Edge-assisted Collaborative Image Recognition for Mobile Augmented Reality

GUOHAO LAN, Duke University, USA
ZIDA LIU, Pennsylvania State University, USA
YUNFAN ZHANG and TIM SCARGILL, Duke University, USA
JOVAN STOJKOVIC, University of Illinois at Urbana-Champaign, USA
CARLEE JOE-WONG, Carnegie Mellon University, USA
MARIA GORLATOVA, Duke University, USA

Mobile Augmented Reality (AR), which overlays digital content on the real-world scenes surrounding a user, is bringing immersive interactive experiences where the real and virtual worlds are tightly coupled. To enable seamless and precise AR experiences, an image recognition system that can accurately recognize the object in the camera view with low system latency is required. However, due to the pervasiveness and severity of image distortions, an effective and robust image recognition solution for "in the wild" mobile AR is still elusive. In this article, we present CollabAR, an edge-assisted system that provides *distortion-tolerant image recognition* for mobile AR with *imperceptible system latency*. CollabAR incorporates both *distortion-tolerant* and *collaborative* image recognition modules in its design. The former enables distortion-adaptive image recognition to improve the robustness against image distortions, while the latter exploits the spatial-temporal correlation among mobile AR users to improve recognition accuracy. Moreover, as it is difficult to collect a large-scale image distortion dataset, we propose a Cycle-Consistent Generative Adversarial Network-based data augmentation method to synthesize realistic image distortion. Our evaluation demonstrates that CollabAR achieves over 85% recognition accuracy for "in the wild" images with severe distortions, while reducing the end-to-end system latency to as low as 18.2 ms.

CCS Concepts: • **Computing methodologies → Distributed algorithms**; **Mixed/augmented reality;**

Additional Key Words and Phrases: Edge computing, collaborative augmented reality, mobile image recognition, cycle-consistent generative adversarial networks

An early version of this work appeared at ACM/IEEE IPSN 2020 [1].
This work was supported in part by the Lord Foundation of North Carolina and by NSF awards CSR-1903136 and CNS-1908051.
Authors' addresses: G. Lan, Y. Zhang, T. Scargill, and M. Gorlatova, Duke University, Durham, NC, 27705; emails: {guohao.lan, yunfan.zhang, timothyjames.scargill, maria.gorlatova}@duke.edu; Z. Liu, Pennsylvania State University, University Park, PA, 16802; email: zjl5310@psu.edu; J. Stojkovic, University of Illinois at Urbana-Champaign, Champaign, IL, 61820; email: jovans2@illinois.edu; C. Joe-Wong, Carnegie Mellon University, Mountain View, CA, 94035; email: cjoewong@andrew.cmu.edu.

1550-4859/2021/09-ART9 $15.00
https://doi.org/10.1145/3469033

ACM Reference format:
Guohao Lan, Zida Liu, Yunfan Zhang, Tim Scargill, Jovan Stojkovic, Carlee Joe-Wong, and Maria Gorlatova. 2021. Edge-assisted Collaborative Image Recognition for Mobile Augmented Reality. *ACM Trans. Sen. Netw.* 18, 1, Article 9 (September 2021), 31 pages.
https://doi.org/10.1145/3469033

1 INTRODUCTION

Mobile **augmented reality (AR)**, which overlays digital content with the real world around a user, has recently jumped from the pages of science fiction novels into the hands of consumers. To enable a seamless contextual AR experience, an effective image recognition system that can *accurately recognize objects* in the camera view of the mobile device with *imperceptible latency* is required [2, 3]. While this may come across as a solved problem given the recent advancements in **deep neural networks (DNNs)** [4, 5] and mobile offloading [2, 6, 7], there are three practical aspects that have been largely overlooked in existing solutions.

First, in real-world mobile AR scenarios, a large proportion of images taken by the smartphone or the head-mounted AR device contains distortions. For instance, images frequently contain motion blur caused by the motion of the user [2, 8]. Gaussian blur appears when the camera is defocusing or is used in an underwater AR scenario [9]. Noise is also inevitable in low illumination conditions [10] or with the use of low-quality image sensors [11]. Indeed, using two different commodity mobile AR devices, our measurement study (Section 3.3) indicates that *over 70% of the images can be corrupted by distortions* in different practical scenarios. Given the high distortion proportion, simply filtering out the distorted images, as has been suggested in prior work [2, 3, 12], can hardly solve the problem.

Second, for image recognition, applying DNNs trained on large-scale datasets, e.g., ImageNet [13] and Caltech-256 [14], directly to images that contain severe image distortion is difficult and often results in dramatic performance degradation [10, 15, 16]. This deficiency results from the *domain shift problem*, where distorted images fail to share the same feature distribution with the clear training images [15, 17, 18]. Indeed, when testing distorted images with a pre-trained MobileNetV2 [5] network on the Caltech-256 dataset, we observe that even a small amount of distortion in the images can significantly affect the recognition accuracy (Section 3.2). Although deblurring methods and spatial filters [19] can be used to reduce distortions, they also remove the fine-scaled image details that are useful for image recognition. There have also been attempts to overcome this limitation by designing distortion-robust DNNs [10, 15, 16, 18], but their improvements are limited when the images contain multiple distortions or when the distortion level is high.

Lastly, to achieve imperceptible recognition latency for resource-constrained mobile AR devices, a number of works have explored computation offloading [2, 6, 20, 21] and approximate computation reuse [7, 22] to reduce the system overhead. Although the state-of-the-art solutions have reduced the end-to-end system latency below 33ms [6, 20], such performance is achieved in distortion-free or distortion-modest environments. The challenge in mitigating the trade-off between recognition accuracy and system latency for mobile AR with severe multiple distortions has not been addressed.

To fill this gap, we present CollabAR, an edge-assisted Collaborative image recognition framework for mobile AR. Thanks to the recent advances in AR development platforms, AR applications have become more accessible on a wide range of mobile devices. We have seen many dedicated AR devices, such as Microsoft HoloLens [23] and Magic Leap One [24], as well as Google ARCore [25] and Apple ARKit [26] SDKs that work on a wide range of mobile phones and tablets. The pervasive deployment of mobile AR will offer numerous opportunities for multi-user collaboration [12, 27]. In fact, we have seen significant research and commercial efforts in multi-user AR platforms and applications [28–34]. Examples include the "shared AR experience" feature in Pokemon Go [31] that allows users to share and view the same virtual character in the same space and the "local lenses" feature in Snapchat [32] that enables large-scale collaborative AR at landmark, iconic locations such as London's Carnaby Street to provide shared AR experience. Examples of industrial adopters of shared AR are Ford and Lockheed Martin, whose designers and engineers use the Microsoft HoloLens to speed up collaboration and prototyping [33, 34]. In addition to the pervasiveness of mobile AR, users that are in close proximity usually exhibit a strong correlation in both device usage and spatial-temporal context [35]. The requested image recognition tasks are usually correlated temporally (e.g., consecutive image frames from the same user [6, 7, 36]) or spatially (e.g., different users aim to recognize the same object [12, 22]). Although the images are taken from different positions or at different times, they contain the same object [22]. Thus, unlike conventional image recognition systems where individual users independently complete their tasks to recognize the same object [2, 6, 20], CollabAR embraces the concept of *collaborative image recognition* in its design and exploits the *temporally and spatially correlated* images captured by the users to improve the image recognition accuracy.

Bringing this high-level concept into a holistic system requires overcoming several challenges. First, we need to address the domain adaptation problem [17] caused by the image distortions. As DNNs can adapt to a particular distortion type but not multiple types at the same time [10, 17], we need to adaptively select a DNN that has been fine-tuned to recognize a specific type of distorted images in runtime. We introduce the *distortion-tolerant image recognizer*, which incorporates the *image distortion classifier* and *a set of dedicated recognition experts*. CollabAR employs the image distortion classifier to identify the most significant distortion contained in the image and triggers one of the dedicated recognition experts to handle the distorted image. This distortion-adaptive selection of a recognizer ensures a lower feature domain mismatch between the image and the selected DNN and a better robustness against distortions.

Second, to enable collaborative image recognition, we need to identify a set of spatially and temporally correlated images that contain the same object. Prior works rely on image feature vectors [22] or feature maps [7] to identify temporally correlated images (e.g., continuous frames in a video). However, image features are vulnerable to distortions and result in high lookup errors in practical AR scenarios. Moreover, existing methods assume spatial differences between images to be small, e.g., adjacent image frames in a video. However, in mobile AR scenarios, correlated images can be taken from different angles and positions with large spatial differences and, thus, result in feature mismatch and lookup error. CollabAR introduces the *anchor-based pose estimation* and the *spatial-temporal image lookup module* to efficiently identify correlated images.

Third, as the correlated images suffer from various distortions with different severity levels, they lead to heterogeneity in the recognition confidence and accuracy. Conventional multi-view image recognition systems [37, 38] lack a reliable metric to differentiate the heterogeneity and are not able to optimally aggregate the multi-view results [39]. In this work, we propose the **Auxiliary-assisted Multi-view Ensemble Learning (AMEL)** to dynamically aggregate the heterogeneous recognition results of the correlated images.

Fourth, to improve robustness against image distortion, a large volume of distorted images is required during the training to fine-tune the DNNs. Unfortunately, in practice, it is difficult to collect a large number of images with different types of distortions. In our previous work [1], we apply mathematical models to synthesize images with different distortions. However, the gap between the mathematically synthesized distortions and the real-world distortions [40, 41] results in drops in recognition accuracy when the system is applied to "in the wild" AR scenarios. To alleviate this problem, we formulate the task of image distortion synthesis as an *"unpaired image-to-image translation"* problem and propose the use of **Cycle-Consistent Generative Adversarial Networks (CycleGANs)** [42] to generate realistic image distortions.

Lastly, having all the system building blocks in mind, we aim to provide imperceptible system latency (i.e., below 33ms to ensure 30fps continuous recognition), without sacrificing accuracy. The targeted low system latency can leave more time and space for many other computation-intensive tasks (e.g., virtual object rendering) that are running on mobile AR devices. We explore several design options (i.e., selection of DNN models, what to offload and what to process locally) and conduct a comprehensive system profiling to guide the final optimized implementation. We propose an edge-assisted system framework to mitigate the trade-off between recognition accuracy and system latency.

In this article, we present CollabAR, a collaborative image recognition system for mobile AR. Our main contributions are:

- We propose the *distortion-tolerant image recognizer* to resolve the domain adaptation caused by image distortions. Specifically, we design the image distortion classifier to enable distortion-adaptive DNN selection and design a set of dedicated recognition experts to achieve distortion-tolerant image recognition for "in the wild" mobile AR. Compared to conventional methods, the proposed solution improves the recognition accuracy by 17.6% over different image distortions and distortion levels.
- To further boost the recognition accuracy, CollabAR embraces collaboration opportunities among mobile AR users and exploits the spatial-temporal correlation among images for multi-view image recognition. CollabAR introduces the *anchor-based image lookup module* to effectively identify the spatially and temporally correlated images, and the *auxiliary-assisted multi-view ensemble learning framework* to dynamically aggregate the heterogeneous multi-view results. Compared to the single-view-based recognition, the proposed method can improve the recognition accuracy by 9.3% to 19.4% in severe multiple distortions scenario.
- We implement CollabAR on four different commodity devices and evaluate its performance on two multi-view image datasets, one public and one collected by ourselves. The evaluation demonstrates that CollabAR achieves over 85% recognition accuracy for "in the wild" images that contain severe multiple distortions, while reducing the end-to-end system latency to as low as 18.2ms for commodity mobile devices.

The rest of the article is organized as follows. Section 2 reviews the related work. Section 3 introduces the background and challenges. Section 4 presents the system overview. Section 5 introduces the real-world multi-view multi-distortion dataset we collected. Section 6 describes our CycleGAN-based image distortion augmentation. The design details of the distortion-tolerant recognizer and the collaborative multi-view image recognition framework are given in Sections 7 and 8, respectively. We present the evaluation in Section 9 and conclude the article in Section 10.

The research artifacts, including the multi-view multiple distortions image dataset we collected (Section 5) and the source codes for both the distortion-tolerant image recognizer and the auxiliary-assisted multi-view ensemble learning framework, are available at https://github.com/CollabAR-Source/.

2 RELATED WORK

As a holistic image recognition system for multi-user mobile AR, CollabAR builds on a body of prior work in mobile AR, distortion-tolerant image recognition, distributed image recognition, and GAN-based image augmentation and restoration.

2.1 Image Recognition for Mobile AR

Before overlaying the rendered virtual objects on the view of a user, mobile AR systems leverage image recognition to identify the object appearing in the view. For this purpose, image retrieval [2, 20], image localization [3], and DNN-based image recognition methods [6] are widely used in prior work. Image-retrieval-based solutions, such as OverLay [2] and Jaguar [20], exploit salient feature descriptors to match the image in the user's view with the annotated image stored in the database. However, salient descriptors are vulnerable to image distortions; a large proportion of images cannot be correctly recognized but are simply filtered out [2]. Image-localization-based methods require the server to maintain the annotated image dataset of the service area [43], which is computationally heavy and involves a large volume of data [3]. Moreover, image localization also relies on salient image features and fails when the images contain any type of distortion. Lastly, DNN-based solutions [6] perform well with pristine images, but even a small amount of distortion can lead to a dramatic performance drop [10, 15]. Instead of filtering out the distorted images, CollabAR incorporates the distortion-tolerant image recognizer to enable image recognition for mobile AR systems.

