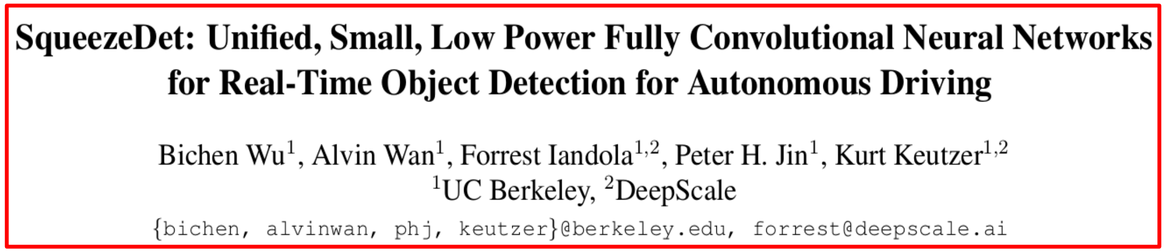
11-13-2018 SqueezeDet for Pedestrian Detection

https://medium.com/omnius/fast-object-detection-with-squeezedet-on-keras-5cdd124b46ce

Single-shot Object Detection:
(1) identify the locations, usually boxes, and the classes of multiple objects within the image.
(2) provide with a confidence value of its prediction for filtering.

Define anchor boxes: before the training, define a grid over the image. At each point of this grid, define multiple boxes with different aspect ratios tailored to the dataset.



SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving

Bichen Wu[1], Alvin Wan[1], Forrest Iandola[1,2], Peter H. Jin[1], Kurt Keutzer[1,2]
[1]UC Berkeley, [2]DeepScale
{bichen, alvinwan, phj, keutzer}@berkeley.edu, forrest@deepscale.ai

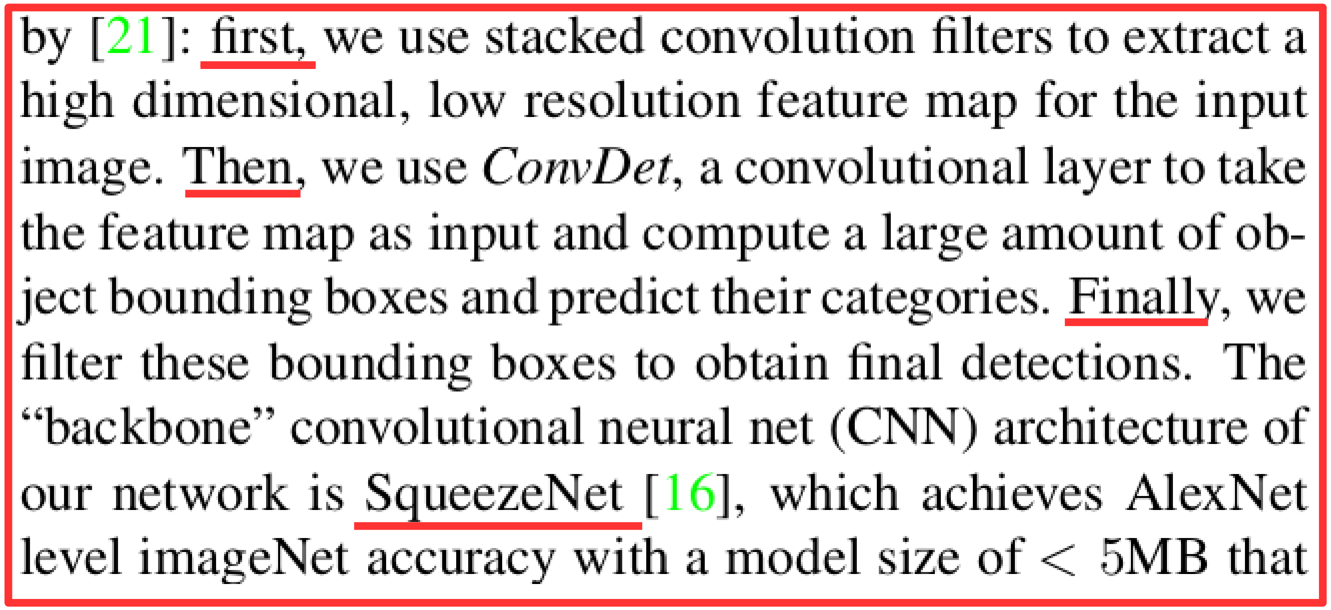
(1) Total model size is less than 8MB.
(2) The inference speed: 57.2 FPS 3 with input image resolution of 1242x375.

