11-6-2018 Facial Recognition Keras

https://github.com/krasserm/face-recognition

Use Oxford University paper as the base line to build.

11-10-2018 Two Million Images for Facial Recognition Oxford University

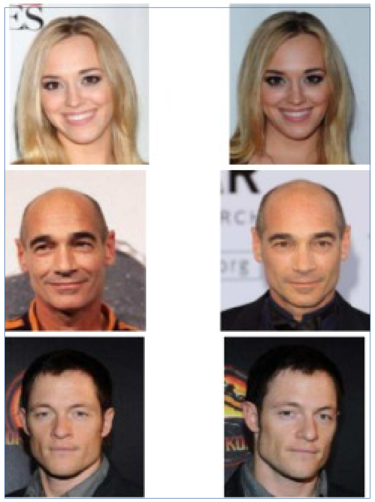
Shallow techniques (with no deep learning) start with local image descriptors such as SIFT, LBP, HOG to form overall face descriptor by pooling, for example the Fisher Vector. **Deep techniques**, e.g., CNN (convolutional neural networks) feature extractor, example as DeepFace using a dataset of 4 million examples spanning 4000 unique identities, in addition a siamese network architecture, where the same CNN is applied to pairs of faces to obtain descriptors that are then compared using the Euclidean distance.

DeepFace achieved the best performance on the Labelled Faces in the Wild (LFW; [8]) benchmark as well as the Youtube Faces in the Wild (YFW; [32]) benchmark. The authors extended this work by increasing the size of the dataset by 2 orders of magnitude with ***10 million identities and 50 images per identity***.

Google used a dataset of **200 million face identities** and 800 million image face pairs.

