Neural Networks for Improving Wearable Device Security

Zackory Erickson
University of Wisconsin–La Crosse
erickson.zack@uwlax.edu

Jared Compiano
University of North Carolina at Chapel Hill
compiano@live.unc.edu

Richard Shin
University of California, Berkeley
ricshin@berkeley.edu

***Abstract*—Wearable devices with first person cameras, such as Google Glass, are becoming ubiquitous. Additionally, the importance of security will increase as these devices are likely to capture private or sensitive photos. In this paper we introduce a Convolutional Neural Network (CNN) which is capable of detecting various security risks with relatively high accuracy. Our approach is also capable of classifying images within approximately half a second–fast enough for a rapid, mobile environment. We also investigate numerous routes for improving accuracy with our CNN classifier including object segmentation and artificial data generation. Finally we verify our approach using a manually collected dataset which was tested using two different Android devices.**

I. Introduction

Wearable devices such as Google Glass,[1] Samsung Gear,[2] Narrative Clip,[3] and Autographer[4] are becoming increasingly pervasive. Despite this, there still remain an assortment of security issues to be resolved [7]. One major issue with wearable devices, is that these devices are in first person, allowing the camera on the wearable device to relay everything a user sees. This becomes a prominent target for exploitation since these devices can also be used for lifelogging, which allows users to capture and share aspects of their everyday lives. As wearable devices become more prevalent, malware will follow suit, and the importance of security will be critical as cameras on these devices will likely capture private or sensitive information.

Various visual malware applications have already been demonstrated for smartphones [9] and soon these apps will also make their way into the wearable device space. Mike Lady and Kim Paterson, graduate students from California Polytechnic State University, have already created a proof-of-concept malware app for Google Glass called Malnotes [2]. Their application presents itself as a note-taking app for users who install it; however, roughly every 10 seconds Malnotes takes a picture and sends it to a remote server–all without the user's knowledge or consent.

Utilizing similar methods, a malware developer could watch a wearable device owner type passwords into their computer, answer security questions when logging into their bank account, or read off the numbers of a user's credit card when a purchase is made at a retail store.

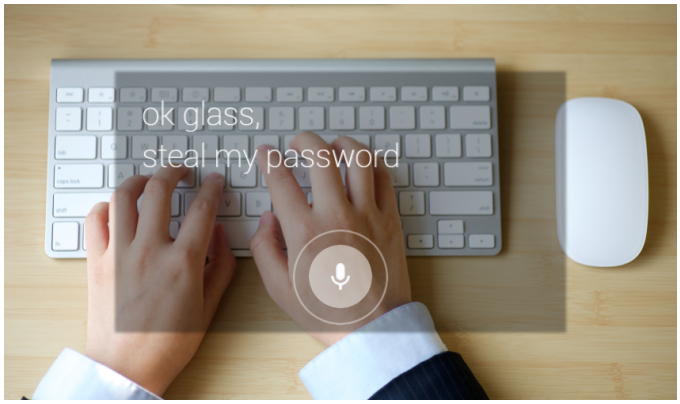

Fig. 1: Security measure requirements become more transparent with the integration of first person cameras in wearable devices, such as Google Glass. (Photo background from SiliconCloud)

In this paper, we propose a machine learning method for identifying and securing images that contain private or personal information. We achieve this using a convolutional neural network (CNN) that is effective on all wearable devices–malware infected or not. Convolutional neural networks have become one of the leading algorithms within the computer vision field, where they have demonstrated high accuracy when used for image classification [5]. Moreover, CNNs can be easily implemented on a cloud or web based system to lessen the computation burden on wearable devices.

A few projects have already begun undertaking the problem of wearable device and lifelogging security, such as Safavi et al. [7], which proposes six different improvements to Google Glass that could greatly enhance privacy. Another project is PlaceAvoider [8], which notifies users when a picture of a private location (such as a bathroom or office space) has been taken. While PlaceAvoider aims to detect the presence of an entire location or room within an image, our project aims to detect the presence of specific objects that pose a privacy threat–no matter where the objects are located. Furthermore,

[1]Google Glass: http://www.google.com/glass/start/
[2]Samsung Gear: http://www.samsung.com/us/mobile/wearable-tech
[3]Narrative: http://getnarrative.com
[4]Autographer: http://autographer.com

PlaceAvoider requires a user to manually train their algorithm by taking pictures of a distinct room, whereas our project does not require any user interaction–similar to many modern day anti-malware applications.

