An analysis of automatic image filtering on WeChat Moments

Jeffrey Knockel, Lotus Ruan, and Masashi Crete-Nishihata
The Citizen Lab, Munk School of Global Affairs, University of Toronto
{jeff,lotusruan,masashi}@citizenlab.ca

Abstract

We report results from a series of experiments that uncover mechanisms used to filter images on WeChat, the most popular social media platform in China. Our results inform strategies for evading image filtering on the application. By performing tests on a collection of politically sensitive images filtered by WeChat, we found that WeChat uses two different algorithms to filter, an Optical Character Recognition (OCR)-based algorithm that filters images containing sensitive text, and a visual-based algorithm that filters images that are visually similar to those on an image blacklist. The OCR-based algorithm has implementation similarities to many common OCR algorithms that allow us to create text images that evade filtering. We found that the visual-based algorithm does not use any machine learning approach that uses high level classification of an image to determine whether it is sensitive; however, we discovered multiple implementation details of the visual-based algorithm that inform the creation of images that are visually similar to those blacklisted but that evade filtering. This study is the first in-depth technical analysis of image filtering on WeChat, and we hope that our methods will serve as a road map for studying image censorship on other platforms.

1 Introduction

WeChat is the most popular social media platform in China and the fourth largest in the world [7]. Users in China spend a third of their online time on WeChat and typically return to the app ten times a day or more [34]. The application is owned and operated by Tencent, one of the largest technology companies in China. One of its most frequently used functions is WeChat Moments [33], a feature similar to the Facebook Timeline in which users can share images and other content.

Any application operating in China is subject to government mandated information controls and companies are expected to invest in technology and personnel to carry out content regulations [28]. Prior work on WeChat has analyzed keyword-based filtering [6, 30, 31], censorship of Public Posts (a platform designed for companies and users to make public blog posts requiring manual approval for account registration) [25], and has reported cases of image filtering [6, 30]. Missing from the literature is an in-depth analysis of how image filtering is technically implemented. We address this gap through a series of experiments. By performing tests on a collection of politically sensitive images filtered by WeChat, we uncover mechanisms WeChat uses to filter images, and through these results we identify multiple evasion strategies.

We found that WeChat Moments uses two different algorithms to filter images, an Optical Character Recognition (OCR)-based algorithm that filters images containing sensitive text, and a visual-based algorithm that filters images that are visually similar to those on an image blacklist. We found that the OCR-based algorithm has implementation similarities to many common OCR algorithms that allow us to create image text that evades filtering. We found that the visual-based algorithm is not based on any machine learning approach that uses high level classification of an image to determine whether it is sensitive or not; however, we identified multiple implementation details of the visual-based algorithm that allow us to create images that are visually similar to those blacklisted but that evade filtering.

Through our findings we provide a better understanding of how image filtering is implemented on an application with over one billion users. We hope that our methods will generalize and serve as a road map for studying image censorship on other platforms.

2 Related work

Content censorship can be implemented *client-side* (*i.e.*, on an application itself) or *server-side* (*i.e.*, on a remote server). In a client-side implementation, the rules to perform censorship are inside of the application running on a user's device. An application with client-side keyword censorship will often have a built-in list of keywords, which can be updated when the client connects with a server. If any banned keywords are present in a user's message before the message is sent, the message is not sent. In a server-side implementation the rules to perform censorship are on a remote server. When a message is sent, it passes through the server which checks if banned keywords are present and, if detected, blocks the message.

Prior work on applications that implement client-side censorship has used reverse engineering techniques to extract keyword lists used to trigger censorship on chat apps [17, 5, 12], live streaming services [18], and online games [19]. Studies of server-side censorship generally rely on sample testing in which researchers either (1) develop a set of content suspected to be blocked by a platform, send the sample to the platform, and record the results [24, 31]; or (2) select public posts and monitor them for deletion [1, 9, 14, 38, 25, 3].