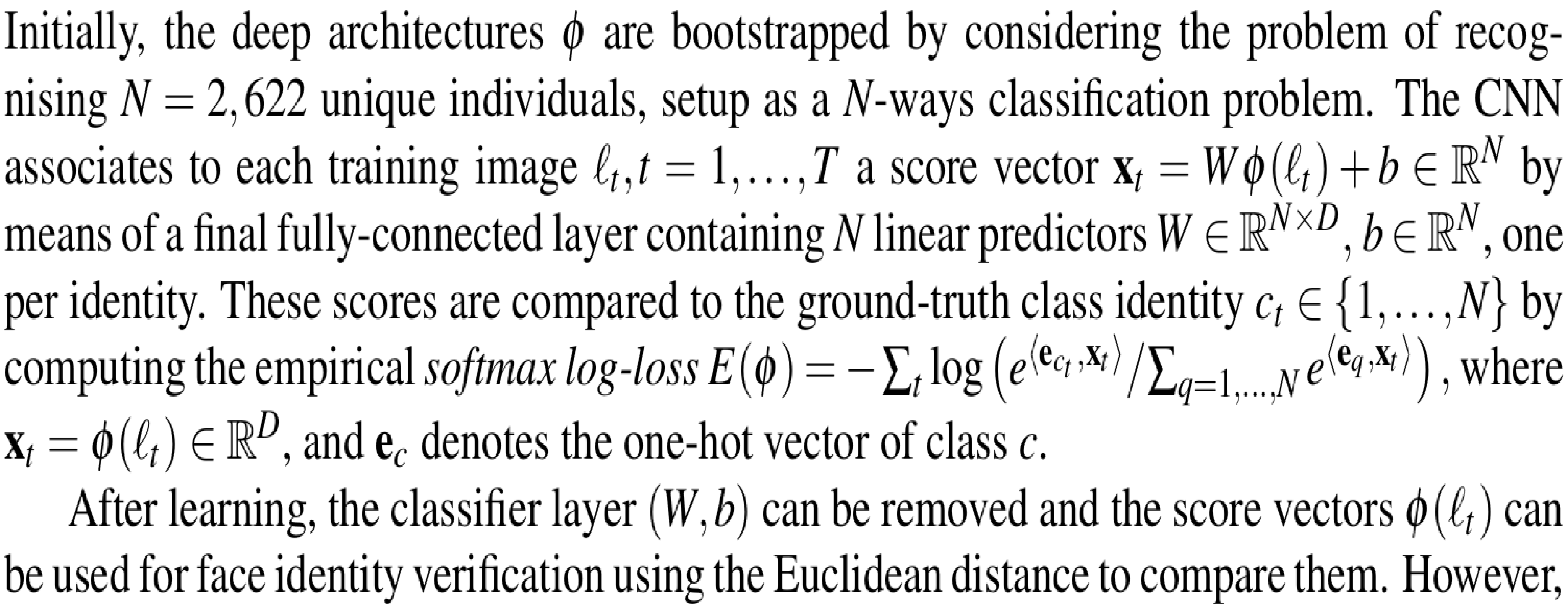
Table 1. Image test result

Method	Images	Networks	Acc.
Fisher Vector Faces [21]	-	-	93.10
DeepFace [29]	4M	3	97.35
Fusion [30]	500M	5	98.37
DeepID-2,3		200	99.47
FaceNet [17]	200M	1	98.87
FaceNet [17] + Alignment	200M	1	99.63
Oxford	2.6M	1	98.95



Harry Li, Ph.D. 2018

11-10-2018 Theory of Deep Techniques for Facial Recognition Oxford University

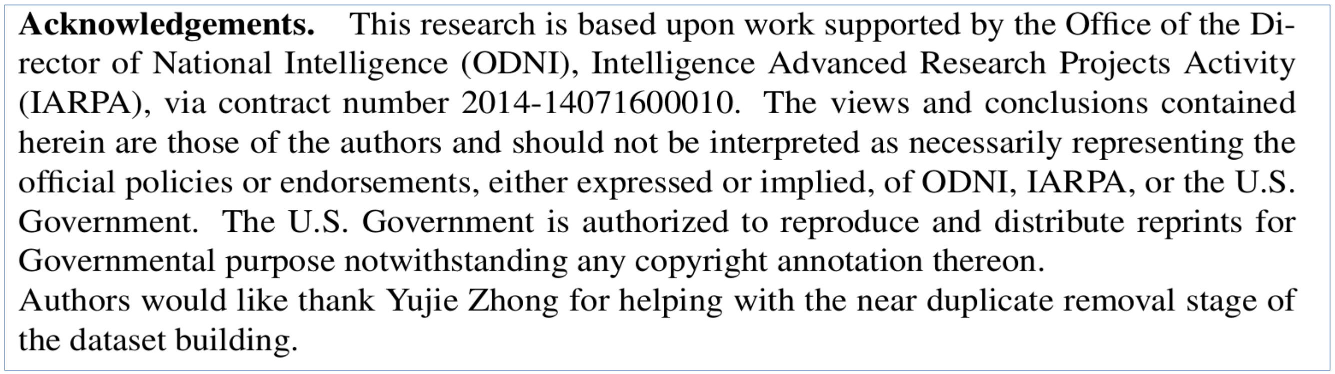


Initially, the deep architectures ϕ are bootstrapped by considering the problem of recognising $N = 2,622$ unique individuals, setup as a N-ways classification problem. The CNN associates to each training image $\ell_t, t = 1, \ldots, T$ a score vector $\mathbf{x}_t = W\phi(\ell_t) + b \in \mathbb{R}^N$ by means of a final fully-connected layer containing N linear predictors $W \in \mathbb{R}^{N \times D}$, $b \in \mathbb{R}^N$, one per identity. These scores are compared to the ground-truth class identity $c_t \in \{1, \ldots, N\}$ by computing the empirical *softmax log-loss* $E(\phi) = -\sum_t \log\left(e^{\langle \mathbf{e}_{c_t}, \mathbf{x}_t \rangle} / \sum_{q=1,\ldots,N} e^{\langle \mathbf{e}_q, \mathbf{x}_t \rangle}\right)$, where $\mathbf{x}_t = \phi(\ell_t) \in \mathbb{R}^D$, and $\mathbf{e}_c$ denotes the one-hot vector of class c.