Research Challenges. This work addresses several research challenges in order to make securing private images on a malware infected device possible. Detecting whether an image contains private information remains difficult due to inherent ties with computer vision. Even more, recognizing various small objects such as credit cards provides a larger challenge with object segmentation often being necessary. Current wearable and lifelogging devices have limited computational power, which also restricts the amounts of computation we can perform on an actual device. Lastly, classifying an image should be achievable within a time frame of 500 ms or less. Maintaining this short time frame means image classification can be completed within a real time scenario and will not impede on a user's interactions with their device.

Although securing images with private or sensitive information presents a variety of challenges, the remainder of this paper will present initial steps that can be taken towards reaching this goal. Section II describes our adversarial and system models. Section III delves into our image classification techniques, including our CNN; artificial data; and object segmentation. Section IV describes our implementation on an Android device while Section V provides an evaluation of our entire system with supporting graphs. Lastly we propose further improvements for the overall system in Section VI before concluding within Section VII.

II. Approaches

Our goal is to create a minimal user-interaction wearable device system that can identify specific security threats to the user. We begin this section by defining the assumptions of the problem we aim to mitigate.

A. Adversary Model

We assume that our user has a wearable device with camera and internet functionality. The user has unknowingly installed malicious applications or the applications have been installed by a Trojan, but the operating system of the device is not compromised. Additionally, these malicious applications have the proper camera and network permissions and wish to steal data such as credit card numbers, ATM pin numbers, safe combinations, etc.

B. System Model

In our model, an image is classified into one of multiple sensitive categories or as a nonthreatening miscellaneous object. If the user is looking at a sensitive or private object, our approach would disable all applications' access to the camera until the sensitive object is no longer within the user's view.

The approach we propose contains three main components: a simple privacy policy, image classification method, and a policy enforcement mechanism. We describe each of these components below:

- ***Privacy Policy.*** Our privacy policy aims to target very specific threats. Captured images will be classified into one of many predefined image categories before a wearable device application can receive the image data. For example, computer screens, smartphones, numeric access keypads, keyboards, ATMs, cash registers, safes, credit cards, or toilets may be image categories that are deemed important security flags that the system must identify.

- ***Image Classification.*** The image classifier builds a model from training on various classes of images which is then used to classify new images received from a wearable device. The classifier must be able to properly classify images that contain distortions such as motion blurring or occlusions. Additionally, the classifier will be used in a near real time setting and thus should classify images quickly (under half a second). The image classifier can be also be implemented within a cloud service to lessen the computational burden on a wearable device or a small scale classifier can be setup on a wearable device to reduce network transmission times.

- ***Policy Enforcement.*** We propose two methods for enforcing the privacy policy, both of which would be integrated as part of the device's OS and would sit as a layer above the camera. The first proposed method involves continually (every 5-10 seconds) classifying the images being viewed by the wearable device. If the captured image is classified into one of the risk categories defined by the implemented privacy policy, no application on the device will be allowed access to the camera until the next miscellaneous picture is classified. The benefit of this method is that previously taken images can be used to better predict any security risks for the currently viewed image.[5] The main drawback with continual image capture is an increased demand on a device's battery. Subsequently, our second method involves all requests to take a picture being redirected to our classification layer. From here, our layer accesses the camera directly to take the picture and then classifies the captured image. If the image is classified as safe, then we pass it back up to the requesting application. However, if the image is classified as containing a privacy threat, we drop the image and return a failure response back to the requesting application. This second method is unable to use previous images to improve predictions; yet, it benefits from minimal additional demand being placed on a device's battery.

[5]This concept is not pursued further in this paper. See [8, pg. 5, III-B]

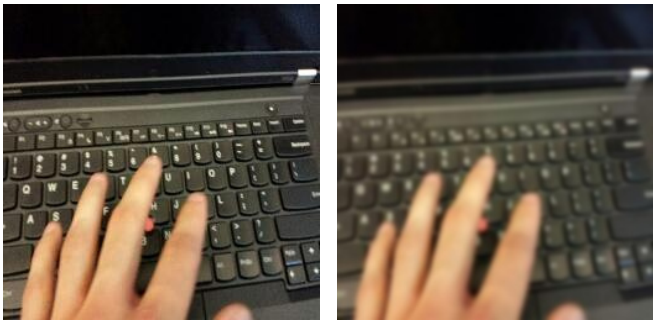

Fig. 2: An original unblurred image next to its artificially blurred image.

