11-767 On-Device Machine Learning - Final Report

Jiajun Bao
jiajunb@andrew.cmu.edu

Songhao Jia
songhaoj@andrew.cmu.edu

Zongyue Zhao
zongyuez@andrew.cmu.edu

December 12, 2021

1 Motivation

The SARS-CoV-2 virus and COVID-19 disease have made wearing a mask not only a measure to protect ourselves, but a requirement to enter many public places. Checking if people entering public places are wearing masks is important but painstaking. Having a person conducting inspections leads to a high risk of exposure to the disease of inspectors, potentially accelerating the spread of the epidemic. Existing mask detection systems either have a large memory footprint [1, 2] or have a high latency [3]. All the issues prevent the models to be deployed on an resource-constrained device. Also, most of the efforts are on tuning the features to have a model with a good accuracy. However, a real-world system is more that a good-performed model. For example, one major problem with existing mask detection systems is that they all need to be in a constant state of detection, meaning that the relevant components such as cameras and mask detection models have to be kept working at all times. While this allows for timely mask detection for each person passing by, this state of operation also undoubtedly wastes a lot of energy due to the large amount of time when no one is passing by, such as the entrance of the classroom after classes have started. In our project, we designed and implemented a energy-efficient end-to-end mask detection system to automate the process of checking whether people entering a public place are wearing masks.

[0]All code of the project are publicly avaliable at https://github.com/andromeda-0/11-767/tree/main/labs/project_final

2 Task Definition

We designed and implemented a mask detection system with the following requirements and features:

1. The system should be able to accurately recognize if the people in the image captured the camera are correctly wearing mask. The input includes an audio of surrounding and an image captured by the camera. The output is a message which indicates if the user is wearing a mask.

2. The whole system should have reasonably low memory footprints and energy consumption so that it can be deployed on a resource-constrained device. Specifically, it should efficiently run on a jetson nano 2GB.

3. The whole detection pipeline should have a low latency so that the user does not need to wait a long time to get a result. Specifically, the overall latency should always be less than 500 ms per image.

4. To avoid meaningless energy consumption, the classification model should be called only if there is a user in front of the camera waiting to be tested. Therefore, it is worthwhile to build a light-weight trigger system that runs non-stop, and the prediction model is called only when necessary. Meanwhile, it is possible that energy cost has a negative correlation w.r.t. system latency, like the design choice whether to let the camera run non-stop. We need to find a reasonable trade-off between these two aspects.

3 Related Work

Wake word Spotting Wake word spotting, or wake word detection, is the task of detecting keywords of interest in a continuous audio stream. It has been widely used in voice assistants. Traditional wake word spotting model consists of an DNN-based acoustic model and a Hidden Markov Model[4, 5]. During runtime, Viberbi decoding is used to produces the start and end senone of the wake word in the input audio. Also, single-stage approaches are developed, where a single feed-forward DNN is s trained to predict subkeyword targets. This approach has a small footprint, while overperform the two-stage approach [6]. Convolutional Neural Networks (CNNs) has also been used for acoustic modeling and have achieved decent improvements[7].

Mask Detection. We conducted a research for existing work in the domain of facemask detection. Since there is a limited amount of published work, we chose open source work on github for experiments. [8] provides a powerful facemask detector with several different techniques, like opencv, pytorch, tensorflow, as backbone, and works well on both video and

single images. Another facemask we have examined is from [9]. It is open source and shares the pretrained weight. However, there are obvious problems with these jobs, and we have no way to use them directly. First, previous works had limited support for Jetson Nano. Since Jetson Nano has relatively limited memory, computing power and bandwidth, these works can not deploy into the system properly. Furthermore, they do not have a trigger-like mechanism to wake up the mask detection system, which is not consistent with our goal.

Another issue that we were interested in, specific to the mask-detection problem, was to utilize pre-trained network weights. Because the real human-mask dataset [10] we found contains only 853 images, the classification backbone of a detection network would easily overfit to the training set if trained from scratch. [11] proposed a solution by adopting transfer learning (over Inception v3 [12]) for mask detection. The authors discussed their choices on data augmentation and fine tuning, and achieved high mIoU on the SMFD dataset.