WeChat hosts user-generated content through three main features: chat functions, WeChat Moments, and the Public Account platform. Previous research has documented censorship on all of these features [25, 30, 31]. WeChat implements filtering server-side and only enables censorship for users with accounts registered to mainland China phone numbers [31]. Censorship on WeChat is not transparent: the message or post containing sensitive content simply does not appear on the receiver's end and no notice is given to the sender that their message is blocked or why it was blocked. Previous research [6, 30, 35] has reported incidents of images being blocked on WeChat but has not systematically investigated the technical mechanism used to implement image filtering. Our study contributes to the literature with the first in-depth technical analysis of image filtering on WeChat.

Neural network-based image classifiers are known to be vulnerable to adversarial examples, *i.e.*, images with minor modifications that cause them to be misclassified [32, 10]. While early work in generating adversarial examples assumed a whitebox threat model where all implementation details including the trained gradients of the target network were known, more recent work [23, 4, 13] is capable of generating adversarial examples by estimating the network's gradients. However, the work assuming the most restrictive threat model [13] is limited in that it still assumes that the attacker can acquire probability scores for the top k categories for arbitrary images. In our work, we have found that not only does WeChat not use a machine learning approach for filtering images but, even if it did, the threat model for evading a censorship filter is even more restrictive than that currently assumed in the adversarial example literature, as the only signal available is whether or not an uploaded image is filtered. Our work differs from that in the adversarial example literature in that we construct evasion techniques by discovering and exploiting other important implementation details of the filtering algorithm that are effective independently of whether the filter uses machine learning classification.

3 Analysis

We measured which images were automatically filtered on WeChat Moments by posting images using an account registered to a non-Chinese phone number and measuring whether they were visible using an account registered to a Chinese phone number. We found that WeChat uses two different filtering mechanisms to filter images: an OCR-based approach that searches images for sensitive text and a visual-based approach that visually compares an uploaded image against a list of blacklisted images. In this section we describe how testing for and understanding implementation details of both of these filtering methods led to effective evasion techniques.

3.1 OCR-based filtering

We found that one approach that Tencent uses to filter sensitive images is to use OCR technology to detect sensitive text inside the image. In this section we outline ways to evade WeChat's OCR filtering discovered by identifying two different stages of the OCR algorithm. To restrict our analysis to automated filtering, we only considered images filtered within 60 seconds of being posted, as we found that images filtered using OCR were typically removed in 5 to 30 seconds.

3.1.1 Grayscale conversion

OCR algorithms may use different strategies to recognize text. However, at a high level, we found that WeChat's OCR algorithm shares implementation details with other algorithms. As most OCR algorithms do not operate directly on color images, the first step they take is to convert a color image to grayscale so that it only consists of black, white, and intermediate shades of gray, as this largely simplifies text recognition since the algorithms only need to operate on one channel.

To test if WeChat's OCR filtering algorithm grayscale-converted color images, we designed test images that would evade filtering if the OCR algorithm converted uploaded images to grayscale. We designed the images to contain text hidden in the hue of an image in such a way that it is easily legible by a person reading it in color but such that once it is converted to grayscale, the text disappears and is invisible to the OCR algorithm. If the image evaded censorship, then the OCR algorithm converts images to grayscale (see Figure A.1 for an illustration)[1].

As we did not know which formula the OCR algorithm used to convert color images to grayscale, we evaluated multiple possibilities. In principle, the gray intensity of a color pixel could be calculated according to any function of its red, green, and blue intensities. We evaluated three different formulas, the average [36, Chapter 9], lightness [36, Chapter 9], and luminosity [26, 22] formulas:

- $\textit{average}(r, g, b) = \frac{1}{3}(r + g + b)$
- $\textit{lightness}(r, g, b) = \frac{1}{2}\max(r, g, b) + \frac{1}{2}\min(r, g, b)$
- $\textit{luminosity}(r, g, b) = 0.299r + 0.587g + 0.114b.$

