EagleEye: Wearable Camera-based Person Identification in Crowded Urban Spaces

Juheon Yi
johnyi0606@snu.ac.kr
Seoul National University
Seoul, Korea

Sunghyun Choi
sungh.choi@samsung.com
Samsung Research
Seoul, Korea

Youngki Lee
youngkilee@snu.ac.kr
Seoul National University
Seoul, Korea

Abstract

We present EagleEye, an AR-based system that identifies missing person (or people) in large, crowded urban spaces. Designing EagleEye involves critical technical challenges for both accuracy and latency. Firstly, despite recent advances in Deep Neural Network (DNN)-based face identification, we observe that state-of-the-art models fail to accurately identify Low-Resolution (LR) faces. Accordingly, we design a novel Identity Clarification Network to recover missing details in the LR faces, which enhances true positives by 78% with only 14% false positives. Furthermore, designing EagleEye involves unique challenges compared to recent continuous mobile vision systems in that it requires running a series of complex DNNs multiple times on a high-resolution image. To tackle the challenge, we develop Content-Adaptive Parallel Execution to optimize complex multi-DNN face identification pipeline execution latency using heterogeneous processors on mobile and cloud. Our results show that EagleEye achieves 9.07× faster latency compared to naive execution, with only 108 KBytes of data offloaded.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computer systems organization** → **Real-time system architecture**.

Keywords

Mobile Deep Learning, Person Identification, Heterogeneous Processors, Mobile-Cloud Cooperation, Multi-DNN Execution

ACM Reference Format:
Juheon Yi, Sunghyun Choi, and Youngki Lee. 2020. EagleEye: Wearable Camera-based Person Identification in Crowded Urban Spaces . In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3380881

1 Introduction

Imagine a parent looking for her/his missing child in a highly crowded square. In many cases, a swarm of people in front of her/his eyes will quickly overload cognitive abilities; our motivational study shows that it takes ≈16 seconds to locate a person in a crowded scene (See Section 3 for details). An Augmented Reality (AR)-based service with smart glasses or a smartphone will be extremely helpful if it can capture the large crowd from distance and pinpoint the missing child in real-time (Figure 1). Despite recent advances in person identification techniques using various features such as face [14, 55, 67], gait [27, 66] or sound [7, 19], fast and accurate person identification in crowded urban spaces remains a highly challenging problem.

Figure 1: Example usage scenario of EagleEye: parent finding a missing child. More examples in Section 2.

In this paper, we propose EagleEye, a wearable camera-based system to identify missing person(s) in large, crowded urban spaces. It continuously captures the image stream of the place using commodity mobile cameras, identifies person(s) of interests, and shows where the target is in the scene in (soft) real-time. EagleEye not only shows a good example of future AR applications based on real-time analysis of complex scenes, but also characterizes the workload of future multi-DNN mobile deep learning systems.

Designing EagleEye involves critical technical challenges for both identification accuracy and latency.

• **Recognition Accuracy.** Compared to prior systems [60, 61, 73] that aim at identifying 1 or 2 faces in close vicinity (e.g., engaged in a conversation), the key challenge in building EagleEye is accurately detecting and recognizing distant small faces. In crowded spaces, individual faces often appear very small, with facial details blurred out. Recent Deep Neural Network (DNN)-based face recognition has shown remarkable progress in accurately identifying faces under various unconstrained settings [14, 30, 47] (e.g., variations in pose, occlusion, or illumination). However, the state-of-the-art techniques still fail to provide robust performance for Low-Resolution (LR) faces. Our study shows that Equal Error Rate, the value in the ROC curve where false acceptance and false rejection rates are identical, of the state-of-the-art DNN [14] grows from 9% to 27% when resolution drops from 112×112 to 14×14 (Section 3).

MobiCom '20, September 21–25, 2020, London, United Kingdom

ACM ISBN 978-1-4503-7085-1/20/09…$15.00
https://doi.org/10.1145/3372224.3380881

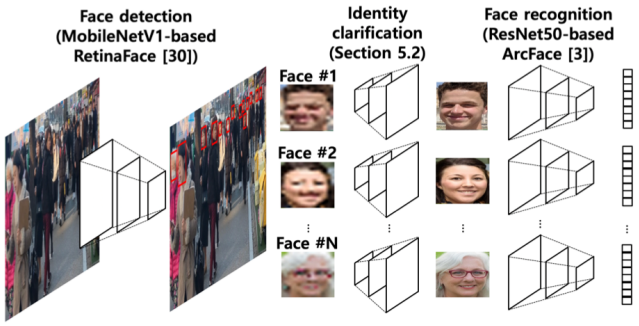


Figure 2: Multi-DNN face identification pipeline.

• **Identification Latency.** More importantly, it is challenging to analyze a crowded scene in (soft) real-time to allow users to sweep large spaces quickly. EagleEye imposes unique challenges compared to recent DNN-based continuous mobile vision systems [28, 35, 53, 58, 62, 68, 71]. Firstly, as shown in Figure 2, EagleEye requires running a series of complex DNNs multiple times for a single scene: face detection network once over a scene, our resolution enhancing network (introduced in Section 5.2) and face recognition network per each face. This is very different from prior systems that run a single DNN only once over a scene. Secondly, each DNN is highly complex to achieve high accuracy, incurring significant latency. Face detectors employ feature pyramid [52] which upsamples features in latter layers and adds up to earlier layers to detect small faces. Also, state-of-the-art recognizers are heavy ResNet-based. Finally, prior work mostly downsample the input frames (e.g., 300×300 [22]) to reduce complexity (this was possible as they analyze a small number of large, primary objects in vicinity). However, EagleEye should run the identification pipeline on high-resolution frames to detect a large number of distant faces that appear very small.

It is highly challenging to run a complex multi-DNN pipeline over high-resolution images in real-time. It is not even trivial to simply port state-of-the-art DNNs to mobile deep learning frameworks (e.g., TensorFlow-Lite) due to the limited number of supported operations. The challenge aggravates considering the execution latency. For instance, a lightweight MobileNet [31] can only process two 1080p frames per second on high-end mobile GPU (Table 1). Naive execution of EagleEye's entire pipeline takes 14 seconds for a scene with 30 faces (Figure 5). We can consider multithreading or offloading, but they are not also straightforward to apply. Multithreading degrades performance due to resource contention over limited mobile resources (e.g. GPU, CPU, memory). Also, 3G/LTE network with low bandwidth is likely the only wireless network available in crowded outdoor environments, making offloading non-trivial.

To tackle the challenges, we design and develop a suite of novel techniques and adopt them in EagleEye.

• **Identity Clarification Network.** We first design a novel end-to-end face identification pipeline to identify small faces accurately. Our key idea is to add *Identity-Clarification Network (ICN)* on conventional 2-step pipeline (detection-recognition) to recover missing facial details in LR faces, thus resulting in a 3-step pipeline (detection-clarification-recognition as shown in Figure 2). ICN adopts a state-of-the-art image super-resolution network as the baseline and innovates it with specialized training loss functions to enhance LR faces for accurate recognition; note that prior super-resolution networks focus on generating perceptually natural images and fail to preserve identities, making them ill-suited for recognition [48] (See Section 5). Also, ICN enables identity-preserving reconstruction using reference images (*probes*) of the target, commonly available in our scenarios (e.g., photos of children provided by parents). We observe that the complexity of LR face recognition results from accepting positive identities rather than denying negative identities (see Section 5.2 for details). Thus, biasing ICN on the target improves LR face recognition accuracy with only a small increase in false positives. Overall, our ICN-enabled pipeline improves true positives by 78% with 14% false positives, against the 2-step identification pipeline.