Object Detection , which is the combined task of classification and localization [13], seems a natural fit for the task of face-mask detection, especially for complex scenes containing multiple faces. There are an abundant amount of domain-unspecific prior work for object detection. For example, [14] proposed a cost-efficient approach that forms a joint representation of bounding box localization quality and classification. As these two metrics are no longer trained separately while being used together during inference, their correspondence is enhanced. [14] also allowed any arbitrary distribution on bounding box locations and proposed a generalized form of Focal loss to handle continuous labels. However, we eventually determined from empirical analysis that object detection pipelines [15, 16], even used with state-of-the art backbones specifically optimized for mobile computing [17, 18], were too slow for our jetson-nano 2GB board.

Classification Models are important in our task setting, whether as the backbone of two-stage object detection pipelines or "as is" in the image classification setting. In this domain, [18] presented a class of efficient models called MobileNets for mobile and embedded vision applications. The authors introduced two simple global hyperparameters that efficiently trade off between latency and accuracy. [19] presented ResNet which greatly improved the performance of cnn-based vision models on image classification tasks. The authors presented a residual learning framework to ease the training of networks, specifically for vision tasks. They provided comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth.

4 Main Results & Discussion

In this section, we will first present the overall design of our system, then discuss the accuracy-latency results for the audio and visual module. Finally, we will discuss the trade-offs we made with respect to energy costs.

4.1 System Design

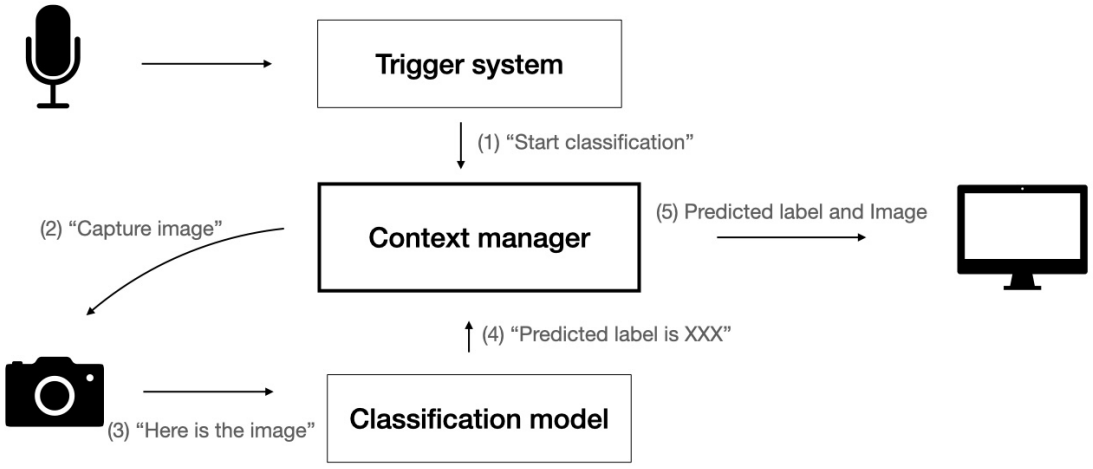


Figure 1: System diagram

Our system consists of three service and three I/O devices, shown in the diagram 1. All services are loaded when the system starts. However, only the audio trigger system and the camera run in the "listening" mode. The classification model is only called when it receives "start" signals from the trigger system. The context manager is used to coordinate and orchestrate I/O devices and other service. Table 1 shows the latency and power cost of each module. The total system latency to trigger and do inference from the camera is **245.0** ms.

Module	**Latency [ms]**	**Power [W]**	**Total Power [W]**
Device Standby	-	3.0	3.0
Voice Trigger	0.05	0.3	3.3
Camera Stream	89.3*	1.1	4.4 (Total Standby)
Classification Module	155.6	2.9	7.3

Table 1: Latency and Power Consumption of Each Module. The energy consumption values are measured together to better reflect the potential coupling and resource competition between modules. *The camera stream itself does not have a latency, however, it slows down the classification model by the reported amount, due to the scarcity in power and computational resources. More details can be found in Section 4.3.5.

