12-2018

Exploring Photo Privacy Protection on Smartphones

David Darling
University of Arkansas, Fayetteville

University of Arkansas

Exploring Photo Privacy Protection on Smartphones

by

David Darling

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Bachelor of Science in Computer Science with Honors

APPROVED, THESIS COMMITTEE

Qinghua Li, Ph.D.
Honors Advisor and Committee Chair

Matthew Patitz, Ph.D.
Committee Member

Khoa Luu, Ph.D.
Committee Member

FAYYETEVILLE, ARKANSAS
November 2018

Abstract

The proliferation of modern smartphone camera use in the past decade has resulted in unprecedented numbers of personal photos being taken and stored on popular devices. However, it has also caused privacy concerns. These photos sometimes contain potentially harmful information if they were to be leaked such as the personally identifiable information found on ID cards or in legal documents. With current security measures on iOS and Android phones, it is possible for 3rd party apps downloaded from official app stores or other locations to access the photo libraries on these devices without user knowledge or consent. Additionally, the prevalence of smartphone cameras in public has reduced personal privacy, as strangers are commonly photographed without permission. To mitigate the privacy risk posed by apps and unwanted public photos, this research project explores 3 main topics: developing a two-step method including permission analysis and system call analysis to identify the possibility of 3rd party applications accessing sensitive photos without user knowledge, developing an automated classifier to identify and protect private photos in smartphone media storage, and creating an accurate computer vision model for identifying bystanders in photos, so that their faces might be later blurred or otherwise obfuscated to protect their privacy. The resulting data from the system call analysis will hopefully improve public awareness on the vulnerabilities created by downloading untrustworthy apps. The private photo classifier and bystander detection model are able to achieve acceptable accuracy on the test datasets and can be used in future works to implement working systems to protect individual privacy in the aforementioned threat cases.

Acknowledgments

I would like to thank Dr. Qinghua Li, my advisor, for all of his tremendous help and time to assist me through the undergraduate research process. I would also like to thank my research partner, Dr. Ang Li, for all of his great suggestions and advice during my first steps into mobile security topics. My committee members were also a great help during this time with useful advice.

Finally, I thank my friends and family for their continued faith in me and their encouragement throughout my undergraduate career. I would not have been able to accomplish what I have without their support.

Table of Contents

List of Tables

List of Figures

I. Introduction

A. Problem

With the advent of smartphone cameras, personal photography has exploded with millions of iOS and Android cell phone users purchasing devices each year. While this relatively recent development has brought the previously professional realm of photography to the everyday user, it has also presented an all new set of challenges towards protecting personal information. Some of the largest vulnerabilities that currently exist for privacy protection are the current app permissions protocols for iOS and Android devices. Under the existing scheme, users who download and install 3rd party apps are asked once to grant an app permission to access media storage and/or the phone's network connection. This permission, once granted, will then allow that app access to the internet and phone media essentially in perpetuity. This approach certainly makes security settings easy to manage for the user but fails to address specific cases where users might not want particular photo files read by apps which might then transmit them over a network. Because 3rd party apps are almost never fully transparent in their behavior, users must currently rely on automated screening procedures implemented by Apple and Google to protect them from potentially malicious behavior from apps on the App Store and Google Play Store respectively. These screening processes have already been shown to be ineffective in alarming security breaches such as the discovery of 145 malware-infected Google Play apps [1], or the recent trend of researchers finding unauthorized data exfiltration in multiple App Store apps [2]. Beyond official app stores, users who wish to download apps from other sources often have no protection whatsoever from hidden malicious behavior in apps. The failure of these screening processes coupled with the lack of individual file access control clearly presents opportunities for attackers to access and transmit private photos along with any information contained within.

In addition to the dangers of unauthorized photo accesses, the privacy of individuals in public has also been adversely affected by the onset of smartphone photography. A large number of photos taken in public locations tend to contain random bystanders. With recent estimates of roughly 1.2 trillion digital photos being taken worldwide in 2017 [3], the number of bystanders being captured unawares is likely extremely high. There is currently no common system of ensuring the privacy of strangers in public photos although companies such as Google have implemented systems for automatically blurring faces on photos captured for Google Street View. Many individuals may want their faces obfuscated or otherwise removed from other peoples' photos especially when these photos might be uploaded to social media or some other publicly accessible location. This desire for privacy creates a demand for an automated system of detecting and removing faces which are not the targets of photos which could possibly be included in social media platforms to give users the option of removing strangers from their photos.

B. Proposed Solutions

A new method for monitoring the behavior of Android applications has been developed to determine whether a given app is making photo read accesses without user knowledge. To demonstrate the efficacy of this method, the 15 top, free apps on the Google Play Store have been analyzed to track their file accesses. This behavior tracking is achieved by recording all file read system calls within the photo media directory on an emulated Android device. Any app which is found to read from the media directory and possesses internet access should be considered to be able to leak user photo data to unauthorized entities. However, this information on app behavior is not meant to definitively identify actual malicious behavior, but rather demonstrate how common the potential for information leaks are in modern app stores.

Potential threats posed by untrustworthy apps towards private photos can be mitigated with a system consisting of an automated photo classifier to identify photos which need to be protected. This system would need to be implemented into future Android and iOS releases as an integral part of the operating systems similar to how current app permissions schemes are implemented in order to be effective. The photo classifier maintains responsibility for identifying images which contain potentially sensitive information such as legal documents, id cards, or even some selfies which a user might not want leaked or viewed by anyone else. This identification would occur upon saving any photo file to the device media directory. Upon identifying a private photo, the file's name and path would be saved to a cache. A device's OS would then need to perform a single additional security check anytime a read request is made for the photo directory from an installed app; accessing the cache to ensure the photo is not listed. If the photo is listed, the read request could be terminated at the user's discretion. Several prototypes of the classifier described above have been developed for and tested to measure classifier accuracy across different models and learning algorithms. On either iOS or Android, the changes will result in strengthened photo privacy for users, as read requests could be reliably managed.

Addressing the problem of bystander privacy in public photos also involves an automated system of bystander detection. Computer vision techniques can be applied to recognize and distinguish the targets of photos from strangers in photos with an acceptable degree of accuracy. These models require the use of several distinguishing facial features such as width, height, blurriness, and eye direction to be effective. In some cases, different machine learning algorithms offered improved accuracy over others or improved performance, so a detailed comparison of various algorithms is presented to demonstrate the benefits and

drawbacks of each system. By recognizing bystanders' faces in photos, it should then be possible to implement some type of obfuscation technique such as face blurring to ensure strangers' privacy is maintained, although implementation of such techniques is not the focus of this project.