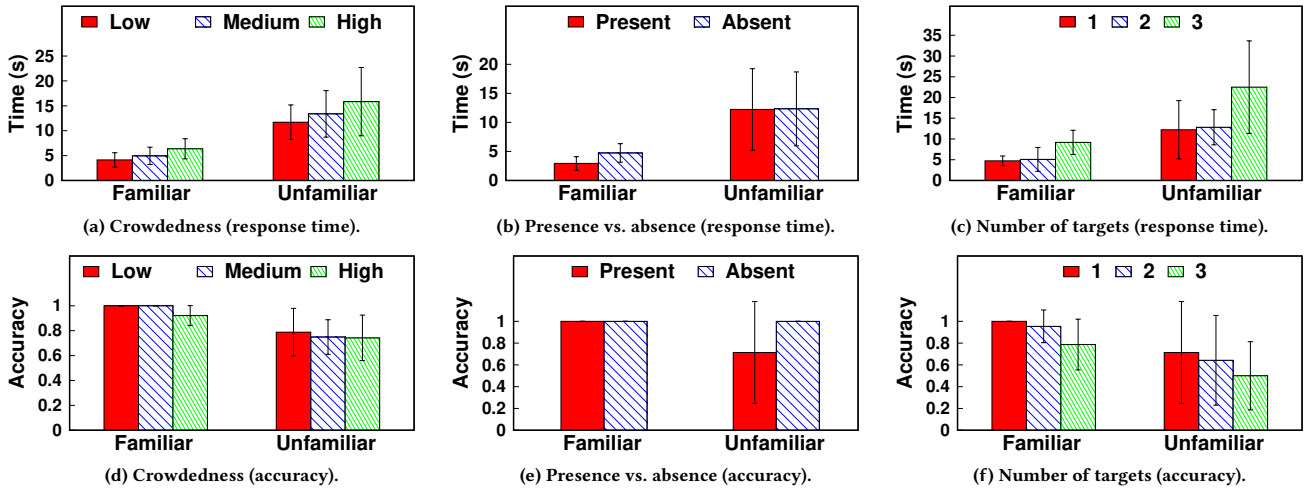
• **Multi-DNN Execution Pipeline.** Our workload (i.e., running a series of DNNs multiple times on high-resolution images) requires a differentiated strategy to optimize the heavy computation. We develop a runtime system with *Content-Adaptive Parallel Execution* to run a multi-DNN face identification pipeline at low latency. The key idea behind this approach is to divide the high-resolution image into multiple sub-regions and selectively enable different components in the pipeline, depending on the content. For instance, ICN is only applied to a region with LR faces while the entire pipeline is not executed for a background region with no faces. Furthermore, we exploit the spatial independence of face recognition workload (i.e., identifying faces in different sub-regions does not have dependency) to parallelize and pipeline the execution on heterogeneous processors on the mobile and cloud. Overall, our technique accelerates the latency by 9.07× with only 108 KBytes of data offloaded.

Our major contributions are summarized as follows:

- To the best of our knowledge, this is the first end-to-end mobile system that provides accurate and low-latency person identification in crowded urban spaces.
- We design a novel face identification pipeline capable of accurately identifying small faces in crowded spaces. By employing Identity Clarification Network to recover facial details of LR faces, we enhance true positives by 78% with 14% false positives.
- We design a runtime system to handle the unique workload of EagleEye (i.e., processing high-resolution images with multiple DNNs for complex scene analysis). We believe this will be an unexplored common workload for many mobile/wearable-based continuous vision applications. We utilize a suite of techniques to minimize the end-to-end latency to as low as 946 ms (9.07× faster than naive execution).
- We conduct extensive controlled and in-the-wild study (with real implementations and various datasets), validating the effectiveness of our proposed system.

2 Motivating Scenarios

Finding a Missing Child. In crowded squares or amusement parks, there are many cases where a parent loses track of her/his child. In such incidents, it is difficult to find the missing child with naked eyes since she/he becomes cognitively overloaded to identify many people in vicinity. EagleEye can help the parent: by sweeping the mobile device to capture the space from distance, it can help quickly pinpoint possible faces and narrow down a specific area to



(a) Crowdedness (response time).
(b) Presence vs. absence (response time).
(c) Number of targets (response time).
(d) Crowdedness (accuracy).
(e) Presence vs. absence (accuracy).
(f) Number of targets (accuracy).

Figure 3: Human cognitive abilities on identifying faces in crowded scenes: response time and accuracy.

search, so that the parent can find the child before the child moves to a different place. Similarly, police officers can use EagleEye to chase criminals in crowded malls, streets, squares, etc.

Children Counting in Field Trips. Teachers in kindergarten regularly take children out for field trips to catch educationally-depicting behaviors hardly captured in classroom settings. However, in reality, teachers spend most of the time counting children to make sure they are in place. EagleEye can be of extensive use to reduce the cognitive burden for the teachers so that they can focus on the original goal.

Social Services for Familiar Strangers. EagleEye can be used to build an interesting social service to connect people. For example, it can be used to identify familiar strangers (people whom we met in the past but do not remember the details) to help with interaction; a person attending a social event can use EagleEye to identify them and get an early heads-up before they are in close proximity to avoid embarrassing moments.

3 Preliminary Studies

To motivate EagleEye, we first conduct a few studies to verify (1) how quickly humans identify face(s) in crowded urban spaces and (2) whether it is feasible in terms of accuracy and speed to employ face recognition algorithms to aid humans' cognitive abilities.

3.1 How Fast Can Humans Identify Faces?

Prior studies report that it takes for humans about 700 ms to detect a face in a scene [46], and about 1 second to recognize the identity of a single face image [40]. We extend the experiments to study how long it takes to identify target(s) in crowded scenes. We first recruit 6 college students (5 males and 1 female, age 24-28) as subjects for dataset collection, and take videos of them blending inside the crowd in various urban spaces including college campus, downtown streets, and subway stations. Next, we recruit 11 students (10 males and 1 female, age 24-32) who are of mutual acquaintances with the subjects (denoted as *Familiar*), and 14 other students (12 males and 2 females, age 20-26) who have never seen the subjects before (denoted as *Unfamiliar*).

In the experiments, the participants are seated in front of the screen with a similar setup as in [46]. Each participant is first shown faces of 1 to 3 target identities. Afterwards, a scene image (1080p resolution) is shown, in which target(s) may or may not exist. The participant clicks the location in the scene where she/he finds each target. Response time is measured as the duration between when the scene is displayed and when the participant finishes identifying all targets. The scenes are classified into three levels of crowdedness (examples are shown in Figure 16): i) *Low* (less than 10 people in close distance with face sizes at least 30×30 pixels), ii) *High* (more than 20 people with face sizes smaller than 14×14), and iii) *Medium* (between *Low* and *High*). Each participant is shown 5 scenes per each category (15 in total) and was asked to be as precise as possible.

Figure 3 shows the response time/accuracy results. Our experimental results are summarized as follows (unless specified, the reported results are on *High* scenes):

- Overall, it takes 6.37 and 15.83 seconds on average to identify familiar and unfamiliar faces in crowded scenes, respectively, showing noticeable cognitive loads.
- It takes longer to identify unfamiliar faces than familiar ones.
- Not only does it take longer to identify a target in more crowded scenes, but the accuracy also drops (Figures 3(a) and (d)).
- Especially for the *Familiar* group, it takes longer to confirm the absence of target than presence. (Figures 3(b) and (e)). We observe that it is because when participants fail to locate the target in the scene, they start looking over again multiple times to confirm their decision.
- It takes longer to identify multiple targets, and accuracy drops as well (Figures 3(c) and (f)).

The above results clearly show the human's vulnerability to cognitive overload. While the study was designed as identifying the target person(s) in a scene image for controllability of the experiment, we conjecture that the cognitive overload will be greater in real-world settings where the scene does not fit into a single view.

3.2 DNN-Based Face Recognition: Status Quo

Faces in crowded spaces captured from a distance experience high variations in pose, occlusion, illumination, and resolution, making

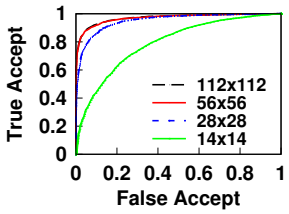


Figure 4: Face verification accuracy.

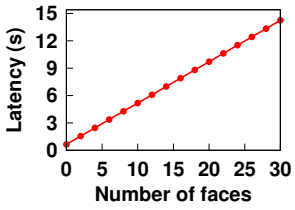
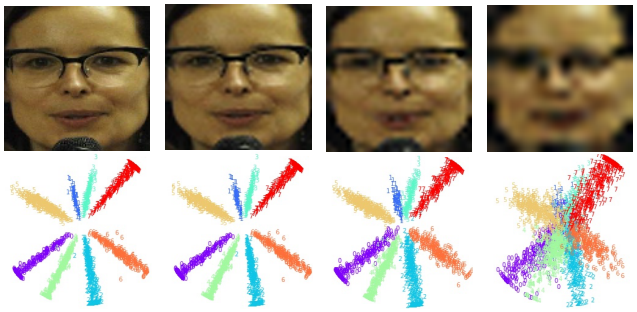


Figure 5: Latency of face identification pipeline.



(a) 112×112. (b) 56×56. (c) 28×28. (d) 14×14.

Figure 6: Feature map visualization for varying resolutions (points with same color represents same identity).

accurate recognition very challenging. While prior algorithms have achieved robust performance (e.g., over 90% accuracy) for the first three [14, 30, 47], the Low-Resolution (LR) face recognition problem has not been fully studied yet.

We conduct a study to analyze the difficulty of LR face recognition. We first train ResNet50 with ArcFace loss [14] on MS1M dataset [25], and test performance on 50 identities in VGGFace2 [6] testset (50 images per identity). Figure 4 shows that verification (determining whether two faces match or not) accuracy drops significantly as resolution decreases. Equal Error Rate (EER), the value in the ROC curve where false acceptance and false rejection rate are identical, grows as high as 0.27 when the resolution is 14×14.

For further analysis, we run a small study with 8 identities in VGGFace2 [6] testset. We train ResNet50 [29] with 2-dimensional output features using SphereFace loss [55]. Figure 6 visualizes the trained features for varying resolutions, where the points with the same color represent the same identity. We observe that when the resolution is high (e.g., 112×112), features for each identity form non-overlapping sharp clusters. However, as resolution drops, clusters become wider and start to overlap with each other, becoming indistinguishable.

