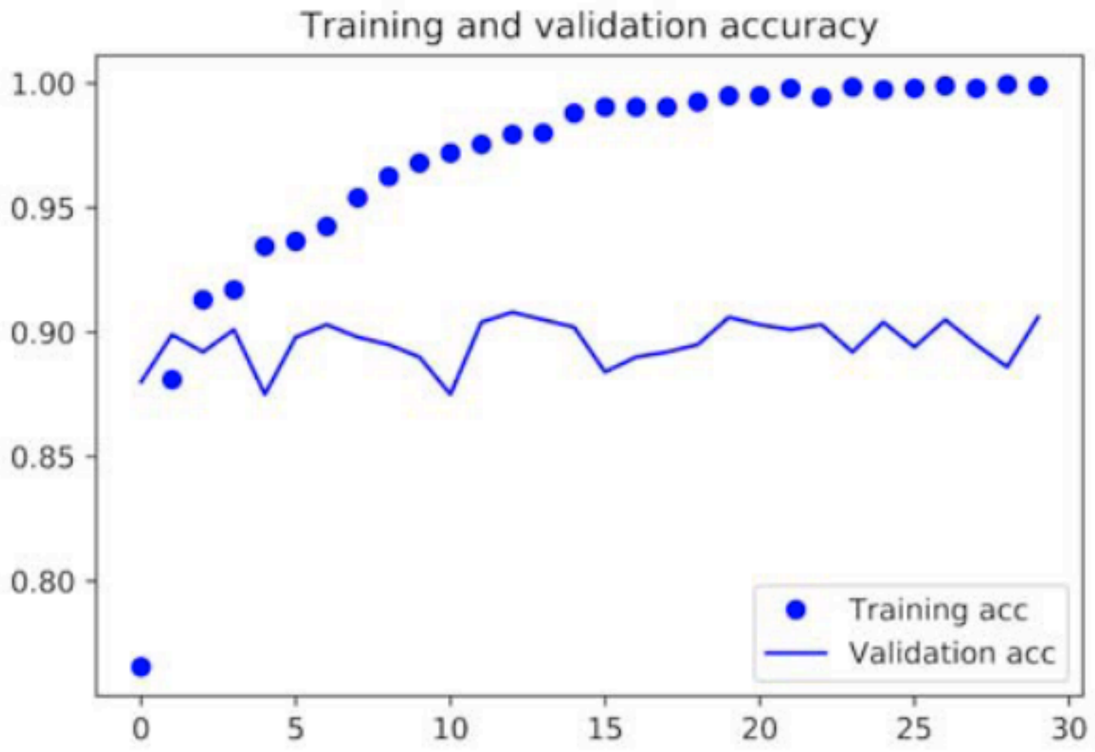
Transfer Learning

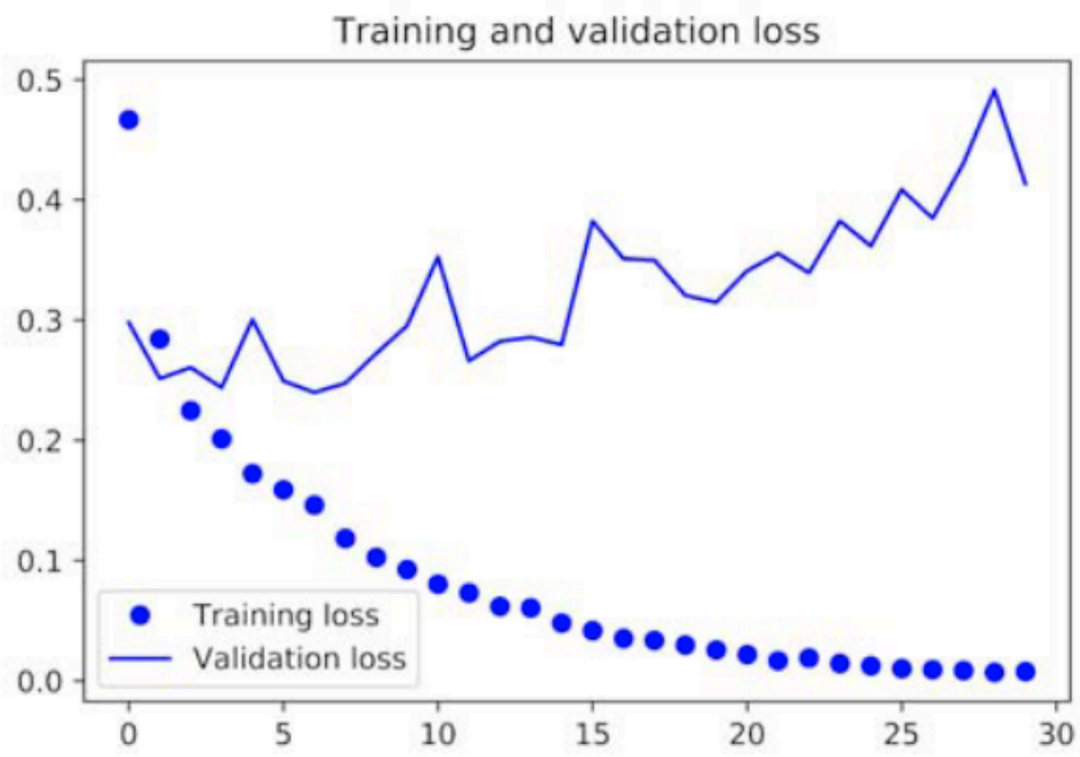
Fast Feature Extraction without Data Augmentation