C. Contributions

The contributions of this thesis are summarized as follows:

- A new methodology is devised for monitoring Android applications to determine if image read operations are occurring without user knowledge. This process uses system call tracing to monitor real-time behavior which is a marked improvement over just performing permission extraction and analysis.
- Several private photo classifiers are implemented and evaluated using state of the art neural networks and extracted histogram of oriented gradients (HOG) features with a support vector machine (SVM) classifier.
- An extended and improved feature-based model for automatic identification of target and bystander faces in photos is designed. Compared with the state-of-the-art model developed by Ang Li [7], our model adds a new gaze-tracking feature and redesigns the facial position and size features. These changes are intended to increase the overall accuracy of the model.

D. Thesis Organization

The remainder of the thesis is organized into four chapters. Chapter 2 provides an overview of the topics presented in later chapters along with the relevant information necessary

to understanding Android OS architecture, Android application security permissions, machine learning based photo classification, and computer vision techniques surrounding facial recognition. Chapter 3 details the methods and results of analyzing the system calls of various 3rd party apps along with their security manifests. The information gained from the dynamic analyses of these apps is compiled into a single table in the second section. Chapter 4 contains the procedures involved in creating the public/private photo classifier. Performance considerations are also discussed in this chapter. Chapter 5 pertains to the target/bystander acquisition models, with the first section detailing the selection of features and methods of capturing them. The second section compares the accuracy and results from the learning algorithms chosen for this approach. Chapter 6 offers a conclusion to the topics presented in prior chapters along with considerations for future work to improve the solutions presented.

II. Background

A. Android Architecture and System Calls

The Android operating system is an open source collection of software components primarily intended for mobile devices. At its core, the OS is based upon the Linux kernel and makes use of the core features provided by the kernel such as memory management and security [4]. The main components of the software stack are outlined in figure 1:

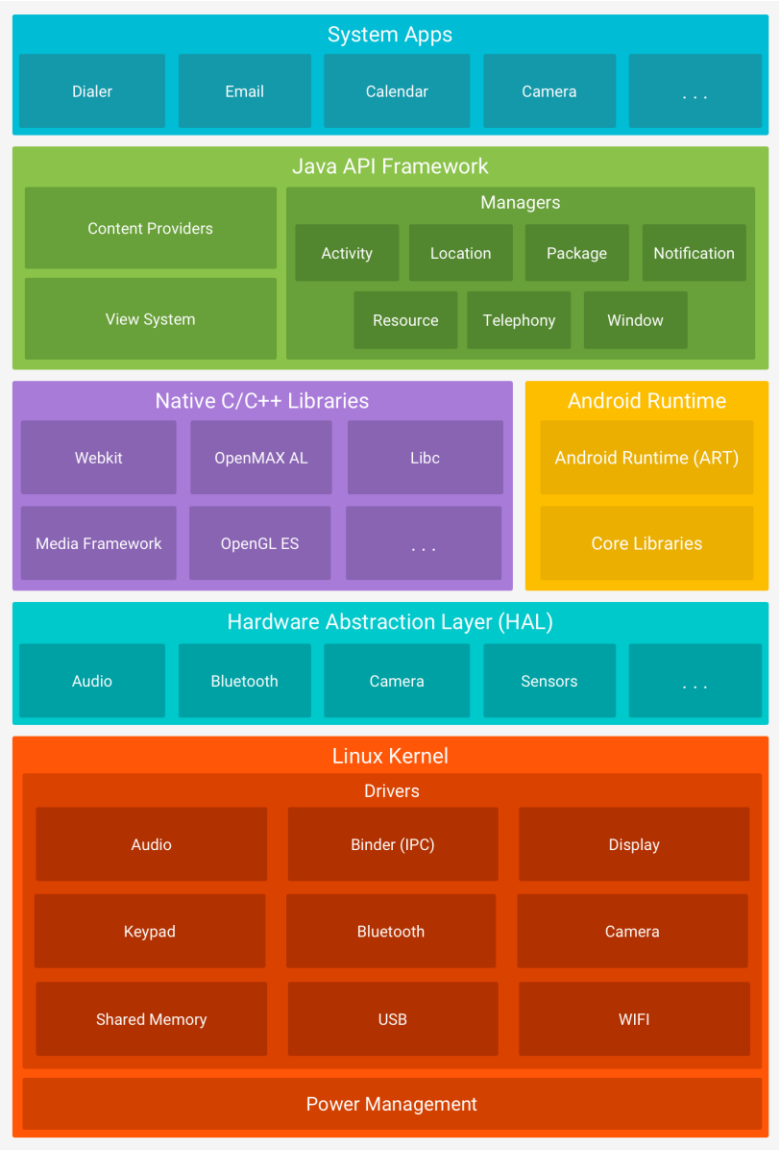


Figure 1: The Android software stack taken from https://developer.android.com/guide/platform/

The system calls of processes running on Android are requests made to the kernel for some hardware level operations to be performed. For example, to read some file, a running application would call the Linux read function with the following signature: *ssize_t read(int fd, void *buf, size_t count).* Generally, when developing an application, the developer makes use of libraries or APIs rather than directly writing system calls, but all higher-level calls are eventually translated to basic functions carried out by the OS. In this way, tracing the lowest level activity of a process allows a completely transparent look at what sort of activity is taking place. Because Android makes use of Linux at its core, all of the system calls made by processes are Linux calls. This similarity makes it possible to use Linux command line utilities. Of these utilities, Strace allows the tracing and recording of nearly any process' system calls by using the ptrace feature of the kernel [5]. By using Strace, the system call activity of any installed app can be recorded from the moment of its launch to the time the process is terminated including while the application is running in the background.

B. Android Application Security Permissions

All android apps must specify the permissions they require in their app manifest file. Permissions are classified based on their perceived threat to end users' privacy. The various classifications for Android permissions are normal, signature permissions, and dangerous permissions. In this scheme, only dangerous permissions require express user authorization upon app installation to be granted. These are permissions which could be extremely harmful with untrustworthy apps such as sending SMS messages, having access to the device camera, and recording audio with the device microphone [6]. Normal permissions, on the other hand, can be listed as a group in the app manifest and are always granted upon installation by the OS. Internet access is included in normal app permissions. For apps to be able to read from a device's media

gallery, they need to request the READ_EXTERNAL_STORAGE permission which is included in the dangerous class. The combination of these permissions theoretically grants any app the ability to read from a user's media gallery and transmit any data over the device internet connection. This topic has been covered in greater detail by Dr. Ang Li, who performed permission extraction of hundreds of Android applications to determine the number of popular applications which possess both external storage and internet permissions [7].