https://www.amazon.com/EVGA-GeForce-GAMING-Graphics-12G-P4-2990-KR/dp/B00UVN21RQ

TITAN X GPU

EVGA GeForce GTX TITAN X 12GB GAMING, Play 4k with Ease Graphics Card 12G-P4-2990-KR

Buy used:
$673.08
+ $8.42 shipping

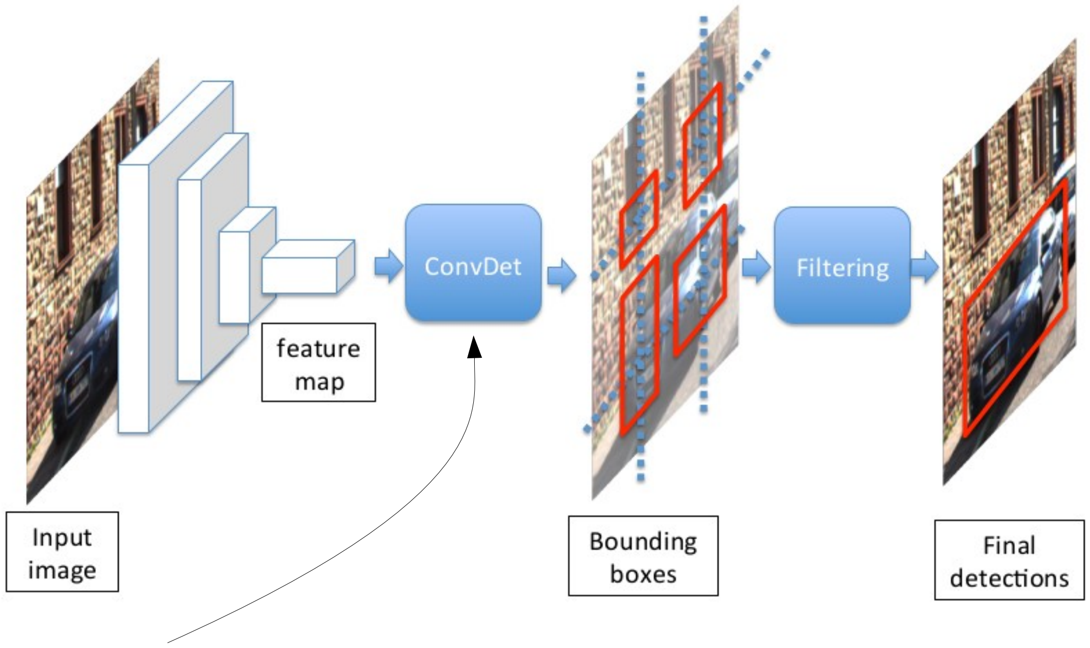


by [21]: first, we use stacked convolution filters to extract a high dimensional, low resolution feature map for the input image. Then, we use *ConvDet*, a convolutional layer to take the feature map as input and compute a large amount of object bounding boxes and predict their categories. Finally, we filter these bounding boxes to obtain final detections. The "backbone" convolutional neural net (CNN) architecture of our network is SqueezeNet [16], which achieves AlexNet level imageNet accuracy with a model size of $<$ 5MB that

Harry Li, Ph.D. 2018

11-13-2018 SqueezeDet Detection Pipeline

Inspired by YOLO, a single-stage detection pipeline



SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving
Bichen Wu 1 , Alvin Wan 1 , Forrest Iandola 1,2 , Peter H. Jin 1 , Kurt Keutzer 1,2
1 UC Berkeley, 2 DeepScale

