University of Illinois at Chicago

Privacy Preservation in Shared Images

by

Christopher Tran

PhD Qualifier Examination Paper

Department of Computer Science

April 2018

University of Illinois at Chicago

Abstract

Department of Computer Science

PhD Qualifier Examination Paper

by Christopher Tran

Social media and the access to high quality cameras in smartphones has driven the popularity of image sharing on platforms such as Instagram and Facebook. Often times, users may have different preferences of sharing for different sets of friends. Other times, sensitive information of leaked in the form of SSNs or faces. This paper will explore different approaches to automatically preserving privacy of shared images. One method in managing photo privacy is by leveraging tags created for organizational purposes to create accurate access-control rules. More recent works take advantage of advances in computer vision. In one work, users can specify private regions and image transformations can work to cover the sensitive information. Additionally, advances in deep learning are being used to learn object-privacy information from social media images and to automatically recommend privacy settings for identified sensitive images. We will compare and contrast different methods in detail and explore some potential future work in privacy management utilizing ideas from different methods.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Social networking and social media sites (e.g. Facebook, Flickr, Instagram, etc.) allow people to share information and communicate with others online. At the same time, users do not want to share all of their information with everyone, and privacy has emerged as a serious concern. For example, suppose a user James has his date of birth on his profile page. James may only want certain friends to know his date of birth, while keeping that information private from other connections.

Social networking and social media sites allow users to customize their own privacy policies. On Facebook, there is a "Privacy Settings and Tools" page which allows a user to control what kind of information is shared with friends and the public. Additionally, Facebook has a friends *list* where a user can specify whether a profile item is visible to all friends in a particular list.

Unfortunately, previous studies have shown that users fail to utilize and maintain their privacy policies [1–4], due in part to complex interfaces [4]. For example, on Facebook, users must manually assign each friend to a list (or multiple lists), which can be time consuming depending on the number of friends. Many lists may need to be created depending on the fine-grained privacy preferences a user may have. Also, in some cases, there is a discrepancy between an individual's intention to disclose information, and their actual personal information disclosure [5, 6].

In addition to profile information, privacy disclosures could come from shared images on social media sites. For example, users on Facebook can tag other users in images, which may reveal private information of users in the image [7]. In particular, users often share private images about themselves, their friends, and classmates, without being aware of the consequences such photos can have on their future lives [8].

It is clear that methods need to be developed to help users manage their privacy in social networking sites. This work in particular will focus on three main methods for managing privacy in *shared images*: (1) tag-based access control, (2) privacy preserving image transformations, and (3) privacy assistants for images. Finally, some avenues for future work utilizing ideas from aforementioned methods and online social network research.

Chapter 2

Background

Privacy is a growing concern in online social networks (OSNs), but while users express high levels of concern for their privacy, they do not implement strict controls over their profiles. This may be due in part to users' poor understanding of privacy interfaces and the visibility of their profiles [1–4], as well as different levels of knowledge for managing privacy [9].

Recently, work has been done on automating the process of preserving privacy. Fang and LeFevre [10] propose a privacy wizard for OSNs to help users grant privileges of profile information to friends. The wizard asks users about their preference for friends accessing certain profile information (e.g. date of birth, political affiliation, etc.) and uses that input along with community detection methods for their network of friends to build a classifier to automatically assign privacy labels to unlabeled friends. In mobile location-sharing applications, Ravichandran et al. [11] studied the problem of predicting a user's privacy preference for sharing their location. They attempt to learn a set of default policies to help in customizing users' preferences on sharing their location.

These previous works have mainly focused on specific profile or user items. However, these are insufficient in addressing challenges provided by shared image files. Privacy in online photos may vary substantially due to social contexts and actual image content. Ahern et al. [7] explored common themes in concerns for online and mobile photo sharing. In particular, there were four main themes in their privacy considerations taxonomy: security, identity, social disclosure, and convenience. An overwhelming number of participants in their study show that concern for security of others, as well as identity and social disclosure are important concerns in privacy of images. For example, parents showed overwhelming concerned with security of others (presumably photos of their children). Shared photos may accidentally be shown to unwanted recipients.

Following these observations, automated methods for security image privacy have been proposed such as:

- **Using tags for access control** Yeung et al. [12] attempt to provide a system which allows users to create expressive access control policies on their photos using tags and linked data. Klemperer et al. [13] attempt to use tags used for organizational purposes to create access-control policies. They also explored if tagging with access-control in mind helps in creating better policies.

- **Image Encryption** Images can be perturbed or edited so that sensitive and private information is not leaked to the public. Ilia et al. [14] attempt to prevent privacy leakage in social network images by automatically blurring faces. Other work has attempted to encrypt images using various methods. Ra et al. [15] propose splitting an image into private and public parts so that public images do not expose any information, while receivers of private images can reconstruct the original image. Additionally, He et al. [16] propose a new perturbation technique for partial regions of images to be shared as a public image.

- **Privacy Automation and Recommendation** In more recent works, systems have been developed to *automate* image privacy settings or give *recommendations* of settings to image owners. Zerr et al. [17] use visual features of images to classify images as *public* or *private*. More recently, advances in deep learning have been shown to perform well in image classification tasks [18] and image segmentation [19, 20]. Deep features from CNNs have been used by Tonge and Caragea [21] for classifying if images in Flickr are public or private. Yu et al. [22] have used advances in image segmentation and hierarchical methods for predicting privacy settings for images.

Chapter 3

Tags for Access Control

On many photo sharing platforms (PSPs), users can assign tags to photos for organization, search, communication, and description. Automated and assisted tagging, such as GPS location tagging or face recognition tools make tags more available and decrease user burden on maintaining tags. The availability of photo tags could potentially help users create and maintain fine-grained access control policies for photo sharing. Klemperer et al. perform an 18-participant user study on personal photos to explore feasibility of tag-based access-control rules for sharing photos [13]. In this work, the authors focus on three main research questions:

Q1) Can organizational tags be repurposed as-is to create effective access-control rules?

Q2) Does tagging with access control in mind improve the performance of tag-based access control?

Q3) How do users engage with the concept of tag-based access control?

3.1 Problem Statement

In tagging for access control, a friend[1] should be allowed or denied for viewing a photo based on certain tags assigned by the original uploader of a photo. For a particular user of a PSP, define P to be the set of photos uploaded by the user and F to be the set of friends. The task of access control can be defined as mapping a pair of photo $p \in P$ and friend $f \in F$ to $\{allow, deny\}$, where $allow$ means that friend f is allowed to view

[1] A friend is defined as a social link in a social network such as friendship in Facebook or followers on Flikr. In general it is anyone who has default access to a user's profile.

photo p and *deny* means that friend f is denied access to photo p. Formally:

$$P \times F \rightarrow \{allow, deny\},$$

where the mapping is dependent on the tags for each photo created by the original uploader.

3.2 Methodology

To learn how user tagging can be used in creating access-control rules, a user study was done using various photo tagging activities. Tagging was done on the Picasa desktop photo software and custom web interfaces. Users in the study were asked to complete three main tasks:

T1) Organizational Tagging

T2) Tagging for Access Control

T3) Refinement and Wrap-up

Participants were recruited via advertisements on Craigslist, the university's newspaper, on a university research participant website, and on posters around Pittsburgh. The advertisement was for English-speaking participants who take at least 100 photos per year, and participants were required that they add tags or captions to their photos "often" or "always". Each of the 18 final qualified user was asked to upload 40 photos that they had tagged in the past. The photos provided were stripped of their original tags, for use in the tagging activities.

The three tagging tasks were designed to provide insight into the three main research questions. In the first task, the users focused on creating tags for organizational purposes, using tags that help organize and search their photos. The second and third task focused on organizational tagging, but also with tagging for access control in mind. From these three tasks, insight can be gained about participants' tagging strategies (Q3). Machine generated rules were also created from the tasks to evaluate whether organizational tags are sufficient for access control (Q1) and whether tagging with access control in mind would improve performance (Q2).

3.2.1 Organizational Tagging

The goal of the first task was to evaluate the effectiveness of organizational tags for access-control. Users were asked to tag their photos with the objective of being able to find their photos more easily in the future.

In addition to the tagging procedure, the participants were asked their sharing preferences for each photo used in the study. These access-control preferences were used as the ground-truth for evaluating machine generated rules for access-control. Participants were asked for a list of friends: three people with whom they had a close relationship and four to seven with whom they had less close relationships. For each combination of friend and photo, the participants were asked to select a preference for access [13]:

- **Strong allow**: Allow access; would be very upset if the friend was not able to view the photo.
- **Weak allow**: Allow access; would be upset if the friend were not able to view the photo.
- **Neutral**: No preference between allowing and denying access.
- **Weak deny**: Deny access; would not be very upset if the friend were able to view the photo.
- **Strong deny**: Deny access; would be very upset if the friend were able to view the photo.

3.2.2 Tagging for Access Control

For T2, machine generated best-fit access-control rules were created for each of the participants' friends. To create the machine generated rules, the authors used the c4.5 decision-tree algorithm [23]. The labels for weak and strong preferences were lumped together, creating two *allow* and *deny* labels for the decision tree algorithm. The algorithm was trained on each friend for using the allow and deny preferences. Photos with neutral preferences were ignored during the training.

Each friend was assigned a set of if/then rules in the form of "If *tagged/not tagged* with *tag*, then *allow/deny*". If there were no rules generated for a friend, a default rule of allow or deny was used instead. To familiarize the participant for the idea of tagging with access-control in mind, an interface was used to show examples of rules that were generated from the organizational tagging task. Figure 3.1 displays an example of the interface used to show examples of rules to participants.