Fig. 3: Two credit cards; the left is found in real life; the right is a computer generated card. (Right photo by Discover)

III. Image Classification

A. Convolutional Neural Network

Although there are a variety of algorithms to choose from when classifying images, the most recent ILSVRC 2013 competition [1] has shown that convolutional neural networks are overwhelmingly the current algorithm of choice due to their success with image classification [5]. In addition to having a high accuracy rate, CNNs are also able to classify images quickly while running on GPUs, making CNNs an optimal fit for our work. The importance of fast classification times could not be overemphasized as our system must be able to run unnoticeably in the background so a user does not experience wait time between taking photos. When implementing our CNN we used a community supported, high performance framework called Caffe [4]. Caffe is a C++/CUDA framework that provides fast development with Python wrappers. With CUDA we are able to take advantage of onboard NVIDIA GPUs and provide up to a 10x improvement on classification and training times when compared to using CPUs [4]. Overall our CNN contains 5 convolutional layers, with interspersed max pooling layers, rectified linear units (ReLUs), local response normalization, dropout layers, and a softmax layer to complete the process.

B. Training Set

Although there are many different types of objects containing sensitive information that users may want to prevent malware apps from capturing, we aimed to classify three objects we felt should remain private in almost every situation. These three object classes are ATMs, computer keyboards, and credit cards.

Our starting training dataset contains 8,000 ATMs, 20,000 keyboards, and 4,000 credit cards, all of which have been downloaded from search engines with erroneous images manually filtered out. This dataset also contains 54,000 miscellaneous images collected from the ImageNet database [3], thus our CNN is also capable of classifying objects that do not contain a privacy risk. When combined, we have a dataset totaling 86,000 images with each image scaled to 256x256 pixels. We also created a duplicate dataset with images rescaled to 128x128 pixels. With two datasets we were able to train two separate CNN models with the goal of comparing how the accuracy of the CNN varies depending on the image size (pixel count) of the training data.

C. Artificial Data

When we began training our CNN we had roughly 349,000 images, which is far more than the 86,000 images in our initially collected training set. This was due to our generation of artificial data which included various rotations, blurs, and mirrors (vertically and horizontally) of the original images. Figure 2 shows an example of an artificially blurred image with its original unblurred image.

Although generating artificial data takes a fair amount of time and also increases the time needed to train a CNN model, the resulting CNN performs with higher accuracy when testing against real world data. This is due to the fact that pictures captured from a wearable device often include motion blurring and various lighting effects depending on the environment. Additionally, some of our initial training data does not properly simulate what can be observed in real life settings. Our original credit card training images, for example, are downloaded from image searches with a majority tending to be computer generated. Because of this, our original training set of 4,000 credit cards poorly represents credit cards found in a real setting, as shown in Figure 3. A credit card lying on a table can be seen at any given rotation and thus it is exceptionally important for us to artificially generate realistic credit cards. In section V-A we discuss results of how artificial data generation affects the accuracy of our CNN.

IV. Implementation

To test the viability of our solution in a mobile environment we implemented our CNN within a web service and set up a small scale Android app for testing purposes.

A. Web Service

When setting up our CNN classifier we wanted to make it accessible to smartphones and wearable devices. As a consequence, we implemented our classifier within a Python

TABLE I: How the image size used to train a CNN affects the CNN's accuracy when classifying images into one of four different classes.

Image Sizes	ATM	Keyboard	Credit Card	Miscellaneous
128 x 128	51.1%	62.1%	13.4%	67.4%
256 x 256	76.7%	83.8%	14.9%	73.5%

TABLE II: The CNN's classification accuracy when trained on the original dataset versus the artificial dataset.

	ATM	Keyboard	Credit Card	Miscellaneous
Original dataset	76.7%	83.8%	14.9%	73.5%
Artificial dataset	72.2%	85.1%	34.3%	76.5%

web service using web.py[6]. Our web service was then set up on a local computer that has 132 GB of memory and 2 Tesla C2075 6 GB GPUs.

B. Android Application

We developed an Android application to test our CNN and to demonstrate a proof-of-concept for our privacy approach. Our application performs all of the classification steps, but does not disable access to the camera as described in Section II-B. When a picture is taken, the image is sent via POST request to our web service to be classified. Once the application receives a response from the web service we are able to manually confirm whether the image was classified correctly. The application then sends a final POST request that stores the image, predicted classification, actual classification, and measured timestamps.