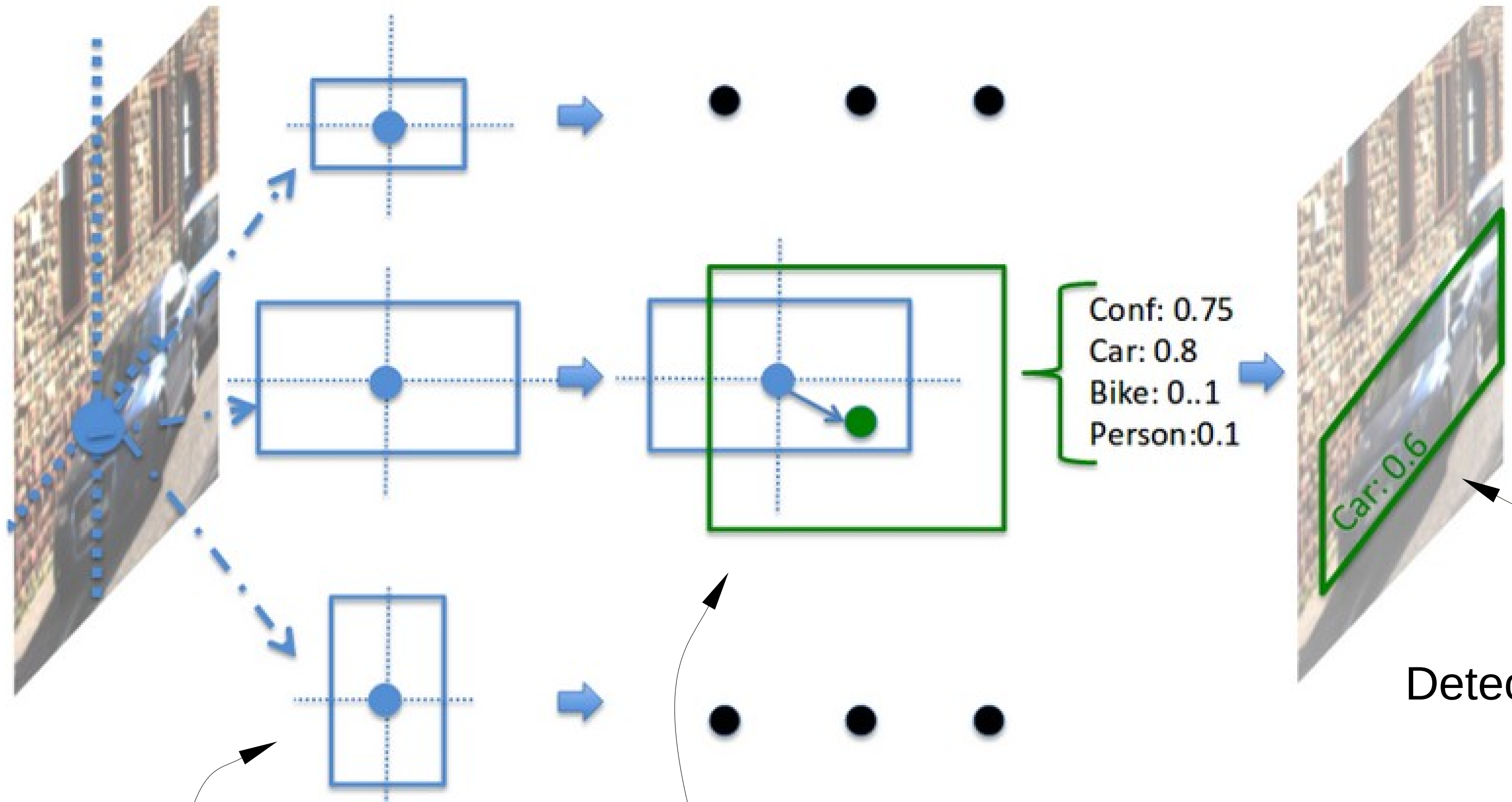
ConvDet is essentially a convolutional layer that is trained to output bounding box coordinates and class probabilities. It works as a sliding window that moves through each spatial position on the feature map.