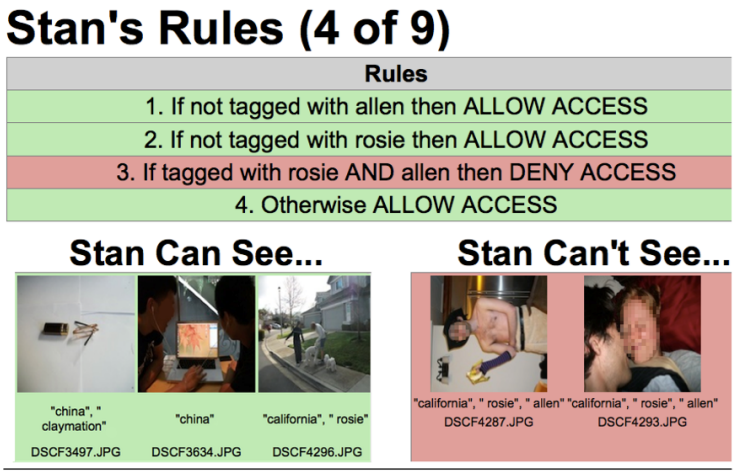

FIGURE 3.1: Example of machine generated rules.

After showing the participants the example machine-generated rules, they were asked to complete the second task. In T2, the participants were asked to add to and/or delete from the tags added in T1. The objective was changed to creating tags for access-control in addition to the original task of finding photos more easily. After the task was completed new machine generated rules were created based on the newly tagged photos. Participants were shown the new rule sets along with the photos that were misclassified. They were asked how upset they were about the misclassifications and how they might change the tags or rule sets.

3.2.3 Refinement and Results

In T3, the participants were instructed to add to and/or delete tags from T2. In this way, they could apply what they learned during the rule review to potentially create better tags for access-control. Additionally, an interview was conduced about participants' photo-tagging and sharing habits. The final set of machine-generated rules were not shown to the participants.

3.2.3.1 Demographics

Table 3.1 shows the demographics for the 18 participants in the study. The subjects were generally young (between 18-32) and mostly technology focused, with 10 of 18 participants identifying as STEM professionals or students. Each participant provided 40 to 48 photos, and expressed 6847 access preferences. Of those preferences, 15.7% were strong allow, 40.8% were weak allow, 17.5% were neutral, 14.9% were weak deny, and 11.0% were strong deny. The distribution of preferences for each participant varied widely. For example, P11 allowed 87.8% of access combinations, while P04 denied 80.0%

Code	Age	Gender	Occupation	Photos/year	Tag software
P01	23	F	STEM professional	1001-5000	Picasa
P02	20	F	engineering student	101-500	iPhoto
P03	27	F	service industry	501-1000	Picasa, Facebook, Tumblr
P04	32	F	STEM professional	1001-5000	Skydrive
P05	24	F	student	1001-5000	iPhoto
P06	23	M	engineering student	501-1000	Picasa
P07	22	M	engineering student	101-500	Picasa
P08	24	M	student	101-500	Picasa
P09	18	M	engineering student	5001+	Picasa
P10	24	M	STEM professional	5001+	Photoshop Album
P11	26	M	STEM professional	1001-5000	Flickr
P12	28	F	art, writing	5001+	Lightroom
P13	23	M	clothing designer	51-100	Twitter, Yfrog, Photobucket
P14	20	M	engineering student	1001-5000	Picasa
P15	19	F	music student	501-1000	Picasa
P16	29	F	anthropology student	1001-5000	iPhoto
P17	25	M	STEM professional	501-1000	Picasa
P18	18	F	art student	1001-5000	iPhoto

TABLE 3.1: Tagging Study Demographics.

of access combinations. In the first task, an average of 2.6 tags per photo were used by the participants. Considering only unique tags created, the average was 1.2 per photo.

3.2.3.2 Results

For non-neutral photo-friend combinations in T1, the generated rules from organizational tags only created about 7.8% conflicts (false allow or false deny). Compared to a control policy of allow all or deny all (based on majority label), the default policy resulted in more than twice as many conflicts with rules generated in T1 (15.8% to 7.8% significant[2], paired Wilcoxon, $\alpha = 0.05$). Figure 3.2 displays the conflict rate for each participant per task. From this figure, we can observe that in most cases, the default policy gave the highest percentage of conflicts, and each subsequent task decreased the conflict rate.

Examining T1 results in more detail, most conflicts (83.5%) the suggested rules disagreed with were the weak preferences. This helps to bolster the claim that effective access-control rules can be made from organizational tags. Additionally, only a small number of photos were responsible for most conflicts for each participant. In T1, only 27% of photos were misclassified by the rules generated from organizational tags.

[2]For the rest of this section, significance tests use Friedman repeated-measures test to establish differences, then paired Wilcoxon tests to separately compare T1 to T2 and T2 to T3. $\alpha = 0.05$ [13]

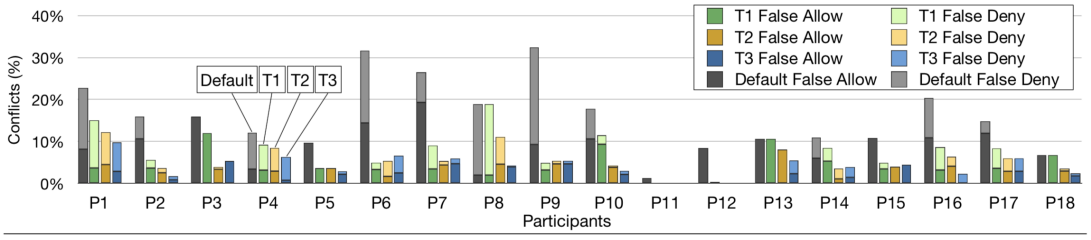

FIGURE 3.2: Tagging Task Results

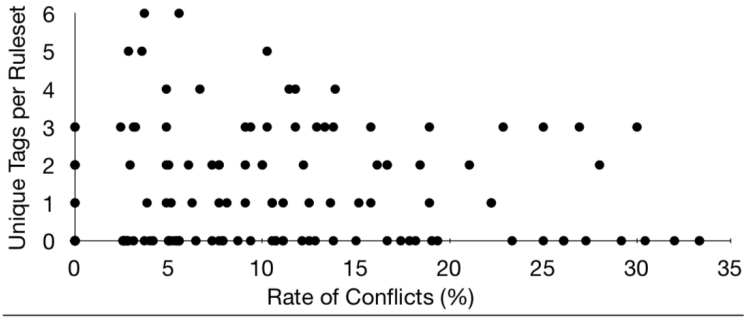

FIGURE 3.3: Percentage conflict vs rule complexity

Rule complexity is also an important factor in generating access-control rules. If a policy has too many rules or clauses, it could be incomprehensible to users. Rule complexity is measured in two ways: counting the number of non-default rules generated for each friend, and counting the number of unique tags in rules. For the first task, more than half of the 168 participant-friend combination resulted in default-only rules. The rest of the 75 non-default cases had an average of 2.8 rules for each friend. Non-default rulesets contained 2.4 unique tags on average. Figure 3.3 shows a plot of the rate of conflict with the rule complexity measured in unique tags per ruleset. The plot shows that many rules are both simple and effective. For example, a simple ruleset would be to allow a spouse to see all photos or denying access of photos to the public. One can also observe that there are many rulesets in the top left (high complexity, low conflict) and the bottom right (low complexity, high conflict), indicating that there is some tradeoff between accuracy and rule complexity.

3.2.4 Reactions to Tag-Based Rules

From the results, organizational tags can be seen to reasonably create rulesets for access control. To get a better understanding of user perception of using machine-generated rules as well as reactions to general tag-based access control policies, users were shown sample rules and surveyed during the user study. In many cases the rules had a positive response. One example is that a participant P05 said the rules for a former teacher made sense because the rules "conceptualized what [she] was thinking." Another participant,

P17, said rules created for a friend with a child his son's age matched "the intuitive rule that [he] made".

However, the rules created from the organizational tags did not all receive positive responses. Some rules were deemed to be too general or too specific, with an example being that denying a friend access to all photos tagged with "gf" was too general, and not all photos needed to be restricted. In some cases, the rules were not restrictive enough, with one participant noting that they were upset that a rule "seemed roughly accurate," but a former teacher was able to see one embarrassing photo.

In addition to the reactions of sample rules, it is important to understand the reaction to tag-based access control itself. Participants were asked a question using a Likert scale of whether the concept of tag-based access control made sense. 13 participants said the concept made complete or some sense, two were neutral, one said it did not make sense, and two said it was dependent on the circumstances. In the case when participants said the concept made sense, they said that using tags would save time or worked well from the examples used. Those that were neutral expressed concerns relating to scalability and the participant that said it did not make sense explained that they would need "a large number of tags to make easier rules."

3.2.5 Tagging for Access Control

Organizational tags have been shown to perform reasonably well in providing access control to photos provided by the participants. However, the results in T2 show that the overall conflict rate improved significantly from 7.8% to 5.2%, with the average improvement of 2.7% across participants. From T2 to T3, the conflict rate improved to 4.2% with average conflict rate dropping by 1.1% across participants. During rule review, participants were asked how upset they were about each conflict (false allow and false deny), where false allows caused them to be significantly more upset ($p < 0.01$). This is an unsurprising result since users may want to keep more sensitive information away from friends rather than not showing some interesting pictures.

Figure 3.4 shows the unique tags and total tag instances deleted in T2 and T3 for each participant. Participants averaged about 33 modifications of T1 tags during T2 with an average modification of 4 unique tags, although these results are skewed since P18 made 108 additions, 5 of which were unique additions, while P11 made no modifications. Most of the changes for T2 were additions (97%). For T3, participants made fewer refinements with an average of 19 additions or deletions.