To evaluate each formula, we created images containing filtered text in six different colors: red, (1.0, 0, 0); yellow, (1.0, 1.0, 0); green, (0, 1.0, 0); cyan, (0, 1.0, 1.0); blue, (0, 1.0, 1.0); and magenta, (1.0, 0, 1.0); where (r, g, b) is the color in RGB colorspace and 1.0 is the highest intensity of each channel. These six colors were chosen because they have maximum saturation in the HSL colorspace and a simple representation in the RGB colorspace. For each color (r, g, b) and for grayscale formula f, we created an image whose text was color (r, g, b) and whose background was the gray color (Y, Y, Y), where $Y = f(r, g, b)$. We chose for the text content of each image a selection of 25 keyword combinations randomly chosen from a set of keywords we found from testing to be filtered via OCR filtering (see Figure A.2 for an illustration). We used the same text content for each image.

After performing this test, we found that only when choosing the intensity of the gray background as given by the luminosity formula could we consistently evade filtering for every tested color. The other formulas failed to evade censorship with most colors. The averaging algorithm only evaded filtering when using red or cyan text, and the lightness algorithm only evaded when using green or magenta text.

To confirm that using the luminosity formula to choose the text's background color consistently evaded WeChat's OCR filtering, we performed a more extensive test using only that algorithm. We selected five lists of 25 randomly chosen keywords we found from testing to be blocked. We also selected five lists of 10, 5, 2, and 1 keyword(s) chosen at random. For each of these lists, we created six images, one for each of the same six colors we used in the previous experiment. Our results were that all 150 images evaded filtering. These results show that we can consistently evade WeChat's filtering by hiding colored text on a gray background chosen by the luminosity of the text and that WeChat's OCR algorithm uses the same or similar formula for grayscale conversion.

[1]Due to the volume of illustrations and data, all figures are in Appendix A.

3.1.2 Blob merging

After converting a colored image to grayscale, another step in most OCR algorithms is to apply a thresholding algorithm to the grayscale image to convert each pixel, which may be some shade of gray, to either completely black or completely white such that there are no shades of gray in between. Then, after thresholding, some OCR algorithms perform a step called blob merging. In order to recognize each character, these algorithms try to determine which blobs in an image correspond to each character. Many characters such as the English letter "i" are made up of unconnected components. In languages such as Chinese, individual characters can be made up of many unconnected components (*e.g.*, 診). OCR algorithms use a variety of algorithms to try to combine these blobs into characters and to evaluate which combinations produce the most recognizable characters.

To test whether WeChat's OCR filtering performed blob merging, we experimented with uploading images that could be easily read by a person but that would be difficult to read by an algorithm piecing together blobs. To do this, we experimented with using two different patterns to fill text instead of using solid colors. Specifically, we used a tiled square pattern and a tiled letter pattern (see Figure A.3), both black on white. Using these patterns causes most characters to be made up of a large number of disconnected blobs in a way that is easily readable by most people but that is difficult for OCR algorithms performing blob merging. The second pattern that tiles English letters was designed to especially confuse an OCR algorithm by tricking it into finding the letters in the tiles as opposed to the larger characters that they compose.

To test if blobs of this type affected the OCR algorithm, we created a series of test images. We selected five lists of 25 randomly chosen keywords we knew to be blocked. We also selected five lists of 10, 5, 2, and 1 keyword(s) chosen at random. For each of these lists, we created two images, one with the text patterned in squares and another patterned in letters. For images with a large number of keywords, we decreased the font size to ensure that the generated images fit within a 1000×1000 pixel image. This is to ensure that images did not become too large and to ensure that they would not be downscaled, as we had previously experienced some images larger than 1000×1000 downscaled by WeChat. We did this to control for any effects that downscaling the images could have on our experiment such as blurring the text.

