

라 가 영

Using Real-world Images



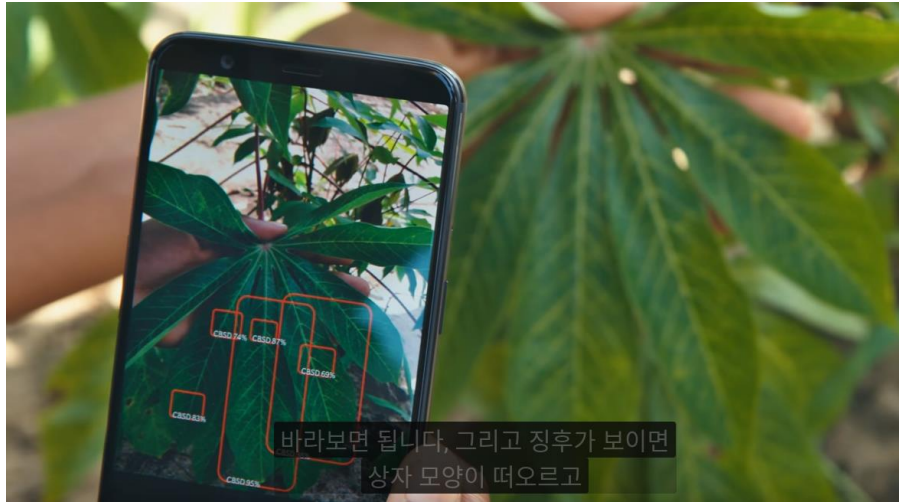
© DreamWorks II Distribution Co., LLC All Rights Reserved.

A conversation with Andrew Ng



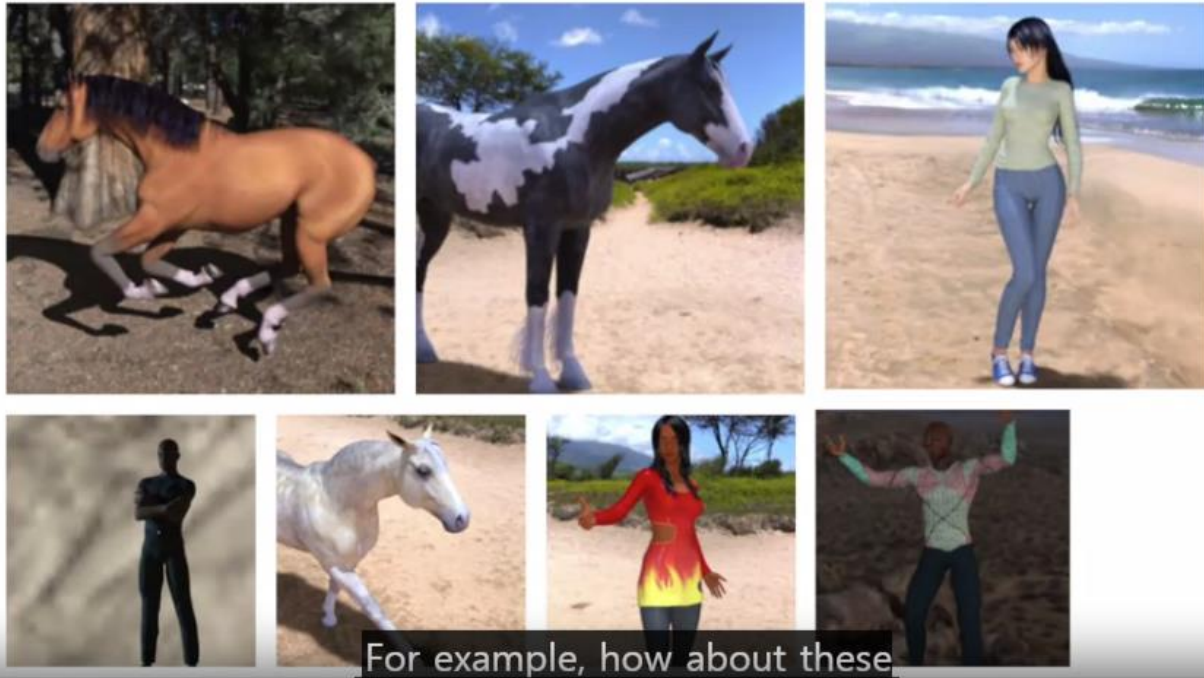
- ▶ 앞에서의 예제는 모든 사진이 28×28 크기이며, 피사체가 중앙에 위치함 \rightarrow 학습에 용이함
- ▶ 반면, 실제 이미지는 피사체가 가운데 없을 수도 있으며, 크기 또한 서로 다르고 생김새와 색상이 다양함 \rightarrow 학습에 제한점
 - ▶ Ex) 말 이미지의 경우, 움직이는 역동적인 모습일 수 있으며, 사람 이미지의 경우, 남자이거나 여자 일 수 있음
- ▶ 이러한 제한점에도 불구하고 많은 현실 문제에서 convolutional network가 사용되고 있음
 - ▶ Ex) 무인 자동차가 카메라 이미지를 활용하여 다른 차량이나 보행자를 피하는 경우 ...