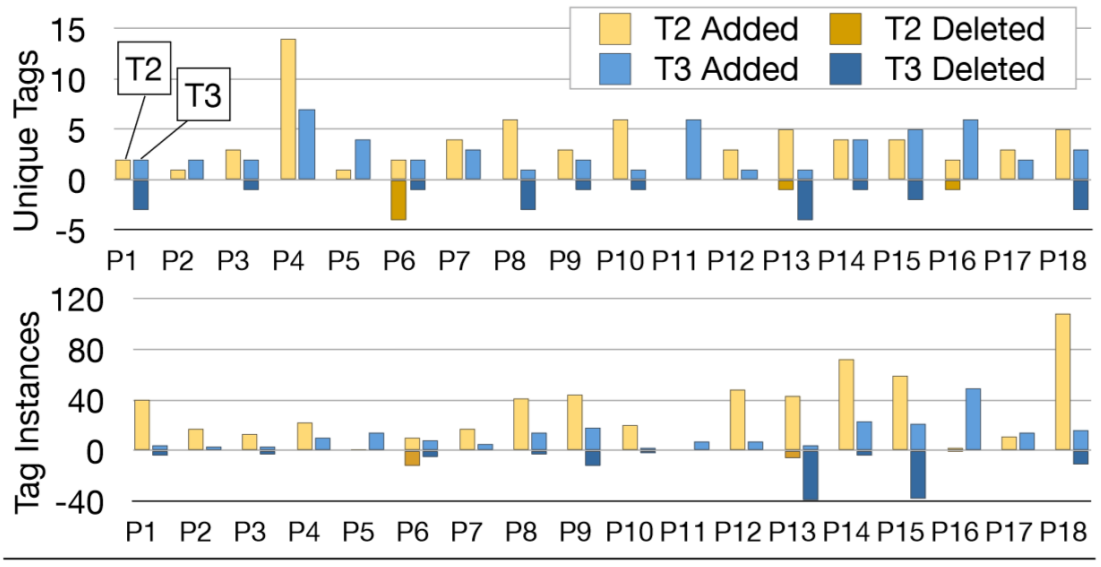

FIGURE 3.4: Modifications in T2 and T3

When creating tags with access control in mind, participants seemed to follow one of three tagging strategies: using content-based tags, using specific tags for access control, or using a hybrid of the previous two strategies. P05, who used a content-based strategy added "kiddos" to photos containing children for parents who might not want their children's photos being shared. In an access control strategy, participants would include tags such as "private", "public", or "for friends."

3.3 Conclusion

Organizational tags have been used in generating rules for access-control systems of shared images. In the study done, Klemperer et al. [13] created machine-generated rules that performed reasonably well for creating access-control policies. In addition, when users are organizing photos by creating tags with access control in mind, rules created from tags are more accurate in allowing or denying a friend.

3.3.1 Critique

There are several limitations for using user created tags in generating access-control policies. First, users are required to tag their photos in order to leverage machine-generated rules for access-control. In order to participate in this study, users were required to respond that they "often" or "always" tag photos in the recruitment survey. This may not always be the case, and those who do not tag or who tag very little may not be able to leverage tagging for access-control. Additionally, if access-control policies

using tags were to be implemented, users would need to be educated on tagging with access-control in mind.

Following the previous limitations, using tags as done in this study is not a scalable method for access-control. Firstly, users would be required to tag all photos that they wish to have access-control policies. Additionally, rules created from tags are done on a friend to friend basis, with user specified access-control preferences for each photo-friend combination. In practice, this is not feasible, since users can have upwards of hundreds of friends and countless photos. Also, in this study, there is no evaluation on new photos and generating new rules for new photos. It is unclear whether rules generated from tags could generalize to new photos.

To address limitations, there are several additions that could be made. For example, automatic tagging can be done to alleviate user input needed for self-tagging images [24]. In this way, tags can still be leveraged for access-control. Unfortunately, this does not solve the problem of needing ground truth labels for each photo-friend combination.

In the following sections, some automated solutions for preserving privacy in images are presented. These methods require little or no input from the image owner, which address these limitations of the previously discussed work.

Chapter 4

Privacy Preserving Transforms

In many cases, only certain sections of a shared image contain sensitive information. For example, SSNs or faces that do not wish to be exposed only take up a portion of a picture, or shared photos of people may accidentally be shown to unwanted recipients (such as friends of a person tagged in the picture). Additionally, due to the nature of most PSPs being a centralized platform where photos are uploaded, saved, and shared on the cloud, adversaries can take advantage of this centralization to steal and leak photos. Automatic encryption techniques could help alleviate these issues.

4.1 Problem Statement

An RGB (Red, Green, and Blue) image is described by pixels, where each pixel is denoted as an 8-bit value in three channels, where each value characterizes one particular color. Transformation of an RGB image to the JPEG [25] format usually takes four steps:

1) **Transform an RGB image to its YUV representation** The YUV representation maps an image to three color layers: Y, U, and V.

2) **DCT transformation** Each YUV layer is divided into 8x8 pixel blocks in which a Discrete-Cosine Transform (DCT) is applied. DCT expresses a finite set of data points as a sum of cosine functions at different frequencies that is invertible. In the DCT matrix, the first component is called the Direct Current (DC) component and the rest are Alternating Current (AC) components.

3) **Quantization** DCT coefficients need to be rounded to the nearest integers. A quantization table is used to quantize transformed integers.

4) **Entropy coding** DC and AC coefficients are separately encoded using differential value encoding and run-length encoding. The encoded outputs are compressed using Huffman coding algorithm and all visual information is included in the DCT coefficients, quantization tables, and Huffman coding table.

The goal is to preserve the privacy of an image by protecting sensitive information, (e.g. faces, SSNs, etc.) while taking advantage of storage resources and sharing capabilities of PSPs. In other words, protecting the sensitive information of images PSP-side. Additionally, many PSPs do not allow fully encrypted images to be uploaded directly, since images need to support PSP image transformations (e.g. cropping, compression, etc.), so encryption techniques must produce images that can be uploaded to PSPs.

4.2 P3 for Photo Privacy

One method for preserving privacy of images is called P3 [15], and is the most similar to the main method to be discussed in the next section. P3 leverages the sparsity of JPEG compression in the DCT domain (i.e. a few coefficients provide most of the information to reconstruct the pixels) and the high quality of images from modern cameras. P3 addresses the problem of privacy-preserving photo sharing by splitting an image into a *public* part and a *private* part. To do this, P3 encodes the most significant bits of the significant DCT coefficients in the private part, while leaving everything else in the public part. Specifically, in the public image, all DC components are removed and AC coefficients whose absolute values are greater than a pre-defined threshold are also removed. In the private part, DC components and the rest of the AC components are stored. In this way, images produced from P3 are still regular JPEG images, and support PSP side image transformation and operations. On the receiver side, the private image must be decrypted and reconstructed using both the private image and public image.

However, P3 has some limitations. First, P3 does not differentiate between sensitive regions and instead does a perturbation on the image level. Second, although image transformations on PSPs can be done on the separate images produced by P3, it is not specifically designed to support image processing techniques. Because of this limitation, reconstruction of an image in P3 becomes more difficult in the presence of transformations. Finally, P3 loses many fine details in images when the image is recovered on receiver side, especially in the presence of image processing. Figure 4.1b shows the recovery of an image using P3 [16]. We can see the image appears to be blurry, which shows P3 loses information when recovering the original image.

(a) Original image

(b) Recovered image of (a) in P3

(c) Recovered image of (a) in PuPPIeS

FIGURE 4.1: Comparison of image recovery methods

To address the limitations of P3, He et al. propose PuPPIeS [16], a more robust image privacy sharing scheme. Figure 4.1c shows that PuPPIeS can recover a better quality image compared to P3.

4.3 PuPPIeS

In this work He et al. propose a privacy preserving image sharing method that addresses three main challenges for sharing images on PSPs.

- **C1** Perturbed images should maximize usability while also protecting privacy. For example, an image with the statue of liberty can still be useful when comparing to other photos of the statue of liberty even if faces and bodies are protected.
- **C2** Systems should be able to support image transforms on PSPs. The ability to support popular image transforms that occur on PSPs such as cropping, scaling, and compression are important for sharing as well as saving storage on the cloud.
- **C3** Privacy preserving perturbations should reliably block out sensitive information, but should be flexible so that users can personalize ROIs and customize the desirable privacy levels.

To do this, regions of interests (ROIs) are protected using image perturbation techniques with security keys, and the image is stored on the PSP with the secured private regions. Security keys are distributed *individually* by the sender through secure channels, so that only individuals can decrypt specific perturbed regions. In general, the security key does not decrypt the entire region, and only decrypts ROIs the sender wishes to give to specific receivers.

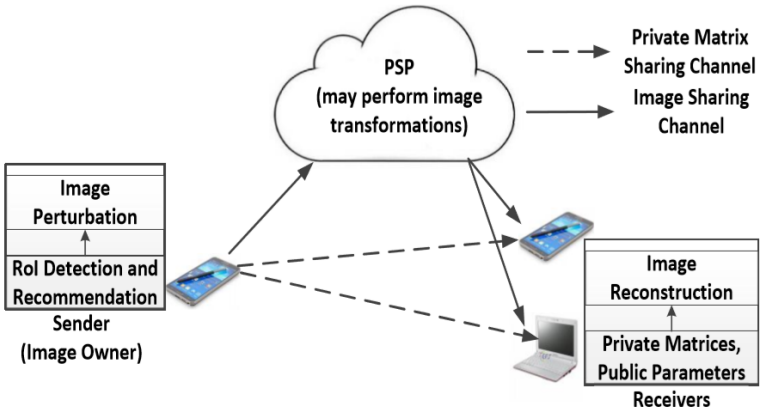

FIGURE 4.2: System architecture of PuPPIeS

4.3.1 Methodology

Figure 4.2 gives an overview of PuPPIeS. In this diagram, there is the image sender, the PSP, and one or more image receivers. The workflow of PuPPIeS is described via each component.