Our results were that square-patterned text evaded filtering in 92% of our tests, and letter-patterned text evaded filtering in 100% of our tests. Two tests of the square pattern failed to evade filtering. Both of them were test images containing 25 keywords. The reason for this failure is not clear, but we consider two possibilities. One is that the higher number of keywords per image increased the probability that at least one of those keywords would not evade filtering. The second is that images with a larger number of keywords used a smaller font size, and so there were fewer blobs per character, reducing the effectiveness of the evasion strategy. Letters were more effective in evading filtering. This result may be because of the previously suggested hypothesis that the OCR filter would be "distracted" by the letters in the pattern and thus miss the characters in which they collectively form, but it may also be because the letters are less dense insofar as they have fewer black pixels per white. Overall, these results suggest that WeChat's OCR filtering algorithm considers blobs when performing text recognition and that splitting characters into blobs is an effective evasion strategy.

3.2 Visual-based filtering

In addition to OCR-based filtering, to censor images that do not contain text, we found that WeChat uses another filtering algorithm that works by comparing an image's visual similarity to those on a list of blacklisted images. We performed modifications to politically sensitive images blacklisted by WeChat to test different hypotheses concerning how the filter operated and to inform strategies for evading the visual-based filter. Like when testing WeChat's OCR-based filtering, to restrict our analysis to automated filtering, we again only considered images filtered within 60 seconds of being posted, although we found that images filtered using visual-based methods were typically removed within only 10 seconds, often so quickly that they were never visible in our other account's view. As we found that the visual-based method typically takes less time than the OCR-based one, their visual-based algorithm would appear to be less computationally expensive than the one used for OCR filtering.

3.2.1 Grayscale conversion

Similar to our testing of the OCR-based algorithm, we performed an experiment determining if and how the visual-based algorithm converts images to grayscale. Unlike when testing the OCR-based algorithm where the foreground consisted of text, for testing the visual-based algorithm our foreground consisted of the white pixels of a black-and-white image. We took a political cartoon featuring the Hong Kong and PRC flags and thresholded it to black and white. After verifying that the thresholded version of the image was still filtered, we used the white pixels of the image as the foreground and the black pixels of the image as the background (see Figure A.4 for an illustration). For each of the six different foreground colors, we again selected the background according to three grayscale algorithms, and found that, like when testing the OCR algorithm, only when using the luminosity formula were the images consistently filtered.

3.2.2 Machine learning classification

Machine learning can be used to classify images into high level categories such as "cat" or "dog" depending on the contents of the image. If WeChat chose to use a machine learning classification approach, they could attempt to train a network to recognize whether an image may lead to government reprimands. However, training a network against such a nebulous and nuanced category would be rather difficult considering the vagueness of Chinese content regulations and the fluidity of what is considered sensitive [21]. Instead, they might identify certain categories of images that would be potentially sensitive, such as images of Falun Gong practitioners or of Liu Xiaobo, the late Nobel prize-winning dissident, and then classify whether images belong to these sensitive categories.

In our analysis, we found ample evidence that they do not use such a categorization system. We investigated different transformations to 15 images that we found from testing to be filtered on WeChat (see Figure A.5). For instance, many image transformations such as mirroring typically preserve the semantic meaning of an image (*e.g.*, a mirrored image of a cat is still a cat). However, when we mirrored the 15 images, we found that none of them were filtered after mirroring (see Figure A.6 for an illustration). Other semantic-preserving operations such as cropping or adding whitespace to an image also evaded filtering. These and other results suggest that no sort of high level machine learning classification system is being used to trigger the observed filtering on WeChat. Rather, these results suggest that there is a specific blacklist of images being maintained by WeChat that each image uploaded is somehow being compared against using some similarity metric.

While many Internet technology companies are known to use machine learning to flag pornographic content [37, 29, 16], in this study we focus on content automatically filtered on WeChat for political reasons. For this type of content, a blacklist approach may be desirable as it allows WeChat to quickly censor specific sensitive images that may be trending or that they are otherwise asked to censor by a government official regardless of the topic or category of the image. However, as we found, this approach also allows one to evade the filter by simple image transformations like mirroring since the filter has no semantic understanding of what the contents of images are.