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

Transfer Learning

Fast Feature Extraction without Data Augmentation





Transfer Learning

Feature Extraction with Data Augmentation

- 훨씬 느리고 비용이 많이 들지만 훈련하는 동안 data augmentation을 사용할 수 있음
- conv_base 모델을 확장하고 입력 데이터를 사용하여 end-to-end로 실행
- 이 기법은 연산비용이 매우 크기 때문에 CPU에는 적용하기 힘듬

```
from keras import models
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = models.Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Transfer Learning

Feature Extraction with Data Augmentation

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

Transfer Learning

Feature Extraction with Data Augmentation

- 모델을 컴파일하고 훈련하기 전에 conv_base를 동결하는것이 매우 중요 ← 훈련하는 동안 가중치가 업데이트 되지 않도록 막음

- Keras에서는 trainable 속성을 False로 설정하여 네트워크를 동결

```
>>> print  ('conv_base를 동경하기 전 훈련되는 가중치의 수: ',  len(model.trainable_weights))
conv_base를 동경하기 전 훈련되는 가중치의 수:  30
>>> conv_base.trainable = False
>>> print  ('conv_base를 동경한 후 훈련되는 가중치의 수: ',  len(model.trainable_weights))
conv_base를 동경한 후 훈련되는 가중치의 수:  4
```

- 변경사항을 적용하려면 모델을 컴파일 → 컴파일 후에 trainable 속성을 변경하면 반드시 모델을 다시 컴파일하지 않으면 변경사항이 적용되지 않음

Transfer Learning

Feature Extraction with Data Augmentation

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

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    )
test_datagen = ImageDataGenerator(rescale=1./255)     #  검증 데이터는 증식되면 안됨

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150,150),
    batch_size=32,
    class_mode='binary'
)
```