2.2 Distortion-tolerant Image Recognition

Recent efforts have been made by the computer vision community in resolving the negative impacts of image distortion on DNNs [10, 15, 16, 18]. A widely used approach to improve the resilience of DNNs on distorted images is to fine-tune the networks with distorted images. Dodge and Karam [10] propose a mixture-of-experts-based model, in which recognition experts are fine-tuned to handle a specific image distortion, and their outputs are aggregated using a gating network. In [15], Ghosh et al. propose a similar master-slave **convolutional neural network (CNN)** architecture for distorted image recognition. In their design, a quality prediction network is used as the master to select the slave CNNs that are fine-tuned with specific image distortions. Another trend is to correct and attenuate the impacts of image distortions on the feature distribution of the DNNs. Borkar and Karam [16] propose an objective metric to identify the most distortion-susceptible convolutional filters in the CNNs. Correction units are added in the network to correct the outputs of the identified filters. Sun et al. [41] propose a feature quantization approach to enhance the robustness of CNNs against distortions. They integrate non-linearity into the convolution operation of the CNNs to attenuate the feature distribution shift due to image distortion. However, prior works mainly focus on the single-view scenario and fail when the image contains multiple distortions or the distortion level is high [10, 15, 16]. In contrast, CollabAR can dynamically aggregate the spatially and temporally correlated images to improve recognition accuracy in the multiple distortions scenario.

2.3 Distributed Image Recognition

Using distributed information collected from multi-camera setups for image and object recognition is a widely studied topic. For instance, in DDNNs [38], distributed deep neural networks are deployed over cloud, edge, and end devices for joint object recognition. By leveraging the geographical diversity of a multi-camera network, DDNNs improve the object recognition accuracy. To improve the accuracy for distortion-tolerant pill image recognition, MobileDeepPill [44] relies on a multi-CNN model to collectively capture the shape, color, and imprint characteristics of pills. Kestrel [45] incorporates a video analytics system that detects and tracks vehicles across heterogeneous multi-camera networks. These existing systems only perform in distortion-free [38] or distortion-modest environments [44]. They are not able to recognize the image if it contains severe or multiple distortions. Moreover, many of them suffer from high end-to-end latency (i.e., 200ms) [45]. In contrast, CollabAR exploits a set of distortion-tolerant recognition experts and spatial-temporal correlations among multi-view images to boost the image recognition performance for "in the wild" mobile AR images with distortion. It also leverages the edge-assisted framework to mitigate the trade-off between recognition accuracy and system latency.

2.4 GAN-based Image Augmentation and Restoration

Generative Adversarial Networks (GANs) [46] and CycleGANs [42] have been widely used for image augmentation. For example, Zhu et al. [47] propose the use of CycleGAN-based data augmentation for image-based emotion classification. They design a CycleGAN-based framework to generate auxiliary data for minority classes in the training dataset. Similarly, Sandfort et al. [48] propose the use of CycleGAN for data augmentation in **computed tomography (CT)** segmentation. The CycleGAN is used to transform contrast CT images into non-contrast images and generates synthetic non-contrast images for DNN training. GAN and CycleGAN have also been used for image restoration. For instance, Engin et al. [49] introduce the Cycle-Dehaze model to recover image from Gaussian blur. Chen et al. [40] leverage GAN to synthesize noisy images, which are then used as the training samples for a CNN-based denoising network. Bulat et al. [50] propose the use of GAN to effectively increase the quality of real-world low-resolution images. By contrast, in this work, we propose the use of CycleGAN to generate three types of distorted images that frequently appear in real-world mobile AR scenarios. The synthesized distorted images are used to improve the recognition robustness of the distortion-tolerant image recognizer.

3 BACKGROUND AND CHALLENGES

We consider three types of image distortion: motion blur, Gaussian blur, and Noise. Below, we first introduce the basics of the image distortions and then provide a measurement study to understand their impacts on image recognition and their pervasiveness in real-world mobile AR scenarios.

3.1 Image Distortions

Motion blur (MBL): Images taken by the smartphone or the head-mounted AR set camera frequently contain motion blur caused by the motion of the user [2]. The severity of motion blur is affected by the *changing speed of the image content* and *the exposure time of the camera* [51]. When the camera is steady, the exposure time is short such that the camera only captures an instantaneous moment of the scene and, thus, forms a sharp image. However, when the mobile device is shaking during the exposure, it causes scene movements and leads to the integration of instantaneous signals from different scenes and, thus, results in motion blur. We can approximate this process by accumulating signals from successive frames [51] and obtain the motion blur image F_{MBL} by the following accumulation process [51]:

$$F_{MBL} = g\left(\frac{1}{T}\int_{t=0}^{T} S(t)dt\right) \simeq g\left(\frac{1}{M}\sum_{i=0}^{M-1} S[i]\right), \tag{1}$$

where T is the exposure time, $S(t)$ is the camera sensor signal at time t, M is the number of frames captured during the exposure time, and $S[i]$ is the ith frame. Lastly, g is the **Camera Response Function (CRF)** that maps the latent signal $S[i]$ into an observed image $\hat{S}[i]$, such that $\hat{S}[i] = g(S[i])$. Generally, the ground-truth CRF is not given, but a common method is to approximate CRF as a gamma curve $g = x^{1/\gamma}$ with $\gamma = 2.2$ [52]. The latent signal $S[i]$ can be obtained from the sharp frame $\hat{S}[i]$ by $S[i] = g^{-1}(\hat{S}[i])$.

Gaussian blur (GBL): Gaussian blur appears when the camera is de-focusing or the image is taken underwater (e.g., in underwater AR [9]) or in a foggy environment [19]. Specifically, during image generation, the camera lens converges all the light rays coming from a point of the object at the focus distance to a point in the image plane. When the light rays converge in front of or behind the image plane, point appears as a blurred spot in the image. The blur level of the image is determined by the diameter of the blurred spot, ρ, which can be defined as [53]

$$\rho = \frac{\mid L - D \mid}{L} \times \frac{f^3}{A(D - f)}, \tag{2}$$

where L is the distance between the object and the camera lens, D is the focus distance, f is the focal length, and A is the aperture size. The larger the value of ρ, the higher the blur level.

Noise (N): Noise is also inevitable in images, which causes random variation in brightness or color information in the pristine images. Possible sources of noise include poor illumination conditions [10], digital zooming, and the use of a low-quality image sensor [11]. Although real-world noise is signal dependent and can hardly be modeled by an explicit math equation, it is widely known that for a given camera sensor, both *high ISO value* [54, 55] and *low light condition* [54] will generate noise in the captured image.

3.2 Impact of Distortion on Image Recognition

Deep neural networks, such as the VGG16 [56] and the MobileNetV2 [5], have shown good performance in image recognition. However, as DNNs are usually trained and tested on images that are pristine, e.g., ImageNet [13], directly applying them to images that contain severe multiple distortions often leads to dramatic performance degradation [16, 19, 57]. This deficiency results from the domain shift problem where real-world distorted images fail to share the same feature distribution with the pristine training images [15, 17, 18].

As a case study to quantify this impact, we investigate the image recognition accuracy of MobileNetV2 and VGG16 on distorted images. Leveraging the dataset we collected (details are given in Section 5), we first pre-train the two networks using the pristine images and then test their performance on the distorted images. Figure 1 shows the examples of images containing different types of distortion at different severity levels. As shown, for each type of distortion, we consider four different distortion levels (details about the distortion level are given in Section 5). The impacts of image distortions on the recognition accuracy of DNNs are shown in Figure 2. We can observe that *even a small amount of distortion in the images could significantly affect the recognition accuracy*. For instance, when the images are containing a low-level motion blur (level 1), we can see a significant accuracy drop of 49.1% and 37.9% on MobileNetV2 and VGG16, respectively, even though the distortion at this level does not hinder the human eyes' ability to recognize the object [58] (e.g., see an example in Figure 1(a)).

3.3 Image Distortion for Real-world Mobile AR

Below, we quantify the extent of image distortion in real-word mobile AR scenarios. We use two commodity AR devices, the Nokia 7.1 smartphone and the Magic Leap One head-mounted AR set, to record videos (at 30fps) in different environments and measure the proportion of recorded frames that suffer from severe image distortions.

First, to quantify *Noise*, we use the Nokia 7.1 to record videos in environments with poor illumination conditions, i.e., indoor dark room (with 7lux). In addition, to study noise resulting from digital zooming, we conduct an experiment in an indoor environment at daytime (with 509lux) and turn on the built-in digital zoom function of the Nokia 7.1 to record videos of objects that are 3 meters away. We cannot record video using Magic Leap One as its image and video recording service is unavailable when the light condition is low, and it supports neither digital nor optical image zooming. Second, to quantify *motion blur*, we ask one user to hold the Nokia 7.1 in his or her hand

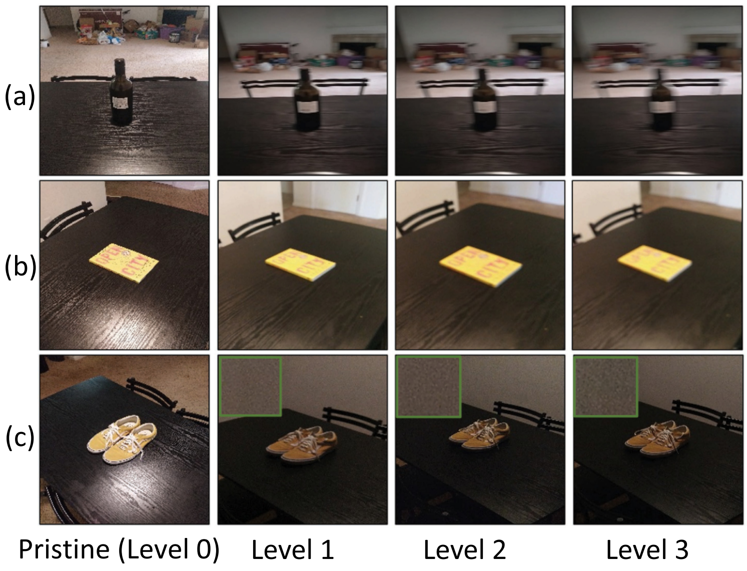


Fig. 1. Examples of images containing different types of distortion with different severity levels: (a) motion blur, (b) Gaussian blur, and (c) noise. The distortion severity increases from left to right. Although distortions do not hinder the human in recognizing the object, they severely affect the performance of DNNs.

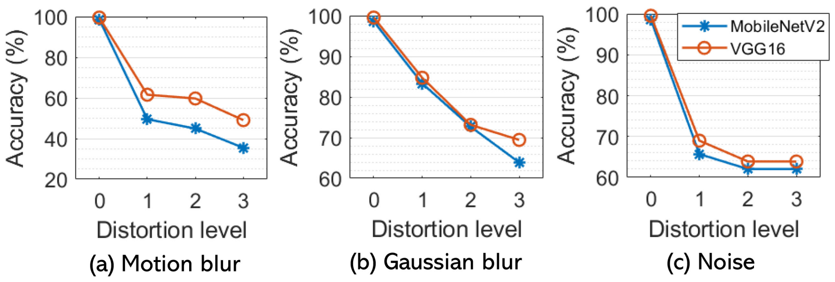


Fig. 2. Impacts of image distortion on the recognition performance of MobileNetV2 and VGG16. For all the three distortion types, the recognition accuracy drops dramatically with the distortion level.

and ask another user to wear the Magic Leap One. They record videos using the device front camera in both indoor corridor (with 178lux) and sunny outdoor (with 9873lux) environments when walking at a normal pace. Lastly, to quantify *Gaussian blur*, we put the Nokia 7.1 in a waterproof case and record videos in both underwater (a bathtub filled with water) and foggy environments (a foggy bathroom).[1] The collected images make up the MobileDistortion dataset, which we made publicly available at https://github.com/CollabAR-Source/Distortion.

To determine if an image suffers from severe distortion, we use three standard distortion detectors to measure the distortion level. Specifically, we apply the image spatial quality estimator [59], the partial-blur image detector [60], and the **Cumulative Probability of Blur Detection (CPBD)** metric [61] to detect noise, motion blur, and Gaussian blur, respectively. The results are shown in

[1]We do not conduct this experiment with a Magic Leap One as we are unaware of the availability of waterproof cases for it.

Table 1. Measurements of Image Distortions in Real-world Mobile AR Scenarios

Distortion	**Setting**	**Hardware**	
		Nokia 7.1	Magic Leap One
Noise	Dark room (7lux)	429/889 = 48.3%	Infeasible
	Camera zoom-in (509lux)	1,378/1,582 = 87.10%	Infeasible
Motion blur	Corridor (178lux)	1,671/3,452 = 48.40%	3,641/3,776 = 96.4%
	Sunny outdoor (9,873lux)	1/2,687 = 0.03%	335/2,766 = 12.11%
Gaussian blur	Foggy	430/935 = 45.9%	Infeasible
	Underwater	809/1,524 = 53.0%	Infeasible

A large proportion of images suffer from severe distortions; simply filtering out the distorted images can hardly solve the problem.

Table 1. The illumination of the environment is measured using the smartphone light sensor that is adjacent to the smartphone camera.