C. Photo Classification and Machine Learning

Photo classification using machine learning algorithms is a well-researched topic with many large companies devoting resources towards developing accurate models for general classification. State of the art photo classifiers are generally implemented as deep convolutional neural networks able to extract millions of features from images for training and often can require many GPU hours to reach acceptable accuracy. Additionally, in order to train a general classifier from scratch, an enormous number of images would be required to correctly train the model to be able to extract useful features in the general case. This firmly places developing new, modern photo classifiers in the realm of major companies such as Google's TensorFlow division which has released several general image classifiers for public use. General photo classifiers are often tested on the ImageNet Large Visual Recognition Challenge to benchmark their performance against other competitors. Error rates are measured as top-5 results which means a guess is counted as incorrect if the top 5 most confident guesses by the model are all incorrect. The most recent TensorFlow model dubbed Inception-v3 was able to achieve a top-5 accuracy of 96.54% [8].

In order to adapt these excellent classifiers to specific tasks such as recognizing ID cards or selfies in images to flag them, it would normally be required to expend large amounts of

time and resources to achieve acceptable accuracy. However, a technique known as transfer learning has been developed in recent research which enables a fraction of the normal computation time required to develop a useful classification model. Transfer learning on image classifying neural networks uses the fact that the network has been trained to be able to identify and extract useful features from photos for identification in the general case. This ability to extract useful features can then be transferred to new types of images by training only the top layer of the neural net. Training just the top layer takes significantly less time, as there are orders of magnitude fewer propagations and computations. The figure below shows the architecture of the Inception-v3 model along with the specific layers affected by transfer learning.

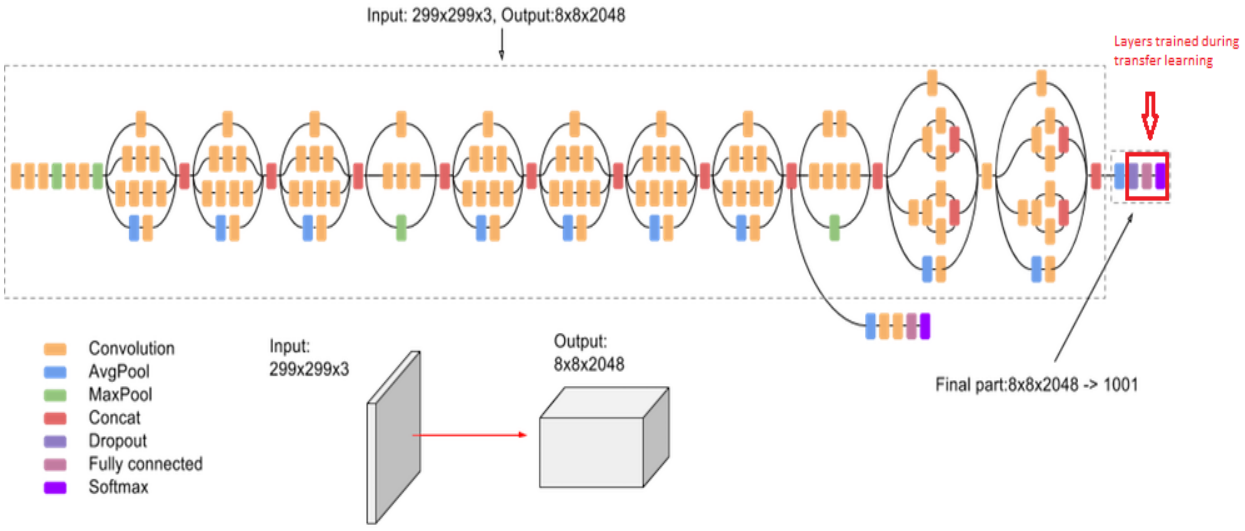


Figure 2: High level architecture of the Inception-v3 model taken from https://cloud.google.com/tpu/docs/inception-v3-advanced. Edited to show layers affected by transfer learning.

This figure illustrates how complex modern classifier networks can be. In this case, the final feature vector extraction layers can be seen on the far right of the image. The classification layers are the final 3 dropout, fully connected, and softmax layers seen surrounded by the red box. These are the only portions of the model that are affected during the training process.

Transfer learning allows for excellent accuracy with minimal spent computation time, but there are other methods of image feature extraction besides using neural networks that could be useful. There are many sets of image features which have been developed for gleaning information out of visual patterns. These include things such as blobs, edges, corners, and ridges all of which can be detected with various algorithms to describe something useful in an image. Scale-invariant feature transform (SIFT) is one such algorithm which calculates key points in images which can then be used for object recognition or a variety of applications. Speeded up robust features (SURF) is an algorithm inspired by SIFT which greatly speeds up the feature extraction process and offers robustness against image transformations as the name suggests. The histogram of oriented gradients (HOG) feature descriptor is a departure from the other two algorithms, as it functions by dividing an image into small sections and computing the histogram of gradient directions for each cell which gives edge directions [9]. An example visualization of HOG descriptors can be seen in figure 3 below.