4.2 Wake word spotting as the trigger

Given its wide application in voice assistant, wake word spotting is used in our system as the trigger. Following [6], we adopt the single-stage approach: a feed-forward DNN is trained to predict subkeyword targets; an additional classification layer is trained to take features from the previous DNN module and classifies which keyword in predefined set accords to the input. This approach has attractive for on-device inference, because it have a small memory and energy footprints, while having a decent performance. Specifically, we used code from open-sourced solution[1]. As shown in table 2, our model supports 10 keywords, which are grouped into 2 intents. In our current demo, both intents will trigger the classification model. The original goal of having the "greeting" is to help debugging the system, collect logs, and enable more advanced interaction with the user (e.g. released a pre-recorded greeting). Future work could extend additional features with this intent.

Intents	**Keywords**
Mask Detection	check me in hey tartan I wear a mask mask detection I want to check in I have worn a mask
Greeting	how are you hello hi I am

Table 2: Supported wake words: our supports 10 keywords, which are grouped into 2 intents.

Although there are many open-sourced solutions for wake word spotting, ones specifically designed for on-device inference like Picovoice are deficient. We recorded the sounds for each keyword and combine their audio with various noise to synthesize a dataset for evaluation at a signal-to-noise ratios (SNR) of 9, resulting in a balanced dataset with 500 examples. We use a classification task (which wake word does the audio corresponds to; "no-a-wake-word" is added to the category to represent the negative cases.) for the evaluation. The accuracy of the wake word spotting is 94.4%. This accuracy is comparable to the numbers in prior work [20], which is also shown in figure 2. We test the wake word spotting module on cpu, where the latency of the wake word spotting is 0.05ms, and the power consumption is 0.3w. Given its small power consumption, we will have this module always running.

[1]picovoice: https://github.com/Picovoice

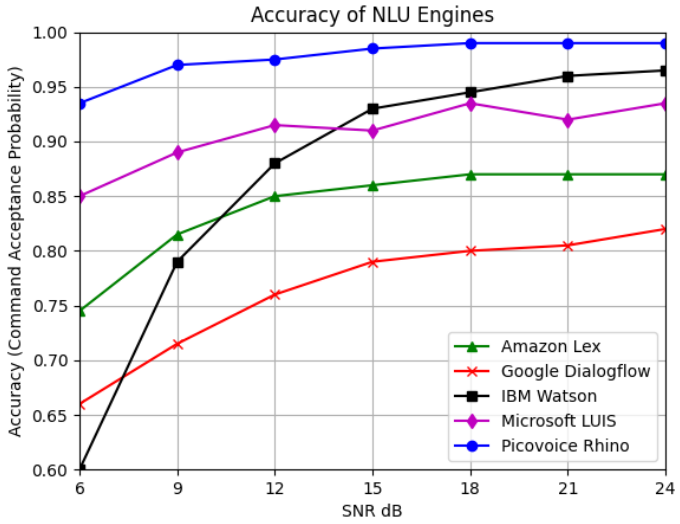


Figure 2: The performance of open-source solutions at different Signal-to-noise ratios [20]

4.3 Visual Recognition

In this subsection, we will discuss the dataset on which we trained our models, the steps for reproduction, as well as the results in terms of accuracy, latency, and energy-consumption.

4.3.1 Dataset

The classification model, as shown in Figure 1, takes in an RGB image and outputs one of the three possible class labels: "CMFD" (correctly masked face), "IMFD" (incorrectly masked face), and "NMFD" (not masked face). Figure 3 shows an example for each of these classes.

(a) CMFD (b) IMFD (c) NMFD

Figure 3: Dataset used for training and quantitative evaluation.

We trained our models over the combination of the synthetic human-mask dataset [21] and its parent unmasked dataset [22]. The reason why we chose the synthetic dataset over real masked datasets is discussed in Section 5.2. For the train-validation-test split, we selected the first $50,000$ indices for training, used the next $10,000$ indices for validation, and held out the remaining $10,000$ indices for off-device testing to evaluate the accuracy of the models. Due to the storage limitation on-device, we only deployed the first 100 images in the testset for the latency evaluation.