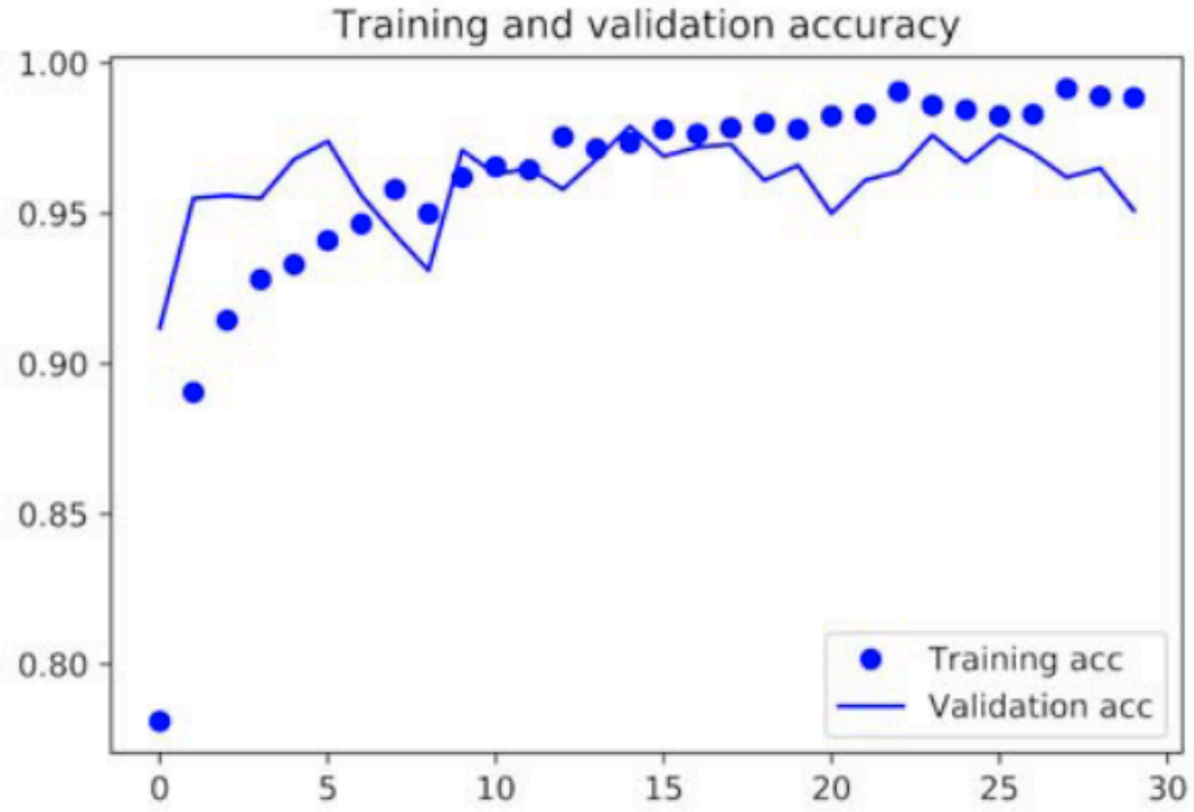
Transfer Learning

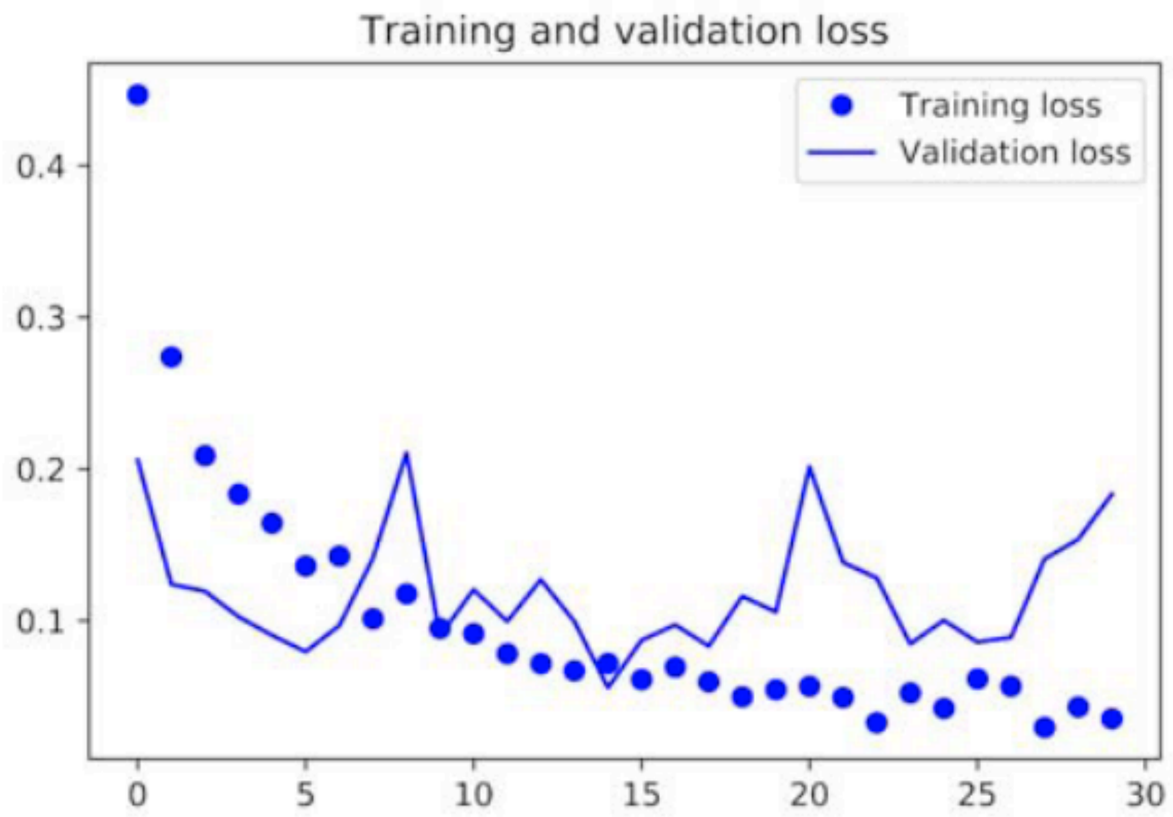
Feature Extraction with Data Augmentation

```
validation_generator = test_datagen.flow_from_directory(
    validation_dir,
    target_size=(150,150),
    batch_size=32,
    class_mode='binary'
)

model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=2e-5),metrics=['acc'])

history = model.fit_generator(
    train_generator,
    steps_per_epoch=100,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=50,
    verbose=2,
)
model.save('cats_and_dogs_small_5.h5')
```

