THE UNIVERSITY OF CHICAGO

HETEROGENEITY AND SCALABILITY IN MACHINE LEARNING BASED SENSING AND MONITORING SYSTEMS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
ZHUJUN XIAO

CHICAGO, ILLINOIS
AUGUST 2021

This dissertation is dedicated to

my parents who have raised me into the person I am today

my amazing husband who supports me in all things great and small

"A journey of a thousand miles starts with a single step."

— Laozi

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude and appreciation to my advisors, Heather Zheng and Ben Y. Zhao, for their continuous support and patience during my PhD study. Working with Heather and Ben was an extraordinary experience. Their immense knowledge, enthusiasm for research, and creative research ideas have guided me in every step throughout this process. Beyond academic support, they also extended a great amount of assistance in daily life. My dissertation would not have been possible without their support and guidance.

I would also like to thank Junchen Jiang for serving on my committee. I'm glad to have had the opportunity to collaborate with Junchen. His valuable advice and feedbacks have helped me successfully complete my dissertation.

I wish to extent my gratitude to all my collaborators throughout my PhD study. I enjoyed working with Yanzi Zhu, Zhijing Li, Yuxin Chen, Yuanshun Yao, Jenna Cryan, Bolun Wang, Bimal Viswanath and Zhengxu Xia. I very much appreciate all their hard work and practical suggestions. I also want to thank the undergraduate students I have worked with, Yi Hong Gordon Cheo and Angela Liu.

I studied under Bo Hu, Hui Feng, Yuedong Xu and Tao Yang at Fudan University. I also had the pleasure of working with Nina Taft and Sai Teja Peddinti at Google, Yuanchao Shu and Stefan Saroiu at Microsoft Research. I thank them for their patient help, guidance and insightful conversations.

During the past five years, I truly enjoyed being part of SANDLab. I not only learned so much from my labmates, but also enjoyed all the fun time we had together. Particularly helpful to me were Zhuolin Yang, Huiying Li and Yuxin Chen, who supported me greatly during the COVID-19 pandemic. I also would like to thank Emily Wenger, Shiliang Tang, Xinyi Zhang, Shawn Shan, Christian Cianfarani and Arjun Nitin Bhagoji, for their valuable feedbacks during my practice talks, and for all the fun time we had together.

Finally, my special thanks go to my parents, my husband and my best friend for their unconditional love and continuous encouragement.

ABSTRACT

Today, significant research efforts are spent on designing machine learning (ML) models to extract useful information from data. While ML models have shown superior performance in research studies, deploying them in practice is a challenging process. My dissertation considers the task of integrating ML models to real-world distributed systems, focusing on a number of sensing and monitoring applications. For such applications, ML deployments must overcome the dual challenges of heterogeneity and scalability. Heterogeneity refers to the need for ML models to be customized for instances of the application (e.g., a specific geographic location). Scalability refers to the challenges of training and deploying distinct models across a large number of nodes.

In this dissertation, through specific case studies of high-impact problems, I demonstrate how I address these challenges using a system approach. My research considers both physical sensing and cyber monitoring applications. In physical sensing applications, heterogeneity mainly comes from sensors' local environments, while in cyber monitoring scenarios, human factor is the key source of heterogeneity. In both types of applications, I study how heterogeneity affects the ML system design and evaluation, and propose new methods to address the dual challenge of heterogeneity and scalability.

First, for the task of spectrum monitoring in cellular networks, I present a novel system design that can efficiently train and deploy deep neural network (DNN) models on a large number of spectrum sensors. Since each sensor observes complex, heterogeneous, and time-varying spectrum data, it is hard to train and deploy accurate DNN-based spectrum sensing models at scale. My work addresses this challenge by leveraging the hierarchical network structure of cellular networks, where I build context-agnostic models for spectrum usage and apply transfer learning to minimize training cost and dataset constraints.

Then, I consider the problem of evaluating ML-based video analytic pipelines (VAPs) with heterogeneous workloads, under the scenario of vehicle detection for smart-city monitoring. The difficulty of proper VAP evaluation lies in the diverse video workloads caused by

heterogeneous camera locations. Existing VAP evaluations do not consider such heterogeneity, and as a result, produce premature or ambiguous results. My work addresses this gap by building the first VAP benchmark, which provides proper and comprehensive evaluation of VAPs by characterizing the complex dependencies of VAP performance on video content characteristics.

Next, I consider a challenging problem in the area of cyber monitoring, where ML models are applied to detect unacceptable face edits in online face images. Using a user study, we find the definition of unacceptable face edit can vary per user and application context, indicating the need for personalized protection. Meeting such diverse user needs is extremely challenging since it requires ML models to accurately recognize face edits in each photo. We address this challenge using a system approach. By integrating the system function of tracking original image copies, we successfully convert the extremely hard problem of recognizing any edit in an image into a feasible ML problem of comparing two images. The end result is an efficient photo moderation tool that allows users to define their own face edit policy and provides personalized protection accordingly.

Finally, I summarize my work and discuss future directions.

CHAPTER 1

INTRODUCTION

Machine learning (ML) techniques, particularly Deep Neural Networks (DNN), have become the de facto tool for solving many challenging research problems. From image recognition [117], automatic language translation [235] to medical diagnosis [201], ML models have produced superior results on benchmark datasets, even surpassing human performance [187, 48, 174].

Deploying these ML models in practice, however, remains a significant challenge. Many surveys have shown high failure rates of ML in real-world deployments. A survey conducted by International Data Corporation (IDC) on 2,473 organizations [86] shows that, half of ML projects fail for one in four companies. Similarly, Dimensional Research reported that 78% of AI/ML projects stall at some stage before deployment [215], while VentureBeat reported that 87% of data science projects never make it into production [36]. Clearly, there is a gap between ML in research and ML in real-world deployment.

So what causes such gap? In ML research studies, the common practice is to first train ML models on training data and then test on a another set of test data. Here the primary goal is to design the model to maximize its predictive accuracy on the test data, under the ideal system assumption that the model runs on a single node with no resource constraints. However, in real-world deployments, ML models are integrated into large-scale (distributed) systems, thus facing system-level challenges such as cost constraints, heterogeneity, robustness, potential bias, transparency and scalability. These challenges are rarely considered by ML research studies at ML modeling stage, but must be addressed during real-world deployment.

1.1 System-level Challenges in Sensing and Monitoring Systems

My dissertation considers the task of integrating ML models to real-world distributed systems, focusing on a number of **sensing and monitoring applications**. In such applications, data is first collected from a large number of distributed nodes (e.g., traffic cameras), and then fed into ML models to detect certain events or objects (e.g., cars). Our research studies identify two system-level challenges that exist widely in ML-based sensing and monitoring systems, and yet they are not well studied.

- **Heterogeneity.** Heterogeneity refers to the need for ML models to be customized for instances of the application. Heterogeneity may apply to heterogeneous hardware devices, network conditions, geographic locations and even user preferences. Without considering heterogeneity, ML models may fail in new environments, resulting in unstable and unpredictable system performance.

  My research considers both physical sensing and cyber monitoring applications. In physical sensing applications, heterogeneity mainly comes from sensors' local environments, while in cyber monitoring scenarios, human factor is the key source of heterogeneity.
- **Scalability.** In traditional distributed systems, a system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity [199]. The same challenge exists in ML based sensing and monitoring systems since ML models are often trained and deployed across a large number of nodes, e.g., deploying car detection models on traffic cameras all over the city.

1.2 Overview of My Research

In this dissertation, through specific case studies of high-impact problems, I demonstrate how I address these challenges using a system approach. We believe there are two critical steps when integrating ML into sensing and monitoring systems (shown in Figure 1.1),

ML model design/evaluation and *ML system design/evaluation*. While previous ML studies mainly focus on proposing better ML models, my goal is to propose new system designs and evaluation platforms to address the dual challenges of heterogeneity and scalability.

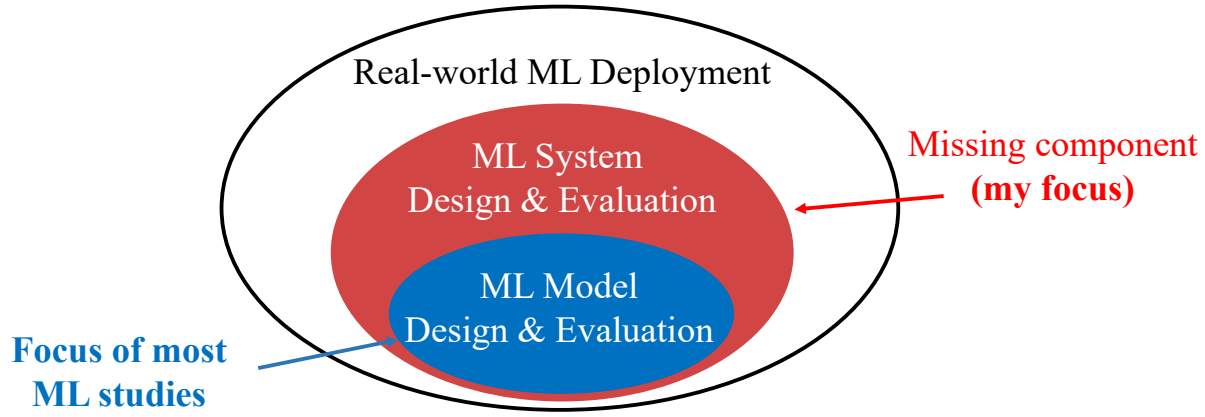


Figure 1.1: My work proposes new system designs and evaluation platforms that address the dual challenges of heterogeneity and scalability in ML based sensing and monitoring systems.

First, for the task of spectrum monitoring in cellular networks, I present a novel system design that can efficiently train and deploy DNN models on a large number of spectrum sensors with heterogeneous context.

Next, I consider the problem of evaluating ML-based video analytic pipelines (VAPs) with heterogeneous workloads, under the scenario of vehicle detection for smart-city monitoring. This project shows that heterogeneity and scalability affect not only the system design, but also the system evaluation.

Finally, I propose a ML-enabled cyber monitoring system, where ML models are applied to detect unacceptable face edits in online images. This project demonstrates how heterogeneous user preferences affect the design and evaluation of ML systems.

In the following, I will briefly introduce those projects.

1.3 Scaling Deep Learning Models for Spectrum Monitoring

Spectrum management in cellular networks is a challenging task that will only increase in difficulty as complexity grows in hardware, configurations, and new access technology (e.g. LTE for IoT devices). Wireless providers need robust and flexible tools to monitor and detect faults and misbehavior in physical spectrum usage, and to deploy them at scale. We explore the design of such a system by building DNN models to capture spectrum usage patterns and use them as baselines to detect spectrum usage anomalies resulting from faults and misuse.

Using detailed LTE spectrum measurements, we show that the key challenge facing this system is the complex, heterogeneous, and time-varying spectrum observations seen by each observer, caused by inherent radio propagation dynamics and diverse mobility patterns. Such complexity makes it hard to train and deploy accurate DNN spectrum anomaly detection models at scale.

We address this challenge by building context-agnostic models for spectrum usage and applying transfer learning to minimize training time and dataset constraints. Specifically, our solution includes two steps:

- Within each LTE cell, it is feasible to build a single, context-free DNN model that accurately models normal spectrum usage pattern for the task of anomaly detection. Our DNN model does not use supervised learning to classify an event as normal or anomalous. Instead, we train a DNN model on sequences of spectrum measurements, and it recognizes events as anomalies when they deviate significantly from events expected or predicted by the model. More importantly, our model runs on both mobile and static observers to detect spectrum anomalies on the fly without any modifications.
- Across LTE cells, the DNN model trained for a given LTE cell is not directly reusable at the other cells, but can be used to quickly train their models through transfer learning. Only a small amount of local spectrum measurements at the target cell is required. Our results

show using transfer learning instead of training from scratch reduces required training data by a factor of 288. Since different LTE bands (frequency carrier, downlink or uplink) display different spectrum patterns, they require different DNN models customized for that band. The same transfer learning method can be applied to quickly train the model for a frequency band using existing models for other bands as a starting point.

The above design has two key features. First, the spectrum DNN model is context-agnostic and can be easily deployed on a wide range of spectrum observers, static and mobile, and adapted using a minimal amount of local spectrum measurements. Each observer does not need to store a large number of models for each context, or switch to a new model whenever it moves. Instead, it runs the same DNN model regardless of its context, and only needs to switch to a new model when moving into a different cell. Second, the anomaly detection is general in that it avoids cellular-specific knowledge and can detect any events that affect spectrum usage.

1.4 Evaluating Video Analytic Pipelines with Heterogenous Workloads

Deep learning based video analytics is becoming the solution to many safety and management tasks. Its wide deployment, however, must first address the tension between inference accuracy and resource (compute/network) cost. This has led to the development of video analytics pipelines (VAPs), which reduce resource cost by combining DNN compression/speedup techniques with video processing heuristics.

Today, VAPs are evaluated using some corpus of past video samples that represent the target scenario(s). Our measurement study on existing VAPs, however, shows that today's methods for evaluating VAPs are incomplete, often producing premature conclusions or ambiguous results. This is because each VAP's performance varies substantially in different environments (even under the same scenario). We find that, the challenge of proper VAP

evaluation lies in the diverse video content characteristics caused by environmental heterogeneity.

Therefore, we argue that accurate VAP evaluation must first characterize the complex interaction between VAP's performance and video characteristics, which we refer to as VAP performance clarity. We then design and implement Yoda, the first VAP benchmark to achieve performance clarity. Specifically, we propose to characterize VAP performance using a carefully curated set of videos that serve to evaluate different aspects of VAPs. Our design is based on the observation that each VAP is inherently modular and can be broken into a set of "global" primitives. Each primitive leverages a distinct set of video processing heuristics to optimize the accuracy/cost tradeoff, and thus can be profiled independently (against its associated video features) and then (re)assembled to profile full VAPs. This modular structure allows us to efficiently profile each full VAP by combining its corresponding primitive-specific profiles.

We show that Yoda substantially improves VAP evaluations by (1) providing a comprehensive, transparent assessment of VAP performance and its dependencies on video characteristics; (2) explicitly identifying fine-grained VAP behaviors that were previously hidden by large performance variance; and (3) revealing strengths/weaknesses among different VAPs and new design opportunities.

1.5 Enabling Personalized Face Edit Protection

Today, popular photo sharing sites and social networks allow users to edit and alter their faces in images for a variety of uses, from subtle touchups to dramatic alterations like aging and face swaps. The ubiquity of these facial editing tools pose a significant risk to users, whose personal photos may be downloaded, their facial features manipulated or distorted, and the results reposted for purposes of harassment or bullying. There is a clear need for "photo moderation tools" that protect users against undesired or unacceptable edits of their online images.

However, using a user study, we find the definition of unacceptable face edit can vary per user and application context, indicating the need for personalized protection. Therefore, our goal is to build a system that provides personalized protection from unacceptable photo manipulation on photo sharing network services. Unlike existing approaches that try to detect *any* manipulations in images, our proposed system, *Aletheia*, determines whether modifications made in a newly shared image are acceptable based on the owner's personalized preferences.

Providing personalized protection for different users is extremely challenging since it requires ML models to accurately recognize face edits in each photo. We address this challenge using a system approach. By integrating the system function of tracking original image copies, we successfully convert the extremely hard problem of recognizing any edit in an image into a feasible ML problem of comparing two images. Specifically, our system includes a policy manager that allows users to define their own face edit policy, and an image inspector that not only determines whether an incoming photo is original or edited, but also locates the original copy if it is an edited photo. Our system also applies a set of semantic face attribute extractors to recognize edits in a photo by comparing its face attributes with those of its original copy.

Using a set of experiments, we show that *Aletheia* (1) can effectively flag image edits that violate user policies; (2) is accurate and fast at scale; (3) can effectively recognize in-the-wild face edits submitted by 8 participants (without any instruction on tools and formats); (4) provides a feeling of protection to real users.

1.6 Structure

The structure of this dissertation is as follows:

- In Chapter 2, I present a novel system design for distributed spectrum monitoring in wide-area cellular networks, where DNN models are distributed to a large number of spectrum

sensors that are either static or mobile. Leveraging the hierarchical network structure of cellular networks, our system could efficiently train and deploy DNN models for complex, heterogeneous, and time-varying spectrum data seen by each sensor.

- In Chapter 3, I study how to evaluate ML-based video analytic pipelines. I build Yoda, the first benchmark to comprehensively evaluate VAP on heterogeneous video contents by characterizing the complex dependencies of VAP's system performance on video content characteristics.
- In Chapter 4, I consider a cyber sensing scenario, where ML models are applied to detect unacceptable edits in online face images. Our proposed system allows each user to define their own face edit policy, and provides personalized detection based on users' preferences.
- In Chapter 5, I summarize my work and discuss future direction.

CHAPTER 2

SCALING DEEP LEARNING MODELS FOR SPECTRUM MONITORING

In this chapter, I study the problem of how to address heterogeneity and scalability with **efficient system designs**, using ML-based spectrum monitoring system as a case study. We propose a novel system design for a distributed spectrum monitoring framework in wide-area cellular networks, where DNN models are deployed to detect spectrum usage anomalies on the fly. In this system, DNN models are distributed to a large scale of static and mobile spectrum observers located throughout the network. Using detailed LTE spectrum measurements, we show that the key challenge is the complex, heterogeneous, and time-varying spectrum data seen by each sensor. Such complexity makes it hard to train and deploy accurate DNN spectrum anomaly detection models at scale.

Inspired by the hierarchical network structure of cellular networks, we address this challenge by building context-free DNN models within a cellular cell and applying transfer learning to minimize training time and dataset constraints across cells.

2.1 Introduction

Cellular providers spend billions of dollars acquiring radio spectrum for network capacity and coverage. Yet spectrum management, specifically detection of faults from spectrum interference, remains a costly and ad hoc process, often involving manual diagnosis following customer complaints and operational failure logs. What makes detection hard is that interference can come from a variety of complex sources at any physical location, ranging from intentional spectrum misuse and misconfigured transmitters to RF leakage from cable plants and connectors. For example, interference from a misconfigured amplifier led to persistent quality-of-service issues for a tier 1 service provider [239].

These problems will grow in severity and scale in the near future. Advances in both re-

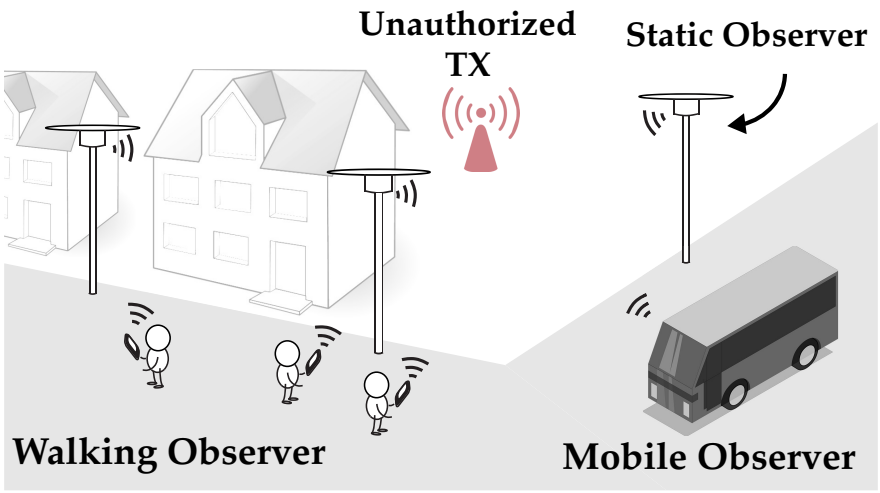


Figure 2.1: Spectrum anomaly detection by multiple observers.

configurable hardware and spectrum usage policies make it easy to misuse spectrum without authorization. There is already evidence of these misuse attacks in China, where growth in unauthorized transmissions has prompted new initiatives to outlaw spectrum misuse [141]. Furthermore, cellular interfaces for IoT are coming, optimized for the network and energy needs of IoT devices. Adoption of these interfaces has the side effect of increasing security risks for LTE and nearby spectrum bands. A compromised device working on behalf of an attacker can perform jamming or denial of service attacks on cellular bands.

Clearly, cellular networks need robust and flexible tools to detect faults and misbehavior in spectrum usage, which we hereby refer to as *spectrum anomalies*. Despite open calls for automated management tools by the 3GPP standards body, current proposals are still limited to simplistic fail-stop fault models, and only on faults *within* the LTE infrastructure [42, 198, 246]. Cellular carriers often evaluate physical spectrum usage by wardriving with specialized devices, and these activities are severely limited by high human and equipment costs [88].

Instead, we believe that cellular networks require *general* solutions capable of detecting a range of radio spectrum anomalies, from *transmissions at unexpected power levels*, to *interference from misconfigured devices* and *unauthorized transmitters*. Anomalies can appear anywhere in the physical network, and their detection requires a large-scale, distributed spectrum monitoring system.

In this paper, we explore the design of a general, scalable system for detecting spectrum anomalies in wide-area LTE networks. As shown in Figure 2.1, the system consists of two components: (1) a scalable, distributed spectrum monitoring system that measures physical spectrum usage using both static and mobile observers[1] distributed across the network, and (2) a general anomaly detection system that builds DNN models using these measurements, and runs them at each observer as baselines to detect spectrum usage anomalies. Our work builds on multiple prior efforts: some of which examined the feasibility of distributed spectrum monitoring using commodity devices [193, 64], while others validated the benefits of DNN-based spectrum anomaly detection using a single static observer [200, 102].

Despite significant progress, a large gap remains between current proposals and a feasible system for cellular networks. More specifically, the unanswered question is *how can context-sensitive DNN models scale to detect spectrum anomalies at geographically distributed locations*? The general understanding today is that wireless measurements, *e.g.* spectrum usage, depend on the context of the observer, *i.e.* time, location, and mobility status. Ideally, each observer should run a model tailored to the current context. This renders the system impractical, given the amount of training overhead and run-time complexity required[2]. A practical alternative is to explore a unified, context-free model for all observers, and whether such models can be trained, deployed and validated. But will the elimination of "context-awareness" from the DNN model design degrade the accuracy of spectrum anomaly detection?

We answer these questions through an empirical study on LTE networks, using detailed spectrum measurements across multiple LTE bands and cells (totalling 20TB of measurement data). Our efforts lead to three key findings.

- Within each LTE cell, it is feasible to build a single, context-free DNN model that ac-

1. Spectrum monitoring requires both static and mobile observers to enforce coverage and scale. We assume that these observers are recruited by the carriers to perform spectrum monitoring and anomaly detection, and are well-behaved. This simplification allows us to focus on the problem of scaling DNN models for anomaly detection.

2. It is impractical to assume that the system must build models for each physical location, and that each observer must change its DNN model whenever it moves.

curately models normal spectrum usage pattern for the task of anomaly detection. Our DNN model does not use supervised learning to classify an event as normal or anomalous. Instead, we train a long-short term memory (LSTM)[3] model on sequences of spectrum measurements, and it recognizes events as anomalies when they deviate significantly from events expected or predicted by the model. More importantly, our model runs on both mobile and static observers to detect spectrum anomalies on the fly without any modifications. This puts a hard limit on the training overhead and run-time complexity. We also show that deep autoencoder, another DNN model, can be designed to offer the same properties.

- Across LTE cells, the DNN model trained for a given LTE cell is not directly reusable at the other cells, but can be used to quickly train their models through *transfer learning*[4]. Only a small amount of local spectrum measurements at the target cell is required. Our results show using transfer learning instead of training from scratch reduces required training data by a factor of 288.
- Since different LTE bands (frequency carrier, downlink or uplink) display different spectrum patterns, they require different DNN models customized for that band. The same transfer learning method can be applied to quickly train the model for a frequency band using existing models for other bands as a starting point.

Together, these findings demonstrate the feasibility of deploying a practical model for LTE spectrum anomaly detection on top of the distributed spectrum monitoring system. Specifically, the system first trains a general DNN model for normal spectrum usage, *i.e.* teacher model, using past spectrum measurements from trusted observers. It then distributes this teacher model to each individual LTE cell's basestation, who uses a small amount of local spectrum measurements (contributed by trusted observers in the cell) to quickly calibrate

3. An LSTM model is a recurrent neural network that is particularly adept at modeling and predicting temporal sequences.

4. In transfer learning, we start with a highly-tuned model, and customize it for a given context by further training it with limited training data from that context.

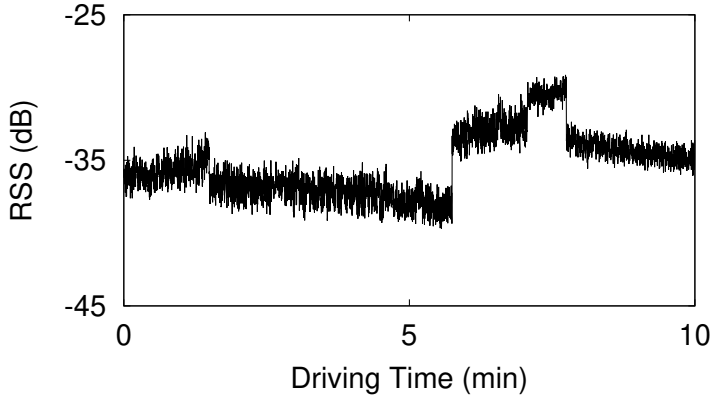


Figure 2.2: RSS varies largely over a 10-minute monitoring window, for a mobile observer.

the model, and distributes this cell-specific model to all the observers in the cell. Each observer runs the same DNN model regardless of its context, and only needs to switch to a new model when it moves into a different cell.

The above design has two key features. First, the spectrum DNN model is context-free and can be easily deployed on a wide range of spectrum observers, static and mobile, and adapted using a minimal amount of local spectrum measurements. Second, the anomaly detection is general in that it avoids cellular-specific knowledge and can detect any events that affect spectrum usage.

2.2 Preliminaries

To provide context for our later discussions, we present in this section the spectrum measurement dataset used in our empirical study, and our initial analysis on patterns in today's LTE spectrum usage. We also present existing models for spectrum anomaly detection, and evaluate their performance using our spectrum measurements in the presence of spectrum anomalies.

2.2.1 Analysis of LTE Spectrum Usage

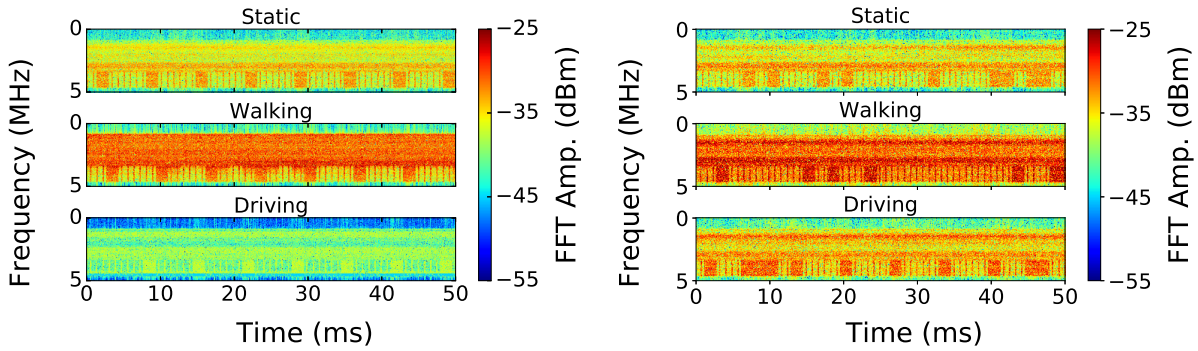
Our Dataset. We performed signal measurements on three major LTE carriers in the US, including three FDD[5] downlink bands of AT&T (880MHz), T-Mobile (729MHz), and Verizon (749MHz), and one FDD uplink band of AT&T (830 MHz). We used USRP N210 devices to capture 5 MHz spectrum within each LTE band.

While prior works on spectrum misuse detection [69, 137, 164, 200] only considered static observers, we performed measurements on LTE spectrum usage using both static and mobile observers (walking, driving). Our measurements were performed at two areas: a large university campus and an urban downtown area, separated by a distance of 8 miles. For each area, we verified that the observers were in the same LTE cell during the measurement period and the measurement range is within 1 mile.

Our measurements were performed between January and March 2018, and repeated in June 2018 to examine potential temporal variations. Specifically, we set up three static observers (well separated) in the university campus and collected measurements continuously for 7 days, and two static, well-separated observers in the downtown area for 3 days of continuous measurements. Walking and driving experiments were done for 45 min per day for 8 days. In total, the dataset contains more than 20 TB of signal data.

Spectrum Usage $\neq$ RSS. Many prior works [69, 137, 164, 71, 263] have used measured received signal strength (RSS) as the base for spectrum anomaly detection, where an anomaly occurs if the current RSS deviates from a pre-defined range. Our measurement shows that RSS is not a viable base for mobile observers since it changes significantly and unpredictably over time. Figure 2.2 shows a random segment of RSS collected by a mobile observer over 10 minutes. Here the sudden rise of RSS values can be the result of multipath fading or interference from an unauthorized transmitter in proximity, which are indistinguishable using RSS data.

5. LTE supports both frequency division duplexing (FDD) and time division duplexing (TDD). Since most US cellular carriers use FDD, we focus on LTE FDD bands.



(a) Initial spectrograms.

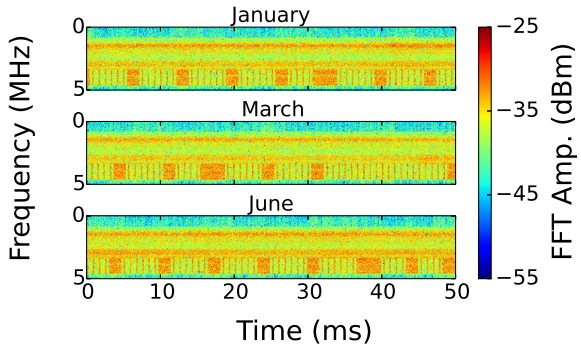
(b) Spectrograms after 10 minutes.

Figure 2.3: Spectrograms captured by spectrum observers under different contexts, based on the measurements on the 880MHz downlink band.

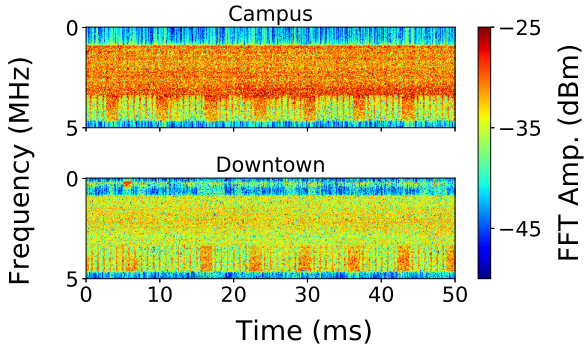
Time-Frequency Patterns of Spectrum Usage. Instead, we analyze spectrum usage in terms of the time-frequency spectrogram of the received signal. Spectrograms capture fine-grained signal amplitude over time at sub-frequencies, and are widely used for spectrum analysis. In absense of any anomaly, Figure 2.3 plots a spectrogram segment of $50ms$ at three observers (static, walking, driving) and another $50ms$ segment at each of the same observers about 10 minutes later. Despite the large difference in signal amplitude across users and time, we can observe visible temporal patterns from all six segments, in the form of bursts of high-power transmission along the time dimension.

We study these patterns in detail and arrive at two key observation. *First*, the pattern is complex, especially in the temporal domain. Periodicity analysis shows that signal fluctuation peaks reside at 1200Hz, 160Hz and 60Hz, indicating that the key periodic pattern occurs every $0.21ms$, $1.6ms$, and $6ms$. More frequencies of transmission bursts exist in addition to these main peaks, indicating more fine-grained temporal patterns beneath the obvious bursty patterns we observed. *Second*, the *short-term* patterns of the spectrum usage share some general shape. Carefully formed, they could serve as reliable "fingerprints" of normal spectrum usage.

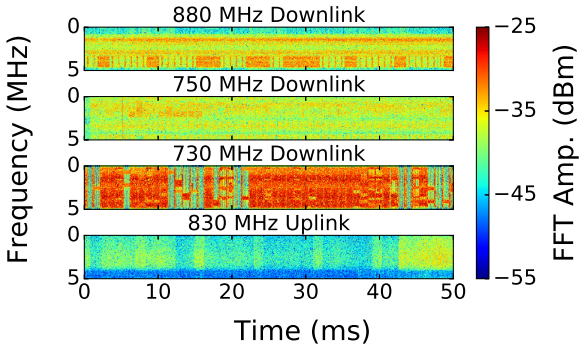
Spectrum Patterns across Time, Cells and Bands. We visually compare the spectrum usage patterns observed across time, LTE cells, and LTE bands. We found that



(a) Across different time periods (880MHz).



(b) Across different cells (880MHz).



(c) Across different bands.

Figure 2.4: Comparing spectrogram samples across time, LTE cells, and LTE bands.

the short-term usage patterns are fairly consistent over time (by comparing observations in January-March, and June), differ slightly across cells (campus vs. downtown), but show more visible differences across LTE bands. Figure 2.4 illustrates this using a set of spectrogram samples. Periodicity analysis also confirms these observations.

2.2.2 Models for Spectrum Anomaly Detection

The above analysis suggests that it is feasible to build general spectrum anomaly detection by modeling the time-frequency patterns of normal LTE spectrum usage. The hypothesis is that the presence of a spectrum anomaly will produce visible changes to the patterns extracted from the measured signals, which trigger the detection of the anomaly. This type of anomaly detection prioritizes generality: training/building the model using normal spectrum usage

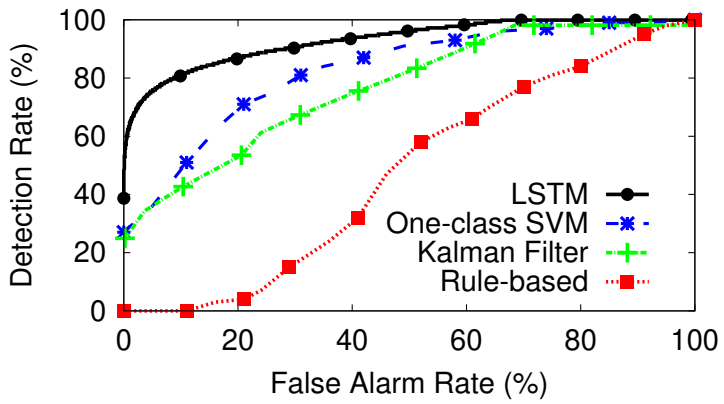


Figure 2.5: Anomaly detection performance of non-DNN (one-class SVM, Kalman filter, Rule-based) and DNN (LSTM) models, based on measurements at the DL 880MHz band.

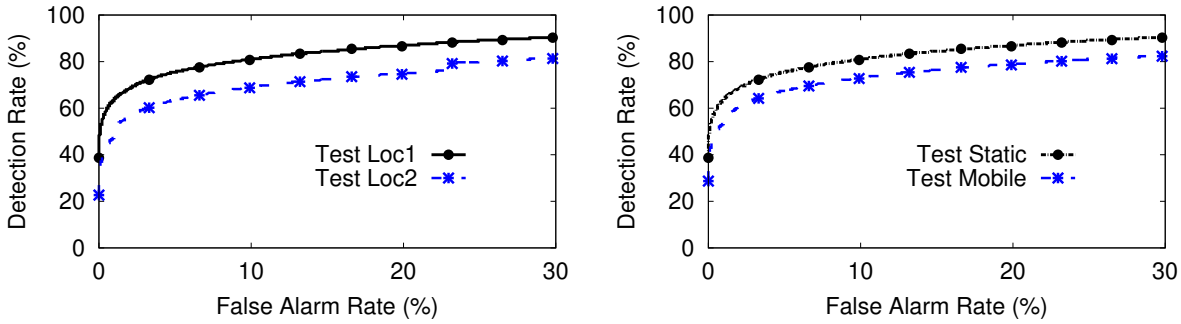
without requiring any knowledge or labeling on anomaly instances.

There are multiple existing approaches of modeling spectrum usage patterns from the spectrogram, ranging from the classical methods of Kalman filter, one-class SVM [164, 181] to the recent proposal of neural network models (LSTM [200] and deep autoencoder [102], details presented later in §2.4). Existing works on spectrum anomaly detection only considered static observers.

We implemented and evaluated these approaches using our LTE measurements. A small portion of our measurements were conducted when anomalies were present. More details on these anomalies are described later in §2.6.1. For all the experiments, the observers were placed within $50m$ of the misuse transmitter.

Evaluation at Static Observers. For all the approaches (Kalman filter, one-class SVM, LSTM, deep autoencoder), we use as the model input the signal spectrogram over $256ms$ (we have tested other segment lengths between $32ms$ and $256ms$ and found that they do not change the conclusion). We train the models using past spectrum measurements in absence of anomalies at each static observer. We also included a RSS-based method that uses a threshold to detect the presence of anomaly (Rule-based). The performance of the LSTM and deep autoencoder models are similar so we only included the LSTM result for brevity.

Figure 2.5 plots the results in terms of anomaly detection rate vs. false alarm rate. The



(a) Test across locations.

(b) Test cross context.

Figure 2.6: Anomaly detection performance when (re)using a LSTM model customized for a static observer at location 1. (a) running the model at location 1 and location 2. (b) running the model at location 1 and a mobile observer near location 1 (all based on measurements at the DL 880MHz band).

results are similar across the four LTE bands so we only show the result in the 880MHz downlink band for brevity. We see that the DNN model (LSTM in this case) largely outperforms the three non-DNN alternatives, achieving a high detection rate and a low false alarm rate at the same time. This finding aligns with that of recent works [200, 102].

We believe the reason behind the above conclusion is that LTE spectrum usage patterns are complex, making it hard to perform proper feature engineering. Additional complexity comes from possible correlations between feature dimensions (different sub-frequencies of the transmission). All of these make it difficult for traditional methods (e.g., , one-class SVM) to model the spectrum usage. After identifying good features that capture key spectrum patterns, one-class SVM may be able to achieve good performance. But doing so requires deep understanding of the data and much heavier efforts on feature engineering. The complexity further exacerbates when building the models for mobile observers.