4.3.2 Model Selection and Reproduction Steps

Due to the on-device nature of our mask detection project, we replaced the classifier of MobileNet v3-small [18] with a single dense layer following an 2D adaptive average pooling layer. We used the Adam optimizer with a 10^{-3} learning rate and no weight decay. We used unweighted cross-entropy loss (more details in Section 5.2) for the synthetic dataset. As we observed no significant difference in performance regarding different batch sizes, we selected the maximum batch size (in power of 2) allowed by the graphical memory in training. In addition, the transfer learning results reported below are obtained by freezing the pre-trained weights [23] in the feature extractor and solely train the last dense layer.

4.3.3 Classification Accuracy

Model	oA	mA
MobileNet V3 - Large, Transfer Learning, @1024x1024	99.15%	99.16%
MobileNet V3 - Large, End-to-End, @1024x1024	99.98%	99.98%
Simplified Model, Transfer Learning, @1024x1024	98.48%	98.52%
Simplified Model, End-to-End, @1024x1024	99.99%	99.99%
Simplified Model, End-to-End, @224x224	99.89%	99.89%
Simplified Model, End-to-End, @64x64	99.88%	99.88%
Simplified Model, End-to-End, @32x32	99.75%	99.76%

Table 3: Inference Accuracy of the Classification Model

We evaluated the performance in prediction with two metrics: overall accuracy (oA) and the mean accuracy over classes (mA), in order to address false-promising results in case the model is dominated by certain classes. Following the approach we used to address class imbalance (Section 5.2), we initially adopted transfer learning. However, as shown in the first four rows of Table 3, the synthetic dataset is sufficient in size to train the model end-to-end.

Another topic worth investigating is how input resolution influences the prediction accuracy, especially in our latency-sensitive task setting. As demonstrated in the last four rows of Table 3, down-sampling the input resolution to 32×32 leads to minor impact on test accuracy. This helps us to achieve a low on-device latency, as discussed in the next subsection.

We also run our system in-the-wild with our own faces, shown in Figure 4 (the ethical reasons behind why we did not test on random people in CMU are discussed in Section 6). The model generally works well in-the-wild, given that it successfully handles changes in mask color (light azure to heavy cyan) and lighting conditions. However, we did notice a failure case where the model identified a incorrectly-masked face as not-masked-face. One hypothesis for this issue is that the synthetic "IMFD" images (Figure 3b) usually contain a unnatural "wrap" in the mask, and thus biased from actual "IMFD" images. Nonetheless, this phenomenon indicate that we should consider to train the models in a binary-classification manner and classify all incorrectly masked faces together with not-masked faces. This solution probably aligns with our use case better.

4.3.4 Time Performance - Classification Model

We evaluated latency with two approaches: a) **stable mean latency**, which is a useful metric when the classification model is kept running and a sufficient amount of images is always piled to be processed; and b) **cold-start latency**, which is a useful metric when the classification model is put to sleep between handler calls. As such, the second metric better aligns with our trigger-based system.

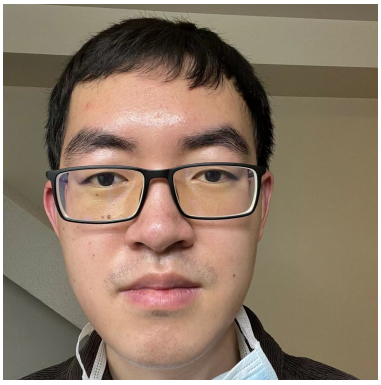
Table 4 provides the comparison based on these two metrics. The second column shows the mean latency calculated without the first 20 images in the synthetic test set, and the third column shows the latency upon a cold start.

The first two rows of Table 4 demonstrates how simplifying the architecture of mobileNetV3 reduced latency by a great margin. On top of that, we resolved the conflict between the high start-up cost from using the on-device GPU and the high model latency when inferring with the CPU, by down-sampling the input images to 32x32. This is made possible by our empirical analysis on model accuracy, and the cold-start time cost was reduced by 250x to merely 268.7 ms.