$$\boxed{\max_c Pr(\text{class}_c|\text{Object}) * Pr(\text{Object}) * \text{IOU}^{pred}_{truth}} \quad (1)$$

Confidence of bounding box prediction

bounding boxes as anchor

11-13-2018 Bounding Box Transformation



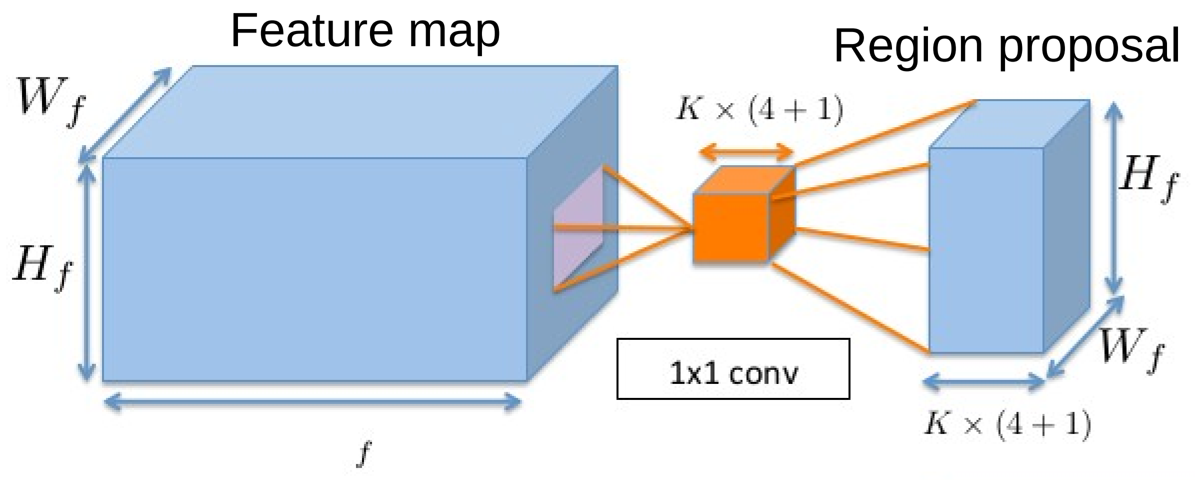
Detections

Bounding boxes as anchor

Bounding box transformation

SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving
Bichen Wu 1 , Alvin Wan 1 , Forrest Iandola 1,2 , Peter H. Jin 1 , Kurt Keutzer 1,2
1 UC Berkeley, 2 DeepScale

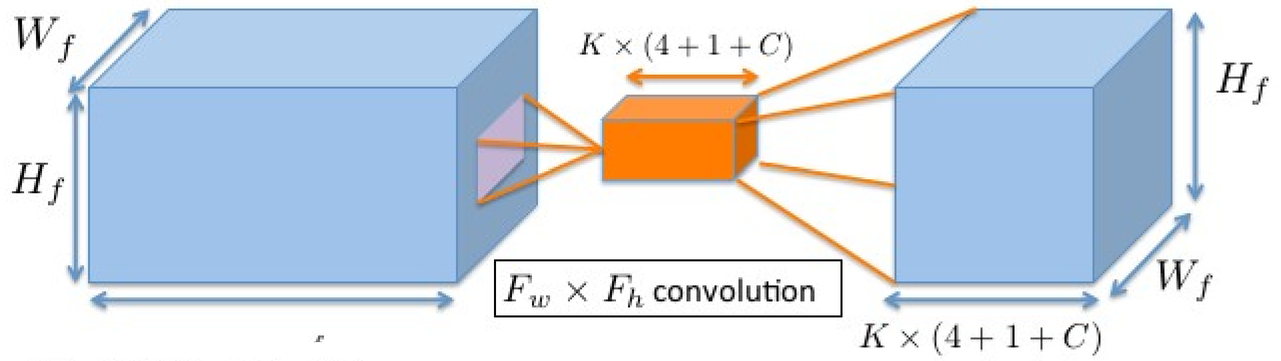
11-13-2018 Squeeze Net vs YOLO Architecture