After learning, the classifier layer (W, b) can be removed and the score vectors $\phi(\ell_t)$ can be used for face identity verification using the Euclidean distance to compare them. However,

The input to all networks is a face image of size 224 × 224

From the authors of the Oxford paper



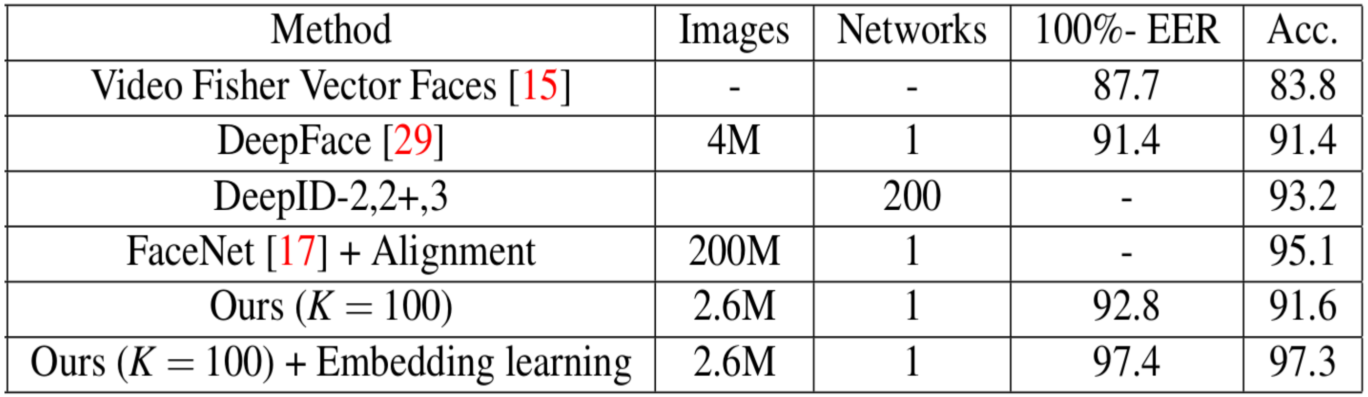
Acknowledgements. This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract number 2014-14071600010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.
Authors would like thank Yujie Zhong for helping with the near duplicate removal stage of the dataset building.

11-10-2018 Data Sets, Hardware and Results for Facial Recognition

Reference: Oxford University

Labeled Faces in the Wild dataset (**LFW**) [8]. It contains 13,233 images with 5,749 identities, and is the standard benchmark for automatic face verification.

The second dataset is YouTube Faces (**YTF**) [32]. It contains 3,425 videos of 1,595 people collected from YouTube, with an average of 2 videos per identity, and is a standard benchmark for face verification in video.



Method	Images	Networks	100%- EER	Acc.
Video Fisher Vector Faces [15]	-	-	87.7	83.8
DeepFace [29]	4M	1	91.4	91.4
DeepID-2,2+,3		200	-	93.2
FaceNet [17] + Alignment	200M	1	-	95.1
Ours ($K = 100$)	2.6M	1	92.8	91.6
Ours ($K = 100$) + Embedding learning	2.6M	1	97.4	97.3

Table 2. Video test result

Implementated on the MATLAB toolbox MatCon-vNet [31] linked to NVIDIA CuDNN libraries to accelerate training. All experiments were carried on NVIDIA Titan Black GPUs with 6GB memory, using 4 GPUs together due to the very significant memory footprint.

NVIDIA TITAN V VOLTA GPU 12GB HBM2 GRAPHICS VIDEO CARD 900-1G500-2500-000-NEW

Harry Li, Ph.D. 2018

11-10-2018 Build CNN Keras Model for Facial Recognition

https://aboveintelligent.com/face-recognition-with-keras-and-opencv-2baf2a83b799

VGG-Face model is the same as the VGG16 model used to identity 1000 classes of object in the ImageNet competition. The VGG16 name simply states the model originated from the **Visual Geometry Group** and that it was 16 trainable layers. The main difference between the VGG16-ImageNet and VGG-Face model is the set of calibrated weights as the training sets were different.

Code sample:

https://github.com/mzaradzki/neuralnets/tree/master/vgg_faces_keras

11-10-2018 AWS Deep Learning AMIs

https://aws.amazon.com/rekognition/

Amazon Rekognition
Easily add intelligent image and video analysis to your applications.

AWS the infrastructure and tools to accelerate deep learning in the cloud to quickly launch Amazon EC2 instances pre-installed with popular deep learning frameworks such as Apache MXNet and Gluon, TensorFlow, Microsoft Cognitive Toolkit, Caffe, Caffe2, Theano, Torch, PyTorch, Chainer, and Keras.

To train sophisticated, custom AI models and new algorithms.

Amazon EC2 GPU or CPU instances.