Moving forward, we performed image resizing offline, which is similar to having the camera to capture images in a native 32x32 resolution. The cold-start latency was significantly reduced by another 35% mainly because it was a burden to I/O 1024x1024 images from the

(a) CMFD - Correctly Classified as CMFD

(b) CMFD - Correctly Classified as CMFD



(c) NMFD - Correctly Classsified as NMFD

(d) NMFD - Correctly Classified as NMFD

(e) IMFD - Incorrectly Classified as NMFD

(f) IMFD - Correctly Classsified as IMFD

Figure 4: Real Human-Mask Faces Evaluated.

SD card (rather than the actual computational cost to resize images). This finding is further asserted by the fact that Jetson Nano issues the following warnings (shown in Figure 5) when reading 1024x1024 images while running both the audio and visual recognition models.

```
Overflow - reader is not reading fast enough.
Overflow - reader is not reading fast enough.
```

Figure 5: Warning from reading original images.

Following the same path to reduce I/O costs with the SD card as much as possible, we hooked the camera stream with the classification model so that image arrays no longer needs to leave memory. This eventually reduced our cold-start latency to 155.6 ms (averaged over 10 cold-starts, the program is put to sleep for 10 seconds between each pair), as shown in the last row in Table 4), faster than 1/436 of our initial brute-force attempt.

Models & Techniques	**Mean Model Latency, excluding first 20 images**	**Cold Start Model Latency,**
MobileNet V3 - Large, GPU, @1024x1024	145.6	67,468.7
Simplified Model, GPU, @1024x1024	36.8	55,251.6
Simplified Model, GPU, @224x224, Online	29.4	50,822.3
Simplified Model, GPU, @32x32, Online	31.8	52,168.6
Simplified Model, CPU, @224x224, Online	2,802.7	2,961.7
Simplified Model, CPU, @32x32, Online	155.1	268.7
Simplified Model, CPU, @32x32, Offline	140.8	179.2
Simplified Model, CPU, @32x32, from Camera	-	**155.6**

Table 4: Latency in [ms/image] of the Classification Model. The time cost of system initialization (one-time, prior to handler calls) and the calculation of accuracy metrics is excluded, as neither will have an impact on the system response time. The latency from the camera will be discussed in the following subsection.

4.3.5 Time Performance - Camera

There are essentially two design choices regarding the camera: to keep the video stream always on and read from the buffer when desired by the classification model, or to launch the camera (and shut it off) every time the trigger called. Thus, we have measured latency caused by the camera (averaged over 10 runs) in each of these two scenarios, as shown in Table 5. It is indicated that launch/terminate the camera at each run would significantly impact (577 ms) the system latency and become the new system bottleneck, whereas running the camera non-stop would lead to a minor cost from using the buffer, and an intermediate burden (89 ms) that slows the classification model down (as the camera competes with the model for power). As the combined latency of the visual recognition module now becomes 244.7 ms under the second approach, it fulfills the engineering requirements to drive the overall latency below 500 ms.

Method	Terminate if not used	Always on
Latency [ms]	577.1	**0.2** (buffer) + **89.1** (slowing the model)

Table 5: Camera Mean Latency. The values here are also reported from cold-starts, averaged over 10 runs with 10 seconds between each two.

4.4 Energy Analysis

In Section 2, we listed a energy-saving criterion. Does our trigger-based system fulfill the task statement? Moreover, is it worth it to keep the camera always on? While it certainly reduced the system latency by a great margin, keeping the camera stream running may introduce extra stand-by energy costs. Hence, it is worthwhile to investigate the extent of this cost before settling down with the current design choice.

Design	Camera - Always On	Camera Off if Not Used	Baseline: No Trigger
Standby Power [W]	4.4	3.3	7.0
Inference Power [W]	7.3	5.8	-
Inference Power Addition [W]	2.9	2.5	-
Inference Time [s/image]	0.250	0.738	-
Inference Cost Overhead [J/image]	0.725	1.845	-
Amortized Total Power [W] @ $\leq$ 1 image/min	4.4	3.3	7.0
Amortized Total Power [W] @ 10 image/min	4.5	3.6	7.0
Amortized Total Power [W] @ 60 Image/min	5.1	5.1	7.0

Table 6: Energy Cost of the System

Table 6 compares the energy costs between the two design choices and the baseline without our trigger system. The last three rows presents the amortized energy cost in three different use cases - mask detection is used sparsely (e.g., for a personal study room), more frequently (e.g., for the entrance to a classroom), or triggered every second (e.g., people wait in lines to enter the room). We argue that our trigger system saves more than 25% energy than the baseline in all use cases, and the price paid ($\leq$ 1.1 W) to keep the camera always on is insufficient to outbid the benefit in latency $>$ 450 ms). Thus, we chose to keep the camera always running in our final design.