2.3 Key Concept: Making Spectrum Models Context-free

Our empirical analysis validates the observation of prior works [200, 102], where each static observer individually trains DNN models to detect spectrum anomalies. But *can such a*

context-specific model be deployed on a large-scale distributed monitoring system, where the spectrum observers are distributed across a wide area, and can be mobile or static? Next, we answer this question empirically, testing whether models trained by a given static observer can be "reused" by another static observer at a different location and another mobile observer. Again we observe a consistent trend across all four LTE bands, and show the result for the 880MHz band for brevity.

Test I: Reusing Models across Locations. Using our LTE measurements at three static observers (in the same LTE cell), we apply the same approach of [200, 102] to train, for each observer, the corresponding DNN models (LSTM and deep autoencoder). We then run the models customized for one observer at the other two observers both with and without the presence of spectrum anomalies. We considered a range of spectrum anomalies in the form of unauthorized transmissions used in §2.2.2.

Our results show that when reusing a spectrum LSTM model at a different location, the model is less accurate in capturing normal spectrum patterns. Thus the anomaly detection rate drops considerably (Figure 2.6(a)). The same applies to the autoencoder model.

Test II: Reusing Models at Mobile Observers. We also experimented with "reusing" models trained for a static observer at a mobile observer (walking at 3mph). Both observers were in close proximity (to reduce the impact of location change). Results in Figure 2.6(b) show a similar trend of performance degradation.

Our Focus: Scaling the DNN Models. Together, these experiments suggest that since wireless measurements depend on the context of the observer, *i.e.* time, location, and mobility status, ideally each observer should run a DNN model tailored to the current context.

Unfortunately, this is impractical under our targeted scenario because the system must build models for each physical location in the network and user context, and each observer must change its DNN model whenever it moves. Such requirement leads to significant training overhead and run-time complexity.

This motivates us to explore a practical alternative: building a *unified* model for all the observers, with the goal of prioritizing scale and ease of deployment, minimizing training overhead, and maintaining reasonable accuracy. In the following sections, we tackle this new problem in two steps: first designing a single context-free DNN model for anomaly detection in a single LTE cell (§2.4), then extending the single cell model to train models for many other LTE cells and bands using transfer learning (§2.5).

2.4 A Single Model Per LTE Cell

In this section, we focus on designing a single DNN model for a single LTE cell, which will be deployed on all the observers in the cell without any modification. Our hypothesis is that within a cell, the normal downlink spectrum usage seen by each observer comes from the same basestation, thus it is possible to train the DNN model to capture a *unified* form of spectrum pattern that is context-free, *i.e.* does not depend on mobility pattern and precise location within the cell. For uplink, each observer sees aggregated transmissions from many LTE users, and the normal spectrum usage could also display context-free patterns. Thus our goal is to design models to automatically discover these context-free patterns, and to validate whether they are sufficient for anomaly detection.

Our study considers two DNN models, LSTM and deep autoencoder. Both are known for capturing complex, temporal patterns in the target data that can be difficult to detect with simpler models [121, 200, 102]. In the following, we start with a brief introduction of the two models, and then describe the steps taken to build and train a context-free version of these models using our spectrum data. We also evaluate the models at both static and mobile observers, in terms of how they model and predict future spectrum usage. Later in §2.6 we evaluate the corresponding anomaly detection systems.

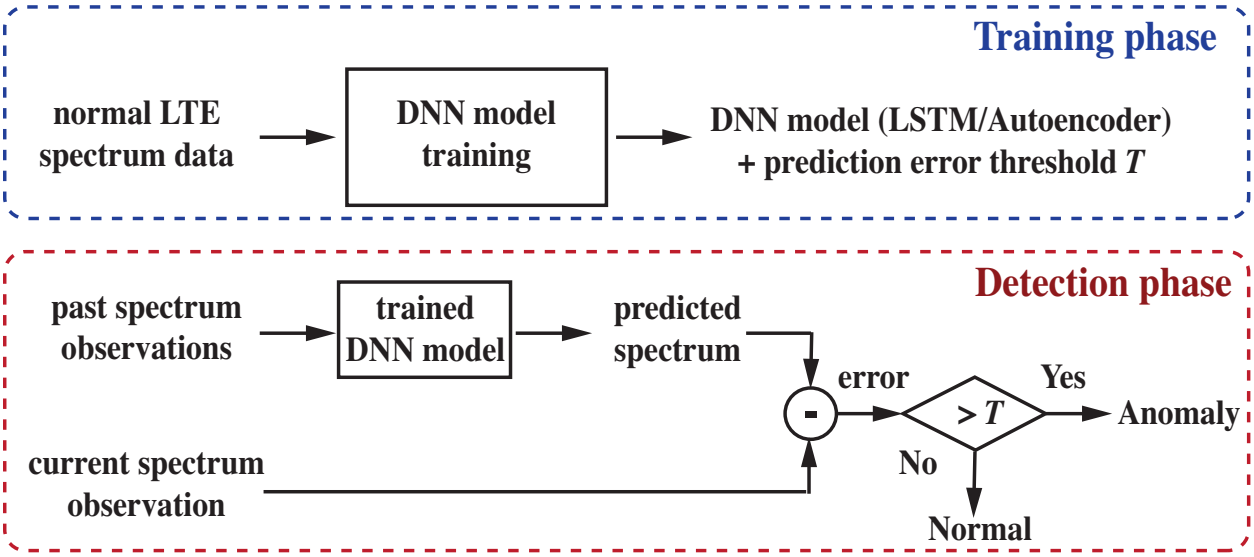


Figure 2.7: Anomaly detection using DNN models of spectrum usage.

2.4.1 Background: LSTM and Deep Autoencoder

LSTM is a special type of Recurrent neural network (RNN), well-known for its capability of capturing comprehensive and intricate patterns embedded in sequential data. RNNs are networks with loops, allowing information to persist. As shown in Figure 2.8, a RNN unit consists of multiple copies of the same network, each passing a message to a successor. Each LSTM model maintains an internal state in each RNN unit, and often consists of multiple stacked layers, forming an architecture similar to feed-forward neural network. This allows learning of complex relationships in sequential data. Normally another fully connected layer is attached at the end of the model for classification or prediction. Details on LSTM can be found in [121].

A stacked (or deep) autoencoder (details in [102]) is a DNN model designed to learn efficient data representation (or encoding) in an unsupervised way. It learns to compress the data from the input layer into representations, and then reconstructs the original data using the representations at the output layer. This process forces the autoencoder to extract the most useful features of the data.

Anomaly Detection. The above predictive models enable anomaly detection without

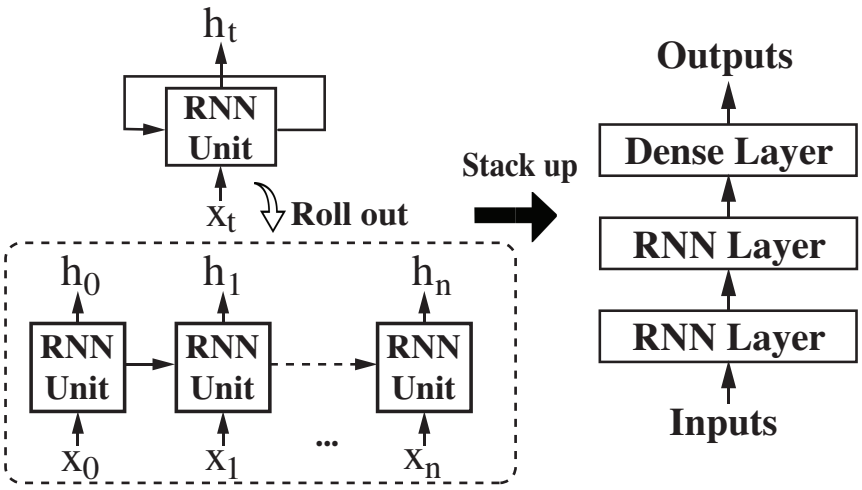


Figure 2.8: RNN unit and stacked RNN network.

prior knowledge of anomaly. The intuition is that since each model is trained using normal spectrum data, it *cannot* accurately predict data that contains anomaly, leading to large model prediction errors that trigger the anomaly detection. Figure 2.8 plots the anomaly detection process. We first train the model using spectrum observations in absense of anomaly, where given past values, the model predicts the next few values in the sequential data. Next, given a present spectrum observation, we use the model (and past observations) to predict the present spectrogram, and compare it to the observed spectrogram. If the prediction error is larger than a threshold (details in §2.6), an anomaly is present.

Finding Clean Training Data. Ideally the model should be trained with measurements in absense of anomaly, which are hard to verify in practice. Fortunately, several works have confirmed that RNN, particularly LSTM can tolerate limited presence of anomalies in the training data without affecting anomaly detection performance [111, 81]. Under our target scenario, the system can choose training data from trusted observers who did not observe notable cellular service degradations at the time of data collection, thus the mass majority of the training data are in absense of anomaly.

2.4.2 Unified Models of Spectrum Usage

We now describe the unique steps we take to build and train a per-cell, unified predictive model on spectrum usage. While our description below is for LSTM, we apply the same process to build and train the deep autoencoder model.

Input to LSTM. We feed the raw spectrogram of the wireless signal into the LSTM model. Our intuition is that sufficiently powerful LSTMs operating on raw signals can extract meaningful patterns, while an alternative LSTM operating on aggregate statistics is vulnerable to poor choice of statistics, and could miss valuable dimensions of the data.

Given our LTE measurements, we configure the LSTM model to use $x = 25.6$ms of measured signal as input to predict the next $y = 6.4$ms. We chose these parameters because our spectrum analysis in §2.2 shows that the longest periodic pattern occurs at 6ms, thus a target frame of $y = 6.4$ms should be sufficiently large to include all the key patterns. We also experimented with other x values and found that 25.6ms offers the best performance under our target scenarios. We leave the optimization of x and y to future work.

Making the Model Context-free. Our predictive model is context-free, so that it can be deployed on all the observers in the current cell, regardless of their physical location and mobility status. We take two steps to make the predictive model context-free.

First, we apply *linear transformation* to expose the intrinsic spectrum usage patterns. As mentioned earlier, the input signal data displays a large variance across observing locations, which is an inherent property of radio propagation. Such high variance can cause LSTM (and autoencoder) to miss detailed temporal patterns and correlations among sub-frequencies, but focus solely on absolute power values. To expose these intrinsic spectrum patterns, we apply linear transformation, i.e., , mean-centering and scaling, to the input FFT amplitudes, and filter out input sequences that only contain noise (no signal at all). As a result, each input sequence to the LTSM model now has a zero mean and a variance of 1. This transformation is similar to the idea of *contrast stretching*, a common pre-processing technique in computer

vision that exposes patterns by transforming pixel intensities to increase contrast [45].

Second, we use as training data a mixture of spectrum measurements collected by both static and mobile observers within the cell. Compared to context-specific models, this certainly minimizes the model training time and data requirements. One concern is that mixing training data from many sources can potentially increase the ambiguity between normal data and anomalies. For example, the normal spectrum usage seen at observer A could be similar to the anomalous spectrum usage seen by another observer B. When the training data includes those measurements from A, the trained model could misclassify B's observation of an anomaly as normal.

We took a detailed look at our measurement data (with anomalies), but did not identify any of such events. While our anomaly instances are limited in scale, this result suggests that the probability of such events is low in practice. An advanced attacker could form its misuse signals to imitate normal spectrum usage, but this is challenging since the observers will observe the aggregated signals from the attacker and the legitimate LTE transmitters.

Model Training. To train these models, we divide our per-cell LTE measurement data into two portions: one used for training, and one for testing. Both datasets contain measurements collected by static, walking and driving observers. The observers in the testing dataset do not appear in the training dataset. Overall, for each cell, the ratio of the training data volume and the test data volume is 2.5:1. Each model is trained to minimize the Root Mean Square Error (RMSE) between the predicted signal and the ground-truth signal (after transformation) in the training dataset.

Evaluation: Model Prediction Error

We evaluate how well the DNN models predict the spectrum usage of the immediate future, so that they can quickly detect anomalies that disturb the spectrum usage pattern. As an illustrative example, Figure 2.9 plots the actual spectrogram (on a randomly chosen 50 ms segment) and the output of the LSTM prediction model (after reversing the linear

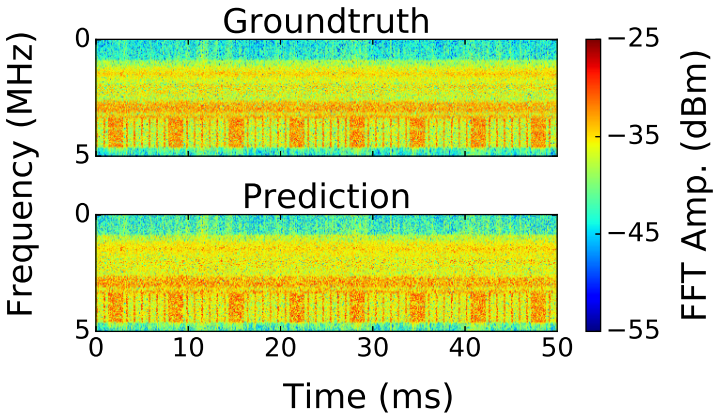


Figure 2.9: Spectrograms of actual and LSTM predicted LTE signal.

transformation). To reconstruct the 50ms segment from the prediction result, we cascade 8 segments of 6.4ms prediction results, each predicted from previously observed 25.6ms measurements. We see that the LSTM model is able to recover the key patterns in spectrum usage across sub-frequencies and time.

Evaluation Metric: Spectrogram Prediction Error (dB). We evaluate each model by the difference between the true signal spectrogram and the model prediction. Specifically, we calculate the RMSE between the true FFT amplitude (dBm) across sub-frequencies of the LTE band and the LSTM model prediction values after being inverse-transformed back to FFT amplitude (dBm). In a nutshell, the RMSE value approximates the amplitude spectrogram error in a single dB value. We refer to this metric as the prediction error (dB). Because our test data has many observers, we will present the mean and standard deviation of the prediction error across all the observations in the test dataset.

The prediction error directly links to the accuracy of anomaly detection. The smaller the prediction error is in absence of anomaly, the better the predictive model is and the higher accuracy the model has during anomaly detection. We evaluate the anomaly detection performance later in §2.6, which yields consistent results.

Unified vs. Customized Models. We evaluate our unified models by comparing them to models customized to individual observer's context. The results of LSTM and autoencoder

are similar to each other: the prediction error of autoencoder is 3-8% higher than that of LSTM. We only show the LSTM results for brevity.

Table 2.1(a) shows the mean and standard deviation of the prediction error (dB) of our unified model and those of the models customized to three individual locations. The location-specific models, when running on a different location, produce large prediction errors (5.21 dB rather than 2.5dB). But the unified model is always as good as or even better than all the location-specific models.

We repeat the experiment in the time domain. Table 2.1(b) confirms that training data over day and night can also be mixed together when building the unified model. We also use data collected in June 2018 to further test our model (trained using measurements from January and March 2018), and the unified model consistently provides better prediction than those designed for specific time periods of the day. The difference between the models is less visible compared to that in Table 2.1(a), indicating that physical location has a much heavier impact on spectrum monitoring than time.

We also experiment with the mobility context. We group the measurements by their mobility context: static (mixed locations), walking, and driving ($\leq$ 25 mph). In addition to the unified model, we also trained mobility-specific models for each of the three contexts. Table 2.1(c) shows that the unified model and the mobility-specific models perform similarly. For both, the average prediction error is bounded by 2.62 dB with a very low variance (0.26).

Overall, the unified model achieves the best prediction performance, 2.58 dB (0.19), when tested at a diverse set of observers. This can be attributed to two factors. First, the model's timing configuration (using 25.6 ms data to predict next 6.4 ms data) allows LSTM to capture critical spectrum usage patterns, and yet remains small enough to make the model robust against context changes. Second, the linear transformation allows LSTM to focus on intrinsic patterns of signal spectrogram, which remains consistent across different mobility context, time periods, and locations.

It should be noted that a related challenge is whether and how such unified model per LTE

(a) Across Location

Test \ Train	Loc 1	Loc 2	Loc 3	Unified
Loc 1	**2.71 (0.38)**	5.21 (12.74)	2.97 (0.56)	2.67 (0.32)
Loc 2	3.41 (0.64)	**2.46 (0.76)**	3.67 (1.09)	2.47 (0.24)
Loc 3	2.68 (0.46)	4.56 (7.03)	**2.63 (0.26)**	2.61 (0.29)
Mixed	3.70 (1.79)	4.86 (13.84)	4.07 (1.88)	**2.59 (0.23)**

(b) Across Time Periods

Test \ Train	Day	Night	Unified
Day time	**2.59 (0.23)**	2.74 (0.22)	2.53 (0.18)
Night	2.63 (0.23)	**2.71 (0.27)**	2.61 (0.22)
Mixed	2.59 (0.18)	2.74 (0.18)	**2.55 (0.22)**

(c) Across Mobility Context

Test \ Train	Static	Walking	Driving	Unified
Static	**2.52 (0.21)**	2.47 (0.23)	2.57 (0.33)	2.43 (0.23)
Walking	2.47 (0.29)	**2.62 (0.23)**	2.67 (0.27)	2.56 (0.18)
Driving	2.64 (0.29)	2.58 (0.26)	**2.59 (0.26)**	2.57 (0.23)
Mixed	2.66 (0.30)	2.54 (0.27)	2.61 (0.23)	**2.58 (0.19)**

Table 2.1: Prediction error (dB) of LSTM models under different training configuration. Numbers in parenthesis show the standard deviation of prediction error (dB). Here we show the result from the 880MHz band while the other bands lead to similar conclusions.

cell can be used near cell boundaries, where an observer can potentially pick up signals from multiple basestations. When these basestations operate on the same frequency band, the observer could see signal patterns that are different from those at in-cell locations. This must be treated with care to minimize false alarms. A potential solution is to utilize measurements at cell boundaries.

Model Complexity. We implement our LSTM and autoencoder models on a NVIDIA Titan X GPU, where it takes < 10ms for prediction on each data segment. Existing works have successfully deployed efficient LSTM models on mobile devices [62, 180]. The LSTM model in [180] has 5 layers, each with 500 LSTM units, and runs efficiently on Nexus 5 Android smartphones. In comparison, our LSTM has fewer parameters (2 LSTM layers, 64 units each) and should also run efficiently on common mobile devices.

	880 MHz	750 MHz	730 MHz	830 MHz (UL)
Testing: Early 2018	2.58 (0.19)	2.70 (0.25)	2.54 (0.26)	3.14 (0.78)
Testing: June 2018	2.61 (0.21)	2.72 (0.24)	2.52 (0.25)	3.11 (0.73)

Table 2.2: Model prediction error (dB) of our unified LSTM model.

2.4.3 Models for Different Spectrum Bands

We take a closer look at the unified models built for each of the four LTE bands. Recall that our analysis in §2.2.1 shows that LTE bands display different spectrum usage patterns. The uplink (UL) (830 MHz) is particularly different from the downlink (DL) bands.

Interestingly, the final model structure also differs between the DL and UL bands. For LSTM, the three DL bands share the same structure: 2 LSTM layers of 64 units plus 1 dense layer, while the UL band requires an extra LSTM layer. The same applies to Autoencoder: the DL models have 4 dense layers while the UL model has 6 dense layers. This is somewhat intuitive since LTE DL signals originate from a single strong transmitter (basestation), while UL signals are aggregates of many weak transmitters. The UL spectrum patterns are more complex, requiring more neurons to learn.

Table 2.2 lists the prediction errors of the four bands using LSTM. The three DL bands perform similarly, while the UL band experiences larger prediction errors. We also verified the same model using spectrum measurement data collected a few months later (June 2018), and the results are consistent.

2.5 Beyond the Per-Cell Model

We now consider the problem of building spectrum anomaly detection models for many LTE cells and multiple LTE bands. One can simply train a model for each LTE cell and band, but the training overhead and data collection requirements are practically prohibitive. Like other DNN models, LSTM and deep autoencoder require large amount of diverse training data and can take hours and even days to finish training. Currently, our models take 2 hours to finish with 1 day worth of training data and almost 24 hours with 8 days of training data. Practical

deployment will likely need larger and more diverse training data, and more frequent training to adapt the models. Thus it is critical to reduce the model training overhead across all the LTE cells. In the following we discuss and compare potential solutions to address the issue of training overhead.

2.5.1 Can Models be Reused?

The most immediate solution would be to reuse the model across cells and across bands. But does this really work?

Test I: Reusing Models across LTE Cells. We apply the LSTM model trained for one LTE cell to another LTE cell (same network carrier, same frequency band, same technology, just a different basestation), and observe sizable performance degradation in both model prediction and anomaly detection. For the 880 MHz downlink band, the average prediction error raises to 3.36 dB from the baseline of 2.58 dB and the standard deviation jumps from 0.19 to 0.56. This is likely because the two basestations are configured differently so their spectrum usage patterns differ.

Test II: Reusing Models across LTE Bands. We apply the LSTM model trained on the 880 MHz DL band to the 730 MHz DL band. The average prediction error and the standard deviation grow to 3.54 dB (0.71) compared to 2.70 dB (0.25). This is because the two bands show visible differences in spectrum usage patterns, which are captured by LSTM to produce a precise model for each. Autoencoder shows the same trend.

Together, these results show that models trained for a specific LTE cell and a specific LTE band are in general *not* reusable across cells and bands. This does not contradict with our conclusion in §2.4 where the per-cell model can be reused within the same cell. In a given cell, the spectrum usage pattern is fairly consistent because it reflects the behavior of either a single strong transmitter (DL) or aggregation of many weak transmitters (UL).

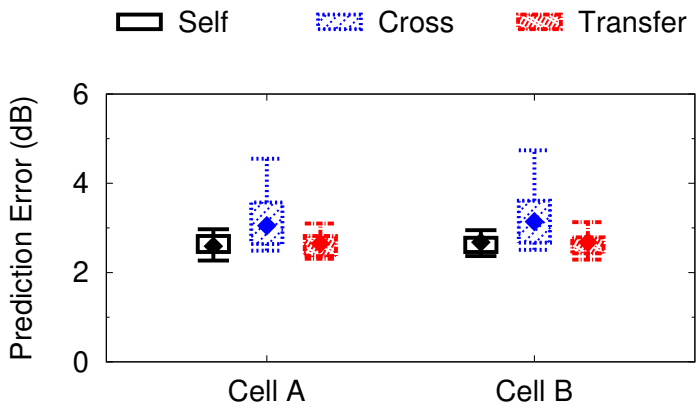


Figure 2.10: Prediction error of LSTM models transferred across different LTE cells in the 880 MHz DL band.

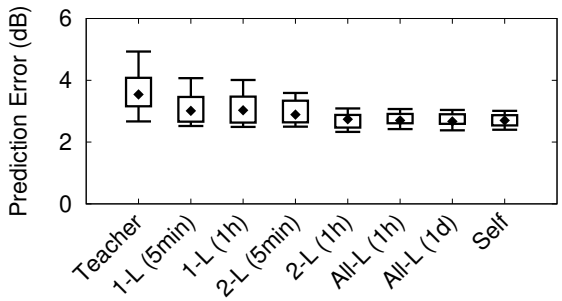


Figure 2.11: Prediction error of LSTM models with different transfer options (transfer model of DL 880 MHz to DL 730 MHz).

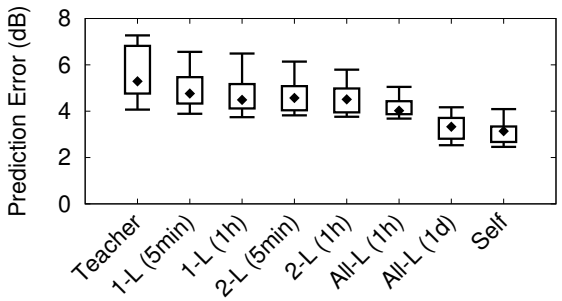


Figure 2.12: Prediction error of LSTM models with different transfer options (transfer model of DL 880 MHz to UL 830 MHz).

2.5.2 Fast Training via Transfer Learning

To speed up model training at many cells, we consider an alternative solution, *transfer learning* [204], which adapts a pre-trained DNN model to a new scenario using limited training data. It leverages the underlying similarity between tasks associated with two models. By transferring model architecture and weights from a pre-trained model (*teacher*) to the new model (*student*), one can bootstrap and fine-tune the student model with limited training data.

Transfer learning is suitable for our problem context because LTE cells share similar spectrum usage characteristics (§2.2), especially for DL bands since only LTE base stations are transmitting. Next we show that transfer learning can be used to quickly adapt a pre-

trained LSTM model to a new LTE cell and even to a new LTE band, reducing training data volume by a factor of 288.

We note that a similar concept of "knowledge transfer" has been applied to wireless networking design, using knowledge collected by one basestation to help configure another basestation (e.g., , spectrum handoff [146], operating modes for energy saving [152] and content caching strategies [53]). Our solution is motivated by these existing works, but our contribution lies in the *novel application* of transfer learning to the problem of spectrum anomaly detection, and a detailed validation using real-world LTE measurements.

When applying transfer learning, we first build a student model by copying the teacher model, then use local spectrum measurement data to refine the model. Here the transfer process depends on k, the number of model layers "allowed" to be updated [204]. The simplest form of transfer learning only updates on the last (dense) layer of the model, while the more advanced ones allow more or all the layers to be updated. By allowing more layers to be updated, the student model can better adapt to the new scenario but requires more local data to reach convergence.

Transfer across LTE Cells. To test the effectiveness of transfer learning, we transfer the LSTM model trained on LTE cell A (using 1 day of training data) to another cell B (same carrier, frequency band, technology), and fine-tune B's model with 5 minutes of new spectrum data collected in cell B. Here we chose 5-min because the tuning process converges. We consider the simplest transfer approach of *fine-tuning 1 layer*, freeze all weights of the two LSTM layers and only update weights in the last dense layer. For comparison, we train another LSTM model for B from scratch using the same amount of training data as of A (1 day).

Figure 2.10 shows the prediction error of the model trained from scratch (Self), the model directly borrowed from the other cell (Cross), and the transferred model (Transfer). The transferred model's error mean (2.65 dB) is extremely close to that of the model trained with the full data (2.59 dB). Yet this model only requires fine tuning the last dense layer

using 5-min local spectrum data, comparing with 1-day worth of data for Self (a factor of 288 reduction). The same conclusion holds when we test the autoencoder model.

Transfer across LTE Bands. Since different LTE bands display different spectrum patterns, we expect more efforts to complete the transfer learning. Since our LSTM model has three layers, we experimented with three transfer approaches: 1L, 2L, All-L, respectively, to reflect the number of model layers it needs to fine tune. We also include the results of copying the teacher model (Teacher) and training from scratch (Self).

Figure 2.11 shows the quantile distribution of the LSTM model prediction error (in absense of anomaly) by transferring the model of the DL 880 MHz band to the DL 730 MHz band, with different transfer options. Since the underlying temporal patterns differ between these two bands, only fine-tuning the last dense layer is insufficient (the average prediction error is 3.13 dB compared to 2.70 dB of Self) even after adding more training data. In the end, fine tuning 2 layers with 1 hour of data achieves comparable performance of Self. Interestingly, fine-tuning all 3 layers with 1 day worth of training data slightly outperforms training from scratch (Self). Again the same trend applies to the autoencoder model.

We also seek to transfer the downlink model (e.g., , 880 MHz) to the uplink band (830 MHz). As mentioned in §2.4.3, when trained from scratch, the 830 MHz band requires more dense layers for both LSTM and autoencoder than the three downlink bands. Thus direct transfer between the two types of bands might not be as effective as the above case. Figure 2.12 confirms that for LSTM, even after fine-tuning all the layers (of the transferred 880 MHz model), the prediction error is still not on par with that of Self (which needs an extra LSTM layer). Therefore, while transfer learning can potentially be applied to quickly customize LSTM (and autoencoder) models across bands, choosing the right teacher model can be a critical requirement. We leave this topic to future work.

2.6 Evaluation

In this section, we use real anomaly instances to evaluate our anomaly detection system built on the DNN models. Our evaluation seeks to answer the following questions: (1) whether our unified model performs as good as the (impractical) oracle system that builds context-specific model for each location, (2) how models trained via transfer learning perform in anomaly detection compared to those trained from scratch.

Choosing Anomaly Threshold. Our DNN models detect an anomaly if the difference between the measured data and the model predicted data exceeds a threshold. The threshold also determines the false alarm rate. To determine this threshold, we partition the model training data to two subsets: the training set and the validation set. After a model is trained, we use the validation data to calculate the statistics of the model prediction error. For our dataset, these errors can be modeled using a Gaussian distribution. From this distribution we can calculate, for each false alarm rate, the corresponding threshold on the prediction error.

Next, we discuss our experiments using two types of anomalies: unauthorized transmitters and misconfigured LTE basestations. For all the experiments, the detection rate of autoencoder is similar to that of LSTM (only 2-4% worse while keeping the same false alarm rate). Thus we only show the LSTM result for brevity.

2.6.1 Detecting Unauthorized Transmissions

We generated anomalies in the form of unauthorized spectrum usage, where an "unlicensed" transmitter (USRP N210) broadcasts various types of signals. Our anomaly instances include transmissions using the entire 5MHz band and OFDM signals (like LTE), using a portion of the 5MHz band (1, 2, 3MHz) with OFDM signals, and a narrowband misuse with QPSK and BPSK signals. When the anomaly is on, we set up a static observer and a walking observer in proximity (within 50 meters) who collect LTE measurements and perform anomaly detection.

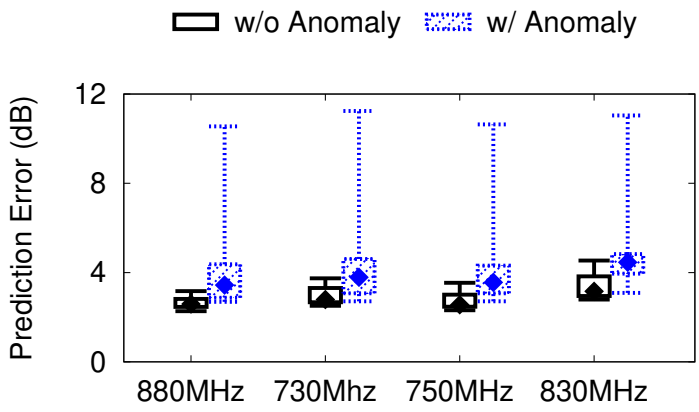


Figure 2.13: Quantiles of model prediction errors w/ and w/o anomalies.

The walking observer follows a pre-defined route for all the anomaly instances. We did not have any driving observers since they quickly go out of range of our low power transmitters. In total, we performed 100 experiments, each of 5 minutes long.

Ethics. We are very aware of the potential impact of our experiments on cellular users, and took extensive precautions to ensure that these experiments had no impact on cellular users or basestations. First, we chose the second floor of an older campus building with heavy concrete walls and floors as our setting. We first measure signal propagation properties in the building by setting the transmitter frequency to 900MHz (closest unlicensed band), and then using a spectrum analyzer to measure signal strength at numerous locations inside and outside of the building. We confirmed that the thick concrete walls and floors completely blocked signals beyond the immediate open hallway and adjoining offices and no signals were observed outside the building or on floors above or below. Next, we scheduled cellular experiments late at night and on weekends, when the campus building is generally unoccupied. Finally, between experiments, one student walked the entire length of the hallway and checked to make sure no one else is on the floor. We did not encounter any other occupants of the building during our experiments. We also periodically used spectrum analyzers to (re)confirm that our transmissions are strictly constrained to the second floor.

Results: Anomaly Detection Accuracy. Figure 2.13 compares the quantile distribution (5%, 25%, 50%, 75%, 95%) of the model prediction error (dB) per spectrum observer, *i.e.* the RMSE between the measured and predicted spectrograms, across all the measurement instances, with and without anomalies. The presence of anomalies largely increases the prediction error. The two distributions are reasonably separated for the three DL bands, but overlap slightly for the UL band (830MHz). Next, Figure 2.14 plots the RoC result (anomaly detection rate vs. false alarm rate) for the four LTE bands. Here we average the detection result across all the measurement instances collected by the static and mobile observers, producing the average detection rate per spectrum observer. We see that for the three DL bands, the anomaly detection results are on par with each other, while the UL band is less effective. Yet these results are still significant better than non-DNN solutions (see Figure 2.5).

In our experiments, misdetection occurs when the anomaly's signal power is low and the observer is further away from the misuse. In practical deployment, one can improve the anomaly detection rate by deploying more observers for density and coverage, further pushing the need for a context-free model.

Results: Unified vs. Customized Models. We compare our unified model to an oracle system that builds context-specific models for each location. We compute the anomaly detection result for the oracle system by training a model for each static observer using its own past observations, and testing the model using the anomaly instances in range of the static observer. Note that our unified model is tested on all the anomaly instances and on both the static and mobile observers. As shown in Figure 2.15, our unified model performs as well as the oracle.

Results: Transferred vs. Self-trained Models. Similarly, we compare the anomaly detection performance of models transferred from other cells with those of models trained from scratch. Figure 2.15 shows that the transferred model achieves almost identical anomaly detection performance while greatly reducing the training overhead.

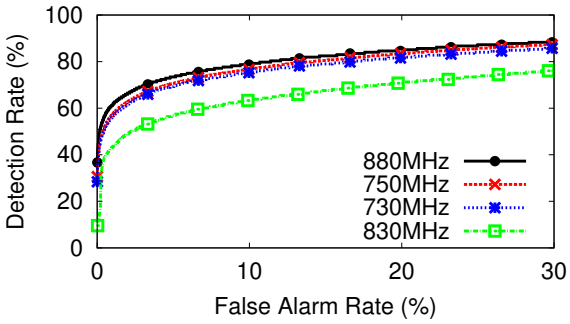


Figure 2.14: Detection rate vs. false alarm rate of unauthorized transmitters of different bands.

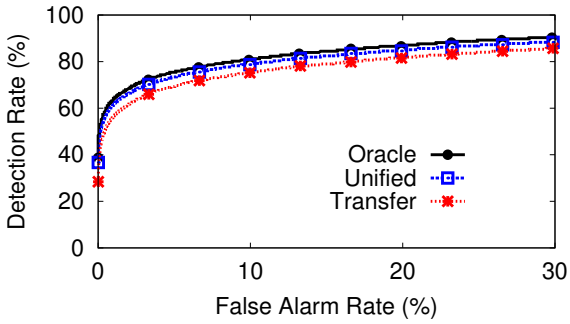


Figure 2.15: Our unified and transfered models are on par with the oracle design that uses location-specific models.

2.6.2 Detecting TX Misconfigurations

We also study anomalies of basestation misconfiguration implemented by modifying our LTE measurement traces described in §2.2. This will not produce any impact on cellular services. We consider two types of misconfiguration: (1) the misconfigured basestation stops transmitting signals at some or all sub-frequencies; (2) the misconfigured basestation suddenly changes its transmit power level. We produce both instances by modifying our LTE downlink measurement traces, replacing them with replays of measured noise signals or increasing/decreasing the amplitude of the received signals.

We note that these anomalies could also be detected by other methods, since (strong) static observers will likely detect changes in the spectrogram. Instead, we use these to show that our unified model can detect *general* types of anomalies beyond unauthorized transmissions. That is, the *same* model can detect both unauthorized transmissions and misconfiguration of basestations.

Detection Results. Table 2.3 lists the average detection rate per spectrum observer under 1% false alarm rate for different categories of misconfiguration. Here "x% F down" means transmissions on x% of frequency is replaced as noise. "ΔP =x dB" means transmit power is modified by x dB. Even at a very low 1% false alarm rate, the same unified model (as in §2.6.1) can effectively detect anomalies caused by misconfiguration, and the detection

rate correlates with the severity of the anomaly. While linear transformation used to build our model "suppresses" the impact of transmit power level, our model can still detect sudden basestation power changes because it creates notable changes in the spectrum usage pattern.

	ΔP=3 dB	ΔP=5 dB	33% F down	66% F down	100% F down
880 MHz	58%	78%	52%	87%	100%
750 MHz	60%	81%	57%	90%	100%
730 MHz	53%	78%	54%	88%	100%

Table 2.3: Anomaly detection rate at 1% false alarm rate.

2.6.3 Recognizing Anomaly Types

In addition to detecting anomalies on the fly, our LSTM based approach can potentially recognize each individual type of anomalies, by examining the temporal pattern of the prediction error.

Spectrum Misuse. Figure 2.16 shows a sample event where the misuse starts at 4s and the LSTM prediction error elevates immediately (from 2dB to 8-30dB), indicating an anomaly event. As long as the misuse is present, the prediction error remains elevated and displays a large variance over time. Therefore, an observer can detect this anomaly by taking a spectrum measurement at any given time.

Misconfigured LTE Transmission Frequency. Figure 2.17 shows a randomly chosen example, where the basestation enters an outage at time 2s and recovers at time 6s. These two sudden changes trigger two sharp spikes (immediately) in the LSTM prediction error, which remains elevated during the outage. Compared to the misuse scenario that introduces an extra transmitter, the mean and variance of prediction error during the frequency outage are relatively smaller. On the other hand, the unique feature of the prediction error (a spike followed by elevated but stable values) can be used to distinguish this anomaly from the misuse ones.