First, we apply the image spatial quality estimator [59] to quantify image noise and leverage the **Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE)** [59] as the quality metric. An image is considered to have severe noise if the BRISQUE is larger than 30 (which corresponds to the level 1 noise shown in Figure 1). In poor-illumination conditions, over 48.3% of the images recorded by the Nokia 7.1 suffer from severe noise. Moreover, when using digital zooming, 87.1% of the images suffer from severe noise due to the additive noise introduced in digital image processing.

Second, we apply the partial-blur image detector [60] to quantify motion blur. It uses the scaled mean magnitudes of the image's Fast Fourier Transforms as the blurry value for motion blur measurement. An image is considered to have severe motion blur when its blurry value is smaller than 1.45 (corresponding to the level 1 motion blur in Figure 1). As shown in Table 1, for Magic Leap One, 96.4% and 12.11% of the recorded images suffer from severe motion blur in the corridor and the outdoor environments, respectively. For Nokia 7.1, 48.40% and 0.03% of the images contain severe motion blur in these two environments. As expected, we observe that there is less motion blurring in the sunny outdoor environment (9,873lux) than in the corridor (178lux): as the camera exposure time is shorter with a higher illumination and the impact of motion on the image is proportional to the exposure time, there is less motion blurring in high-illumination environments.

Lastly, we apply the CPBD metric [61] to measure Gaussian blur. An image is considered to have severe Gaussian blur if the CPBD value is smaller than 0.48 (corresponding to level 1 Gaussian blur in Figure 1). The results show that 45.9% and 53.0% of the images suffer from severe Gaussian blur in the underwater and foggy environments, respectively.

Overall, our measurements highlight the pervasiveness and severity of image distortions in real-world scenarios. Given the high distortion proportion, simply filtering out the distorted images, as has been suggested in prior work [2, 3, 12], can hardly solve the distortion problem for practical AR applications.

4 SYSTEM OVERVIEW

The architecture of CollabAR is shown in Figure 3, which incorporates three components: (1) the distortion-tolerant image recognizer, (2) the correlated image lookup module, and (3) the auxiliary-assisted multi-view ensembler. They are deployed on the edge server to mitigate the trade-off between recognition accuracy and end-to-end system latency (detailed system measurements are given in Section 9.4). In addition to the server, the client is running an anchor-based pose estimation module, which leverages the cloud anchors provided by the Google ARCore [25] to track the location and orientation of the mobile device.

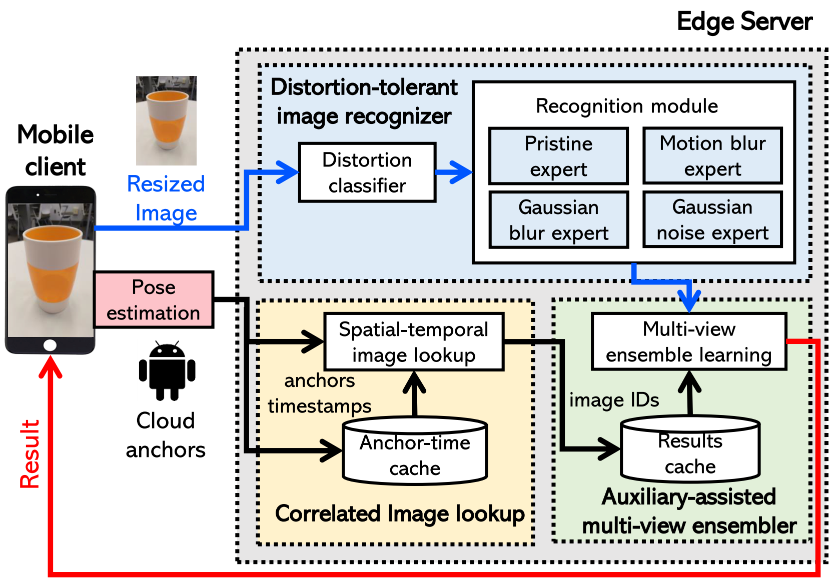


Fig. 3. System architecture of CollabAR. The client takes an image of the object and sends it to the edge along with the cloud anchors in the camera view. The image is used as the input for image recognition and the cloud anchors are used for correlated image lookup.

When the user takes an image of the object, the mobile client sends the image along with the IDs of the anchors in the camera view to the server. Having received the data, the server marks the image with a unique image ID and a timestamp. The image is used as the input for the distortion-tolerant image recognizer, while the anchor IDs and the timestamp are used as references for spatial-temporal image lookup. In CollabAR, anchor IDs and timestamps are stored in the *anchor-time cache* in the format of $<$ imageID, {anchorIDs}, timestamp $>$, while the recognition results of the images are stored in the *results cache* in the format of $<$ imageID, inferenceResult $>$ for future reuse and aggregation. In our design, we do not share information among the clients. Instead, clients communicate directly with the edge server to ensure high recognition accuracy and low end-to-end latency. Note that CollabAR can be readily adapted to video-input-based design, as videos can be treated as streams of image frames in this scenario. In fact, based on the end-to-end system latency measured in Section 9.4, CollabAR can support 50fps continuous recognition.

Distortion-tolerant image recognizer: The distortion image recognizer incorporates an image distortion classifier (Section 7.1) and four recognition experts (Section 7.2) to resolve the domain adaptation problem [17] caused by the image distortions. As DNNs can adapt to a particular distortion but not multiple distortions at the same time [10, 17], we need to identify the most significant distortion in the image and adaptively select a DNN that is dedicated to the detected distortion. As shown in Figure 3, for a new image received from the client, the distortion classifier detects if the image is pristine or if it contains any type of distortion (i.e., motion blur, defocus blur, or noise). Then, based on the detected distortion type, one of the four recognition experts is triggered for the recognition. Each of the recognition experts is a CNN that has been fine-tuned to recognize a specific type of distorted images. The inference result of the expert is sent to the auxiliary-assisted multi-view ensembler for aggregation and is stored in the results cache for reuse.

Correlated image lookup: To enable collaborative image recognition, the correlated image lookup module (Section 8.1) searches the Anchor-time cache to find previous images that are spatially and temporally correlated with the new one. The anchor IDs are used as the references to identify the spatial correlation, while the timestamps are used to determine the temporal correlation. The correlated images could be the images that are captured by different users or the continuous image frames that are obtained by a single device. The IDs of the correlated images are forwarded to the auxiliary-assisted multi-view ensembler.

Auxiliary-assisted multi-view ensembler: Based on the identified image IDs, the multi-view ensembler retrieves the inference results of the correlated images from the results cache. Together with the recognition result of the new image, the retrieved results are aggregated by the multi-view ensemble learning module to improve the recognition accuracy. Specifically, as the correlated images suffer from various distortions with different severity levels, they are unequal in quality and lead to heterogeneity in the recognition accuracy. CollabAR leverages **Auxiliary-assisted Multi-view Ensemble Learning (AMEL)** (Section 8.2) to dynamically aggregate the heterogeneous results provided by the correlated images.

As an image-recognition-based solution, CollabAR does not support multi-object recognition in its current design. Instead, we follow the observations that in many commercial mobile AR applications (e.g., the BBC Civilisations AR [62] and the QuiverVision [63]) the visual scene only contains one major object of interest. In these scenarios, CollabAR can be used to recognize the object of interest directly and link it to the associated holographic content without the need for fiducial markers.

5 REAL-WORLD MULTI-VIEW MULTI-DISTORTION IMAGE DATASET

In this section, we introduce the **Real-world Multi-View Multi-Distortion image Dataset (RMVMDD)** we have collected to study the impact of “in the wild” image distortion on multi-view AR.

Our dataset includes a pristine image set, three single-distortion image sets, and four multiple-distortion sets. For each of the image sets, six categories of everyday objects are considered: *cup*, *shoe*, *bottle*, *book*, *box*, and *pen*. Each category has *six instances*. For each instance, images are taken from *six different views* (six different angles with a 60° angle difference between any two adjacent views). The details are summarized in Table 2. All the images are collected by a commodity Nokia 7.1 smartphone. The resolution of the images is 3024×4032. The dataset is publicly available at https://github.com/CollabAR-Source/RMVMDD.

Pristine image set: We consider three object sizes for the pristine image set. Specifically, we adjust the distance between the camera and the object such that the object occupies approximately the whole, half, and one-ninth of the total area of the image. In total, we collect $6 \times 6 \times 6 \times 3 = 648$ pristine images (examples are shown in Figure 4).

Motion blur image set: The severity of motion blur is decided by the changing speed of the image content (relative speed between the camera and the object) and the exposure time of the camera. As the object is stationary, we create a constant horizontal movement and fix the relative speed by placing the smartphone on a rotation turntable. The turntable is rotating constantly at an angular speed of $7.2°/s$ (rotating a full circle every 50 seconds). To generate different levels of motion blur, for each object category and instance, we use the smartphone to record a short video at each of the six views at 120fps. Then, we synthesize three different motion blur levels by adjusting the exposure time of the camera. Specifically, following Equation (1), we approximate three exposure times of 0.1s, 0.125s, and 0.2s by accumulating 12, 16, and 24 successive image frames in the 120fps video. We fix the distance between the rotating camera and the stationary object at 60cm to make the object occupy approximately one-ninth of the total area of the image. We collect $6 \times 6 \times 6 \times 3 = 648$ Motion blur images in total.

Table 2. Summary of the RMVMDD Dataset

Pristine image set	Object categories	6
	Number of views	6
	Size of object in image	3
	Number of instances	6
Total pristine images	$6 \times 6 \times 3 \times 6 = 648$	
Single distortion image set	Types of distortion	3
	Distortion levels	3
Total single-distortion images	$6 \times 6 \times 6 \times 3 \times 3 = 1{,}944$	
Multiple distortion image set	Types of combination	4
	Distortion levels	3
Total multi-distortion images	$6 \times 6 \times 6 \times 4 \times 3 = 2{,}592$	
Total images	5,184	

It has 5,184 images in total, including 648 pristine images, 1,944 single-distortion images, and 2,592 multiple-distortion images.

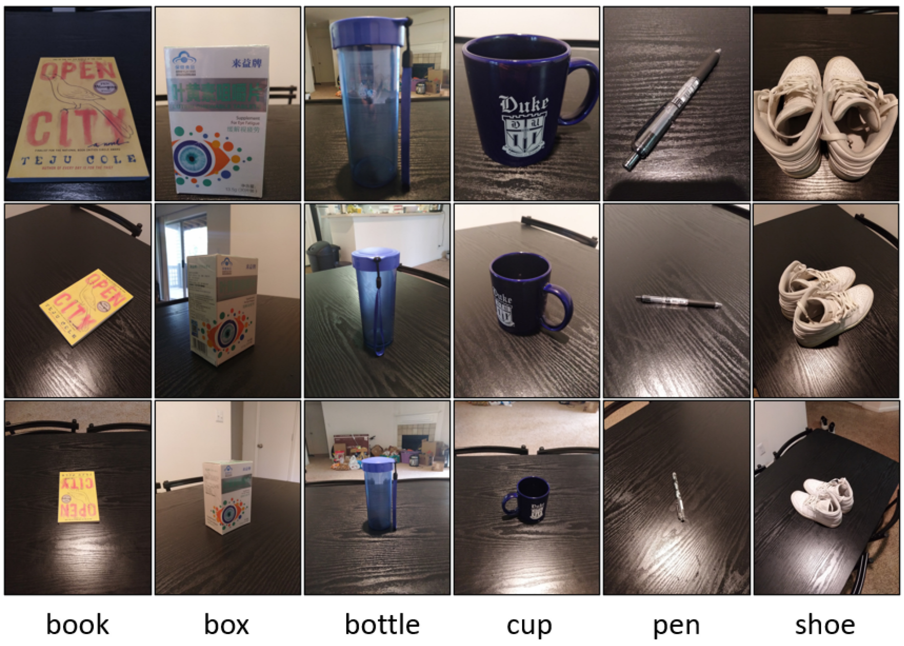


Fig. 4. Examples of the pristine images that are collected in our RMVMDD dataset.

Gaussian blur image set: To generate three different levels of Gaussian blur, following Equation (2), we adjust the camera focal length, f, to 10cm, 12cm, and 15cm, respectively, while having the other three parameters L, D, and A fixed. The distance between the camera and object is approximately 60cm such that the object occupies one-ninth of the total area of the image. We collect $6 \times 6 \times 6 \times 3 = 648$ Gaussian blur images in total.

Noise image set: We use the smartphone in a dark room (with 7lux) and set the ISO value of the camera to 400, 800, and 1600, respectively, to generate images with three different levels of noise. As the ISO value may also affect the image brightness, to have a constant image brightness, we manually adjust the shutter speed for exposure compensation [54] for different ISO values. We collect $6 \times 6 \times 6 \times 3 = 648$ Noise images in total.

Multi-distortion image sets: We also collect images that contain multiple distortions. Specifically, the three image distortions lead to four multi-distortion combinations: *Noise + Motion blur, Noise + Gaussian blur, Motion blur + Gaussian blur*, and *Noise + Motion blur + Gaussian blur*. For each of the combinations, three different distortion levels are considered. Following the methods used in generating single-distortion images, we collect the multi-distortion images by adjusting the corresponding parameters during the image capturing. For instance, to collect images with "a low-level of Noise + Motion blur," we set the ISO value to 400 and camera exposure time of 0.1s. In total, we collect $6 \times 6 \times 6 \times 4 \times 3 = 2{,}592$ multi-distortion images.

