Flexible Masked and Unmasked Detection in Public Area

Yuwei Chen Chengzhe Li
College of Information and Computer Sciences, University of Massachuesetts, Amherst
140 Governors Dr., Amherst, MA 01003
https://www.cics.umass.edu/

Abstract

COVID-19 pandemic is no doubt to be the most difficult global problem that we are currently facing. Up to Dec 2020, there are over 63 million confirmed cases and 1.4 million deaths globally. In Jun 2020, the World Health Organization (WHO) endorsed masks to be an effective method to prevent the spread of the virus. To urge and remind people wearing masks in public area, we are designing a computer vision solution. This solution consists of two components. One component is a mask detector that use MobileNetV1 model from Google to classify whether a given face portrait wearing masks or not. The other component is a face detector that applies MobileNetV2 backbone for RetinaNet model to detect faces in dense faces in images, and Haar Cascades in real-time faces capturing through WebCam. The accuracy achieved in mask detection is 97.5%, and the accuracy in face detection for RetinaNet and Haar Cascades are 97.95% and 93.9% respectively.

1. Introduction

COVID-19 is a highly infectious respiratory disease. The primary way that COVID-19 virus appears to spread is by close person-to-person contact via droplets or skin touch. Therefore, people wearing masks for their own safety are concerned with reducing the flow of germs and the infected number of people reduction. To deal with this, we start our work by detecting who wears the masks and who are not in public area. Since checking if a person is wearing mask is difficult and wasting a lot of manpower especially in crowd, we are introducing computer vision techniques in tackle this issue. Computer vision is a following section of Deep learning particularly an area of convolution neural network (CNN)[33]. Added with one main thing is CNN supports very high configuration Graphic Processing Units (GPU) thus as real time image or video extraction of visualization is a bitter task. As we require people mask having or not which call a surveillance system there is a need for powerful validation such as video stream analysis that is fulfilled by advanced CNN [19].

In this paper, we introduce a mask detector which is trained by real-world masked and unmasked faces dataset and artificial generated masked faces dataset. We are using transfer learning on a classical light weight deep-learning model. Then we compare several classic object detection models in considering hardware requirement, applicability in various data source, and speed aspects. We finally come up with a high accuracy flexible face mask detection solution that proved to be equipment friendly as well as time saving.

The organization for the rest of the paper is as follows. Section 2 reviews previous related works. Section 3 describes the datasets we selected for mask detection model and data augmentation we applied. Followed by illustration of model training and predicting results for mask detector component. Section 4 shows face detector model selection with comparison of different characteristics in models and predicting performance in experiment. Section 6 presents the conclusions and possible extension of future work.

1.1. Background

Traditional object detection uses multiple steps detectors. Some of those are Viola and Jones[32] detector, which extracts useful features using Haar cascades on integral images for face detection. HOG[5] which plots the distribution of directions and magnitudes of oriented gradients to extract features, and it applies sliding-window in pedestrian recognition. Few years later, a deformable part-based model (DPM) is proposed and uses the same window selection methods to capture objects and extending dense detectors to more general object categories. Those handcrafted features consume heavy computational efforts and lacking of flexibility.

Recent purposed deep learning based detector can improve the efficiency in features selection. There are two popular methods: one-stage object detectors and two-stage

object detectors. The differences between them is whether the region proposals is separated as one stage. Generally, two-stage methods can provide higher detection performance but need to pay for time expense. Typical models as R-CNN series models(R-CNN[6], Fast R-CNN[9], Faster R-CNN[10]) start from using selective search to propose candidate regions to building a region proposal network(RPN) to make region proposal more efficient, and than feed those captured features from the previous stage to a selected neural network model for detection.

One-stage detectors implements all functions including regions selection, features extraction and features detection in a single neural network model. In practice, these detectors may not achieve same performance as the previous but they show a great improvement in computational speed. Modern models such as YOLO series models(YOLO v1[25], YOLO v2 [26]YOLO v3[27], YOLO v4[4]) divide the image into consistent blocks and using non maximum suppression to decide the best regions to capture given features then feed these features into a neural network model. Later on, in order to improve detection accuracy, Lin[22] proposes Retina Network (RetinaNet) by combining an SSD and FPN architecture, which also include a novel focal loss function to mitigate class imbalance problem.