3.3 How Fast Can DNNs Identify Faces?

Conventional face identification pipelines operate in a 2-step manner (i.e., face detection on the image and face recognition on each detected face sequentially). In our scenarios, both steps require significant computation. First, the detection network should run on a high-resolution frame to detect distant faces that appear very small. In such settings, providing real-time performance is challenging; Table 1 shows that YOLOv2 [63], one of the fastest networks that can be used for face detection, takes more than 9 seconds to process a 1080p frame. Second, recognition latency increases proportionally

Table 1: Inference time of DNNs with TensorFlow-Lite running on LG V50 (Qualcomm Adreno 640 GPU).

	Model	
Input size	MobileNetV1 [31] (Classification)	YOLO-v2 [63] (Detection)
224×224	24 ms	357 ms
640×360	55 ms	1,477 ms
1,280×720	209 ms	5,009 ms
1,920×1,080	452 ms	9,367 ms

Table 2: Complexity and latency of component DNNs. FLOPs are measured with *tf.profiler.profile()* function.

Task	Model	FLOPs	Inference time
Face detection	RetinaFace [15] (MobileNetV1-based)	9.54 G	648 ms per 1080p image
Identity clarification	Ours (Section 5.2)	15.84 G	166 ms per 14×14 face
Face recognition	ArcFace [14] (ResNet50-based)	10.21 G	287 ms per 112×112 face

to the number of faces, which can be very large in crowded scenes. Figure 5 shows that naively running the state-of-the-art multi-DNN face identification pipeline composed of DNNs summarized in Table 2 [1] takes more than 14 seconds to process a scene with 30 faces even on a high-end LG V50 with Qualcomm Adreno 640 GPU.

3.4 Summary

In crowded spaces, humans become cognitively overloaded, clearly necessitating the need for a system to aid their abilities. However, DNN-based face recognition algorithms cannot be applied directly as they fail to identify LR faces accurately, and naive execution incurs significant latency.

4 EagleEye: System Overview

4.1 Design Considerations

High Recognition Accuracy. Our primary objective is to design a face identification pipeline capable of accurately identifying target(s) in crowded spaces, even when he/she appears very small.

Soft Real-Time Performance. While enabling an accurate face identification pipeline, our goal is to provide soft real-time performance (e.g., 1 fps) for application usability. We aim to devise techniques to optimize various latency components in the end-to-end system while incurring a minimum loss in recognition accuracy.

Use of Commodity Mobile Camera. We aim at achieving high accuracy using frames captured by cameras of commodity smartphones or wearable glasses (e.g., 1080p frames at 30 fps [17]). If cameras with higher resolution or optical zoom-in are available, our approach can help cover a more extensive search area.

Minimal Use of Offloading. In our common use cases (i.e., a moving user in crowded outdoor environments), we assume that

[1] These are the state-of-the-art not only in terms of accuracy but also in terms of complexity. For face detectors, comparable networks are heavy VGG16 [65] or ResNet101 [33]-based. Recent face recognizers are based on 64-layered ResNet [55, 67].

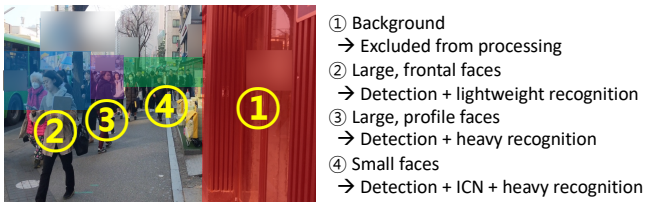


Figure 7: Operation of EagleEye in a nutshell.

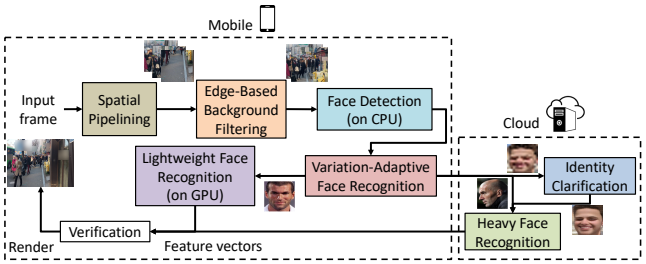


Figure 8: EagleEye system overview.

the availability of edge servers and Wi-Fi connectivity are limited. For robust performance, we aim to minimize the amount of data offloaded to the cloud and run most of the computation on local.

4.2 Operational Flow

Figure 7 shows the nutshell operation of EagleEye: given a crowded scene image, we adaptively process each region with different pipelines depending on the content. For background regions, we do not run any DNN. For non-background regions, we run face detection and adaptively select the latter part of the pipeline to process each detected face based on different variations: i) large, frontal faces (which are very easy to recognize) are processed with a lightweight recognition network, ii) large, profile faces (whose resolutions are sufficient but pose variations make recognition difficult) are processed with a heavy recognition network, and iii) small faces are first processed with *Identity Clarification Network)* (which enhances resolution of LR faces for accurate recognition) and then with heavy recognition network. Finally, exploiting the spatial independence of the task, we process each region and face in parallel on heterogeneous processors on mobile and cloud.

Figure 8 shows the operational flow of EagleEye. We employ *Content-Adaptive Parallel Execution* to run the complex multi-DNN face identification pipeline at low latency using heterogeneous processors on mobile and cloud. Given an input frame, *Spatial Pipelining* first divides it into spatial blocks, so that each block can be processed in a pipelined and parallel manner. Afterwards, *Edge-Based Background Filtering* rules out background blocks with edge intensity lower than a threshold. For the remaining blocks, we detect faces on the mobile CPU. Each detected face is scheduled to a different pipeline by *Variation-Adaptive Face recognition.* Large, frontal faces are processed by lightweight recognition network running on mobile GPU. The rest is offloaded to the cloud, where large, profile faces are processed by heavy recognition network, and small faces are processed by ICN and then by heavy recognition network.

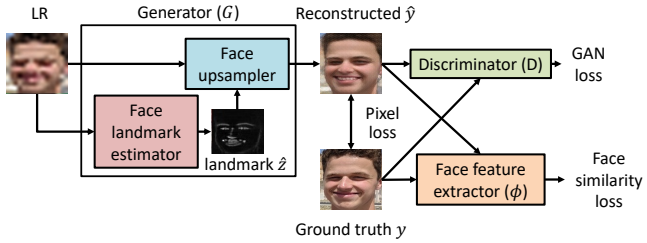


Figure 9: Identity Clarification Network: overview.

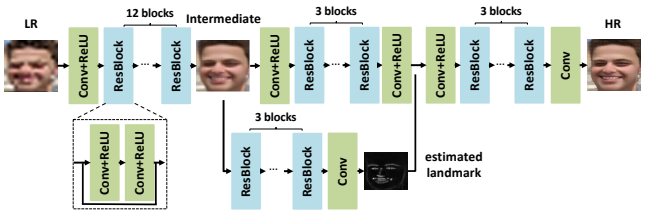


Figure 10: Generator network architecture.

5 Identity Clarification-Enabled Face Identification Pipeline

In this section, we detail our novel 3-step face identification pipeline. It operates as shown in Figure 2: i) detect faces in the scene, ii) enhance each LR face with ICN, and iii) extract feature vectors for each face with recognition network.

5.1 Face Detection

The first step of our pipeline is face detection. The detection network should be accurate in detecting small faces, since faces missed in this step would lose the chance of being identified at all. At the same time, it should be lightweight so that it can run in (soft) real-time. We experiment various state-of-the-art DNNs and select RetinaFace detector [15] with MobileNetV1 [31] backbone for the following reasons: i) it adopts context module which has been proven very effective in detecting small faces [59, 65], and ii) it is the fastest among the state-of-the-art group due to its lightweight backbone network (others are heavy VGG16-based [65] or ResNet101-based [33]).

5.2 Identity Clarification Network

LR faces lack details crucial for identification. To enhance recognition accuracy, we design *ICN*, which enhances the resolution of LR faces using Generative Adversarial Network (GAN). As conventional GANs reconstruct faces with significant distortion from the original identity (Figure 11), we adapt GAN to reconstruct identity-preserving faces by using various loss functions, as well as a specialized training methodology (*Identity-Specific Fine-Tuning*).

Network Architecture. Figure 9 shows the overview of ICN. For generator G, we adopt Residual block [29]-based architecture similar to FSRNet [12] as shown in Figure 10, which has shown high reconstruction performance. Furthermore, we employ anti-aliasing convolutional and pooling layers [72] to improve robustness to pixel misalignment in face detection and cropping process. We employ various additional networks and loss functions to train ICN to preserve identity as follows.

Figure 11: GANs reconstruct realistic faces, but fail to preserve the face identity.

Following the convention in super-resolution [1, 51], the generator is trained to minimize the pixel-wise L2 loss between the reconstructed face and the ground truth,

$$L_{pixel} = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(\|y_{i,j} - \tilde{y}_{i,j}\|^2 + \|y_{i,j} - \hat{y}_{i,j}\|^2\right), \quad (1)$$

where H, W are height and width, $\tilde{y}$ and $\hat{y}$ are the intermediate and final High-Resolution (HR) face in Figure 10, respectively, and y is the ground truth.