Misconfigured LTE Transmit Power. Figure 2.18 shows an example that includes three events, each with a different amount of power change. Here the LSTM prediction

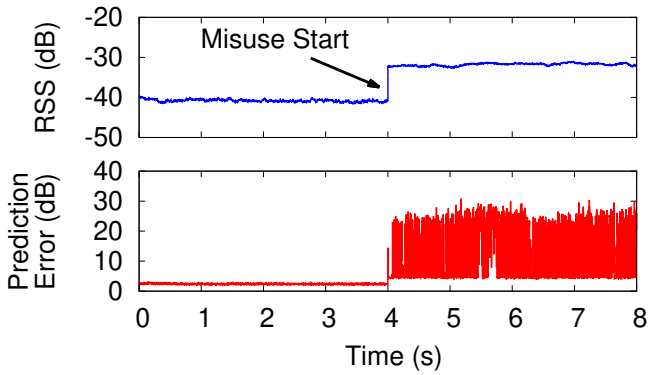


Figure 2.16: Unauthorized TX.

error displays a spike at the time of sudden power change, but falls back to normal values (< 3 dB) immediately. This means that to detect such anomaly, the observer will need to monitor the spectrum band continuously. We note that such anomaly is inherently harder to detect without continuous monitoring, as transmissions post power change still display similar spectrum usage patterns. By simply looking at each instantaneous observation, such change in power can also be caused by moving to a new location or human body blockage.

The above benchmarks indicate that each anomaly type displays unique patterns (Figure 2.16–2.18). Furthermore, the amplitude of the prediction error also shows strong correlation with the strength of the anomaly, which can be used to facilitate fault diagnosis and localization.

The same task is challenging by just observing the RSS values. For example, the RSS patterns of the unauthorized transmission in Figure 2.16 and the sudden rising of LTE transmit power in Figure 2.18 are indistinguishable. But the corresponding LSTM prediction errors are largely different.

2.7 Related Work

Anomaly Detection and Diagnosis in Wireless Networks. Existing works can be divided into three categories, depending on who runs anomaly detection and diagnosis. The first category involves system administrators. Many [77, 60, 196, 112, 130] use system logs

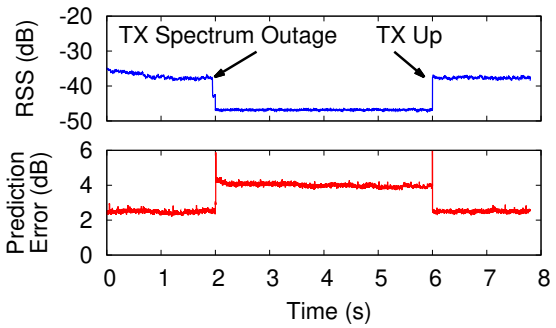


Figure 2.17: Misconfigured LTE transmit frequency.

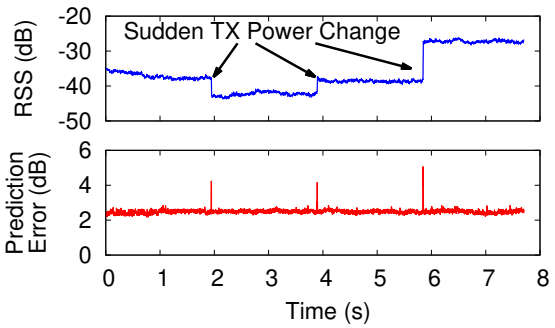


Figure 2.18: Misconfigured LTE transmit power

or Key Performance Indicators (KPIs) to detect network outages and performance degradations [210]. The second category uses diagnosis by network clients. WiFiProfiler [65] studied 802.11 fault detection using information of client's wireless configuration and measurement data from the transport layer and above. The third category uses third-party devices to monitor and detect physical layer anomalies such as spectrum misuse, using metrics like signal strength variations [230, 72].

Our work falls into the third category. Our key contributions are the novel application of DNN (LSTM and deep autoencoder) and transfer learning to the problem of spectrum anomaly detection, the design of a context-free DNN model, and the empirical study using measurements by both static and mobile sniffers.

Misuse Detection for Opportunistic Spectrum Access. Existing works have studied the issue of spectrum usage violation where a secondary user tries to transmit when a nearby primary user is inactive. They consider individual features of signal transmissions, including average received signal strength distribution over space [69, 137, 164, 165, 185, 51], signal strength variation [71, 263], physical channel properties [74, 207, 251], signal amplitude difference between direct and reflected paths [168] as well as airtime utilization [236]. Most of these designs were based on abstract propagation models, which do not capture real world settings.

Our work considers the issue of spectrum anomalies due to unauthorized transmitters

and misconfiguration of LTE basestations. Our work differs from existing works by taking an empirical, data-driven approach. Instead of relying on a fixed set of features, we build DNNs to automatically extract the features required for accurate anomaly detection, and develop context-free DNN models.

Machine Learning for Signal Classification and Anomaly Detection. Early work on anomaly detection focused on statistical hypothesis test [236] and threshold-based methods [230, 137]. Recently, ML models have been applied to the problem of signal classification (*e.g.* [212, 226]) and spectrum misuse detection [200, 164, 102, 249]. [200] used a small scale study to show that LSTM outperforms Kalman sequence predictor. [102] developed an Autoencoder model for spectrum anomaly detection, based on a limited dataset (1000 samples) on the FM band. [164] applied one-class SVM to detect spectrum anomaly (via simulations) while [249] used supervised learning to train Hidden Markov Models.

These existing works focused on anomaly detection by a single static observer without considering the impact of user mobility and location. They used either simulation or few measurement data for validation. Our work has a much broader scope by developing robust, scalable anomaly detection capable of detecting previously unknown anomalies, for both static and mobile observers. We also collected detailed signal measurements on four LTE bands and under different user context to drive our empirical study.

2.8 Conclusion

Conclusion. We show that a scalable DNN model on LTE spectrum usage can be built and deployed on a wide range of observers (static nodes, walking users, buses), enabling real-time spectrum anomaly detection. Its performance matches the "oracle" design that trains customized models for each specific user context (location and mobility). The model remains constant for any observer in a single cell, and can be quickly trained and adapted using a small amount of local spectrum measurements. To the best of our knowledge, this is

the first to show the feasibility of building practical and general spectrum anomaly detection systems for large-scale LTE networks.

Future Work. Moving forward, there are several open questions.

1) Due to the cost of spectrum data collection, we only evaluated our cross-cell design (i.e., fast training via transfer learning) on data collected from two cells. Our design could be further validated with spectrum data of more cells.

2) Anomalies tested in our evaluation are emulated using an "unlicensed" transmitter (USRP N210), in-the-wild spectrum anomaly data could be collected to validate the effectiveness of those DNN models.

3) Spectrum measurements at individual sensor can be noisy [193], or corrupted/modified by well-equipped adversaries. Our detection system could be refined to be robust against such artifacts.

4) The frequency-temporal distribution of the prediction error could be used to distinguish different anomaly types. This needs to be further validated using more anomaly instances in the wild.

CHAPTER 3

EVALUATING VIDEO ANALYTIC PIPELINES WITH HETEROGENOUS WORKLOADS

In this chapter, I explore how heterogeneity and scalability affect **the evaluation of ML based sensing and monitoring system**. Here I focus on evaluating ML-based video analytic pipelines (VAPs), given their increasing popularity in smart city monitoring. The difficulty of proper evaluation in such systems lies in the diverse video workloads caused by heterogeneous environments. Without consideration of such heterogeneity, existing VAP evaluations are incomplete, often producing premature or ambiguous results. Therefore, we build the first VAP benchmark to comprehensively evaluate VAPs by characterizing the complex dependencies of VAP performance on video content characteristics.

3.1 Introduction

Edge video analytics is becoming the modern solution to many critical tasks [13]. With the ability to accurately detect, recognize and track objects on the fly, it can quickly detect and respond to traffic accidents and hazard events [28, 1, 12, 16, 31, 32, 27], monitor and enforce physical distance during COVID-19 [7, 29], auto-manage retail stores and factories [24], and perform surveillance functions to make the world safer [6, 5].

Deployment of edge video analytics at scale, however, must address the tension between inference accuracy and resource cost, i.e., *compute* cost to run inference tasks and/or *bandwidth* cost to transfer data from cameras to servers [67, 194]. This tension continues to grow as video sources proliferate at the network's edge [6, 5, 30, 15, 35], separated from the heavy compute power necessary to run large deep neural networks (DNNs) by a bandwidth-constrained mobile network.

In response, researchers have developed numerous *video analytics pipelines (VAPs)* to optimize the accuracy and cost tradeoff [139, 134, 228, 262, 133, 128, 113, 70, 261, 203, 131,

264, 96, 162], by combining DNN model compression/speedup techniques with video processing heuristics such as frame sampling and image downsizing (see Figure 3.1). For instance, Chameleon [134] shows that intelligently subsampling traffic video frames at the cameras can effectively reduce network and compute costs without degrading inference accuracy.

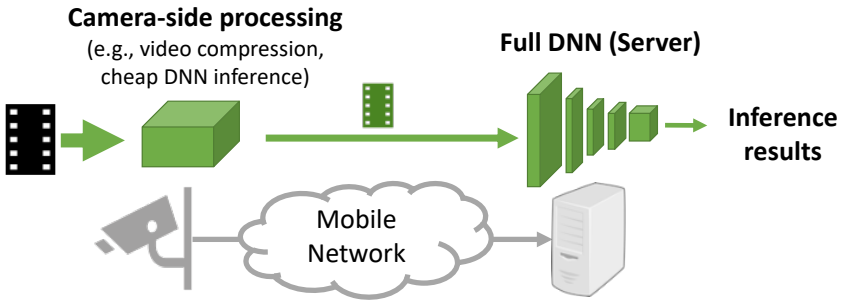


Figure 3.1: Illustration of a video analytics pipeline (VAP).

As edge video analytics and VAPs continue to evolve, accurate and transparent evaluation of VAPs becomes critical. For instance, operators of edge video analytics need to know what the optimal VAP is for a given video input, how often the network/compute usage exceeds a budget, or how often accuracy drops below a threshold.

Evaluating VAPs Today, VAPs are evaluated using some corpus of past video samples that represent the target scenario(s). After running VAPs on these videos, their performance (i.e., the accuracy and cost tradeoff) is analyzed and compared against each other. Following this method, we run an empirical study to evaluate seven VAPs from recent papers, using a large chunk (14.5 hours) of traffic videos. Our study shows that today's evaluation method is insufficient to characterize VAPs, often leading to partial/premature conclusions on the efficacy of a VAP and across VAPs. This is because VAP performance has a strong dependency on video content – it can vary substantially across videos even in the same scenario (*e.g.*, highway traffic cameras), and drift dramatically over time when operating on the same camera. Therefore, today's evaluation is either biased by the use of short video clips or produces vague results over long videos, i.e., an excessively wide distribution of possible cost-accuracy outcomes.

Our measurement study suggests that an ideal evaluation of VAPs must have high performance coverage and low performance variance. Here, "high coverage" means the evaluation reveals both good and bad performance of a VAP, whereas "low variance" means the evaluation could accurately estimate the VAP's performance on individual videos. And the strong dependency of VAP performance on video content suggests such ideal evaluation must characterize the complex interactions between video workloads and a VAP's performance. Doing so presents three distinct benefits for VAP design and deployment: (1) providing a comprehensive assessment of VAPs under diverse video characteristics; (2) understanding how/why each VAP's performance varies across videos; (3) revealing relative strengths among VAPs under different video content characteristics. We refer to this new evaluation requirement as *VAP performance clarity*.

Achieving performance clarity A direct approach would test VAPs exhaustively on a large collection of mobile video workloads, *e.g. existing* video collections developed for testing DNN models [87, 160, 126, 269, 80, 116, 79]. Yet these are designed to evaluate DNN architectures rather than VAPs, thus lack sufficient coverage of video characteristics that will affect VAP performance. An alternative is to build a database of empirical workloads that covers all possible video feature value combinations, and use them to test VAPs. This is intractable, however, since it would require a large database capturing an exponential number of video feature combinations.

Instead, we propose to characterize VAP performance using a carefully curated set of videos that serve to evaluate different aspects of VAPs. Our design is based on the observation that each VAP is inherently modular and can be broken into a set of "global" primitives. Each primitive leverages a distinct set of video processing heuristics to optimize the accuracy/cost tradeoff, and thus can be profiled independently (against its associated video features) and then (re)assembled to profile full VAPs. This modular structure allows us to efficiently profile each full VAP by combining its corresponding primitive-specific profiles. Note that some prior works also observe independent VAP modules but use it to refine par-

ticular VAP designs [128, 134]. In contrast, we leverage this observation to design accurate evaluation of many VAPs.

We present **Yoda, the first VAP benchmark** designed to achieve performance clarity. Using a carefully curated set of benchmark videos (67 minutes in length), Yoda focuses on characterizing the complex dependencies of VAP performance on mobile video content characteristics, and does so efficiently. For each VAP v, Yoda builds a performance clarity profile ($\mathbb{P}_v$) by running v on a set of benchmark videos parameterized by a set of video features, both chosen based on v's design primitives. The resulting $\mathbb{P}_v$ is a lookup table that lists v's performance (the accuracy/cost relationship) under different video feature values. This provides a comprehensive and transparent assessment of v's performance and its dependencies on video features. We show Yoda's contributions towards VAP evaluation, in three concrete aspects.

- **Performance clarity** – Yoda accurately captures existing VAPs' performance and their dependencies on video features. It largely outperforms existing VAP evaluations with higher coverage (the completeness of the evaluation) and lower variance (the ambiguity of the evaluation outcome).
- **Performance prediction** – Using $\mathbb{P}_v$, Yoda can efficiently estimate v's performance for videos *not* included in the benchmark set, without running v. This takes 2 orders of magnitude less computation than running v on the video.
- **Practical insight for VAP deployment** – Yoda's VAP profiles expose strengths and weaknesses among existing VAPs, and the underlying deployment scenarios and video features associated with these conclusions. These insights allow us to identify previously hidden gaps and opportunities to guide/motivate future VAP designs.

Though Yoda serves well on the seven VAPs considered in this work, it is *not* without limitations. Currently, Yoda's content features and benchmark videos are not future-proof (e.g., Yoda does not support multi-stream/multi-query VAPs). For distributed VAPs that han-

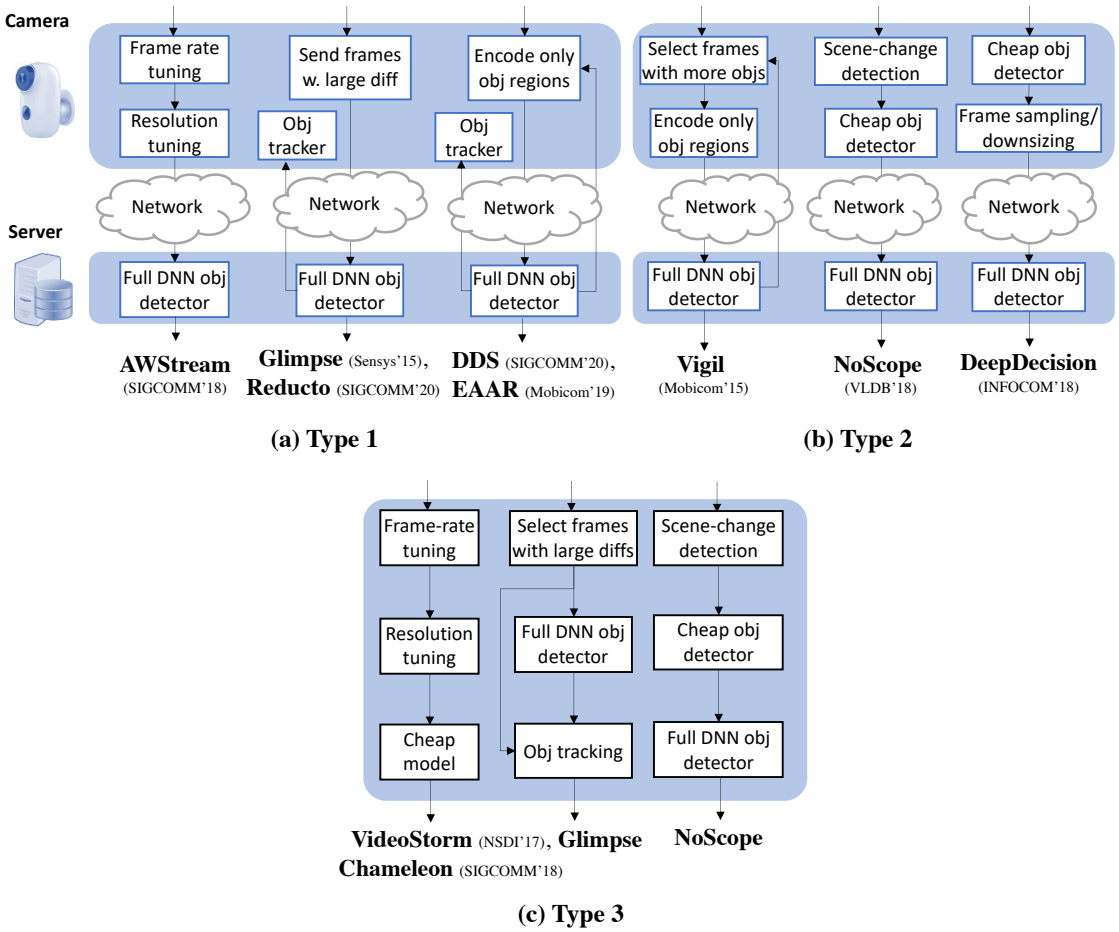


Figure 3.2: Schematic illustration of some example VAPs grouped into three general types. (Differences within each general scheme are omitted here.) Our goal is not to list all VAPs; instead, we seek to identify common techniques and their performance.

dle bandwidth-constrained connections, Yoda only evaluates reductions in average network bandwidth usage but not the impact of bandwidth fluctuation.

Nonetheless, as the first attempt at benchmarking VAPs' performance clarity, Yoda suggests a viable path towards profiling the dependencies of VAPs' performance on video content via a *modularized* approach. Our goal is not to realize an "ideal" benchmark; rather, we provide a concrete implementation of the proposed benchmark, which validates the need for performance clarity and initial feasibility on accurate performance evaluations of VAPs, and provides new insights for VAP design and deployment. We release the Yoda toolkit in https://yoda.cs.uchicago.edu and plan to expand our study to include other VAPs and

additional video features.

3.2 Preliminaries

In this section, we present an overview on existing VAPs, focusing on their design objectives and evaluations.

3.2.1 VAP Design

Computer-vision DNNs are generally optimized for high accuracy. However, the compute and network cost to achieve such accuracy can be high[1]. This tension between accuracy and cost has stimulated many ongoing efforts to develop video analytics pipelines (VAPs) [261, 134, 264, 162, 262, 133, 139, 155, 93]. VAPs reduce network/compute cost while maintaining high inference accuracy, by combining DNN compression/speedup methods and video processing heuristics such as frame sampling and image downsizing.

Existing VAPs fall in three general types (Figure 3.2).

- **Type 1: Saving network cost when the camera has low local compute power.** The camera only encodes video frames and runs simple tracking algorithms, but does not perform any inference that requires accelerators such as GPUs. Instead, a VAP saves network cost by selecting a subset of frames/pixels to send to the server for DNN inference. For example, *AWStream* [261] adapts video frame rate, resolution and quality. *Glimpse* [70] and *Reducto* [155] send only frames that contain new objects (e.g., identified by measuring inter-frame difference). Similarly, *EAAR* [162] and *DDS* [93] only encode regions that are likely relevant to the inference task.
- **Type 2: Saving network cost when the camera and the server split the inference task.** Here the camera device is equipped with some inference power (e.g., with a low-

1. For instance, running state-of-the-art object detector at 30fps requires one NVidia GTX Titan X GPU (¿$1.1K) [126] and streaming the video at 720p ($\sim$ 5Mbps) costs $2K/day for AT&T 4G LTE network ($50 for 30GB data before the speed drops to a measly 128kbps [3]).

power GPU) and thus can run a cheap DNN. For example, *Vigil* [264] runs a cheap object detector on the camera to identify regions containing most objects and sends only these regions to the server for full DNN inference. *NoScope* [139] first identifies frames with significant pixel changes and runs a cheap DNN (fine-tuned per video stream) on these frames. Only when the cheap DNN has low confidence will the frames be sent to the server for further inference.

- **Type 3: Saving compute cost of a resource-constrained edge device.** The third type of VAPs reduces compute cost, when a camera device (or edge server) has moderate compute power to run some inference locally. *Videostorm* [262] and *Chameleon* [134] uniformly sample frames, downsize the sampled frames to a lower resolution, and process them using a less accurate yet cheaper DNN model. We note that *Glimpse* (Type 1), *NoScope* (Type 2) can also be applied here to reduce compute cost, and thus fall into this type.

3.2.2 How Are VAPs Evaluated Today?

Today's evaluation empirically tests and compares the VAPs' performance (accuracy, cost) on a set of videos collected from the target scenario(s) [139, 262, 128, 134], e.g., some traffic videos recorded by fixed cameras in urban crossroads. Table 3.1 lists the target scenarios and videos (sources and lengths) used to evaluate some recent VAPs.

Such evaluation relies on an implicit assumption:

Today's evaluation assumption: *A VAP's performance under a target scenario can be represented by its performance seen on a set of long videos of the same scenario.*

Unfortunately, this is not always true. Our own measurement study shows that a VAP's performance can vary dramatically among videos of the same scenario (see §3.3.2).

VAP	Target scenarios (sources of videos) "YT" = YouTube, "P" = proprietary	Total duration (*# of videos*)
Glimpse [70]	Moving traffic cams (YT) + Face (P)	65min (*30*)
AWStream [261]	Fixed traffic cams (MOT16) + AR (P)	6.3min (*4*)
Vigil [264]	Campus cams + Indoor (P)	3min (*3*)
Reducto [155]	Fixed traffic cams (YT)	250min (25)
Chameleon [134]	Fixed traffic cams + Indoor (P)	525min (*15*)
DDS [93]	Fixed & Moving traffic cams (YT)	30.7min (*16*)

Table 3.1: Today, VAPs are evaluated on videos of one or two scenarios as a whole. For consistency, we only list object detection datasets.

3.3 What Is Missing in Today's VAP Evaluation

As video analytics and VAPs continue to evolve, accurate and transparent evaluation of VAPs is crucial to their real-world adoption. In this work, we are interested in understanding whether today's VAP evaluation methods (§3.2.2) can fulfill this requirement. Since existing VAP proposals generally run evaluation using different datasets, one cannot directly assess and compare their performance from their reported results. Instead, our empirical study evaluates 7 popular VAP designs using the same video datasets (14.5 hours in total) that consist of a much larger and more diverse collection of traffic videos. Our analysis reveals significant VAP performance variability across videos of the same target scenario, suggesting that today's evaluation method is insufficient to characterize VAPs. We then discuss its implications for a better VAP evaluation, which lead to the development of Yoda.

3.3.1 Our Empirical Study on VAP Evaluation

We start by discussing the methodology behind our measurement study.

VAPs studied We study and compare the performance of 7 recent VAPs on the task of *object detection*. These include AWStream [261], Glimpse [70], Vigil [264], NoScope [139][2], Videostorm [262], Reducto [155], and DDS [93]. They cover a wide range of today's VAP

2. We include NoScope in our study since it is also applicable to object detection, although it was only evaluated on binary classification.

design techniques illustrated in Figure 3.2.

For consistency, we configure all these VAPs to operate on videos of (30fps, 720p) and all use the same pre-trained DNN model as their *full DNN model*. To choose the full DNN model, we experimente with several popular choices (e.g., FasterRCNN-ResNet101 [26], Yolo [214]) and select FasterRCNN-ResNet101 since it produces the highest accuracy in object detection. Later we also repeat our experiments using Yolo, and find that while the absolute VAP performance varies slightly, the key findings remain the same. Finally, we consider the scenario where VAPs are "optimally configured" to eliminate potential inconsistency or errors introduced by imperfect system configuration. For each video segment (≈30s), we configure each VAP by picking its best parameter values (e.g., frame sampling rate of VideoStorm, or inter-frame difference threshold of Glimpse) that minimize cost while achieving over 0.9 inference accuracy in the first 1/3 of the segment. We then test and report the VAP performance on the rest of the video segment. We believe this consideration helps increase the fairness and transparency of our VAP evaluation.

Our "coverage" dataset To show a more complete picture of VAP performance, we compile a *coverage set* of public traffic videos from a diverse video sources at a much larger scale than existing works. We target specifically traffic videos since they are commonly used in VAP evaluation (see Table 3.1). When compiling our dataset, we seek to include public traffic videos from diverse sources, covering different scenarios (fixed or moving cameras; day or night; highway, city or rural streets), and videos displaying a wide range of content characteristics and dynamics, e.g., object speeds, sizes, object arrival rate.

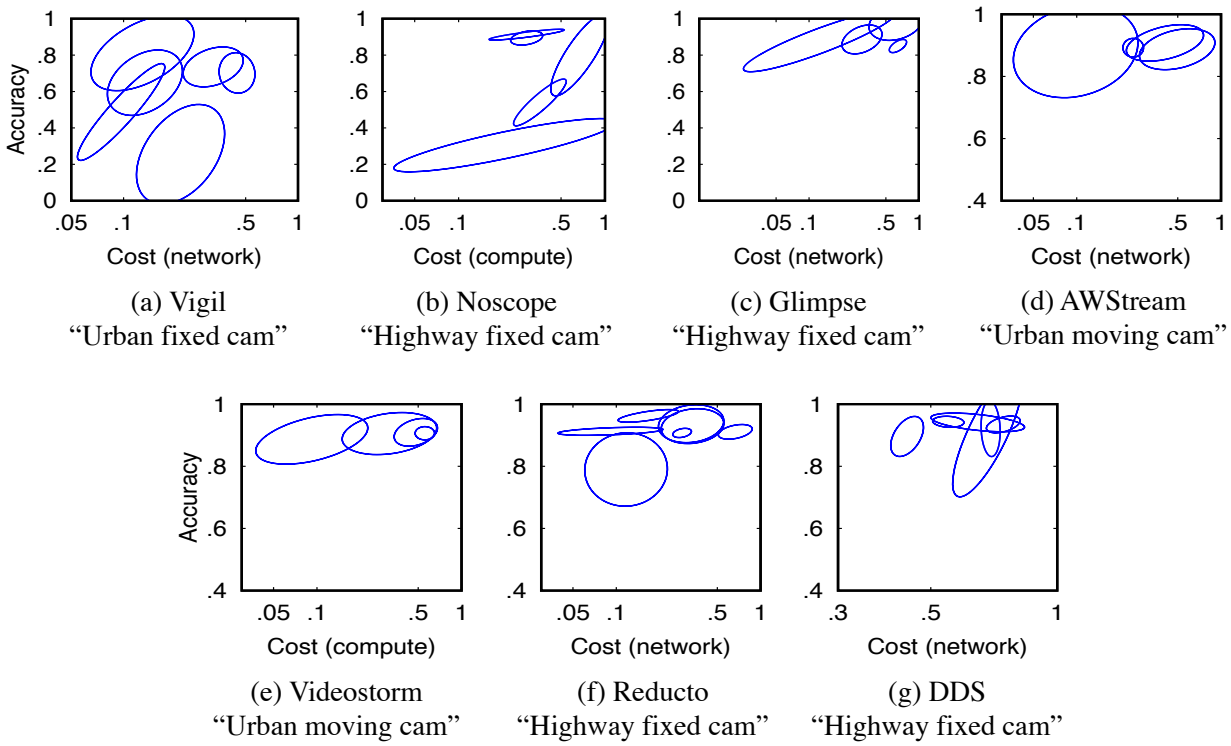
With these in mind, our final coverage set consists of 14.5 hours of traffic videos from multiple sources: YouTube (32 long videos, 10-47 minutes each), Waymo [234] (5 hours), KITTI [106] (20 minutes), and MOT [183] (8 minutes). All videos are split into 2112 segments (≈30s per segment).

Performance metrics used We measure each VAP's performance using the following three metrics:

- **Accuracy** is measured by the F1 score of a VAP's detected objects [97]. We obtain the "ground truth" results by running the full DNN on the uncompressed video frames (rather than the human-annotated labels). This way, any inaccuracy will be due to VAP designs (e.g., video compression, DNN distillation), rather than errors made by the full DNN itself. This is consistent with recent work (e.g., [261, 264, 262, 134, 190, 139]).
- Normalized **network cost** defines the data size sent by the camera to the server divided by the size of the original video. Reducing network cost is crucial when deploying VAPs in bandwidth-constrained networks [194].
- Normalized **compute cost** is the average GPU usage (on a NVidia GTX Titan Xp) per frame divided by that of the full DNN model. Since the cost is normalized against running the full DNN model on the same GPU, it is less dependent on the particular choice of GPU. Note that when a VAP (e.g., Glimpse) reduces both compute and network costs, we will specify which is being considered.

Finding 1: *Performance of a VAP can vary dramatically even among videos of the same scenario.*

We acknowledge that there are other aspects of VAP performance beyond these metrics. Our choice of these metrics is based on two reasons. First, these metrics are directly related to video content. For example, evaluating things like how adaptive a VAP is to bandwidth variations is important but deviates from our main goal of understanding the impact of video content. Similarly, metrics like throughput, processing delay or energy consumption are crucial but also highly sensitive to the implementation details (e.g., pipelining or parallelization) and hardware platform. Second, these metrics can be translated into practical objectives. The feasibility of deploying a VAP depends on whether its costs fit the provisioned compute/network resources or the deployment budget. Although we do not evaluate other performance metrics (e.g., throughput, latency) explicitly, we believe they are highly correlated with the network and compute cost considered by our study. For example, when



(a) Vigil "Urban fixed cam" (b) Noscope "Highway fixed cam" (c) Glimpse "Highway fixed cam" (d) AWStream "Urban moving cam"

(e) Videostorm "Urban moving cam" (f) Reducto "Highway fixed cam" (g) DDS "Highway fixed cam"

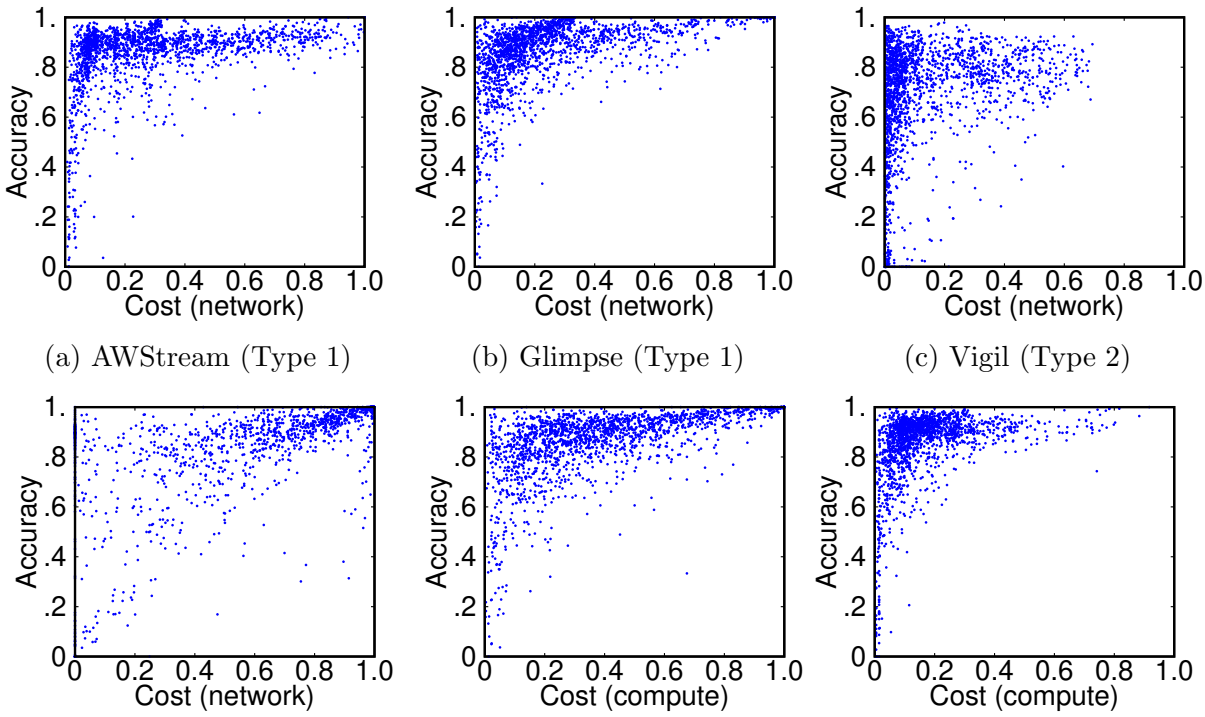
Figure 3.3: Significant performance variability of the same VAP among videos of the same scenario. Each ellipse outlines the 1-σ range of VAP performance across the segments of a video.

a VAP reduces network cost by 2x, this saving can translate into serving 2x video streams while meeting the same inference accuracy target (i.e., 2x throughput).

3.3.2 Key Findings

Finding 2: *Choice of optimal VAP is content-dependent.*

Following the traditional assumption (§3.2), we test each VAP's performance (cost vs. accuracy) in one of the four *scenarios*: {fix-positioned traffic monitoring cameras, moving dashboard cameras} × {on urban streets, or on highway}. Figure 3.3 summarizes each VAP performance range in each video (each over 20 minutes) in one ellipse. We see each VAP's performance can vary dramatically across videos in the *same* scenario. Such *performance heterogeneity* is prevalent across all 7 VAPs and four scenarios considered by our study.



(a) AWStream (Type 1) (b) Glimpse (Type 1) (c) Vigil (Type 2)

(d) NoScope (Type 2) (e) Glimpse (Type 3) (f) VideoStorm (Type 3)

Figure 3.4: Significant performance variability across video segments. Each dot shows the accuracy and cost of a segment.

To reveal the full range of performance variability, Figure 3.4 plots the performance distributions of the 5 VAPs on all the video segments in the coverage dataset (each dot shows the performance on one segment). While the overall trends align with findings of prior work (VAPs trade accuracy drop for saving network/compute cost), we do see that each VAP has a significant performance variability across video segments.

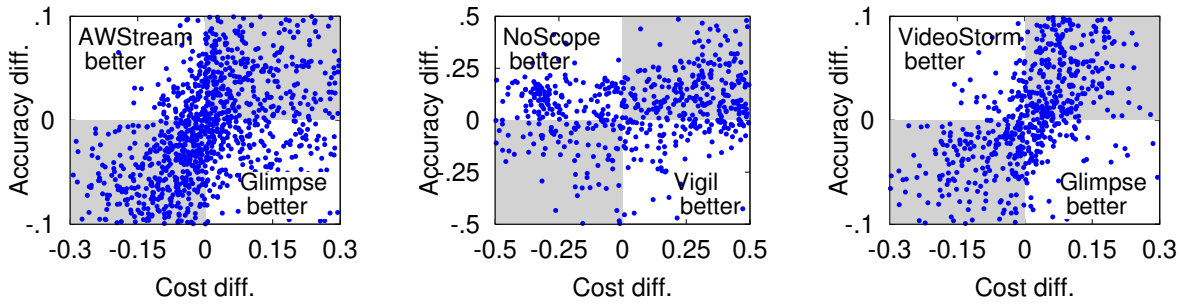
Even when we restrict the accuracy to a small range ([0.90, 0.95]), the relative standard deviation of cost across segments can be 45-102% and the gap between 5^{th} and 95^{th} percentiles is always over 90% (shown in Table 3.2). Here, relative standard deviation is defined as the ratio of the standard deviation to the mean, which is a popular metric to measure the dispersion of a distribution.

Performance variance does not always lead to suboptimal choice of VAP, if one VAP *always* outperforms others. Unfortunately, that is not true for VAPs. We illustrate this by compar-

Cost when acc. is in [0.9, 0.95]	VAP type 1		VAP type 2		VAP type 3	
	AWStream	Glimpse	Vigil	NoScope	Glimpse	VideoStorm
Mean	0.34	0.27	0.15	0.67	0.41	0.20
Relative StdDev	73%	64%	102%	48%	45%	68%

Table 3.2: Even when we narrow the range of accuracy in Figure 3.4 to [0.90,0.95], the cost (network or compute) across segments could vary significantly. This can be seen from the relative standard deviation values in the table.

ing VAPs in pairs. In each pair, one VAP acts as a "reference", and we subtract the other VAP's cost and accuracy on each video segment by those of the reference. Figure 3.5 shows the results of three VAP pairs and marks the region where one VAP is strictly better than the other (higher accuracy *and* lower cost). Clearly, the choice of best VAP varies across video segments and is content-dependent. Thus, it is crucial for VAP operators and developers to understand under *what videos* would one VAP perform better than others.



(a) AWStream vs. Glimpse (Type 1)

(b) NoScope vs. Vigil (Type 2)

(c) VideoStorm vs. Glimpse (Type 3)

Figure 3.5: Each VAP outperforms the other on many video segments. each dot shows the relative accuracy & cost between two VAPs on one video segment.

Together, these findings cast doubt over the current VAP evaluation methodology:

Key Takeaway: *Empirically testing and comparing VAPs on some specific video workloads can be incomplete.*

3.3.3 Discussion

Our measurement has shown that if a VAP is evaluated on only a handful of videos, the results may fail to reveal its true performance range and variance in a target scenario. An immediate response is "why not using a better test dataset?"

Why not using a representative dataset Intuitively, with a set of "representative" videos per scenario, we can get the most common VAP performance by testing VAPs on these videos. Unfortunately, this solution is impractical for two reasons. First, cameras deployed at different locations or future locations will likely generate video workloads with different content characteristics beyond those captured by the empirical tests. Second, since video analytics applications are continuously evolving, representative workloads do not yet exist. Thus, these tests might overestimate/underestimate the VAP performance and lead to wrong choice of VAP in deployment.

Why not using a larger dataset Testing a VAP on a larger number of videos might offer a more complete view of its performance range and variance. Yet a "just adding data" approach will provide little insight on performance distribution on videos outside of the test dataset, and why a VAP's performance varies across videos.

3.4 Key Concept: Achieving Performance Clarity in VAP Evaluation

Different from prior work that evaluates VAPs using only empirical tests, we propose a new methodology for VAP evaluation: **achieving performance clarity**. The goal of performance clarity is to not only identify a VAP's performance under a wide range of video content, but also characterize how video content characteristics affect its performance. This produces a comprehensive and transparent assessment of VAP performance. In the following, we first present the key concept behind performance clarity and its benefits, and then discuss potential solutions to achieve performance clarity.