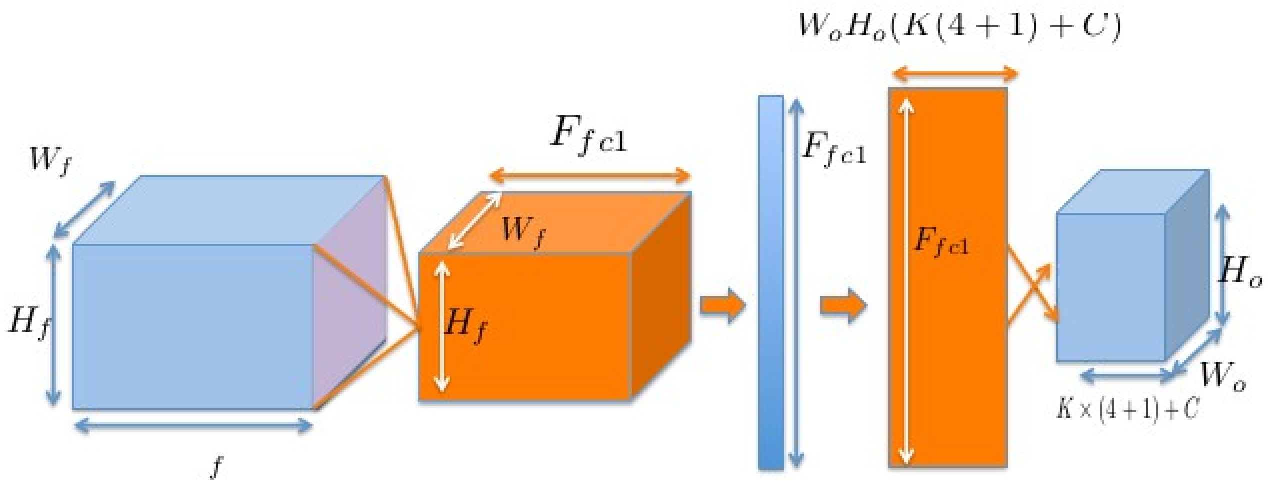


SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving
Bichen Wu 1 , Alvin Wan 1 , Forrest Iandola 1,2 , Peter H. Jin 1 , Kurt Keutzer 1,2
1 UC Berkeley, 2 DeepScale

(a) Last layer of Region Proposal Network (RPN) is a 1x1 convolution with K × (4 + 1) outputs.

(b) The ConvDet layer is a F w × F h convolution with output size of K × (5 + C).