V. Results

To test our approach we used two different Android devices in place of a Google Glass. These devices were a Samsung Galaxy S4 and a Samsung Galaxy Young. When compared to Google Glass[7,8] (2 GB RAM and 5 MP camera), the Samsung Galaxy S4[9] has slightly higher quality hardware (2 GB RAM and 13 MP camera), whereas the Samsung Galaxy Young[10] has lower quality hardware (160 MB RAM and 2 MP camera).

A. Datasets

To test our CNN we collected a single unified set of images which were manually taken from various locations using the Samsung Galaxy S4. In total we collected 363 images with the following distributions: 90 ATMs, 74 keyboards, 67 credit cards, and 132 miscellaneous.

[6]web.py: http://webpy.org/
[7]Google Glass RAM: https://plus.google.com/+GoogleGlass/posts/1tSYsPCCsGf
[8]Google Glass Specs: https://support.google.com/glass/answer/3064128
[9]Samsung Galaxy S4 Specs: http://www.gsmarena.com/samsung_i9500_galaxy_s4-5125.php
[10]Samsung Galaxy Young Specs: http://www.samsung.com/latin_en/consumer/mobile-phones/mobile-phones/smartphone/GT-S5360MAKPGU-spec

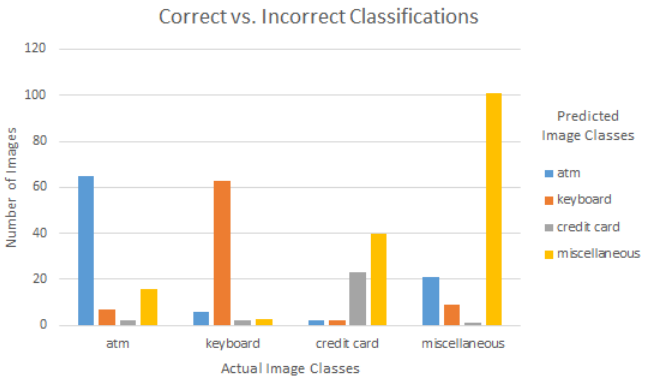

Fig. 4: The number of correctly vs. incorrectly classified images, including what each image was classified as. For example, there were 90 images of ATMs. 65 images were classified as ATMs, 7 as keyboards, 2 as credit cards, and 16 were classified as miscellaneous.

This collected dataset, or test set, will act as the basis for the following graphs and comparisons within Section V. To begin, we first compare how training on different image sizes affects the accuracy of our CNN. Table I shows how the accuracy (percentage of correctly classified images) of a CNN differs when two different models are trained from images sizes of either 256x256 pixels or 128x128 pixels and then tested against our test set. There is a considerable difference when we train the CNN on differing image sizes as our CNN trained on 256x256 pixel images provides a higher accuracy for every image class.

Next, we kept the images scaled to 256x256 and trained two new CNN models. The first CNN was trained using only the original dataset of 86,000 images, whereas the second CNN was trained using the artificially generated dataset of 349,000 images described in Section III-C. Table II then shows how the CNN accuracy varies when trained on either original or artificial data. Although both datasets show similar accuracies, we find that a CNN trained on the artificial dataset is slightly more robust for difficult classifications such as credit cards.

B. Classification Evaluation

Given these results, we selected the artificial dataset of 256x256 pixel images to begin further testing for accuracy. A more detailed graph of what various images were classified as is shown in Figure 4. It is interesting to note in Figure 4 how only a few images of credit cards are actually classified as such. This displays how unrealistic our credit card training set is, which creates a stronger impetus for generating artificial data.

Another useful measurement is how much error we are accumulating. Figure 5 displays a more compressed view for our correct vs. incorrect classifications and highlights our type I and type II errors (false positive and false negative

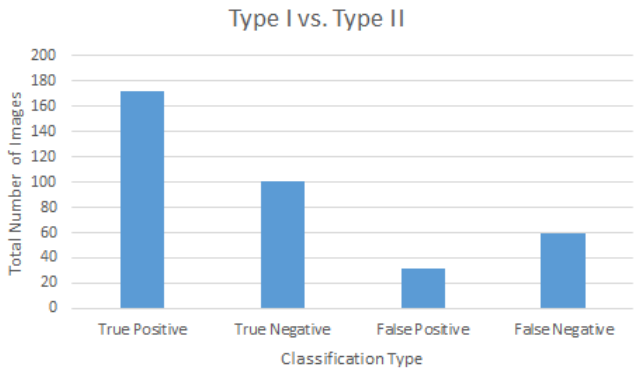

Fig. 5: A more consolidated view of correct vs. incorrect classifications with focus on type I and type II errors.