Terminology	Definition
Video content features	Features that measure content-level characteristics of a video (e.g., avg object speed). See §3.5.2.
Performance Clarity (PC)	Comprehensive performance assessment of VAPs under different video content features. See §3.4.1.
PC Profile ($\mathbb{P}_v$)	A lookup table that maps video content features to performance of VAP v (e.g., Figure 3.6)
Cost-saving Strategy	A particular heuristic to save computer/network cost. See §3.5.1.
VAP Primitives	A set of cost-saving strategies that seek to reduce same type of redundancies. See §3.5.1.

Table 3.3: Definition of terminologies used in Yoda.

To facilitate the discussion below, Table 3.3 summarizes the key terminologies and notations used by our work.

3.4.1 Defining Performance Clarity

The *performance clarity* (PC) of a VAP defines *how video content features affect the VAP's performance*[3]. Formally, PC of a VAP v is a lookup table $\mathbb{P}_v$ that maps from a point x in the space of video content features to v's performance (in cost and accuracy) on videos that match x. This is illustrated by Figure 3.6. Compared to existing evaluations that are either incomplete (e.g., single-scenario tests in Figure 3.3) and/or ambiguous (e.g., high performance variability in Figure 3.4), PC offers a comprehensive and clear characterization of VAP performance and its variation.

The key insight behind PC is the following. It is the *VAP performance's dependencies on video content features* that cause the VAP performance variabilities. As these content features vary across videos (in the same scenario), so does VAP performance. To illustrate this, we featurize each video in our coverage dataset along four content features (more features discussed later in §3.5.2), and plot in Figure 3.7 the Pearson's correlation coefficients between individual features and cost of VAPs when keeping accuracy between 0.9 and 0.95 (to avoid

3. While performance clarity reveals correlations between content features and VAP performance, it does not equal to interpretation of DNNs or VAPs.

PC Profile of VAP v

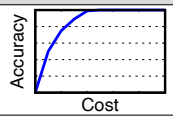
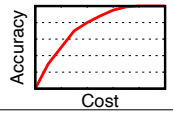
Video Content Features			Performance (cost-acc curve)
average object speed	…	% of video frames containing objects	
[1.0, 1.5]	…	[0.0, 0.25]	
[1.5, 2.0]	…	[0.5, 0.75]	
⋮			

Figure 3.6: An abstract illustration of a VAP v's performance clarity (PC) profile. Compared to either testing a VAP on few videos or reporting its performance distribution over many videos, this profile provides a more complete picture of the VAP's performance by describing its relationship with video content features, which drastically reduces the ambiguity of performance compared to those in Figure 3.4.

cost variance caused by accuracy variance). We single out the impact of each feature by restricting other features to a small range less than 50% of their respective value ranges. The results show a strong correlation between each VAP's performance and the content features.

Benefits of PC A VAP v's performance variation and its content dependency come from the v's design, *i.e.* they are *inherent* to v. Thus v's PC profile ($\mathbb{P}_v$) can offer useful insights on its design and deployment. Below are two usage cases.

1. To **estimate v's performance on any target video**, we can directly combine $\mathbb{P}_v$ with the content feature distribution of the video, which can be quickly obtained by scanning through the video. The computation cost is significantly less than running v on the video (verified in §3.6).

2. To **identify when one VAP outperforms another**, we can directly compare two VAPs' PC profiles to identify in which parts of the content feature space is one VAP better. Again there is no need to run VAPs on any video.

Later in §3.6 we use these two tasks to evaluate the accuracy and benefits of our PC

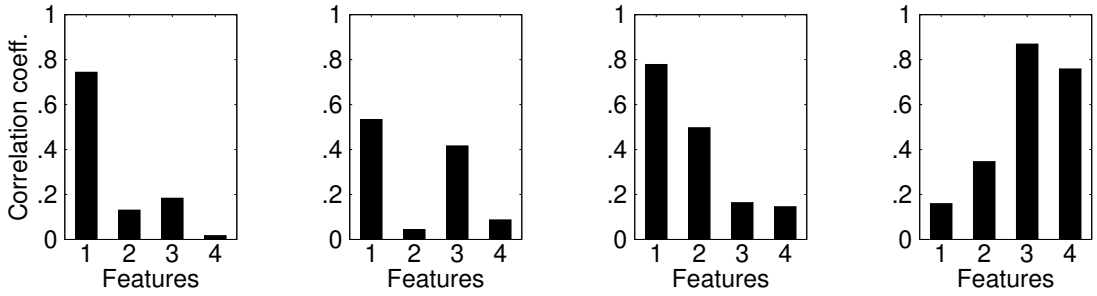
profiler Yoda.

3.4.2 Baseline Solution: Feature-based Profiling

Building an accurate PC profile is challenging. A straightforward solution is to create a corpus of videos that span all combinations of relevant content feature values, and test VAPs on these videos. Unfortunately, this can be prohibitively expensive due to the complex relationship between VAP performance and content features. Specifically, our measurement study (*e.g.* Figure 3.7) lead to two observations.

- *Heterogeneity impact of features:* Different VAPs are affected by *different* sets of features. For example, VideoStorm is sensitive to average object speed (f_1) but not per-object area (f_3); yet NoScope is highly sensitive to f_3 but not f_1.
- *Combinatorial impact of features:* A VAP can be affected by *multiple* features. For instance, Glimpse is highly correlated with the features of object speed (f_1) and fraction of frames with objects (f_2), and AWStream is sensitive to f_1 and f_3. Therefore, *it is insufficient to test VAPs on videos that vary along only one feature at a time.*

Thus, to cover all possible feature value combinations, we need $O(n^{|\mathbb{F}|})$ videos, where $\mathbb{F}$ is the list of content features and n is the number of possible values per feature. To put it into perspective, let us assume that there are 7 content features, each having 4 distinct value buckets (e.g., low, median, high and very high), and we need three 30-second videos to measure VAP performance for each of the 4^7 feature value combinations. These are not overestimation: there are at least 7 content features that might affect DNN accuracy or VAP performance (see §3.5.2), and in our dataset we split each feature in four buckets as well. The resulting dataset would be over **400 hours**, much longer than any VAP test datasets ever created. Since many VAPs do not reduce compute cost, evaluating their performance on this hypothetical dataset would take 400 hours even when using one NVidia GTX Titan X GPU card running the state-of-the-art object detector at 30fps [126].



(a) VideoStorm (b) AWStream (c) Glimpse (d) NoScope

Figure 3.7: Strong correlation between video content features and VAP performance. The four features are: (f_1) *avg. object speed*, (f_2) *% of frames with objects*, (f_3) *10%ile of per-object area*, and (f_4) *avg. confidence score per object*.

3.4.3 Proposed: Primitive-based Profiling

Instead of profiling a VAP as a monolithic entity, we modularize it into multiple primitives (§3.5.1), each of which can be profiled separately. The rationale is two-fold.

1. **A primitive is affected by fewer features than a VAP.** Each primitive only leverages, and is thus affected by, a particular set of video content characteristics. For example, many VAPs reduce video frame rates to save cost, and its efficacy depends only on temporal-related features like object speeds. Yet these features have little impact on "orthogonal" techniques like image downsizing or model compression.

2. **Primitives have independent impacts on a VAP's performance.** As we will show in §3.5.1, the performance of a VAP can be approximated by multiplying the performance of each primitive when other primitives are set to their corresponding most accurate, expensive strategies. In other words, these primitives can be profiled individually, based on which the full VAP performance can be constructed.

Reducing profiling cost Since each primitive is profiled using *only* the video features relevant to its cost-saving strategy, the VAP profiling overhead can be drastically reduced, from $O(n^{|\mathbb{F}|})$ to $O(n^{|\mathbb{F}_1|} + n^{|\mathbb{F}_2|} + \cdots) = O(n^{\max_i |\mathbb{F}_i|}) \ll O(n^{|\mathbb{F}|})$, where $\mathbb{F}_i$ is the feature set related to the i^{th} primitive. Using primitive-based profiling, our eventual dataset consists

VAP	*Temporal pruning*	*Spatial pruning*	*Model pruning*
VideoStorm[262]	✔ (uniform sampling)	✔(quality downsize)	✔ (model selection)
NoScope[139]	✔ (diff-triggered)		✔(specialization)
AWStream[261]	✔ (uniform sampling)	✔(quality downsize)	
Glimpse[70]	✔ (diff-triggered)		✔(fixed tiny model)
Vigil[264]	✔ (diff-triggered)	✔ (region cropping)	
Chameleon[134]	✔ (uniform sampling)	✔(quality downsize)	✔(model selection)
VideoEdge[128]	✔(uniform sampling)	✔(quality downsize)	
DDS[203]		✔ (region cropping)	
EAAR[162]	✔ (diff-triggered)	✔ (region cropping)	
Reducto[155]	✔ (diff-triggered)		
WEG[228]			✔(specialization)

Table 3.4: Modularizing some example VAPs into primitives.

of only **67.5 minutes** of videos, more than two orders of magnitudes less than that of feature-based profiling (400 hours)!

3.5 Yoda: Practical VAP Profiling

We now describe our design of Yoda, the first VAP benchmark to achieve performance clarity. Yoda builds a PC profile for each VAP, by applying the aforementioned primitive-based profiling. In the following, we first present how Yoda modularizes a VAP into independent primitives (§3.5.1) and chooses content features and benchmark videos to profile each primitive (§3.5.2), followed by two core functions offered by Yoda: VAP profiler and VAP performance predictor (§3.5.3).

3.5.1 Modularizing VAPs into Primitives

A VAP may employ one or more *cost-saving strategies* to reduce redundancies in video frames, pixels, and DNN parameters. Observing this inherent modularity, Yoda categorizes these strategies into three *primitives* (see Table 3.4).[4]

4. Some prior work also reduces redundancies across multiple concurrent queries [133] or camera streams [131]. We leave them to future work.

- **Primitive #1: Temporal pruning** drops frames to reduce inter-frame redundancies using at least two strategies. *Uniform frame selection* (e.g., [261, 262]) uniformly samples a fraction of frames for further analysis and then carries over their detected objects to future unsampled frames (e.g., via object tracking). It works well if neighboring frames are similar. *Trigger-based frame selection* (e.g., [139, 70]) skips frames until a heuristic (e.g., significant difference between frames) signals potential arrivals of new objects. It works well when most frames have few objects of interest.
- **Primitive #2: Spatial pruning** reencodes video to reduce redundancies among pixels. Specifically, *image quality downsizing* (e.g., [261, 134]) reduces the video quality (e.g., from 1080p to 360p), which still achieves high accuracy if objects are large. Another strategy, *region cropping* (e.g., [264, 203]), saves bandwidth by encoding only pixels relevant to the task. It can be very effective in, for instance, traffic videos where most vehicles/pedestrians appear small.
- **Primitive #3: Model pruning** leverages the fact that videos often have specific object classes/scenes (e.g., traffic videos contain mostly vehicles/pedestrians with static background), and trims the full DNN to reduce compute cost while still achieving high accuracy. *Model selection* (e.g., [262, 134]) picks a simple yet accurate DNN model from a few pre-trained models with various capacities. *Model specialization* (e.g., [139, 228]) trains a smaller DNN just for particular scenes/objects and if it fails, falls back to the full DNN.

Finally, for each primitive, Yoda also defines an **oracle strategy** that does no cost reduction: 100% frame selection (for temporal pruning, original video quality (for spatial pruning), and full-size DNN (for model pruning). Since the primitives essentially trade accuracy for cost savings, these oracle strategies serve as the most accurate yet most costly strategies.

Independence across primitives As different primitives seek to remove *agnostic* redundancies in video/model, we empirically observe that individual primitives affect VAP performance independently. For instance, the efficacy of spatial-pruning strategies is largely dependent on object sizes/shapes, whereas the efficacy of model-pruning strategies depends

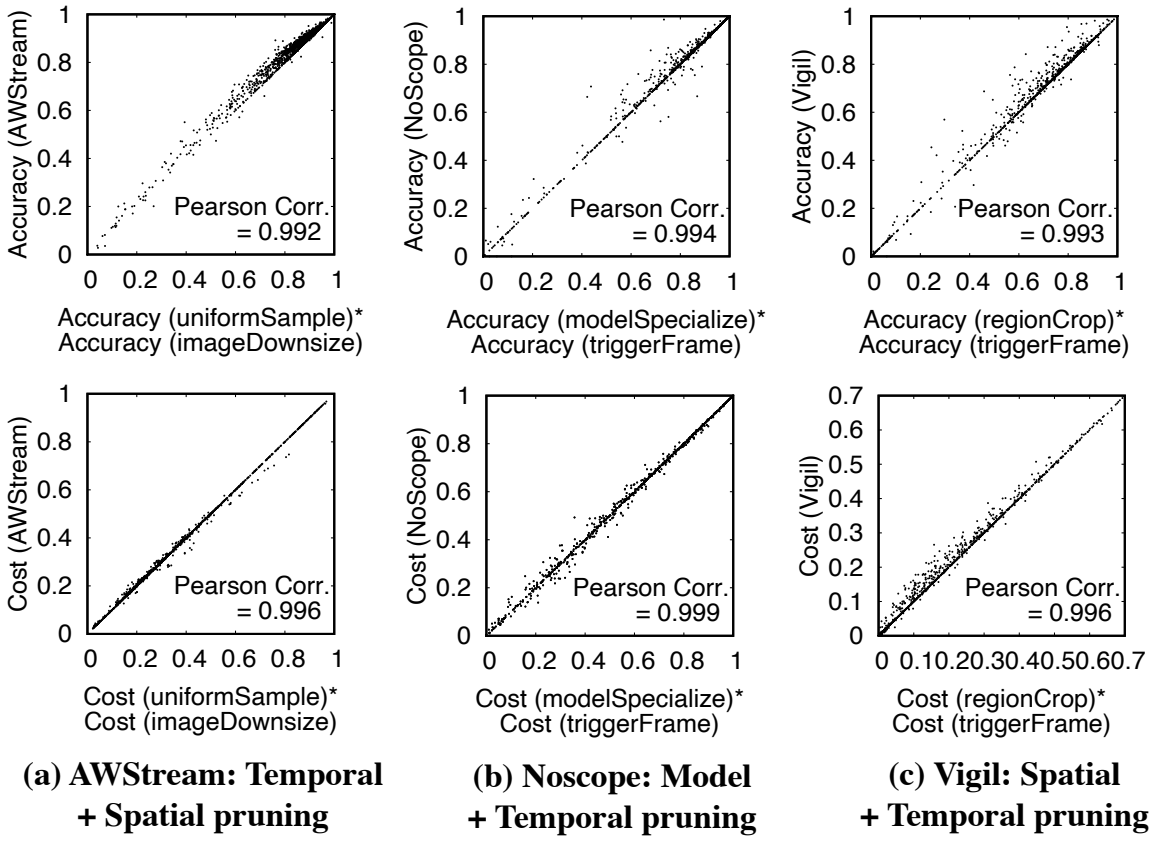


Figure 3.8: Empirical validation of the cross-primitive independence on AWStream, NoScope and Vigil: VAP performance can be approximated by the product of individual cost-saving strategies' performance. Each dot is a video segment in our dataset. Table 3.5 validates the independence property on more strategy pairs.

on the scene complexity or skewness in object class distributions, both of which are agnostic to object sizes/shapes.

Figure 3.8 and Table 3.5 empirically validate the property of cross-primitive independence on existing VAPs. For a VAP v, we first measure performance of each individual strategy by replacing other strategies with their respective oracle strategies. For instance, we measure the performance (cost and accuracy) of v's spatial-pruning strategy by running it on all video segments in the coverage set while setting v's temporal-pruning primitive to the oracle strategy (full frame rate). We then compare the performance of the full VAP and the *multiplication* of performance of its individual primitives. We do so using Pearson's

	Model pruning + Temporal pruning			
	$M_1 + T_1$	$M_2 + T_1$	$M_1 + T_2$	$M_2 + T_2$
Accuracy	0.994	0.997	0.991	0.994
Cost	1	1	1	0.999
	Spatial pruning + Temporal pruning			
	$S_1 + T_1$	$S_2 + T_1$	$S_1 + T_2$	$S_2 + T_2$
Accuracy	0.992	0.993	0.989	0.993
Cost	0.996	0.987	0.999	0.996
	Spatial pruning + Model pruning			
	$S_1 + M_1$	$S_2 + M_1$	$S_1 + M_2$	$S_2 + M_2$
Accuracy	0.955	0.914	0.993	0.999
Cost	1	1	1	1

Table 3.5: Independence property between any pair of strategies from model-pruning strategies (M_1: model selection, M_2: model specialization), temporal-pruning strategies (T_1: uniform sampling, T_2: trigger-based frame selection), and spatial-pruning strategies (S_1: image downsizing, S_2: region cropping). Each value shows the Pearson's correlation coefficient between the performance (accuracy or network cost) when the two strategies are combined and the product of the performance when each strategy is used separately. The high correlations suggest the cross-primitive independence is common.

correlation. Using this methodology, Table 3.5 shows that the independence property largely holds on different pairs of strategies from two distinct primitives. Figure 3.8 shows three concrete examples (AWStream, NoScope and Vigil), where each VAP's performance (both accuracy and cost) closely matches the multiplication of its primitives.

We acknowledge that the cross-primitive independence is empirical and there can be exceptions to it. For instance, when spatial pruning downsizes video frames to an extremely low resolution, no object can be detected regardless of the temporal pruning strategy. In this case, the efficacy of temporal pruning is affected by spatial pruning, though this is unlikely to occur in practice as VAPs aim to maintain a high accuracy.

Nevertheless, we believe cross-primitive independence property is still valuable. By breaking down each VAP to individual primitives (strategies) each related to a subset of content characteristics, we can dramatically reduce the cost of profiling VAPs in an exponential feature space. Likewise, developers of new strategies can apply the same method (of Figure 3.8) to verify if the independence assumption holds.

Video content feature		Definition
Per object	*object speed*	the reciprocal of IoU between the bounding boxes of the same object detected in two consecutive frames
	object area	the bounding box size of each object divided by the frame size
	confidence score	the confidence score of each detected object given by the full DNN
Per frame	*total area of objects*	fraction of pixels covered by all object bounding boxes in a frame
	object count	the number of objects per frame
Per second	*object arrival rate*	# of new arrival objects per second
Per segment	*% frames with objects*	percentage of frames containing objects

Table 3.6: Summary of video content features

3.5.2 Selecting Benchmark Features and Videos

Following the above discussion, Yoda profiles a VAP by first profiling its individual primitives and assembling them to construct the full VAP profile. To profile a primitive, Yoda first selects its associated video features and video datasets.

Feature selection We first create a set of 43 *candidate* content features based on 7 general content-level features (summarized in Table 3.6) known in the computer vision community to influence object detection accuracy (e.g., [160, 97]) and potentially VAP performance. Among them, 6 features are defined either per object, per frame, or per second. We pair them with 7 statistics per video segment: mean, standard deviation, and {10, 25, 50, 75, 90}th percentiles. Thereby, together with one per-segment feature (i.e., *% frames with objects*), each video segment can be represented by 43 content features.

For each primitive, we then select the subset of features (from the candiate set) that correlate with its strategies. Specifically, we pick features that have strong correlations (over 0.3 absolute Pearson correlation, a threshold suggested in [189]) with at least one strategy of the four VAPs studied in §3.3. Here we intentionally leave out three VAPs (AWStream, Reducto, DDS) and use them as a holdout to test the generalizability of Yoda (§3.6.2). To avoid selecting strongly correlated features while capturing as many distinct factors as possible, we iteratively select a new feature only when it has a low correlation with those

Primitives	Selected features
Temporal pruning	*% of frames with objects*, *avg. object speed*, *avg. confidence score*
Spatial pruning	*% of frames with objects*, *avg. total area of objects*, *10%ile of per-object area*
Model pruning	*10%ile of per-object area*, *avg. confidence score*

Table 3.7: Yoda selects a subset of features for each of the three primitives, from the 43 candidate video features.

already selected. Table 3.7 summarizes the selected features of each primitive. These features can characterize the PC profiles of existing VAPs at a sufficient fine granularity. We observe only diminishing improvements with more features. That said, Yoda can be expanded with more features as more VAPs are developed.

Video selection For each primitive, Yoda selects a subset of video segments from our coverage set (§3.3) to cover all of its feasible[5] feature value combinations. We first evenly split the range of values per feature into $n = 4$ *feature value buckets* (we use feature value and feature value bucket interchangeably). For each combination of feature values, we pick at most $k = 4$ video segments from our coverage set. n and k can be increased if more videos are added. As a result, Yoda selects 29 minutes of videos for temporal pruning, 19 minutes for spatial pruning, and 21 minutes for model pruning.

We should stress that the goal of video selection is *not* to be representative of a certain scenario (in fact it includes videos from different scenarios); instead, it finds videos to cover each important feature value combinations that heavily influence VAP performance. This process enables PC profiling which ultimately helps produce accurate performance estimation of any particular scenario and workload (explained in §3.4.1). On the other hand, Yoda meets this goal with only a small fraction of the coverage video set, because there is a highly uneven distribution of content features across video segments (e.g., highway traffic videos contain mostly fast objects).

Potential selection bias and mitigation The features selected by Yoda might be biased,

5. Some feature value combinations may be infeasible; e.g., large per-object area but small total area of objects.

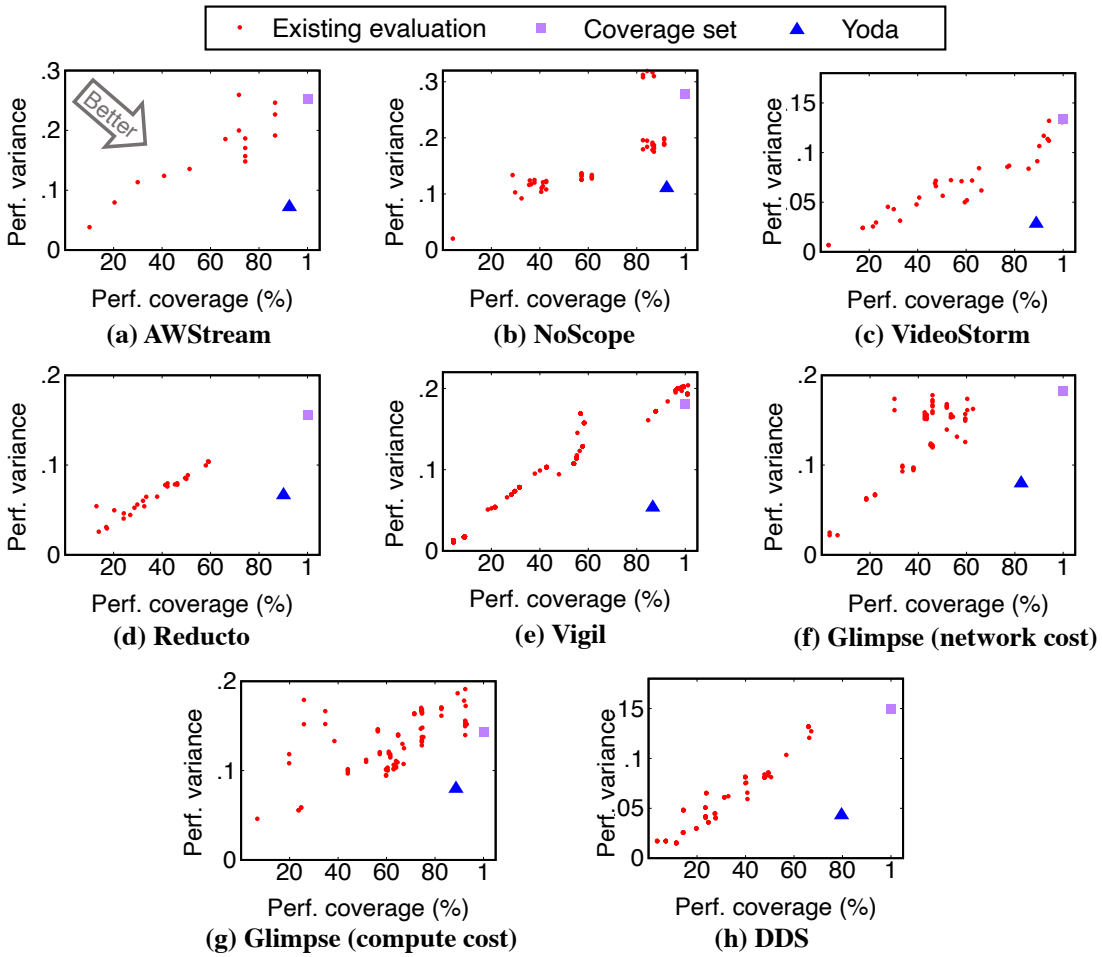


Figure 3.9: Yoda achieves a much higher level of performance clarity (higher coverage and lower variance), compared to existing evaluation methods. A high coverage means Yoda reveals both good and bad performance of a VAP, whereas a low variance means Yoda accurately estimates a VAP's performance on new videos.

since we only pick the features relevant to the existing four VAPs. We partially examine Yoda's generality by showing that it can successfully profile AWStream, the VAP held out from our feature/video selection process (§3.6.2). As a future topic, Yoda could be extended to additional and future VAPs by applying the above feature/video selection process.

3.5.3 YODA's Workflow

Using the proposed primitive-based profiling, Yoda offers two key functions for its users: *VAP profiler* that produces a PC profile $\mathbb{P}_v$ for each VAP v, and *VAP performance estimator*

that directly estimates v's performance on a target video using $\mathbb{P}_v$ without the need to run v on the video.

In the following, we use $v = (t, s, m)$ to denote a VAP, with t, s and m being its temporal-pruning strategy, spatial-pruning strategy, and model-pruning strategy, respectively. The PC profile of $v = (t, s, m)$ is a lookup table $\mathbb{P}_v$ (or $\mathbb{P}_{t,s,m}$) that maps a feature value combination x in the feature space of $\mathbb{F}$ to the expected performance in accuracy and cost $\mathbb{P}_{t,s,m}(x)$.

VAP profiler Leveraging the property of cross-primitive independence (§3.5.1), Yoda builds the PC profile of $v = (t, s, m)$ in two steps. First, we build a per-primitive profile of each of its strategies. The temporal-pruning profile of v, for instance, is $\mathbb{P}_{t,s^*,m^*}$, where t^*, s^* and m^* denote the oracle strategies (see §3.5.1) of temporal pruning, spatial pruning and model pruning, respectively. That is, we build $\mathbb{P}_{t,s^*,m^*}$ by setting v's spatial and model pruning strategies to their oracle ones and testing it on the benchmark videos for temporal pruning (introduced in §3.5.2). Second, we build the full PC profile as

$$\mathbb{P}_{t,s,m}(x) = \mathbb{P}_{t,s^*,m^*}(x) \cdot \mathbb{P}_{t^*,s,m^*}(x) \cdot \mathbb{P}_{t^*,s^*,m}(x) \tag{3.1}$$

VAP performance estimator In practice, operators often need to estimate a VAP's performance on a new (long) video.

The challenge is that naive featurization will require annotating every object (by human annotation or running a full DNN), which can be painstakingly slow. Fortunately, obtaining the *distribution* of feature values over an entire video does not require accurate results on each single frame. Instead, we show that running a low-cost object detector (e.g., SSD with MobileNet) on aggressively sampled frames can still yield reliable estimate of the overall feature value distribution. For instance, to get the distribution of *per-object area*, we run SSD with MobileNet on 10x uniformly sampled frames to get the area of each detected object and use the distribution of these areas as the result. This way, Yoda can quickly scan a long video and produce reliable estimation of the distribution of each feature value.

Once the feature distribution is known, Yoda then uses $\mathbb{P}_v$ to directly map the feature value distribution to v's performance on the video.

We implement Yoda as a ready-to-use toolkit for profiling and evaluating VAPs, and plan to release the toolkit to the research community. The toolkit provides a shared library (API) for emulating and benchmarking VAPs.

3.5.4 Using Yoda: VAP Emulator & Benchmark Interface

We implement Yoda as a ready-to-use toolkit for profiling and evaluating VAPs, and release the toolkit to the research community. The toolkit provides a shared library (API) for emulating and benchmarking VAPs.

The Yoda VAP API Yoda includes a VAP API built in Python, and the corresponding implementations of the five existing VAPs. The core component is an operator `Filter`, which can be instantiated to perform various cost-saving strategies, such as frame sampling, image cropping, and model compression. `Filter` exposes the following interface:

```
Filter(self, segment, decision, results)
```

where `segment` encapsulates the file descriptor of a video segment and its metadata (e.g., frame rate, original resolution), `decision` encodes the common decisions per frame (e.g., drop frame or not, which DNN to run inference), and `results` is optional and includes the per-frame object detection results (e.g., bounding boxes, confidence) so far.

Using this API, one can implement a VAP as a sequence of `Filters`. Each `Filter` takes as input a `segment` and the `results` and `decision` of previous frames, executes a cost-saving strategy of certain primitive, and subsequently updates the segment (sampling or resizing) and appends any new decision or results. Notably, `decision` also indicates whether to move on to the next segment and which `Filter` to execute the next. Yoda also provides a module to implement the logic to pick a VAP's configuration based on a sample of video.

VAP emulator Each `Filter` can be run by a camera or a server to emulate server-side

VAPs as well as VAPs spanning both a camera and a server (Figure 3.2). When testing a VAP on a video, the emulator sequentially feeds video segments through the filters until the decision indicates the next step. Yoda also allows one to specify the length of a video segment, thus supporting VAPs whose operation on one frame depends on results of previous frame(s). When the test is complete, an *evaluator* calculates the network cost (bandwidth usage to send data between the camera and server), the compute cost (GPU usage), and inference accuracy. That said, Yoda currently does not measure other metrics such as inference delay or support optimizations to reduce delay (such as pipelining).

Benchmark automation Once a VAP is implemented using the provided API, Yoda will create its PC profile automatically using the VAP profiler introduced in §3.5.3. For each primitive, Yoda first replaces the other primitives with their respective oracle strategies (which are themselves specific instances of `Filter` provided in the Yoda API), and uses the VAP emulator to run it on the selected benchmark videos (§3.5.2) to create the per-primitive profile. Finally, Yoda puts all per-primitive profiles together to create the PC profile of the VAP.

3.6 Evaluation

We evaluate the efficacy of Yoda in achieving VAP performance clarity. Specifically, we conduct experiments to answer the following questions:

- Does Yoda achieve higher VAP performance clarity, compared to existing solutions that emprically test VAPs on a corpus of videos? (§3.6.1)
- Is Yoda's primitive-based profiling accurate and efficient? Can it generalize to new VAPs? (§3.6.2)
- Can Yoda accurately predict a VAP's performance on new videos at a low computation cost? (§3.6.3)
- Does Yoda provide new insights for VAP design and deployment? (§3.6.4)

3.6.1 Yoda's Performance Clarity

As defined in §3.4.1, performance clarity aims at providing a comprehensive characterization of VAP performance. Here, we measure the level of achieved performance clarity by two dimensions: *coverage* (the completeness of the evaluation, the higher the better) and *variance* (the ambiguity of the evaluation outcome, the lower the better). The intuition is that an ideal VAP performance evaluation should have high performance coverage and low variance. A high coverage means Yoda reveals both good and bad performance of a VAP, whereas a low variance means Yoda accurately estimates a VAP's performance on new videos. The specific metrics of *coverage* and *variance* are defined as follows. Given a VAP v's PC profile $\mathbb{P}_v$ (measured from our benchmark dataset of 67-minute videos), Yoda first uses $\mathbb{P}_v$ to estimate v's cost at a specific accuracy range ([0.9,0.95]) for all videos in the coverage dataset excluding our benchmark videos. Then, we compute Yoda's coverage as the observed cost value range, normalized by the observed cost value range when testing v on the whole coverage dataset (14.5 hours of videos). Next, we compute the standard deviation of Yoda's cost values as Yoda's variance. Figure 3.9 shows the results of 7 pipelines in the blue boxes: Yoda achieves high coverage (>90%) and low variance (<0.2). The figures show one scenario per VAP, but the conclusion holds in other scenarios.

Yoda vs. existing methods Figure 3.9 also compares the (coverage, variance) results from the traditional evaluation method, which tests the performance on a long video (or a set of videos) from the target scenario (represented by the red dots). For fairness, each test video is no shorter than our benchmark video. We see that the coverage fluctuates significantly across videos and the variance per video is much higher. This confirms that traditional evaluations lead to either incomplete/partial conclusions or ambiguous results (as we have shown in §3.3.2). As a reference point, when the traditional evaluation uses the entire coverage dataset (14.5 hours), the variance exceeds 0.25, again significantly larger than Yoda.

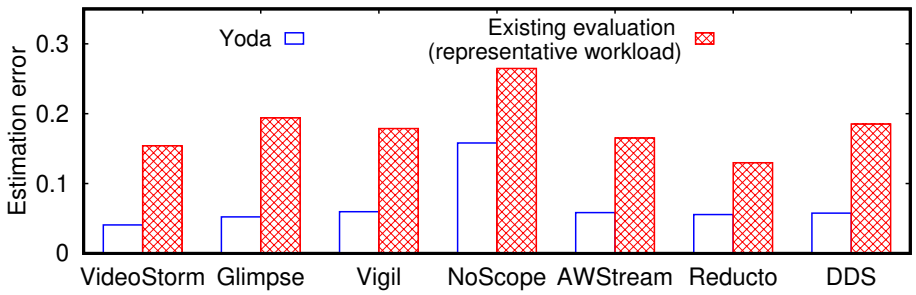


Figure 3.10: Yoda provides more accurate estimation of VAP performance on new videos than traditional profiling using representative workload per scenario.

Microscopic study on VAP performance estimation We take a further step to examine the benefit of elevated performance clarity, using the task of per-video VAP performance estimation. Given Yoda's $\mathbb{P}_v$, we directly estimate a VAP v's performance on any video, and compare it to the ground truth result obtained by running v on the video. Again we keep the accuracy to [0.9,0.95] and measure the absolute difference between the cost value predicted by $\mathbb{P}_v$ and the ground truth, which we refer to as cost "estimation error". As reference, we apply an "traditional profiler" to estimate v's cost in the same accuracy range by running v on a representative long workload under the same scenario of the test video, and compare it against the ground truth.

Figure 3.10 plots the median estimation errors of both Yoda's profiler and the traditional profiler, across all the long videos in the coverage dataset (that are not used for profiling). We see that Yoda's profiler is much more accurate than the traditional profiler at estimating VAP performance on new videos.

3.6.2 Primitive-based Profiling: Accuracy, Cost, and Generality

Yoda's efficiency partly stems from its *primitive-based* profiling, which tests a VAP on only videos that vary along the primitive-related features. To evaluate it, we compare Yoda with an expensive profiler built on the whole coverage set.

Accuracy We measure the discrepancy between the PC profile built on the whole coverage

	Glimpse	Vigil	VideoStorm	AWStream	NoScope	Reducto	DDS
Profile diff (Yoda vs. coverage set)	0.043	0.005	0.048	0.063	0.083	0.030	0.058

Table 3.8: Discrepancy between the PC profile built on Yoda selected videos and the PC profile built on coverage set videos.

set and Yoda's PC profile. The average differences between the profiled performance curves (cost differences at same accuracy levels) are listed in Table 3.8 for each of the seven VAPs, and are all very low. This corroborates our intuition in §3.5.2 that a small subset of videos is sufficient to profile PC, since the feature distribution in the coverage set is highly uneven.

Profiling cost Profiling a VAP is a one-time cost (i.e., no need to repeat unless the VAP changes its design). The computation cost of profiling depends on the VAP design. Intuitively, VAPs that do not optimize/reduce compute cost will incur a higher overhead. Thus we present the result of AWStream, a VAP that does not optimize for compute cost. To profile AWStream, Yoda needs to run the VAP process on ∼72k frames using the full DNN model for object detection, at four different video quality levels and twelve different frame sampling rates. When running on an Amazon EC2 machine (instance p2.16xlarge that has 16 GPUs and costs $14.4/hr), the profiling takes 8.5 minutes and cost $2. Even a VAP, such as VideoStorm, that needs to profile all three primitives takes only 22.2 minutes and cost $5.3.

Generality Can Yoda accurately profile a new VAP not considered by Yoda's feature and benchmark video selection process? As mentioned earlier, we intentionally held out three of the seven VAPs (AWStream, Reducto, and DDS) from the feature/video selection process (§3.5.2). Nonetheless, Figures 3.9 & 3.10 and Table 3.8 show that Yoda achieve similar profiling effectiveness on these holdout VAPs as on other VAPs. While this does not prove that Yoda generalizes to all future VAPs, it does indicates that Yoda might profile new VAPs as accurately as the other VAPs used in its feature selection process.

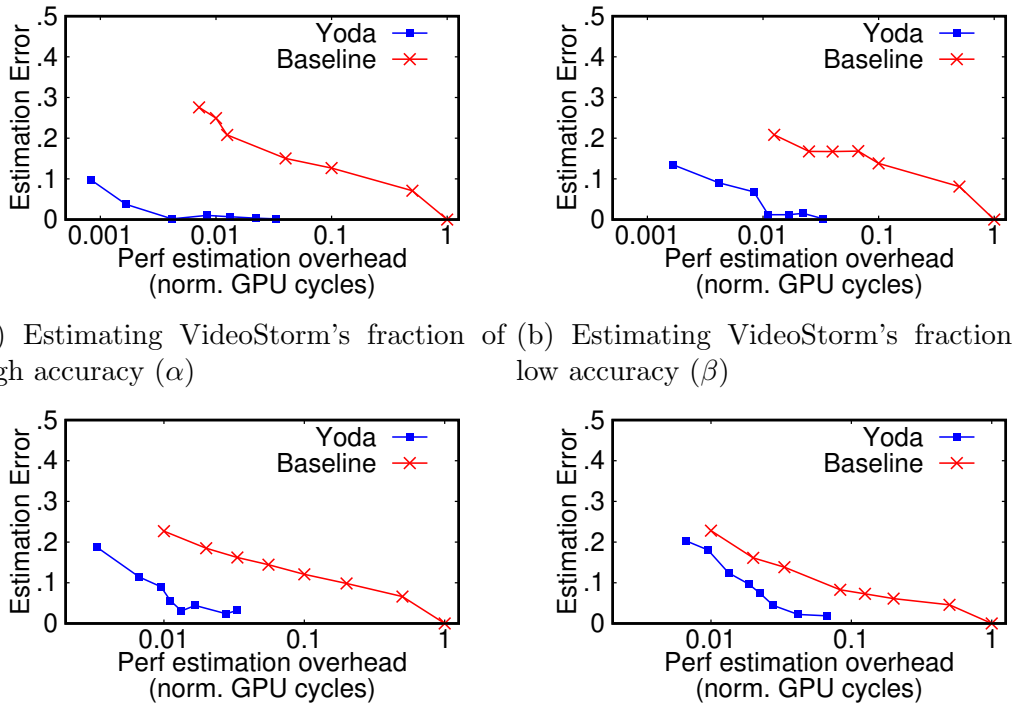
3.6.3 Fast-yet-accurate Performance Estimation

Recall that Yoda offers a useful function of directly estimating a VAP v's performance on any video, without running v on the video. We have validated the quality of performance estimation in Figure 3.10 (§3.6.1), using the task of estimating cost at a specific accuracy range ([0.9,0.95]). Below we provide more results on its estimation accuracy and computation cost.