As reconstructing HR faces is very challenging, recent studies have shown that employing a facial landmark estimation network to guide the reconstruction process yields superior performance [4, 12]. We adopt the approach to estimate facial landmarks from the intermediate HR face instead of directly from the LR face. The facial landmark estimation network is trained to minimize the MSE between estimated and ground truth landmarks,

$$L_{landmark} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i,j}\|z^n_{i,j} - \hat{z}^n_{i,j}\|^2, \quad (2)$$

where $\hat{z}^n_{i,j}$ is the estimated heatmap of the n-th landmark at pixel (i, j) and z is the ground truth.

Recent studies have shown that GAN [21] plays an important role in reconstructing realistic images. We employ WGAN-GP [23] for improved training stability, whose loss is defined as:

$$L_{GAN} = -D(\hat{y}) = -D(G(x)), \quad (3)$$

where $G(x)$ denotes the HR face reconstructed by the generator, and D denotes the discriminator that classifies whether the reconstructed face looks real or not, which is trained by minimizing the following loss function (refer to the original paper [23] for details),

$$L_{Discriminator} = D(\hat{y}) - D(y) + \lambda(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2. \quad (4)$$

We also enforce the reconstructed face to have similar features with the ground truth by minimizing the face similarity loss

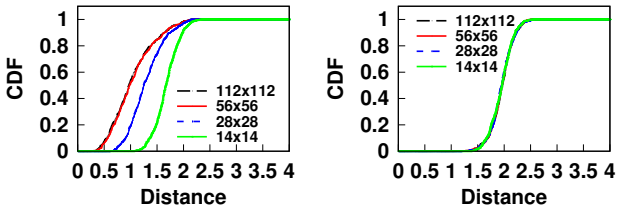
$$L_{face} = \frac{1}{d}\|\psi(y) - \psi(\hat{y})\|^2, \quad (5)$$

where $\psi(\cdot)$ denotes d-dimensional feature vector extracted by the VGG16 network trained on ImageNet [13].

Finally, the above loss functions are combined as a weighted sum and minimized in the training process,

$$L_{total} = L_{pixel} + 50 \cdot L_{landmark} + 0.1 \cdot L_{GAN} + 0.001 \cdot L_{face}. \quad (6)$$

Identity-Specific Fine-Tuning. Baseline ICN aims to adapt conventional GANs to overcome their limitation (i.e., reconstructing perceptually realistic faces at the cost of significant distortion from the ground truth). However, we notice that it still often reconstructs faces with distorted identity from the original. Accordingly, we need another step to employ ICN for our purpose of accurate recognition.



(a) Same identity pair. (b) Different identity pair.

Figure 12: CDF of face distances for varying resolutions.

Before introducing our approach, we further dig deeper into the LR face recognition problem. Figure 12 shows that as resolution decreases, L2 distance between features of faces with the same identity increases significantly, whereas those of different identities remain identical. In other words, the difficulty of LR face recognition comes from the hardship of accepting positive pair of faces, rather than denying negative pairs. Therefore, LR face recognition accuracy can be enhanced if we can bring back the features of faces with the same identity close to each other.

To this end, we develop *Identity-Specific Fine-Tuning* to re-train ICN with reference images (*probes*) of the target, which is commonly available in our target scenarios (e.g., photos of children provided by parents). Such re-training process enables ICN to instill the facial details of the target into the input LR face, thus making it easier to recognize when a LR face of target identity is captured. While such biasing may also increase false positives caused by LR faces that do not match the target identity pulled towards the probes, we observe that such cases only occur for ones that are very close to the target in feature space, thus yielding gain in true positives outweighing false positives (78% vs. 14% as shown in Section 8.3).

Probe Requirements. To fine-tune the ICN to instill facial details of the target, Identity-Specific Fine-Tuning requires probe images with rich facial details. As an initial study we collect the probes with high-resolution, and leave detailed analysis of the impact of the composition of probes (e.g., pose or occlusion) as future work.

Data Augmentation. To diversify the probes as well as boost robustness to various real-world degradation, we also utilize the following augmentation techniques:

- **Illumination.** Change value (V) component in HSV color space.
- **Blur.** Apply Gaussian blur with varying kernel sizes.
- **Noise.** Add Gaussian noise with varying variance.
- **Flip.** Apply horizontal flip.
- **Downsampling.** Resize with different downsampling kernels. (e.g., bicubic, nearest neighbor).

Scalability. Finally, the overhead of fine-tuning the baseline ICN pre-trained on a large-scale face dataset to a specific target identity is not significant (e.g., takes about 20 minutes on a single NVIDIA GTX 2080Ti GPU). Thus, we expect it can be flexibly re-trained at deployment as the target changes.

5.3 Face Recognition and Service Provision

At the final stage, state-of-the-art ResNet50-based ArcFace [14] runs on each face to extract 512-dimensional feature vector, which is compared to that of the target probes. Those with distance below the threshold are highlighted on the screen so that the user can take further actions. To compensate for possible motion between the

(a) Raw frame. (b) Edges. (c) Filtered.

Figure 13: Edge-based background filtering.

image capture and output rendering (about 1 second as our evaluation shows), we can employ motion tracking to shift the bounding boxes using approaches used in prior detection systems [10, 53].

6 Real-Time Multi-DNN Execution

In this section, we detail our runtime system to execute the multi-DNN face identification pipeline at low latency. We start with workload characterization by identifying the sources of latency, followed by our proposed *Content-Adaptive Parallel Execution*.

6.1 Workload Characterization

Sequential Execution of Multiple DNNs. Identifying target person(s) in a crowded scene requires a sequential execution of multiple complex DNNs (i.e., face detection, identity clarification, and recognition) whose individual complexities are summarized in Table 2.

High-Resolution Input. Conventional object detection networks downsample the input images to reduce complexity (e.g., 416×416 [63] or 300×300 [22]). However, in our case, the input image size should be retained large (e.g., 1080p), so that small faces have enough pixels to be detected. As the complexity of DNN inference grows proportionally to the image size, latency becomes significant when processing such high-resolution images.

Repetitive Execution for Each Face. ICN and recognition network must repeatedly run for each face detected by the face detection network. The latency increases proportionally to the number of faces in the scene, which becomes significant in crowded spaces.

6.2 Content-Adaptive Parallel Execution

6.2.1 *Optimization Strategies*

Content-Adaptive Pipeline Selection. We adaptively process each region of the image with different pipelines depending on the content. This helps optimize the latency incurred when processing a large number of faces, while maintaining high recognition accuracy.

Spatial Independence and Parallelism. Identifying faces in different regions of the image is spatially independent. Furthermore, recognizing each detected face can be executed simultaneously. To take full advantage of such opportunities for parallelism, we divide the image into spatial blocks and process them in a pipelined and parallel manner using heterogeneous processors on mobile and cloud. This helps optimizing the latency of multi-DNN execution on high-resolution images.

6.2.2 *Content-Adaptive Pipeline Selection*

We develop techniques to optimize the latency of complex multi-DNN face identification pipeline execution while maintaining high accuracy. Specifically, *Edge-Based Background Filtering* rules out

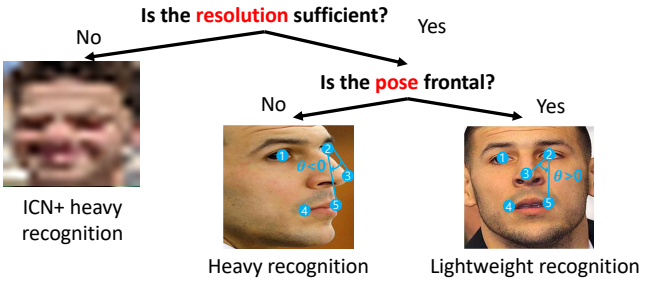


Figure 14: Variation-Adaptive Face Recognition.

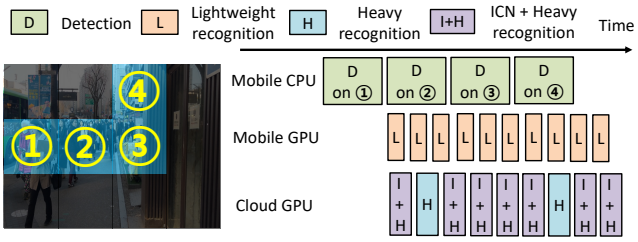


Figure 15: Spatial Pipelining on heterogeneous processors.

background regions where faces do not exist at all. *Variation-Adaptive Face Recognition* selects different recognition pipelines depending on recognition difficulty.

Edge-Based Background Filtering. Running face detection on regions where faces do not exist at all (e.g., background) is a wasteful computation. To mitigate the problem, we use edges in the image to rule out such regions before running the identification pipeline. Specifically, given a frame as shown in Figure 13(a), we detect edges as in Figure 13(b), filter out blocks with edge intensity below a threshold as depicted in Figure 13(c), and run face detection only on the remaining blocks. Note that edge detectors are extremely lightweight, especially considering that we can even detect edges on downsampled images. For example, the time complexity of Canny edge detector [5] for $H \times W$ frame is $O(HW \cdot \log(HW))$, and it runs in less than 2 ms for 360p frame on LG V50. Thus, its overhead is minimal even when the edge detection is not effective for some scenes having full of objects and no background regions.