11-10-2018 Facial Recognition Celebrity Recognition

https://aws.amazon.com/rekognition/

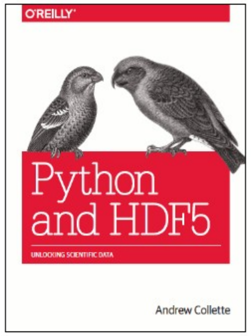
Amazon Rekognition

Easily add intelligent image and video analysis to your applications.

10-30-2018 Save Trained Network with h5py

https://www.quora.com/How-do-I-save-a-convolution-neural-network-model-after-training-I-am-working-in-Python

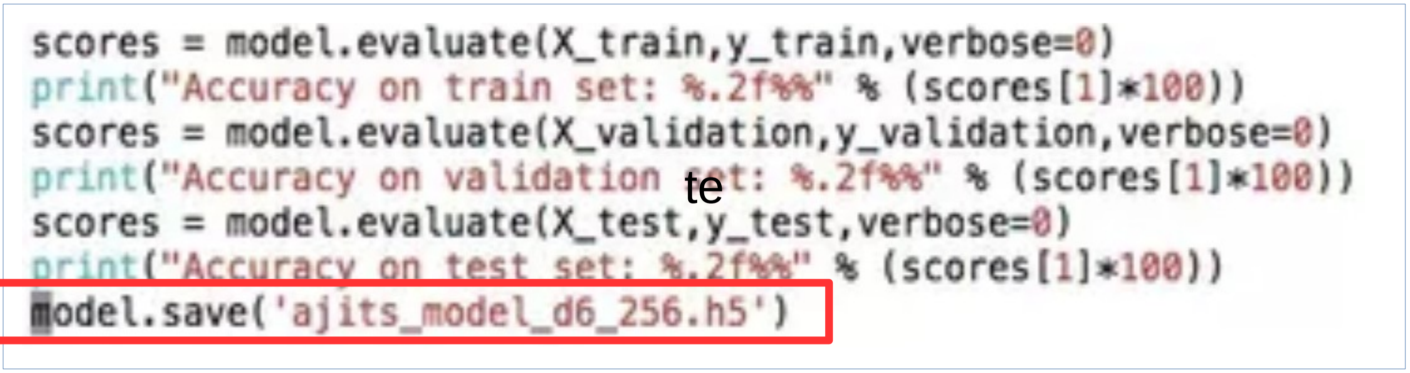
Keras for saving a model is just one line of code after training



http://www.h5py.org/

Model.save('harry_convnet_mnist.h5')

Example:



```
scores = model.evaluate(X_train,y_train,verbose=0)
print("Accuracy on train set: %.2f%%" % (scores[1]*100))
scores = model.evaluate(X_validation,y_validation,verbose=0)
print("Accuracy on validation set: %.2f%%" % (scores[1]*100))
scores = model.evaluate(X_test,y_test,verbose=0)
print("Accuracy on test set: %.2f%%" % (scores[1]*100))
model.save('ajits_model_d6_256.h5')
```

te

Example: 7-1convnets-NumeralDet-saveTrained.py

```
#-------------save trained model---------*
import h5py
model.save('harryTest.h5')
#-end
```