We consider the task to understand the variability of VAP accuracy throughout a target video. For this we define two metrics on accuracy variability: (1) fraction of video segments whose accuracy is above 0.85, denoted by α; and (2) fraction of video segments whose accuracy is below 0.7, denoted by β. Such metrics are useful in practice since operators often need to maintain accuracy at an acceptable level. We use the accuracy distribution of actually running the VAP on the video as the ground truth and define the estimation error by $|\alpha_{estimated} - \alpha_{real}|$ and $|\beta_{estimated} - \beta_{real}|$.

We also evaluate Yoda against a "resource friendly" baseline that actually runs v on a sample set of video frames, whose estimation accuracy and overhead depend on the sampling rate. Note that as explained in §3.5.3, Yoda's performance estimator also needs to scan a sample set of video frames to measure the video's feature value distribution. Thus its accuracy and overhead also vary with the sampling rate.

Figure 3.11 shows the estimation errors of Yoda and baseline on VideoStorm and AW-Stream, as a function of the estimation overhead (amount of GPU cycles consumed), for 5 hours of dashcam videos (not used during profiling). Here Yoda uses MobileNet-SSD [26] as the cheap object detector to scan the videos. For clarity, we normalize the estimation overhead by the amount of GPU cycles consumed by running each VAP on the full video. We see that Yoda achieves nearly perfect estimation at a much lower cost, i.e., nearly 2 orders of magnitude faster than running the VAP on the video.



(a) Estimating VideoStorm's fraction of high accuracy (α)

(b) Estimating VideoStorm's fraction of low accuracy (β)

(c) Estimating AWStream's fraction of high accuracy (α)

(d) Estimating AWStream's fraction of low accuracy (β)

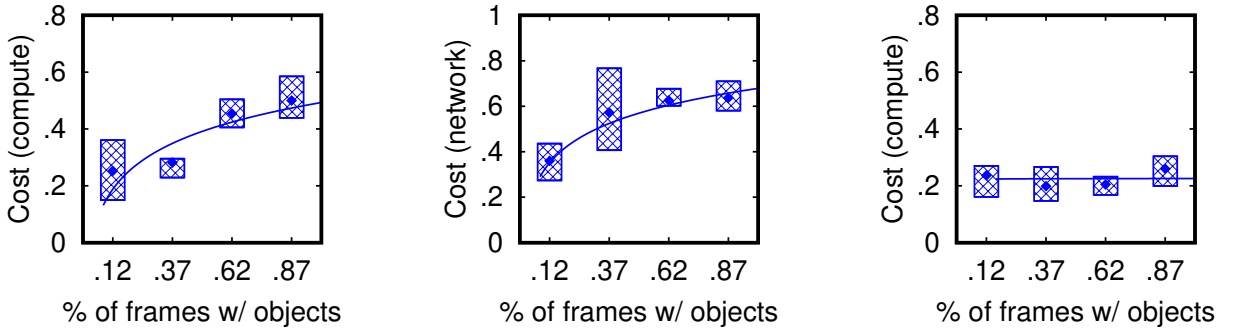
Figure 3.11: Yoda estimates VAP performance faster and more accurately than actually running VAP on the test videos.

3.6.4 Practical Insights for VAP Deployment

By providing a comprehensive profiling on VAP performance, Yoda also identifies new insights for guiding VAP design and deployment. We highlight two concrete use cases here.

Conditional correlations among features Figure 3.12 shows the performance of Glimpse's temporal pruning strategy against two features: x_1 (*% of frames with objects*) and x_2 (*average object speed*). For better visualization, we only show the minimum cost while maintaining accuracy over 0.9 (i.e., a slice of the cost-accuracy tradeoff). Figure 3.12(a) and (b) show that both compute and network costs are strongly correlated with x_1 when x_2 is over 1.6 (which is a typical vehicle speed in highway videos). But when x_2 is below 1.6 (Figure 3.12(c)), the correlation becomes remarkably weaker.[6]

6. A closer look at the selected frames shows that frame difference-triggered selection is no longer effective when the object speeds are so low that the frame difference triggered by their movement can easily be confused

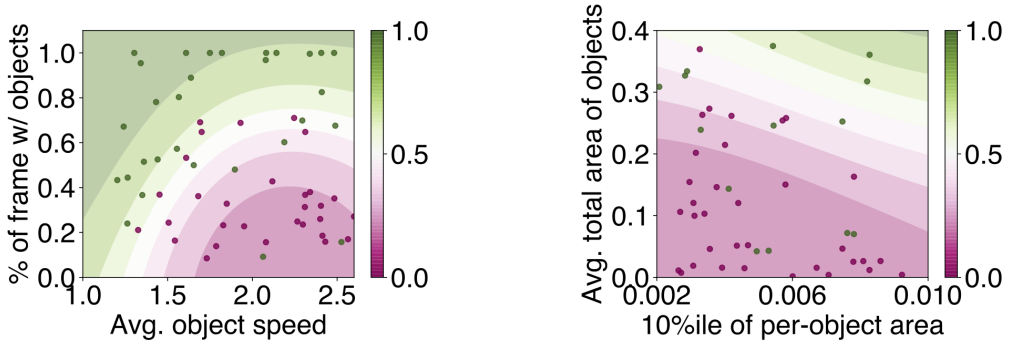


(a) Compute cost v.s. x_1 when $x_2 \geq 1.6$.

(b) Network cost v.s. x_1 when $x_2 \geq 1.6$.

(c) Compute cost v.s. x_1 when $x_2 < 1.6$.

Figure 3.12: Impact of feature x_1 (*% of frames with objects*) on performance depends on the value of feature x_2 (*avg. object speed*). Each box shows the mean and 25$^{\text{th}}$ and 75$^{\text{th}}$ %iles.



(a) Temporal pruning: Uniform vs. frame diff-triggered selection

(b) Spatial pruning: Image quality downsizing vs. region cropping

Figure 3.13: In both primitives, there is no single strategy that fits in all type of content. The coloring indicates that where one strategy is likely better than the other.

This result implies that when testing VAPs that use this pruning strategy (e.g., NoScope, Glimpse), *the traditional method may either miss this correlation* (if most test videos have slow moving objects) or *claim a strong correlation* (if most test videos have fast moving objects). In contrast, Yoda reveals not only both correlation patterns, but also *when* they emerge, which helps to decide if a VAP should be deployed in certain video content.

Informed choices of VAP strategies As a case study, let us consider two temporal-pruning strategies (uniform frame selection vs. frame difference-triggered frame selection)

with pixel differences caused by noises in the background.

and two spatial-pruning strategies (image quality downsizing vs. image region cropping). Figure 3.13(a) shows the *operating regime* of each temporal-pruning strategy: frame difference-triggered selection is better when only a small fraction of frames contain objects and these objects move fast (magenta). Otherwise, uniform frame sampling is better (green). Similarly, Figure 3.13(b) shows that image quality downsizing is likely to be better if the objects are large and occupy more space in frames (green), and otherwise the image cropping strategy is better (magenta). These differences stem from how various strategies interact with videos. For instance, image quality downsizing eliminates redundant pixels *in* large objects of interest (which can be detected with less pixels), whereas image cropping eliminates redundant pixels *outside of* objects of interest by subtracting background.

These results have significant practical implications. For instance, for urban traffic videos during peak hours, AWStream (uniform frame sampling and image downsizing) is better than Glimpse (frame difference-triggered frame selection), because the vehicles appear frequently and in large numbers and move slowly and often in relatively big sizes (crossroad cameras tend to be closer to the road than highway cameras), so it falls in the green regions of both graphs. In contrast, for urban traffic videos during off-peak hours, where many large-size objects move quickly (i.e., magenta in Figure 3.13(a) and green in Figure 3.13(b)), we should create a new VAP that combines Glimpse's temporal-pruning strategy and AWStream's spatial pruning strategy.

3.7 Related Work

3.7.1 Video Analytics Pipelines

Besides the VAPs described in §3.2, there are other VAPs that utilize the same three primitives: temporal pruning (e.g., [162, 250, 192, 61, 242, 213]), spatial pruning (e.g., [162, 252, 177, 213]) and model pruning (e.g., [100, 192, 254, 256, 213]). Some work also reduces the compute/communication cost of computer-vision inference pipelines, through

super resolution (e.g., [248, 229, 73]), splitting the DNN between camera and server (e.g., [131, 237, 113, 96, 124]), DNN-aware cloud/edge resource scheduling (e.g., [113, 138, 202, 161, 260]), cross-camera or cross-application correlations (e.g., [66, 132, 260, 167, 209]), scalable data management and execution frameworks (e.g., [175, 147, 206, 163]), and DNN architectures tailored to balance throughput and accuracy (e.g., [214, 125, 58, 247, 159, 133, 99]). Many of these techniques leverage content-level characteristics, such as the ones we have discussed. We hope that by revealing the importance (and feasibility) of PC, future work can extend Yoda to support these VAPs. While a few prior works have mentioned the issue of performance variability on some VAPs, the results were limited and only based on a handful of video features (e.g., object size [93]). To the best of our knowledge, our work is the first to systematically study (using measurements & building benchmarks) how video content features affect VAP performances.

3.7.2 Edge/Video Analytics Benchmarks

Several benchmarks of video analytics systems have been proposed for various focuses, including throughput of video database (e.g., [116, 206]), video encoding efficiency (e.g., [33, 172]), and shared library to implement video inference pipelines (e.g., [17]). More general benchmarks catered for edge network environments are proposed as well [50, 179, 144]. Also related to Yoda are those benchmarking vision-task accuracies (e.g., [107, 8]) and their tradeoffs with throughput/latency (e.g., [126]). While most benchmarks focus on average performance across images/videos, some did observe that vision models perform differently across content [52] and can be sensitive to video encoding [119] or training data quality [266]. Yoda takes one step further to systematically reveal the influence of video content features on VAP performance. Recent efforts in computer vision similarly demonstrate that features of the test data affect the performance of a classification model (e.g., [49, 105]), though they focus on perturbing the features to improve model robustness whereas Yoda seeks to reveal the hidden relationship between VAP performance and content features.

Traditionally, the systems community has benefited from thorough performance benchmarking of data analytics systems under a wide range of workloads (e.g., [46, 82]), and our work is one example of this line of work in the context of video analytics.

3.8 Conclusion

Conclusion. Our work is a response to the recent trend of building efficient mobile video analytics systems, at the expense of significant performance variability caused by video content dependency. We present a measurement study to shed light on this issue for the first time, and propose the first VAP benchmark that elevates performance clarity (how video content affects performance). Although Yoda only scratches the surface of VAP performance clarity, it is shown to be effective and capable of identifying hidden design tradeoffs.

Limitations. We highlight three key limitations of Yoda that demand future work. First, Yoda is not future-proof—its design could be biased by existing VAPs. It assumes the VAPs process videos frame by frame and the vision task is object detection. This does not always hold: some optimized DNN (e.g., [159]) can directly run inference on encoded videos, although not widely used in VAPs. Second, although the GPU usage is correctly measured, our implementation of VAPs may not incur unnecessary overhead on CPU, so CPU compute cost is not supported. Third, Yoda can only test VAPs that run on one client and one server (covering all VAPs in Table 3.4), but does not yet support multi-stream or multi-query VAPs.

Future Work. Yoda is extensible because we modularize the pipelines into primitives. For other cost-saving strategies that do not belong to temporal, spatial and model pruning, another primitive can be added to Yoda following the same methodology of feature and video selection. For example, we identify two other primitives in existing VAPs, i.e., cross-video pruning which utilizes the content-level redundancies across multiple video streams [131] and cross-query pruning which utilizes similarity across different queries [133]. For each of them,

we could apply similar feature and video selection to achieve the performance clarity. Then we add the selected videos to our current selected video set. However, the independency between primitives may not always hold; when it does not, the profiling cost may increase as the feature space cannot be cleanly separated.

CHAPTER 4

ENABLING PERSONALIZED FACE EDIT PROTECTION

While Chapter 2 and Chapter 3 demonstrate how to integrate ML to physical sensing applications, in this chapter, I study ML deployment in cyber monitoring applications. In physical sensing applications, useful information is extracted from data collected from widely-distributed sensors, so heterogeneity mainly comes from sensors' local environments. In cyber monitoring applications, ML models are developed on online contents that users upload to online social platforms. Therefore, human factor becomes the key source of heterogeneity. Specifically, I focus on how **heterogeneous user needs affect the design and evaluation of ML-based cyber monitoring systems**.

My work considers the pressing problem of detecting unacceptable face edit in online face photos. Using a user study, we find the definition of unacceptable face edit can vary per user and application context, indicating the need for personalized protection. Meeting such diverse user needs is extremely challenging since it requires ML models to accurately recognize face edits in each photo. We address this challenge using a system approach. By integrating the system function of tracking original image copies, we successfully convert the extremely hard problem of recognizing any edit in an image into a feasible ML problem of comparing two images. The end result is an efficient photo moderation tool that allows users to define their own face edit policy and provides personalized protection accordingly.

4.1 Introduction

Social media is a key channel for engaging in cyber social interactions today, and users' self-presentation plays a key part in their online behavior. To achieve a desired presentation of themselves to their audience, users can choose to have their online face images digitally edited or refined in particular ways.

Recent advances in computer graphics and artificial intelligence (AI) have opened the

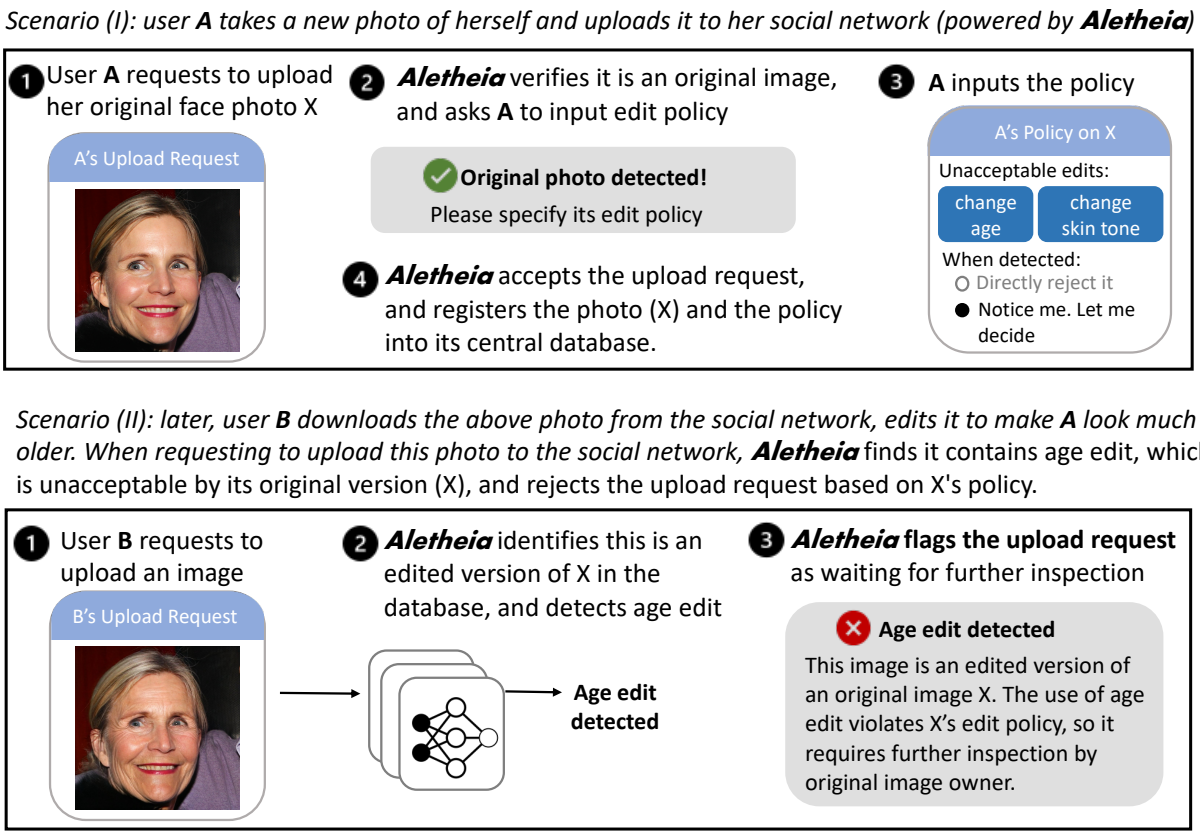


Figure 4.1: We propose *Aletheia*, a tool that provides personalized protection against unacceptable face editing. In *Aletheia*, owners of original face photos specify their willingness to accept or not accept different types of edits to their online face photos. In scenario (I): a user A specifies a policy that disallows changes in age or skin tone. In scenario (II): a bully B tries to share an age-changed photo of user A. It is detected as violating A's face edit policy, and the upload is flagged for moderation per A's policy.

door for fast and easy facial editing in digital photos. Today, numerous image and video applications already include functionality for a variety of high quality facial editing and transformations. With the touch of a button, everyday users can perform subtle touchups on a face photo, or change the person's age and many other facial features, and the results look so realistic that they fool the naked eye. This can be seen from the edited examples in Figure 4.2, produced by today's low-cost face edit tools.

However, as face editing tools grow more common on social networks and photo sharing apps, the negative impact from abusing these tools also grows. As we share our photos online with family, friends and coworkers on a variety of social and professional platforms, any of

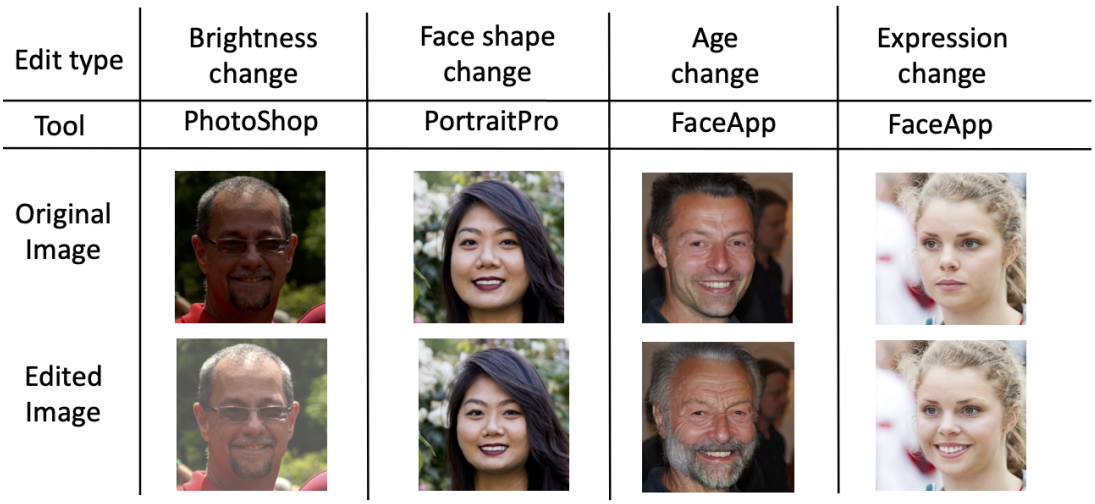


Figure 4.2: Examples of different types of face edits done by common edit tools.

them can be downloaded, edited and then shared for malicious purposes. For working adults, photos from LinkedIn might be edited and then posted to discredit or harass someone in the workplace. For younger users, photo editing is already used for cyberbullying [171], a problem experienced firsthand by a majority of teens [44, 108, 85]. A bully can download photos of a target, modify their facial features, and then share the edited photos online. Even as platforms move to curb bullying [186], experts predict photo editing tools are likely to become 'a potent tool of cyber-stalking and bullying' [143].

There is a clear need for "photo moderation tools" that protect users against any new uploads that contain undesired or unacceptable edits of their online images. What makes this type of image moderation difficult, however, is that the question of what is an "acceptable" photo edit is likely subjective and context-dependent. Some users may enjoy it when family and friends apply silly edits to their photos and share them on their social networks, while others may be carefully curating a public persona using photos, and object to any edits applied to their online photos. *Therefore, any image moderation tool on social networks or photo sharing services must walk a fine line between reliable detection of unwanted face editing, and overly sensitive filters that prevent normal social interactions and collaborations using photo editing tools.*

Despite the need to protect user photos from unacceptable edits, there is a lack of understanding on how users perceive photo edits by others, and how to build photo moderation tools to meet individual user needs. Existing research in the field of human computer interaction mainly examines how users self-edit their own photos (e.g., [173, 238, 238, 63, 150]), but not their attitudes towards edits and reposts of their photos by others. Many have proposed methods to protect online photo privacy with respect to *viewership*, by controlling who can view the photos (e.g., [101, 231, 188, 233, 55, 232]) or adding artifacts to obfuscate faces (e.g., [154, 115, 153]) or other sensitive information [114, 153] on photos. Yet these do not address issues of protecting users from unauthorized and undesired edits of their photos by others. That is the focus of our work.

We note that current content moderation tools also do not meet the above mentioned needs. To date, research on image analysis largely focuses on the problem of *detecting whether a face photo has been manipulated* (e.g., [84, 244]). This is motivated by detection of deepfakes [156, 257], often in the context of AI-generated fake photos/videos that misrepresent public figures (e.g., politicians and celebrities) or fabricate fake news events used in disinformation campaigns. These detectors assume that the original photos are *not available* and mainly operate by detecting specific forms of digital artifacts on the photos left by media editing and generation tools. Under our problem context, these detectors often fail to even detect the fact that a face has been edited, much less the type of face editing or whether it is acceptable to the owner (see Table 4.8, §4.6.3).

Personalized Protection against Unacceptable Face Edits. In this paper, we propose and explore *Aletheia*, a novel photo moderation tool that protects each individual user against the uploading and sharing of unacceptably edited versions of their online face photos. *Aletheia* operates on image hosting services and social media networks to protect their users with regards to their posted face photos. As illustrated in Figure 4.1, *Aletheia* allows owners of original face photos to specify their willingness to accept or not accept different categories of edits to their online face photos. For example, a user A on Microsoft

OneDrive may specify a policy that permits facial retouching on their personal photos, but not changes in age or skin tone. A bully B trying to share an age-changed photo of A is detected by *Aletheia* as violating A's face edit policy, and the upload is flagged for moderation or rejected. By implementing such personalized protection, *Aletheia* provides each user with full agency in deciding who can edit their online photos and which edits are acceptable. We believe this will stimulate healthy social interaction and collaboration enabled by face editing tools, while protecting users from their malicious misuse.

Our Contributions. Our work targets the critical challenge of user-specified moderation of face edits in photos. We hope our work brings attention to this important area, and spurs follow-up work to take us closer to robust photo moderation systems that meet the diverse needs of different users. Specifically, our work makes three contributions.

First, we performed a detailed user study (IRB approved) to understand how users perceive acceptable edits to their online facial images. Responses from 100 participants show significant variance in both general level of acceptance of face edits by others, as well as across specific types of face edits, validating our assertion that face edit protection needs to be *personalized*.

Second, we designed *Aletheia* to implement face edit protection customized for each individual user. We addressed the challenging problem of "recognizing" the type of edits included in an incoming photo, and reconciling them with the original copy's preferences to determine whether the upload is permissible. Such detection of unacceptable edits is feasible, because we developed an efficient and accurate "reference-based" edit recognition mechanism, a significant departure from existing solutions for detecting deepfakes. We implemented a prototype of *Aletheia* and plan to release it to the research community to stimulate further work.

Finally, we use IRB approved methods to evaluate our prototype for efficacy in detecting unacceptable face edits, scalability, and usability.

1. Our user study showed that *Aletheia* can effectively flag image edits that violate user policies – across 9 different dimensions of face edits, *Aletheia* correctly identifies 93.8% of photos that were marked as unacceptable by real-world users;

2. Our data-driven experiments (using 821K original faces and 42K edited images) tested *Aletheia* at scale, showing that it is both accurate (>97% accuracy) and fast (<1s per image) in determining the status of incoming photos (edited or original) and locating their original copies, and can already reliably recognize nine edit types;

3. Our exploration study showed *Aletheia* can effectively recognize in-the-wild face edits submitted by 8 participants (without any instruction on tools and formats);

4. We performed a second user study to examined the usability of *Aletheia*, where the majority of participants of our study stated that *Aletheia* induced a feeling of protection and will use *Aletheia* to protect their photos.

As one of the first proposals to address image moderation through personalized face edit protection, our work contains a number of limitations, which we discuss in §4.8.

All images in our user studies and shown in illustrations in this paper come from the NVLabs Flickr-Faces-HQ dataset [140] and Google's DeepFake Detection Dataset [94, 218], with licenses that grant free use for non-commercial research and educational purposes.

4.2 Preliminaries

We begin by providing background for our work on personalized face edit protection.

Motivations for Sharing Images Online. With record numbers of images posted online daily, social media has become a key channel to communicate and engage in cyber social interactions. When users post images online, particularly of themselves, they consider personal motivations including how they wish to present themselves to online audience. Existing works, especially psychology research, have shown that self-presentation is a significant

motivating factor for decision making in social interactions [54], and more importantly, self-presentation behavior varies across online social networks (OSNs) [75, 89, 103]. Personality traits, self-esteem, experience on the OSN, and audience groups heavily influence how users choose to present themselves to others online. Particularly, depending on who their audience is across different platforms, users may engage in selective self-presentation tailored to that audience.

Motivations and Impact for Posting Edited Images. To achieve a desired presentation of themselves to their audience, some users prefer to have their face images (or "selfies") edited or refined in particular ways [173, 238]. In lieu of changing one's physical appearance, many have turned to digital face editing techniques to alter their appearance online [238, 63, 150]. On the other hand, existing works also show that the types and intensity of which edits are desirable varies across users, with many users preferring their photos remain in the original state without any editing [173]. We note that existing works mainly focus on studying how users edit their own photos but not how users react when their photos are edited and reposted online by *other people*.

The presence of edited photos in OSNs will likely influence their viewers. Recent studies show that observing other users' content on social media sites affects how users wish to present themselves. For example, existing works found that viewing idealized smile images on social media sites may increase dissatisfaction with that user's own facial appearance, lower self-esteem, and increase consideration for cosmetic surgery [222, 68, 104].

Accessibility of Face Editing Tools. Thanks to advances in computer vision, face editing tools are now easily accessible to users in the form of standalone apps (e.g., FaceApp, Facetune) or built-in tools within OSNs (e.g., Snapchat, Instagram). With just a few taps on a screen, non-experts can generate high quality transformations of face images that often look realistic to the naked eye. Today, many face editing tools/formats are available at zero or low cost, allowing users to smooth skin, reshape face, change age, expression, eye color, hair color, etc. Earlier in Figure 4.2, we show examples of face edits using today's tools.

Misuse of Face Editing Tools. Alongside the growth of cyber social interaction, cyberbullying and harassment continue to grow as well, particularly for young women [241, 217]. The accessibility and usability of face editing tools introduces new opportunities for misuse by bad actors. Rather than making benign edits for beautification or humor, face editing tools are now used to harass and bully individuals, resulting in both a violation of privacy and potential damage to mental well-being. For example, numerous free apps can turn any smiling face into a grumpy crying face or add/exaggerate unflattering facial features. Studies on cyberbullying have reported cases where one student pasted another student's face onto intrusive backgrounds and distributed them with friends [151], and others of photo edits to exaggerate embarrassing facial features [171]. Today, these edits require no effort beyond a smartphone and a photo.

Misuse of Deepfakes. Another important trend in cyber social media is the growing misuse of deepfakes, whereby high quality *fictional* images or videos can be generated by a computing device [122]. On numerous occasions, bad actors have leveraged deepfakes to depict a political figure (e.g., House Speaker Nancy Pelosi) or celebrity (e.g., actor Tom Cruise) saying/doing something they never said/did or to fabricate false news events to sway public opinion [109]. As tools for generating deepfakes are easily accessible (e.g., apps like Reface and Wombo already allow users to generate deepfakes from their smartphones), deepfakes are also increasingly used to target everyday users [217, 182]. Given their negative impacts, deepfakes have received significant attentions from government, industry and academic communities, triggering active efforts and campaigns to detect and deter deepfakes [225, 40, 39]), as well as social studies to measure their effects in society [109, 129, 78] and how users spread deepfakes online [41].

Focus of Our Work. We note that very little work has been done to address non-deepfake, edited (face) images produced by other people as a form of cyber social interaction, including how users perceive those edited photos and whether unacceptable edits can be detected by systems or others. That is the focus of our work. Later in §4.4, we discuss in

detail previous attempts at detecting limited types of face edits, why they remain insufficient for protecting users, and propose our new method to detect and moderate various types of edited face images that can be deployed by today's photo hosting services and OSNs.

4.3 Motivation: Need for Personalized Face Edit Policies

As face editing evolves into a double edged sword for cyber social interaction, we see a need to automatically protect our faces against unacceptable edits. To realize such protection, we first need customizable policies to clearly define (un)acceptable face edits for individual users. To better understand individual user's perspectives on (un)acceptable face edits, we conducted a user study.

4.3.1 Our User Study on Face Edit Policies

Our study was approved by our Institutional Review Board.

Participants. We recruited participants via the online crowd source platform Prolific[1]. The survey was designed to take 15 minutes on average, and each participant received $3 as compensation. We collected responses from 100 individuals, among which 53 identify as male (47 female). The majority of our participants are between 18-29 years old (60%), with 22% between 30-39, 16% between 40-49, and 2% between 50-59 years old. Two-thirds (64%) of the participants actively upload images/videos showing people's faces at least a few times a month, and 75% of all participants indicated they observe edited images online at least a few times a month or more frequently.

Procedure. The participants completed an online survey study consisting of multiple choice and free response questions about their preferences for (dis)allowing certain face edit types, and perceptions of privacy and security when posting images online.

When compiling edit types, we categorize today's face editing types into five groups by

1. https://www.prolific.co/

their effects [84], which are shown in Table 4.1. The first category is "global retouch," which refines image-level perception without changing content. Example edit types are adjusting brightness and applying basic Instagram filters. The second category inserts stickers, memes, accessories onto images as decorations. The next three categories create semantic changes to faces and facial attributes (e.g., adding makeup, changing age, gender appearance, hairstyle or face shape), facial expressions (e.g., make the person smile or cry), and identity (e.g., faceswap to another person).

Category	**Example Edit Types**
Global retouch	change photo brightness; add filter effect
Insert sticker	add sunglasses/emoji
Change facial attributes	add makeup; change age, gender appearance, hair, face shape
Change expression	non-smile $\rightarrow$ smile, smile $\rightarrow$ crying
Change identity	swap two faces

Table 4.1: Overview face edit types.

Task. We asked participants to suppose they are sharing a photo online to a platform similar to Facebook or Instagram. The platform has face edit detection capabilities, like *Aletheia*, and can (dis)allow specific edits as specified by the user. We then asked participants a series of multiple choice and free response questions about their perceptions and concerns regarding posting/editing images online, how they would set their own policy, such as with *Aletheia*, and their preferences for how the platform should act.

Conditions. We presented two scenarios (in random order): the photo is viewable to *friends and family only* or viewable to the *public*. For each scenario, we showed 10 examples of edit types (based on 2 photos of the same person from Google's DeepFake Detection Dataset [94, 218]), and asked them to rate how likely they would allow each type of edit. Examples of edit types are listed in Table 4.1. For each edit, we asked participants to indicate which level of editing they would typically allow, ranging from never allow edits to allow any edits. After evaluating all edit types individually, we presented several new edited images, based on the same original image they previously evaluated, and asked them to select which

edited images they would allow. Figure 4.3 shows example questions from our study.

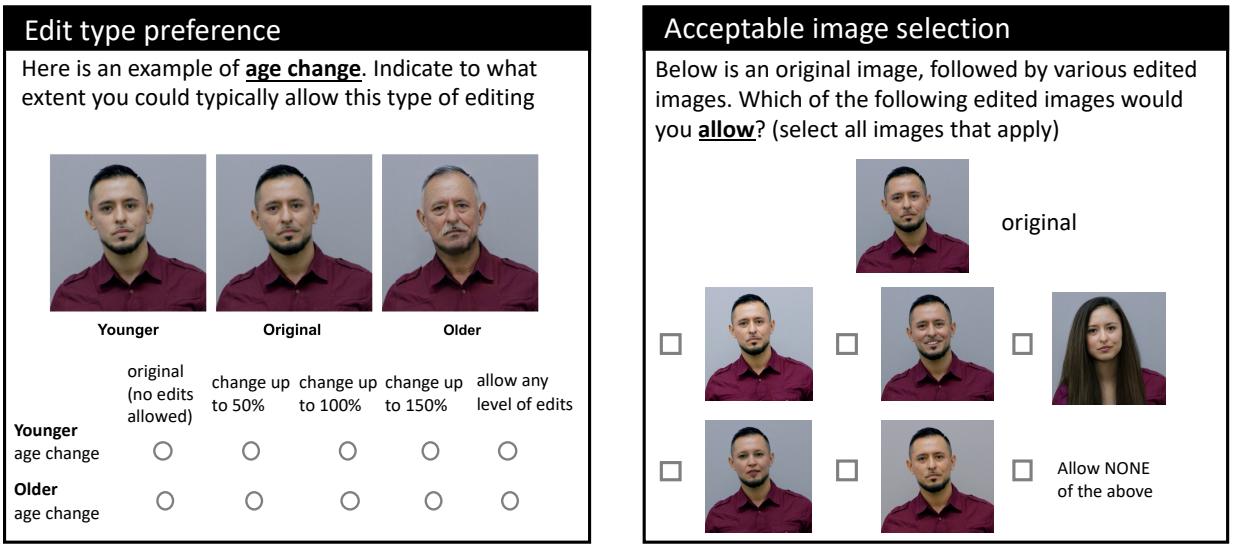


Figure 4.3: Example user study questions. (Left) Participants were asked about their preferences for each edit type. (Right) We showed participants new examples of edited images at once and ask which they would (dis)allow.

4.3.2 Key Result: The Need for Personalized Face Edit Protection

Results from our study show a need for personalized policies that allow users to decide which edits they deem (un)acceptable for photos shared online.

Finding #1: Users have varying levels of (un)acceptance for different face edits. In line with previous work, participants showed significant variation in their image editing preferences. On average, participants would (dis)allow *any* amount, of 1/2 of the editing types presented, with 8 participants allowing *all* edits and 3 not allowing *any* edits. However, the type and acceptable threshold of specific edits allowed or disallowed, and the reasons why, varied significantly. We found no consistent patterns in the types of edits acceptable across participants.

Over half (53%) of all participants indicated they were somewhat or very concerned that other people may edit their images and repost them online. To further explore the participants' reasoning behind their individual selections, after evaluating both scenarios,

Reason	Example
None/ Very Few Edits	"I **wouldn't want anybody editing my photos, whether I know them or bot** [*sic*]. It feels intrusive."
Specific Edits Only	"It doesn't matter to me who can see it, I **just don't want specific edits** done to me." "I think I **would allow more non-invasive stickers (e.g., hat, glasses)**, as the assumption is that the edits would be good natured" "Sometimes **some edits end up making the pictures weird**"
Allow More Edits from Friends/ Family	"**Hopefully friends and family would only make changes for the better** and would be more considerate wifh [*sic*] their choices." "Well I would find it **funny if my friends did some of those edits** to me **but I would be a bit annoyed if a random person** did some of those to my photo."
Allow More Edits from General Public	"... I think I would have a bit **more fun doing more extreme edits for public viewing** just because potentially more people will be seeing them rather than friends and family who already know what I look like."
General Indifference	"I feel like **whatever you show in private for the most part you should be able to show in public**"

Table 4.2: Participant reasons for personalized preferences.

we asked participants to explain in their own words. The reasons expressed fall into 5 general categories: prefer no edits from anyone ever, prefer only specific edits (regardless of the audience), would allow more edits from friends/family, would allow more edits for public photos, or general indifference in whomever may be editing photos. Some examples of these reasons are shown in Table 4.2. When asked if they would go back and change any of their responses for either scenario (public vs. friends and family only), 72% of participants expressed confidence in their original choices, suggesting people are actively aware of their individual online privacy preferences.

Finding #2: As new face editing tools and types are developed, users expect the platform to adapt to cover them. We also asked participants about how their policy preferences may change over time, and how they feel the platform should react to edited images. Over 60% of participants expressed concern about new editing methods not originally covered in their policy, and would like to be able to adjust their policy as new editing methods are developed, or their personal preferences change over time. Upon

detection of an edited image that violates their policy, a majority of participants (87%) want the platform to notify them. However, participants showed an even split regarding whether the photo should immediately be removed from the platform or wait until the user can review the photo and decide if they want it removed. Further, participants also showed a 2:1 split regarding whether the platform should flag as many potentially edited images as possible, even if some are false positives, as opposed to a less strict implementation that may miss some unacceptable edits.