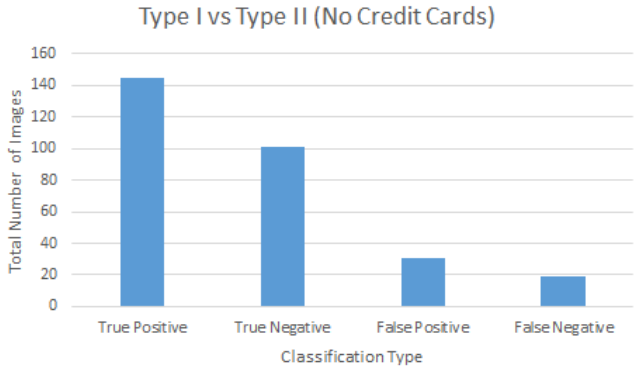

Fig. 6: The number of false negative classifications greatly reduces when credit cards are removed from the test set.

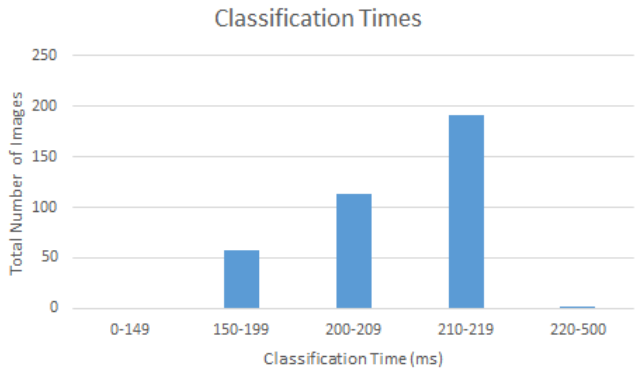

Fig. 7: Time the CNN spent classifying each image within our test set.

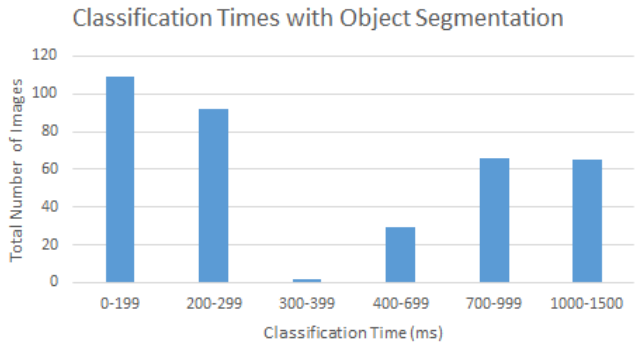

Fig. 8: A slight upsurge in classification times in shown with object segmentation being performed.

respectively). A false positive represents an image that poses no privacy threat, but was classified as containing a privacy threat. Moreover, a false negative is dangerous as it represents an image that *does* contain a privacy threat, such as a keyboard, that was incorrectly classified as miscellaneous (no privacy threat). From Figure 5 it appears that our CNN classifies a large number of false negatives. However, this alarming trend is mitigated when we remove credit cards from our test set–as can be seen in Figure 6. In fact, 68% of all false negatives are caused from misclassifying credit cards as miscellaneous.

C. Time Performance

Though a highly accurate CNN model is critically important, it becomes impractical for use with wearable devices if it requires too much time to classify an image. For this reason we also evaluate the performance of our CNN and overall system in terms of time. We first evaluate how long the CNN takes to classify each image in our test set which is measured from the moment our web service receives an image to the moment the web service returns a response to the wearable device. As Figure 7 shows, almost all of the images in our test set are classified in under 220 ms. This rapid classification time provides the network transmission time a 300 ms window to stay around our half-second goal. Despite this, the classification times displayed in Figure 7 do not account for object segmentation. In Figure 8 we explore how classification time increases when object segmentation is performed on images that are initially classified as miscellaneous. We see an upsurge in classification time due to the CNN reclassifying various small portions of an image (maximum 3 segmented objects).