Transfer Learning

Feature Extraction with Data Augmentation

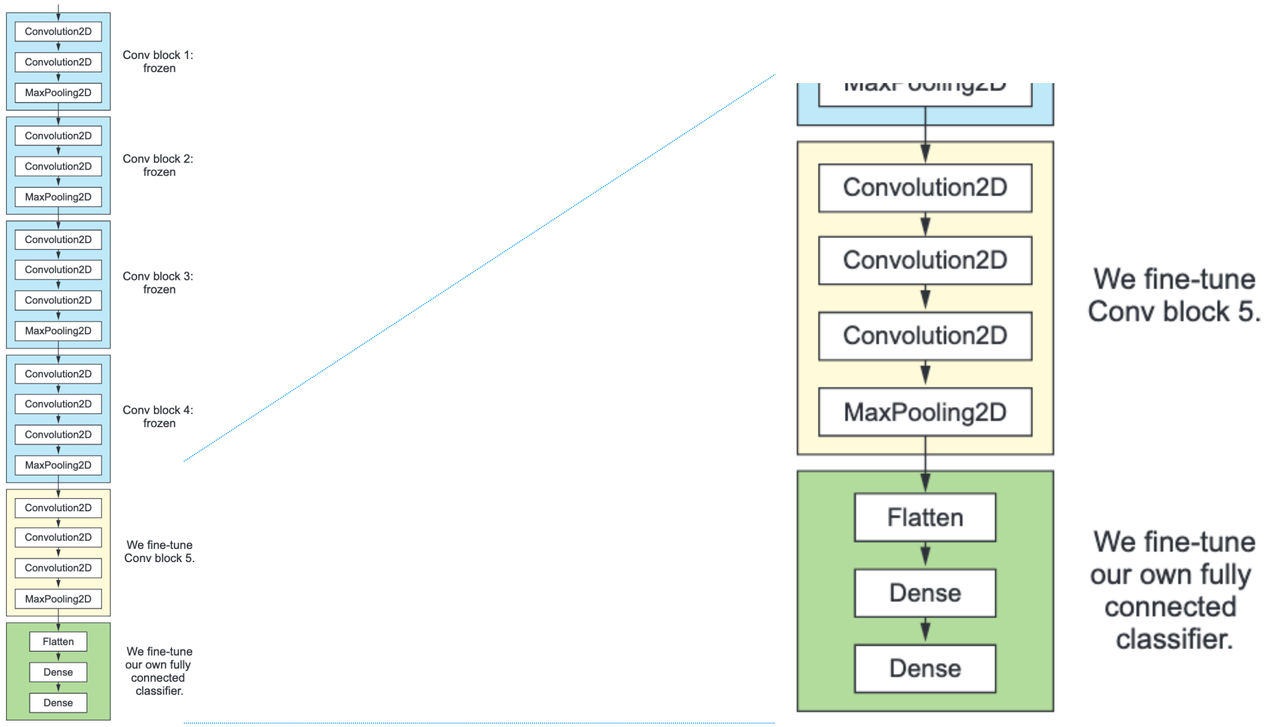




Transfer Learning

Fine Tuning

- 특성추출에 사용했던 동결모델의 상위 층 몇개를 동결에서 해제하고 모델에 새로 추가한 층(여기서는 완전연결분류기)와 함께 훈련



Transfer Learning

Fine Tuning

- Convnet 하위층들은 좀 더 일반적이고 재사용 가능한 특성들을 인코딩. 상위층들은 좀 더 특화된 특성을 인코딩 → 새로운 문제에 재활용하도록 수정이 필요한 것은 구체적인 특성이므로 이들을 미세 조정하는 것이 유리