4.4 Key Concept: Personalized Face Edit Protection via Reference-based Edit Recognition

Our user study made an important observation: *individual users vary significantly in deciding what face edits are acceptable to their online photos.* Thus an image moderation system should act according to personalized policy that defines what are and are not acceptable face edits. Unfortunately, none of today's image moderation systems/tools support such type of protection. This motivates us to design *Aletheia*, an image moderation tool for personalized face edit protection.

4.4.1 Design Goals and Assumptions

We design *Aletheia* to achieve two key goals:

- allow users to specify (and update) their personalized policy on unacceptable face edits on their original photos;
- identify images containing unacceptable face edits (relative to their original version) and trigger subsequent actions, e.g., rejecting upload requests, further review by the original photo's owner.

Assumptions. With the goal of preventing the use of face editing tools in online harassment and cyberbullying campaigns, we design *Aletheia* to target users who are familiar with

everyday technology, but not experts or strongly motivated adversaries. That is, we do not anticipate users who analyze the internal design of *Aletheia* to craft customized adversarial attacks. Also, our current implementation of *Aletheia* targets common face photos (i.e., front-shot of a single face as illustrated by the figures throughout the paper).

4.4.2 Why Existing Face Edit Detection Fails

One key requirement of face edit protection is to effectively detect *if* and *how* an image has been edited. There is considerable effort by security and computer vision communities to develop techniques that detect face edits in photos. Here we categorize those efforts and show why they are insufficient.

Verifying embedded image signature. This solution seeks to provide integrity guarantees of an image via cryptographically-secure digital signatures. Such signatures would be applied to images at their creation (e.g., by smart cameras), and validated by the consumer (e.g., digital photo frame). This approach severely restricts flexibility needed by content editors who need to refine and edit images before it is ready for consumption. However, this type of tools [145, 255, 176] have no flexibility of defining what edits would be detected, so it is not suitable for personalized face edit detection.

Examining image edit logs. Some face edit tools can log specific edits into the image's metadata [38]. Online services can extract the edit log and reject images with unauthorized edits. However, this approach only works if there is consensus and standardization by all edit tools, and is easily bypassed by any edit tools that disregard them.

Inspecting image visual content. Numerous tools, from graphics-based to deep learning-based, try to detect face edits in a target image by studying its visual content. These are generally referred to as image manipulation detection and/or deep-fake detection. These approaches are *reference-free*, i.e., they do not assume the knowledge of the original image (or even if there is one) and operate directly on the target image. Along this

line, existing works mostly target specific types of edits (e.g., faceswap [156, 257], image splicing [127]) or a specific tool (e.g., adobe photoshop [244]). They function by detecting the presence of edits based on digital artifacts introduced during face editing. These include visual artifacts (e.g., resolution inconsistency [157], temporal inconsistency [110]), behavioral anomalies (e.g., inconsistent head pose and expression [39], abnormal eye blink pattern [156]), and DNN model artifacts (e.g., GAN fingerprints [259] and abnormal neuron behaviors of face recognition models [243]).

These methods are unsuitable for our problem context for multiple reasons. First, they target *detection* on the presence of face edits rather than *recognition* of different types of edits. Second, given their reliance on artifacts left by specific types of content editing, they are highly specialized and do not generalize across edit types and tools. Finally, the efficacy of artifact-based detectors will continue to drop as edit tools evolve and eliminate digital artifacts (e.g., removing GAN fingerprints [191], introducing eye blink patterns).

Preventing image edits. Finally, some recent works propose to add artifacts/perturbations to images, so that they interfere with certain face edits produced using deep learning models (i.e., faceswap [221]). However, these are highly customized towards specific types of edits, and thus are not suitable for personalized face edit protection.

4.4.3 Design Intuition

Different from existing efforts, we design *Aletheia* to effectively detect if and how an image has been edited, by applying *reference-based* face edit detection and recognition. Specifically, *Aletheia* is an image moderator system run by an online photo service provider (e.g., OSNs or photo sharing services) to protect original face images in the service[2]. *Aletheia* achieves face edit protection by operating on a pair of images: a target image x and its original/unedited version x_0. Upon receiving a request to upload a face image x, *Aletheia* first identifies

2. Multiple services can also cooperate to expand the protection coverage across services. In this paper, for simplicity, we consider a single service.

whether x is an original photo or an edited one. If x is edited, *Aletheia* locates it original version x_0 from its database of original faces, and compares x_0 and x to identify whether x contains any unacceptable face edits defined by x_0.

Aletheia differs from existing works by applying reference-based edit moderation. It presents three key advantages.

- *Aletheia* allows users to define personalized protection rules for their original face images.
- *Aletheia* transforms the extremely challenging problem of recognizing types of face edits in any single image (i.e., reference-free edit recognition) to a manageable problem of comparing two images and recognizing their differences. To the best of our knowledge, there is no reference-free system that can recognize the broad types of edits applied to an image.
- Since we design *Aletheia* to recognize face edits by extracting and comparing semantic face attributes (e.g., age, expression) of the original and edited images, *Aletheia* is agnostic to edit tools and can scale to new edit types.

4.5 *Aletheia*: Image Moderation for Personalized Face Edit Protection

4.5.1 System Architecture

Aletheia consists of three components: **(1) a face edit policy manager** that allows each user, when uploading an original face photo, specify their policy that defines unacceptable face edits and the subsequent system action upon detecting such edits; **(2) an image inspector** that for each incoming image x into the system, inspects the image to determine whether it is an original image, if so, asks to input policy, and if not, locates its original version x_0; and **(3) an edit recognizer** that compares x and x_0 to determine whether x contains any unacceptable edits defined by x_0. In addition to these three components, *Aletheia* also maintains an internal **database** that stores the registered original face photos and their edit policy. Figure 4.4 illustrates *Aletheia*'s operation pipeline under two scenarios.

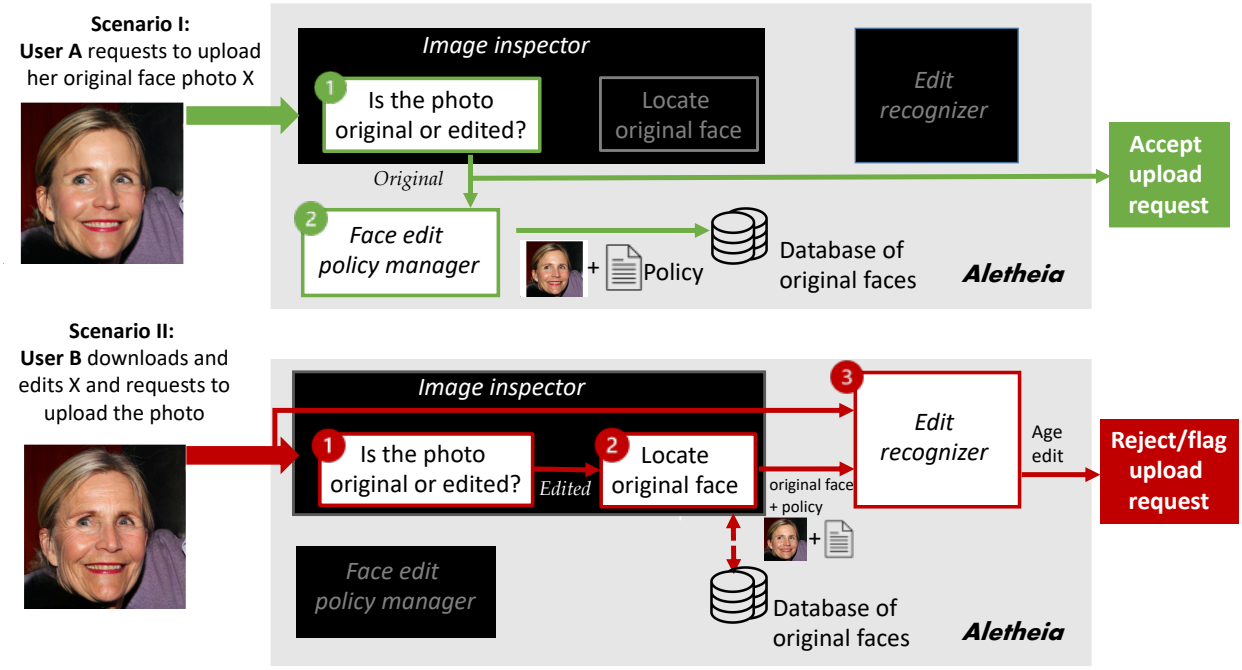


Figure 4.4: Overview of *Aletheia*'s operation when users request to upload original (scenario I) and edited photos (scenario II). For scenario II, we illustrate a case where the photo contains unacceptable edit (i.e., age edit). If the edited photo does not contain any unacceptable edit, *Aletheia* will accept the upload request.

In scenario I, the input image to *Aletheia* is an original face photo, and we mark the operating system components in green. In this case, first the **image inspector** verifies the input image is an original face photo, then the **policy manager** interacts with the user to define the edit policy on this photo, and registers the photo and its policy into the database. The end result is that the photo is accepted.

In scenario II, the input image is an age-edited face photo, which violates the user policy that disallows age edit, and we mark the operating components in red. In this case, the **image inspector** first detects that the input image is an edited photo and then moves to locate its original face photo from the database, together with its edit policy. The **edit recognizer** then compares the original and edited photos to recognize edits, and applies the edit policy to determine whether there is any unaccepted edit. If so, the upload request is either rejected or flagged for further moderation (depending on the policy). If not, the upload request is accepted.

In our current design and implementation of *Aletheia*, we mainly focused on **image inspector** and **edit recognizer**, which we describe next. For the **policy manager**, we employed a simple design that promotes the user to input a binary choice on each edit type (i.e., accepted or unaccepted), guided by examples of edited images. Clearly, *Aletheia* would benefit largely by a carefully designed interactive interface that provides clear interpretation on face edits and guides users in defining their policy. We leave this to future work.

4.5.2 The Image Inspector

We now provide a detailed description of *Aletheia*'s Image Inspector. As illustrated by Figure 4.4, the inspector consists of two components: (i) deciding whether the input image is original or edited, (ii) locating the original face photo of an edited image.

Is It Original or Edited?

For an incoming image x, the inspector first determines whether x is an original face photo or an edited copy. This takes a two-step process to boost accuracy while lowering computation cost.

(1) Fast filtering using "image provenance". *Aletheia* first applies a "provenance" based filtering mechanism to identify edited photos by inspecting the image's metadata. Specifically, the image hosting service running *Aletheia*, when publishing any image on the service, embeds its provenance data (i.e., a string identifying the image's original copy) into the image's metadata[3]. This applies to both original face photos and their edited versions published by the service (see Figure 4.5). As a result, any face images downloaded from the service should already contain the provenance data. Assuming that normal use/edit do not

3. Intuitively, the provenance data should be embedded into images such that it remains intact through normal photo usage/edit. We found that embedding into the metadata that already exists internally in the image file is a viable solution. This is widely supported by image formats such as JPEG, DNG, PNG and TIFF, etc, and does not modify the visual content of the image. Removing or modifying these metadata is possible but requires applying manual efforts or specific tools [223]. In fact, we tested 10 editing tools considered by our work and found that none of them remove or modify the metadata field.

remove the provenance data, *Aletheia* can simply inspect the provenance data in the target image x to decide whether it is edited.

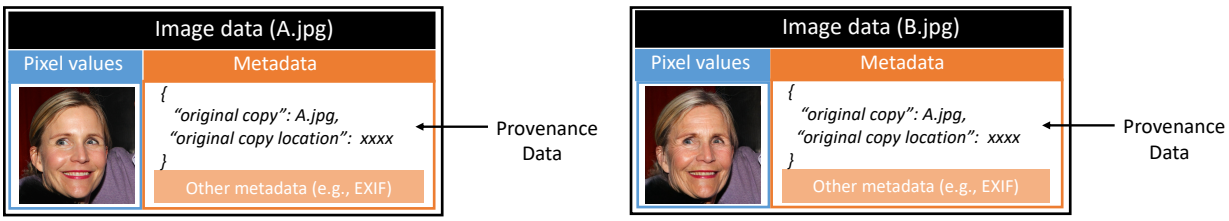


Figure 4.5: *Aletheia* embeds the provenance data into an image metadata before publishing it on the image hosting service. This applies to both original face photos and their edited versions published by the service.

(2) Verifying original photos by perceptual hashing based image search. Our design also addresses the scenario where the provenance data is intentionally or accidentally removed/modified, so that the image "claims" to be an original face photo. Upon receiving such an image, *Aletheia* runs a verification program that compares the image's visual content with those stored in the original face database. Intuitively, a new original face photo should be reasonably different from those stored in *Aletheia*'s database of original photos. That is, the image's minimum perceptual distance to those in the database should be higher than some threshold. Such distance can be computed by a database-level image search/comparison.

We designed our inspector to realize this concept, and more importantly, to address two key challenges in practical deployment, which we summarize below.

The first challenge is to make the image search process works at scale. The mass scale of today's social networks or photo services makes the database-level image search intractable if we compare images in the raw pixel level. Instead, *Aletheia* applies perceptual hashing [20] to convert each image's content into a single compact representation. Since the hash of an image captures its perceptual content (e.g., color, shape, and texture), the hamming distance between hashes of images can be used to approximate their perceptual distance [20]. These perceptual hashes are also fast-to-compute and compact (e.g., 64 bit), making them a good fit when searching over hundreds of millions, or even billions, of images. Specifically, our

implementation employs *pHash* [20], a popular perceptual hash known be robust against image rotation, skew, and compression. Furthermore, to speed up the search, our prototype applies a ball-tree based structure [166] to index the hashes in our database, and runs an k-nearest neighbor search for the target image's hash on the ball tree. The cost of the search is on the order of $O(dlogN)$ [166], where d is the dimension of the hash (i.e., 64 bit), and N is the number of images in our database of original faces.

The second challenge is to flag edited photos even when the edits introduce large changes of diverse types. To do so, *Aletheia* builds multiple content representations to expose similarity between an original photo and its edited versions. Our current implementation uses two representations, whole-image and background-only, and computes two hashes for each image, one per representation. For each representation, *Aletheia* runs the database-level image search to identify the candidate image that is the most similar to the target image. This produces two candidates. Finally, *Aletheia* compares each of the two candidates to the target image x in terms of raw visual content, measured by Structural Similarity Index (SSIM) [268], a widely-adopted measure on perceived change in image visual content. If any of the two SSIM score is higher than a threshold θ_{SSIM} (i.e., the target image is sufficiently similar to a known original face), the target image is an edited photo; otherwise, it is an original photo.

Locating the Original Copy

After detecting the target image x is an edited face photo, *Aletheia* moves to locate its original copy x_0 from the database of original faces. If the target image has no provenance data, the previous step should have already found its original copy via database image search. But if the target image comes with the provenance data that announces its original copy x_0, *Aletheia* needs to verify whether the declared x_0 is indeed the original copy. Again this verification is done in two steps. First, *Aletheia* checks whether x and its declared original copy are sufficiently similar in visual content, again by computing their SSIM. If their SSIM

$> \theta_{SSIM}$, the verification passes. If not, *Aletheia* moves to apply the database image search to locate the true original copy, using the same threshold θ_{SSIM}. Finally, *Aletheia* configures θ_{SSIM} based on the detection false positive rate. Intuitively, the choice of θ_{SSIM} should ensure that, with a very high probability (e.g., p=99%), any two distinct original photos should not be verified as a pair of an original photo and its edited version. Thus, *Aletheia* sets θ_{SSIM} as the top p-percentile value on the SSIM score of image pairs sampled from the database.

4.5.3 The Edit Recognizer

Given a target image and its original version, *Aletheia*'s edit recognizer applies a semantic based image comparison algorithm to recognize edits on the target image. As shown in Figure 4.6, *Aletheia* first extracts, for each image, a set of relevant semantic attributes (e.g., those related to the unacceptable edits defined by the original photo's edit policy), and compare the attributes of two images. Example attributes include age, identity, facial expression, face shape, skin tone and hair color.

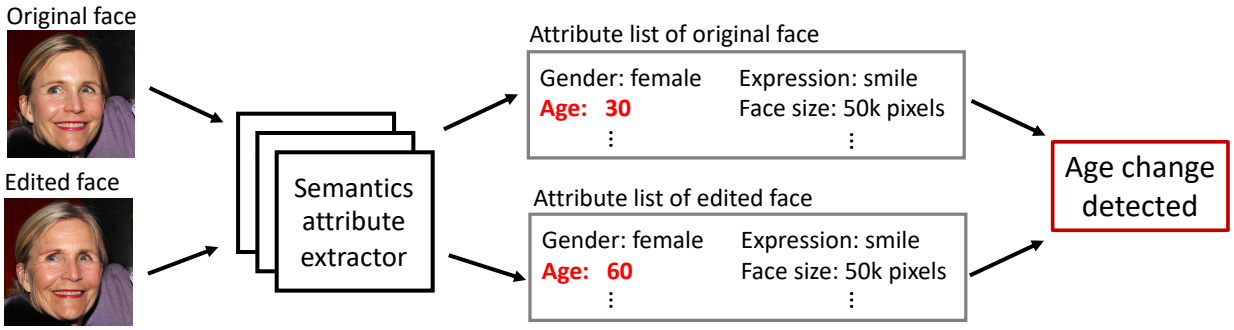


Figure 4.6: *Aletheia* detects the edit types by comparing semantic attributes of target image and its original copy.

Focusing on extracting semantic attributes displayed by individual images, *Aletheia*'s edit recognizer achieves the following properties for practical deployment:

- It is modular and scalable. Given a list of unacceptable edits, the system runs a set of

stand-alone attribute extractors corresponding to these edits. As new face edits appear, the system can be expanded by adding new attribute extractors.

- It is tool-independent by identifying natural semantics of images rather than tool-specific features.
- It is agile against advancement of face edit tools, and remains effective even as edit tools perfect themselves to produce "natural" images without any artifacts.

Modular semantic attribute extractors. We built face attribute extractors leveraging existing (and ongoing) efforts on predicting semantic attributes from images (e.g., [169, 267, 220, 224, 47, 92, 9]). Table 4.3 lists 9 attributor extractors employing off-the-shelf, pre-trained models and the decision heuristics to detect 9 types of edit, which are included in our prototype. We leave the design of a broader set of attribute extractors to future work.

Among these nine extractors, the first eight are learning based, leveraging pre-trained deep learning models to extract a person's age, identity vector, facial expression, gender, skin tone, hair color, face shape, and the presence of eyeglasses. For faceshape, we treat more than 5% changes in the detected face as an indicator of the edit. For eyeglasses, we use the face segmentation result to identify whether any eyeglasses are present on the face. Finally, the last attribute extractor (brightness) is a graphic metric computed directly from the image pixel values. While there are various measures of brightness, we choose a common metric that calculates the average of R+G+B values across all the pixels, producing a value between 0 (dark) and 255 (bright). This is represented by **Avg.U** (x) in Table 4.3. We empirically set a threshold of 10 to detect any "reasonable" brightness change. Since some local edits on the image, e.g., changing hair color, could also change the **Avg.U** (x) value, we add an additional requirement of more than half of pixels having brightness changes.

Edit Type	Semantic Attribute Extractor	Decision Heuristics
Change Age	Pre-trained **Age** Classifier [220]	| **Age** (x_0) - **Age** (x_t) | $\geq$ 5 years
Faceswap	Pre-trained Face Verification Model [224]	**Identity** (x_0) $\neq$ **Identity** (x_t)
Change Expression	Pre-trained **Expression** Classifier [47]	**Expression** (x_0) $\neq$ **Expression** (x_t)
Change Gender appearance	Pre-trained **Gender Appearance** Classifier [92]	**GenderApp** (x_0) $\neq$ **GenderApp** (x_t)
Change Skin tone	Pre-trained Face Segmentation Model [9] + Skin Color Identifier$^{\diamond}$	**FaceColor** (x_0) $\neq$ **FaceColor** (x_t)
Change Hair color	Pre-trained Face Segmentation Model [9] + Hair Color Identifier†	**HairColor** (x_0) $\neq$ **HairColor** (x_t)
Change Faceshape	Pre-trained Face Segmentation Model [9]	| **FaceSize** (x_0) - **FaceSize** (x_t) |/**FaceSize** (x_0) $\geq$ 5%
Add/Remove Eyeglasses	Pre-trained Face Segmentation Model [9]	**EyeglassDetected**(x_0) $\neq$ **EyeglassDetected**(x_t)
Change Brightness	**U**: per-pixel brightness (i.e., mean of its RGB color coordinates)	|**Avg.U** (x_0) - **Avg.U** (x_t) | $\geq$ 10 & more than half of the pixels having brightness change

Table 4.3: *Aletheia* uses 9 modular attribute extractors to recognize 9 types of edits on x_t, using its original copy x_0 as reference. $^{\diamond}$*We follow the human skintone color palettes defined by [23] to define skin color.* †*We follow the hair color palettes defined by [14] to identify hair color.*

4.5.4 Prototype of Aletheia

We built a prototype of *Aletheia* in Python, leveraging existing libraries on attribute extractors. We plan to release our code to the research community. Our modular implementation provides extensibility – one can add new semantic attribute extractors or experiment with other image similarity metrics and image hashes for database search.

Implementation details. In perceptual hashing based image search, our implementation employs *pHash* [20], a popular perceptual hash known be robust against image rotation, skew, and compression. Furthermore, to speed up the search, our prototype applies a ball-tree based structure [166] to index the hashes in our database, and runs an k-nearest neighbor search for the target image's hash on the ball tree. The cost of the search is on the order of $O(dlogN)$ [166], where d is the dimension of the hash (i.e., 64 bit), and N is the number of images in our database of original faces.

4.6 Evaluation

4.6.1 Overview of Evaluations

We evaluated *Aletheia*'s effectiveness and usability using four forms of evaluations. All studies were approved by our Institutional Review Board. Below we provide a high-level summary of these evaluations with a preview of their results.

I. Testing *Aletheia*'s effectiveness using a user study. We used the data collected by our user study in §4.3 (100 participants) to examine *Aletheia*'s ability in detecting face edits that each participant labeled as unacceptable. *Aletheia* closely matched each participant's preference: it flagged 93.8% of unacceptably edited images, and accepted 87.2% of acceptable edits.

II. Testing *Aletheia*'s effectiveness and complexity at scale. We tested *Aletheia* at scale using a large-scale face dataset that contains both original faces (821,358 images) and their edited versions (42,547 images). We found that *Aletheia*'s inspector component can accurately (>97% accuracy) detect whether an image is original or edited and for each edited image, find its original copy; *Aletheia*'s edit recognizer component can recognize many face edit types at high accuracy (86-99.4%). Furthermore, *Aletheia* is computational efficient – even when the database contains 750K original face photos, the total processing time is $<1s$ per image, when using a commodity server with just a single CPU and a single GPU.

III. Exploring *Aletheia*'s effectiveness against in-the-wild face edits. We also tested *Aletheia* on face photos freely edited by real users. Across 415 edited face images submitted by 8 volunteers, *Aletheia* achieves a high accuracy (94.2%) in detecting edits and a reasonable accuracy in recognizing edit types (¿75%), except for subtle skin tone edits.

IV. Testing *Aletheia*'s usability using a second user study. Finally, we conducted a second user study (100 participants) to assess users' perceptions on *Aletheia* protecting their online photos and their willingness to use *Aletheia*. The greater part of the partici-

pants indicated that Aletheia induced a feeling of protection and they would use *Aletheia* in practice. They also provided suggestions on how to further improve *Aletheia*, which helped us identify future works in §4.8.

4.6.2 Testing Aletheia*'s Effectiveness with User Studies*

We performed a user study to verify the ability of *Aletheia* to flag image edits that violate user policies, and whether such decisions match human decisions.

Procedure. Using data from the online user study presented in §4.3, we further examined how well *Aletheia* flags (un)acceptable edited images on behalf of each participant. Specifically, as shown in Figure 4.3 (right), after participants defined their preferences regarding specific edit types, we presented them with several edited images at once and asked which they would (dis)allow. For each participant, we used their responses to define a binary policy per edit type, i.e., the edit type is acceptable or unacceptable. A score 0 means the edit is unacceptable, and any score above 0 means the edit is acceptable. Next, we showed each participant several new edited images ("age", "gender appearance", "expression", "brightness", "skin tone", "faceswap") based on the same original image they previously evaluated and asked them to decide which edited images they would allow. This produced a set of user decisions on (un)acceptable face edits which we will use to evaluate *Aletheia*.

We received 576 participant decisions on edited images. After visually inspecting these decisions against the corresponding participant's policy, we found that some decisions do not match the policy. Such mismatch mainly happened when participants "incorrectly" accepted images, predominately those containing skin tone or facial expression edits, while their policy disallowed them. For example, when skin tone change was subtle, participants may not notice the change; when adding a smile to a face, participants may find it harmless and thus accept the image. Such mismatch could be avoided in future with a more sophisticated policy framework that allows users to clearly express, and later refine, their preferences. For consistency, we removed those decisions that clearly do not match the policy. This produced

406 valid human decisions for our following experiment.

Next, for each edited image labeled by the 406 human decisions, we ran *Aletheia* based on each participant's policy to determine whether the image violates the policy or not. We then compared the decision made by *Aletheia* to the corresponding human decision.

Results. Our study produced two key findings.

Aletheia *can accurately flag edited images that users disallow* – Table 4.4 reports how *Aletheia*'s decisions match the participants' decisions. Out of participant decisions labeled as unacceptable (156 out of 406): 93.6% were also detected by *Aletheia* as unacceptable (i.e., it violates the participant's policy); out of participant decisions labeled as acceptable (250): 87.6% were detected by *Aletheia* as acceptable. We see that *Aletheia*'s decisions match participants' decisions on edited images well, especially in flagging unacceptable edits.

		Aletheia's decision	
		unacceptable edit	acceptable edit
Participant's decision	unacceptable edit (156 images)	**146 (93.6%)**	10 (6.4%)
	acceptable edit (250 images)	31 (12.4%)	**219 (87.6%)**

Table 4.4: Comparing decisions made by *Aletheia* on an edited image to those made by participants in our user study. *Aletheia*'s decision is based on the policy specified by each participant, i.e., when detecting that an image violates the user policy, *Aletheia* marks it as unacceptable. The participant decision is made during our user study by the participant viewing the image and marking it as either acceptable or unacceptable.

Sources of mismatch are subtle change and overlap of edits – We also studied the sources of mismatch between *Aletheia*'s decision and participant decision. Out of 156 participant decisions on unacceptable images, 10 (6.4%) were not detected by *Aletheia*. We found that these all came from a single skin tone edited image which contains very subtle change of skin tone (this edit was generated automatically by the skin tone template of Meitu[4]). *Aletheia*'s skin tone detector failed to spot this change given its subtleness. Next, out of 256 participant decisions of "acceptable", 31 (12.4%) were flagged by our tool

4. http://www.meitu.com/

as "unacceptable." By inspecting those images, we found that all of these false alarms are results of natural overlaps between edits, i.e., edits of age and gender appearance also triggered our faceshape detector, while faceswap-based edits also triggered our faceshape, expression and gender appearance detectors.

Discussion. Our results are encouraging and demonstrate the initial feasibility of *Aletheia*. Yet they also confirm the previous observation that our current definition of edit types remains too broad to build and configure accurate detectors. We need better characterization and interpretation of edit types so users can clearly define fine-grained edit protection policies, from which *Aletheia* can build more precise and thus accurate detectors.

4.6.3 Testing Aletheia*'s Effectiveness at Scale*

We then evaluated how *Aletheia* would perform on image hosting services with a large population. Since user study at this scale is intractable, we evaluated *Aletheia* on large-scale datasets that include both original face photos and their edited versions. We designed experiments to verify (1) the accuracy of *Aletheia*'s image inspector, (2) the accuracy of *Aletheia*'s edit recognizer, and (3) the computation cost for both components. Next, we first describe the face datasets, then present each experiment and their results.

Our face datasets. Evaluation of *Aletheia* requires a dataset covering a wide range of face edit types and tools. As none of existing datasets provides edit type labels, we built our own dataset by altering real face images with state-of-the-art editing tools using automatic scripts.

- *Original faces* – By combining several public datasets, we built a dataset containing 821,358 face images of 30,461 identities. This combined set ensures diversity and includes a wide variety of images from celebrities and normal people. For consistency, each image only contains a single face. A detailed summary of the dataset is listed in Table 4.5.

Sub dataset	# of identities	# of images	Type (Source)
CelebA [170]	10,177	202,599	celebrities (Internet)
FFHQ [140]	unknown	70,000	normal people (Flickr)
DeeperForensics [135]	unknown	1,000	faces (YouTube video, using the first frame)
FF++ [218]	unknown	1,000	faces (YouTube video, using the first frame)
IMDB-WIKI [219]	20,284	523,051	actors (IMDb, Wikipedia)
UTKFace [265]	unknown	23,708	faces (Internet)
Total	**>30,461**	**821,358**	**normal people & celebrities**

Table 4.5: Our **original face** dataset includes 820K+ face photos from both normal people and celebrities.

- *Edited faces* – We built and ran scripts to generate edited images from 1000 original face images (randomly sampled), producing 42.5K edited images labeled by the edits. As detailed in Table 4.6, our dataset covers 12 edit types and 10 editing tools, including both commercial software/app (PhotoShop [21], PortraitPro [22], FaceApp [10]) and open-source models (StarGAN [76], AttGAN [118], GANimation [208], HRFAE [258], OpenCV sticker code [18], FF++ [218], DeeperForensics 1.0 [135]). Each image contains a single type of edit. For each edit type, we generate edited images using at least two different tools. Due to variations in both the number of available tools and their edit options, our edited face dataset is not balanced across edit types. To avoid bias, we up-sampled under-represented types when reporting results that aggregate over edit types.

To produce our edited dataset, we built and ran scripts to generate edited images from a set of 1000 original face images (randomly sampled). For instance, we used Android-ViewClient [2] and BlueStacks [4] to emulate human interactions with FaceAPP on Android devices. This applied to all the edit types except faceswap. For facewap edited images, we directly used images from two publicly available faceswap datasets FF++ [218] and DeeperForensics 1.0 [135], created using state-of-the-art deep learning models. Their original copies are included in our original face dataset.

We created a large-scale original face dataset containing 821,358 face images from 30,461 identities, by collecting images from multiple public face datasets. As summarized by Ta-

Category	Edit type	# of images	Edit tools
Global processing	Add filter	3,941	FaceApp, PortraitPro
	Change brightness	5,936	PhotoShop, PortraitPro
Modify facial attributes	Change age	4,059	FaceApp, StarGAN, HRFAE
	Change gender appearance	2,000	AttGAN, StarGAN
	Change face shape	2,954	PhotoShop, PortraitPro
	Change skin tone	2,376	AttGAN, PortraitPro
	Change hair color	2,486	FaceApp, StarGAN
	Resize eye/nose/mouth	7,914	PhotoShop, FaceAPP, PortraitPro
	Add makeup	4,925	FaceApp, PortraitPro
Change facial expression		1,967	FaceAPP, GANimation
Add/Remove Eyeglasses		1,989	FaceApp, OpenCV code
Change face identity (facewap)		2,000	FF++, DeeperForensics
12 Edit Types		**42,547**	**10 Edit Tools**

Table 4.6: **Edited faces:** we generated and labeled more than 42K edited images, covering 12 popular face editing types and 10 popular edit tools (3 commercial and 7 open-source tools).

ble 4.5, the combined dataset ensures diversity and includes a wide variety of images from both celebrities and normal people, from different genders, age groups, resolutions and skin tones. For consistency, each image only contains a single face.

Experiment #1: Accuracy of *Aletheia*'s image inspector. *Aletheia*'s image inspector seeks to recognize whether an image x is edited or original, and if it is edited, locate its original copy from the database. Thus its performance depends on the configuration/scale of the database and the similarity threshold θ_{SSIM}. To ensure a fair evaluation, we split the original face dataset described above into 3 disjoint parts: a small set (2K) randomly chosen from CelebA and FFHQ to configure θ_{SSIM}[5], a significant set (752K) to construct the databased of registered original faces, and a large set (43K) to serve as new original face photos that are used to test the image inspector's performance.

In the end, we tested *Aletheia*'s image inspector using two test datasets: an original photo test set (43K), and an edited photo set (42.5K). We studied the inspector's accuracy as follows: for the original photo test set, compute the probability of identifying the images

5. Following the procedure described in §4.5.2, we computed the SSIM scores across these images and used these results to set θ_{SSIM}=0.5 to reach a 1% false positive rate.

as new original face photos; for the edit image set, compute the probability of identifying the images as edited images and pairing them with their original versions.

Table 4.7 summarizes the accuracy result under different upload requests. For the edited images, we considered cases where the provenance data is either accurate, missing, or modified to change the declared original copy. We see that *Aletheia*'s image inspector offers a high accuracy, i.e., 99.5% when the provenance data is intact and 97.1% when the provenance data is manipulated or removed.

Upload request x_t	**Result of Image Inspector**
x_t= new original face provenance=NULL	**99.54%: correctly identified as a new original face** 0.46%: wrongly identified as an edited face
x_t= edited face provenance= x_t's original copy	**99.51%: correctly identified as edited face and paired with its original copy** 0.49%: correctly identified as edited face, but paired with a wrong original copy
x_t= edited face provenance= NULL	**97.1%: correctly identified as an edited face and paired with its original copy** 2.9%: identified as an new original face photo
x_t= edited face provenance = not x_t's original copy	**97.1%: correctly identified as an edited face and paired with its original copy** 2.9%: correctly identified as an edited face but paired with a wrong original copy

Table 4.7: The status of different upload requests after applying *Aletheia*'s image inspector. These results highlight the high accuracy of *Aletheia*'s inspector in detecting edited photos and pairing them with their original copies.

Next, Table 4.8 compares *Aletheia*'s to today's face edit detectors under the above experiment setting. FFD [84] and CNNDetector [245] are very recent systems that target generalized detection of photo edits, and FAL [244] is a recent tool that targets edits made by commercial app (PhotoShop). Compared to these existing detectors (whose accuracy is less than 50%), *Aletheia*'s image inspector offers significantly improvement in recognizing whether an image is an original or an edited photo.

Experiment #2: Accuracy of *Aletheia*'s edit recognizer. Next we studied the accuracy of *Aletheia*'s edit recognizer, using 42.5K edited face images paired with their original copies. Given an edited face x_t and its original copy x_0, we tested both images

	Detection Accuracy	
Today's Face Edit Detector	Original Face	Edited Face
FAL [244]	38.1%	37.0%
FFD [84]	41.8%	55.6%
CNNDetector [245]	93.5%	9.5%
Aletheia	**99.54%**	**97.1–99.51%**

Table 4.8: When detecting whether *any* face edit is present, recent detectors (FAL, FFD, CNNDetector) perform poorly on our datasets of original and edited face photos, often detecting many original photos as edited and many edited photos as original. *Aletheia* significantly outperforms existing works.

on each of the 9 edit detectors defined in Table 4.3. Since *Aletheia* does not make any assumption on the number of edits in x_t, a single edit could trigger more than one detector, leading to false positives. These false positives can come from natural overlap between edit types, or imperfect attribute extractors used by our current prototype.

Individual Edit Detector	**Edit in the Target Image**											
	Bright-ness	Skin-tone	Hair color	Face swap	Gender	Age	Face shape	Express-ion	Eye glasses	Filter	Make up	EyeNose-Mouth
Brightness	**89.6%**	43.8%	4.9%	0.0%	0.4%	0.3%	0.0%	0.0%	0.0%	27.5%	0.0%	0.0%
Skintone	35.4%	**99.4%**	12.3%	3.5%	11.8%	9.9%	1.4%	2.3%	4.6%	40.2%	4.3%	0.9%
Haircolor	4.3 %	5.1 %	**88.0%**	1.6%	38.1%	26.5%	1.6%	1.0%	2.1%	30.5%	1.0%	0.8%
Faceswap	0.2%	0.0%	12.0%	**86.3%**	39.4%	10.2%	0.0%	0.0%	1.2%	0.0%	0.0%	0.0%
Gender	3.5%	5.0%	5.6%	7.5%	**88.8%**	9.5%	6.0%	8.7%	10.3%	3.6%	12.2%	3.4%
Age	7.8%	20.5%	33.6%	62.3%	42.2%	**88.5%**	9.8%	21.6%	30.9%	11.1%	8.0%	2.2%
Faceshape	2.4%	9.0%	23.9%	5.6%	36.6%	16.2%	**97.0%**	1.2%	9.6%	3.6%	0.9%	0.6%
Expression	6.8%	8.7%	9.6%	17.6%	14.0%	11.5%	7.1%	**95.4%**	12.3%	4.5%	6.4%	3.5%
EyeGlasses	1.0 %	2.7 %	3.8%	4.6%	5.2%	3.2%	4.7%	1.8%	**91.8%**	1.1 %	0.8%	0.4%

Table 4.9: Results on how each edit on a target image gets recognized by *Aletheia*. Each column refers to a specific type of edit e contained by the target image, where the bold entry in the column is the probability the edit e is recognized by *Aletheia*, and the other entries are the false positives on other 8 detectors created by e.

We summarize the results in Table 4.9. Each column represents a type of edit present in the test images; the bold entry in the column marks the probability the edit triggers its own detector, i.e., the edit is recognized (or detected) by *Aletheia*; the other 8 entries in the column are the false positive rates the edit projects on the other 8 detectors. We also include results of the three edits (Filter, Makeup, EyeNoseMouth) that *Aletheia* does not have detectors designed. They are used to evaluate false positives on the 9 detectors.

Edit recognition rate – We see that for the 9 edits that *Aletheia* attempts to recognize,

the recognition rate is reasonably high (86.3% –99.4%), especially since *Aletheia* just uses off-the-shelf pre-trained models. The recognition rates for face swap, gender appearance, age are lower (86.3%-88.8%) than others, indicating that the corresponding attribute extractors are less accurate. The recognition rates for hair color and brightness are also slightly less than 90%, because it is hard to design decision heuristics for these diverse color-based changes. A more specifically designed user policy could help improve the decision accuracy.