Explore an impactful, real-world solution



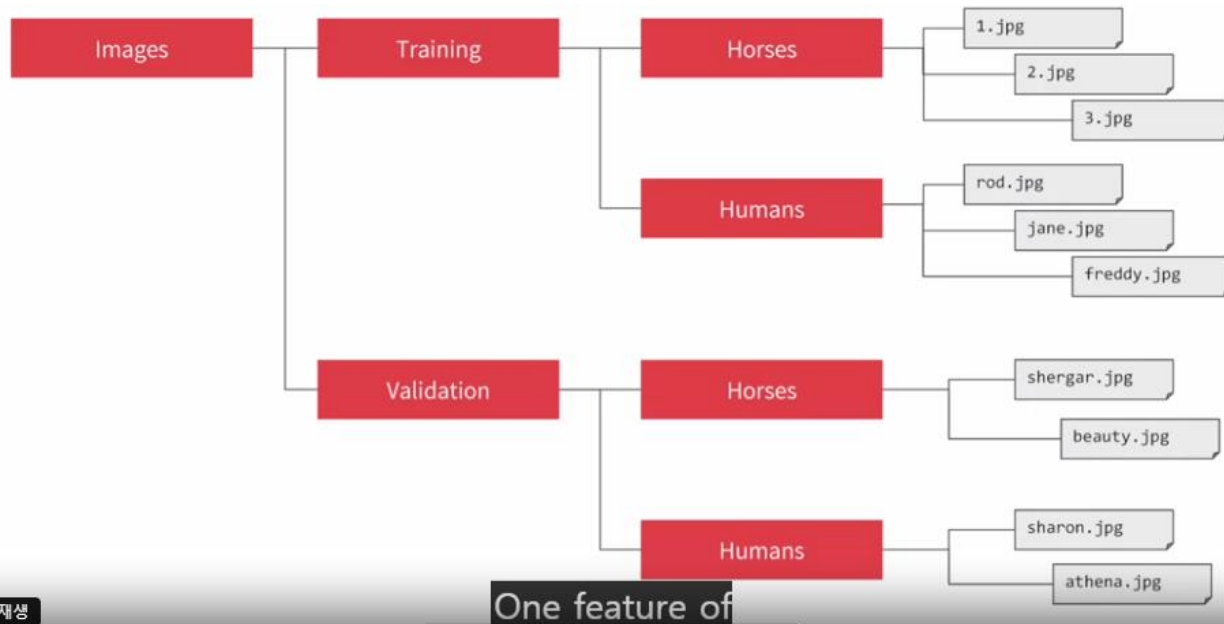
- ▶ 펜실베니아 주립대학교의 플랜트 빌리지 (<https://plantvillage.psu.edu/>)와 국제 열대농업 연구소(<http://www.iita.org/>)의 연구자들이 개발한 어플
- ▶ 간단한 카메라 어플 조작으로 농부들이 스스로 카사바 작물에 영향을 미치는 질병을 탐지할 수 있음
- ▶ 기계학습 기법과 tensorflow를 활용
- ▶ <https://www.youtube.com/watch?v=NIpS-DhayQA>

Understanding ImageGenerator



- ▶ 말과 인간을 구분하려고 하는 경우
 - ▶ 각 사진은 서로 다른 종횡비를 가지고 있으며 피사체의 위치가 제각각 임
 - ▶ 레이블 또한 되어 있지 않음
- ▶ Tensorflow의 API 중 하나인 **ImageGenerator**를 사용하면 손쉽게 해결 가능!