Variation-Adaptive Face Recognition. State-of-the-art recognition networks are designed very complex (e.g., heavy ResNet backbone with a large number of batch normalization layers) to accurately identify faces even under high variations in pose, illumination, etc. However, employing such heavy networks for faces in ideal conditions is an overkill. For example, MobileFaceNet [9] and ResNet50-based ArcFace [14] achieve comparable accuracy on LFW [34] dataset composed of large, frontal faces (98.9% vs. 99.3%), whereas inference time differs by more than 20× (14 ms vs. 287 ms). Therefore, we aim to optimize latency by adaptively processing each face depending on its variation (i.e., recognition difficulty).

Figure 14 depicts our Variation-Adaptive Face Recognition, which utilizes the size of bounding box and 5 face landmarks detected by RetinaFace [15] detector. First, small faces are processed by ICN and then by ResNet50-based ArcFace [14]. For large faces, we estimate the pose using the detected landmarks; for example, if the angle between the line connected by points (2, 3) and (2, 5) measured in

Algorithm 1 Combined operational flow of EagleEye

```
1:  while application is running do
2:      Result ← {}
3:      Frame ← acquireFrameFromCamera()
4:      Edges ← EdgeDetector(Frame)
5:      NonBackground ← BackgroundFilter(Edges)
6:      for Block in NonBackground do
7:          Faces ← FaceDetection(Block)
8:          for face in Faces do
9:              Result ←Result∪AdaptiveFaceRecognition(face)
10:         end for
11:     end for
12:     Render Result on screen
13: end while
```

counterclockwise direction is negative, we can tell that the face is looking to the right. As faces with pose variations are difficult to accurately identify, they are also processed by ResNet50-based ArcFace (ICN is not needed here as resolution is already sufficient). The remaining faces (large and frontal) which are easy to identify are processed by MobileFaceNet [9].

6.2.3 *Execution Planning*

We optimize latency of multi-DNN face identification pipeline by scheduling each component DNN execution to the most suitable processor on mobile and cloud.

Offloading Decision. As our target scenarios assume crowded outdoor environments with congested 3G/LTE network, offloading high-resolution images for detection is impractical; instead, we offload only the detected faces. Specifically, LR faces are suitable for offloading, as their data sizes are very small (e.g., 14×14 pixels) whereas the required computation (i.e., ICN and heavy recognition) incurs significant latency on mobile (e.g., 166+287 ms). We also offload large, profile faces, and leave only the large, frontal faces to be processed by lightweight recognition on mobile.

Mobile Processor Mapping. The mobile needs to run both detection and lightweight recognition. However, simply multithreading the execution on GPU does not help optimize latency, as mobile GPUs lack preemptive multitasking support. Therefore, we utilize heterogeneous processors (CPU and GPU) to parallelize the execution. As dynamically switching the mapping over time is challenging due to high latency overhead of loading DNN on mobile GPUs (e.g., 2 seconds for 118 MB ResNet50-based ArcFace [14] on LG V50 with TensorFlow-Lite), we statically run detection on CPU and recognition on GPU considering the following aspects:

- **Memory I/O.** Running face detection on GPU requires high-resolution images loaded onto GPU memory, and output feature maps from different stages in the feature pyramid (whose size is proportional to the input image size) copied back to CPU to be post-processed to bounding boxes. Considering memory overhead, it is more suitable to run face recognition on GPU whose input/output are small-sized faces and 1D feature vectors.
- **Inference time.** Besides, we observe that the inference speed slowdown of RetinaFace detector running on CPU is 1.22× (648 vs. 793 ms), whereas it is 2.07× for MobileNetV1-based ArcFace recognizer (14 vs. 29 ms). Therefore, running detection on CPU and recognition on GPU is more feasible to optimize overall latency, especially when the number of faces is large.

(a) Low. **(b) Medium.** **(c) High.**

Figure 16: In-the-wild dataset examples.

Table 3: Average and standard deviation of the composition of each face type in the test dataset.

	Low	Medium	High
Large frontal	3.00±2.62	3.85±2.11	5.20±3.73
Large profile	1.00±0.76	1.50±1.49	2.8±1.78
Low-resolution	3.07±1.75	5.45±2.50	8.87±3.64
Total	7.07±1.79	11.10±3.74	16.87±4.78

6.2.4 *Spatial Pipelining*

To further optimize the latency, we exploit the spatial independence of the workload by processing each image sub-block in a pipelined and parallel manner. As depicted in Figure 15, given non-background blocks in a scene, we detect faces in one block on mobile CPU, while simultaneously processing faces detected in another block on mobile and cloud GPU.

Note that we need to divide the image into blocks in an overlapping manner with padding, so as to prevent faces from being split across different blocks (and thereby failing to be detected). While fine dividing increases the chance of higher parallelism, it also increases the computational overhead due to padding. Based on our empirical evaluation on such tradeoff in Section 8.4.3, we divide an image into 4x4 blocks.

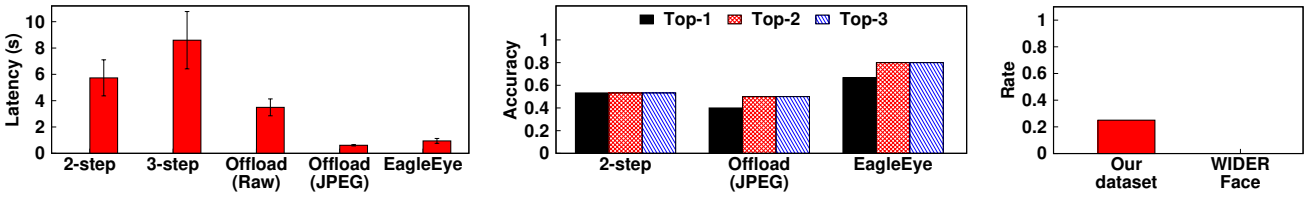
6.2.5 *Putting Things Together*

Algorithm 1 summarizes the combined operational flow. Upon acquiring a frame from the camera, we detect edges (line 4) and filter out background (line 5). For non-background blocks (line 6), we run face detector on CPU (line 7) and process each face adaptively in mobile or cloud GPU (lines 8–10) in a pipelined and parallel manner. Finally, the recognition result is rendered on the screen.

7 EagleEye Implementation

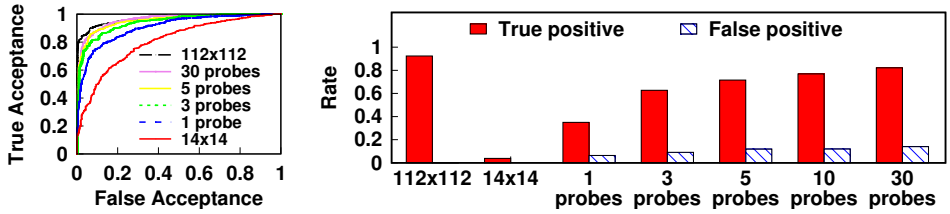
Mobile. We implement the mobile side of EagleEye on two commodity smartphones running on Android 9.0.0: LG V50 with Qualcomm Snapdragon 855 and Adreno 640 GPU and Google Pixel 3 XL with Qualcomm Snapdragon 845 and Adreno 630 GPU. Unless stated otherwise, we report evaluation results on LG V50. RetinaFace [15] and MobileFaceNet [9] are implemented using TensorFlow 1.12.0 and converted to TensorFlow-Lite for mobile deployment. Image processing functions (edge detection, face cropping) are implemented using OpenCV Android SDK 3.4.3. The mobile device is connected to the server via a TCP connection.

Cloud. We implement the cloud side of EagleEye on a desktop PC running on Ubuntu 16.04 OS, equipped with Intel Core i7-8700 3.2



(a) End-to-end latency. (b) Top-K Accuracy. (c) False Alarm increase.

Figure 17: EagleEye performance overview.



(a) Ideal case. (b) Our scenario.

Figure 18: Performance of Identity Clarification Network.

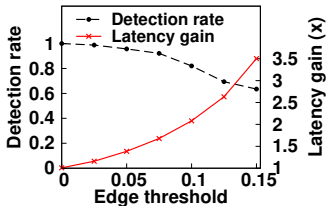


Figure 19: Edge-Based Background Filtering.

GHz CPU and an NVIDIA RTX 2080 Ti GPU (11 GB RAM). We implement most of the cloud-side functions in Python 3.5.2 and utilize Numba [43], a Just-In-Time (JIT) compiler for Python, to accelerate the performance comparable to C/C++. ICN and ResNet50-based ArcFace [14] are implemented using TensorFlow 1.12.0.