5 Challenges and Insights

5.1 Building a sound-base trigger

Having the classification model always running wastes energy, so we only want a very lightweight module to be always "listening". Our original design is to have a audio-based trigger that does not require a wake word. We tried two different approaches: (1) We periodically collect sound. When there is sound, trigger the system. After testing the method, we found that the main issue with this method was the system could not distinguish background noise from user triggers. Also, if the user does not make any sound, the system would fail. (2) Record the sound of typical cases (e. g. "door opening", "people talking"...) and train a classification model based on the dataset. The main issue with this approach is that constructed data are not diverse enough. The accuracy of the resulted model is only 52%, while a random guessing base has an accuracy of 50%. Given the success of wake word spotting in virtual assistant triggering [7], we finally decided to use it in our system.

5.2 Challenges in visual recognition

Mask detection as object detection. We initially treated our task as object detection, that is, to propose a bounding box for each human face present in the scene and output the class prediction for whether the face is correctly masked. However, the state-of-the art methods (Mobilenetv3 + SSDLite[17]/Faster R-CNN [16]) for on-device object detection were still too slow when deployed on the Jetson Nano 2GB. We observed a latency between 6.8 and 7.7 seconds per image, depending on the input resolution and model architecture. Further simplifying the model architecture would cause the model to underfit. Thus, we decided to impose a necessary constraint to the scope of our project by handling single-person scenarios only.

Class imbalance in real human-mask dataset. The real human-mask dataset [10] is small in size and heavily imbalanced when used for classification. The dataset only consists of 853 images in total, but 574 out of the first 700 images we used for training consists of a correctly masked face. Only 49 out of the 700 images do not contain a mask at all, and 77 images contain incorrectly masked faces. This makes it easy for the prediction model to fit to the dominant "correctly-masked" class. Furthermore, even though we applied weighted loss and oversampling to address the class-imbalance issue, 700 images were simply not enough for training. Although we tried to utilize pre-trained weights and only fine-tune the last few layers of our model to make it converge, the high data variance in the dataset was still proven to be too significant to address. To resolve these issues, we trained our models with the large synthetic human mask dataset described in Section 4.3.1.

5.3 Future work

Making the trigger smarter. After deploying to real-world environment, we could collect a large corpus of real-world audio data (if the user permit us to do so), which would solve the cold start problem mentioned in section 5.1. With the data, we might be able to train a trigger module that does not require the user say specific wake words. Note ethical concerns for the data collections part: we have to get permission from the users. Detailed discussion are provided in section 6.

Better photo quality. Figure 6 shows the differences in white balance between images from the phone camera and the on-device camera (same person, same room). As there is no option to change aperture, Figure 6b demonstrates the photo quality captured at the maximum allowed exposure time. The obvious differences in photo quality between the trainingset and device-captured images during inference caused failure. While we tried to address this issue via normalization during both the training and inference, we think the camera has become the bottleneck of our system.

(a) Phone Camera (b) OD Camera @ 33 ms (c) Exposure time $\geq$ 34 ms.

Figure 6: Comparison between Phone Camera and On-device Camera.

6 Ethical Considerations

There may be two ethical issues in our project that need careful consideration.

The first is the issue of data collection. The data we are testing now is still relatively homogeneous, so the dataset used is a synthetic dataset. As the project is further applied, the data handled will be more diverse, so it is necessary to collect a certain amount of real-world data. This may include pictures of many people's faces. This collection may lead to some ethical concerns. One way to mitigate to concern is (1) to make sure the users know when their data are collected; (2) to only collect users' data when they allow; (3) to enable users can still use our system if they do not want their data to be collected.