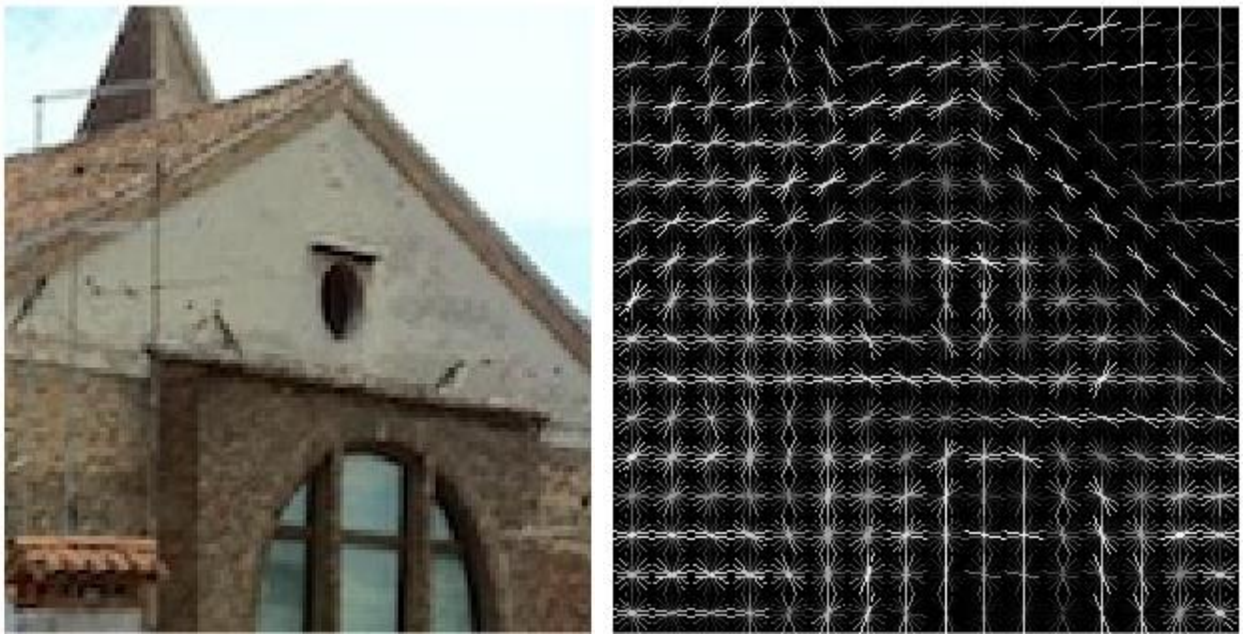


Figure 3: Visualization of HOG descriptors computed on a sample image taken from http://www.vlfeat.org

These feature extraction algorithms can be used in lieu of the more complex and modern neural networks to recognize objects in images. Any sort of supervised learning algorithm could be trained using these extracted features to learn the correct image classes. Some common supervised learning algorithms other than neural networks include support vector machines, linear regression, and decision trees. Each algorithm offers various benefits in supervised learning problems.

D. Facial Detection and Identifying Targets/Bystanders

Facial recognition is another rapidly developing sector of computer vision which involves automated detection and recognition of people based on extracted facial features. There are many different useful features which can be used to uniquely identify faces, but this thesis focuses on simple facial detection which only requires that a model be able to recognize the existence of a human face in a picture. This more general task can be accomplished with varying

accuracy using any of the feature extraction algorithms described in the previous section, as object detection can be easily applied to human faces. The algorithms work just as they would on any other object needing to be recognized but using such a basic model does not allow for extracting facial-specific information. This information such as face height, width, rotation, and many other descriptors can be extremely useful when attempting to make distinctions between detected faces.

In this portion of the research, the main goal is being able to develop a machine learning model which can make distinctions between people who are the focus of a photo and random strangers who happen to be captured in the same picture. This specific application of facial detection is far less researched than a topic such as unique face identification, but a prototype model has been developed by researcher Ang Li and is described in a section of his doctoral dissertation [7]. Ang was able to develop a method of extracting several relevant facial features which were believed to be important in distinguishing between the two types of faces. These facial features were blurriness, pitch, roll, yaw, smiling, face size, and face position. Using all of these in combination, he was able to produce several facial classifiers with varying learning algorithms to compare their accuracy. Ang was able to achieve 93.27% classification accuracy using a gradient boosted decision trees algorithm on his dataset which demonstrates the developed model was effective at distinguishing targets from bystanders. This thesis will focus on improving this model by making a series of changes to the ways several parameters are calculated such as face size and position. Additionally, a new parameter will be added which measures the relative gaze direction of a person's eyes with respect to the camera lens. These changes are outlined in detail in chapter 5.

To give a visual demonstration of the problem being tackled by these models, Figure 4 demonstrates an example image which could be included in a dataset and contains several bystanders behind the two targets Prince Harry and Meghan Markle. An effective model would be able to distinguish the strangers' faces from the actual target faces so that the bystander faces might be obfuscated if needed to improve privacy. In the example image, the targets are located in the foreground and are completely in focus. The bystanders are much less focused, and their faces are relatively smaller than the targets.

Figure 4: An example picture containing two targets and multiple strangers. Original photo by Chris Jackson | Getty Images.

III. Permissions and Behavior Analysis of Android Applications

A. Overview

A method for analyzing any given Android application's behavior has been devised to look for media directory accesses and the potential for user data leakage. This process should provide a means for revealing the exact image read operations performed by 3rd party apps and should identify any applications which warrant a more detailed examination to identify any potentially malicious behavior. In order to demonstrate the effectiveness of this methodology for detecting media accessing behavior, the top 15 free apps from the Google Play Store were selected for permission testing and behavior analysis. This two-step process is an enhancement and further development over Ang Li's permission extraction analysis [7] by incorporating running behavior analysis. In this procedure, the permission testing portion is meant to search for any vulnerability posed by an app's permission settings which are determined when the user runs the app for the first time and allows permissions based on the app's requests. Once an app has been confirmed to be able to leak user data, the next portion of the test can be carried out by dynamically analyzing its behavior through system calls.

Application system calls are recorded using the Strace process monitoring utility. This involved filtering all the system calls made by the main processes of the apps to record just the ones requesting read operations on the test device's media directory. This allows the activity of the app to be monitored so that any suspicious read operations can be detected such as reading while the phone is locked or while an app is running in the background. Although detecting such operations does not necessarily indicate malicious behavior, it does demonstrate the fact that applications may be accessing user photos without the user being aware.

B. Two-Step Analysis

The first step is permission analysis. To gain access to each app's manifest file, the APK file for each app is first downloaded. This can be achieved by finding each desired app on an APK repository site. The specific one used was APKMirror.com which is widely regarded as a secure and monitored service (https://www.apkmirror.com/). This additional step is required because official repositories such as the Google Play Store do not allow easy access to the actual APK files. To be absolutely sure the mirror downloads of the apps were the same as the Google Play downloads, they can be compared using checksums with the most recent official versions of the apps. Once the APK credentials have been confirmed, extracting the AndroidManifest.xml file is a straightforward process consisting of using Android Studio's analyze APK tool [10]. The extraction process outputs the Android manifest file closely to the original, pre-build version so that the exact permissions the app uses are listed in plain text within. Any app which uses normal permissions has internet access according to the Android permissions standards, and any app which uses the external storage read permission must explicitly list it in the manifest. Of the applications tested and listed in Table 1, only the Weather Channel App does not have a combination of INTERNET and READ_EXTERNAL_STORAGE permissions which is required to be able to leak image data.