6 CYCLEGAN-BASED IMAGE DISTORTION AUGMENTATION

To ensure robustness against image distortion, a large volume of distorted images is required to train the DNNs. Below, we introduce our CycleGAN-based image distortion augmentation.

6.1 CycleGAN for Image Distortion Augmentation

Our goal is to synthesize distorted images from pristine images, which are much easier to collect. We formulate this as an unpaired image-to-image translation problem, that is: *translate a pristine image to a distorted image in the absence of paired training examples*. To achieve this, we leverage the CycleGAN model [42], which has been widely used to address domain adaptation in image-based [42], motion-signal-based [64], and acoustic-signal-based [65] sensing and recognition. Formally, we assume that the pristine images are sampled from domain A with distribution P_A, while images that contain a single type of distortion (i.e., Motion blur, Gaussian blur, or Noise) are sampled from domain B with distribution P_B, and $P_A \neq P_B$. We denote a pristine image as $x \sim P_A$ and the distorted image as $y \sim P_B$. For each pair of "pristine-to-distortion" translation (we have three pairs of translation correspond to the three types of image distortion considered in this article), our goal is to learn a set of *two bijective mapping functions*, $G_{AB} : A \rightarrow B$ and $G_{BA} : B \rightarrow A$, between domains A and B, given training samples $\{x_i\}_{i=1}^N$ and $\{y_j\}_{j=1}^M$. Specifically, in this work, $\{x_i\}_{i=1}^N$ is the *pristine image set* collected in Section 5, whereas $\{y_j\}_{j=1}^M$ is the *motion blur*, the *Gaussian blur*, or the *Noise* image set given the targeted distortion type.

Figure 5 shows the architecture of our CycleGAN model, which consists of *two bijective generators* (mapping functions), G_{AB} and G_{BA}, and *two corresponding discriminators* (adversaries), D_A and D_B. The generator G_{AB} ensures the synthesized image, $\hat{y} = G_{AB}(x), x \sim P_A$, is indistinguishable from the real distorted image $y \sim P_B$ by the discriminator D_B that is specifically trained to differentiate $\hat{y}$ from y. Symmetrically, generator G_{BA} ensures the output $\hat{x} = G_{BA}(y), y \sim P_B$ cannot be distinguished from $x \sim P_A$ by its adversary D_A. Moreover, to maintain the cycle consistency of the image translation, CycleGAN ensures that the outputs of the two generators are reversible, such that $G_{BA}(G_{AB}(x)) \approx x$ and $G_{AB}(G_{BA}(y)) \approx y$; e.g., if we translate an image from pristine to Motion blur, and then translate it back from Motion blur to pristine, we should obtain the same pristine image.

6.2 CycleGAN Implementation

6.2.1 Objective Function. Our objective function is composed of three loss terms: the least square adversarial loss [46, 66], the cycle consistency loss [42, 66, 67], and the identity loss [68]. For generators G_{AB} and G_{BA} and discriminators D_B and D_A, we can define the three loss terms as follows:

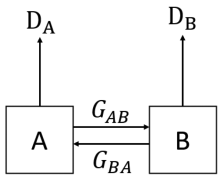


Fig. 5. CycleGAN network that consists of two generators (G_{AB} and G_{BA}) and two discriminators (D_A and D_B). Notation A indicates the domain of the pristine image; B refers to the domain of a specific type of distorted images.

(1) Least square adversarial loss: For G_{AB} and D_B, the corresponding adversarial loss measures how distinguishable the synthesized image $\hat{y} = G_{AB}(x), x \sim P_A$ is from the real image $y \sim P_B$, and is defined as

$$\mathcal{L}_{lsadv}(G_{AB}) = \mathbb{E}_{x \sim P_A}[(D_B(G_{AB}(x)) - 1)^2]. \tag{3}$$

The smaller the $\mathcal{L}_{lsadv}(G_{AB})$, the better the performance of generator G_{AB}. Similarly, we can have the corresponding adversarial loss for G_{BA} and D_A as

$$\mathcal{L}_{lsadv}(G_{BA}) = \mathbb{E}_{y \sim P_B}[(D_A(G_{BA}(y)) - 1)^2]. \tag{4}$$

We use the least square loss here as it has shown better stability during training [66].

(2) Cycle consistency loss: It measures the difference between the original image $x \sim P_A$ and reconstructed image $G_{BA}(G_{AB}(x))$ and enforces the cycle consistency between the two generators, i.e., $G_{BA}(G_{AB}(x)) \approx x$. The cycle consistency loss for G_{AB} is defined as

$$\mathcal{L}_{cycle}(G_{AB}) = \mathbb{E}_{x \sim P_A}[||G_{BA}(G_{AB}(x)) - x||_{L_1}]. \tag{5}$$

Similarly, for G_{BA}, we have

$$\mathcal{L}_{cycle}(G_{BA}) = \mathbb{E}_{y \sim P_B}[||G_{AB}(G_{BA}(y)) - y||_{L_1}]. \tag{6}$$

(3) Identity loss: To preserve the color composition between the synthesized and original images, we adopt the identity loss [68] to regularize the generator, such that it can generate identical images when real images from the target domain are used as the input. The identity loss is defined as

$$\mathcal{L}_{identity}(G_{AB}, G_{BA}) = \mathbb{E}_{y \sim P_A}[||G_{AB}(y) - y||_{L_1}] + \mathbb{E}_{x \sim P_B}[||G_{BA}(x) - x||_{L_1}]. \tag{7}$$

Overall objective: The final objective function for training the two generators is the sum of the three loss terms:

$$\begin{aligned} \mathcal{L}(G_{AB}, G_{BA}) = {} & \mathcal{L}_{lsadv}(G_{AB}) + \mathcal{L}_{lsadv}(G_{BA}) \\ & + \lambda_1 \mathcal{L}_{cycle}(G_{AB}) + \lambda_1 \mathcal{L}_{cycle}(G_{BA}) \\ & + \lambda_2 \mathcal{L}_{identity}(G_{AB}, G_{BA}), \end{aligned} \tag{8}$$

where $\lambda_1, \lambda_2 \geq 0$ are two hyper-parameters that control the contribution of the cycle consistency loss and identity loss to the overall objective. The objective function used in training the two discriminators contains only the least square adversarial loss and is defined as

$$\begin{aligned} \mathcal{L}(D_A, D_B) = {} & \mathbb{E}_{x \sim P_A}[(D_B(G_{AB}(x)))^2] \\ & + \mathbb{E}_{y \sim P_B}[(D_B(y) - 1)^2] \\ & + \mathbb{E}_{y \sim P_B}[(D_A(G_{BA}(y)))^2] \\ & + \mathbb{E}_{x \sim P_A}[(D_A(x) - 1)^2]. \end{aligned} \tag{9}$$

Finally, our CycleGAN is trained by solving

$$\arg\min_{G_{AB}, G_{BA}} \arg\min_{D_A, D_B} [\mathcal{L}(G_{AB}, G_{BA}) + \mathcal{L}(D_A, D_B)]. \tag{10}$$

6.2.2 Implementation. In this work, we are interested in synthesizing three types of distorted images using pristine images as input. We formulate this task as translating original images from the *pristine domain to one of the three distortion domains* (i.e., Motion blur, Gaussian blur, and Noise). Hereafter, instead of using the notation G_{AB}, we specifically use the notation $G_{P\rightarrow MBL}$, $G_{P\rightarrow GBL}$, and $G_{P\rightarrow N}$ to denote the generator of "pristine-to-motion-blur," "pristine-to-Gaussian-blur," and "pristine-to-noise," respectively.

We implement the generators by following the network design proposed by Johnson et al. [69], which has shown great performance in image transformation [42]. Specifically, the network contains two stride-2 convolutional layers, nine residual blocks [70], and two stride-$\frac{1}{2}$ convolutional layers. For the two discriminators, we adopt 70×70 PatchGAN [71], which determines whether the 70×70 overlapping image patches are real or synthesized (fake). By restricting the discriminator's attention in local image patches, PatchGAN improves the quality of the synthesized images and prevents them from lacking high-frequency features [71].

We use the corresponding image sets (one pristine and three distorted image sets) collected in the RMVMDD dataset for training and employ the Adam optimizer [72] with a learning rate of 0.0002 during the training. For $G_{P\rightarrow MBL}$ and $G_{P\rightarrow GBL}$, we set the values of the two parameters, λ_1 and λ_2, to 10. For $G_{P\rightarrow N}$, we set λ_2 to 25 to regularize the brightness of the synthesized images, as the brightness of the noise images is relatively low.

7 DISTORTION-TOLERANT RECOGNIZER

7.1 Image Distortion Classifier

Different types of distortion have distinct impacts on the frequency domain features of the original images. For instance, as discussed in Section 3.1, Gaussian blur can be considered as the process of mapping a sharp pixel point into a blurred spot on the image plane. This is equivalent to applying a circularly shaped low-pass filter on the image, which filters out the high-frequency components in the original image. Similarly, motion blur can be considered as a directional low-pass filter that smooths the original image along the direction of the motion [8]. Lastly, the Fourier transform of noise is approximately constant over the entire frequency domain, whereas the original image contains mostly low-frequency information [73]. Hence, an image with severe noise will have higher signal energy in the high-frequency components theoretically.

Figure 6 gives the Fourier spectrum of images containing different distortions. The spectrum is shifted such that the DC value is displayed in the center of the image. For any point in the spectrum, the farther away it is from the center, the higher is its corresponding frequency. The brightness of the point in the spectrum indicates the energy of the corresponding frequency component in the image. We can see that the pristine image contains components of all frequencies, and most of the high-energy components are in the lower-frequency domain. Differently, Gaussian blur heavily removes the high-frequency components of the image in all directions. Similarly, motion blur removes the high-frequency components in the image along the direction of motion. Thus, we can see a strong diagonal line that is perpendicular to the direction of the motion blur. The thinner the line, the higher the motion blur level, and vice versa. Lastly, because noise often occurs in low-illumination conditions, although noise will introduce irregular pixels, the reduction in brightness of the picture caused by low illumination will weaken the high-frequency components of the picture in all directions. We leverage these distinct frequency domain patterns for distortion classification.

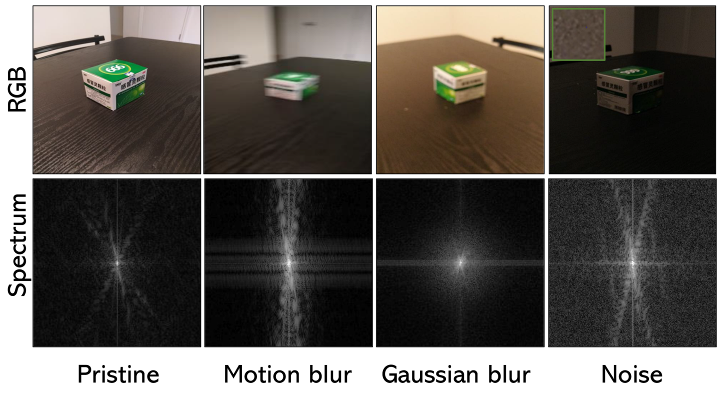


Fig. 6. The Fourier spectrum of images containing different distortions. Different distortions have distinct impacts on the frequency domain features of the images. The patterns are independent of the semantic content in the image.

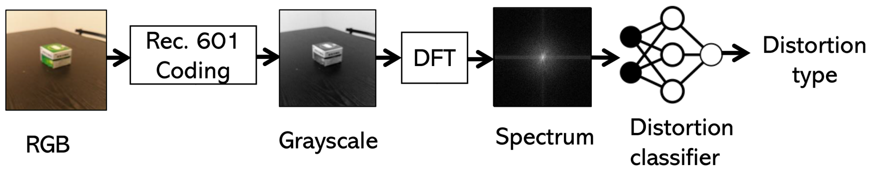


Fig. 7. The pipeline of the image distortion classification. The Fourier spectrum is used as the input of the distortion classifier for training and classification.

Table 3. The Architecture of the Image Distortion Classifier

Layer	**Size In**	**Size Out**	**Filter**
conv1	$224 \times 224 \times 3$	$111 \times 111 \times 32$	3×3, 2
conv2	$111 \times 111 \times 32$	$109 \times 109 \times 16$	3×3, 1
pool1	$109 \times 109 \times 16$	$54 \times 54 \times 16$	2×2, 2
conv3	$54 \times 54 \times 16$	$54 \times 54 \times 1$	1×1, 1
pool2	$54 \times 54 \times 1$	$27 \times 27 \times 1$	2×2, 2
flatten	$27 \times 27 \times 1$	729	
fc	729	4	

The parameters in the Size In and Size Out columns refer to the *height* $\times$ *width* $\times$ *depth*.

Figure 7 shows the pipeline of the image distortion classification. First, we convert the original RGB image into grayscale using the standard Rec. 601 luma coding method [74]. Then, we apply the two-dimensional **discrete Fourier transform (DFT)** to obtain the Fourier spectrum of the grayscale image and shift the zero-frequency component to the center of the spectrum. The centralized spectrum is used as the input for the distortion classifier. As shown in Table 3, the distortion classifier adopts a shallow CNN architecture, which consists of three convolutional layers (*conv1*, *conv2*, and *conv3*), two pooling layers (*pool1* and *pool2*), one flatten layer, and one fully connected layer (*fc*). This shallow design avoids over-fitting and ensures low computation latency.