Just add 2 lines of code

Harry Li, Ph.D. 2018

10-30-2018 Load Trained Network

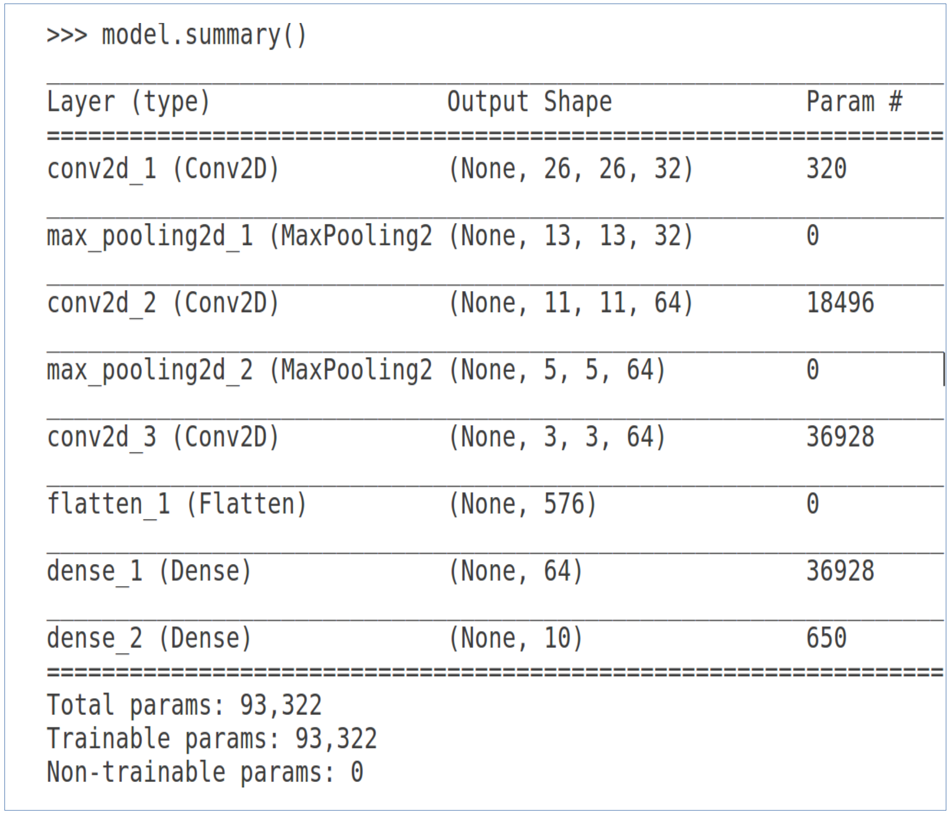
https://stackoverflow.com/questions/35074549/how-to-load-a-model-from-an-hdf5-file-in-keras

Keras for loading a model is just one line of code

If you stored the complete model, not only the weights, in the HDF5 file, then load is as simple as

Example:

```
#----------load trained model-------------------*
from keras.models import load_model
model = load_model('harryTest.h5')
model.summary() #check the model
```



```
>>> model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

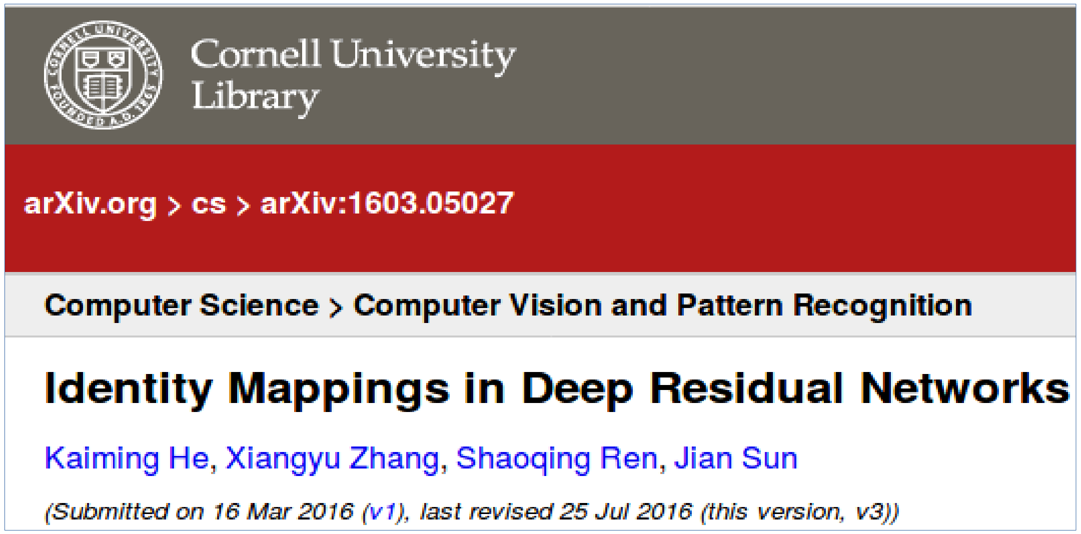
11-6-2018 Keras Supported CNN Architectures

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

- Xception
- InceptionV3
- ResNet50
- VGG16
- VGG19
- MobileNet

ResNet architecture which can be successfully trained at depths of 50-200 for ImageNet and over 1,000 for CIFAR-10

Two major drawbacks with VGGNet: (1) painfully slow to train. (2) Network architecture weights themselves are quite large (in terms of disk/bandwidth). Due to its depth and number of fully-connected nodes, VGG16 is over 533MB and VGG19 is 574MB. This makes deploying VGG a tiresome task.



Cornell University Library

arXiv.org > cs > arXiv:1603.05027

Computer Science > Computer Vision and Pattern Recognition

Identity Mappings in Deep Residual Networks

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

(Submitted on 16 Mar 2016 (v1), last revised 25 Jul 2016 (this version, v3))

https://github.com/KaimingHe/resnet-1k-layers

11-6-2018 Use Keras Supported CNN

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