The second step is behavioral analysis. This portion of this research focuses on recording any system calls by the main application process which attempts to access the media directory on an emulated Android device. The Android emulator system images do not allow debugging tools to be run on phones with Google Play Store functionality, so applications on the test device have to be manually installed. To do this, the emulated device is started, and a connection is then established to the emulated phone using the Android ADB shell utility

launched from the host computer's command line. The ADB shell is then capable of manually installing a downloaded APK file onto the connected device from the host computer. The installed application can be launched from the phone just as an app downloaded from the Google Play Store would be.

Once the desired applications are installed, they are launched, and their main processes can be located by running the *top* Linux utility on the ADB shell to list all running processes. Once the main processes are found, their *pids* are recorded to be used with Strace which is also launched from the ADB shell. Any system calls accessing the phone's media directory are set to be recorded into a log file for each application.

C. Results

In our experiment, the specific device being used is a virtual Google Pixel 2 with Android 9.0 installed. Every tested app was then left to run in the foreground, background, and while the device was locked for 30 minutes in each state. The results of these recordings are presented in Table 1 on the following page.

App Name	Foreground Activity	Background Activity	Phone Locked Activity
Facebook Messenger	YES	NO	NO
Instagram	YES	NO	NO
Snapchat	YES	NO	NO
WhatsApp	YES	NO	NO
Netflix	NO	NO	NO
YouTube	YES	NO	NO
Wish	YES	NO	NO
Spotify	NO	NO	NO
Cash App	YES	NO	NO
Walmart	YES	NO	NO
Amazon	YES	NO	NO
PayPal	YES	NO	NO
Venmo	YES	NO	NO
SoundCloud	NO	NO	NO
The Weather Channel	NO	NO	NO

Table 1: Recorded Photo Gallery Access Activity

These results demonstrate that none of the applications tested were performing media accesses while the apps were running in the background or while the phone was locked. The majority of applications made accesses to the media folder while they were running in the foreground. This most likely suggests that any accesses being made were to allow users to use their saved photos in the app for various services. This behavior is not necessarily surprising, as most of the companies involved in this test have very well-defined reasons for accessing user data outlined in their privacy policies which generally do not allow for accesses other than to use photos with in-app features. However, this testing method could be used on less trustworthy applications to ensure no strange activity is occurring while the apps are running. Although media read request activity occurring in the background or while the phone is locked does not necessarily indicate malicious activity is occurring, it would certainly warrant a closer investigation into why such requests are being made. The analysis method outlined in this thesis is easily reproducible and should prove very valuable for screening less well-known apps in the future.

IV. Private Photo Classifier

A. Approach

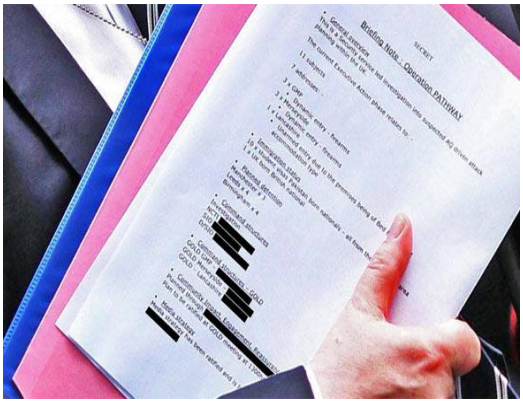
To provide advanced protection for private photos and prevent unauthorized accesses to them, it is necessary to identify private photos first. Manual identification of private photos is tedious and even infeasible when a user has many photos on his phone. This chapter describes the implementation and evaluation of an automated image classifier based on machine learning models which can offer highly accurate predictions about whether a given photo contains some sort of sensitive data. The first step to producing such a classifier is obtaining a reasonably sized dataset to train models with. Unfortunately, there are almost no publicly available datasets containing private photos, a fact which can be attributed to the nature of the contents of such photos. Thus, a custom dataset was created with a total of 5 classes: public, selfie, ID card, document, and family portrait. Example images from all five classes are shown in figure 5. Sample photos for each class were scraped from public sources such as Google Images to create a dataset of around 1000 photos with 200 photos in each class. Although this dataset size is relatively small when compared with some modern, publicly-contributed training sets, it was sufficient for training several accurate models.

In order to discover the learning algorithm and model which provides the highest accuracy, several different approaches are compared. Several state of the art deep convolutional neural networks (DCNNs) are trained and tested. Although these advanced models are expected to achieve excellent results, a more traditional approach is also used to provide a baseline accuracy measurement. This makes use of a multiclass support vector machine which is trained on extracted histogram of oriented gradients (HOG) features. HOG features are chosen for their reported utility in object recognition applications compared with other popular photo feature

extraction algorithms [9]. All of these methods' accuracy and feasibility on mobile devices are compared to find the most accurate and performant model achievable.

Public Class

Selfie Class



Document Class

ID Card Class

Family Portrait Class

Figure 5: Example images from each of the 5 classes. The public class image is a landmark and does not contain any sensitive data. The other images depict things users might not want leaked or accessed to protect their personal privacy. All images taken from Google Images through bulk downloads.

B. Algorithm/Model Comparison

The first model tested was the support vector machine (SVM) trained on HOG features extracted from the training portion of the image dataset. Approximately 90% of the dataset was used for training with the remaining 10% of images being used for the test set. The multiclass SVM algorithm used was provided by the OpenCV library along with the HOG implementation. To extract the HOG features, images were first resized to a standard of 500 x 500 pixels. This is due to the inability of OpenCV's implementation of HOG to deal with variable image sizes. The interpolation method used for this was pixel area relation, but other interpolation algorithms should yield very similar results for the sizes of images dealt with in the dataset, as the resizing generally did not result in a significant difference in image size. Once the processing was completed on all images, the HOG features were computed and used to train the multiclass SVM. After tuning the SVM hyper parameters *C* and *gamma*, final test accuracy resulted in 81.3%. This result forms the baseline performance for the more modern image recognition algorithms.

The DCNN models tested were used with Google's TensorFlow library. This is due to the library's ease of use and compatibility with widely used image recognition models such as the famous Inception-v3 and Inception-v4 models. A variety of models were selected to be compared for their performance requirements and their reported accuracy in large scale image recognition competitions. The DCNN models can be split into two main categories: mobile targeted and computationally demanding. Models such as Inception-v4 are generally designed for use on high-performing computer hardware which is not available on smaller mobile devices. These types of models are expected to result in extremely high-accuracy results, but they might not be suitable for use on a mobile device due to their size and computation demands for

prediction operations. Mobile targeted models such as NASNet (mobile) make use of less computationally complex operations but might result in lower accuracy. Table 2 describes the models tested and what category they fall into.