While the CNN may be fairly quick at classifying a single image, network transmission times often fluctuate. For our project, network transmission is comprised of two events: a wearable device sending an image to the web service and the web service returning a response back to the device. With a strong Wi-Fi connection network times may settle around 100 ms; nevertheless, a poor connection can cause network transmission times to spike much higher (even up to a few seconds). Figure 9 displays this trend of scattered network times measured with the Samsung Galaxy S4. Most of the measured times for network transmission, albeit slightly dispersed, lie under 300 ms, which aligns nicely with our objective of keeping the entire process under half a second.

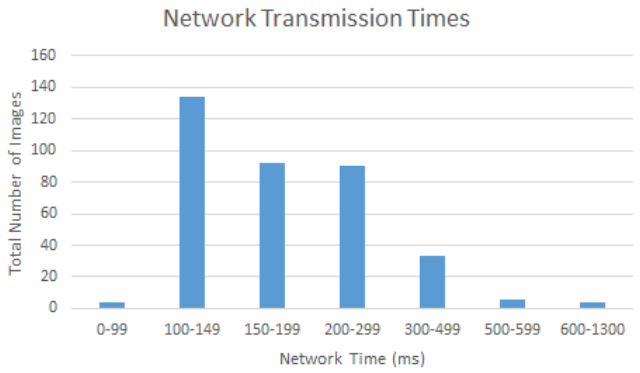

Fig. 9: Total amount of network communication time between the web service and mobile device for all images in the test set.

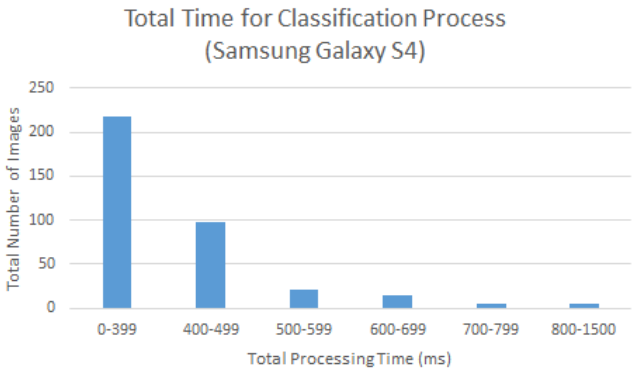

Fig. 10: The time for each image to complete the entire classification process with the Samsung Galaxy S4.

Lastly, we evaluate the total time required for the entire classification process–sending an image through a POST call to the web service, classifying the image without object segmentation, and sending a response back to the device. Figure 10 shows the combined time for each image to go through this entire process using the Samsung Galaxy S4 smartphone. Moreover, we used the Samsung Galaxy Young to determine how lower quality hardware affects timing results. When collecting our test set on the Samsung Galaxy S4 we also collected a similar test set using the Samsung Galaxy Young. Both test sets contained the same number of images from identical locations. Figure 11 displays the combined classification process times for each image when using a Samsung Galaxy Young. Even though the Samsung Galaxy Young device has lower quality hardware compared to the Samsung Galaxy S4 (160 MB vs 2 GB of memory respectively), the total required time for the classification process is sufficiently low in both cases to be used in a real world setting. Consequently, our classification performance is not heavily dependent on a device's hardware quality and thus should be effective on all wearable devices.

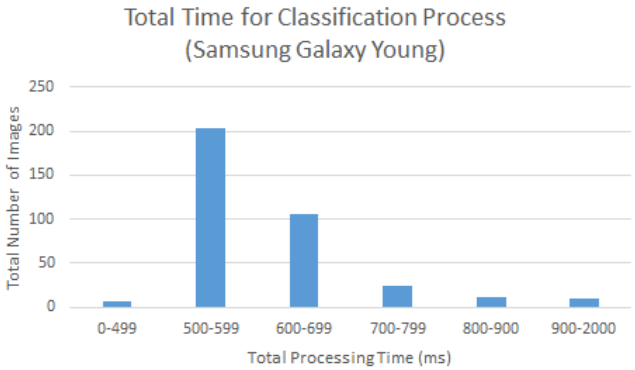

Fig. 11: The time for each image to complete the entire classification process with the Samsung Galaxy Young.