False positives – Another key observation is the visible false positives in the recognition result. This is a mixed outcome of both natural overlap between edits and errors in attribute extractors. While it is hard to separate the two sources, we provide some initial projections. Clearly, the faceswap edit would easily trigger other detectors (except brightness), so we exclude its contribution from our analysis. Overall, we conclude that for three detectors (brightness[6], skin tone, and hair color change), the false positives are mostly due to natural overlap with other edits; for four detectors (faceswap, gender appearance, expression and eyeglass), the false positives are mostly results of errors in their attribute extractors; and for the rest two (age, face shape change) detectors, the false positives are from both factors.

Together, our results indicate that the *Aletheia* prototype can reasonably recognize common face edits, enabling identification of images with unacceptable face edits. On the other hand, upon recognizing multiple edits (especially those known to overlap with each other), *Aletheia* needs to carefully review the list to identify false positives. Along this line, *Aletheia* would largely benefit from further improvements in semantic attribute extractors and a more precise definition and interpretation of edit types. We leave this to future work.

Experiment #3: Computation cost of *Aletheia*'s image inspector and edit recognizer. Finally, we studied the end-to-end delay for *Aletheia*'s image inspector and edit recognizer, when running on a server with a single CPU (Intel Xeon 2.2GHz) and a single NVIDIA Titan RTX GPU (shown in Table 4.10). When the input to *Aletheia* is an original

6. Some edit tools adjust skintone by adjusting brightness. Thus skintone change leads to a high positive rate (43.8%) on the brightness detector.

image, the total processing time is 939ms (all spent on the image inspector). When the input is an edited photo, the processing time is 924ms (24ms on the image inspector, and 920 ms on the edit recognizer). We note that these results were obtained on a low-end server rather than sophisticated servers used by today's photo hosting services.

	Golden copy verification	Golden copy search	edit recognition (all 9 detectors)	total
Edited face	24ms	N/A	920ms	924ms
Original face	N/A	939ms	N/A	939ms

Table 4.10: The computation time of *Aletheia* inspecting a target image, assuming normal input.

4.6.4 Testing Aletheia*'s Effectiveness on In-the-Wild Face Edits*

Previously we tested *Aletheia* on photos produced by scripting 10 editing tools with 12 edit types. In this evaluation, we moved to test *Aletheia* on photos edited by real users in the wild, using their own tools that *Aletheia* has no knowledge of. For this we recruited 8 volunteers (non-authors), presented them with a set of 100 original face images (randomly chosen from the original face dataset), and invited them to edit any of these images as they want. The only instructions given were that they need to log the edit(s) per image and each edit needs to be visible by human eyes. There is no restriction on the number of edits or what the edit tools to use.

Following this process, we received 415 in-the-wild images, each carrying 1-4 different edits (1.9 edits on average). We observed that for all 415 images, the meta data containing the provenance data remained unchanged after the edit.

Accuracy of *Aletheia*'s image inspector. When passing these 415 photos to *Aletheia*'s image inspector, 391 (94.2%) were correctly identified as edited images and paired with their original copies. This value is lower than previous results in Table 4.7 (99.5%). A closer investigation showed that the lower accuracy came from images that have been largely cropped during the edit. Among images not cropped, the inspector's accuracy increases to

97.6%. Thus the image inspector could be further improved to address cropping.

	***Aletheia*'s individual edit detector**								
	Bright-ness	**Skin tone**	**Hair color**	**Face swap**	**Gender appearance**	**Age**	**Face shape**	**Expression**	**Eye-glasses**
Recognition rate	75.8%	59.1%	80.8%	75.0%	97.2%	81.2%	77.6%	82.6%	94.1%
False positive rate	20.8%	36.7%	20.8%	7.5%	7.0%	32.2%	25.5%	11.4%	5.3%

Table 4.11: The performance of *Aletheia*'s individual edit detector on in-the-wild edited images.

Accuracy of *Aletheia*'s edit recognizer. Table 4.11 summarizes the edit recognition result across these 391 images. Here we report the overall result per detector, in terms of its recognition rate (probability of detecting a target edit e when e is present on the image) and false positive rate (probability of detecting e when e is absent). Compared to the result in §4.6.3, the detection rate does decrease (except for gender appearance, which increases to 97.2%), but the false positive rate also decreases considerably. The drop in the recognition rate is particularly visible for skin tone (59%). This is because the vagueness of the edit definition: our detector uses the common human skin color palette and detects edits that introduce sufficient color changes; but the skin tone changes made by our volunteers were often subtle and did not trigger our detector.

4.6.5 Testing Aletheia*'s Usability with User Studies*

To understand the usability of *Aletheia*, we conducted a second user study. Our goal is to assess whether and how *Aletheia* can help real-world users protect their online images, and understand users' opinions on face edit protection to help us improve *Aletheia*.

Participants. Again, we recruited 100 participants via Prolific (IRB-approved). Each participant spent 10 minutes on average and received $2 as compensation. We received 97 valid responses (after removing 3 responses where the participants failed the attention check question). We also asked each participant whether they were concerned about online privacy and 7% indicated that they did not feel concerned at all. Since those privacy-insensitive users

are not our target users, we filter their responses when reporting the results. In the end, we analyzed 90 responses. Of these, 44% identified themselves as female (66% male). The participants cover multiple age groups: 18-29 years old (75%), 30-39 years old (16%), 40-49 years old (7%) and 50-59 years old (2%).

Procedure. Similar to the first study (§4.3), participants completed an online study consisting of multiple choice and free response questions. Unlike the first study, this user study does not focus on an actual policy defined by each participant, but rather their opinions about *Aletheia* as a system. Therefore, instead of having participants define their own policy, we walked participants through the system to help them understand the functionality of *Aletheia*. We first showed each participant the interface of defining the policy for each edit type, together with an example (e.g., age change) Then we presented how the system would block an edited image (e.g., age appearance was changed) when an unacceptable edit was detected.

Task. We asked participants to imagine they are using this system with personalized edit detection when posting an image that contains their face. We then asked a series of questions about the usability of the system, their sense of protection and perceptions of privacy when posting images online. For each question, participants answered a multiple choice question to categorize their opinion, then we asked them to further explain their responses in their own words.

Results. Following similar sentiments from the first study, we found that participants in this study expressed concerns about posting images online, with varying opinions on how they would individually like to use a system like *Aletheia*. Particularly, most users show appreciation for the protection offered by a system such as *Aletheia* and would be interested in using it, while some expressed significant concern about the difficulties of maintaining any level of protection when posting online. Here we discuss our key findings.

User perceptions of* Aletheia*'s protection – We evaluated whether people would like to use *Aletheia* when posting their images online, and how they felt about the level of

protection *Aletheia* would provide. We note that similar to our first user study (§4.3), we found a near even split (49%/51%) among participants regarding whether they are concerned about their image being edited and reposted by others.

First, regarding whether they would use *Aletheia* when posting images, we observed a considerable difference between *edit-concerned* and *edit-unconcerned* participants as seen in Table 4.12. Most of *edit-concerned* participants (68%) were interested in using *Aletheia*. Even among those not concerned about other people editing their images, 42% indicate they would still use *Aletheia* nonetheless, which further shows how *Aletheia* could serve to protect even those unaware of potential bad actors in the wild. We looked further into participants who expressed concern about their images being edited, yet were less willing to use *Aletheia*. For *edit-concerned* participants, 5 (11%) of them chose not to use *Aletheia*. 4 out of these 5 participants expressed that they never shared images on social platforms and they did not feel protected even with *Aletheia*, either because the system could be bypassed or because posting images online never seems safe.

User group	**Yes**	**Neutral**	**No**
edit-concerned	68%	21%	11%
edit-unconcerned	42%	27%	27%

Table 4.12: Participants' responses for whether they would use *Aletheia* when posting images on social media sites. We categorize "definitely would use" and "probably would use" responses as "Yes", and correspondingly "definitely would not" and "probably would not" responses as "No".

Next, we evaluated whether participants feel their images would be protected by *Aletheia*, and the reasons why, in Table 4.13. Overall, 48% of participants found their images definitely or probably protected with *Aletheia*, and expected the system to function properly and detect unwanted edits. A small section of the participants (15%) expressed neutral feelings. Within this group, some think there will always be workarounds for malicious editors, such as posting the edited images on other platforms. Many expressed some slight doubt, but still showed interest in the system as a step in the right direction and better than the current complete lack of image edit protection online.

Response	Reason	Example
Protected (48%)	Trust system works to detect disallowed edits	"I would feel my images **are protected by the system as I can specify** whether I would like them to be modifed [*sic*] in a way I would not like." "The list of edit types that can be detected seems quite comprehensive"
Neutral (15%)	Can not guarantee 100% protection	"I believe not everything can be picked up by technology and **it may miss when a photo has been edited**" "i think they do protect the images **to a certain extent however not fully**"
Not Protected (37%)	Posting images online is never safe (64%)	"I think it's **never safe when we post pictures of ourselves** because they never really leave the internet." "The only situation I'd feel my images to be protected would be if they weren't uploaded to any platform at all."
	The system can be bypassed (23%)	"If someone is going to edit your pictures, they're going to use a platform that allows them to do this." "I think it **could be cheated** easilly [*sic*]" "Pictures can still be extracted and **posted somewhere else**."
	Do not trust the system to work as described (13%)	"I **don't think that the system is advanced enought** [*sic*] to detect these images." "Since i dont know much about the system i would not trust it enough to upload my picture" "Not fully clear on how they'd be able to tell it was altered"

Table 4.13: Participant responses for whether they feel their images were protected with *Aletheia*.

When examining why participants indicated they did not feel protected, we found an overwhelming sense of distrust in general when posting any information online, especially images. Overall, about 37% of participants did not expect protection with *Aletheia*, and we summarize their reasons into three categories. The majority of them (64%) expressed the same reason, i.e., posting images online is never safe and the only means of protection is to not upload the image. Additionally, 23% of participants were worried that the system could be bypassed, such as posting edited images somewhere else. There were 13% of participants who could not trust *Aletheia* either because they did not know how the system works or because they were not convinced that they system could detect edits accurately. Still, this group makes up just over a third of our participants, with most participants expressing interest in *Aletheia* as an effort to increase their privacy protection online. Efforts should be

made to provide a more trusted environment for online social interactions, and we believe *Aletheia* is one step towards a more privacy-friendly service.

User opinions on policy configuration – Since setting up a personal policy for *Aletheia* inevitably requires some extra time, we want to understand whether users would mind spending the time to achieve image edit protection. As reported in Table 4.14, 45% of participants were not concerned, and an additional 32% of participants were neutral, about the time spent on defining the policy, as the protection is deemed worth the initial setup time. However, 23% of participants expressed concern about the time spent, with one participant feeling this may leave many users reverting to default settings. When asked about whether they would prefer to set a single policy for all images, or different policies for each image, overwhelmingly participants (75%) indicated they would prefer a single policy, for simplicity and efficiency. On the other hand, only 31% of participants indicated they would prefer a single policy for different audience groups. Many participants cited concerns over privacy and trust, with several people mentioning they would feel safer posting and allowing edits of photos from certain groups but not the public. This suggests that the design of policy management should serve to spare the users' efforts, while still affording them personalized control. For example, the system could learn users' preferences and set a suggested policy automatically, with options for users to further customize the settings.

Response	**Reason**	**Example**
Not concerned (45%)	Will not take a long time	"I don't think the time is a problem here, when i'm posting a picture I **already know what settings to chose** and it's easy to set them." "it seems to be quick"
Neutral (23%)	No strong opinion	"I just feel neutral about it, I don't know." "I **wouldnt really mind much**."
Concerned (13%)	Takes too long for privacy setting	"**If its too time consuming or complex** to set, users may be put off or may choose minimum protection options" "not a lot of time is spent setting policy"

Table 4.14: Participant responses about time spent setting their policies.

User suggestions for system design – To learn how we can improve *Aletheia*, we

asked the participants what changes, if any, they would make if they were to design this system. Although most of participants expressed that there were no changes they wanted to make, there were a few interesting responses that we could consider to improve *Aletheia* in the future. Four participants mentioned that they would like to get notifications if any edit of their images is detected, and then they would decide if they want it removed. In our first user study, there was also an even split among participants regarding whether the platform should immediately remove the disallowed edited image or notify users to review the image. Therefore, we believe this choice should be open to users in the privacy setting.

Another suggestion that was mentioned by 11 participants relates to the image editors. They expressed, "People who often violate policies will be blocked from the platform", while they wanted to "set certain friends to have edit permissions" or "allow users to ask for permissions to the original author of the image." This suggests, not only the edit type will affect users' decision on whether allow or disallow edits on their images, but also the person who is making the edits. Therefore, one feature we could add on *Aletheia* is to include both blacklist and whitelist of image editors, which gives users extra flexibility to control how their images could be edited.

Lastly, some participants brought up a desire to implement *Aletheia* on all possible platforms. A widespread implementation would resolve many of the critiques regarding the ability of *Aletheia* to protect images posted online. This would require significant cooperation across photo sharing platforms, and we leave this to future work.

4.7 Related Work

We now discuss related work on understanding and addressing online photo privacy.

User Perceptions of Photo Privacy Online. As users share increasingly more photos online, attitudes towards the cultural and privacy aspects of photo sharing continue to evolve [184, 43]. Recent studies show that OSN users generally express specific awareness

and concern for privacy when posting photos of themselves or others [90, 75, 158]. Even on extremely popular OSNs designed specifically for sharing images (e.g., Snapchat, Instagram, Pinterest), privacy still remains a large concern for users [123].

At the same time, existing works also show that concerns for privacy do not necessarily mean users will maintain consistent privacy-enhancing behaviors online, as users maintain different ideas and preferences regarding online privacy [136]. In particular, although users tend to express general concern over their online privacy, the extent and specificity of these concerns vary across users, along with the actions users decide to take [195, 197]. Perceptions of privacy often relate to the user's personal traits (e.g., age, gender), and their different definitions of what privacy means to them individually [211, 148]. For instance, Kwasny *et al.* [148] found that males tend to show more focus on personal convenience while females focus more on the privacy and safety of themselves and others, and younger people focus more on control over their information disclosed while older people are less focused on privacy of information in general. This suggests members of different population groups would likely prefer different privacy settings for their personal profiles, which are not always readily available on OSNs.

Similarly, studies on cyberbullying and harrassment also show that users maintain different definitions of what constitutes harassment, and as a result, online communities often struggle to enforce anti-harassment policies [57, 56].

Protecting Online User's Photo Privacy. While the perceptions of online privacy are highly personal, the availability and usability for establishing personal privacy settings varies across platforms, resulting in different expectations and actualities for user privacy protection [142]. As summarized by [154], existing protection strategies can be broadly categorized into two approaches: 1) deploying privacy policies to control viewership or moderate content, and 2) injecting digital artifacts onto images to obfuscate privacy-sensitive content.

For policy-based protection, researchers have spent considerable efforts to improve privacy policy management. These include strategies that assist users in choosing/configuring their

privacy settings, either automatically [101, 231] or semi-automatically [188], visualization tools to better explain privacy settings to users [178, 95], as well as methods to collaboratively manage privacy settings on photos containing multiple faces [233, 55, 232]. The most relevant to our work is [231], where Squicciarini *et al.* show the desire for personalized preferences when posting photos on social media sites and propose to predict user policies based on audience group privacy [231]. However, existing works mainly focus on regulating who can see an image, rather than moderating/protecting the content of images from unwanted editing and reposting. A more recent work [98] proposes to employ human users as "digital juries" to moderate online content. Yet the overhead of human juries and the inherent difference of privacy perception across users make this approach unsuitable for moderating/detecting images with unacceptable edits.

The second approach operates on the media content to be protected, by obfuscating (thus protecting) sensitive information. This can be done automatically via hardware [205] and software [37] features or manually configured by human users themselves. Obfuscation has been widely used as a popular privacy-enhancing technology for images, where sensitive contents (e.g., scene element [114, 153], bystander [91], face/person [154, 115, 153]) are obfuscated/blurred before an image is uploaded. Obfuscation also helps collaborative privacy management. A recent work proposes to apply obfuscation/blurring to "anonymize" users who are unwilling to show their faces in any co-owned photos [253]. Clearly, applying obfuscation before uploading a user's face photo could effectively prevent others from editing the face. However, doing so is also shown to adversely affect viewers' experiences [154, 114], defeating the original purpose of sharing self-images online.

Overall, our discussion above shows that despite existing efforts on online privacy protection, there is a lack of deep understanding and practical implementation of personalized, online face photo protection with regards to who can edit and which edits are (un)acceptable per individual user. Our work seeks to address this gap.

4.8 Conclusion

Conclusion. This work addresses the challenge of personalized protection against unauthorized face edits, allowing social platforms to support collaborative interactions via enhanced photos while restoring user agency and control over how their images can be altered. To the best of our knowledge, our work is the first to explore and propose a solution to address what is likely a growing problem. We hope this paper brings more attention to this important problem and spurs continued efforts to reduce potential abuse of face editing tools in collaborative environments.

Future Work. As the first system targeting personalized face edit protection, *Aletheia* is subject to a number of limitations. At the same time, it also provides opportunities for further development and exploration. We summarize key points below.

1) Precise, usable policy for personalized face edit protection. Our current policy specification adopts a simple structure, i.e., a binary acceptability decision on each of several possible types of facial edits. While these edit types are intuitive and easy to understand, we recognize three broad challenges in clearly defining face edit types that are consistently interpreted across users.

First, some edit types are much less defined compared to others (e.g., changing face shape vs. changing identity). Similarly, types of face edits listed by face edit tools and research literature are often broadly and vaguely defined. This affects the granularity of *Aletheia*'s edit type recognition. In our user study, we observed that participants' choices out of 5 acceptance levels (from "never allow edits" to "allow any edits") vary significantly. However, due to the capabilities of our attribute extractors, currently *Aletheia* only supports a binary acceptability decision. We believe, as face edit types/granularities are better defined, and fine-grained facial attribute extractors are developed, *Aletheia* could provide improved face edit protection with multiple levels.

Second, many face edits are naturally correlated (or overlapping). For example, changing

the age of a person may involve changes to eyes, nose, mouth, skin tone, and hair. Thus altering a target's age likely triggers multiple attribute changes, and *Aletheia* could recognize five edit types rather than one. These "natural" dependencies may or may not be correctly observed by users when configuring personalized policies. As a future topic, a systematic approach could be developed to interpret and decompose face edit types, and an interactive interface to guide users in defining usable policy. More precisely specified policies can help *Aletheia* better capture a user's true preferences for better personalized image moderation.

A third challenge is that defining certain edit types such as gender appearance often means relying on common stereotypes that fail to properly capture real world diversity. For example, our initial user policies only considered gender appearance for binary male/female genders. Further, detection of gender appearance change relies on stereotypes of a "typical" male or female (e.g., longer/shorter hair), and fails to represent real world gender diversity. Similarly, detection of face aging draws on general stereotypes (e.g., gray hair, wrinkles), and may also fail to represent diversity of real world populations. We note that these overly granular tools are still effective for detecting efforts to bully or harass using edited photos, since those attacks often target users based on exaggerated versions of common stereotypes. Regardless, much work remains in developing a more nuanced and powerful policy specification that better reflects user diversity.

2) Automatic policy setting via learning users' preferences. In our usability study, we found that 23% of participants were concern about the time spent on policy setting. One participant even expressed, "I would implement a self learning AI into the underlying system, so that it would become more efficient over time." Similar to prior systems [101, 231, 188, 178, 95] that aim to help users configure their privacy setting (image viewership), a future design of policy manager could include ML tools to learn users' preferences towards different edit types. Interactive interface could be used to interact with users to obtain their decisions on a limited number of edited images, then train ML models to predict users' preferences to help set their image policies automatically.

3) Large-scale user studies. For privacy purposes, the user survey in this study asks participants to imagine the images are of themselves, rather than actually using modified versions of their own photos. This has two effects: the participant is less familiar with the original image (and therefore how it may have been edited), and they likely feel less personally invested in how the edited image appears, since the image shows someone else's face. We believe this may have made it more difficult to notice certain edits on the images. In future work, we will perform a large scale, in-the-wild study where users would evaluate this system using their own images.

4) Integration with multiple photo-sharing platforms. The current design of *Aletheia* supports an individual photo-sharing platform. While this can be effective to protect users if deployed by a very large platform like Instagram, we could achieve much more impact if multiple platforms collaborate to protect all users' photos. As several participants from our user study (§4.6.5) noted, even with *Aletheia*, images posted online still face the risk of being downloaded and reposted on another platform. To ensure protection, a natural extension to this work would consider privacy-preserving ways to share personalized user policies across platforms, so unacceptably edited images could be detected across organizational boundaries.

5) Co-design of hashing based original copy search and attribute extraction. Currently, we design image inspector and edit recognizers independently, i.e., selecting hashing function and attribute extractors separately. Perceptual hashing is applied because it captures the image's perceptual content, which is irrelevant to edit recognition. This might cause conflicts or gaps between two components. For example, the uploaded image has similar background with one original image in database, resulting in small hash distance. But when checking the facial attributes, we find that the two images are similar only because of the background and the faces are completely different. A possible solution is to also apply hashing to facial attributes (e.g., face identity). In this way, another candidate of original copy returned from database is the image that matches the uploaded image the most in attribute space. Then further inspection could be conducted to determine whether the

identified original copy candidate is correct or not.

CHAPTER 5

DISCUSSION AND FUTURE DIRECTIONS

The goal of this dissertation is to achieve practical ML deployment in sensing and monitoring systems. To do so, I first explore the obstacles that hinder a successful deployment. Using measurements and user studies, I identify two common system-level challenges, i.e., *heterogeneity* and *scalability*. Then, I propose to solve these two challenges by integrating efficient system design and evaluation platforms into the deployment process.

After successfully designing and evaluating three sensing and monitoring systems (including both physical sensing and cyber monitoring), my findings indicate that ML deployment in the real world requires interdisciplinary research at the intersection of traditional systems and machine learning communities. This dissertation follows this direction and solves two specific challenges. At the system level, my efforts help ML models scale to different environments (e.g., sensor locations, mobility status) and accommodate a wide variety of users' needs.

Next, I will summarize my key contributions and takeaways, and then outline several interesting future directions.

5.1 Summary of Contributions

This dissertation is part of the effort to integrate ML to real-world systems. Specifically, my work has contributed to the ML deployment by addressing the dual challenges of heterogeneity and scalability in sensing and monitoring systems. In summary, my contributions are listed as follows:

- Using empirical measurements and user studies, we find two common system-level challenges, i.e., heterogeneity and scalability. Particularly, we identify multiple heterogeneity sources that affect the system performance, including sensor local environments, sensor mobility status and user preferences.

- In the spectrum monitoring project, the contribution is twofold. From the application development perspective, we build the system from scratch, including problem formulation, model input representation and model design, which could serve as the basic framework for future research on spectrum monitoring. From the perspective of ML deployment, our system scales DNN models to distributed monitoring nodes at different locations. The system design could be applied to other sensing systems with similar requirements.
- Additionally, we propose the first benchmark for video analytics pipelines. This project shows that heterogeneity and scalability not only affect the system design, but also the system evaluation. Careful designs are needed for accurate and transparent evaluation of ML systems. Therefore, we propose an evaluation methodology, which provides comprehensive evaluation by characterizing the correlation between ML system performance and environmental heterogeneities.
- Finally, another contribution lies in our study on personalized face edit protection. We design and build a novel image moderation system to detect and flag unacceptable face edits based on user-defined policies to provide personalized protection. Our proposed system demonstrates how to incorporate diverse user preferences into the design of ML systems.

5.2 Key Takeaways

ML has shown a lot of potential in research studies, but putting ML into practical use requires efforts beyond just ML model design and evaluation. Here, I would like to share a few lessons that I have learned in my experiences of building ML-based systems.

System-level vs. Model-level Solution. When we talk about ML, in most cases we mean ML models and associated algorithms. Countless efforts have been made to propose and subsequently improve ML models in different applications. However, many existing ML studies are conducted on pre-collected (static) datasets. When the model performance is not satisfying (e.g., low accuracy, not generalizable), a common choice is to come up with

model-level solutions, e.g., proposing new models or adding more training data.

While there remains little doubt that model-level solutions could lead to better performance on existing datasets, the added complexities often render such solutions impractical in real-world settings. In our studies, we find that some problems can be better solved at the system level by reformulating the problem, changing the data representation or adding new system components.

For example, in the spectrum monitoring project, the challenge is that wireless measurements, (e.g., spectrum usage), depend on the context of the sensors, (i.e., time, location, and mobility status). A typical model-level solution would solve this problem either by building a universal DNN model for all the sensors, or running DNN models tailored for each location. However, they are both impractical. In the "universal model" solution, the model needs to be very complex to handle all the heterogeneities, which greatly increases the run-time complexity. As for one-model-per-location solution, the amount of training overhead would be significant since each sensor must change its DNN model whenever it moves. Instead, our proposed system achieves better tradeoff between model complexity and training overhead using hierarchical model deployment strategy: a single model per cell.

Another example is rethinking the problem formulation in our face edit protection project. Traditionally, research on face edits largely focuses on the problem of *detecting whether a face photo has been manipulated*. This is motivated by detection of deepfakes, often in the context of AI-generated fake photos/videos that misrepresent public figures (e.g., politicians and celebrities) or fabricate fake news events used in disinformation campaigns. However, in the context of online image sharing, what we really care about is whether the original image owner(s) would *accept* the edit or not. A simple model-level adjustment is to change the binary classifier of detecting manipulated or original to a multiclass classifier of recognizing different edit types. However, building such a classifier is extremely challenging since edited tools are developed so fast and different tools make different styles of changes. Instead, we address this challenge using a system approach, i.e., integrating the system function of

tracking original image copies. In this way, we successfully reformulates the problem to a feasible ML problem of comparing two images. This change allows for robust face edit protection across a large user population.

The Importance of Domain Knowledge. Domain-specific expertise is critical throughout the entire process of ML deployment, including problem formulation, data collection and even choosing ML model parameters. Therefore, in all three projects, we first explore the underlying structure and dependency within the distributed sensing and monitoring systems, and then integrate them into our designs. By doing so, we achieve practical ML deployment.

In the design of spectrum monitoring system, our domain insights of cellular networks inspired us to build a hierarchical scheme for scaling the model to many sensors. Specifically, cellular networks employ an hierarchical network structure for communication. Within a cell, all devices are connected to the same base station. The cell switch only happens when a device moves across cells. So, within a cell, all sensors monitor the spectrum of the same base station, where the heterogeneity is mainly from noise and signal propagation. Those can be easily solved by normalization. This motivated us to build a single unified model within a cell. And across cells, sensors monitor the spectrum of different base stations with different behaviors (e.g., modulation, frequency bands), so we need to customize the model for each cell. In the end, our hierarchical solution perfectly aligns with the hierarchical structure of the cellular networks. Whenever the observer switches connected cells, it also switches the spectrum monitoring model.

Domain knowledge is also crucial to help understand our data during data preprocessing and representation. For example, when translating our raw measurements of LTE spectrum into the inputs of DNN models, our knowledge of LTE transmission suggested patterns that exist in both time and frequency domains could be utilized to separate normal traffic and anomalies. Therefore, we built the model input as time-frequency spectrograms, which capture fine-grained signal amplitude over time at sub-frequencies.

5.3 Future Directions

This dissertation addresses two specific system challenges in one type of ML systems. Moving forward, there remains many interesting problems left unexplored in the broader context of building ML systems. Here, I list a few directions that I find most interesting.

Robust ML System with Noisy Sensing Data. Environmental noise is very common in sensing systems, and could have a huge impact on system performance. Possible sources of environmental noise include daily changes in the weather (e.g., fog, rain), pollution in atmosphere and motions of the edge sensors. Therefore, when deploying ML models in sensing and monitoring systems, it is crucial to address and ensure reliability in the presence of noise, especially for safety-critical applications. For example, for object detection models deployed in autonomous driving systems, noise in images (e.g., blurring, brightness change) may lead to faulty detection results and cause serious consequences.

The fundamental reason behind such faulty results is that the ML models do not generalize well to unseen input. Therefore, existing mitigation techniques mainly focus on improving the model's generalization capability by training the model with noisy inputs or synthetic datasets. However, this solution is limited by the availability of large and diverse data covering all types of environmental noise. Alternatively, we could address this problem using a system approach. Different sensors could be deployed in the sensing system, providing extra information sources for the ML model to make decisions. Instead of simple concatenation at input level, intelligent sensor fusion techniques could be developed. For example, the ML model could be trained to learn confidence scores for the data collected by each sensor, and only generate prediction results on data with high confidence.

ML System Interpretability. When testing the usability of our proposed photo moderation tool, we find that many users share the same concern: they can not trust our system because they do not know how the system works. This indicates a need for better interpretability, especially in such systems directly linked to human interaction.

However, achieving interpretability is known to be challenging when ML/DL is applied. While there are existing efforts on explaining the decisions made by ML/DL models (such as LIME [216]), there remains two significant limitations. First, they focus on the models only, rather than the whole system. However, in real-world applications, model interpretability alone is not sufficient to guarantee interpretable and trustable systems, especially with the presence of multiple DL models (e.g., there are deep candidate generation model and deep ranking models in YouTube recommendation system [83]). Future research could consider the interactions between different system components so that the logic of the whole system can be inspected and trusted. Second, most existing studies aim towards ML researchers as the target audience, but there lacks enough research to explain the rationale behind the decisions for a broad range of users who are not ML experts. Collaboration among ML, system and Human-Computer Interaction (HCI) researchers would be needed to improve the interpretability and usability of ML systems. This includes using interpretable features, and designing intuitive visualization techniques/interface.

Protect End User Privacy from ML Models. In the face edit protection project, we show that there exists general distrust of online photo sharing services among privacy-sensitive users. Participants expressed that posting images online is never safe and the only means of protection is to not upload the image. This concern has become increasingly prevalent, as increased evidence has shown that ML models could infer sensitive information from people's social media posts , and such information may be used for user tracking or targeted advertising. Currently, many users remain unaware of how, or if, they can opt out of such tracking.

While there are efforts to build to detect online privacy violation (such as our face edit protection tool), most of them come into effect *after* the privacy violation, which sometimes can not undo the damage. Instead, we can build proactive tools which could protect images and posts we share online from potential misuses of unauthorized ML models. One example would be a tool that helps individuals inoculate their images against unauthorized

facial recognition models [227]. Similarly, for face edit protection, we could build tools to proactively protect our face images by adding imperceptible pixel-level alterations to our own photos before sharing them online. In this way, if our images are collected by attackers and used for malicious edit, the added alterations will distort the output of face edit models.

REFERENCES

[1] AI traffic video analytics platform being developed. https://www.traffictechnologytoday.com/news/traffic-management/ai-traffic-video-analytics-platform-being-developed.html.

[2] Android view client. https://github.com/dtmilano/AndroidViewClient.

[3] AT&T unlimited data plans with talk & text. https://www.att.com/plans/unlimited-data-plans/.

[4] BlueStacks Android Emulator. https://www.bluestacks.com/.

[5] British transport police: CCTV. http://www.btp.police.uk/advice_and_information/safety_on_and_near_the_railway/cctv.aspx.

[6] Can 30,000 cameras help solve Chicago's crime problem? https://www.nytimes.com/2018/05/26/us/chicago-police-surveillance.html.

[7] Can physical distance monitoring & smart cameras help businesses reopen faster? https://www.wwt.com/article/can-physical-distance-monitoring-and-smart-cameras-help-businesses-reopen-faster.

[8] Duke MTMC (Multi-Target, Multi-Camera). https://megapixels.cc/duke_mtmc/.

[9] Face segmentation model. https://github.com/zllrunning/face-parsing.PyTorch.

[10] FaceAPP. https://faceapp.com/app.

[11] FFmpeg. https://www.ffmpeg.org/.

[12] GoodVision: Smart traffic data analytics. https://goodvisionlive.com/.

[13] A guide to video analytics: Applications and opportunities. https://tryolabs.com/resources/video-analytics-guide/.

[14] Hair color palette. https://colorswall.com/palette/43520/.

[15] Humans can't watch all the surveillance cameras out there, so computers are. https://slate.com/technology/2019/06/video-surveillance-analytics-software-artificial-intelligence-dangerous.html.

[16] intuvision VA traffic use case. https://www.intuvisiontech.com/intuvisionVA_solutions/intuvisionVA_traffic.

[17] Microsoft rocket video analytics platform. https://github.com/microsoft/Microsoft-Rocket-Video-Analytics-Platform.

[18] Open-source implementation for adding sticker. https://github.com/charlielito/snapchat-filters-opencv/.

[19] OpenCV library. https://opencv.org/.

[20] Perceptual hash implementation. https://github.com/JohannesBuchner/imagehash.

[21] PhotoShop. https://www.photoshop.com.

[22] PortraitPro. https://www.anthropics.com/portraitpro/.

[23] Skin tone palette. https://www.summitprintingpro.com/graphic-design/tutorials/skin-tone-correction.html.

[24] Smart retail, digital transformation of retail business. https://gyrus.ai/blog/smart-retail-location-point-of-sale-analytics-camera/.

[25] Streamlink. https://streamlink.github.io/.

[26] Tensorflow detection model zoo. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

[27] Traffic video analytics – case study report. https://www.microsoft.com/en-us/research/publication/traffic-video-analytics-case-study-report/.

[28] TrafficVision: Traffic intelligence from video. http://www.trafficvision.com/.

[29] Video analytics for public health use cases. https://www.briefcam.com/video-analytics-for-public-health-use-cases/.

[30] Video analytics market size worth $9.4 billion by 2025 — CAGR: 22.8%. https://www.grandviewresearch.com/press-release/global-video-analytics-market.

[31] The Vision Zero Initiative. http://www.visionzeroinitiative.com/.

[32] Vision zero video analytics partnerships. https://bellevuewa.gov/city-government/departments/transportation/safety-and-maintenance/traffic-safety/vision-zero/video-analytics.

[33] Xiph.org video test media. https://media.xiph.org/video/derf/.

[34] youtube-dl. https://youtube-dl.org/.

[35] Data generated by new surveillance cameras to increase exponentially in the coming years. http://www.securityinfowatch.com/news/12160483/, 2016.

[36] Why do 87% of data science projects never make it into production. VentureBeat, July 2019. https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/.

[37] Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. I-pic: A platform for privacy-compliant image capture. In *Proc. of MobiSys*, 2016.

[38] Adobe. Content authenticity initiative (CAI). `https://contentauthenticity.org/approach`, 2020.

[39] Shruti Agarwal, Tarek El-Gaaly, Hany Farid, and Ser-Nam Lim. Detecting deep-fake videos from appearance and behavior. *arXiv:2004.14491*, 2020.

[40] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *Proc. of CVPR workshops*, 2019.

[41] Saifuddin Ahmed. Who inadvertently shares deepfakes? analyzing the role of political interest, cognitive ability, and social network size. *Telematics and Informatics*, 57, 2021.

[42] M. Amirijoo et al. Cell outage management in LTE networks. In *Proc. of ISWCS*, 2009.

[43] Mary Jean Amon, Rakibul Hasan, Kurt Hugenberg, Bennett I Bertenthal, and Apu Kapadia. Influencing photo sharing decisions on social media: A case of paradoxical findings. In *Proc. of S&P*, 2020.

[44] Monica Anderson. A majority of teens have experienced some form of cyberbullying. Pew Research Center, Sept. 2018. `https://www.pewresearch.org/internet/2018/09/27/a-majority-of-teens-have-experienced-some-form-of-cyberbullying/`.

[45] Tarik Arici, Salih Dikbas, and Yucel Altunbasak. A histogram modification framework and its application for image contrast enhancement. *IEEE Transactions on Image Processing*, 18, 2009.

[46] Timothy G Armstrong, Vamsi Ponnekanti, Dhruba Borthakur, and Mark Callaghan. Linkbench: a database benchmark based on the facebook social graph. In *Proc. of SIGMOD*, 2013.

[47] Octavio Arriaga, Matias Valdenegro-Toro, and Paul Plöger. Real-time convolutional neural networks for emotion and gender classification. *arXiv:1710.07557*, 2017.

[48] Yannis M Assael, Brendan Shillingford, Shimon Whiteson, and Nando De Freitas. Lipnet: End-to-end sentence-level lipreading. *arXiv preprint arXiv:1611.01599*, 2016.

[49] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019.

[50] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al. Benchmarking TinyML systems: Challenges and direction. In *Proc. of MLsys workshop*, 2020.

[51] Tarun Bansal, Bo Chen, and Prasun Sinha. Fastprobe: Malicious user detection in cognitive radio networks through active transmissions. In *Proc. of INFOCOM*, 2014.

[52] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Proc. of NeurIPS*, 2019.

[53] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Anticipatory caching in small cell networks: A transfer learning approach. In *Proc. of WAN*, 2014.

[54] Roy F Baumeister. A self-presentational view of social phenomena. *Psychological bulletin*, 91(1), 1982.

[55] Andrew Besmer and Heather Richter Lipford. Moving beyond untagging: photo privacy in a tagged world. In *Proc. of CHI*, 2010.

[56] Lindsay Blackwell, Jill Dimond, Sarita Schoenebeck, and Cliff Lampe. Classification and its consequences for online harassment: Design insights from heartmob. *ACM on Human-Computer Interaction*, 1(CSCW), 2017.

[57] Lindsay Blackwell, Nicole Ellison, Natasha Elliott-Deflo, and Raz Schwartz. Harassment in social virtual reality: Challenges for platform governance. *ACM on Human-Computer Interaction*, 3(CSCW), 2019.

[58] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *Proc. of ICML*, 2017.

[59] Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001.

[60] Anne Bouillard, Aurore Junier, and Benoit Ronot. Hidden anomaly detection in telecommunication networks. In *Proc. of CNSM*, 2012.

[61] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya R Dulloor. Scaling video analytics on constrained edge nodes. In *Proc. of MLsys*, 2019.