8 Evaluation

8.1 Experiment Setup

DNN Training. We train our face detector on WIDER Face [69] train dataset. Also, we train our face recognizers (both the light and heavy models) on MS1M [25] dataset. ICN is trained on FFHQ dataset [41]. As FFHQ dataset does not contain face landmark labels, we employ state-of-the-art network [3] to estimate face landmarks and use them as ground truth labels.

Datasets. We evaluate EagleEye with two different datasets: *single faces* and *crowded scenes*. For single faces, we collect 50 identities in VGGFace2 [6] testset, with 50 samples per each identity. For the scenes, we use in-the-wild images (mostly containing faces of a single ethnicity group) collected and classified depending on crowdedness (i.e., *Low*, *Medium*, and *High*) as described in Section 3.1 (examples are shown in Figure 16). The detailed composition of the faces in the scene dataset are summarized in Table 3. We also categorize the dataset depending on whether the target is present or not. Furthermore, we also collect scene images from WIDER Face [69] test dataset, which contains diverse ethnicity groups (15 images per each crowdedness category).

Evaluation Protocols and Metrics. We evaluate the performance of EagleEye with the following evaluation protocols and metrics:

• **Latency:** the time interval between the start and the end of the pipeline execution, measured on mobile.

• **Equal Error Rate (EER):** the value in the ROC curve where the false acceptance and false rejection rates are identical.

• **True Positive (TP) & False Positive (FP):** the rate in which the test faces are correctly/wrongly accepted as the target, respectively, given a fixed threshold.

• **Top-K Accuracy:** the percentage of images in which the distance between the target face and the probe is within the top K-th among all faces in the scene (applies for scenes with the target present). This can also be interpreted as recall for a single target.

• **False Alarm:** the percentage of images in which the system falsely detects that the target is present in the scene (applies for scenes with the target absent).

Comparison Schemes. We compare the performance of EagleEye with the following comparison schemes:

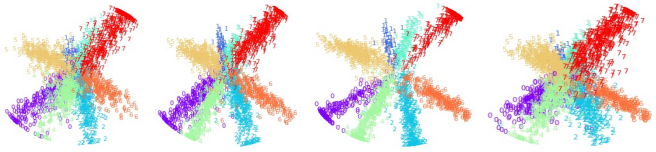
• **2-step baseline** runs the conventional 2-step identification pipeline (MobileNetV1-based RetinaFace and ResNet50-based ArcFace) all on the mobile sequentially.

• **3-step baseline** runs our proposed 3-step identification pipeline (MobileNetV1-based RetinaFace, ICN, and ResNet50-based ArcFace) all on the mobile sequentially.

• **Full offload** fully offloads the image to the cloud over LTE and runs the 3-step identification pipeline. The image is sent either raw or after JPEG compression. Note: we run this experiment under a normal LTE performance (≈11 Mbps), and it is likely that the performance of full offloading could be worse than what we report in crowded outdoor environments.

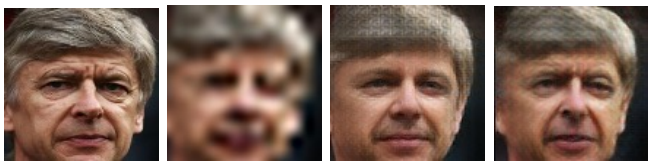
8.2 Performance Overview

We first evaluate the overall performance of EagleEye compared with alternatives for *High* scenes. Figure 17 shows the results. Firstly, as shown in Figure 17(a), EagleEye outperforms the latency of the 3-step baseline by 9.07× (with only 108 KBytes of data offloaded to the cloud). Also, it shows the highest Top-K accuracy (80% of Top-2 accuracy vs. 53% for the 2-step baseline) at the reasonable increase of false alarms (Figure 17(b) and (c)). A reason for the increase of the false alarm is that our dataset contains the faces of the same ethnicity group, increasing the chance of similar-looking identities with the target. For the WIDER Face dataset which contains more diverse ethnicity groups, we did not observe any false alarm increase. Note that the accuracy and false alarms are better with *Medium* and *Low* scenes, as shown in Figure 25.



(a) 14×14. (b) Baseline ICN. (c) Ideally fine-tuned. (d) Fine-tuned to identity #6 (orange).

Figure 20: Feature map visualization for ICN.



(a) 112×112. (b) 14×14. (c) Baseline. (d) Fine-tuned

Figure 21: Reconstruction example of ICN.

(a) Raw frame. (b) Detected edges. (c) 59% of blocks left. (d) 30% of blocks left. (e) 8% of blocks left.

Figure 22: Example operation of Edge-Based Background Filtering.

Interestingly, while fully offloading JPEG-compressed images achieves the smallest latency, we observe that its Top-2 accuracy drops to 50% as shown in Figure 17(b), as compression artifacts hinder reconstruction performance of ICN and recognition network. We could apply video compression (e.g., H.264) to minimize latency more, but it would further degrade performance as it adopts motion vector-based inter-frame encoding, incurring additional distortion in the faces. As compression artifact reduction is a challenging problem, recent attempts have been made to design specialized DNNs for it [24, 56]. Thus, we conjecture that solving this issue will not be trivial and leave detailed investigation as future work.

8.3 Identity Clarification Network

We evaluate the performance of ICN with a varying number of probes used for Identity-Specific Fine-Tuning. Figure 18 shows the results for (a) ideal cases (ICN trained for individual faces) and (b) our scenarios (ICN trained with a target identity), respectively. For the ideal case, ICN recovers the accuracy of 14×14 faces similar to 112×112 with about 5 probes only. For our scenarios, as the number of probes increases, ICN injects more facial details of the target to the input LR face, significantly increasing the chance to identify the target with a relatively small increase in the FP. Figure 18(b) shows that the gain in TP (78%) outweighs that of FP (14%). We further analyze the reasons for accuracy improvement using a simple example with the 8 identities (the same setting as in Section 3.2). From the 14×14 LR faces whose features severely overlap with each other (Figure 20(a)), the baseline ICN (without fine-tuning) clusters each identity's features more tightly, but some overlapping regions still remain (Figure 20(b)). When enhancing each LR face with ICN fine-tuned with corresponding probes, we observe each feature cluster is separated even more clearly (Figure 20(c)). In the case of applying ICN fine-tuned to target identity #6 (orange samples), Figure 20(d) shows that the samples corresponding to the target are grouped to form a tight cluster. While other identity groups are pulled towards the target, the cases where the pulled samples overlap with those of the target (false positive) are not dominant.

Finally, Figure 21 shows the face reconstruction examples of ICN. Baseline ICN reconstructs a face quite similar to the ground truth but lacks some fine attributes (e.g., wrinkles) in the ground truth face. Identity-Specific Fine-tuning enables the ICN to instill such details in the reconstructed face, thus enabling accurate recognition.

8.4 Content-Adaptive Parallel Execution

8.4.1 *Edge-Based Background Filtering*

Next, we evaluate the performance of our Edge-Based Background Filtering method. Figure 19 shows the detection rate and latency gain as we increase the edge intensity threshold. Higher threshold results in higher latency gain, but at the cost of loss in detection rate. We observe threshold between 0.05 and 0.08 balances the tradeoff, and we empirically set it as 0.08 which achieves 1.76× latency gain with 8.7% loss in detection rate. Figure 22 shows an example of image blocks being filtered for different thresholds (covered in black in Figure 22(c)–(e)). With a higher threshold, blocks containing large faces starts to get ruled out. The tradeoff can be more aggressively made if our system can only focus on identifying distant, small faces while relying on users to recognize large, closer faces.

8.4.2 *Variation-Adaptive Face Recognition*

To evaluate the effectiveness of Variation-Adaptive Face Recognition, we synthesize a group of faces, which contains 10 samples per each case classified in Figure 14. We compare our technique (adapting the recognition pipeline based on pose and resolution) with the following baselines: (i) running a lightweight recognizer (MobileFaceNet [9]) on all faces (denoted as *Base light*), (ii) running ICN and a heavy recognizer (ResNet50-based ArcFace [14]) on all faces (denoted as *Base full*), (iii) adaptively applying the lightweight and heavy recognizers based on the resolution only (denoted as *Res-only*). We did not apply our parallel and pipelined execution for this experiment so that only the relative comparisons are meaningful.

Figure 23 shows that our approach achieves comparable accuracy with *Base full*, while reducing the latency by 1.80×. On the contrary, *Base light* and *Base full* suffer from low accuracy and significantly high latency, respectively. The *Res-only* yields fairly high accuracy gain with small latency overhead, but the accuracy remains lower than *Base full* as large profile faces processed by light MobileFaceNet results in inaccurate decisions.