As object detectors are usually deployed on mobile or embedded devices, where the computational resources are very limited, Mobile Network (MobileNet) [14] is proposed. It uses depth-wise convolution to extract features and channel wised convolutions to adjust channel numbers, so the computational cost of MobileNet is much lower than networks using standard convolutions.

In our research, there are two detectors to be built. Namely, a mask detector and a face detector. The mask detector works for classifing if a given face is with mask, and the face detector is finding faces in a given image or a short video captured through WebCam. For the former detector, a pre-trained MobileNet with Inception net backbone is built and fine-tuned for better fit in our data. The latter one we are comparing different pretrained neural network models as well as handcrafted detectors such as RetinaNet[13], HoG etc. and end up with computational friendly models for various data sources.

2. Mask Detection

2.1. Dataset

There are three original datasets in training the mask detection model. The first one is VGGFace2, which is a large-scale face recognition dataset downloaded from Google Image Search with large variations in post, age, illumination, ethnicity and profession. Since the size of the whole dataset exceeds our computational power, we are using the test set which maintains the diversity of the whole one with less size. There are 8000 images selected.

The second one is from Kaggle which records people's social activities with labeled bounding box to capture their faces, we are calling it People in Social Activities(PSA) here. This dataset is including masked faces and unmasked faces. While doing social activities, people's faces will have different poses and angles and it could help compensate the shortage of not having masked faces in different poses and angles from the first dataset. There are a total of 853 images.

The third one is head portrait images also from Kaggle with masked and unmasked faces, we use Head Portraits Images(HPI) as representation. This dataset includes masked faces and unmasked faces in real world with high resolution. This dataset includes augmented head portrait with samll skewing and mirroring which can boost up the total size of masked faces and unmasked faces in model training. Below are some samples from each dataset. There are a total of 11766 images.

Figure 1. Sample Images from Three Datasets

2.2. Data Preparation

Not all face images can contribute to our model training, some of them will distract the classification. Such cases are faces being too small to recognize, cartoon faces existing, etc.To alleviate the distraction, we applied methods including setting up a limitation of pixels dimension size, and removing cartoon faces. After preprocessing, we find that there are imbalanced classes in masked faces and unmasked faces with a ratio of 4:1. Some data augmentation methods including slight skewing(+/-45 degrees) or zoom in/out are feasible in our case because we may expect small tilted cameras or head shaking. However, it is not

expected to see a rotated faces in real world. To improve the generalizability of our masked faces. Advanced data augmentation methods come into play, such methods include using GAN[1] to boost up the data diversity for each class. However, it increases both the training time and the model size.

Inspired by MaskTheFace[2] project, it applied face recognition library to detect facial features including nose bridge, eye brow, chin, etc. then adjust masks template accordingly and paste it to the face. Although this method is greatly depending on the angle diversity of masks, it works well in approximating real world masked faces with less time and computational power. And each step is fully under control. We applies this method to generate artificial masks on half of the VGGFace2 dataset.These are some of the results.

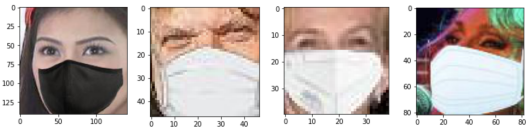

Figure 2. Artificial Generated Masked Faces

We applied a ratio of 16:2:1 to split the dataset. With a total of 18433 images in training set, 3212 images in validation set, and 1403 in test set. Both classes are now balanced. Implementations can be found on https://github.com/GHAliceC/MaskDetector

	Train		Validation		Test	
	Masked	Unmasked	Masked	Unmasked	Masked	Unmasked
PSA	1427	1016	431	215	386	96
HPI	5000	5000	483	483	400	400
VGG-Face2	3200	2800	800	800	89	32

Figure 3. Dataset Statistics for Mask detector

2.3. Model Selection

To illustrate the advantage of using MobileNet as our mask detector model, we compare several classic deep learning models including ResNet50[12], VGG19[30], in terms of depth, model size, and numbers of parameters for reference(table 1).
MobileNet_v1 outperforms the other two models in parameter size as well as model size. According to the experiment[24], the MobileNet model achieves less running time for a single image classificaiton compared to VGG-19 and ResNet50.