7.2 Recognition Experts

Based on the output of the distortion classifier, one of the four dedicated recognition experts is selected for the image recognition. We consider either MobileNetV2 [5] or the VGG16 [56] as the base DNN model for the recognition experts. We initialize all the CNN layers with the values trained on the full ImageNet dataset. Then, to fine-tune the experts on a target dataset, we add a fully connected layer as the last layer with the number of units corresponding to the number of classes in the target dataset. In the fine-tuning process, we first train the pristine expert, $Expert_P$, using pristine images in the target dataset (RMVMDD collected in Section 5). Then, the three distortion experts, i.e., motion blur expert $Expert_{MBL}$, Gaussian blur expert $Expert_{GBL}$, and Noise expert $Expert_N$, are initialized with the weights of $Expert_P$ and fine-tuned using the corresponding distortion images. During the fine-tuning, half of the images in the mini-batch are pristine and the other half are distorted images synthesized by using the CycleGAN model introduced in Section 6. The augmented distorted images contain distortion on all three severity levels. This ensures better robustness against variations in the distortion level (i.e., it helps in minimizing the effect of domain-induced changes in feature distribution [17]) and helps the CNNs to learn features from both pristine and distorted images [75].

8 COLLABORATIVE MULTI-VIEW IMAGE RECOGNITION

Mobile users that are in close proximity usually exhibit a strong correlation in both device usage and spatial-temporal contexts [22, 35]. This is especially true in the mobile AR scenario where the requested image recognition tasks are usually correlated temporally (e.g., consecutive image frames from the same user [6, 7, 36]) and spatially (e.g., different users aim to recognize the same object [12, 22]). For instance, in a museum, visitors that are in *close proximity* may use the AR application to recognize and augment information of the *same artwork*. Although images are taken from different positions or at different times, they contain the same object [22, 76]. CollabAR exploits the spatial-temporal correlation among the images to enable collaborative multi-view image recognition.

8.1 Correlated Image Lookup

8.1.1 Anchor-based Pose Estimation. To identify the spatially and temporally correlated images, CollabAR exploits Google ARCore [25] to estimate the pose (the position and orientation) of the device when the image was taken. When the user is moving in space, ARCore leverages a process called **concurrent odometry and mapping (COM),** which aggregates the image frames taken by the camera and the inertial data captured by the device's motion sensors to simultaneously estimate the orientation and location of the mobile device relative to the world.

Specifically, CollabAR leverages the Cloud anchors [77] provided by ARCore as the references for pose estimation. An anchor is a virtual identifier that describes a fixed pose in the physical space. The numerical pose estimation of the anchor is updated by the ARCore over time, which makes the anchor a stable pose reference against cumulative tracking errors [3, 78]. In our design, anchors are created and managed by the administrator in advance. When running the application, anchors in the space can be resolved by the users to establish a common spatial reference among them. In general, an anchor should be placed at the location of the object of interest. However, to ensure low system latency in resolving the anchor and provide better user experience, we take two practical factors into account when deciding the placement of the anchor in the space. First, when the object of interest has reflective surfaces or surfaces without visual features (e.g., shiny objects or white walls), we place the anchors at locations where the background scene contains easily distinguishable visual features. Second, we also consider the possible viewing positions and angles

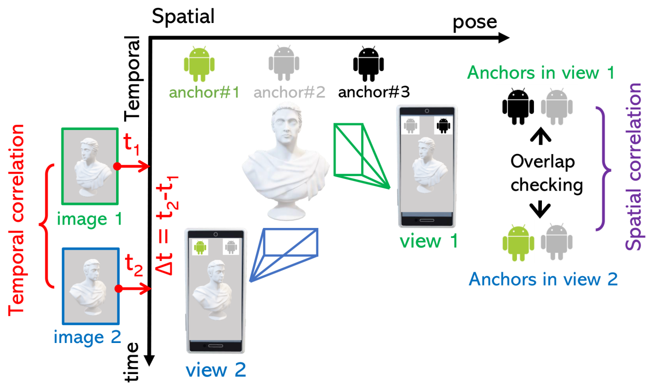


Fig. 8. An example of correlated image lookup. Images of the same artwork are taken from different poses, $view1$ and $view2$, at different times, t_1 and t_2. The anchors in the image views are used to check the spatial correlation, and the timestamps are used to check the temporal correlation.

of the potential users and select locations that make the anchors more likely to appear in the users' view. The Google Cloud Anchor service can resolve up to 20 cloud anchors simultaneously, and each anchor can cover a 4m × 4m area [77, 79]. As we are only using the anchors to achieve coarse-grained pose estimation, 20 cloud anchors can cover an area up to 320m^2 to support the mobility of the users. Lastly, although the resolving time increases with the number of anchors placed, the average time spent in resolving a single anchor is only 1 second based on our measurement and 1.5 seconds according to previous work [30]. Lastly, since CollabAR only requires coarse-grained spatial estimation, the pose estimation accuracy of Google ARCore does not affect its performance in identifying the spatial correlation among multiple users.

As a toy example shown in Figure 8, three cloud anchors have been placed in space to identify three different poses. When the user is taking an image of the artwork from $view1$ at time t_1, the anchors in the camera view will be sent to the edge and associated with the timestamp t_1. As shown previously in Figure 3, the anchors and the timestamps are stored in the *anchor-time cache* in the format of $<$ imageID, {anchorIDs}, timestamps $>$. When a new image, $image2$, is taken at time t_2 from $view2$ either by the same or by a different user, the *spatial-temporal image lookup module* searches the anchor-time cache to find previous images that are spatially and temporally correlated with the new one.

8.1.2 Spatial-temporal Image Lookup. For any image in the cache, $image_{Cached}$, it is considered as spatially correlated with the new image, $image_{New}$, if the images satisfy

$$\{\text{anchorIDs}\}_{\text{New}} \cap \{\text{anchorIDs}\}_{\text{Cached}} \neq \emptyset, \tag{11}$$

where $\{\text{anchorIDs}\}_{\text{New}}$ and $\{\text{anchorIDs}\}_{\text{Cached}}$ are the sets of anchors that appeared in views of $image_{New}$ and $image_{Cached}$, respectively. If two images contain the same anchor in their views, they are spatially correlated. For instance, in Figure 8, $image1$ contains the same anchor, $anchor\#2$, that also appears in the view of $image2$. Thus, they are spatially correlated. Similarly, the cached image is considered as temporally correlated to the new image if the images satisfy

$$\Delta t = t_{New} - t_{Cached} \quad \text{and} \quad \Delta t \leqq \text{T}_{\text{fresh}}, \tag{12}$$

where t_{New} and t_{Cached} are the timestamps of the new and cached images, respectively. Δt is the freshness of $image_{Cached}$ with respect to $image_{New}$. If the freshness Δt is within the freshness threshold, T_{fresh}, $image_{Cached}$ is considered as temporally correlated with $image_{New}$. Note that

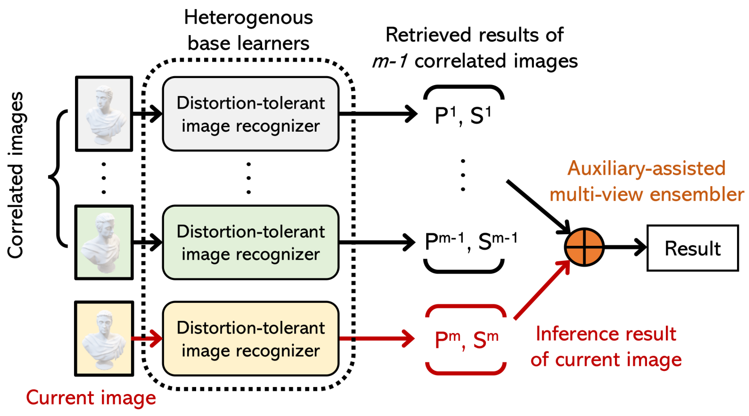

Heterogenous base learners
Retrieved results of *m-1* correlated images
Correlated images
Distortion-tolerant image recognizer
Distortion-tolerant image recognizer
Distortion-tolerant image recognizer
P¹, S¹
Pᵐ⁻¹, Sᵐ⁻¹
Pᵐ, Sᵐ
Auxiliary-assisted multi-view ensembler
Result
Inference result of current image
Current image

Fig. 9. The architecture of Auxiliary-assisted Multi-view Ensemble Learning (AMEL). Different base learners (i.e., different recognition experts) are used for the recognition. The outputs of the previous images, i.e., the probability vectors, are stored in the results cache. During aggregation, the probability vector of the current image, $\mathcal{P}^m$, and the corresponding auxiliary information, $\mathcal{S}^m$, are combined with that of the $m-1$ spatial-temporal correlated images to improve the overall accuracy. The set of auxiliary features, $(\mathcal{S}^1, \ldots, \mathcal{S}^m)$, is used to assist the multi-view ensembler in discriminating the confidence of the m inference results.

the setting of the freshness threshold $\mathrm{T_{fresh}}$ varies with the application scenarios. In cases where the object at a given position changes frequently, e.g., animals in the zoo, we should use a lower freshness threshold. Differently, in scenarios where the positions of the objects do not change very often, e.g., exhibited items in a museum or large appliances in a building, a higher freshness threshold can be tolerated. An image in the anchor-time cache is considered as spatially and temporally correlated to the new image if they satisfy Equations (11) and (12) at the same time. The IDs of the identified correlated images are forwarded to the multi-view ensembler for aggregation.

8.2 Auxiliary-assisted Multi-view Ensemble Learning

In image recognition, ensembling the results of multiple base classifiers is a widely used approach to improve the overall recognition accuracy [38, 80]. Following this trend, CollabAR aggregates the recognition results of the spatially and temporally correlated images to improve the recognition accuracy of the current image.

As shown previously in Figure 3, using the image IDs identified by the correlated image lookup module as the key, the multi-view ensembler retrieves the results of the correlated images from the results cache. This is done by retrieving any stored records in the result cache, i.e., $<\mathrm{imageID}, \mathrm{inferenceResult}>$, with imageID matches to the identified images. As shown in Figure 9, assuming that $m-1$ correlated images are identified, the inference result of the current image (the probability vector $\mathcal{P}^m$) is aggregated with that of the $m-1$ correlated images $(\mathcal{P}^1, \ldots, \mathcal{P}^{m-1})$ by the ensembler. However, given the heterogeneity of the m images (i.e., images are captured from different angles, suffer from different distortions with different distortion levels), the images lead to unequal recognition performance. To quantify their performance and help the ensembler in aggregating the results dynamically, auxiliary features [39, 81] can be used. In this work, we propose to use the **normalized entropy** as the auxiliary feature. The normalized entropy $\mathcal{S}^k$ measures the recognition quality and the confidence of the distortion-tolerant image recognizer on the recognition of the kth image instance. More specifically, $\mathcal{S}^k$ can be calculated from the probability vector as follows:

$$\mathcal{S}^k(\mathcal{P}^k) = -\sum_{i=1}^{|C|} \frac{p_i \log p_i}{\log |C|}, \tag{13}$$

where $\mathcal{P}^k = \{p_1, \ldots, p_{|C|}\}$ is the probability vector of the kth base learner on the image instance, and $|C|$ is the number of object categories in the dataset. The value of normalized entropy is between 0 and 1. A value close to 0 indicates that the distortion-tolerant image recognizer is confident about its recognition on the image instance, whereas a value close to 1 means that it is not confident.

In AMEL, the set of normalized entropy, $(\mathcal{S}^1, \ldots, \mathcal{S}^m)$, is used as the weight in averaging heterogeneous outputs of the spatial-temporal correlated images. Specifically, given the current image and the $m-1$ correlated images, the final aggregated recognition output $\mathbf{P}$ can be expressed as $\mathbf{P} = \mathcal{N}\left(\sum_{i=1}^{m}(1-\mathcal{S}^i)\mathcal{P}_i\right)$, where $\mathcal{P}_i$ is the probability vector of the ith image, $(1-\mathcal{S}^i)$ is the associated weight, and $\mathcal{N}$ is the normalization operator that scales the entries in the aggregated vector to the range of [0, 1]. This allows AMEL to dynamically adjust the weights in aggregating the m images during the classification. Overall, as the images result in unequal recognition accuracy, AMEL is more robust against this variance than standard averaging methods, which would assign equal weights to multi-view images during the aggregation [37].

Discussion. Note that CollabAR supports multi-user collaboration in both asynchronous and synchronous manners. It can also aggregate the consecutive image frames (e.g., image frames in the video) captured by a single user to improve the recognition accuracy, as long as the images are spatially and temporally correlated. There are a number of potential application scenarios in which multiple users are co-located in space but not in time. For example, members of maintenance, construction [82], and cleaning crews [83] could leave virtual annotations for members in the next inspection. These virtual annotations are used to highlight specific areas or objects, which need to be checked, repaired, or sanitized [82]. Similarly, different product, automotive, or aerospace designers and engineers working on the same project may leave feedback and suggestions when they visit a studio or a factory floor [34]. These use cases are potential examples of how CollabAR can be used for asynchronous collaboration.