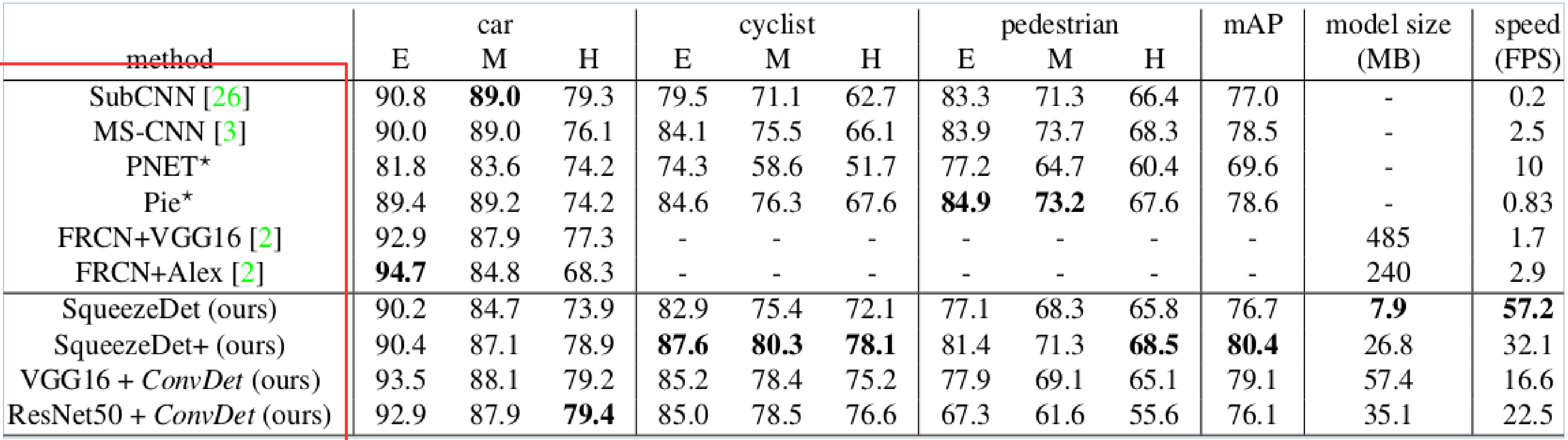




(c) The detection layer of YOLO [21] contains 2 fully connected lay- ers. The first one is of size W f H f Ch f F f c1 . The second one is of size F f c1 W o H o K(5 + C).

Harry Li, Ph.D. 2018

11-13-2018 Performance Comparison and github Repo



method	car			cyclist			pedestrian			mAP	model size	speed
	E	M	H	E	M	H	E	M	H		(MB)	(FPS)
SubCNN [26]	90.8	**89.0**	79.3	79.5	71.1	62.7	83.3	71.3	66.4	77.0	-	0.2
MS-CNN [3]	90.0	89.0	76.1	84.1	75.5	66.1	83.9	73.7	68.3	78.5	-	2.5
PNET*	81.8	83.6	74.2	74.3	58.6	51.7	77.2	64.7	60.4	69.6	-	10
Pie*	89.4	89.2	74.2	84.6	76.3	67.6	**84.9**	**73.2**	67.6	78.6	-	0.83
FRCN+VGG16 [2]	92.9	87.9	77.3	-	-	-	-	-	-	-	485	1.7
FRCN+Alex [2]	**94.7**	84.8	68.3	-	-	-	-	-	-	-	240	2.9
SqueezeDet (ours)	90.2	84.7	73.9	82.9	75.4	72.1	77.1	68.3	65.8	76.7	**7.9**	**57.2**
SqueezeDet+ (ours)	90.4	87.1	78.9	**87.6**	**80.3**	**78.1**	81.4	71.3	**68.5**	**80.4**	26.8	32.1
VGG16 + *ConvDet* (ours)	93.5	88.1	79.2	85.2	78.4	75.2	77.9	69.1	65.1	79.1	57.4	16.6
ResNet50 + *ConvDet* (ours)	92.9	87.9	**79.4**	85.0	78.5	76.6	67.3	61.6	55.6	76.1	35.1	22.5

https://github.com/omni-us/squeezedet-keras

Reference: SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving
Bichen Wu 1 , Alvin Wan 1 , Forrest Iandola 1,2 , Peter H. Jin 1 , Kurt Keutzer 1,2
1 UC Berkeley, 2 DeepScale