During the training process, We first try loosing the last fully connected layer with a small learning rate. The training set achieves an accuracy around 97% in average at the first 10 epochs, but the validation accuracy first goes high to 95%, then as more epochs come, it drops to 90%. There might be an overfitting situation for our training set. Then we add a dropout layers by setting the percentage as 0.2. Although both training accuracy and validation accuracy increases, the values is not even as high as the previous practice with around 96% for training and 95% for validation. Since we are only training the last fully connected layer and applying dropout at the same time, we may fail to fully learn the features of our dataset before muting some of important features. We then try to unfreeze more layers to learn. We unfreeze the last convolutional layer and the training and validation statistics is showed in below section.

2.4. Results

We are ploting two graphs(Figure 4) to show the learning curve of our fine-tuned MobileNet model. The graph on top is the accuracy change for 10 epoch, and the lower graph is the loss change during training. Blue line is the learning curve for training set, and orange line is for validation set.
Both dataset show a trend of increasing in accuracy during

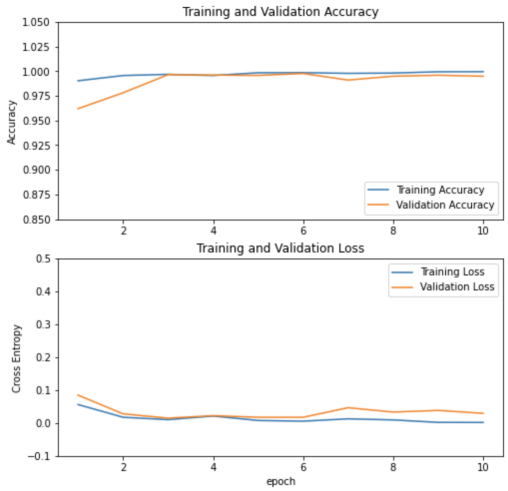

Figure 4. Train and Validation

training. At the early few epochs, there is an overlap. One of the explanation is that there might be some "easier classified" samples in validation set so that the model can make good classification on validation set. There are some explanations on the possibility of using a high ratio in dropout layer when this situation occurs. They point out that higher dropout rate artificially poses a barrier for training set to continue learning. Given that there is not much difference in training accuracy and validation

Models	Depth	Model Size(MB)	Parameters(Million)
MobileNet_v1[14]	28	16.9	4.24
ResNet50[12]	50	96	25.6
VGG19[30]	19	535	138

Table 1. Comparison on different CNN models

accuracy, we think these explanation is not applicable in our case.

After fitting our model to test set, we achieve a high accuracy as 97.5%. Since only measuring accuracy cannot give us a throughout picture of model performance, in addition to that, two models with the same accuracy on the same test set may have different use cases. We are using confusion matrix[31] to show the Sensitivity and specificity of the model performance.(Figure 5) There are four components included in the matrix, namely, true positive(TP), true negative(TN), false positive(FP), false negative(FN). The transformation between accuracy and the matrix is

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

In our research, there are true positive and true negative shows unmasked people are classified as without mask, and masked people are classified as with mask. The bottom left grid is false negative showing masked people being classified as unmasked, and the top right grid represents opposite situation. Our model has a better performance to finding masked people that unmasked people. One of the explanations is that there might be some occlusion rather than masks in faces that our model cannot recognize them. In the future, we can dive deep in separating cases in face occlusion to optimize the false positive rate.

The line plot on the left is Receiver Operating Characteristicon (ROC) curve[11] which can further show the comparison of true positive rate and false positive rate. On the bottom right area of the orange curve is Area Under the Curve(AUC) representing the sensitivity of the model for our dataset. The closer the curve reaches top left corner, the better the model is. Since the ideal value for AUC is 1.0, and we achieve 0.977 in our case, this model is proved to be a good classifier.

3. Face Detection

Since face detection and object detection is one of the most popular topics among Computer Vision community, there are numerous mature and well-developed networks can be used like MTCNN[35], Faster RCNN[28], etc.. However, which one is the most suitable one for our project, especially when we want to deploy our application to poor performance devices like IoT devices or monitor cameras. Also, how to adjust the model so they can detect a masked face correctly, which might be hard for the original design of some networks. We investigated, researched and compared 6 popular face detection networks, and 3 traditional manually built features.

3.1. Possible Approaches

3.1.1 Neural Networks Approaches

As a popular approach for object detection, Neural Networks, especially Deep Neural Networks(DNNs)[20] works pretty well on face detection. However, DNNs usually requires an advanced hardware environment set up for training and making inferences. Which is hard to achieve real-time on poor performance devices. We investigated some potential networks as listed.