Sender There are two main operations at the sender-side: (1) ROI detection and recommendation, and (2) image perturbation of ROIs. To automatically detect ROIs in images, the detection module is based on techniques using Face Detection [26], Optical Character Recognition (OCR) [27], and General Object Detection [28]. The ROI detection module automatically detects all human faces, text blocks, and top-N general objects in the image. ROIs are split into disjoint regions after merging regions from the three object detection results. In this way, image owners can secure disjoint regions with different privacy matrices, or combine disjoint regions to form a new ROI. Image owners can accept, deny, or modify the automatically generated ROIs. Additionally, image owners can manually mark other ROIs that are not automatically recognized. This offers a more personalized privacy setting for sharing photos.

Receiver The receiver will retrieve *public data* about the downloaded image and recovers the perturbed ROIs using *private data* from the sender. *Public data* includes mR, K, ROI information, ZInd in Algorithms 4.2 and 4.3, ID of private matrix P_i, and the transformation type at PSP side. The receiver will first do ROI decomposition from retrieving the size and locations of ROIs, and recover the original image using the shared private matrix P_i. The private matrix is shared using secure channels between the sender and receiver. There are two scenarios on the receiving side:

- **Scenario 1: no image transformation at PSP-side.** In this case, once the receiver gets the private matrix P' and the public data, the original image can

be recovered using DCT reconstruction. The DCT reconstruction coefficients are given by the following equation:

$$\bar{b_i} = \big((e_i - \bar{p_i} + 1024) \bmod 2048 \big) - 1024. \tag{4.1}$$

In Eq. (4.1), e_i is the i-th component in a 64-dimensional vector e corresponding to a block in ROI in the perturbed image, $\bar{p_i}$ is the i-th component in the private matrix P' and $\bar{b_i}$ is the reconstructed i-th entry in the coefficient vector.

- **Scenario 2: image transformation at PSP-side.** In this scenario, the PSP performs an image transformation on the perturbed image. The difference in this case is that the recover process must be done on the perturbed and transformed image, rather than just a perturbed image. A different private matrix P'', called a "shadow ROI matrix" must be used, which is generated by the private matrix P' and public data including the transformation type used by the PSP. The recovery works in the same way as in scenario 1, but with the new private matrix P''.

In general, a receiver may not recover the entire image, since each perturbation may have different privacy matrices, while the receiver may only get a subset of all matrices according to sender preferences.

PSP The assumption is that the PSP can perform any type of image transformation (e.g. cropping, rotation, etc.) and that PSPs can store perturbed images as well as the image's public data.

Communications between sender and receiver The sender only shares the private matrix P with the receiver via secure channels.

4.3.2 Details

Image encryption is done at sender-side using DCT coefficient perturbation. Let R be a specific ROI detected by the ROI module. In this work, the assumption is that R is divided into many fixed size blocks of 8x8, which also corresponds to a privacy matrix P of size 8x8 vectorized into a 64-dimensional vector P'. Four methods are presented for perturbation.

PuPPIeS-N: equally treating all coefficients

For a given block B_k ($1 \leq k \leq K$, K is the number of blocks) in region R, its DCT coefficient vector is a 64-dimensional vector. Let $B^k = \{ b_i^k \mid 0 \leq i \leq 63 \}$ be the DCT

coefficient vector for the k-th block in region R. The private vector P' is defined as $P' = \{ p'_i \mid 0 \leq i \leq 63 \}$. The encryption is defined as E^k, where $E^k = \{ e^k_i \mid 0 \leq i \leq 63 \}$, such that:

$$e^k_i = b^k_i + p'_i. \tag{4.2}$$

Unfortunately, this approach has a major drawback. All DC coefficients are secured by the same single number p'_0, where $-1024 \leq p'_0 \leq 1023$ (as explained in JPEG), so an adversary could recover the DC components with brute force.

PuPPIeS-B: discriminantly treating DC and AC

A more robust method treats DC and AC components differently, so that DC coefficients are more secured by different random numbers. Recall the DC component is the first entry so that we have:

$$e^k_0 = b^k_0 + p'_\ell,\ \ell = (k \bmod 64), \tag{4.3}$$

$$e^k_i = b^k_i + p'_i,\ \text{for } 1 \leq i \leq 63, \tag{4.4}$$

where Eq. (4.3) and (4.4) are the perturbed DC and AC components respectively. In this way, the entire private vector P' is used in perturbing the DC coefficients.

One drawback of the *PuPPIeS-B* method is that it introduces significant overhead, since it increases image size by about 10 times in the worst case scenario (perturbing the entire image). To reduce the perturbed image size, another method *PuPPIeS-C* is introduced.

PuPPIeS-C: reducing perturbed image size

The reason for the increase in image size is the default Huffman coding tables are optimized for each original image according to the DCT coefficients. Since PuPPIeS adds random numbers to the coefficients, the Huffman coding tables become less optimized, since the frequency distribution changes. To address this issue, Huffman coding tables can be reconstructed based on the new distribution of DCT coefficients after perturbation.

Let Q' be a 64-dimensional vector called a *private range matrix* which is used to control the range of each entry in P'. The intuition comes from DC coefficients being the most informative while AC components are less informative. Thus, in the implementation of *PuPPIeS-C*, P is used to perturb the DC component while P and Q are used to perturb the AC components as shown in Algorithm 4.1. This is because visual information of most natural images is concentrated in lower frequency coefficients (i.e. DC coefficients)

[25, 29]. Algorithm 4.2 is used to generate the private range vector Q'. Two parameters mR and K are used, where mR is the minimum range of entries in P, and K is the number of coefficients the algorithm perturbs.

Algorithm 4.1 PuPPIeS-C

Input: a ROI region R in the original image I, the vectorized private matrix P', and the vectorized privater range matrix Q'

Output: An image I' perturbed on a given ROI using P'

1: $k \leftarrow 0$; {B^k denotes the k-th block in R}
2: **for** each DCT coefficient in k-th block B^k in R **do**
3: $\quad B_0^k \leftarrow B_0^k + P'_j; j \leftarrow (k \bmod 64)$;
4: $\quad k \leftarrow k + 1$;
5: $\quad$ **for** $i \leftarrow 1$ **to** 63 **do**
6: $\quad\quad B_i^k \leftarrow B_i^k + (P'_i \bmod Q'_i)$
7: write all coefficient blocks to a new image I';
8: **return** I';

Algorithm 4.2 Privacy range matrix Q' generation

Input: (i) mR: minimum range of entries in P; (ii) K: number of coefficients to be perturbed.

Output: Vectorized privacy range matrix Q'

1: $r \leftarrow 2048$;
2: **for** $i \leftarrow 0$ **to** 63 **do**
3: $\quad Q'_i \leftarrow r$;
4: $\quad$ **if** $r > mR$ **then**
5: $\quad\quad r \leftarrow r/2$;
6: $\quad$ **if** $i \geq K$ **then**
7: $\quad\quad r \leftarrow 1$;
8: **return** Q';

PuPPIeS-Z: shifting perturbed image size to public parameters

In *PuPPIeS-B* and *PuPPIeS-C*, parameters R, mR, and K are public and stored together with the perturbed image. Parameters are only needed if they are interested in the perturbed regions, so to reduce the size of perturbed images, some information can be shifted to public parameters. In general, the more consecutive zeros in AC coefficients, higher encoding efficiency and compression ratio can be achieved. Instead of perturbing consecutive zero blocks, skipping them is a possibility, but perturbation may change a value to zero which is not distinguishable from original zeros. An additional check for AC entries becoming zero are added, and stored in a new zero coefficient set called $ZInd$, as referred to in Algorithm 4.3. Therefore the new public parameters are R, mR, K, and $ZInd$.

To understand the extra overhead for perturbation, the methods discussed were applied to *The PASCAL Visual Object Classes Challenge 2007* [30] dataset. Table 4.1 shows the

Algorithm 4.3 PuPPIeS-Z

Input: a ROI region R in the original image I, the vectorized private matrix P', and the vectorized privater range matrix Q'

Output: An image I' perturbed on a given ROI using P', and the positions of new zeros (ZInd)

$k \leftarrow 0$; {B^k denotes the k-th block in R}
$ZInd \leftarrow \emptyset$
for each DCT coefficient in k-th block B^k in R **do**
    $B_0^k \leftarrow B_0^k + P'_j; j \leftarrow (k \bmod 64)$;
    $k \leftarrow k + 1$;
    for $i \leftarrow 1$ **to** 63 **do**
        if $B_i^k != 0$ **then**
            $B_i^k \leftarrow B_i^k + (P'_i \bmod Q'_i)$;
        if $B_i^k = 0$ **then**
            add (k, i) to $ZInd$;
write all coefficient blocks to a new image I';
return I';

normalized file size after perturbation. From the table, it can be seen that *PuPPIeS-B* increases image size by about 10 times on average, while *PuPPIeS-C* and *PuPPIeS-Z* do not introduce too much overhead. Particularly, *PuPPIeS-Z* reduces the overhead overall compared to *PuPPIeS-C*, with a smaller standard deviation and smaller max file size.

Support Image Transformations Uploaded images to PSPs can be transformed PSP-side such as scaling, cropping, compresstion, etc. The algorithms discussed can support popular images (i.e. perturbed images can still be recovered under transformations).

Linear Transformations Linear transformations generally operate on either the YUV domain (scaling, cropping, rotation) or the frequency domain (filtering, overlapping). Consider a YUV domain transformation, which is formulated as a linear operation of pixel blocks in the YUV space. Let b be the YUV domain representation of a DCT coefficient block B. Then define a linear transformation f where $f(B) = b$ and $f(P)$ is the YUV representation of the privacy matrix P. Then $f(B + P) = f(B) + f(P) = b'$, where b' is the YUV representation of the perturbed block. Therefore the "shadow ROI matrix" which is derived from the K block of $f(P)$ for each privacy matrix P. DCT transformations are done in the same manner.