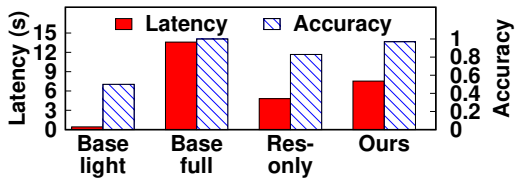
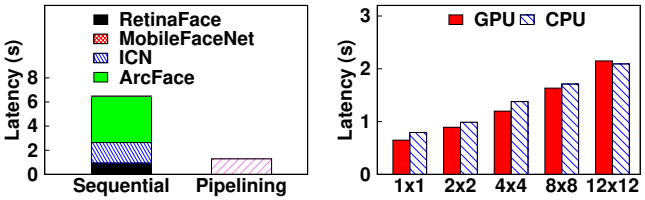
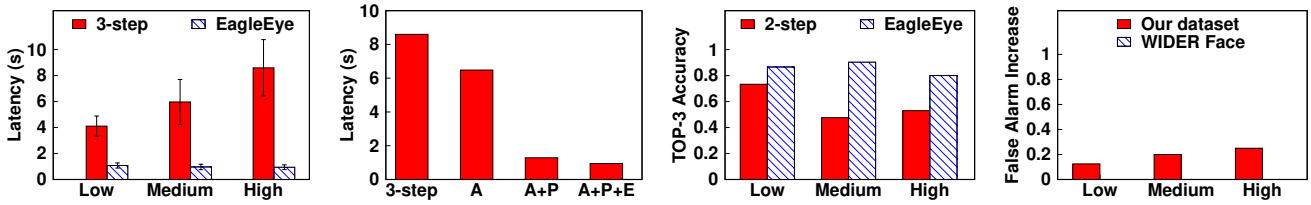


Figure 23: Performance of Variation-Adaptive Face Recognition.



(a) End-to-end latency. (b) Face detection latency.

Figure 24: Performance of Spatial Pipelining.



(a) Latency. (b) Latency breakdown. (c) Top-3 Accuracy. (d) False alarm increase.

Figure 25: End-to-end latency for varying crowdedness.

8.4.3 *Spatial Pipelining*

Figure 24(a) shows the performance of Spatial Pipelining on *High* scenes. Our pipelining yields 5.03× acceleration compared to the baseline that runs face detection and processes faces with Variation-Adaptive Face Recognition sequentially using the mobile GPU (denoted as *Sequential*).

We further analyze the effect of the number of blocks to parallelize. Figure 24(b) shows the latency of face detector with varying number of blocks. We need to divide the image in an overlapping manner to prevent faces split across blocks, which increases computational overhead due to repetitive face detection on the overlapping regions. Thus, the larger the number of blocks, the higher the latency overhead. Considering the tradeoff between such cost and gain for parallelism, we divide the image into 4×4 blocks by default.

8.5 Performance for Varying Crowdedness

Figure 25(a) shows the end-to-end latency comparison of 3-step baseline and EagleEye. The latency of EagleEye remains similar regardless of crowdedness, mainly because we pipeline and parallelize the execution on mobile and cloud. However, the latency of 3-step increases with more crowded scenes since recognition latency increases proportionally to the number of faces. Accordingly, we conjecture that the latency gain will be greater as crowdedness increases even more. Furthermore, current bottleneck remains at the face detection stage, and we expect that the latency will be further reduced as face detectors become more optimized.

Figure 25(b) shows the latency breakdown on *High* scenes for gradually adding on the components of EagleEye: Variation-Adaptive Face Recognition (A), Spatial Pipelining (P), and Edge-Based Background Filtering (E). Combining each component yields a synergetic gain, achieving 9.07× acceleration compared to the 3-step baseline.

Finally, Figure 25(c) shows the Top-3 accuracy and false alarm increase of EagleEye compared to the 2-step baseline. Overall, EagleEye yields 27.6% accuracy gain, with accuracy above 80% even for *High* scenes. Figure 25(d) shows that at the cost of such accuracy gain, EagleEye results in 19.1% increased false alarm. Such increase is due mainly to the fact that our dataset contains the people with the same ethnicity, and we observe no increase in false alarm in case of WIDER Face dataset.

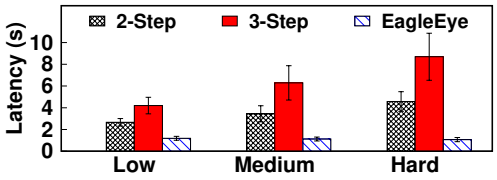


Figure 26: Latency evaluation on Google Pixel 3 XL.

8.6 Performance on Other Mobile Devices

Lastly, we evaluate the end-to-end latency on Google Pixel 3 XL to validate the performance of EagleEye on other mobile devices. The inference times of MobileNetV1-based RetinaFace, ICN, ResNet50-based ArcFace, and MobileFaceNet are 918, 225, 193, 18 ms, respectively. Figure 26 shows that the latency performance of EagleEye and gain compared to 3-step baseline are similar (8.14× for *Hard* scenes) to previous results, indicating that EagleEye shows consistent performance on other devices.

9 Related Work

Face Recognition. Rapid development of CNNs, along with large scale face datasets [6, 25], has enabled significant improvement in face recognition accuracy [14, 55, 67]. However, state-of-the-art methods fail to accurately identify LR faces. EagleEye inserts a novel ICN to the conventional 2-step pipeline (i.e., detection and recognition) to improve LR face recognition accuracy.

Image Super-Resolution. Starting from SRCNN [16], computer vision community has studied various CNN-based approaches for image super-resolution [1, 51]. Several studies have also targeted super-resolving LR faces [4, 12]. However, existing approaches are heavily GAN [21]-driven; they reconstruct real-looking faces, but the identity is often distorted (Figure 11).

Object Detection for High-Resolution Images. Several attempts have been made to optimize latency in detecting objects in high-resolution images by pipelining and parallelizing the processing on different subregions of the image [20, 64]. Similar to these work, EagleEye designs Content-Adaptive Parallel Execution to optimize latency in identifying faces in a high-resolution scene image. Several studies also optimize energy consumption by dynamically adapting frame resolution depending on the content of the scene [32, 57]. These approaches can also be integrated with EagleEye to make the system even more practical.

Continuous Mobile Vision. LiKamWa *et al.* [49] optimize energy of image sensors. Starfish [50] supports concurrency for multiple vision applications. Gabriel [26] uses cloudlets for cognitive assistance. OverLay [36] and MARVEL [8] utilize cloud for location-based mobile AR services. In line with various continuous mobile vision systems, EagleEye provides a novel AR-based service to identify missing person(s) in crowded urban spaces.

Mobile Deep Learning. Several studies have tackled the challenge of on-device deep learning by model compression [45, 71], inference speed acceleration [2, 35, 44, 68], and model size adaptation [54, 70]. However, existing systems mostly focused on running a single DNN on downsampled images (e.g., 300×300) to analyze one or a small number of large, primary object(s) in vicinity.

There have been a few attempts to run multiple DNNs on mobile devices, but they cannot be directly applied for EagleEye. DeepEye [58] parallelizes convolutional layer execution and fully connected layer loading to minimize multi-DNN execution latency. However, running multi-DNN face identification pipeline in EagleEye requires optimization in computation rather than memory footprint. NestDNN [18] adaptively selects DNN from a catalog generated by pruning based on available resources. This approach is unlikely to be effective as our primary goal is to execute the face identification pipeline at low latency without accuracy degradation.

Offloading for Mobile Vision. MCDNN [28] and DeepDecision [62] dynamically execute DNN on cloud or mobile based on available resources. VisualPrint [37] offloads extracted features rather than raw images to save bandwidth. Glimpse [10] tracks objects by offloading only trigger frames for detection and tracking them in the mobile. Liu *et al.* [53] pipeline network transmission and DNN inference to optimize latency. However, existing systems process the input image as a whole, either on mobile or cloud at a given time; such approaches can result in significant latency in case of running complex multi-DNN pipeline. To optimize latency, EagleEye divides the workload both spatially and temporally based on content analysis and parallelizes the execution on mobile and cloud.

10 Discussion and Future Work

Generality. The workload of many future multi-DNN-enabled applications is similar to EagleEye in that they require running a series of complex DNNs repetitively to detect objects in a high-resolution scene image and analyze each identified instance (e.g., text identification, pedestrian identification, etc.). For such applications, our Content-Adaptive Parallel Execution can be generally adapted to enhance performance by applying different pipeline depending on the content and parallelizing the execution over heterogeneous processors on mobile and cloud.

Integration with Other Features. EagleEye can be integrated with other identification methods that utilize various human features (e.g., gait [66] or sound [7]) to enhance accuracy and robustness in more diverse scenarios. Especially, InSight [66] targets similar scenarios with EagleEye, but with different feature (i.e., motion). Furthermore, recent studies on person re-identification [11, 38, 39] (verifying whether two persons captured from two different cameras match based on entire body analysis) have shown significant improvement, which can also be combined with EagleEye.

Privacy. EagleEye raises privacy issues in that it takes pictures of scenes with a number of people present. We would like to note that our goal is to verify whether the target identity exists in the public scene (of which taking picture is not illegal), not to analyze the identities of each individual. Furthermore, our service does not assume storing any captured scene image.