DCNN Model Type	Model Name
Mobile Targeted	MobileNet
Mobile Targeted	NASNet-A (mobile)
Computationally Demanding	Inception-v3
Computationally Demanding	Inception-v4
Computationally Demanding	NASNet-A (large)

Table 2: Tested DCNN models and their performance classification

A transfer learning approach was taken to train the models due to the limited dataset size. This enables the models to make use of the previously learned classification process while teaching them to recognize the 5 custom classes of photos: public, selfie, ID card, documents, family portrait. Once the transfer learning process was completed for each model, they were each evaluated on a randomly selected test portion of the dataset. The average results for 10 evaluations per model are listed in Table 3.

Model Name	Average Test Accuracy Over 10 Evaluations
NASNET-A (large)	94.1%
Inception-v4	91.4%
Inception-v3	91.3%
NASNet-A (mobile)	89.7%
MobileNet-v2	88.5%
SVM + HOG Features	81.3%

Table 3: Comparing the Classification accuracy of tested DCNN models and the SVM model

Clearly, the computationally demanding models outperformed all of the mobile models by at least 1% or more. NASNet-A (large) in this case was the single best classification model tested with an accuracy of over 94%. However, this model requires the most computation operations per prediction out of all of the other tested models, and may not be suitable for the smaller, mobile ARM architecture processers. NASNet-A (mobile), by comparison, experiences a 4.7% reduction in accuracy but requires only an average of 564 million multiply-add operations per image where NASNet-A (large) requires 23.4 billion such operations for a single image, a remarkable reduction in computation of 97.6% [11]. This makes the NASNet-A (mobile) model highly desirable for applications where larger hardware might not be available.

V. Target/Bystander Recognition

A. Approach

Improving the existing model [7] for identifying targets and strangers in photos requires careful selection of features to use and planning on how to improve the ways these features are computed. The existing model includes the following features: smiling, face size, face position, facial pitch, facial roll, facial yaw, and facial blur [7]. The features which are most effective in predictions from that model are left unchanged such as the Gaussian blurriness feature which is a major contributor to accuracy in the existing model. The features changed in the proposed approach are the face size and face location computation. Additionally, a new feature is added to the model which measures the eye gaze direction of each detected face relative to the camera's point of view.

The original method for computing the face size in Li's model [7] involves taking the area of the square region given by the Android SDK's FaceDetector class. This method is able to provide a useful metric for face size, but the square area can often be significantly larger than the detected face. Additionally, some square detection areas will be more accurately fitting than others which can misrepresent the true face size in an image. To create a more accurate measurement, a different framework, OpenFace, is used in this thesis for its state of the art facial detection algorithms and feature extraction modules [12]. OpenFace is able to extract multiple points around a face as shown in Figure 6. These point coordinates can then be used to create a highly accurate measure of a person's facial height and width. The proposed new model makes use of this method to replace the original face size calculation with the square area computed from the face height and width coordinates.

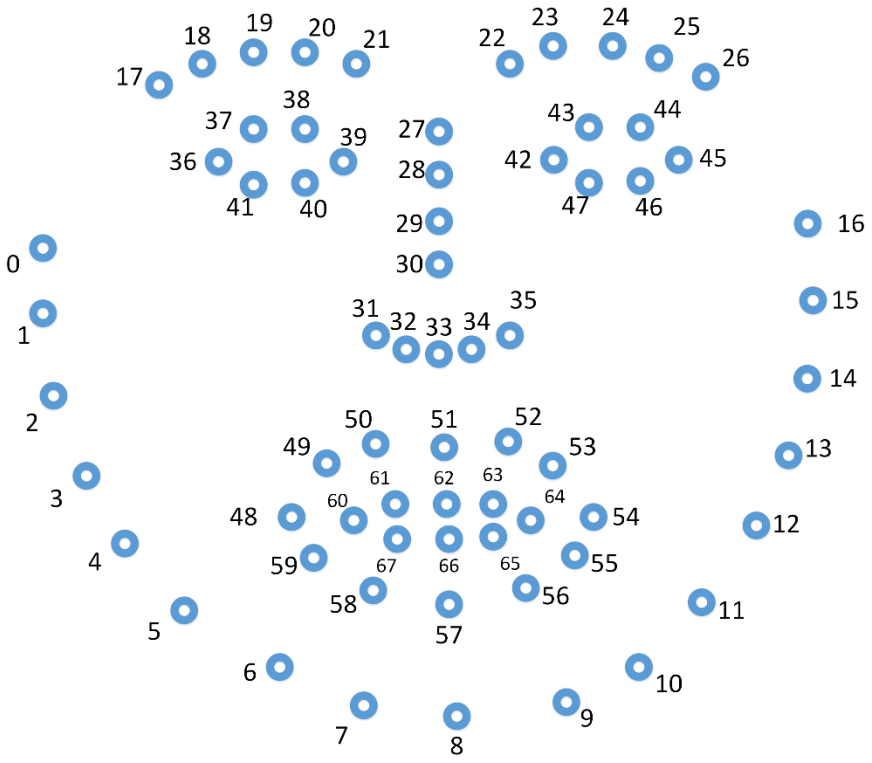


Figure 6: A diagram of all output facial landmark points extracted by OpenFace. Taken from: https://github.com/TadasBaltrusaitis/OpenFace/wiki/Output-Format

The computation of face location is also altered from the original model [7]. The original method splits an input image into 3 sections: left, middle, and right in the horizontal direction. This enables a binary classification system where a face located outside the middle section of an image is unlikely to be a target in an image. To better adapt this method to a feature-based classification model, the parameter is changed to the distance the detected head deviates from the absolute center of the image in the x and y pixel coordinate directions. This parameter change results in a spectrum of values so that the likelihood of the face being the target continuously increases the lower the measured deviation from the camera is and vice versa. The idea behind this change is that strictly classifying facial position in a binary manner can lead to very rigid

prediction guidelines which could negatively impact a classifier's accuracy. By creating a distance-based parameter, the prediction should have much less heuristic rigidity.

The final modification to the original classification model is the addition of an entirely new eye gaze measurement. This measurement can be implemented using OpenFace's gaze tracking module [13]. The module measures any detected face's eye gaze direction with respect to the camera sensor. The output is the gaze vector for each eye which can be traced to the camera origin. A greater traced difference from the camera origin indicates a likelihood the person is looking away from the camera whereas a relatively small difference from the origin indicates the person is looking at or very nearly at the camera lens when the picture is taken. Example processed images with gaze tracking and feature extraction are shown in Figures 7 and 8. While this metric will not always correctly aid in identifying targets (strangers will sometimes be looking at the camera when a picture is taken, and targets will sometimes be looking away), it should be the case that the vast majority of targets in photos will also possess identifying attributes such as a focused face or centrally located face. Strangers who are looking at the camera should often possess opposing features such as having their face turned away or being in the background of the image. This plethora of identifying features should mitigate the majority of cases where strangers are looking at the camera.