Non-Linear Compression *PuPPIeS* can also support compression on PSPs. For non-linear transformations, the quantization tables of the original image and the perturbed image (denoted by T and T' respectively) are needed. The receiver can know each

Scheme	mean	median	std	min	max
PuPPIeS-Base	10.45	9.69	3.88	5.01	85.8
PuPPIeS-Compression	1.46	1.41	0.230	1.15	6.26
PuPPIeS-Zero	1.23	1.22	0.064	1.10	1.80

TABLE 4.1: Perturbation size increase

perturbed block using T. In the same manner, let B' be a block in the perturbed image. To get the original DCT coefficient, it is sufficient to subtract the privacy matrix P from the original block B. Then, the receiver calculates the compressed DCT coefficient block of B using T'.

4.4 Conclusion

In this paper, He et al. [16] develop an algorithm for preserving the privacy of images called PuPPIeS, which supports transforms in PSPs. PuPPIeS partially encrypts regions of interests (ROIs) that may contain sensitive information such as SSNs, faces, and other information. Images are perturbed by modifying DCT coefficients using a private matrix that is only accessible through secure channels from the receiver. Sensitive regions are recommended to the image owners who can allow or deny the regions, while also adding their own.

4.4.1 Critique

Although PuPPIeS is very effective at perturbing images to minimize privacy leakage, the system relies on low level visual features using Face Recognition [26], Optical Character Recognition [27], and Object Detection [28]. There are two problems with this method.

First, users are required to inspect the ROIs detected using the system, and allow or deny the encryption for these regions. Additionally, regions not detected may have to be manually added by end-users. This is advantageous in the fact that perturbations can be personalized by users, and protection of privacy is immediately apparent in the final public image. This is advantageous compared to tagging with access control in mind [13], as discussed previously. However, the burden of protecting privacy is still on the user, as the ROIs may not exactly match user preference. Also, users may not be aware of some private information in their own images [8].

Second, low-level features are not, by design, able to actually define privacy in social images. The PuPPIeS system assumes that privacy sensitive objects in images can be captured by the low level features it uses for ROI detection. The main challenge

in preserving photo privacy is generating *discriminative* features in detecting privacy. In this work and in other works, low-level image features are not very informative in describing the private information in images.

In the next section, an automated method for detecting privacy-sensitive objects is explored, that does not make strict assumptions on image features for preserving privacy.

Chapter 5

Privacy Assistants for Images

The previous sections discussed methods for preserving privacy of images where much of the responsibility was on the user. In tag-based access controls, users must set organizational tags that not only help them to find their photos later on, but if the privacy of their images is a concern, they have to design tags for access-control in mind. For PuPPIeS, the module can automatically detect region of interests (ROIs) to recommend perturbations in images to the users. In addition, users can also select their own regions for perturbation so that they can share images without risk of privacy leaks. However, this method does not do much to alleviate the responsibility and initiative needed for a user to protect their privacy. Users still must manually accept or deny recommendations for perturbations, or even set their own regions. Additionally, ROIs are detected using a combination Face Detection [26], Optical Character Recognition (OCR) [27], and Object Detection [28], which do not explicitly find sensitive information in images. For example, it is possible a combination of OCR and object detection may detect a ROI as an unimportant sign (such as a McDonald's sign), for which the user would have to deny the recommendation. This is a cumbersome task. In the following sections, the discussion will be on automatically or semi-automatically assisting users with managing privacy in shared images.

5.1 Problem Statement

In this section, the goal is to automatically assign privacy settings for images. Formally, given a set of privacy settings P (e.g. $P = \{public, private\}$) and an image i in a set of images I, a system should be able to assign a privacy setting for each image i. Specifically:

$$f : I \rightarrow P, \tag{5.1}$$

where f is a mapping from an image to a privacy setting.

5.2 Privacy-Aware Image Classification

Previous works have looked at privacy-aware image classification. Zerr et al. [17] crawled images from Flickr and performed user studies to hand-label images as *public* or *private*. They considered computer vision tools such as face detection [31], as well as image features such as color histograms, edge-direction coherence vector [32], and SIFT features [33]. Additionally, they considered **stemmed** tags extracted from the labeled dataset of private and public photos. Using those features, they use SVM for classifying if an image was private or public.

In more recent work, advances in computer vision and deep learning using convolutional neural networks (CNNs) have been used in predicting privacy settings. Tonge and Caragea [21] have used deep features in predicting whether an image on Flickr is public or private. In this work, they apply deep CNNs to their Flickr dataset, and compare performance against SIFT and GIST [33, 34] features in predicting privacy settings, and show that deep visual features outperform standard features in classification. Additionally, they leverage their model to perform automatic tagging of images, and show that using automatic tags combined with user tags increases prediction accuracy. Spyromitros-Xioufis et al. [35] have performed privacy classification based on visual features. They compare VLAD [36] features with the VGG-16 model [37] learned with the ImageNet ILSRVC 2014 dataset [38]. In addition, they learn semantic features using the VGG-16 model, and formulate a problem of privacy-related topic modeling using LDA [39]. In this case, each image is created as a document with $n = 10$ semantic feature concepts with a private image corpus.

5.3 iPrivacy

Previous work has focused primarily on image classification, where the task is to predict a privacy setting given an image, and has shown that deep features can perform significantly better compared to handcrafted visual features such as color historgrams, GIST, and SIFT. In addition, previous work has attempted to feed images or image features directly into prediction. Yu et al. [22] propose using finer level of detail in image classification by detecting privacy-sensitive objects from images.

The idea is to detect privacy-sensitive objects from images being shared, and identify privacy settings based on objects in images. The challenge in this task is determining what object classes should be detected from images, and how to use the object detection results to recommend or automatically set privacy settings for shared images. Additionally, predicting and setting privacy settings is a time limited task, since users want to get their recommendations quickly. If a standard approach to detecting objects is used, computational cost will grow linearly with the number of object classes for detection and recognition. Hierarchical approaches [40, 41] could be used, but suffer from the inter-level error propagation problem (i.e. mistakes made at parent nodes propagate to child nodes).

5.3.1 Overview

To address the previously mentioned limitations, the iPrivacy system takes four main steps:

1) Deep CNNs are leveraged to perform semantic image segmentation and to identify a large number of object classes from social images. An algorithm to achieve object-privacy alignment is developed to learn the privacy relatedness of objects and identify a set of privacy-sensitive object classes.

2) A coarse-to-fine representation of the large number of privacy-sensitive object classes are learned via a visual tree. In this way, inter-related learning tasks are given automatically for identifying privacy-sensitive object classes.

3) A hierarchical deep multi-task learning (HD-MTL) algorithm is developed to learn deep CNNs and a discriminative tree classifier to jointly learn the visual tree.

4) A soft prediction method is used to explore multiple paths of the learned classifier. This achieves a better and more stable prediction on privacy-sensitive object classes.

In the following sections, each step is described in detail.

5.3.2 Object-Privacy Alginment

In the first step of the iPrivacy system, large-scale social images and their privacy settings are leveraged to identify a large-set of privacy-sensitive object classes by using semantic image segmentation to associate with given privacy settings. In this work, the privacy

settings are partitioned into three groups: (1) public, (2) private, and (3) shared with friends or family.

Deep CNNs have shown strong ability on many image tasks including semantic image segmentation [19, 20, 42]. For semantic image segmentation, a fully convolutional neural network, along with a conditional random field (CRF) is learned to integrate neighboring pixels. Results from the semantic object regions are extracted from images and used for automatic object-privacy alignment.

Each social image is partitioned into a set of semantic object regions, and each semantic object region may correspond to one certain type of class. An image can be described by all its object classes. A set of 1000 object classes is used, and a social image is then described as a 1000-dimensional sparse representation of object classes. The semantic similarity between two images is defined as:

$$\kappa_I(X_i, X_j) = \sum_{l=1}^{1000} \delta(X_i^l, X_j^l), \tag{5.2}$$

where X_i and X_j are the sparse representation of images i and j, and $\delta(X_i^l, X_j^l)$ is defined as:

$$\delta(X_i^l, X_j^l) = \begin{cases} 1, & \text{if } X_i^l = X_j^l = 1; \\ 0, & \text{otherwise;} \end{cases} \tag{5.3}$$

where $X_i^l = X_j^l = 1$ indicates that two images i and j contain the same object class l. In this way, social images are grouped into a large number of clusters according to their similarity κ_I.

Each cluster contains semantically-similar images based on the semantic object segmentation. The privacy settings for each cluster are merged to generate a list of common privacy settings, which are ranked according to their occurrence frequencies. Top ranked frequencies define the privacy settings for similar images. Given a list of privacy settings P for a given cluster, relevance scores are calculated for each privacy setting $t \in P$ and object class $C_i \in C$ pair, where $C = \{C_1, \ldots, C_i, \ldots, C_n\}$ is the given cluster. The relevance score $\gamma(C_i, t)$ is defined as:

$$\gamma(C_i, t) = \frac{|\Psi(C_i, t)|}{|\Psi(C, P)|}, \tag{5.4}$$

where $\Psi(C, P)$ is the full set of semantically-similar images that are related with a set of privacy settings P in cluster C, $\Psi(C_i, t) \subseteq \Psi(C, P)$ and is the set of semantically-similar images which contain the privacy-sensitive object class C_i assigned with privacy setting t. In other words, $\gamma(C_i, t)$ is the proportion of images with object class C_i and privacy setting t, in the entire cluster C for all privacy settings in the list of P. After getting the initial object-privacy relevance score, the score is refined using co-occurrences of objects within clusters. The intuition is that privacy settings are needed at an image level, and similar object classes should have similar privacy settings.