Faster RCNN[[18][28]
Since Convolutional Neural Networks(CNNs) has been showed useful for analysing images, the amount of parameters and computation has been a limitation of further research. The Region based CNNs(R-CNNs)[6] is the idea derive from CNNs, which divide an image into regions and analyse them separately or combinedly. However, the speed is still one of the biggest obstacle since it's a two-stage detector. Researchers proposed Fast RCNN[9], Faster RCNN[28] to handle the speed limitation. The one we investigated is Face detection with Faster RCNN[18], which achieves an impressive accuracy with 0.38s inference time on a Nvidia K40c graphic card.

Blazeface[3]
Blazeface is a lightweight face detection framework developed for mobile devices by Google. Which can run as fast as 200-1000+ FPS on flagship devices. It contains a lightweight feature extractor inspired but distinct with MobileNetV1/V2[14][29], a GPU-friendly anchor scheme modified from Single Shot MultiBox Detector (SSD)[23], and an improved tie resolution strategy alternative to non-maximum suppression. However, the open source framework only works with JavaScript version Tensorflow.

MTCNN[35]
Multi-task Cascaded Convolution Neural Net-

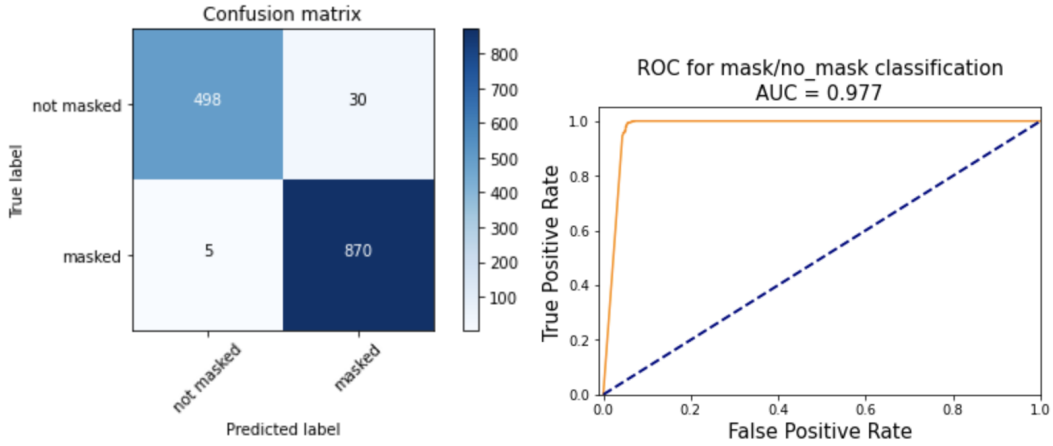

Figure 5. Evaluations on Test set

works(MTCNNs) which is a combination of Viola-Jones[32] cascade face detector and Convolutional Neural Networks. They adopts a cascaded structure with three stages deep convolutional networks that predict face and landmark location in a coarse-to-fine manner. They also proposed a new learning method but which falls out of our discussion in this paper. However, it's reasonable to have a slower inference speed than Single Shot MultiBox Detector[23] because of more stages. It runs 5.4 FPS on a CPU-Based TensorFlow in a Core i7-3612QM.

RetinaNet[13]

RetinaNet is a framework built for both speed and accuracy on object detection proposed by Facebook. Benefit from their novel Focal Loss focuses training and their structure based on ResNet[20]. They are able to outperform the state-of-the-art two-state detectors as a one-state detector. It works impressively well on dense object. RetinaFace[7] is the study based on RetinaNet for face detection, which also illustrates a better accuracy than other one-state or two-state face detectors. It runs 20.7 FPS on GTX 1060 GPU for a 1024x687 image.

MobileNet[14][29]

MobileNetV1/V2 is a class of efficiency models designed for mobile and embedded vision application developed by Google. It's been extremely popular since it was published. It also matches our targeted platform, mobile or embedded vision devices. We found an open-source project[34] for face detection based on MobileNet and SSD, which runs 60 FPS on GTX 1080 graphic card, using less than 300 MB memory for a single inference.

SSD[23]

Single Shot Multibox Detector has been used a lot on the architectures we talked above. It has multiple advantages that suits our project over some two-stages or more detectors (e.g. Dual Shot Face Detector[21]). Compare to two-stage detector, SSD doesn't need an initial object proposals generation step. Leveraging the one-stage benefit, SSD is usually faster and more efficient than two-states detector, however, it may sacrifice the performance of detecting small objects to gain speed. We used a SSD300 detector with Res10 backbone as the benchmark for evaluating.