Understanding ImageGenerator



▶ 이미지를 입력하면 이미지가 포함된 디렉토리를 기반으로 **자동으로 라벨을 생성함**

▶ 각각의 입력된 이미지는 training set 혹은 validation set의 디렉토리로 구분 됨

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

▶ Keras에서는 `Keras.preprocessing.image`로 사용 가능

Understanding ImageGenerator

```
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(300, 300),
    batch_size=128,
    class_mode='binary')
```

- ▶ imageGenerator의 인스턴스화
- ▶ 데이터를 정규화하기 위해 rescale을 지정해줌
- ▶ 인스턴스를 호출하여 사용
- ▶ 해당하는 디렉토리의 이미지가 로딩됨
 - ▶ 사용할 때는 generator가 아닌 directory를 직접 호출 해야함
- ▶ 매개 변수에서 이미지 사이즈와 모양은 모두 동일 해야함
 - ▶ 일관성을 위해 자동으로 크기가 조절됨
- ▶ 말과 인간으로 구분하기 때문에 binary

Defining a ConvNet to use complex images

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu',
                           input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

- ▶ 이전 코드와 비슷하지만, convolution과 pooling을 3번 한다는 점, 이미지 크기가 커졌다는 점이 다름
- ▶ 또한, 컬러 이미지이기 때문에 픽셀당 3바이트가 존재
 - ▶ 빨강, 초록, 파랑
- ▶ Sigmoid 함수를 사용하여 0 혹은 1로 구분
 - ▶ Softmax 뉴런을 추가하여 구분 가능

Training the ConvNet with fit_generator

```
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['acc'])
```

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=8,
    epochs=15,
    validation_data=validation_generator,
    validation_steps=8,
    verbose=2)
```

- ▶ 앞선 패션 예제에서는 cross_entropy를 사용하였지만 이번에는 둘 중 하나로 분류하기 때문에, binary_crossentropy를 사용
- ▶ Learning rate를 조절하면서 성능 실험 가능
- ▶ Model.fit이 아닌 model.fit_generator를 사용
- ▶ Training set은 1024개의 이미지를 128개씩 8번에 걸쳐서 로딩
- ▶ Validation set은 256개의 이미지를 32개씩 8번에 걸쳐서 로딩

Training the ConvNet with fit_generator

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

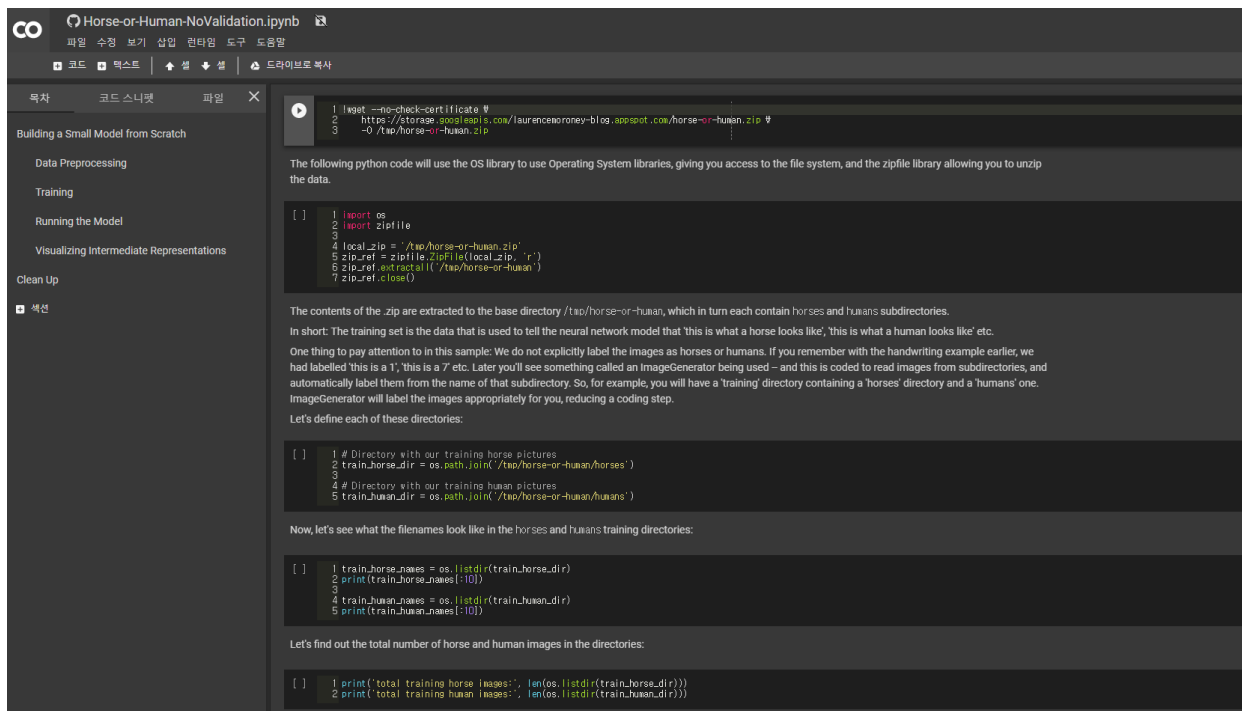
for fn in uploaded.keys():

    # predicting images
    path = '/content/' + fn
    img = image.load_img(path, target_size=(300, 300))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    print(classes[0])
    if classes[0]>0.5:
        print(fn + " is a human")
    else:
        print(fn + " is a horse")
```