[62] Qingqing Cao, Niranjan Balasubramanian, and Aruna Balasubramanian. MobiRNN: Efficient recurrent neural network execution on mobile GPU. In *Proc. of EMDL*, 2017.

[63] Jiyoung Chae. Virtual makeover: Selfie-taking and social media use increase selfie-editing frequency through social comparison. *Computers in Human Behavior*, 66, 2017.

[64] Ayon Chakraborty, Udit Gupta, and Samir R Das. Benchmarking resource usage for spectrum sensing on commodity mobile devices. In *Proc. of HotWireless*, 2016.

[65] Ranveer Chandra, Venkata N Padmanabhan, and Ming Zhang. Wifiprofiler: cooperative diagnosis in wireless lans. In *Proc. of MobiSys*, 2006.

[66] Mengyuan Chao, Radu Stoleru, Liuyi Jin, Shuochao Yao, Maxwell Maurice, and Roger Blalock. AMVP: Adaptive CNN-based multitask video processing on mobile stream processing platforms. In *Proc. of SEC*, 2020.

[67] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107, 2019.

[68] Jonlin Chen, Masaru Ishii, Kristin L Bater, Halley Darrach, David Liao, Pauline P Huynh, Isabel P Reh, Jason C Nellis, Anisha R Kumar, and Lisa E Ishii. Association between the use of social media and photograph editing applications, self-esteem, and cosmetic surgery acceptance. *JAMA facial plastic surgery*, 21(5), 2019.

[69] Ruiliang Chen, Jung-Min Park, and Jeffrey H Reed. Defense against primary user emulation attacks in cognitive radio networks. *IEEE JSAC*, 26(1):25–37, 2008.

[70] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proc. of SenSys*, 2015.

[71] Zesheng Chen, Todor Cooklev, Chao Chen, and Carlos Pomalaza-Ráez. Modeling primary user emulation attacks and defenses in cognitive radio networks. In *Proc. of IPCCC*, 2009.

[72] Yu-Chung Cheng et al. Automating cross-layer diagnosis of enterprise wireless networks. In *Proc. of SIGCOMM*, 2007.

[73] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. In *Proc. of MLsys*, 2019.

[74] Wen-Long Chin et al. Channel-based detection of primary user emulation attacks in cognitive radios. In *Proc. of VTC*, 2012.

[75] Tae Rang Choi and Yongjun Sung. Instagram versus snapchat: Self-expression and privacy concern on social media. *Telematics and Informatics*, 35(8), 2018.

[76] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proc. of CVPR*, 2018.

[77] Gabriela F Ciocarlie, Ulf Lindqvist, Szabolcs Nováczki, and Henning Sanneck. Detecting anomalies in cellular networks using an ensemble method. In *Proc. of CNSM*, 2013.

[78] Justin D Cochran and Stuart A Napshin. Deepfakes: awareness, concerns, and platform accountability. *Cyberpsychology, Behavior, and Social Networking*, 24, 2021.

[79] Cody Coleman, Daniel Kang, Deepak Narayanan, Luigi Nardi, Tian Zhao, Jian Zhang, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark. *ACM SIGOPS Operating Systems Review*, 53, 2019.

[80] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100, 2017.

[81] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Net.*, 5, 1994.

[82] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with YCSB. In *Proc. of SoCC*, 2010.

[83] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *Proc. of the ACM conference on recommender systems*, 2016.

[84] Hao Dang, Feng Liu, Joel Stehouwer, Xiaoming Liu, and Anil K Jain. On the detection of digital face manipulation. In *Proc. of CVPR*, 2020.

[85] Michelle R. Davis. Students create fake online profiles to bully peers. Education Week, April 2012. `https://www.edweek.org/ew/articles/2012/04/04/27facebook.h31.html`.

[86] Ryan Daws. IDC: Half of AI projects fail for one in four companies. Tech-Forge Media, July 2019. `https://artificialintelligence-news.com/2019/07/09/idc-half-ai-projects-fail-companies/`.

[87] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of CVPR*, 2009.

[88] Paul Denisowski. Recognizing and resolving LTE/CATV interference issues. *White Paper, Rohde and Schwarz*, 2011.

[89] Michael A DeVito, Jeremy Birnholtz, and Jeffery T Hancock. Platforms, people, and perception: Using affordances to understand self-presentation on social media. In *Proc. of CSCW*, 2017.

[90] Amandeep Dhir, Torbjørn Torsheim, Ståle Pallesen, and Cecilie S Andreassen. Do online privacy concerns predict selfie behavior among adolescents, young adults and adults? *Frontiers in Psychology*, 8, 2017.

[91] Mariella Dimiccoli, Juan Marín, and Edison Thomaz. Mitigating bystander privacy concerns in egocentric activity recognition with deep learning and intentional image degradation. In *Proc. of IMWUT (Interactive, Mobile, Wearable and Ubiquitous Technologies)*, 2018.

[92] Won Donghyeon. Gender and race classification with face images. `https://github.com/wondonghyeon/face-classification`.

[93] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-driven video streaming for deep learning inference. In *Proc. of SIGCOMM*, 2020.

[94] Nicholas Dufour, Andrew Gully, Per Karlsson, Alexey Victor Vorbyov, Thomas Leung, Jeremiah Childs, and Christoph Bregler. Deepfakes detection dataset by Google & JigSaw.

[95] Serge Egelman, Andrew Oates, and Shriram Krishnamurthi. Oops, i did it again: Mitigating repeated access control errors on facebook. In *Proc. of CHI*, 2011.

[96] John Emmons, Sadjad Fouladi, Ganesh Ananthanarayanan, Shivaram Venkataraman, Silvio Savarese, and Keith Winstein. Cracking open the dnn black-box: Video analytics with DNNs across the camera-cloud boundary. In *Proc. of HotEdgeVideo*, 2019.

[97] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88, 2010.

[98] Jenny Fan and Amy X Zhang. Digital juries: A civics-oriented approach to platform governance. In *Proc. of CHI*, 2020.

[99] Biyi Fang, Xiao Zeng, Faen Zhang, Hui Xu, and Mi Zhang. FlexDNN: Input-adaptive on-device deep learning for efficient mobile vision. In *Proc. of SEC*, 2020.

[100] Biyi Fang, Xiao Zeng, and Mi Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proc. of MobiCom*, 2018.

[101] Lujun Fang and Kristen LeFevre. Privacy wizards for social networking sites. In *Proc. of WWW*, 2010.

[102] Qingsong Feng et al. Anomaly detection of spectrum in wireless communication via deep auto-encoders. *The Journal of Supercomputing*, 2017.

[103] Guo Freeman and Divine Maloney. Body, avatar, and me: The presentation and perception of self in social virtual reality. *ACM on Human-Computer Interaction*, 4(CSCW3), 2021.

[104] Ohad Fried, Jennifer Jacobs, Adam Finkelstein, and Maneesh Agrawala. Editing self-image. *Communications of the ACM*, 63(3), 2020.

[105] Timnit Gebru, Judy Hoffman, and Li Fei-Fei. Fine-grained recognition in the wild: A multi-task domain adaptation approach. In *Proc. of ICCV*, 2017.

[106] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proc. of CVPR*, 2012.

[107] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. of CVPR*, 2012.

[108] Sherri Gordon. Why kids are using Instagram to bully. VeryWellFamily, December 2019. `https://www.verywellfamily.com/how-kids-use-instagram-to-bully-460579`.

[109] Samuel Greengard. Will deepfakes do deep damage? *Communications of the ACM*, 63, 2019.

[110] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *Proc. of International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018.

[111] Tian Guo et al. Robust online time series prediction with recurrent neural networks. In *Proc. of DSAA*, 2016.

[112] Vijay K Gurbani et al. Detecting and predicting outages in mobile networks with log data. In *Proc. of ICC*, 2017.

[113] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. MCDNN: An approximation-based execution framework for deep stream processing under resource constraints. In *Proc. of MobiSys*, 2016.

[114] Rakibul Hasan, Eman Hassan, Yifang Li, Kelly Caine, David J Crandall, Roberto Hoyle, and Apu Kapadia. Viewer experience of obscuring scene elements in photos to enhance privacy. In *Proc. of CHI*, 2018.

[115] Rakibul Hasan, Yifang Li, Eman Hassan, Kelly Caine, David J Crandall, Roberto Hoyle, and Apu Kapadia. Can privacy be satisfying? on improving viewer satisfaction for privacy-enhanced photos using aesthetic transforms. In *Proc. of CHI*, 2019.

[116] Brandon Haynes, Amrita Mazumdar, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. Visual road: A video data management benchmark. In *Proc. of SIGMOD*, 2019.

[117] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

[118] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, (11), 2019.

[119] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[120] J.F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *Proc. of ECCV*, 2012.

[121] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9, 1997.

[122] Stephanie Houde, Vera Liao, Jacquelyn Martino, Michael Muller, David Piorkowski, John Richards, Justin Weisz, and Yunfeng Zhang. Business (mis) use cases of generative AI. *arXiv:2003.07679*, 2020.

[123] Roberto Hoyle, Luke Stark, Qatrunnada Ismail, David Crandall, Apu Kapadia, and Denise Anthony. Privacy norms and preferences for photos posted online. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 27, 2020.

[124] Ke-Jou Hsu, Ketan Bhardwaj, and Ada Gavrilovska. Couper: Dnn model slicing for visual analytics containers at the edge. In *Proc. of SEC*, 2019.

[125] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[126] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proc. of CVPR*, 2017.

[127] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A. Efros. Fighting fake news: Image splice detection via learned self-consistency. In *Proc. of ECCV*, 2018.

[128] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Victor Bahl, and Matthai Philipose. VideoEdge: Processing camera streams using hierarchical clusters. In *Proc. of SEC*, 2018.

[129] Serena Iacobucci, Roberta De Cicco, Francesca Michetti, Riccardo Palumbo, and Stefano Pagliaro. Deepfakes unmasked: The effects of information priming and bullshit receptivity on deepfake recognition and sharing intention. *Cyberpsychology, Behavior, and Social Networking*, 24, 2021.

[130] Anand Padmanabha Iyer, Li Erran Li, and Ion Stoica. Automating diagnosis of cellular radio access network problems. In *Proc. of MobiCom*, 2017.

[131] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph Gonzalez. Scaling video analytics systems to large camera deployments. In *Proc. of HotMobile*, 2019.

[132] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez. Spatula: Efficient cross-camera video analytics on large camera networks. In *Proc. of SEC*, 2020.

[133] Angela H. Jiang, Daniel L.-K. Wong, Christopher Canel, Lilia Tang, Ishan Misra, Michael Kaminsky, Michael A. Kozuch, Padmanabhan Pillai, David G. Andersen, and Gregory R. Ganger. Mainstream: Dynamic stem-sharing for multi-tenant video processing. In *Proc. of USENIX ATC*, 2018.

[134] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *Proc. of SIGCOMM*, 2018.

[135] Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. In *Proc. of CVPR*, 2020.

[136] Adam N Joinson, Ulf-Dietrich Reips, Tom Buchanan, and Carina B Paine Schofield. Privacy, trust, and self-disclosure online. *Human–Computer Interaction*, 25(1), 2010.

[137] Praveen Kaligineedi, Majid Khabbazian, and Vijay K Bhargava. Malicious user detection in a cognitive radio cooperative sensing system. *IEEE Transactions on Wireless Communications*, 9, 2010.

[138] Daniel Kang, Peter Bailis, and Matei Zaharia. BlazeIt: optimizing declarative aggregation and limit queries for neural network-based video analytics. In *Proc. of VLDB*, volume 13, 2019.

[139] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. NoScope: optimizing neural network queries over video at scale. In *Proc. of VLDB*, 2017.

[140] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. of CVPR*, 2019.

[141] Jiayun Ke. Shanghai wants law on radio spectrum. Shine.cn, March 2018. `https://www.shine.cn/news/metro/1803061282/`.

[142] Murat Kezer, Barış Sevi, Zeynep Cemalcilar, and Lemi Baruh. Age differences in privacy attitudes, literacy and privacy management on facebook. *Cyberpsychology: Journal of Psychosocial Research on Cyberspace*, 10, 2016.

[143] Will Knight. Facebook is making its own AI deepfakes to head off a disinformation disaster. MIT Technology Review, Sept. 2019. `https://www.technologyreview.com/2019/09/05/65353/facebook-is-making-ai-deepfakes-to-head-off-a-disinformation-disaster/`.

[144] Oleg Kolosov, Gala Yadgar, Sumit Maheshwari, and Emina Soljanin. Benchmarking in the dark: On the absence of comprehensive edge datasets. In *Proc. of USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2020.

[145] Paweł Korus. Digital image integrity–a survey of protection and verification techniques. *Digital Signal Processing*, 71, 2017.

[146] AM Koushik, Elizabeth Bentley, Fei Hu, and Sunil Kumar. A hardware testbed for learning-based spectrum handoff in cognitive radio networks. *Journal of Network and Computer Applications*, 106, 2018.

[147] Sanjay Krishnan, Adam Dziedzic, and Aaron J Elmore. Deeplens: Towards a visual data management system. *arXiv preprint arXiv:1812.07607*, 2018.

[148] Michelle Kwasny, Kelly Caine, Wendy A Rogers, and Arthur D Fisk. Privacy and technology: folk definitions and perspectives. In *CHI Extended Abstracts on Human Factors in Computing Systems*. 2008.

[149] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[150] Minsun Lee and Hyun-Hwa Lee. Social media photo activity, internalization, appearance comparison, and body satisfaction: The moderating role of photo-editing behavior. *Computers in Human Behavior*, 114, 2021.

[151] Qing Li. Cyberbullying in schools: A research of gender differences. *School Psychology International*, 2006.

[152] Rongpeng Li et al. TACT: A transfer actor-critic learning framework for energy saving in cellular radio access networks. *IEEE Transactions on Wireless Communications*, 13, 2014.

[153] Yifang Li, Nishant Vishwamitra, Hongxin Hu, and Kelly Caine. Towards a taxonomy of content sensitivity and sharing preferences for photos. In *Proc. of CHI*, 2020.

[154] Yifang Li, Nishant Vishwamitra, Bart P Knijnenburg, Hongxin Hu, and Kelly Caine. Effectiveness and users' experience of obfuscation as a privacy-enhancing technology for sharing photos. In *Proc. of CSCW*, 2018.

[155] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Harry Xu, and Ravi Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proc. of SIGCOMM*, 2020.

[156] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In ictu oculi: Exposing AI created fake videos by detecting eye blinking. In *Proc. of International Workshop on Information Forensics and Security (WIFS)*, 2018.

[157] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. In *Proc. of CVPR Workshops*, 2019.

[158] Kaitai Liang, Joseph K Liu, Rongxing Lu, and Duncan S Wong. Privacy concerns for photo sharing in online social networks. *IEEE Internet Computing*, 19(2), 2014.

[159] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proc. of ICCV*, 2019.

[160] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. of ECCV*, 2014.

[161] Liangkai Liu, Jiamin Chen, Marco Brocanelli, and Weisong Shi. E2M: an energy-efficient middleware for computer vision applications on autonomous mobile robots. In *Proc. of SEC*, 2019.

[162] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *Proc. of MobiCom*, 2019.

[163] Peng Liu, Bozhao Qi, and Suman Banerjee. Edgeeye: An edge service framework for real-time intelligent video analytics. In *Proc. of International Workshop on Edge Systems, Analytics and Networking*, 2018.

[164] Song Liu, Yingying Chen, Wade Trappe, and Larry J Greenstein. Aldo: An anomaly detection framework for dynamic spectrum access networks. In *Proc. of INFOCOM*, 2009.

[165] Song Liu, Larry J Greenstein, Wade Trappe, and Yingying Chen. Detecting anomalous spectrum usage in dynamic spectrum access networks. *Ad Hoc Networks*, 10(5):831–844, 2012.

[166] Ting Liu, Andrew W Moore, and Alexander Gray. New algorithms for efficient high-dimensional nonparametric classification. *Journal of Machine Learning Research*, 7, 2006.

[167] Xiaochen Liu, Pradipta Ghosh, Oytun Ulutan, BS Manjunath, Kevin Chan, and Ramesh Govindan. Caesar: cross-camera complex activity recognition. In *Proc. of SenSys*, 2019.

[168] Yao Liu, Peng Ning, and Huaiyu Dai. Authenticating primary users' signals in cognitive radio networks via integrated cryptographic and wireless link signatures. In *Proc. of S&P*, 2010.

[169] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proc. of ICCV*, 2015.

[170] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. of ICCV*, 2015.

[171] Taylor Lorenz. Teens are being bullied 'constantly' on Instagram. The Atlantic, Oct 2018. `https://www.theatlantic.com/technology/archive/2018/10/teens-face-relentless-bullying-instagram/572164/`.

[172] Andrea Lottarini, Alex Ramirez, Joel Coburn, Martha A Kim, Parthasarathy Ranganathan, Daniel Stodolsky, and Mark Wachsler. vbench: Benchmarking video transcoding in the cloud. *ACM SIGPLAN Notices*, 53, 2018.

[173] Emily Lowe-Calverley and Rachel Grieve. Self-ie love: Predictors of image editing intentions on facebook. *Telematics and Informatics*, 35(1), 2018.

[174] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on LFW with gaussianface. In *Proc. of AAAI*, 2015.

[175] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. Optasia: A relational platform for efficient large-scale video analytics. In *Proc. of SoCC*, 2016.

[176] Xudong Lv and Z Jane Wang. Perceptual image hashing based on shape contexts and local feature points. *IEEE Transactions on Information Forensics and Security*, 7, 2012.

[177] Huizi Mao, Taeyoung Kong, and William J Dally. Catdet: Cascaded tracked detector for efficient object detection from video. In *Proc. of MLsys*, 2019.

[178] Alessandra Mazzia, Kristen LeFevre, and Eytan Adar. The pviz comprehension tool for social network privacy settings. In *Proc. of SOUPS*, 2012.

[179] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal de Lara, and Blesson Varghese. Defog: fog computing benchmarks. In *Proc. of SEC*, 2019.

[180] Ian McGraw et al. Personalized speech recognition on mobile devices. In *Proc. of ICASSP*, 2016.

[181] Raman K Mehra and J Peschon. An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica*, 7(5):637–640, 1971.

[182] Kirsti Melville. The insidious rise of deepfake porn videos — and one woman who won't be silenced. ABC News Australia, August 2019. `https://www.abc.net.au/news/2019-08-30/deepfake-revenge-porn-noelle-martin-story-of-image-based-abuse/11437774`.

[183] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831*, 2016.

[184] Andrew D Miller and W Keith Edwards. Give and take: a study of consumer photo-sharing culture and practice. In *Proc. of CHI*, 2007.

[185] Alexander W Min, Kyu-Han Kim, and Kang G Shin. Robust cooperative sensing via state estimation in cognitive radio networks. In *Proc. of DySPAN*, 2011.

[186] Stephanie Mlot. Instagram launches new features to curb online bullying. PC Magazine, May 2020. `https://www.pcmag.com/news/instagram-launches-new-features-to-curb-online-bullying`.

[187] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

[188] Mainack Mondal, Günce Su Yilmaz, Noah Hirsch, Mohammad Taha Khan, Michael Tang, Christopher Tran, Chris Kanich, Blase Ur, and Elena Zheleva. Moving beyond set-it-and-forget-it privacy settings on social media. In *Proc. of CCS*, 2019.

[189] Mavuto M Mukaka. A guide to appropriate use of correlation coefficient in medical research. *Malawi medical journal*, 24, 2012.

[190] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for efficient video inference. In *Proc. of ICCV*, 2019.

[191] Joao C Neves, Ruben Tolosana, Ruben Vera-Rodriguez, Vasco Lopes, Hugo Proença, and Julian Fierrez. Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection. *IEEE Journal of Selected Topics in Signal Processing*, 14, 2020.

[192] Vinod Nigade, Lin Wang, and Henri Bal. Clownfish: Edge and cloud symbiosis for video stream analytics. In *Proc. of SEC*, 2020.

[193] Ana Nika et al. Empirical validation of commodity spectrum monitoring. In *Proc. of SenSys*, 2016.

[194] Shadi A Noghabi, Landon Cox, Sharad Agarwal, and Ganesh Ananthanarayanan. The emerging landscape of edge computing. *GetMobile: Mobile Computing and Communications*, 23, 2020.

[195] Oded Nov and Sunil Wattal. Social computing privacy concerns: antecedents and effects. In *Proc. of CHI*, 2009.

[196] Szabolcs Nováczki. An improved anomaly detection and diagnosis framework for mobile network operators. In *Proc. of DRCN*, 2013.

[197] Easwar A Nyshadham and David Castano. Affect and online privacy concerns. *Available at SSRN 2051044*, 2012.

[198] O. Onireti et al. A cell outage management framework for dense heterogeneous networks. *IEEE Trans. on Vehicular Technology*, 65, 2016.

[199] B Cli ord Neuman. Scale in distributed systems. *ISI/USC*, page 68, 1994.

[200] Timothy J O'Shea, T Charles Clancy, and Robert W McGwier. Recurrent neural radio anomaly detection. *arXiv preprint arXiv:1611.00301*, 2016.

[201] Ahmed Fawzi Otoom, Emad E Abdallah, Yousef Kilani, Ahmed Kefaye, and Mohammad Ashour. Effective diagnosis and monitoring of heart disease. *International Journal of Software Engineering and Its Applications*, 9, 2015.

[202] Sandeep P. Chinchali, Eyal Cidon, Evgenya Pergament, Tianshu Chu, and Sachin Katti. Neural networks meet physical networks: Distributed inference between edge devices and the cloud. In *Proc. of HotNets*, 2018.

[203] Chrisma Pakha, Aakanksha Chowdhery, and Junchen Jiang. Reinventing video streaming for distributed vision analytics. In *Proc. of HotCloud*, 2018.

[204] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2010.

[205] Francesco Pittaluga and Sanjeev J Koppal. Privacy preserving optics for miniature vision sensors. In *Proc. of CVPR*, 2015.

[206] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics (TOG)*, 37, 2018.

[207] Di Pu, Yuan Shi, Andrei V Ilyashenko, and Alexander M Wyglinski. Detecting primary user emulation attack in cognitive radio networks. In *Proc. of GLOBECOM*, 2011.

[208] A. Pumarola, A. Agudo, A.M. Martinez, A. Sanfeliu, and F. Moreno-Noguer. GANimation: One-shot anatomically consistent facial animation. In *Proc. of ECCV*, 2018.

[209] Hang Qiu, Xiaochen Liu, Swati Rallapalli, Archith J Bency, Kevin Chan, Rahul Urgaonkar, BS Manjunath, and Ramesh Govindan. Kestrel: Video analytics for augmented multi-camera vehicle tracking. In *Proc. of International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018.

[210] Lili Qiu, Paramvir Bahl, Ananth Rao, and Lidong Zhou. Troubleshooting wireless mesh networks. In *Proc. of SIGCOMM*, 2006.

[211] Kelly Quinn, Dmitry Epstein, and Brenda Moon. We care about different things: Non-elite conceptualizations of social media privacy. *Social Media+ Society*, 5(3), 2019.

[212] Sreeraj Rajendran et al. Distributed deep learning models for wireless signal classification with low-cost spectrum sensors. *arXiv preprint arXiv:1707.08908*, 2017.

[213] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *Proc. of INFOCOM*, 2018.

[214] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[215] Dimensional Research. What data scientists tell us about AI model training today. Alegion, May 2019. `https://content.alegion.com/dimensional-researchs-survey/`.

[216] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should I trust you?" explaining the predictions of any classifier. In *Proc. of SIGKDD*, 2016.

[217] Melody Lee Rood and John Schriner. The Internet never forgets: Image-based sexual abuse and the workplace. *Handbook of Research on Cyberbullying and Online Harassment in the Workplace*, pages 107–128, 2021.

[218] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *Proc. of ICCV*, 2019.

[219] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proc. of ICCV workshops*, 2015.

[220] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, 126, 2018.

[221] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. In *Proc. of CVPR Workshops*, 2020.

[222] Ariane Sampson, Huw G Jeremiah, Manoharan Andiappan, and J Tim Newton. The effect of viewing idealised smile images versus nature images via social media on immediate facial satisfaction in young adults: A randomised controlled trial. *Journal of orthodontics*, 47(1), 2020.

[223] Casey Schmidt. The absolute easiest way to remove metadata from photos. `https://www.canto.com/blog/remove-metadata-from-photo/`, 2020.

[224] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. of CVPR*, 2015.

[225] Selim Seferbekov. Youtube, twitter hunt down deepfakes. `https://github.com/selimsef/dfdc_deepfake_challenge`, 2020.

[226] Ahmed Selim et al. Spectrum monitoring for radar bands using deep convolutional neural networks. *arXiv preprint arXiv:1705.00462*, 2017.

[227] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting privacy against unauthorized deep learning models. In *Proc. of USENIX Security*, 2020.

[228] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *Proc. of CVPR*, 2017.

[229] Jacob Shermeyer and Adam Van Etten. The effects of super-resolution on object detection performance in satellite imagery. In *Proc. of CVPR Workshops*, 2019.

[230] Anmol Sheth, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas Sicker. Mojo: A distributed physical layer anomaly detection system for 802.11 wlans. In *Proc. of MobiSys*, 2006.

[231] Anna Cinzia Squicciarini, Smitha Sundareswaran, Dan Lin, and Josh Wede. A3p: adaptive policy prediction for shared images over popular content sharing sites. In *Proc. of ACM conference on Hypertext and hypermedia*, 2011.

[232] J. M. Such and N. Criado. Resolving multi-party privacy conflicts in social media. *IEEE Transactions on Knowledge and Data Engineering*, 28, 2016.

[233] Jose M Such, Joel Porter, Sören Preibusch, and Adam Joinson. Photo privacy conflicts in social media: A large-scale empirical study. In *Proc. of CHI*, 2017.

[234] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. of CVPR*, 2019.

[235] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[236] Kefeng Tan, Kai Zeng, Daniel Wu, and Prasant Mohapatra. Detecting spectrum misuse in wireless networks. In *Proc. of MASS*, 2012.

[237] Surat Teerapittayanon, Bradley McDanel, and HT Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *Proc. of ICDCS*, 2017.

[238] Marika Tiggemann, Isabella Anderberg, and Zoe Brown. Uploading your best self: Selfie editing and body dissatisfaction. *Body image*, 33, 2020.

[239] Viavi. Interference hunting is fundamental to quality service. RCR Wireless News, Nov. 2016. `https://www.rcrwireless.com/20161101/network-infrastructure/interference-hunting-fundamental-quality-service`.

[240] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57, 2004.

[241] Jessica Vitak, Kalyani Chadha, Linda Steiner, and Zahra Ashktorab. Identifying women's experiences with and strategies for mitigating negative effects of online harassment. In *Proc. of CSCW*, 2017.

[242] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *Proc. of SEC*, 2018.

[243] Run Wang, Lei Ma, Felix Juefei-Xu, Xiaofei Xie, Jian Wang, and Yang Liu. Fakespotter: A simple baseline for spotting ai-synthesized fake faces. *arXiv:1909.06122*, 2019.

[244] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A Efros. Detecting photoshopped faces by scripting photoshop. In *Proc. of ICCV*, 2019.

[245] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-generated images are surprisingly easy to spot... for now. In *Proc. of CVPR*, 2020.

[246] Wei Wang, Jin Zhang, and Qian Zhang. Cooperative cell outage detection in self-organizing femtocell networks. In *Proc. of INFOCOM*, 2013.

[247] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proc. of ECCV*, 2018.

[248] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *Proc. of HotCloud*, 2019.

[249] Honghao Wei, Yunfeng Jia, and Lei Wang. Spectrum anomalies autonomous detection in cognitive radio using hidden markov models. In *Proc. of IAEAC*, 2015.

[250] Hao Wu, Jinghao Feng, Xuejin Tian, Edward Sun, Yunxin Liu, Bo Dong, Fengyuan Xu, and Sheng Zhong. EMO: Real-time emotion recognition from single-eye images for resource-constrained eyewear devices. In *Proc. of MobiSys*, 2020.

[251] Xiongwei Xie and Weichao Wang. Detecting primary user emulation attacks in cognitive radio networks via physical layer network coding. *Procedia Computer Science*, 21:430–435, 2013.

[252] Xiufeng Xie and Kyu-Han Kim. Source compression with bounded dnn perception loss for iot edge computer vision. In *Proc. of MobiCom*, 2019.

[253] Lei Xu, Ting Bao, Liehuang Zhu, and Yan Zhang. Trust-based privacy-preserving photo sharing in online social networks. *IEEE Transactions on Multimedia*, 21(3), 2018.

[254] Ran Xu, Chen-lin Zhang, Pengcheng Wang, Jayoung Lee, Subrata Mitra, Somali Chaterji, Yin Li, and Saurabh Bagchi. ApproxDet: content and contention-aware approximate object detection for mobiles. In *Proc. of SenSys*, 2020.

[255] Cai-Ping Yan, Chi-Man Pun, and Xiao-Chen Yuan. Quaternion-based image hashing for adaptive tampering localization. *IEEE Transactions on Information Forensics and Security*, 11, 2016.

[256] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proc. of CVPR*, 2020.

[257] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. In *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[258] Xu Yao, Gilles Puy, Alasdair Newson, Yann Gousseau, and Pierre Hellier. High resolution face age editing. *arXiv:2005.04410*, 2020.

[259] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proc. of CVPR*, 2019.

[260] Xiao Zeng, Biyi Fang, Haichen Shen, and Mi Zhang. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In *Proc. of SenSys*, 2020.

[261] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A Lee. AWStream: adaptive wide-area streaming analytics. In *Proc. of SIGCOMM*, 2018.

[262] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *Proc. of NSDI*, 2017.

[263] Linyuan Zhang, Guoru Ding, Qihui Wu, and Zhu Han. Spectrum sensing under spectrum misuse behaviors: A multi-hypothesis test perspective. *IEEE Transactions on Information Forensics and Security*, 13(4), 2018.

[264] Tan Zhang, Aakanksha Chowdhery, Paramvir Victor Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *Proc. of MobiCom*, 2015.

[265] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proc. of CVPR*, 2017.

[266] Yuan Zhao, Jiasi Chen, and Samet Oymak. On the role of dataset quality and heterogeneity in model confidence. *arXiv preprint arXiv:2002.09831*, 2020.

[267] Xin Zheng, Yanqing Guo, Huaibo Huang, Yi Li, and Ran He. A survey to deep facial attribute analysis. *International Journal of Computer Vision*, 128, 2020.

[268] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

[269] Hongyu Zhu, Mohamed Akrout, Bojian Zheng, Andrew Pelegris, Amar Phanishayee, Bianca Schroeder, and Gennady Pekhimenko. Tbd: Benchmarking and analyzing deep neural network training. *arXiv preprint arXiv:1803.06905*, 2018.

APPENDIX A

EVALUATING VIDEO ANALYTIC PIPELINES WITH HETEROGENOUS WORKLOADS

A.1 Details of Our Dataset

Video source. To build our video dataset, we collected videos from various sources.

- *YouTube*: We first located videos and live streams on YouTube by searching keywords such as "traffic", "highway". Then, using YouTube-dl [34] or streamlink [25], we obtained each video's best resolution setting and downloaded it in mp4 format.
- *Waymo*: Part of our video dataset was from the Waymo Open Dataset [234]. Although the original dataset contains videos from 5 camera angles (front, front left, front right, side left, side right), we only selected videos of the front camera to be consistent with the dashboard videos we collected from YouTube.
- *KITTI*: We also included the labeled videos in the KITTI Vision Benchmark Suite [106].
- *MOT*: Videos from two existing datasets, MOT15 [149] and MOT16 [183], were also added to our dataset.

For each video, we first used FFmpeg [11] to resize it into 4 different resolutions (720p, 540p, 480p, and 360p), and then converted all videos into images (JPEG files).

Object detection results. We ran four different object detection models (FasterRCNN with ResNet101/ResNet50/Inception v2 and SSD with MobileNet) to obtain the object detection results over all video frames at four resolutions. All those detection models were downloaded from TensorFlow's detection model zoo [26]. The labels in all object detection results followed MSCOCO [160] dataset format. The object detection results obtained from full model (FasterRCNN with ResNet101) at highest resolution (720p) were used as the ground truth.

A.2 Pipeline Implementation

Our measurement study compared the performance of 7 VAPs on different video sets. To achieve a fair comparison, we made the following modifications on top of their original implementations.

- We implemented all 7 VAPs following the original papers. While some VAPs (such as NoScope [139] and AWStream [261]) do provide their prototype implementations, others do not. The original implementations of NoScope and AWStream are in different languages. To compare them in the same environment, we implemented all pipelines using Python and evaluated them on the same Linux machine with GPUs.
- Because our goal is to understand the performance of the cost-saving strategies applied by those pipelines, our implementation of those pipelines focused on the components related to cost saving.
- We evaluated the performance of all 7 VAPs on the same vision task, i.e., object detection. However, the cheap model used at camera-side in Vigil [264] and NoScope [139] were not for object detection, therefore, we switched their camera-side cheap model to small object detectors.

Moreover, each pipeline provides some configuration parameters (e.g., frame sampling rate of VideoStorm) so that pipeline operators could explore the pipeline's inherent accuracy-cost tradeoff to meet his own requirements on accuracy and cost. We chose those configuration parameters by emulating how VAPs should be used in practice. For each pipeline, the first few seconds of one video was used for profiling to choose the best configuration (minimal cost to achieve some target accuracy). In our experiments, for each video segment, we profiled on the first 1/3 of the video to obtain the best configuration, and then reported performance with the best configuration on the remaining 2/3 video. Our target accuracy (F1 score) was set to be 0.9. Here we list all the implementation details and configuration parameter setting we explored for each VAP.

VideoStorm [262]. VideoStorm uniformly samples frames, downsizes the sampled frames to a lower resolution, and processes these frames using a slightly less accurate yet much cheaper DNN model. Here the cost is measured as the compute cost, i.e., GPU usage. We noticed that full DNN model will resize each frame to the same size before taking it as input, so frame downsizing has small impact on GPU usage, therefore, we only include fame rate tuning and model selection in our implementation. Specifically, the configuration space we explored is frame sampling rate $\in [20, 15, 10, 5, 4, 3, 2, 1]$, and model selected from a set of pretrained models downloaded from TensorFlow object detection model zoo [26], including FasterRCNN with ResNet101/ResNet50/Inception v2 and SSD with MobileNet. All the models are pretrained on MSCOCO object detection dataset [160].

AWStream [261]. AWStream saves network cost (bandwidth) by applying a set of degradation functions to raw video and only sending the degraded video to server. We included two degradation functions, frame sampling and frame downsizing, in our implementation of AWStream. We explored the same space of frame sampling rate as used in VideoStorm, and the video was downsized to a set of different resolutions, including 540p, 480p and 360p.

Glimpse [70]. Glimpse saves compute cost and network cost by carefully sampling frames to send to the server and tracking the detected objects at the client side. The frame sampling mechanism relies on the pixel-level difference between frames and only triggers a frame transfer when pixel-level difference is above certain threshold. Similarly, tracking mechanism triggers a frame transfer when tracking fails. We implemented the frame difference by computing the number of pixels changed using OpenCV [19]. The criteria of a pixel change is the difference between two grayscale pixel values greater than 35. Instead of tracking with optical flow [59], we implemented the tracker using OpenCV KCF tracker [19, 120] in order to reduce the tracking computation cost and boost tracking accuracy.

Vigil [264]. To reduce bandwidth, Vigil runs a cheap analytic model on the camera to decide which frames/pixels in camera feeds contain most objects and sends only these regions (i.e., drop frames without objects and set non-object pixels to single color). We tried

non-DNN object detector (such Harr Cascade classifier [240]) as camera-side cheap model first, but the accuracy drops significantly on most of videos in our dataset. This is because Harr Cascade classifier has low recall, and objects missed on the camera side can not be recovered at server side. Therefore, for camera-side model, we also used a DNN model, FaseterRCNN with Inception v2, which is more accurate than Harr and much smaller (30%) compared to full DNN model.

NoScope [139]. NoScope applies two cost-saving strategies. It first identifies frames with significant pixel-level changes and only runs inference on those selected frames. During inference, it will first run a cheap DNN (which is fine-tuned on per video stream); only when the cheap DNN has low confidence score will a full DNN be triggered for inference. In the original design of NoScope, the target application is binary classification (i.e., one frame contains target object or not), therefore the cheap model can be very small and accurate (only a few convolutional layers), and thus combining two strategies, NoScope achieves two to three magnitude of speed-ups. However, to compare all VAPs fairly, we use object detection as our target application, so we replaced NoScope's small model with a small object detector (SSD with MobileNet). Because the cost gap between a small object detector and full DNN is much smaller than that between a small binary classifier and full DNN, the performance of NoScope we reported in Figure 3.4 is not as impressive as performance reported in original paper. But still, we can observe that the performance of NoScope varies significantly across videos.

Reducto [155]. Reducto uses frame filtering to save compute cost (GPU usage) and network cost (bandwidth). The frame filtering is achieved by computing low-level feature differences between video frames and ignoring frames whose low-level features are too close to earlier frames. After filtering the target video based on low-level features, only a subset of the video frames are sent to the full DNN model to generate final object detection results. Low-level features we implemented include pixel, area, corner, and edge features. The closeness between two consecutive video frames is determined by whether the feature difference are

above thresholds which are dynamically determined for each video at the profile stage.

DDS [93]. DDS saves network cost (bandwidth) by continuously sending low quality video to the server where a full DNN model determines whether to resend the video with higher quality to boost the inference accuracy. The client side of DDS exploits APIs implemented in FFmpeg to change video quality. The video quality tuning is achieved by selecting quantization parameter(QP) (from 26, 28, 30, 32, 34, 36, 38, 40). At the server side of DDS, the transmitted video is decoded and an object detection model, such as FasterRCNN with ResNet101, is used to provide detection results (including bounding boxes, object types, and confidence scores). The object detection results on low quality video serve as the guidance of high quality video transmitting in the next round. Then the object detection results on high quality video and those on low quality video are combined to obtain the final detection results.