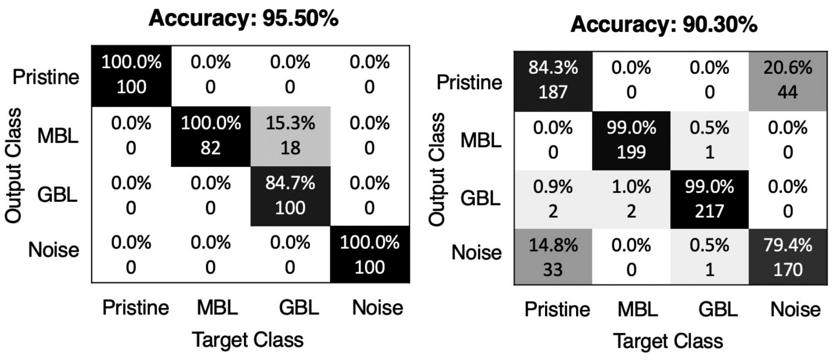
In general, asynchronous collaboration that employs image recognition to generate virtual content is advantageous because it does not require the printing of fiducial markers, and it does not damage the aesthetic quality of the space in which it is used. Furthermore, the use cases listed above may involve challenging visual conditions and image distortion. For instance, industrial or warehouse settings often contain dark environments, introducing Gaussian noise; busy users with many spaces to check may be moving quickly, introducing motion blur. As such, a distortion-tolerant system is required to support swift and accurate recognition in practical applications.

9 EVALUATION

Below, we provide a comprehensive evaluation of CollabAR. We examine the overall accuracy of CollabAR in image recognition as well as its end-to-end system latency and memory usage.

9.1 System Implementation

The client of CollabAR is implemented on Android smartphones. We leverage the Google ARCore SDK to realize the anchor-based pose estimation module introduced in Section 8.1.1. The edge server is a desktop with an Intel i7-8700k CPU and a Nvidia GTX 1080 GPU. We realize the server of CollabAR using the Python Flask framework. The distortion-tolerant image recognizer and the multi-view ensembler are implemented using Keras 2.3 on top of the TensorFlow 2.0 framework. The client and the server communicate through the HTTP protocol.



(a) MobileDistortion (b) RMVMDD

Fig. 10. Confusion matrix and accuracy of the image distortion classifier on different datasets: (a) MobileDistortion and (b) RMVMDD. On average, the distortion classifier achieves 95.5% and 90.3% accuracy on the two datasets, respectively.

9.2 Distortion Classifier Performance

We first evaluate the performance of the image distortion classifier using the MobileDistortion (collected in Section 3.3) and RMVMDD (collected in Section 5) datasets. We perform threefold cross-validation on each of the two datasets.

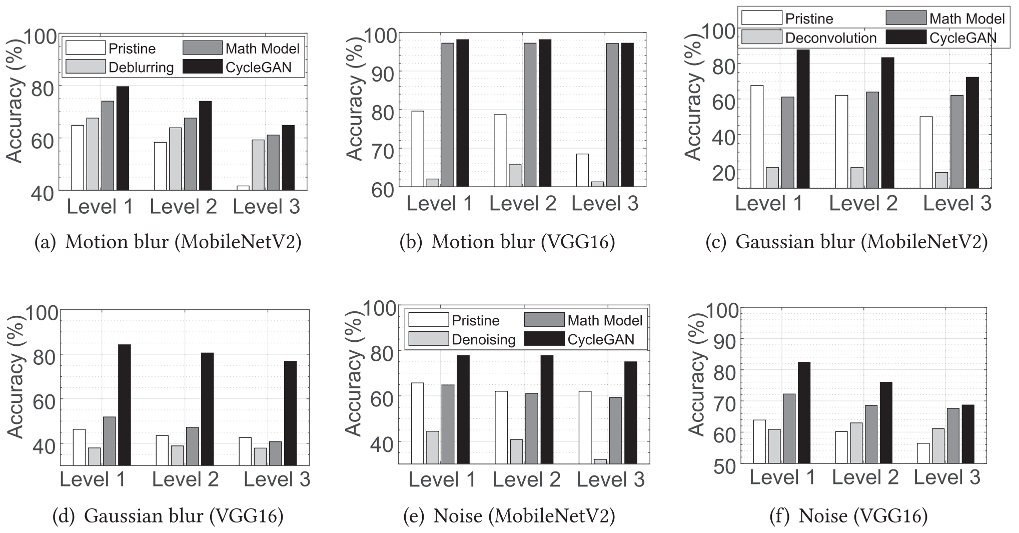
The confusion matrix and the accuracy of the image distortion classifier are shown in Figure 10. The classifier achieves 95.5% and 90.3% accuracy on the MobileDistortion, and RMVMDD datasets, respectively. For MobileDistortion, most of the classification errors happen in differentiating MBL and GBL. As shown in Figure 10(a), 15.3% of the GBL distorted images are misclassified as MBL. This is due to the similarity between these two distortions. Specifically, MBL and GBL have similar impact on the Fourier spectrum and result in misclassification when the direction of the motion blur is parallel or perpendicular to the central horizontal line of the image. However, for the RMVMDD dataset, as shown in Figure 10(b), the classification error between MBL and GBL is less than 1%. This is because we only consider horizontal motion blur during the data collection. Specifically, as shown in Figure 6, we can see a strong vertical line in the Fourier spectrum for images that are containing GBL. This distinct pattern helps the classifier to differentiate MBL and GBL. In addition, we also notice that when the noise level is low, i.e., images in the RMVMDD dataset that are captured when the ISO value equals 400, the distorted images are sometimes misclassified as pristine images. This is intuitive, as the impact of the image distortion on the Fourier spectrum is limited when the distortion level is low and, thus, results in similar spectrum feature as pristine images. Overall, on the two image datasets, the distortion classifier achieves 93% accuracy on average.

9.3 Image Recognition Accuracy

Below, we evaluate the image recognition accuracy of CollabAR using the RMVMDD dataset. We first examine the performance of the distortion-tolerant image recognizer, followed by the evaluation of CollabAR in multi-view and multiple distortions scenarios.

9.3.1 Performance of Distortion-tolerant Image Recognizer. We examine the accuracy of both MobileNet-based and VGG-based implementations. We consider the single-distortion scenario and gradually increase the distortion level to investigate its impact on the recognition accuracy.

Setup. We consider four baselines in this experiment: (1) We use the pristine images to train the *Pristine* expert and use it for the image recognition without considering distortion-specific



(a) Motion blur (MobileNetV2) (b) Motion blur (VGG16) (c) Gaussian blur (MobileNetV2)

(d) Gaussian blur (VGG16) (e) Noise (MobileNetV2) (f) Noise (VGG16)

Fig. 11. Performance of the distortion-tolerant image recognizer given different distortions, distortion levels, DNN architectures, and baseline methods.

recognition. This represents the worst case and shows how different image distortions may affect the recognition performance. (2) We apply different image restoration methods to recover the distorted images. Specifically, we apply the non-local means denoising [84] for noise filtering, the scale-iterative up-scaling network [85] to restore images from Gaussian blur, and the image deconvolution CNN [86] to recover images from motion blur. Then, the restored images are used as the inputs to the Pristine expert for recognition. This method investigates the recognition performance when image restoration methods are used to address the image distortions. (3) We apply mathematical models to synthesize distorted images. Specifically, following our previous work [1], we use the zero-mean additive Gaussian noise model for Noise [11], the non-uniform motion blur kernel for Motion blur [18], and the two-dimensional Gaussian kernel for Gaussian blur [19]. The augmented images are used to train the three distortion-specific recognition experts. This method investigates the recognition performance when mathematical-model-based image augmentation is used. (4) Lastly, we leverage the CycleGAN-based image augmentation to synthesize distorted images and train the three recognition experts. The results are shown in Figure 11. We have the following major observations.

First, on average, in terms of three distortion types, the VGG16-based implementation outperforms the MobileNetV2-based one by only 2% on average. Although VGG16 is deeper and has more convolutional layers, MobileNetV2 takes advantages of the depth-wise separable convolution as an efficient building block to ensure more efficient image feature extraction and uses a linear bottleneck for better feature transformation [5]. One exception is the Motion blur; as shown in Figures 11(a) and 11(b), VGG16 is more robust against motion blur and achieves higher accuracy than MobileNetV2.

Second, the *CycleGAN*-based data augmentation significantly improves the robustness of the distortion-tolerant image recognizers against the image distortions. Taking the MobileNetV2-based implementation as an example, the *CycleGAN*-based method improves the accuracy of the *Pristine* expert by 17.9%, 21.2%, and 13.6% on average over the three distortion levels for Motion blur, Gaussian blur, and Noise, respectively. Similarly, when comparing with the *Math Model*-based

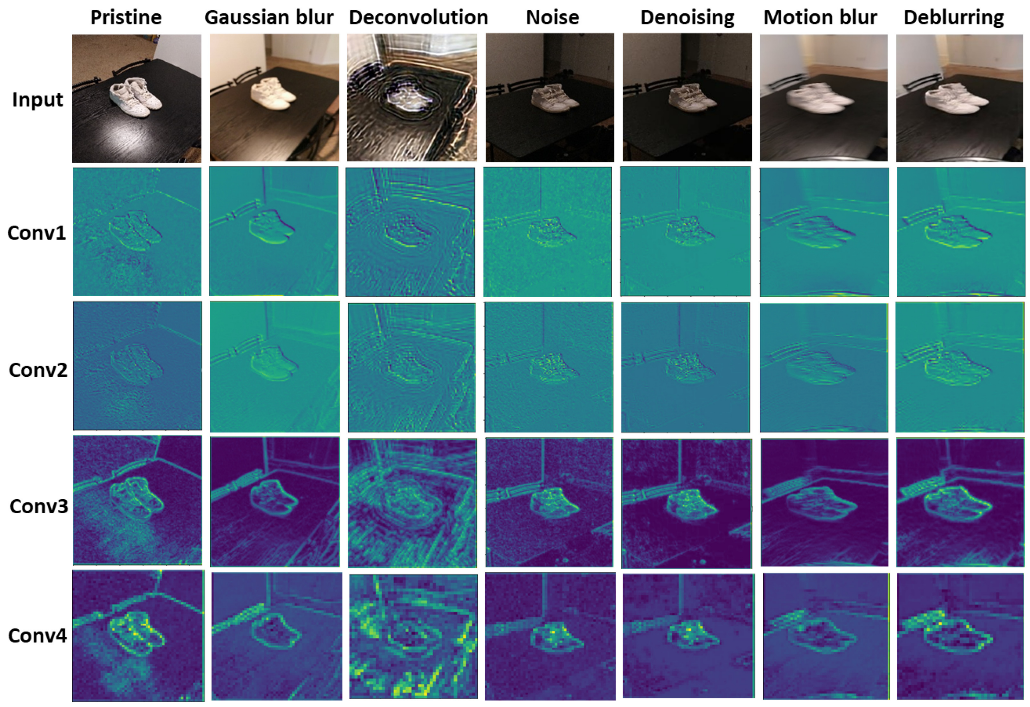


Fig. 12. Feature maps generated by different convolutional layers of the MobileNetV2: rows from top to bottom correspond to raw image input and feature map generated from the first (Conv1), second (Conv2), third (Conv3), and the fourth (Con4) convolutional layer; columns from left to right correspond to Pristine image, Gaussian blur image, Gaussian blur image recovered by deconvolution, Noise image, Noise image recovered by denoising, Motion blur image, and Motion blur image recovered by deblurring. Overall, image restoration methods introduce unexpected "perturbations" to the restored images that can reduce the recognition performance of the DNN.

data augmentation, the *CycleGAN*-based method improves the accuracy by 5.2%, 18.7%, and 15.1% on average over the distortion levels for Motion blur, Gaussian blur, and Noise, respectively.

Third, although image restoration methods can reduce distortions, the restored images are friendly only to human eyes but contain unexpected "perturbations" that will hurt the performance of the DNN [87]. As shown in Figure 11, we can observe that, in most of the examined scenarios, the image restoration methods (*Deblurring*, *Deconvolution*, or *Denoising*) achieve the lowest recognition accuracy. To illustrate, we visualize the intermediate feature maps generated by the lower convolutional layers of MobileNetV2, as lower layers of CNN contain color, texture, and low-level image features [88]. As shown in Figure 12, we plot the feature maps generated by the first to the fourth convolutional layers (after the activation operation) of the MobileNetV2. For better visual effects, each of the feature maps is averaged over the three color-specific feature maps in the RGB channels and resized using the bi-cubic interpolation to have the same size as the input image. The bright areas in these feature maps represent extracted feature points that have high activation values, which will have a high impact on the final classification results (as they are more likely to be maintained in the final layer).

We can notice that the *deconvolution* method introduces additional perturbations in feature maps of the restored images, while the *denoising* method removes the original low-level features in the

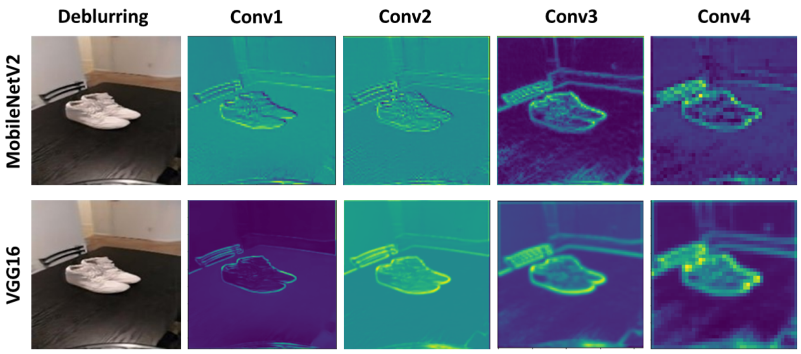


Fig. 13. Feature maps generated by the first to the fourth convolutional layer of MobileNetV2 and VGG16 when taking the Motion blur image recovered by deblurring as the input.

restored images. Thus, both of them result in recognition accuracy that is worse than the distorted images (Gaussian blur or Noise images). Specifically, as shown in Figure 12, compared to the Pristine image, the image restored by the deconvolution method contains many unexpected textures with high activation values in all four feature maps. Similarly, the denoising method removes the low-level features in the original image (i.e., we can notice some sparse feature points in the background of the pristine image, but they are removed in the denoised image). As a consequence, when using the Pristine expert for the recognition, the deconvolution and denoising methods result in poor recognition accuracy. An exception is the Motion blur. The image restored by the deblurring method captures the low-level features of the object well without distortions: lines and texture of the shoes are well maintained in the four feature maps. Overall, the abilities of different restoration methods in image recovering and in maintaining the low-level image features lead to different recognition performance.