- ▶ Colab에서 업로드할 하나 이상의 이미지를 선택함
- ▶ 업로드 키를 누른 상태에서 for문이 반복됨
- ▶ Target_size가 input image size와 동일한지 확인 할 것

Experiment with the horse or human classifier



```
1 !wget --no-check-certificate \
2   https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip \
3   -O /tmp/horse-or-human.zip

The following python code will use the OS library to use Operating System libraries, giving you access to the file system, and the zipfile library allowing you to unzip the data.

[ ]: 1 import os
      2 import zipfile
      3
      4 local_zip = '/tmp/horse-or-human.zip'
      5 zip_ref = zipfile.ZipFile(local_zip, 'r')
      6 zip_ref.extractall('/tmp/horse-or-human')
      7 zip_ref.close()

The contents of the .zip are extracted to the base directory /tmp/horse-or-human, which in turn each contain horses and humans subdirectories.
In short: The training set is the data that is used to tell the neural network model that 'this is what a horse looks like', 'this is what a human looks like' etc.

One thing to pay attention to in this sample: We do not explicitly label the images as horses or humans. If you remember with the handwriting example earlier, we had labelled 'this is a 1', 'this is a 7' etc. Later you'll see something called an ImageGenerator being used - and this is coded to read images from subdirectories, and automatically label them from the name of that subdirectory. So, for example, you will have a 'training' directory containing a 'horses' directory and a 'humans' one. ImageGenerator will label the images appropriately for you, reducing a coding step.
Let's define each of these directories:

[ ]: 1 # Directory with our training horse pictures
      2 train_horse_dir = os.path.join('/tmp/horse-or-human/horses')
      3
      4 # Directory with our training human pictures
      5 train_human_dir = os.path.join('/tmp/horse-or-human/humans')

Now, let's see what the filenames look like in the horses and humans training directories:

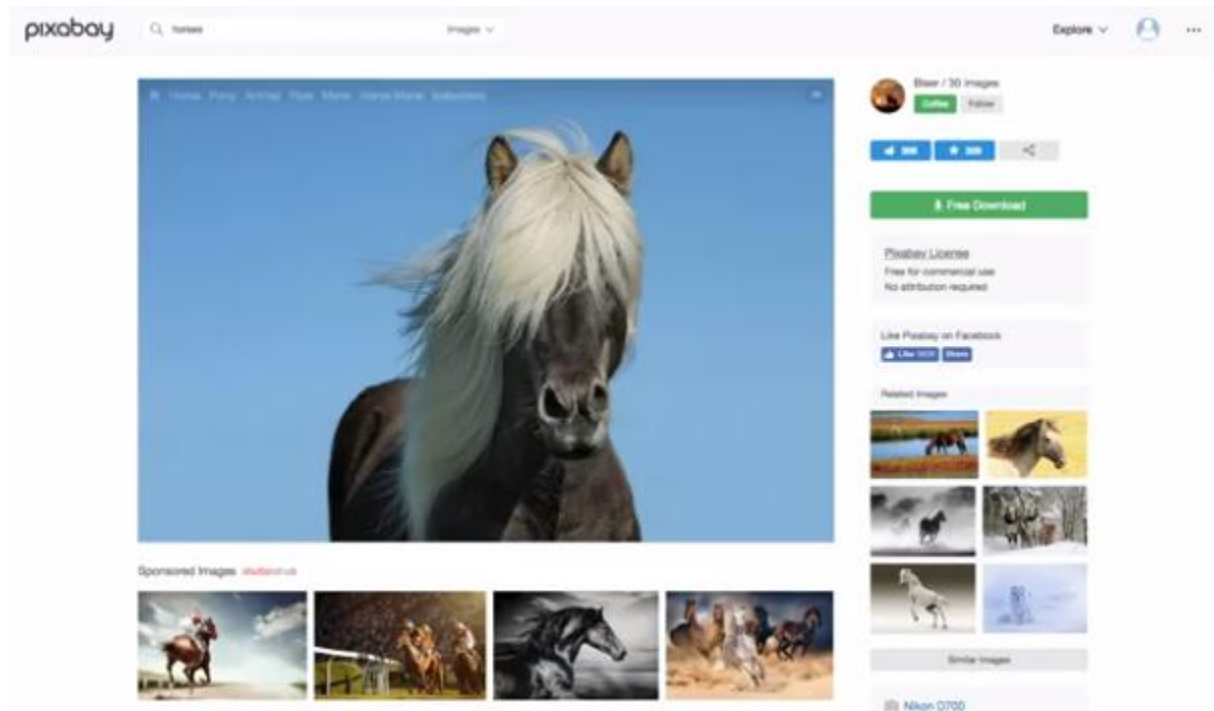
[ ]: 1 train_horse_names = os.listdir(train_horse_dir)
      2 print(train_horse_names[:10])
      3
      4 train_human_names = os.listdir(train_human_dir)
      5 print(train_human_names[:10])

Let's find out the total number of horse and human images in the directories:

[ ]: 1 print('total training horse images:', len(os.listdir(train_horse_dir)))
      2 print('total training human images:', len(os.listdir(train_human_dir)))
```