VI. Further Improvements

A. Object Segmentation

Object segmentation is one area that could vastly improve the CNN's accuracy. The difficulty with object segmentation is that more accurate segmentation often requires a learning algorithm, and using a second learning algorithm greatly increases the time required to classify an image. When object segmentation is performed, it may also be beneficial to send the segmented image through a CNN that is trained on smaller images such as 128x128 pixels, thus helping to reduce the amount of pixelation that occurs from stretching a small segmented object to a 256x256 pixel image.

B. Data

While we performed a variety of manipulations on the original images to create an artificial dataset, there are still many more distortions that simulate what can be seen from images in real life. Because of this, increasing the amount of artificial data may also improve the accuracy of the CNN when classifying realistic or distorted images.

Additionally, adding more images to our dataset [6], equalizing the number of images within each non-miscellaneous class, and providing more classes for the CNN to detect would help improve the overall security provided by our system. With a large enough dataset to train on, it would also be advantageous to further pursue how training on various image sizes affect the accuracy of the CNN. Larger image sizes, such as 512x512 pixels, may increase the CNN accuracy without harming classification time.

C. Online Learning

Another feature that could greatly improve the accuracy of the CNN is performing online learning. Since the CNN will be continually receiving and classifying images from wearable devices, the neural network could potentially be further trained on a subset of these images. For example, when the CNN is fairly certain on its classification (such as with a probability of

95% or higher), the classified image could be used to further train the algorithm for a single iteration.

D. Implementation

Our current project is a proof-of-concept; thus, we performed testing using an Android smartphone rather than a Google Glass. A next logical step would be to perform testing on a Google Glass for real scenario testing and statistics. Finally, a last step would be to implement this work as part of the wearable device's OS to handle most (if not all) calls to the camera. Implementing at the OS level would help prevent malware apps from overriding the security features while the OS has not been compromised.

E. Classifier

Our project is efficient when a wearable device maintains a network connection and can send images to an online classifier. Despite this, it would also be beneficial to implement a small, computationally efficient CNN or other image classifier directly on a device. Doing so would provide a user with some amount of security even when the user loses a network connection.

Lastly, while we only pursued image classification using a convolutional neural network, another option is testing various other learning algorithms in place of a CNN. For instance, one available option is training a k-nearest neighbors algorithm on our dataset and comparing its accuracy and time performance to our existing CNN.

VII. Discussion

As wearable devices with first person cameras become more pervasive, we believe the need for security on these devices will heighten as they will likely capture private or sensitive information. Our results show that a CNN is capable of detecting various security risks with reasonably high accuracy even when trained with small and partially unrepresentative training sets. Moreover, we are able classify images promptly in a mobile environment. Considering our approach requires minimal to no user interaction and is not heavily dependent on a device's hardware, it could be seamlessly integrated into current wearable devices.

Our initial results are encouraging; however, numerous improvements need to be made to enhance accuracy before our system is implemented on a wearable device such as Google Glass. We plan to compare the results of our CNN with those of a k-nearest neighbors algorithm and would like to further test our approach using a Google Glass. We would also like to improve object segmentation to better detect small objects like credit cards which provide a challenge for our current classifier.

Acknowledgment

Professor Dawn Song, Aimee Tabor, Diego Lazares, Abhi Gupta, and Team for Research in Ubiquitous Secure Technology.

This work was supported by the National Science Foundation under grant CCF-0424422.

References

[1] J. Deng A. Berg and L. Fei-Fei. Large scale visual recognition challenge 2013. http://image-net.org/challenges/LSVRC/2013/index.

[2] Andy Greenberg. Researchers' google glass spyware sees what you see. http://www.forbes.com/sites/andygreenberg/2014/03/18/researchers-google-glass-spyware-sees-what-you-see/, March 2014.

[3] R. Socher L.-J. Li K. Li J. Deng, W. Dong and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. http://image-net.org/, 2014.

[4] Yangqing Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, 2008.

[7] Seyedmostafa Safavi and Zarina Shukur. Improving google glass security and privacy by changing the physical and software structure. *Life Science Journal*, 11(5), 2014.

[8] Robert Templeman, Mohammed Korayem, David Crandall, and Apu Kapadia. Placeavoider: Steering first-person cameras away from sensitive spaces. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[9] Robert Templeman, Zahid Rahman, David Crandall, and Apu Kapadia. Placeraider: Virtual theft in physical spaces with smartphones. *arXiv preprint arXiv:1209.5982*, 2012.