```
# import the necessary packages
from keras.applications import ResNet50
from keras.applications import InceptionV3
from keras.applications import Xception # TensorFlow ONLY
from keras.applications import VGG16
from keras.applications import VGG19
from keras.applications import imagenet_utils
from keras.applications.inception_v3 import preprocess_input
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import numpy as np
import argparse
import cv2
```

Note: Weights for VGG16 and VGG19 are > 500MB. ResNet weights are ~100MB, while Inception and Xception weights are between 90-100MB. If this is the first time you are running this script for a given network, these weights will be (automatically) downloaded and cached to your local disk. Depending on your internet speed, this may take awhile. However, once the weights are downloaded, they will not need to be downloaded again, allowing subsequent runs of classify_image.py to be much faster.

11-6-2018 Deploy Cats Dogs Detection CNN

```
import time
import keras
keras.__version__
import numpy as np

from keras import models
import cv2
from keras.models import load_model
model = load_model('cats_and_dogs_small_1.h5')
#model.summary() #check the model

image =cv2.imread('1.jpg',cv2.IMWRITE_JPEG_QUALITY)
image = cv2.resize(image,(150,150))
image=np.reshape(image,[1,150,150,3])

time_curr=int(round(time.time()*1000))
output=model.predict_classes(image)
time_end=int(round(time.time()*1000))
time=time_end-time_curr
print("")
print(output)
print("")
print(time)
print("")
```

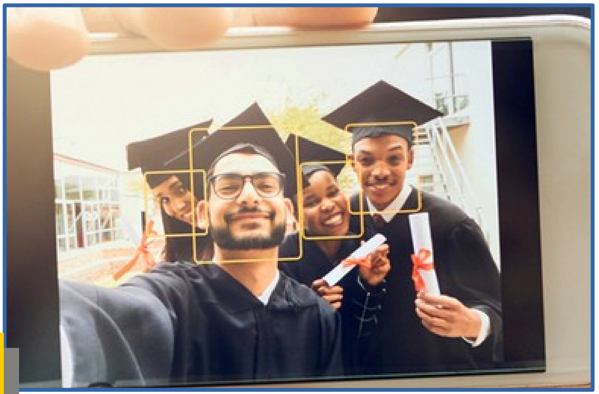
One image with resolution 150x150 with 3 channels (r, g, b) colors

11-10-2018 Amazon Rekognition

https://aws.amazon.com/rekognition/

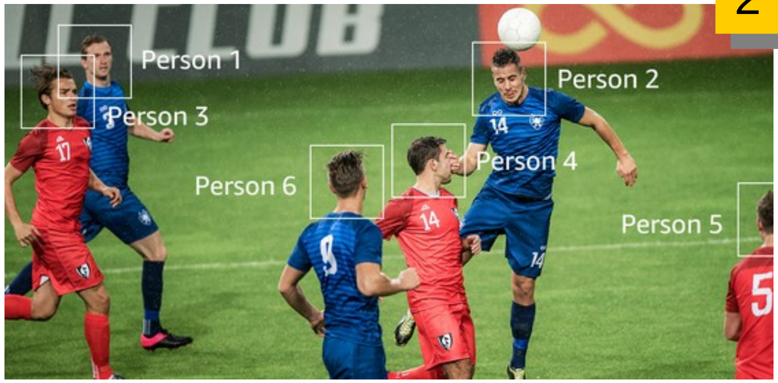
Amazon Rekognition
Easily add intelligent image and video analysis to your applications.

Submit an image or video to the Rekognition API, and the service can identify the objects, people, text, scenes, and activities, as well as detect any inappropriate content, compare faces for a wide variety of user verification, people counting, and public safety use cases



1

Identify a person in a photo or video using your private repository of face images.



2

Pathing: capture the path of people in the scene, use the movement of athletes during a game to identify plays for post-game analysis.

3

Unsafe content detection, identify potentially unsafe or inappropriate content