Future Work. While in this work we focus on optimizing performance on a single scene image (as the user can look or move to a completely different area upon completion of a scene analysis), we plan to extend EagleEye on continuous image stream analysis, which can enhance performance on two aspects: i) *Latency*. Utilizing temporal redundancy of continuous frames, we can save redundant computations (e.g., by caching in [35, 68]). ii) *Accuracy*. Analyzing multiple frames can help LR face recognition accuracy (whose difficulty mainly comes from lack of information in the LR face). Furthermore, computer vision community has recently focused on accurately identifying faces under disguise or impersonation [42]. We plan to incorporate such techniques to diversify EagleEye's usage scenarios (e.g., police chasing a criminal). Finally, we plan to scale EagleEye to a full AR service on smart glasses with further considerations for computing resources and power consumption, and evaluate performance in more diverse scenarios with various levels of crowdedness and network condition settings.

11 Conclusion

In this paper, we presented EagleEye, a wearable camera-based system to identify missing person(s) in large, crowded urban spaces in real-time. To further innovate the performance of the state-of-the-art face identification techniques on LR face recognition, we designed a novel ICN and a training methodology that utilize the probes of the target to recover missing facial details in the LR faces for accurate recognition. We also develop Content-Adaptive Parallel Execution to run the complex multi-DNN face identification pipeline at low latency using heterogeneous processors on mobile and cloud. Our results show that ICN significantly enhances LR face recognition accuracy (true positive by 78% with only 14% false positive), and EagleEye accelerates the latency by 9.07× with only 108 KBytes of data offloaded to the cloud.

Acknowledgments

We sincerely thank our anonymous shepherd and reviewers for their valuable comments. This work was supported by the National Research Foundation of Korea (NRF) grant (No. 2019R1C1C1006088). Youngki Lee is the corresponding author of this work.

References

[1] N. Ahn, B. Kang, and K.-A. Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proc. ECCV*, 2018.

[2] S. Bhattacharya and N. D. Lane. Sparsifying deep learning layers for constrained resource inference on wearables. In *Proc. ACM SenSys*, 2016.

[3] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1021–1030, 2017.

[4] A. Bulat and G. Tzimiropoulos. Super-FAN: Integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–117, 2018.

[5] J. Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

[6] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[7] J. Chauhan, Y. Hu, S. Seneviratne, A. Misra, A. Seneviratne, and Y. Lee. BreathPrint: Breathing acoustics-based user authentication. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 278–291. ACM, 2017.

[8] K. Chen, T. Li, H.-S. Kim, D. E. Culler, and R. H. Katz. MARVEL: Enabling mobile augmented reality with low energy and low latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 292–304. ACM, 2018.

[9] S. Chen, Y. Liu, X. Gao, and Z. Han. MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices. In *Chinese Conference on Biometric Recognition*, pages 428–438. Springer, 2018.

[10] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168. ACM, 2015.

[11] W. Chen, X. Chen, J. Zhang, and K. Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2017.

[12] Y. Chen, Y. Tai, X. Liu, C. Shen, and J. Yang. FSRNet: End-to-end learning face super-resolution with facial priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2492–2501, 2018.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[14] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[15] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou. RetinaFace: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*, 2019.

[16] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[17] EyeSight Rapter AR Glass. https://everysight.com/about-raptor/. Accessed: 15 Dec. 2019.

[18] B. Fang, X. Zeng, and M. Zhang. NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 115–127. ACM, 2018.

[19] K. R. Farrell, R. J. Mammone, and K. T. Assaleh. Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on speech and audio processing*, 2(1):194–205, 1994.

[20] M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6926–6935, 2018.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[22] TensorFlow-Lite Object Detection Demo. https://www.tensorflow.org/lite/models/object_detection/overview. 15 Dec. 2019.

[23] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[24] J. Guo and H. Chao. One-to-many network for visually pleasing compression artifacts reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3038–3047, 2017.

[25] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102. Springer, 2016.

[26] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 68–81. ACM, 2014.

[27] J. Han and B. Bhanu. Individual recognition using gait energy image. *IEEE transactions on pattern analysis and machine intelligence*, 28(2):316–322, 2005.

[28] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. MCDNN: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123–136. ACM, 2016.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] L. He, H. Li, Q. Zhang, and Z. Sun. Dynamic feature learning for partial face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7054–7063, 2018.

[31] A. Howard, M. Zhu, K.-D. Chen, B., W. Wang, T. Weyand, M. An-dreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[32] J. Hu, A. Shearer, S. Rajagopalan, and R. LiKamWa. Banner: An image sensor reconfiguration framework for seamless resolution-based tradeoffs. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 236–248. ACM, 2019.

[33] P. Hu and D. Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017.

[34] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[35] L. N. Huynh, Y. Lee, and R. K. Balan. DeepMon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95. ACM, 2017.

[36] P. Jain, J. Manweiler, and R. Roy Choudhury. OverLay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 331–344. ACM, 2015.

[37] P. Jain, J. Manweiler, and R. Roy Choudhury. Low bandwidth offload for mobile ar. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 237–251. ACM, 2016.

[38] J. Jiao, W.-S. Zheng, A. Wu, X. Zhu, and S. Gong. Deep low-resolution person re-identification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] X.-Y. Jing, X. Zhu, F. Wu, X. You, Q. Liu, D. Yue, R. Hu, and B. Xu. Super-resolution person re-identification with semi-coupled low-rank discriminant dictionary learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 695–704, 2015.

[40] M. Kampf, I. Nachson, and H. Babkoff. A serial test of the laterality of familiar face recognition. *Brain and cognition*, 50(1):35–50, 2002.

[41] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[42] V. Kushwaha, M. Singh, R. Singh, M. Vatsa, N. Ratha, and R. Chellappa. Disguised faces in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2018.

[43] S. K. Lam, A. Pitrou, and S. Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, page 7. ACM, 2015.

[44] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar. DeepX: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, page 23. IEEE Press, 2016.

[45] N. D. Lane, P. Georgiev, and L. Qendro. DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 283–294. ACM, 2015.

[46] M. B. Lewis and A. J. Edmonds. Face detection: Mapping human performance. *Perception*, 32(8):903–920, 2003.

[47] J. Lezama, Q. Qiu, and G. Sapiro. Not afraid of the dark: NIR-VIS face recognition via cross-spectral hallucination and low-rank embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2017.

[48] P. Li, L. Prieto, D. Mery, and P. J. Flynn. On low-resolution face recognition in the wild: Comparisons and new techniques. *IEEE Transactions on Information Forensics and Security*, 14(8):2000–2012, 2019.

[49] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 69–82. ACM, 2013.

[50] R. LiKamWa and L. Zhong. Starfish: Efficient concurrency support for computer vision applications. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 213–226. ACM, 2015.

[51] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on*

Computer Vision and Pattern Recognition Workshops, pages 136–144, 2017.

[52] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[53] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 2019.

[54] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 389–400. ACM, 2018.

[55] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. SphereFace: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[56] G. Lu, W. Ouyang, D. Xu, X. Zhang, Z. Gao, and M.-T. Sun. Deep kalman filtering network for video compression artifact reduction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 568–584, 2018.

[57] E. S. Lubana and R. P. Dick. Digital foveation: An energy-aware machine vision framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2371–2380, 2018.

[58] A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, and F. Kawsar. DeepEye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 68–81. ACM, 2017.

[59] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis. SSH: Single stage headless face detector. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4875–4884, 2017.

[60] L. B. Neto, F. Grijalva, V. R. M. L. Maike, L. C. Martini, D. Florencio, M. C. C. Baranauskas, A. Rocha, and S. Goldenstein. A kinect-based wearable face recognition system to aid visually impaired users. *IEEE Transactions on Human-Machine Systems*, 47(1):52–64, 2016.

[61] S. Panchanathan, S. Chakraborty, and T. McDaniel. Social interaction assistant: a person-centered approach to enrich social interactions for individuals with visual impairments. *IEEE Journal of Selected Topics in Signal Processing*, 10(5):942–951, 2016.

[62] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. DeepDecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1421–1429. IEEE, 2018.

[63] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *IEEE CVPR*, 2017.

[64] V. Ruzicka and F. Franchetti. Fast and accurate object detection in high resolution 4k and 8k video using gpus. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2018.

[65] X. Tang, D. K. Du, Z. He, and J. Liu. Pyramidbox: A context-assisted single shot face detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 797–813, 2018.

[66] H. Wang, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Visually fingerprinting humans without face recognition. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 345–358. ACM, 2015.

[67] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[68] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. DeepCache: Principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 129–144. ACM, 2018.

[69] S. Yang, P. Luo, C.-C. Loy, and X. Tang. WIDER FACE: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[70] S. Yao, Y. Zhao, H. Shao, S. Liu, D. Liu, L. Su, and T. Abdelzaher. FastDeepIoT: Towards understanding and optimizing neural network execution time on mobile and embedded devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 278–291. ACM, 2018.

[71] X. Zeng, K. Cao, and M. Zhang. MobileDeepPill: A small-footprint mobile deep learning system for recognizing unconstrained pill images. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 56–67. ACM, 2017.

[72] R. Zhang. Making convolutional networks shift-invariant again. *International Conference on Machine Learning (ICML)*, 2019.

[73] Y. Zhao, S. Wu, L. Reynolds, and S. Azenkot. A face recognition application for people with visual impairments: Understanding use beyond the lab. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 215. ACM, 2018.