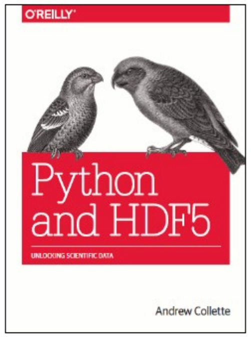
Github code installation guide

https://github.com/omni-us/squeezedet-keras/blob/master/Install.md

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

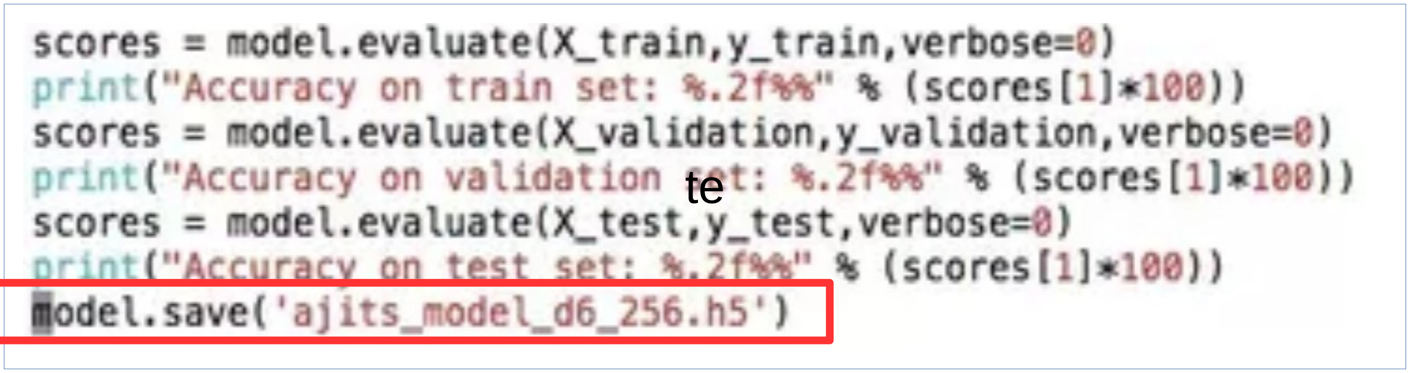
Keras for saving a model is just one line of code after training



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

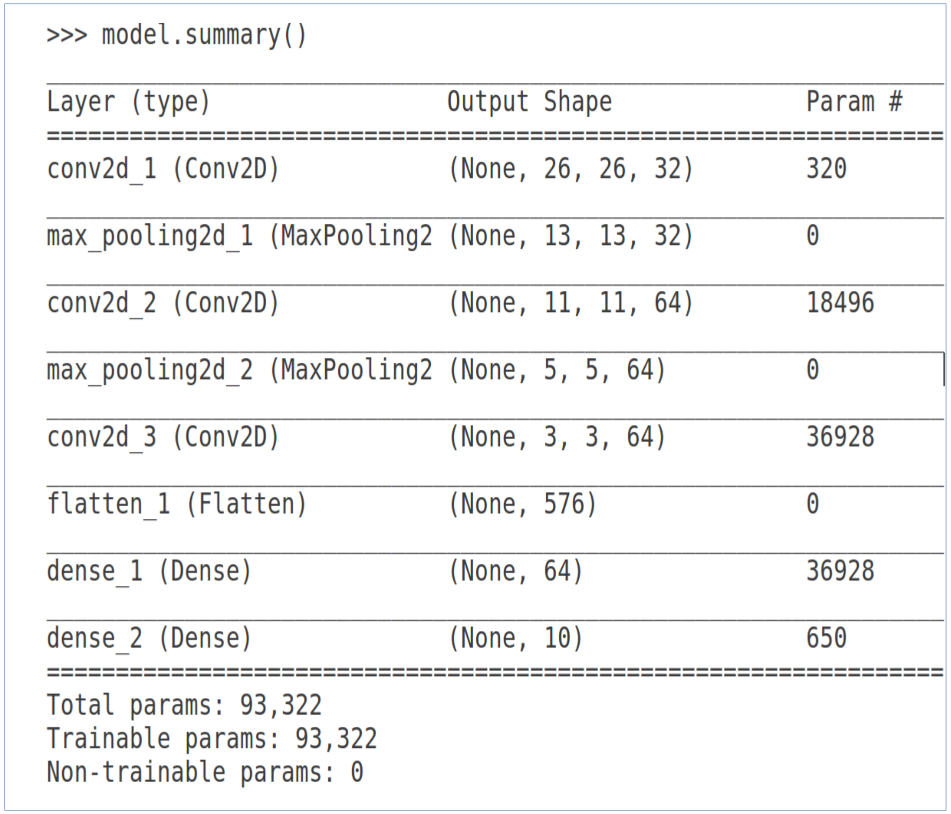
https://stackoverflow.com/questions/35074549/how-to-load-a-model-from-an-hdf5-file-in-keras