- 훈련해야 할 파라미터가 많을수록 과대적합의 위험이 커짐

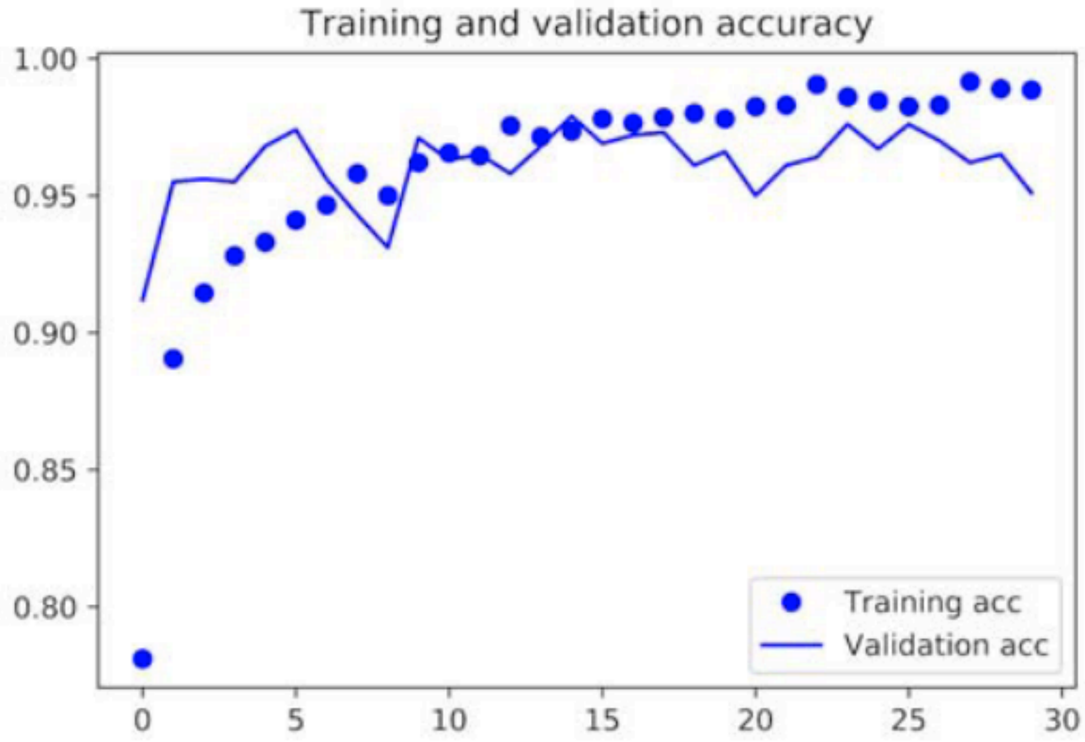
Transfer Learning

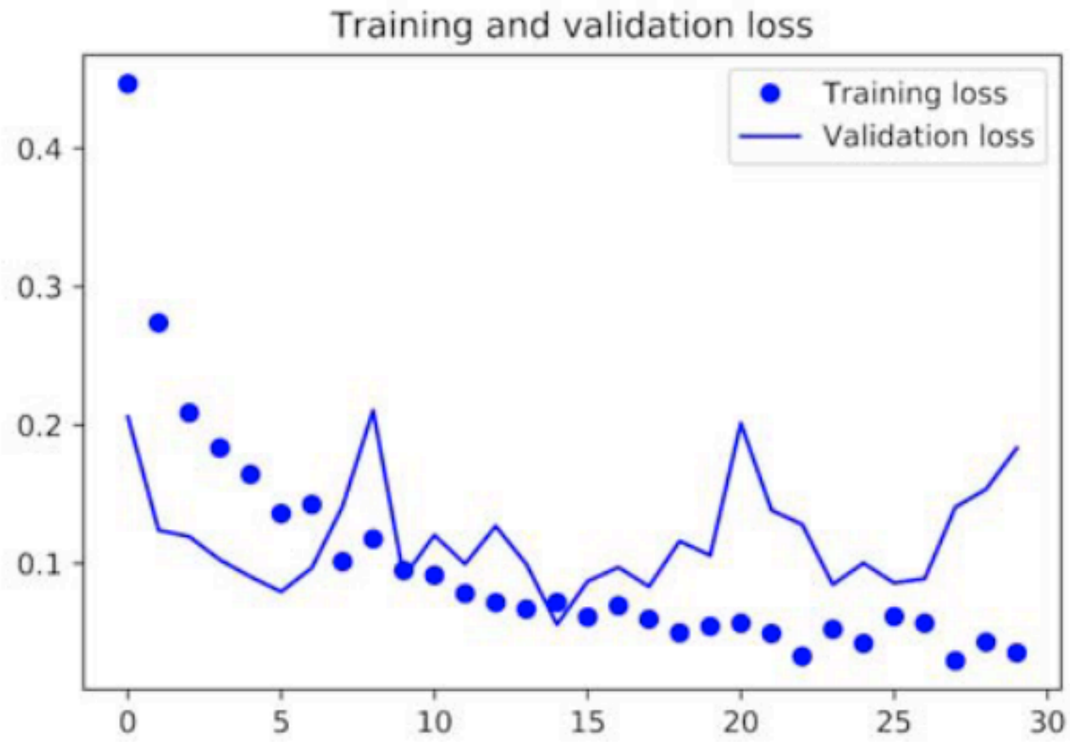
Fine Tuning

```
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
  if layer.name == 'block5_conv1':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False

model.compile(
    loss='binary_crossentropy',
    optimizer=optimizers.RMSprop(lr=1e-5),
    metrics=['acc']
)
history = model.fit_generator(
    train_generator,
    steps_per_epochs=100,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=50
)
```