- ▶ 강의에서 나왔던 코드와 hidden layer 수나 epoch 수가 다름. 스스로 수정해보면서 실습해 보길!
- ▶ <https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/Course%20-%20Part%208%20-%20Lesson%202%20-%20Notebook.ipynb>

Adding automatic validation to test accuracy



- ▶ 트레이닝에 사용한 말들은 갈색이었으나, 하얀색 말이나 앞머리가 있는 말 이미지 등을 업로드하여 유효성을 확인함
- ▶ 사람 이미지 또한 등돌린 사람이나 날개 달린 사람, 말과 함께 있는 사람 이미지를 업로드하여 유효성을 확인함
 - ▶ 종종 틀림 ㅜㅜ

Get hands-on and use validation

```
Course 2 - Part 2 - Lesson 3 - Notebook.ipynb
파일 수정 보기 삽입 런타임 도구 도움말
코드 텍스트 셀 드라이브 복사

1 wget --no-check-certificate #
2   https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip #
3   -O /tmp/horse-or-human.zip

--2019-07-10 00:45:46-- https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.141.128, 2007:f8b0:400c:c06::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.141.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 149574867 (143M) [application/zip]
Saving to: '/tmp/horse-or-human.zip'

/tmp/horse-or-human 100%[=====] 142.65M 109MB/s in 1.4s

2019-07-10 00:45:48 (105 MB/s) - '/tmp/horse-or-human.zip' saved [149574867/149574867]

1 wget --no-check-certificate #
2   https://storage.googleapis.com/laurencemoroney-blog.appspot.com/validation-horse-or-human.zip #
3   -O /tmp/validation-horse-or-human.zip

The following python code will use the OS library to use Operating System libraries, giving you access to the file system, and the zipfile library allowing you to unzip the data.

1 import os
2 import zipfile
3
4 local_zip = '/tmp/horse-or-human.zip'
5 zip_ref = zipfile.ZipFile(local_zip, 'r')
6 zip_ref.extractall('/tmp/horse-or-human')
7
8 local_zip = '/tmp/validation-horse-or-human.zip'
9 zip_ref = zipfile.ZipFile(local_zip, 'r')
10 zip_ref.extractall('/tmp/validation-horse-or-human')
11 zip_ref.close()

The contents of the .zip are extracted to the base directory /tmp/horse-or-human, which in turn each contain horses and humans subdirectories.
In short: The training set is the data that is used to tell the neural network model that 'this is what a horse looks like', 'this is what a human looks like' etc.
One thing to pay attention to in this sample: We do not explicitly label the images as horses or humans. If you remember with the handwriting example earlier, we had labelled 'this is a 1', 'this is a 7' etc. Later you'll see something called an ImageGenerator being used - and this is coded to read images from subdirectories, and automatically label them from the name of that subdirectory. So, for example, you will have a 'training' directory containing a 'horses' directory and a 'humans' one. ImageGenerator will label the images appropriately for you, reducing a coding step.
Let's define each of these directories:
```

- ▶ 직접 찾은 이미지를 업로드하며 실습해보자!!
- ▶ 이미지 사이즈를 300*300으로 실습하였는데, 150*150으로 바꿀 경우, 결과가 달라질 수 있음
- ▶ <https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/Course%20-%20Part%208%20-%20Lesson%204%20-%20Notebook.ipynb>