Figure 7: Example Processed Images using OpenFace with gaze detection shown by bright green vectors. Most of the subjects are gazing at or near the camera sensor. Top photo by Steve Granitz | WireImage. Bottom photo taken from https://pbs.twimg.com/media/BhxWutnCEAAtEQ6.jpg

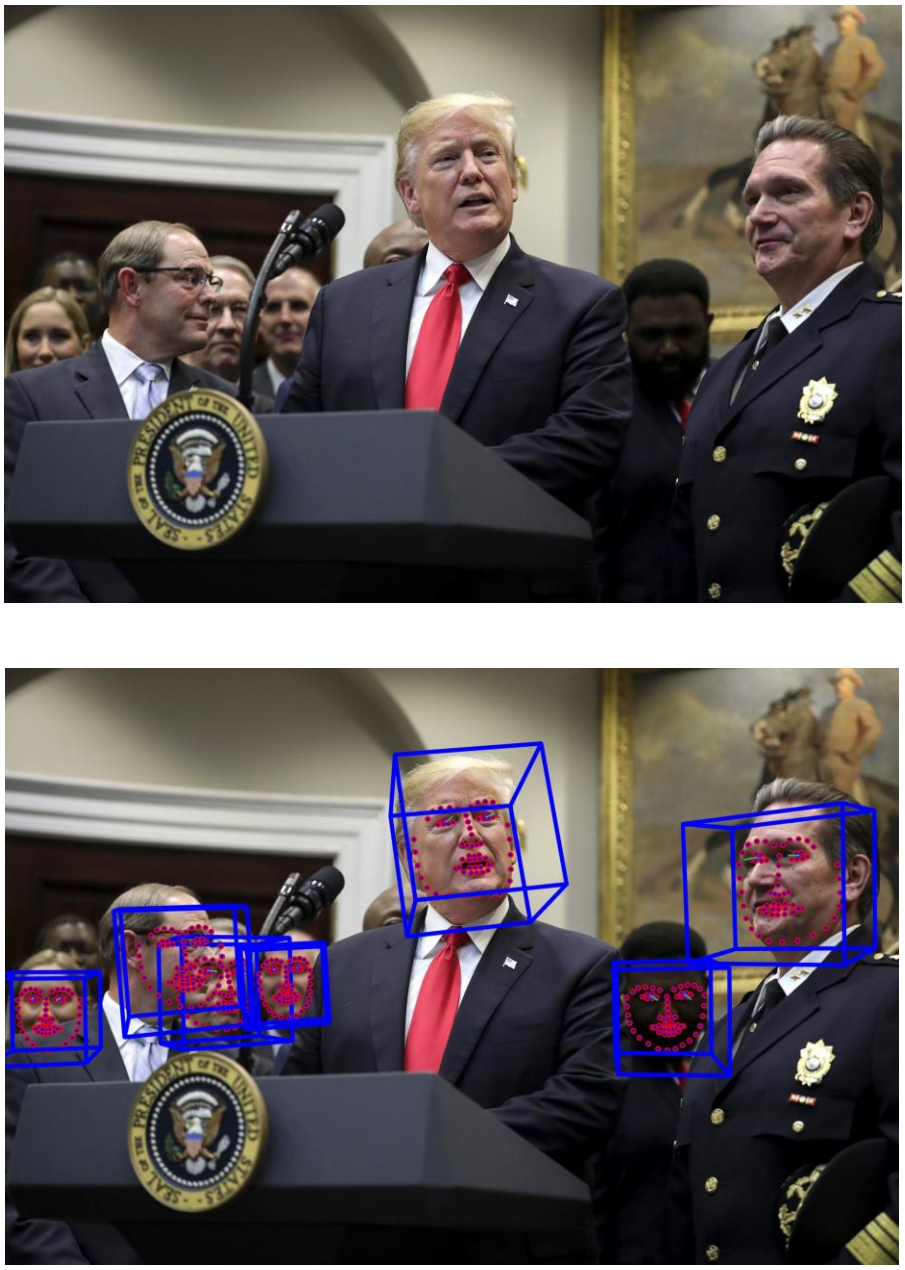


Figure 8: Example Processed Image using OpenFace where subjects are not looking at camera sensor. Gaze detection is shown by bright green vectors. Photo by Oliver Contreras | Washington Post.

The dataset used to train and validate the classifiers consisted of Ang Li's original dataset which has around 200 photos [7] with an additional 100 photos added to increase the overall size for improved training. The dataset is so small due to the very difficult nature of automating image scraping for this application. For a photo to be useful for training, it should feature one or more targets along with one or more bystanders. Photos must be handpicked from search results which is a very time-consuming process. Additionally, each face in each photo must be labeled which increases the time to compile a useful dataset. However, the final dataset consisting of approximately 300 photos was large enough for successful model training.

The selected learning algorithms for classifying faces are gradient boosted decision tree, support vector machine, random forest, and a simple neural network. These algorithms were all featured in Ang Li's work [7] and should provide meaningful comparisons to examine any improvement which occurred. All algorithms were implemented and trained using Sci-kit Learn [14], a robust machine learning library for use with Python. This library simplified the use of each algorithm and the training process, as Python was already being used for extracting and compiling the facial features from images.

B. Classification Results

Each algorithm was trained and evaluated on the dataset 10 times. The evaluation accuracies were averaged together from each iteration. Evaluation and training were carried out using Scikit Learn's library functions. The results from the evaluation are listed in Table 4. The new models trained on the modified feature set are compared against the old models trained on the original feature set [7].

Learning Algorithm	Classification Accuracy (Average of 10 Evaluations)
Gradient Boosted Decision Tree (Original Feature Set)	93.4%
Gradient Boosted Decision Tree	93.8%
Random Forest (Original Feature Set)	92.8%
Random Forest	93.2%
Support Vector Machine (Original Feature Set)	91.7%
Support Vector Machine	92.1%
Multilayer Perceptron (Original Feature Set)	91.4%
Multilayer Perceptron	95.3%

Table 4: The classification accuracy of the evaluated learning algorithms using the new features compared with the accuracy of the algorithms trained on the original feature set.