Lastly, from Figures 11(a) and 11(b), we can also notice that the deblurring method achieves a higher accuracy with MobileNetV2-based implementation than with the VGG16-based one. To explain, we compare the feature maps generated by MobileNetV2 and VGG16 when taking the Motion blur image recovered by the deblurring method as the input. As shown in Figure 13, MobileNetV2 exhibits a better capability in maintaining the texture features than VGG16. Specifically, the texture details of the shoes are maintained better in the feature maps generated by MobileNetV2 than that generated by VGG16. By contrast, VGG16 shows a better capability in extracting the shapes of the objects, i.e., higher activation values on edges of the shoes and the table, but lower activation values on the texture details. Moreover, since we have pre-trained both MobileNetV2 and VGG16 on the ImageNet dataset, they are strongly biased towards recognizing texture features rather than shapes of the objects [89]. Thus, as a combined effect, due to the loss of texture feature, VGG16 achieves a lower recognition accuracy than MobileNetV2 for Motion blur.

9.3.2 Performance of Multi-view Collaboration. Below, we evaluate the performance of CollabAR in the multi-view scenario where spatially and temporally correlated images are aggregated to improve the single-view accuracy. Moreover, we investigate how multi-view collaboration can help in multiple distortions cases.

Setup. We focus on the MobileNetV2-based implementation for the recognition experts, as it demonstrates a similar accuracy to the VGG16-based one but a lower computation latency (details are given in Section 9.4). To microbenchmark the end-to-end system design, we evaluate CollabAR with two design options: (1) with and without the image distortion classifier and (2) with and

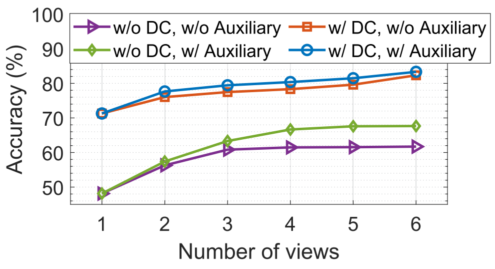


Fig. 14. Accuracy of CollabAR in the multi-view single-distortion scenario.

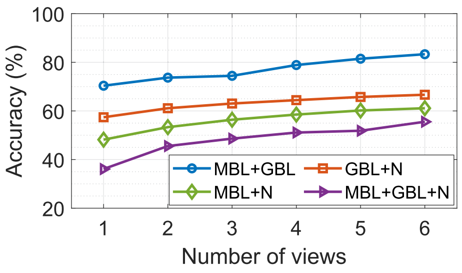


Fig. 15. Accuracy in multi-view multiple-distortions scenario. With six views aggregated, the accuracy can improve by 9.3% to 19.4% given different distortion combinations.

without the auxiliary feature for the multi-view ensembler. We consider up to six views to be aggregated for collaborative recognition.

Multi-view Single-distortion: in the single-distortion case, the image of each view contains one of the three distortions. We set the image distortion to level 3 to evaluate CollabAR with the most severe distortions. The performance is shown in Figure 14. First, with the image distortion classifier, CollabAR can correctly select the recognition expert that is dedicated to the examined distortion image and can significantly improve the recognition accuracy. Given different number of views aggregated, CollabAR with the distortion classifier can improve the overall accuracy by 18% to 23% when compared to the case without a distortion classifier. Second, with the auxiliary feature, CollabAR can dynamically adjust the weights in multi-view aggregation and can improve the overall accuracy up to 7%. Lastly, the accuracy increases with the number of aggregated views. In the case where CollabAR is incorporated with both the distortion classifier and the auxiliary feature, the overall accuracy can be improved by 12% with six views aggregated.

Multi-view Multi-distortion: Below, we examine CollabAR, with both the distortion classifier and the auxiliary feature, in the multiple-distortions scenario. We consider four multiple distortion combinations: MBL+GBL, MBL+N, GBL+N, and MBL+GBL+N. Figure 15 shows the performance of CollabAR given different distortion combinations in the testing image. The distortion is set to a modest level (i.e., level 2 distortion for MBL, GBL, and N). First, we observe a significant drop in the accuracy compared to that in the single-distortion scenario. This is expected: since the dedicated recognition experts are fine-tuned on images with specific types of distortions, multiple distortions will cause a mismatch in the feature distribution between the corrupted testing images and the dedicated recognition experts, and will thus result in a performance drop. Fortunately, the accuracy improves with the number of views aggregated. For different combinations of image distortions, the accuracy can improve by 9.3% to 19.4% given different distortion combinations. Figure 16 shows the accuracy improvements of the multi-view aggregation given different distortion combinations

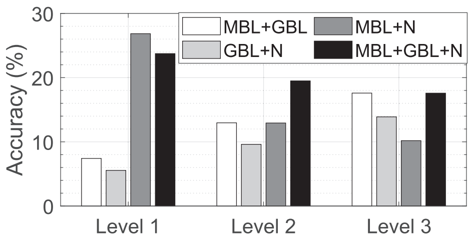


Fig. 16. Accuracy improvement of CollabAR in multi-view multi-distortion scenario given different distortion combinations and distortion levels. With six views aggregated, the overall accuracy is improved by 14.9% on average across different scenarios.

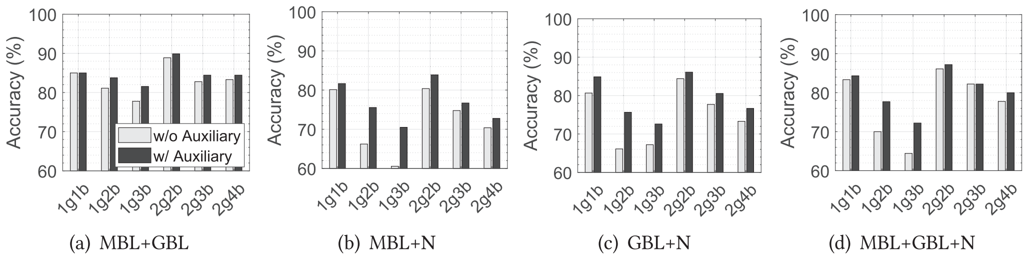


Fig. 17. Performance of CollabAR with and without the auxiliary feature. The images contain multiple distortions and are diverse in the distortion level. With the auxiliary feature, we can always achieve a higher accuracy.

and distortion levels. Overall, with six views aggregated, the overall recognition accuracy can be improved by 14.9% on average.

9.3.3 Advantages of AMEL. Below, we investigate the performance of CollabAR in heterogeneous multi-view multiple distortions scenarios, where images from different views are diverse in distortion type and distortion level. We define a "good view" if the image is pristine, and we define a "bad view" if the image contains high distortion levels (i.e., level 3 distortion for MBL, GBL, and N). We also demonstrate the advantages of the proposed auxiliary-assisted multi-view ensemble over the conventional ensemble method without the auxiliary feature. Figure 17 compares the performance of CollabAR with and without the auxiliary feature. CollabAR can always achieve a higher accuracy with the auxiliary feature. Specifically, the improvement is most pronounced when there are more "bad views" than "good views" in the multi-view aggregation (e.g., "1g2b," "1g3b," "2g3b," and "2g4b," where "1g2b" refers to "one good view and two bad views"). For instance, as shown in Figure 17(d), in a scenario where the image contains three types of distortion, the auxiliary feature can improve the performance by 8% on average and up to 13% in a more diverse case ("1g3b"). Overall, CollabAR achieves 80% and 76% accuracy on average, with and without the auxiliary feature, respectively.

9.4 System Profiling

Below, we present a comprehensive profiling of CollabAR in terms of end-to-end system latency and system memory usage. We consider two deployments of CollabAR: (1) the whole system is deployed on the mobile client and (2) the edge-assisted design in which the computation-intensive recognition pipeline is deployed on the server. We implement CollabAR on three different platforms. Specifically, we use a desktop equipped with an Intel i7-9700k CPU and an Nvidia RTX

Table 4. Specifications of the Three Commodity Platforms

Platform	**CPU**		**RAM**	**GPU**		
	Cores	**Speed**		**Cores**	**GFLOPS**	**Speed**
Desktop	8	3.6GHz	32GB	1,920	6,451	1.68GHz
Nokia 7.1	8	1.8GHz	3GB	1	184.3	720MHz
Pixel 2 XL	8	2.35GHz	4GB	1	567	710MHz

Table 5. The End-to-end Latency (in ms) When the Two Mobile Devices Offload the Image to the Edge Server for Inference

System Implementation	**Mobile Platform**	
	Nokia 7.1	Pixel 2 XL
Distortion Detector + AEML (MobileNetV2)	18.2	20.4
Distortion Detector + AEML (VGG)	19.9	19.8

The server performs inference on the GPU. The results are averaged over 200 trials.

2060 GPU as the edge server. In addition, we consider two commodity smartphones, Nokia 7.1 and Google Pixel 2 XL, to represent middle-range mobile clients. The two smartphones are embedded with the Qualcomm Snapdragon 636 and 835 mobile platforms, respectively, which leads to heterogeneity in both computation and communication performance. The hardware specifications of the three platforms are given in Table 4. We leverage the TensorFlow Lite [90] as the framework when implementing CollabAR on the smartphones. Note that we implement CollabAR on the edge server using the TensorFlow 2.0, which makes it more efficient than the TensorFlow 1.0-based implementation in our previous conference version [1].

9.4.1 End-to-end System Latency. Below, we measure the end-to-end system latency of CollabAR by following the processing pipeline shown in Figure 3. Specifically, we consider both VGG16-based and MobileNetV2-based implementations of the AMEL image recognition component. In our experiment setup, the mobile clients and the edge server are connected through a single-hop Wi-Fi network using the 5GHz wireless channel. Note that the communication latency is largely affected by the wireless network condition. Our measurement is conducted under modest background traffic load. We run 200 trials of the end-to-end image recognition pipeline and report the average time consumed. The results are shown in Table 5. Overall, for the two commodity smartphones that we have examined, the edge-assisted design allows us to achieve more than 50fps continuous image recognition.

9.4.2 Memory Usage. We also measure the runtime memory usage of CollabAR when it is deployed on different hardware platforms. We only report the memory that is allocated to CollabAR: the memory usage before the loading of CollabAR is subtracted from the measurement. The major memory usage is in loading the trained network model, as well as the input and output data of different network layers. The results are shown in Table 6. For both types of implementation, the overall GPU memory usage of CollabAR, including the distortion classifier and the four recognition experts, is well within the available memory of most commodity platforms.

10 CONCLUSION

In this article, we presented CollabAR, an edge-assisted collaborative image recognition system that enables distortion-tolerant image recognition with imperceptible system latency for mobile augmented reality. To achieve this goal, we propose distortion-tolerant image recognition to

Table 6. The Runtime Memory Usage (in MB) when Running CollabAR on Three Hardware Platforms

System Implementation	Processing Unit	Hardware Platform		
		Edge Server	Nokia 7.1	Pixel 2 XL
Distortion Detector + AMEL (MobileNetV2)	CPU	107	23	23
	GPU	1,815	76	85
Distortion Detector + AMEL (VGG16)	CPU	653	230	230
	GPU	3,515	1,202	1,202

The results are averaged over 200 trials.

improve robustness against real-world image distortions, and collaborative multi-view image recognition to improve the overall recognition accuracy. Moreover, as it is difficult to collect a large-scale image distortion dataset, we propose a Cycle-Consistent Generative Adversarial Network-based data augmentation method to synthesize realistic image distortion from a small-scale multi-view multi-distortion dataset we collected. We implement the end-to-end system on four different commodity devices and evaluate its performance on two multi-view image datasets. Our evaluation demonstrates that CollabAR achieves over 85% recognition accuracy for images with severe distortions while reducing the end-to-end system latency to as low as 18.2ms.

ACKNOWLEDGMENTS

We would like to thank our anonymous reviewers and the associate editor Dr. Rui Tan for their insightful comments and guidance on making the best possible representation of our work.

REFERENCES

[1] Z. Liu, G. Lan, J. Stojkovic, Y. Zhang, C. Joe-Wong, and M. Gorlatova. 2020. CollabAR: Edge-assisted collaborative image recognition for mobile augmented reality. In *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'20)*, 301–312.

[2] P. Jain, J. Manweiler, and R. Roy Choudhury. 2015. Overlay: Practical mobile augmented reality. In *Proceedings of the ACM Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'15)*, 331–344.