Transfer Learning

Fine Tuning





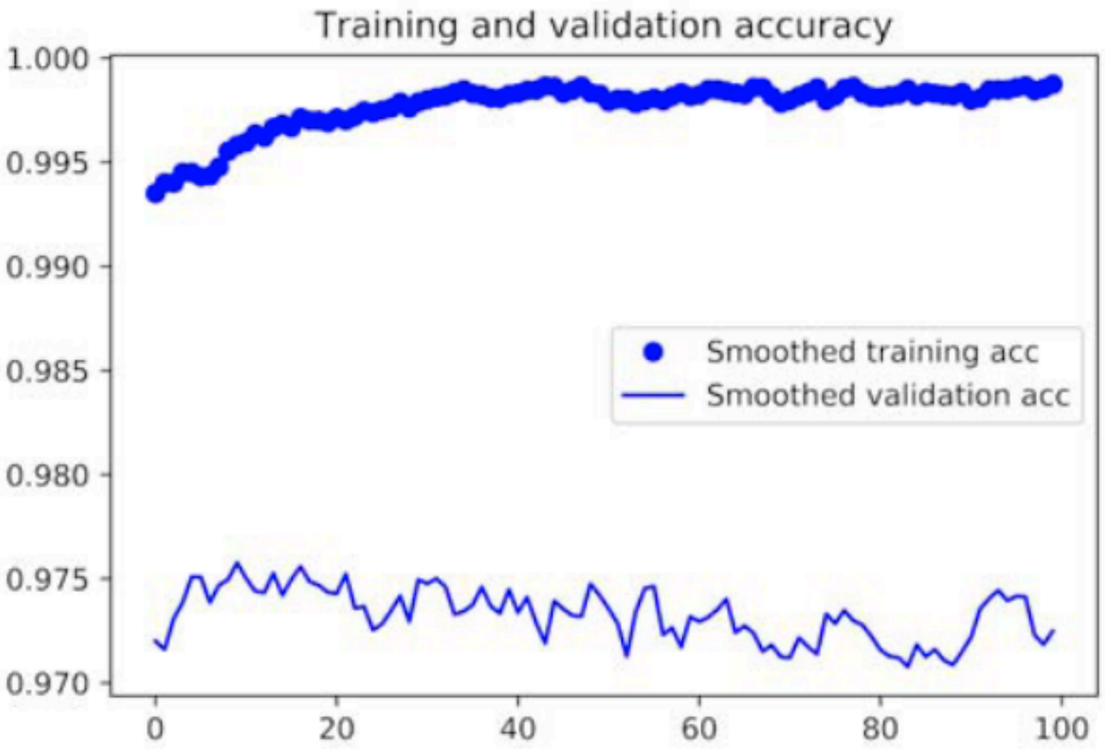
Transfer Learning

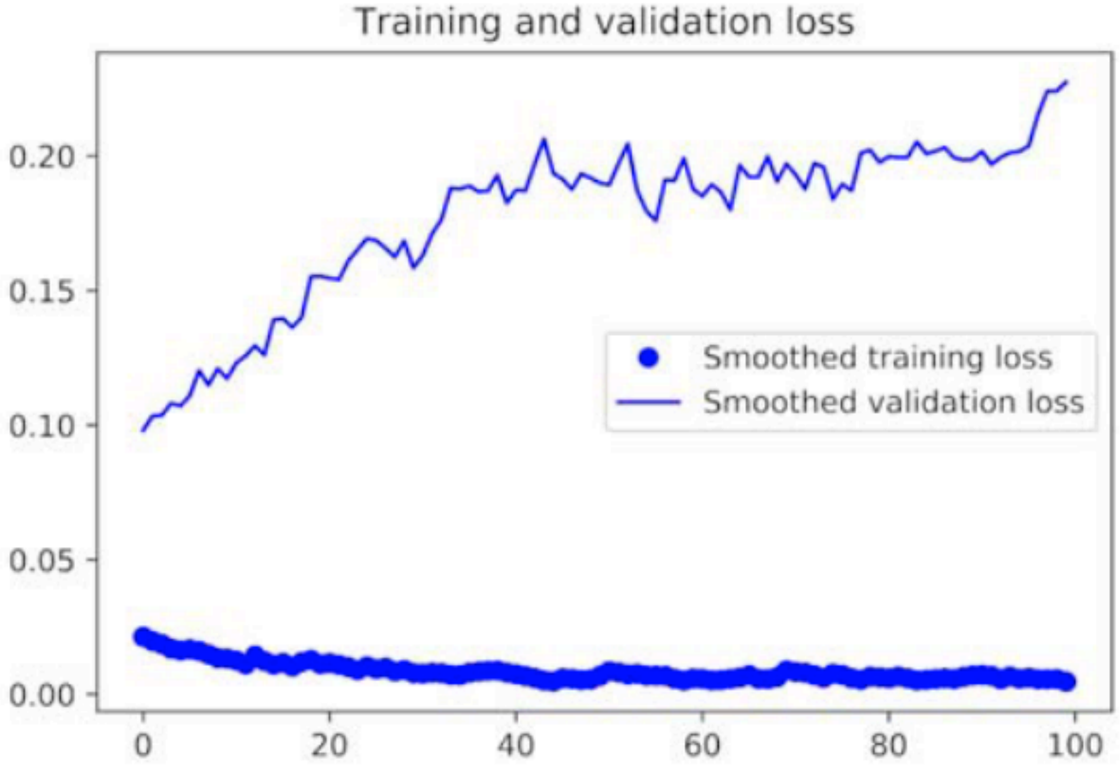
Fine Tuning

```
def smooth_curve(points, factor=0.8):
  smoothed_points=[]
  for point in points:
    if smoothed_points:
      previous = smoothed_points[-1]
      smoothed_points.append(previous*factor+point*(1-factor))
    else:
      smoothed_points.append(point)
  return smoothed_points

plt.plot(epochs, smooth_curve(acc), 'bo', label='Smoothed training acc')
plt.plot(epochs, smooth_curve(val_acc), 'b', label='Smoothed validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, smooth_curve(loss), 'bo', label='Smoothed training loss')
plt.plot(epochs, smooth_curve(val_loss), 'b', label='Smoothed validation loss')
plt.title('Training and validation accuracy')
plt.legend()
plt.show()
```

Transfer Learning

Fine Tuning





Transfer Learning

Fine Tuning

```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150,150),
    batch_size=20,
    class_mode='binary'
)

test_loss, test_acc = model.evaluate_generator(test_generatorm steps=50)
print('test acc:', test_acc)
```