Given a set of object classes, an object co-occurrence network is constructed for refinement of relevance scores iteratively. For two object classes C_i and C_j, their co-occurrence $\phi(C_i, C_j)$ is defined as:

$$\phi(C_i, C_j) = \rho(C_i, C_j) \log \frac{\rho(C_i, C_j)}{\rho(C_i) + \rho(C_j)}, \tag{5.5}$$

where $\rho(C_i, C_j)$ is the co-occurrence probability for two object classes C_i and C_j, $\rho(C_i)$ and $\rho(C_j)$ are individual occurrence probabilities for object class C_i and C_j respectively as defined by:

$$\rho(C_i, C_j) = \frac{N(C_i, C_j)}{N},\ \rho(C_i) = \frac{N(C_i)}{N},\ \rho(C_j) = \frac{N(C_j)}{N}, \tag{5.6}$$

where $N(C_i, C_j)$ is the number of images which contain object classes C_i and C_j simultaneously, $N(C_i)$ and $N(C_j)$ are the number of images which contain object class C_i and C_j respectively, and N is the total number of images. Object classes that have a high co-occurrence probability form edges in an object co-occurrence network. The intuition is that objects that appear together more frequently are strongly related and may share similar privacy settings.

To refine the object-privacy relevance scores, a random walk process is performed on the object co-occurrence network. For a cluster with m object classes, define $r_k(C_i, t)$ to be the object-privacy relevance score for class C_i and privacy setting t after k iterations. Let $r_0(C_i, t) = \gamma(C_i, t)$ be the initial relevance score defined previously and define ψ as an $m \times m$ transition matrix, where ψ_{ij} is the transition probability from class C_i to C_j in the co-occurrence network and is defined as:

$$\psi_{ij} = \frac{\phi(C_i, C_j)}{\sum_{C_k \in \Omega_{C_i}} \phi(C_i, C_k)}, \tag{5.7}$$

where Ω_{C_i} is the set of neighbors of class C_i in the network, $\phi(C_i, C_j)$ is the inter-object correlation between defined in Eq. (5.5). The random walk process for refining relevance score is formulated as:

$$r_k(C_i, t) = \theta \sum_{C_j \in \Omega_{C_i}} r_{k-1}(C_i, t)\psi_{ij} + (1 - \theta)\gamma(C_i, t), \tag{5.8}$$

where θ is a weight parameter (in this work $\theta = 0.4$) for controlling the importance of the initial object-relevance score. For each object class C_i, privacy settings are re-ranked according to the object-privacy relevance scores and the privacy setting with the largest score is selected for the object class C_i. In the current implementation, there are 268 privacy-sensitive object classes.

5.3.3 Visual Tree

A visual tree is constructed to organize the large number of privacy-sensitive object classes. A visual tree is defined as $T = (V, E)$, where V is the set of nodes and E is the set of edges. Each non-leaf node $c \in V$ contains a set of privacy-sensitive object classes $L(c) \subseteq \{C_1, \ldots, C_M\}$. In the case of a leaf node, c contains only one object class ($|L(c)| = 1$). A child node only contains a subset of classes from its parent node.

For two object classes C_i and C_j, their inter-class visual similarity $S(C_i, C_j)$ is defined as:

$$S(C_i, C_j) = \frac{1}{R^2} \sum_{x_l \in C_i} \sum_{x_h \in C_j} \kappa_o(x_l, x_h), \tag{5.9}$$

where κ_o is the Gaussian kernel function for similarity characterization, x_l and x_h are the deep features[1] for the l-th image from class C_i and h-th image from class C_j respectively, and R is the total number of images from each object class. Inter-class visual similarities are used as a proxy to determine inter-class separability. If visual similarities are high, the object classes are harder to separate, so they should be assigned to the same coarse-grained group in visual tree construction, so that incorrect partitioning at high-level nodes are avoided. This helps to alleviate inter-level error propagation when traversing top-down in the visual tree.

Given a current non-leaf node c (which contains K privacy-sensitive object classes), its K object classes are partitioned into B smaller groups, i.e. B child nodes, and is

[1] The original paper does not mention where the deep features come from and in this report are assumed to be the results of CNN training in the image segmentation task.

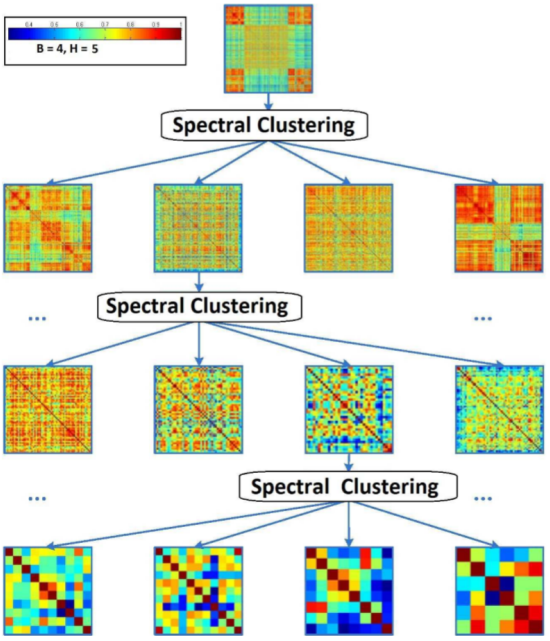

FIGURE 5.1: Visual Tree, $B = 4$

considered the branching factor for node partitioning by minimizing inter-group visual similarity and maximizing intra-group similarity:

$$min \left\{ \psi(c, B) = \sum_{l=1}^{B} \frac{\sum_{C_i \in G_l} \sum_{C_j \in G^c/G_l} S(C_i, C_j)}{\sum_{C_i \in G_l} \sum_{C_h \in G_l} S(C_i, C_h)} \right\}, \tag{5.10}$$

where $G^c = \{ G_l \mid l = 1, ..., B \}$ is used to represent B groups of K privacy-sensitive object classes for the current non-leaf node c, and G^c/G_l is used to represent the other $B - 1$ groups in G^c except G_l.

The node partitioning function using Eq. (5.10) is performed repeatedly until a complete tree is created, such that each leaf node only contains one privacy-sensitive class. If for a given node c, the number of classes $|L(c)| < B$, then only $|L(c)|$ number of children are generated. Finally, a coarse-to-fine visual tree representation is created for the set of privacy-sensitive object classes. Figure 5.1 shows an example of a visual tree with branching factor $B = 4$.

5.3.4 Joint Learning for Deep CNNs and Tree Classifier

Utilizing the visual tree, a tree classifier and deep CNNs are jointly learned for detection of privacy-sensitive object classes. As shown in Figure 5.2 shows that the framework of iPrivacy is divided into three parts: (1) 3 commonly-shared convolutional layers, (2) 2 group-specific convolutional layers and 2 group-specific fully-connected layers, and (3)

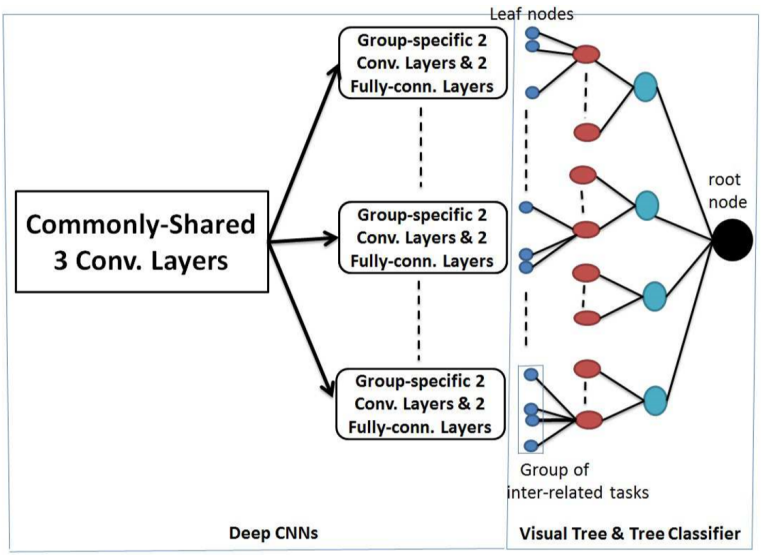

FIGURE 5.2: Illustration of the iPrivacy framework

the last layer for the tree-based softmax classifier layer to replace the flat softmax layer. Dropout value of 0.5 is applied to the 2 group-specific fully-connected layers to prevent over-fitting.

Joint learning of the deep CNNs and the tree classifier on the visual tree is done in a *bottom-up* fashion. A *deep multi-task learning* algorithm is developed to train using a joint objective function to refine the multi-task softmax classifiers for sibling leaf nodes and the deep CNNs for image representation. A *hierarchical deep multi-task learning* (HD-MTL) algorithm is developed to train more discrminative classifiers for high-level nodes.