The multilayer perceptron model (MLP) was the most effective classifier with an average accuracy of 95.3%. This was able to significantly outperform the MLP model trained on the original feature set with an accuracy increase of 3.9%. Additionally, it is clear that each of the other tested algorithms were able to achieve some performance gains using the new feature set over the original. Although the other algorithms weren't able to benefit as much from the

changes as the MLP, the fact that there was an accuracy boost in every evaluated algorithm indicates that the new feature set resulted in an improved model for supervised learning. Ang Li's highest performing algorithm, tested over the original, smaller dataset was a Gradient Boosted Decision Tree (GDBT) which achieved 93.3% overall accuracy [7]. Both the GDBT and MLP algorithms trained on the new dataset and new feature set were able to outperform this.

VI. Conclusion

A. Review of Contributions

The research outlined in the preceding chapters is focused around developing solutions to improve individual privacy with regards to smartphone photos. Being able to analyze and monitor the behavior of any desired application on the Google Play Store is an important first step in identifying malicious activity. The methods outlined in chapter 3 for manually installing an application's APK file and using the ADB shell on a connected Android device is easily repeatable and greatly improves the transparency of 3rd party applications. Strace enables the detection of any media directory access through system call recording. This procedure should enable future researchers to quickly identify applications which should be more closely analyzed through methods such as decompilation.

Chapter 4 introduced approaches towards automated detection of potentially sensitive information in photographs. Extracted HOG features are effective at providing metrics for detecting specific objects such as ID's or faces in selfies which users might not want accessed without their express permission. Even more effective at identifying objects in images, the state of the art neural network models produced by Google such as Inception-v4 and NASNet-A were used to achieve excellent accuracy on the image dataset. These approaches enable an automated system of labeling photos to notify users of sensitive data, so they might be aware of their increased risk in the event of a data leak.

The improvements made to the existing target/bystander feature-based detection model [7] significantly improved the accuracy of the model. This could enable an automated system which can identify individuals whose faces could be obfuscated through some blurring method or otherwise. Being able to automate this process and suggest regions of a photo to blur will

greatly improve the privacy of individuals unintentionally captured in photos in public. However, it will still be ultimately up to the capturer of the image to responsibly obfuscate strangers' faces.

B. Future Work

To expand the feasibility of the photo access detection methods outlined in Chapter 3, a means of analyzing iOS apps will also need to be devised. The App Store does not enable easy access to the application install files like APK mirror sites do for Android devices. Additionally, there is no current utility on iOS for hooking system calls from processes. The closed source nature of the iOS operating system also makes it difficult to develop a utility such as Strace without knowledge of the underlying architecture. In order to devise a similar method which works on iOS, these limitations will need to be bypassed. Otherwise, users will have to continue to rely on App Store screening processes. Beyond expanding to iOS devices, the system call tracing analysis can be improved by adding the ability to detect accesses on specific photos. Therefore, if an application is being used by a user to perform some operation for a single photo, it is possible to detect if it then accesses some other photo that the user did not specify. Fine-grained system call filtering on an individual file level would catch this sort of behavior which the current method does not.

To continue to build upon the classifier presented in Chapter 4, a larger scale project will need to be undertaken to implement such a classifier into a mobile OS kernel. Being able to incorporate such a classifier directly into the kernel would enable users to see requested photo library accesses and block them directly. This could greatly reduce the threat posed by 3rd party applications which are able to access photo libraries on Android at any time once the user grants them external storage read permissions. It is likely the classifier would need to be coupled with a

sort of sensitive photo database cache to ensure the labels on sensitive photos are stored on the device.

The next step in improving the usability of a target/bystander detection system would be to provide an automated method of obfuscating faces. Several blurring methods will need to be analyzed to determine their security and the impact they will have on photos. Additional methods such as replacing faces with some placeholder face might also be interesting to implement. Being able to increase the automation of this process will undoubtedly make it more useful for everyday users. Beyond adding to the system, implementing this detection mechanism in major social media apps perhaps before users upload photos would be a novel feature to help protect the faces of people before they are uploaded to publicly visible profiles.

VII. Publications

[1] Ang Li, David Darling, and Qinghua Li, "PhotoSafer: Content-Based and Context-Aware Private Photo Protection for Smartphones," *in the IEEE Symposium on Privacy-Aware Computing (PAC), 2018*.

VIII. References

[1] Chen, Y. Hu, W. Zhang, X. Xu, Z. (2018, July 30). *Hidden Devil in the Development Life Cycle: Google Play Apps Infected with Windows Executable Files*. Retrieved from https://researchcenter.paloaltonetworks.com/2018/07/unit42-hidden-devil-development-life-cycle-google-play-apps-infected-windows-executable-files/

[2] Reed, T. (2018, September 7). *Mac App Store apps are stealing user data.* Retrieved from https://blog.malwarebytes.com/threat-analysis/2018/09/mac-app-store-apps-are-stealing-user-data/

[3] Richter, F. (2017, August 31). *Smartphones Cause Photography Boom*. Retrieved from https://www.statista.com/chart/10913/number-of-photos-taken-worldwide/

[4] Google. (2018, September 3). *Platform Architecture*. Retrieved from https://developer.android.com/guide/platform/

[5] Chaykovsky, V. *Strace.* (2018). Github repository. https://github.com/strace/strace

[6] Google. (2018, September 11). *Permissions Overview*. Retrieved from https://developer.android.com/guide/topics/permissions/overview

[7] Li, Ang, "Privacy-Preserving Photo Taking and Accessing for Mobile Phones" (2018).*Theses and Dissertations*. 2915. https://scholarworks.uark.edu/etd/2915

[8] Szegedy, C., et al.: Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567v3 [cs.CV], 11 Dec 2015

[9] Dalal, N., Triggs, B. (2005). Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego CA, June 2005. IEEE.

[10] Google. (2018, September 11). *Analyze Your Build with APK Analyzer*. Retrieved from https://developer.android.com/studio/build/apk-analyzer

[11] Zoph. B., et al.: Learning Transferable Architectures for Scalable Image Recognition. arXiv:1707.07012v4 [cs.CV], 11 Apr 2018

[12] OpenFace 2.0: Facial Behavior Analysis Toolkit Tadas Baltrušaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency, IEEE International Conference on Automatic Face and Gesture Recognition, 2018

[13] Rendering of Eyes for Eye-Shape Registration and Gaze Estimation Erroll Wood, Tadas Baltrušaitis, Xucong Zhang, Yusuke Sugano, Peter Robinson, and Andreas Bulling in IEEE International. Conference on Computer Vision (ICCV), 2015

[14] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.