The second is the actual application process. It is inevitable that we will have to record information about people's faces to do mask detection in the practical application process. Although our devices are not connected to cloud services at all and do not transmit information to the cloud, there will be doubts about the existence of the data on the local end. How to eliminate this suspicion may be a place worth thinking about in the future.

References

[1] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face mask detection using transfer learning of inceptionv3," in *BDA*, 2020.

[2] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic," *Measurement*, vol. 167, pp. 108 288 – 108 288, 2020.

[3] ——, "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection," *Sustainable Cities and Society*, vol. 65, p. 102600, Feb. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670720308179

[4] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. N. P. Vitaladevuni, "Multi-task learning and weighted cross-entropy for dnn-based keyword spotting," in *INTERSPEECH*, 2016.

[5] J. Guo, K. Kumatani, M. Sun, M. Wu, A. Raju, N. Strom, and A. Mandal, "Time-delayed bottleneck highway networks using a dft feature for keyword spotting," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5489–5493, 2018.

[6] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, 2014.

[7] C. Jose, Y. Mishchenko, T. Sénéchal, A. Shah, A. Escott, and S. N. P. Vitaladevuni, "Accurate detection of wake word start and end using a cnn," *ArXiv*, vol. abs/2008.03790, 2020.

[8] AIZOOTech, "FaceMaskDetection," Dec. 2021, original-date: 2020-02-18T07:32:04Z. [Online]. Available: https://github.com/AIZOOTech/FaceMaskDetection

[9] Abhinand, "Face Mask Detection in PyTorch," Jun. 2021, original-date: 2020-11-24T06:45:37Z. [Online]. Available: https://github.com/abhinand5/facemask-detector

[10] Larxel, "Face Mask Detection," 2020. [Online]. Available: https://kaggle.com/andrewmvd/face-mask-detection

[11] G. Jignesh Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face Mask Detection Using Transfer Learning of InceptionV3," in *Big Data Analytics*, ser. Lecture Notes in Computer Science, L. Bellatreche, V. Goyal, H. Fujita, A. Mondal, and P. K. Reddy, Eds. Cham: Springer International Publishing, 2020, pp. 81–90.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv:1512.00567 [cs]*, Dec. 2015, arXiv: 1512.00567. [Online]. Available: http://arxiv.org/abs/1512.00567

[13] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A Survey of Modern Deep Learning based Object Detection Models," *arXiv:2104.11892 [cs, eess]*, May 2021, arXiv: 2104.11892. [Online]. Available: http://arxiv.org/abs/2104.11892

[14] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection," *arXiv:2006.04388 [cs]*, Jun. 2020, arXiv: 2006.04388 version: 1. [Online]. Available: http://arxiv.org/abs/2006.04388

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv:1703.06870 [cs]*, Jan. 2018, arXiv: 1703.06870. [Online]. Available: http://arxiv.org/abs/1703.06870

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv:1506.01497 [cs]*, Jan. 2016, arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv:1801.04381 [cs]*, Mar. 2019, arXiv: 1801.04381. [Online]. Available: http://arxiv.org/abs/1801.04381

[18] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," *arXiv:1905.02244 [cs]*, Nov. 2019, arXiv: 1905.02244. [Online]. Available: http://arxiv.org/abs/1905.02244

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778, iSSN: 1063-6919.

[20] R. R. Ian Lavery, Alireza Kenarsari and et al, "Picovoice: an end-to-end platform for building voice products," 2021. [Online]. Available: "https://github.com/Picovoice/rhino"

[21] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "MaskedFace-Net – A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19," *Smart Health*, vol. 19, p. 100144, Mar. 2021, arXiv: 2008.08016. [Online]. Available: http://arxiv.org/abs/2008.08016

[22] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *arXiv:1812.04948 [cs, stat]*, Mar. 2019, arXiv: 1812.04948. [Online]. Available: http://arxiv.org/abs/1812.04948

[23] "pytorch/vision," Dec. 2021, original-date: 2016-11-09T23:11:43Z. [Online]. Available: https://github.com/pytorch/vision