3.1.2 Features Approaches

Other than using Deep Neural Networks, some traditional feature extractor also brings up our attentions. This can also be a good approach for embedded devices since it's been used by digital cameras and other devices for a long time. The speed is stable and the hardware requirement is not as high as some DNN models.

Viola-Jones Haar Cascade[32]

Viola-Jones Haar Cascade is a classical, but still popular approach for object detection. It presents three important concepts. Integral image, which preprocess the image into special format for fast processing and feature computing. Second is the learning algorithm based on Adaboost[8], and the third is the method they called cascade, which allows them to pack a lot of features on their feature set while filtering out regions that are not likely to be the final result. We used the "Haarcascade frontalface Alt" pretrained descriptor from OpenCV[17].

LBP/HoG Features[5][15]

Other than Viola-Jones Haar Cascade, there are some other features can be applied. For example, using Local Binary Patterns[15], or using the Histogram of Oriented Gradients[5]. We also tried to use the profile contained on OpenCV. The result is not very satisfying on accuracy so

they will not be listed on comparison.

3.2. Comparison

Some frameworks are not likely to be deployed to our target devices, they are not tested directly. Some results are took from the original paper or the release websites. To see the performance, we performed different face detection on a same image which is representative. It has masked faces, unmasked faces, and also faces that are blur or small enough.

3.3. Decisions and Trade-offs

We first excluded Faster RCNN and MTCNN, since they are two-state detector and showed longer inference times. We are putting real time application on our first priority, with some tolerance with FPS. However, we still want it to be runnable without an advanced hardware environment. Then, blazeface has an impressive performance on both speed and accuracy. However, the official library only supports TFJS(Tensorflow JavaScript), which we are not familiar with. We end up with using Res10SSD, RetinaNet, and Haar Cascade. The reason will be explained later.

4. Applications

We built 4 applications(in two categories) in total which share the same pipeline. The face detector model will take the original image or the resized original image as input, and generating bounding boxes for faces. Then, the faces will be cropped out to be the input data for mask classifier. Finally, display the bounding box and predicted label (masked/not masked). All applications we built are available on https://github.com/smalllicheng/face-mask.

4.1. Video Stream Application

Video with Haar Cascade

For this application, we used "Haarcascade frontalface alt" from OpenCV as our feature profile. The end to end performance is good on speed, but not optimal on accuracy. The limitation is the profile we are using are trained for frontalfaces, which doesn't adopt well automatically with masked faces. This can be solved by training our own cascade profile. Due to the time limitation we can't finish this, but the Haar Cascade approach is still promising on both speed and accuracy. The top priority of our project is to find out people without masks timely, so it can alarm or make notification. If the frontal face is not detected, it tend to be masked. Another limitation is sizes, OpenCV implementation only considers region greater than 20 x 20 as possible candidates, even it can be solved by enlarge the image, the speed could be drag down. This makes haar cascade a bad candidate for dense faces in a small image, but it is still useful for WebCam or other situation like sensor door. The implementation is on **videofacedetector_haar.py**

Video with Res10SSD

Using Single Shot multi-box Detector with Res10 backbone, the processing speed of video stream is good enough to be called real-time. The tests on image shows that Res10SSD works not extremely optimal with masked faces, especially when there are a lot of faces, the threshold value can be hard to adjust. We cannot really deploy our application to places with dense people makes the lack of real-time experiments. We believe using transfer learning, or just train the model again with masked faces dataset can gives a better accuracy. The implementation is on **video_res10.py**

4.2. Image Application

In real life, monitor applications may not need more than 30 FPS to be real-time. For example, some traffic monitors will take pictures with a certain interval and use images as input. Based on this fact we can have larger and slower models to get a more accuracy result on face detection, and process images as batches to speed it up. We used Haar cascade again for one approach, and RetinaNet over MobileNet for another high accuracy approach. The implementation is on **retinaandmask.py** and **facemask_image.py**.

5. Extensions

Because of the time limitation, we left with a huge space of improvements on this project. We just want to address a little bit about our future directions. With a rising consideration on privacy, what should we do to make sure our application is ethical and legal? Another direction is to combine our application with face recognition. This can be challenging because of lacking facial information while people is wearing mask, but distinguishing who is the person that is not wearing a mask is approachable. Another enormous improvement we can make is applying transfer learning on existing model with the masked crowd dataset. The same improvement can be applied to Haar Cascade too. The pretrained profile doesn't work well with masked face, but training with masked faces should improve it by a lot. Also, our application can be modified to work with video input with just little changes in code.