5.3.4.1 Deep Multi-Task Learning

A deep multi-task learning algorithm is developed to train inter-related classifiers for sibling leaf nodes. Sibling leaf nodes share some similar visual properties as modeled when creating the visual tree so are strongly inter-related. This section describes the deep multi-task learning for a given parent node c_h at the second level[2]. For a given parent node c_h, the multi-task softmax classifiers for sibling leaf nodes are trained simultaneously by optimizing a joint objective function:

$$min \left\{ v \sum_{l=1}^{R} \sum_{j=1}^{B} \xi_j^l + \delta_1 Tr(WW^T) + \frac{\delta_2}{2} Tr(WLW^T) \right\}, \tag{5.11}$$

[2]In this work, the first level is defined as the bottom of the tree, i.e. the leaves of the tree, and higher levels are higher up in the tree.

subject to:

$$\forall_{l=1}^{R}\forall_{j=1}^{B} \ : \ y_j^l(W_j^T \cdot x_j^l + b) \geq 1 - \xi_j^l, \ \xi_j^l \geq 0, \tag{5.12}$$

where R is the number of training images for each class, W_j is the classifier parameter for the jth class C_j, $Tr(\cdot)$ is the trace of a matrix, ξ_j^l is the training error rate, δ_1 and δ_2 are regularization parameters, v is the penalty term, $W = (W_1, \ldots, W_B)$, y_j^l is the class indicator vector of example x_j^l, and L is the Laplacian matrix of the inter-class similarity matrix S. The visual similarities in S are used to approximate the inter-task relationships, and the regularization term $Tr(WLW^T)$ is used to enforce that. The dual problem is defined as:

$$min \left\{ \sum_{j=1}^{B}\sum_{l=1}^{R} \beta_l^j - \frac{1}{2\delta_1}\beta^T Y\mathcal{R}\left(\mathcal{R} + \frac{\delta_2}{\delta_1}\mathcal{R}\left(L \otimes I\right)\mathcal{R}\right)^{-1}\mathcal{R}Y\beta \right\}, \tag{5.13}$$

subject to:

$$\forall_{l=1}^{R}\forall_{j=1}^{B} \ : \ \sum_{l=1}^{R} \beta_l^j \dot{y}_l^j = 0, \ 0 \leq \beta_l^j \leq 1, \tag{5.14}$$

where $Y = \left[y_j^l \mid l \in \{1, \ldots, R\}, j \in \{1, \ldots, B\} \right]^T \in 0, 1^{R \times B}$ and $y_j^l \in \{0, 1\}^{B \times 1}$ is the class indicator vector for training image x_j^l, I is the identity matrix, $\otimes$ is the Kronecker product between two matrices, $\beta = (\beta_1, \ldots, \beta_j, \ldots, \beta_B)$ is the set of dual variables, $\beta_j = (\beta_1^j, \ldots, \beta_l^j, \ldots, \beta_R^j)$, and $\mathcal{R}$ is a block diagonal similarity matrix, where $\mathcal{R}_j \in R^{R \times R}$ is the similarity matrix for R training images for the jth privacy-sensitive object class. The optimal β^* is obtained by solving Eq. (5.13) subject to (5.14). The optimal $\alpha_* = (\alpha_1^*, \ldots, \alpha_j^*, \ldots, \alpha_B^*)$ for the softmax function is computed as:

$$\alpha^* = \frac{1}{2\delta_1}\left(\mathcal{R} + \frac{\delta_2}{\delta_1}\left(\mathcal{R}\left(L \otimes I\right)\mathcal{R}\right)^{-1}\mathcal{R}Y\beta^*\right), \tag{5.15}$$

and the multi-task softmax classifiers for the sibling nodes under parent node c_h are defined as:

$$\forall_{j=1}^{B} \ : \ f_{c_j}^1(x)|_{F_{c_j}^1} = \sum_{l=1}^{R} \alpha_j^{l*}\kappa(x_j^l, x) + b_j^*, \ c_j \in c_h. \tag{5.16}$$

The inter-class visual similarities are embedded into a manifold structure regularization. The softmax classifiers for each set of sibling nodes are trained jointly and separately from children nodes of other parents, which allows the softmax classifiers to learn to discriminate between visually similar objects, while also optimizing a common prediction function shared among the multi-task softmax classifiers for the same parent node.

The errors of the inter-related learning task are backpropagated to update the weights for the deep CNNs [43–46]. The training error rate ξ_j^l in the form of softmax regression is:

$$\xi_j^l = -I\{y_j^l\} \log \left\{ \frac{\exp\left(W_j^T x_j^l + b\right)}{\sum_{i=1}^{B} \exp\left(W_i^T x_i^l + b\right)} \right\}, \tag{5.17}$$

where $I\{y_j^l\}$ is the indicator function such that $I\{y_j^l\} = 1$, if $y_j^l = 1$, otherwise $I\{y_j^l\} = 0$. Eq. (5.11) and (5.12) can be reformulated as:

$$\begin{aligned} \mathcal{L}\left(W, X, Y\right) = \sum_{l=1}^{R} \sum_{j=1}^{B} \xi_j^l + \delta_1 Tr(WW^T) + \frac{\delta_2}{2} Tr(WLW^T) + \\ \lambda \left\{ \sum_{l=1}^{R} \sum_{j=1}^{B} y_j^l \left(W_j^T \cdot x_j^l + b\right) - 1 - \xi_j^l \right\}, \end{aligned} \tag{5.18}$$

where λ is the Lagrange weight parameter. The objective function is optimized using the stochastic alternating direction method of multipliers (ADMM) algorithm [47, 48].

5.3.4.2 Hierarchical Deep Multi-Task Learning

An important principle in the hierarchical object detection approach is that: "an image or an object proposal x from the image should first be assigned into the parent node correctly (i.e. $y_{c_h}^{l+1} \cdot f_{c_h}^{l+1} \geq 0,\ y_{c_h}^{l+1} = +1$), if it can further be assigned into the child node (i.e. $y_{c_j}^{l} \cdot f_{c_j}^{l} \geq 0,\ y_{c_h}^{l+1} = +1$)" [22]. This gives a discriminative regularization term defined as:

$$\forall_{j=1}^{B} \ : \ f_{c_h}^{l+1}(x) - f_{c_j}^{l}(x) \geq 0,\ c_j \in c_h. \tag{5.19}$$

This allows the HD-MTL algorithm to train more discriminative classifiers for high-level nodes, while controlling the inter-level error propagation. In addition, "all images or object proposals, which can be assigned into the same parent node correctly, should further

be able to be assigned into the relevant child nodes", so an inter-level complementarity constraint is defined as:

$$\forall_{h=1}^{B} \ : \ f_{c_h}^{l+1}(x) = \sum_{j=1}^{B} \eta_j f_{c_j}^{l}(x), \tag{5.20}$$

where η_j is defined as the normalized softmax probability across all sibling softmax classifiers:

$$\eta_j = \frac{\exp\left(-f_{c_j}^{l}(x)\right)}{\sum_{i=1}^{B} \exp\left(-f_{c_i}^{l}(x)\right)}. \tag{5.21}$$

The training of the multi-task softmax classifiers for sibling *non-leaf* nodes under a parent node c_k is given by:

$$min\left\{ v \sum_{m=1}^{R} \sum_{h=1}^{B} \xi_j^m + \delta_1 Tr(WW^T) + \frac{\delta_2}{2} Tr(WLW^T) \right\}, \tag{5.22}$$

subject to:

$$\forall_{m=1}^{R} \forall_{h=1}^{B} \ : \ y_h^m \left(W_h^T \cdot x_h^m + b\right) \geq 1 - \xi_h^m, \ \xi_h^m \geq 0, c_h \in c_k \tag{5.23}$$

$$\forall_{j=1}^{B} \ : \ f_{c_h}^{l+1}(x) - f_{c_j}^{l}(x) \geq 0, \ c_j \in c_h \tag{5.24}$$

$$\forall_{h=1}^{B} \ : \ f_{c_h}^{l+1}(x) = \sum_{j=1}^{B} \eta_j f_{c_j}^{l}(x). \tag{5.25}$$

The HD-MTL algorithm provides an iterative solution for large-scale learning of privacy-object classes. By using the subtrees of the visual tree, learning can be done on at most B sibling children and one parent node iteratively on inter-related learning tasks. The tree classifiers can partition out coarse-grained groups of object classes at early stage, which reduces the cost of detecting privacy sensitive objects. Sibling non-leaf nodes are jointly trained using Eq (5.22), and backpropagation is used to update the multi-task softmax classifiers on non-leaf nodes, multi-task softmax classifiers on child nodes, and weights for the deep CNNs.

5.3.5 Soft Prediction

After the tree classifier and deep CNNs are trained, they can be used to predict the label (object class) of an image or x. To perform prediction, an image[3] traverses the tree classifier starting from the root of the visual tree. Its confidence score, $\rho(c_j, x)$, with the multi-task softmax classifier $f^l_{c_j}(x)$ for a non-leaf node c_j at the l-th level is defined as:

$$\rho(c_j, x) = \frac{\exp\left(-f^l_{c_j}(x)\right)}{\sum_{i=1}^{B} \exp\left(-f^l_{c_i}(x)\right)}. \tag{5.26}$$

The margin $\delta(c_j, c_i, x)$ between the confidence scores of sibling nodes c_j and c_i of a parent node c_h is defined as:

$$\delta(c_j, c_i, x) = |\rho(c_j, x) - \rho(c_i, x)|\,,\ c_j, c_i, \in c_h. \tag{5.27}$$

At the final level of the visual tree, the multi-task softmax classifiers for the sibling object classes under parent c are used to assign the image x a privacy-sensitive object class. Given two thresholds T_1 and T_2, an image x is assigned into one sibling node c_j if the confidence $\rho(c_j, x) > T_1$ or the margin for all $c_i, (i \neq j)$, $\delta(c_j, c_i, x) > T_2$. Otherwise, it is assigned into the uncertain category or a new privacy-sensitive object class under the parent node c. This is called a *hard prediction*.

Hard prediction is not ideal, since a wrong child node can be chosen early and the mistake will propagate throughout the tree traversal. Instead, a *soft prediction* method is used, where the top 2 child nodes are chosen to traverse through. The chosen leaf nodes are sorted according to the confidence scores and the image is assigned a top-N[4] privacy-sensitive object classes in the soft prediction approach.

5.4 Conclusion

A new approach called iPrivacy is developed by Yu et al. [22] for privacy setting recommendation for shared images. To do this, massive social images are leveraged, along with semantic image segmentation and object detection methods. Image segmentation and privacy settings for images are used to develop a large set of privacy-sensitive object classes, which is then partitioned using similarity features in a hierarchical way using

[3] In the following example, we just use "image" to refer to the test example.
[4] In the paper, the authors explicitly state top 3 classes.

a visual tree. Deep CNNs and a tree classifier are jointly learned using the visual tree to provide inter-related learning tasks for each level in the tree classifier. Finally, privacy settings of images can be done using segmentation and traversing through the tree classifier using soft prediction.