[3] K. Chen, T. Li, H.-S. Kim, D. E. Culler, and R. H. Katz. 2018. MARVEL: Enabling mobile augmented reality with low energy and low latency. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys'18)*, 292–304.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS'12)*, Vol. 25, 1097–1105.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 4510–4520.

[6] L. Liu, H. Li, and M. Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom'19)*, 1–16.

[7] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. 2018. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom'18)*. 129–144.

[8] H. Ji and C. Liu. 2008. Motion blur identification from image gradients. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. 1–8.

[9] i-MARECULTURE. https://imareculture.eu. [Accessed Dec. 1, 2020].

[10] S. F. Dodge and L. J. Karam. 2018. Quality robust mixtures of deep neural networks. *IEEE Transactions on Image Processing* 27, 11 (2018), 5553–5562.

[11] W. Liu and W. Lin. 2012. Additive white gaussian noise level estimation in SVD domain for images. *IEEE Transactions on Image Processing* 22, 3 (2012) 872–883.

[12] W. Zhang, B. Han, P. Hui, V. Gopalakrishnan, E. Zavesky, and F. Qian. 2018. CARS: Collaborative augmented reality for socialization. In *Proceedings of the ACM International Workshop on Mobile Computing Systems & Applications (HotMobile'18)*. 25–30.

[13] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*. 248–255.

[14] G. Griffin, A. Holub, and P. Perona. 2007. Caltech-256 object category dataset.

[15] S. Ghosh, R. Shet, P. Amon, A. Hutter, and A. Kaup. 2018. Robustness of deep convolutional neural networks for image degradations. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'18)*, 2916–2920.

[16] T. S. Borkar and L. J. Karam. 2019. Deepcorrect: Correcting DNN models against image distortions. *IEEE Transactions on Image Processing* 28, 12 (2019), 6022–6034.

[17] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. 2010. Adapting visual category models to new domains. In *Proceedings of European Conference on Computer Vision (ECCV'10)*. 213–226.

[18] J. Sun, W. Cao, Z. Xu, and J. Ponce. 2015. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 769–777.

[19] J. Flusser, S. Farokhi, C. Höschl, T. Suk, B. Zitová, and M. Pedone. 2015. Recognition of images degraded by gaussian blur. *IEEE Transactions on Image Processing* 25 2 (2015), 790–806.

[20] W. Zhang, B. Han, and P. Hui. 2018. Jaguar: Low latency mobile augmented reality with flexible tracking. In *Proceedings of the ACM International Conference on Multimedia (MM'18)*. 355–363.

[21] S. Shen, Y. Han, X. Wang, and Y. Wang. 2019. Computation offloading with multiple agents in edge-computing-supported IoT. *ACM Transactions on Sensor Networks* 16, 1 (2019), 1–27.

[22] P. Guo, B. Hu, R. Li, and W. Hu. 2018. Foggycache: Cross-device approximate computation reuse. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom'18)*. 19–34.

[23] HoloLens. https://www.microsoft.com/en-us/hololens. [Accessed May 20, 2021].

[24] Magic leap. https://www.magicleap.com/. [Accessed May 20, 2021].

[25] Google ARCore. https://developers.google.com/ar/. [Accessed May 20, 2021].

[26] Apple ARKit. https://developer.apple.com/documentation/arkit. [Accessed Dec. 1, 2020].

[27] X. Ran, C. Slocum, M. Gorlatova, and J. Chen. 2019. ShareAR: Communication-efficient multi-user mobile augmented reality. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets'19)*. 109–116.

[28] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner. 2018. Towards security and privacy for multi-user augmented reality: Foundations with end users. In *Proceedings of IEEE Symposium on Security and Privacy (S&P'18)*. 392–408.

[29] T. Li, N. S. Nguyen, X. Zhang, T. Wang, and B. Sheng. 2020. PROMAR: Practical reference object-based multi-user augmented reality. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM'20)*. 1359–1368.

[30] K. Apicharttrisorn, B. Balasubramanian, J. Chen, R. Sivaraj, Y.-Z. Tsai, R. Jana, S. Krishnamurthy, T. Tran, and Y. Zhou. 2020. Characterization of multi-user augmented reality over cellular networks. In *Proceedings of the Annual IEEE International Conference on Sensing, Communication, and Networking (SECON'20)*. 1–9.

[31] Share AR experience with your buddy. https://niantic.helpshift.com/a/pokemon-go/?s=buddy-pokemon&f=shared-ar-experience-with-your-buddy&p=web. [Accessed May 20, 2021].

[32] S. Stein. 2020. Snapchat's augmented reality lenses can span whole city blocks. https://www.cnet.com/news/snapchats-augmented-reality-lenses-can-span-whole-city-blocks/. [Accessed May 11, 2021].

[33] J. Holland. 2017. Can holograms shape the future of car design? https://medium.com/ford/can-holograms-shape-the-future-of-car-design-dda4fcc4f22b. [Accessed May 11, 2021].

[34] I. Maw. 2019. How Lockheed Martin is using augmented reality in aerospace manufacturing. https://www.engineering.com/story/how-lockheed-martin-is-using-augmented-reality-in-aerospace-manufacturing. [Accessed May 11, 2021].

[35] H. Verkasalo. 2009. Contextual patterns in mobile service usage. *Personal and Ubiquitous Computing* 13, 5 (2009), 331–342.

[36] Y. Li and W. Gao. 2019. DeltaVR: Achieving high-performance mobile VR dynamics through pixel reuse. In *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'19)*. 13–24.

[37] Z. Zhou. 2012. *Ensemble Methods: Foundations and Algorithms.* Chapman and Hall/CRC.

[38] S. Teerapittayanon, B. McDanel, and H.-T. Kung. 2017. Distributed deep neural networks over the cloud, the edge, and end devices. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS'17)*. 328–339.

[39] N. F. Rajani and R. J. Mooney. 2017. Stacking with auxiliary features. In *Proceedings of IJCAI*, 2634–2640.

[40] J. Chen, J. Chen, H. Chao, and M. Yang. 2018. Image blind denoising with generative adversarial network based noise modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 3155–3164.

[41] Z. Sun, M. Ozay, Y. Zhang, X. Liu, and T. Okatani. 2018. Feature quantization for defending against distortion of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 7957–7966.

[42] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'17)*. 2223–2232.

[43] M. Labbe and F. Michaud. 2013. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*29, 3 (2013), 734–745.

[44] X. Zeng, K. Cao, and M. Zhang. 2017. Mobiledeeppill: A. Small-footprint mobile deep learning system for recognizing unconstrained pill images. In *Proceedings of the ACM Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*. 56–67.

[45] H. Qiu, X. Liu, S. Rallapalli, A. J. Bency, K. Chan, R. Urgaonkar, B. Manjunath, and R. Govindan. 2018. Kestrel: Video analytics for augmented multi-camera vehicle tracking. In *Proceedings of the IEEE/ACM International Conference on Internet-of-Things Design and Implementation (IoTDI'18).* 48–59.

[46] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial networks. arXiv preprint arXiv:1406.2661, 2014.

[47] X. Zhu, Y. Liu, J. Li, T. Wan, and Z. Qin. 2018. Emotion classification with data augmentation using generative adversarial networks. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'18).* 349–360.

[48] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers. 2019. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in ct segmentation tasks. *Scientific Reports* 9, 1 (2019), 1–9.

[49] D. Engin, A. Genç, and H. Kemal Ekenel. 2018. Cycle-dehaze: Enhanced CycleGAN for single image dehazing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR'18)*, 825–833.

[50] A. Bulat, J. Yang, and G. Tzimiropoulos. 2018. To learn image super-resolution, use a GAN to learn how to do image degradation first. In *Proceedings of European Conference on Computer Vision (ECCV'18).* 185–200.

[51] S. Nah, T. Hyun Kim, and K. Mu Lee. 2017. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17).* 3883–3891.

[52] Y.-W. Tai, X. Chen, S. Kim, S. J. Kim, F. Li, J. Yang, J. Yu, Y. Matsushita, and M. S. Brown. 2013. Nonlinear camera response functions and image deblurring: Theoretical analysis and practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 10 (2013), 2498–2512.

[53] S. Bae and F. Durand. 2007. Defocus magnification. In Computer Graphics Forum. Wiley Online Library.

[54] J. Xu, L. Zhang, and D. Zhang. 2018. A trilateral weighted sparse coding scheme for real-world image denoising. In *Proceedings of European Conference on Computer Vision (ECCV'18).* 20–36.

[55] J. Anaya and A. Barbu. 2018. RENOIR: A. Dataset for real low-light image noise reduction. *Journal of Visual Communication and Image Representation* 51 (2018) 144–154.

[56] K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[57] Y. Zhou, S. Song, and N. Cheung. 2017. On classification of distorted images with deep convolutional neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'17).* 1213–1217.

[58] S. Dodge and L. Karam. 2019. Human and DNN classification performance on images with quality distortions: A comparative study. *ACM Transactions on Applied Perception* 16, 2 (2019), 1–17.

[59] A. Mittal, A. K. Moorthy, and A. C. Bovik 2012. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing* 21, 12 (2012) 4695–4708.

[60] R. Liu, Z. Li, and J. Jia. 2008. Image partial blur detection and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08).* 1–8.

[61] N. D. Narvekar and L. J. Karam. 2011. A no-reference image blur metric based on the cumulative probability of blur detection (CPBD). *IEEE Transactions on Image Processing* 20, 9 (2011), 2678–2683.

[62] BBC Civilisations AR. https://www.bbc.co.uk/taster/pilots/civilisations-ar. [Accessed May 20, 2021].

[63] QuiverVision. https://quivervision.com/. [Accessed May 20, 2021].

[64] C. Chen, Y. Miao, C. X. Lu, L. Xie, P. Blunsom, A. Markham, and N. Trigoni. 2019. Motiontransformer: Transferring neural inertial tracking between domains. *Proceedings of AAAI* 33, 1 (2019), 8009–8016.

[65] A. Mathur, A. Isopoussu, F. Kawsar, N. Berthouze, and N. D. Lane. 2019. Mic2Mic: Using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems. In *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'19).* 169–180.

[66] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. 2017. Least squares generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17).* 2794–2802.

[67] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. 2016. Learning dense correspondence via 3D-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16).* 117–126.

[68] Y. Taigman, A. Polyak, and L. Wolf. 2016. Unsupervised cross-domain image generation. arXiv:1611.02200, 2016.

[69] J. Johnson, A. Alahi, and L. Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision (ECCV'16).* 694–711.

[70] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16).* 770–778.

[71] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17).* 1125–1134.

[72] D. P. Kingma and J. Ba. 2014. Adam: A. Method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[73] A. Jain. 1989. *Fundamentals of Digital Image Processing.* Prentice Hall.

[74] C. Poynton. 2012. *Digital Video and HD: Algorithms and Interfaces.* Elsevier.
[75] X. Peng, J. Hoffman, Y. Stella, and K. Saenko. 2016. Fine-to-coarse knowledge transfer for low-res image classification. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'16).* 3683–3687.
[76] A. O. Ercan, A. E. Gamal, and L. J. Guibas. 2013. Object tracking in the presence of occlusions using multiple cameras: A sensor network approach. *ACM Transactions on Sensor Networks* 9, 2 (2013), 1–36.
[77] Google cloud anchors. https://developers.google.com/ar/develop/developer-guides/anchors.
[78] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. 2012. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp'12).* 421–430.
[79] X. Ran, C. Slocum, Y.-Z. Tsai, K. Apicharttrisorn, M. Gorlatova, and J. Chen. 2020. Multi-user augmented reality with communication efficient and spatially consistent virtual objects. In *Proceedings of the ACM International Conference on Emerging Networking EXperiments and Technologies (CoNext'20).* 386–398.
[80] Y. Lin, T. Liu, and C. Fuh. 2007. Local ensemble kernel learning for object category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07).* 1–8.
[81] M. M. Derakhshani, S. Masoudnia, A. H. Shaker, O. Mersa, M. A. Sadeghi, M. Rastegari, and B. N. Araabi. 2019. Assisted excitation of activations: A learning technique to improve object detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19).* 9201–9210.
[82] A. Heiskanen. 2020. Fologram and all brick introduce an entirely new way of building in AR. https://aec-business.com/fologram-and-all-brick-introduce-an-entirely-new-way-of-building-in-ar/. [Accessed May 11, 2021].
[83] AR-Check. https://ar-check.com/. [Accessed May 20, 2021].
[84] A. Buades, B. Coll, and J.-M. Morel. 2011. Non-local means denoising. *Image Processing on Line* 1 (2011), 208–212.
[85] M. Ye, D. Lyu, and G. Chen. 2020. Scale-iterative upscaling network for image deblurring. *IEEE Access* 8 (2020), 18 316–18 325.
[86] L. Xu, J. S. Ren, C. Liu, and J. Jia. 2014. Deep convolutional neural network for image deconvolution. In *Proceedings of NeurIPS*, vol. 27, 1790–1798.
[87] J. Su, D. V. Vargas, and K. Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.
[88] Y. Pei, Y. Huang, Q. Zou, X. Zhang, and S. Wang. 2020. Effects of image degradation and degradation removal to CNN-based image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 4 (2020), 1239–1253.
[89] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. 2018. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231, 2018.
[90] Tensorflow lite. https://www.tensorflow.org/lite. [Accessed May 20, 2021].

Received December 2020; revised May 2021; accepted May 2021