6. Conclusion

We present our Flexible Face Mask Detector, including the mask classifier we built based on MobileNet, and the carefully selected face detection method. It can achieve real-time performance without requiring advanced hardware. We also proposed several approaches that suits dif-

(a) Original (b) Ground Truth (c) Haar Cascade

(d) HoG (e) Res10SSD (f) RetinaNet

Figure 6. Performance of different face detection networks

Method	Hardware	Speed(FPS)	Model Size(MB)	Image Size	Accuracy
Faster RCNN[18]	Xeon E5+K40c	2.8	500	350 × 450	95.2
Blazeface[3]	iPhone XS	1666	-	128 × 128	98.61
MTCNN[16]	i7-3612QM	5.4	-	474 × 224	Good
RetinaNet[34]	GTX 1060	20.7	15	1024 × 687	Good
MobileNet[29]	iPhone XS	476	15	128 × 128	97.95
Res10SSD[20]	GTX 970M	25.2	10.1	300 × 300	Normal
Haar Cascade[32]	Pentium III 700MHz	15	0.6	384 × 288	93.9

Table 2. Comparison on different face detection methods

Figure 7. Example of Video Stream Application.

ferent environment. For dense faces without real-time demand, we have RetinaNet for accurate detection of masked faces with an acceptable speed. For real-time application we have Res10SSD and Haar approaches that suit for just a few or dense faces.

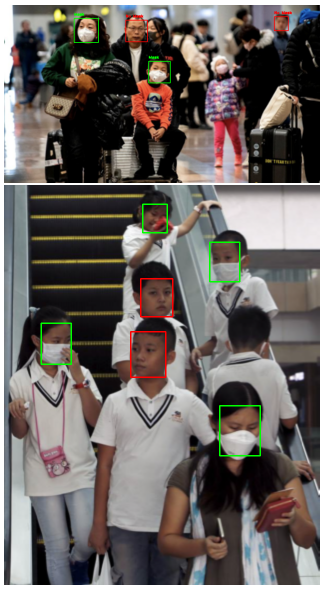

Figure 8. Example of Image Application.

References

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks, 2018. 3

[2] Aqeel Anwar and Arijit Raychowdhury. Masked face recognition for secure authentication, 2020. 3

[3] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus. *arXiv preprint arXiv:1907.05047*, 2019. 4, 7

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE*, 2005. 1, 5

[6] R. Girshick J. Donahue T. Darrell and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014. 2, 4

[7] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *CVPR*, 2020. 5

[8] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999. 5

[9] R. Girshick. Fast r-cnn. 2015. 2, 4

[10] S. Ren K. He R. Girshick and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. 2016. 2

[11] Karimollah Hajian-Tilaki. Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation, 2013. 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3, 4

[13] T. Lin P. Goyal R. Girshick K. He and P. Dollár. Focal loss for dense object detection. In *CVPR*, 2018. 2, 5

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 4, 5

[15] Di Huang, Caifeng Shan, Mohsen Ardabilian, Yunhong Wang, and Liming Chen. Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(6):765–781, 2011. 5

[16] ipazc. Mtcnn. 7

[17] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015. 5

[18] Huaizu Jiang and Erik Learned-Miller. Face detection with the faster r-cnn. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 650–657. IEEE, 2017. 4, 7

[19] A. Koubaa, B. Qureshi, M. Sriti, Y. Javed, and E. Tovar. A service oriented cloud-based management system for the internet-of-drones. 2017. 1

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 4, 5, 7

[21] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: dual shot face detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5060–5069, 2019. 5

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018. 2

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 4, 5

[24] Bosheng Qin and Dongxiao Li. Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19. 3

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 2

[26] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016. 2

[27] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 2

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 4

[29] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 4, 5, 7

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 3, 4

[31] Kai Ming Ting. *Confusion Matrix*, pages 260–260. Springer US, Boston, MA, 2017. 4

[32] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 1, 5, 7

[33] S. Wan, Y. Liang, and Y. Zhang. Deep convolutional neural networks for diabetic retinopathy detection by image classification. 2018. 1

[34] yeephycho. Tensorflow face detector. 5, 7

[35] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016. 4