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
```

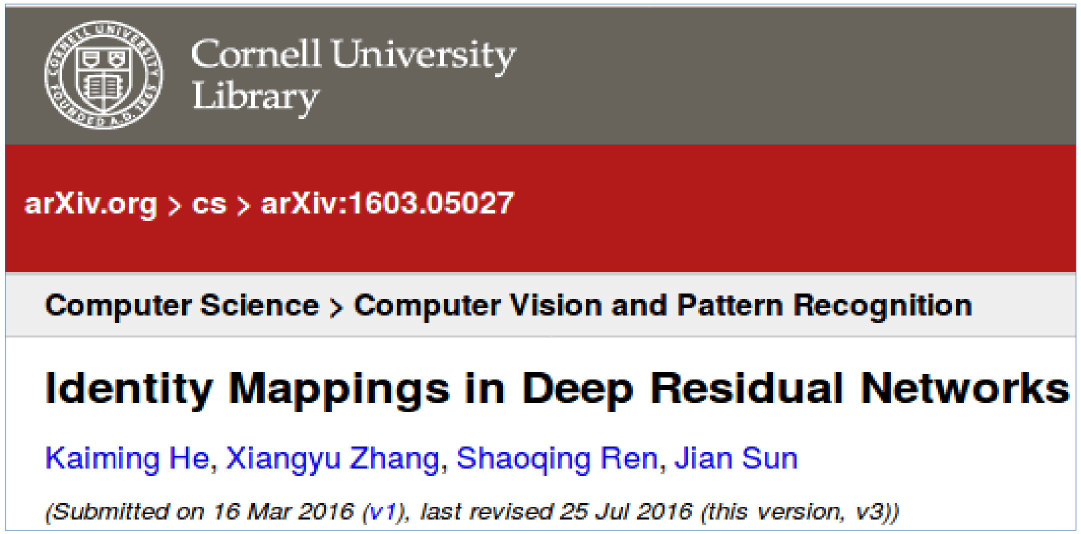
11-6-2018 Keras Supported CNN Architectures

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

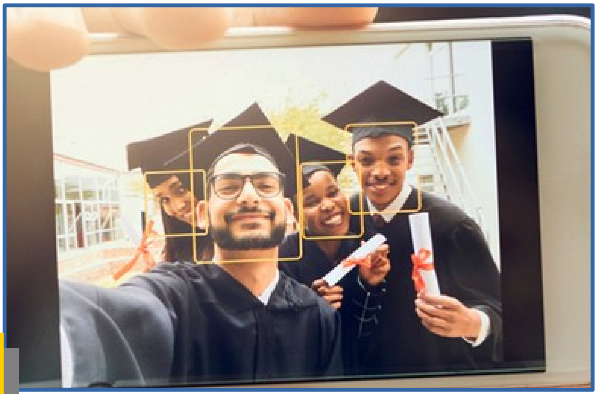
Harry Li, Ph.D. 2018

11-10-2018 Amazon Rekognition

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

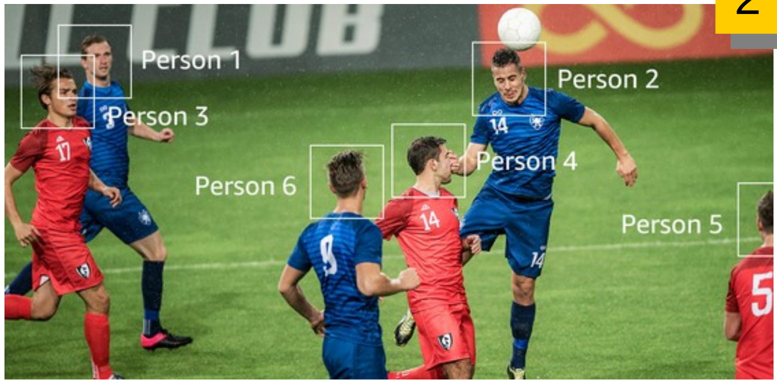
Amazon Rekognition
Easily add intelligent image and video analysis to your applications.

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