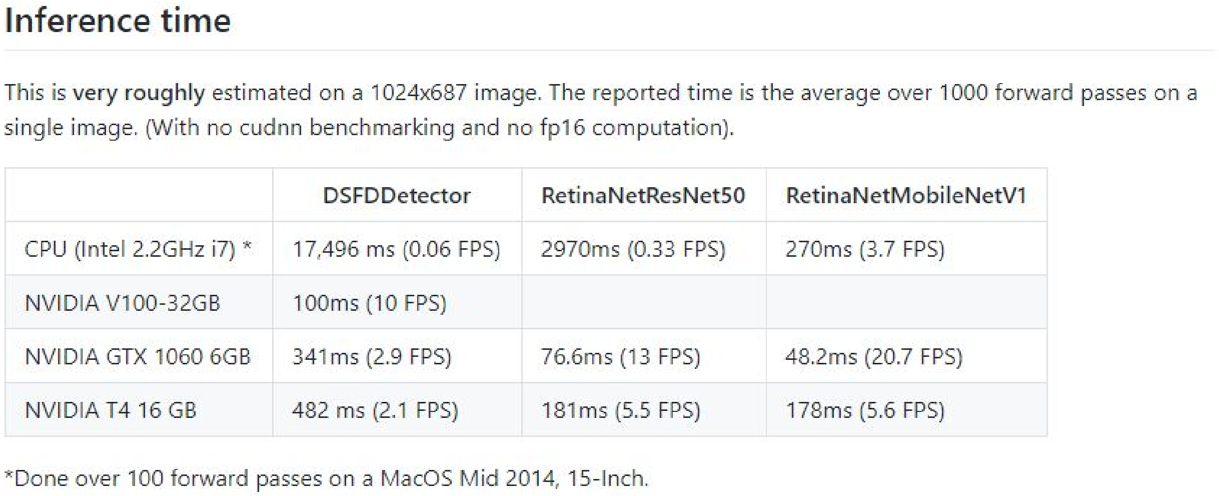
1. We are using MobileNet, since there are many advantages like: light, fast, low-latency. This can help us to deploy our model in mobile environments or camera monitors. And we do want to expand our application to a real-time detector

2. Predicting wearing a mask or not is a simple classification problem.

Face Detection:

Researched MTCNN, YOLO, Blaceface, RetinaNet, MobileNet, Cascade, Faster RCNN from our alumni

Some speed comparison

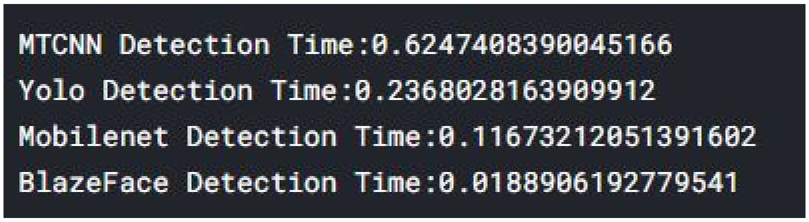


Inference time

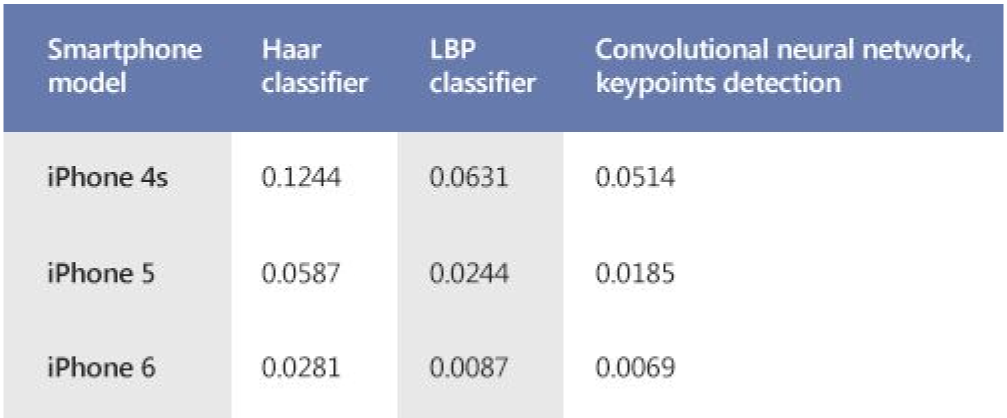
This is **very roughly** estimated on a 1024x687 image. The reported time is the average over 1000 forward passes on a single image. (With no cudnn benchmarking and no fp16 computation).