5.4.1 Critiques

There are some gaps in the discussed paper as well as some limitations of the proposed method. Regarding the paper itself, the architecture of the iPrivacy model is not specified very clearly. The authors mention that there are two group-specific convolutional layers and two group-specific fully connected layers, however they do not mention the intuition behind the training and number of "groups" for that layer. It is not clear whether the backpropagation and training is done using all group-specific layers, or if some type of selective training is done via sibling nodes for each group. Additionally, the evaluations seem to not be extensive enough[5], since accuracy for predicting privacy settings was compared with traditional visual features such as SIFT, GIST, and color histograms, rather than some other deep architecture. For example, Tonge and Caragea [21] and Spyromitros-Xioufis et al. [35] both use deep features in predicting privacy settings for images and showed that deep features have better performance.

The iPrivacy model itself does have some limitations. One limitation is that the system only looks at visual features for predicting privacy settings, much like many previous works. This is not ideal for two reasons. First, visual features are assumed to be highly correlated with privacy-sensitiveness of an image, but there may be other features which may be predictive of privacy, such as tags for images [21, 49]. Second, social network information, such as friend lists and communities, are not utilized in this model. Many times, users may only wish for *some* images to be hidden from certain people [13]. In the iPrivacy model, privacy settings are predicted only from visual features.

Following the previously mentioned limitation, users often have different privacy preferences [7, 9] and models should take into account the user preferences [35]. Feedback could potentially be an important factor in predicting privacy settings for friends, as well as other community features [10, 35].

[5]Note that this report did not go into detail about the experiments and evaluations conducted in [22].

Chapter 6

Conclusions and Future Work

Methods for preserving privacy in shared images have been reviewed in this paper. Three main methods are discussed: (1) using tags for access-control, (2) privacy preserving image transformations and encryption, (3) automatic privacy prediction. Some potential avenues for future work are briefly discussed below.

Bundling additional features and solutions Fan et al. [50] use semantic image segmentation as visual features for predicting privacy of images. It has been shown that tags can potentially be effective in managing privacy settings [13, 21, 49]. One potential future work is to incorporate semantic tags, and potentially other information in the iPrivacy model to get a better system for creating access-control policies. Friendship and community information in social networks are also useful features in predicting privacy settings [10]. Additionally, more fine-grained privacy settings could be explored rather than having coarse-grained privacy settings such as: (1) shared with friends, (2) public, (3) private. Finally, after building a sufficient system for detecting privacy sensitive regions, automatic encryption could be done to protect private regions.

Modeling privacy preferences of users In the works discussed, none of the models explicitly model user privacy preferences. Wisniewski et al. [9] model different privacy management strategies as well as feature awareness for users of social networking sites and show that people can be divided into several types of privacy management strategies. This shows that user privacy preference can be very subjective, so having models learn user preference directly is potentially a better way forward. In addition to images, privacy settings of posts and profile items should be automated. In this vein, some work has been done in the Twitter space. On Twitter, users can only leave a tweet, delete a tweet, or set their profile to private (so all tweets are inaccessible). Work has been

done on characterizing deleted tweets and predicting if a tweet will be deleted [51, 52]. Finally, social network structure should be taken into account (e.g communities, friend relationships, etc.), when assigning privacy labels to posts and shared images. Users do may only wish to deny *some* profile items and have different preferences for each friend [10, 13]. Machine learning models that model the *user's* privacy preference may provide better and more personalized privacy setting recommendation and automation.

Bibliography

[1] Alessandro Acquisti and Ralph Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *International workshop on privacy enhancing technologies*, pages 36–58. Springer, 2006.

[2] Heather Richter Lipford, Andrew Besmer, and Jason Watson. Understanding privacy settings in facebook with an audience view. *Usability, Psychology, and Security (UPSEC)*, 8:1–8, 2008.

[3] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM, 2005.

[4] Katherine Strater and Heather Richter Lipford. Strategies and struggles with privacy in an online social networking community. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1*, pages 111–119. British Computer Society, 2008.

[5] Patricia A Norberg, Daniel R Horne, and David A Horne. The privacy paradox: Personal information disclosure intentions versus behaviors. *Journal of Consumer Affairs*, 41(1):100–126, 2007.

[6] Yabing Liu, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 61–70. ACM, 2011.

[7] Shane Ahern, Dean Eckles, Nathan Good, Simon King, Mor Naaman, and Rahul Nair. Over-Exposed? Privacy Patterns and Considerations in Online and Mobile Photo Sharing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 357–366, 2007. doi: 10.1145/1240624.1240683.

[8] Susan B Barnes. A privacy paradox: Social networking in the united states. *First Monday*, 11(9), 2006.

[9] Pamela J Wisniewski, Bart P Knijnenburg, and Heather Richter Lipford. Making privacy personal: Profiling social network users to inform privacy education and nudging. *International Journal of Human-Computer Studies*, 98:95–108, 2017.

[10] Lujun Fang and Kristen LeFevre. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World wide web*, pages 351–360. ACM, 2010.

[11] Ramprasad Ravichandran, Michael Benisch, Patrick Gage Kelley, and Norman M Sadeh. Capturing social networking privacy preferences. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2009.

[12] Ching-man Au Yeung, Lalana Kagal, Nicholas Gibbins, and Nigel Shadbolt. Providing access control to online photo albums based on tags and linked data. In *AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0*, pages 9–14, 2009.

[13] Peter F Klemperer, Yuan Liang, Michelle L Mazurek, Manya Sleeper, Blase Ur, Lujo Bauer, Lorrie Faith Cranor, Nitin Gupta, Michael K Reiter, and Chapel Hill. Tag, You Can See It! Using Tags for Access Control in Photo Sharing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 377–386, 2012. doi: 10.1145/2207676.2207728.

[14] Panagiotis Ilia, Iasonas Polakis, Elias Athanasopoulos, Federico Maggi, and Sotiris Ioannidis. Face/off: Preventing privacy leakage from photos in social networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 781–792. ACM, 2015.

[15] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. P3: Toward Privacy-Preserving Photo Sharing. *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, 2013.

[16] Jianping He, Bin Liu, Xuan Bao, Hongxia Jin, and George Kesidis. PuPPIeS : Privacy Preserving Partial Image Sharing. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.

[17] Sergej Zerr, Stefan Siersdorfer, Jonathon Hare, and Elena Demidova. Privacy-aware image classification and search. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 35–44. ACM, 2012.

[18] 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.

[19] Chen Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, may 2018.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015. URL https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2015/app/2B{_}011.pdf.

[21] Ashwini Tonge and Cornelia Caragea. Privacy prediction of images shared on social media sites using deep features. *arXiv preprint arXiv:1510.08583*, 2015.

[22] Jun Yu, Baopeng Zhang, Zhengzhong Kuang, Dan Lin, and Jianping Fan. iprivacy: image privacy protection by identifying sensitive objects via deep multi-task learning. *IEEE Transactions on Information Forensics and Security*, 12(5):1005–1016, 2017.

[23] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[24] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

[25] Gregory K Wallace. The JPEG Still Picture Compression Standard. *IEEE Transactions on Consumer Electronics*, 38(1), 1992.

[26] OpenCV. Face Detection using Haar Cascades. URL https://docs.opencv.org/trunk/d7/d8b/tutorial{_}py{_}face{_}detection.html.

[27] Tesseract. Tesseract OCR Project. URL https://github.com/tesseract-ocr/.

[28] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object ? *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[29] Chiou-Ting Hsu and Ja-Ling Wu. Hidden digital watermarks in images. *IEEE Transactions on image processing*, 8(1):58–68, 1999.

[30] M Everingham, L Van˜Gool, C K I Williams, J Winn, and A Zisserman. The {PASCAL} {V}isual {O}bject {C}lasses {C}hallenge 2007 {(VOC2007)} {R}esults. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[31] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[32] Aditya Vailaya, Anil Jain, and Hong Jiang Zhang. On image classification: City images vs. landscapes. *Pattern Recognition*, 31(12):1921–1935, 1998.

[33] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[34] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42 (3):145–175, 2001.

[35] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Adrian Popescu, and Yiannis Kompatsiaris. Personalized privacy-aware image classification. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 71–78. ACM, 2016.

[36] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and Others. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[39] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[40] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Proceedings of the twenty-first international conference on Machine learning*, page 27. ACM, 2004.

[41] Junhui Wang, Xiaotong Shen, and Wei Pan. On large margin hierarchical classification with multiple paths. *Journal of the American Statistical Association*, 104 (487):1213–1223, 2009.

[42] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for Object Segmentation and Fine-grained Localization. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[43] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *International conference on machine learning*, 2014. URL http://proceedings.mlr.press/v32/donahue14.pdf.

[44] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding *. *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 2012.

[46] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[47] Samaneh Azadi and Suvrit Sra. Towards an optimal stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, pages 620–628, 2014.

[48] Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, pages 80–88, 2013.

[49] Anna Squicciarini, Andrea Novelli, Dan Lin, Cornelia Caragea, and Haoti Zhong. From tag to protect: A tag-driven policy recommender system for image sharing. *Privacy, Security and Trust (PST)*, 2017.

[50] Jianping Fan, Zhenzhong Kuang, Baopeng Zhang, Jun Yu, and Dan Lin. iPrivacy: Image Privacy Protection by Identifying Sensitive Objects via Deep Multi-Task Learning. *IEEE Transactions on Information Forensics and Security*, 2016.

[51] Lu Zhou, Wenbo Wang, and Keke Chen. Tweet properly: Analyzing deleted tweets to understand and identify regrettable ones. In *Proceedings of the 25th International Conference on World Wide Web*, pages 603–612. International World Wide Web Conferences Steering Committee, 2016.

[52] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. I wish i didn't say that! analyzing and predicting deleted messages in twitter. *arXiv preprint arXiv:1305.3107*, 2013.