	DSFDDetector	RetinaNetResNet50	RetinaNetMobileNetV1
CPU (Intel 2.2GHz i7) *	17,496 ms (0.06 FPS)	2970ms (0.33 FPS)	270ms (3.7 FPS)
NVIDIA V100-32GB	100ms (10 FPS)		
NVIDIA GTX 1060 6GB	341ms (2.9 FPS)	76.6ms (13 FPS)	48.2ms (20.7 FPS)
NVIDIA T4 16 GB	482 ms (2.1 FPS)	181ms (5.5 FPS)	178ms (5.6 FPS)

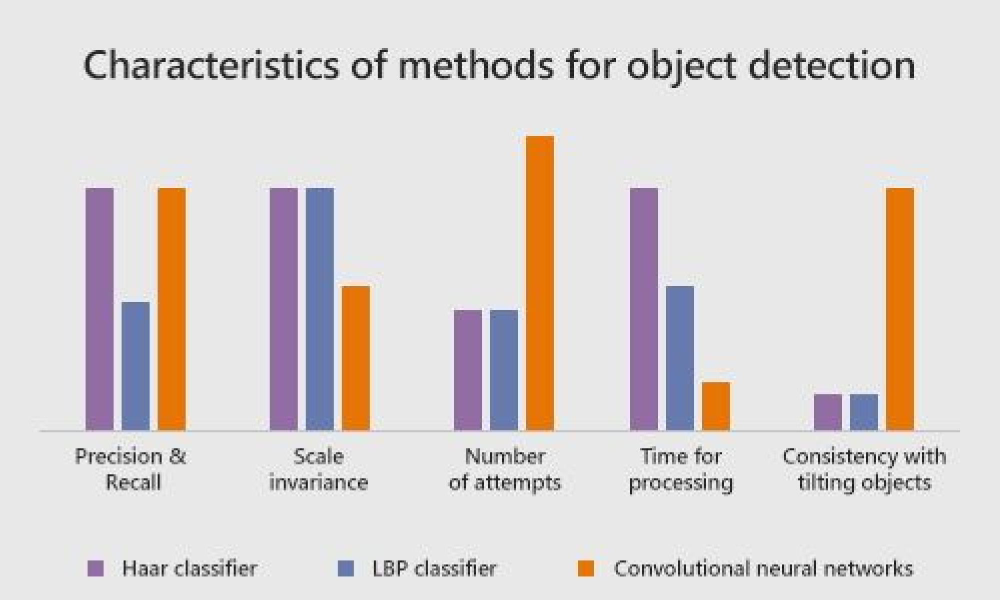
*Done over 100 forward passes on a MacOS Mid 2014, 15-Inch.



```
MTCNN Detection Time:0.6247408390045166
Yolo Detection Time:0.2368028163909912
Mobilenet Detection Time:0.11673212051391602
BlazeFace Detection Time:0.0188906192779541
```



Smartphone model	Haar classifier	LBP classifier	Convolutional neural network, keypoints detection
iPhone 4s	0.1244	0.0631	0.0514
iPhone 5	0.0587	0.0244	0.0185
iPhone 6	0.0281	0.0087	0.0069



The mobileNet approach can reach 60 fps on 16 GB RAM, i7 and 1080 GPU, which is hard to achieve on embedded systems/ IoT devices, and normal smartphones.

Accuracy comparison:
Something

We eventually decided to use haar cascade for face detection, the traditional computer vision approach for some reasons. 1. Saving time from processing, tuning, training models. 2.

Candidate:http://www.computeroptics.smr.ru/eng/KO/PDF/KO41-ENG%20-17/400114.pdf

https://www.kaggle.com/unkownhihi/mobilenet-face-extractor-comparison#Speed-Comparison
Faster RCNN: Model is 500MB , wtf
https://github.com/playerkk/face-py-faster-rcnn
Haar Cascade: Algorithm is easy and efficiency, broadly used on camera etc
C:\Users\dell\Desktop\shared\670\Face-Mask-Detection-master

We can train our own cascade classifier using masked dataset, which will increase the accuracy a lot

Blazeface: Only JS API since it's designed for mobile devices
https://google.github.io/mediapipe/solutions/face_detection.html#example-apps
MTCNN: Too slow
RetinaNet on mobile net: Too slow and high hardware requirement, but can be used in image
RetinaNet on Res152, good accuracy on detecting face either wearing or not wearing a mask
MobileNet: Medium hardware requirement, tensorflow object detection requirement
https://github.com/yeephycho/tensorflow-face-detection
Hog face detection
Not working well
Lbp features
Res10,300x300, bad accuracy on image, but good on video
https://iopscience.iop.org/article/10.1088/1757-899X/732/1/012038

Todo: Build some application and test the accuracy and performance on these models

Haar have some restrictions on features, which is 20x20, so we can upscale the images

Haar may give a high false negative rate, but which is fine in this case. Because if it can't detect the face correctly, you might be wearing a mask. The intention is to find who doesn't wear a mask

To train:
https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html

I requested a trained cascade file from vision_art and am waiting for the response

Current Disadvantages:
Masked face might not be detected correctly, but we care more on who's not wearing the mask

Further improvement:

Train our own cascade classifier
Find a lightweight and real time model for face detection and use transfer learning to train it with masked face
Can be applied to video source