3.2.3 Edge detection

Edges may be used to compare image similarity. Intuitively, edges represent the boundaries of objects and other features in images. There are generally two approaches to edge detection. The first approach, as taken by (*e.g.*) a Sobel filter, involves taking the differences between adjacent pixels. One weakness of this approach is that by signifying small differences as low intensity and larger differences as high intensity, it still does not specify in a 0-or-1 sense which are the "real" edges and which are not. A different approach is to use a technique like Canny edge detection which uses a number of filtering and heuristic techniques to reduce each pixel of a Sobel-filtered image to either black (no edge) or white (an edge is present) (see Figure A.7 for an illustration). As this reduces each pixel to one bit, it is more computationally efficient to use as an image feature.

There is some reason to think that WeChat's filtering may incorporate edge detection. When we searched online patents for references to how Tencent may have implemented their image filtering, we found that in June 2008 Tencent filed a patent in China called "图片检测系统及方法" (System and method for detecting a picture) [39]. In it they describe a real-time system for detecting blacklisted images after being uploaded that performs Canny edge detection before generating a fingerprint.

We found designing experiments to test for the use of Canny edge detection difficult. The algorithm is highly parameterized, and the parameters are often determined dynamically using heuristics based on the contents of an image. Moreover, unlike many image transformations such as grayscale conversion, Canny edge detection is not idempotent, *i.e.*, the canny edge detection of a canny edge detection is not the same as the original canny edge detection. This means that we cannot simply upload an edge-detected image and see if it gets filtered. Instead, we created test images by removing as many potentially relevant features of an image as possible while preserving the edges of an image. We used thresholding to reduce each pixel to either black or white, eliminating any gray pixels from the image, while hopefully largely preserving the edges in the image (see Figure A.8 for an illustration). We performed this technique on the 15 images we tested in the previous section (see Figure A.5) using a threshold dynamically chosen according to Otsu's method [27]. We found that all but two of the images were still filtered after being thresholded. Among the two images that were not filtered, one was the image of Liu Xiaobo's empty chair (Figure A.5 Image #2). This result may be because the threshold chosen by Otsu's method did not distinguish the stripes on the chair. The other was a photograph of Liu Xiaobo and his wife clanging coffee cups (Figure A.5 Image #9). This result may be because thresholding does not preserve edges well with backgrounds with gradients, as thresholding will typically create an erroneous edge where none actually exists.

As an additional test, we took the 15 images thresholded using Otsu's method and inverted them. This technique would preserve the location of all edges while radically altering the intensity of many pixels. We found that among the thirteen images that were filtered after applying Otsu's method, only four images were filtered after they were additionally inverted (see Figure A.5 images 5, 6, 7, and 10). The two images that were not filtered before were also not filtered after being inverted. This result suggests that, if edge detection is used, it is either in addition to other features of the image, or the edge detection algorithm is not one such as the Canny edge detection algorithm which only tracks edges not their "sign", *i.e.*, whether the edge is going from lighter to darker versus darker to lighter.

After our experiment eliminating as many potentially relevant features as possible except for edges, we tried the opposite experiment by eliminating edges by blurring them while keeping other features untouched. We proportionally resized each image such that its smallest dimension(s) is/are 200 pixels (see Section 3.2.4 for why we resized this way). Then we applied a normalized box filter to blur the image, increasing the kernel size until the image is sufficiently blurred to evade filtering.

In general, we saw that for most images WeChat's filter was not robust to blurring (see Figure A.9 for full results). Non-photographic images were generally the easiest to evade filtering by blurring, possibly because they generally have sharper and more well-defined edges.

3.2.4 Resizing

Up until this point, we have been mostly concerned with experimenting with images that have the same aspect ratios. In this section we test how changing images' dimensions affected WeChat's ability to recognize them. For instance, we found WeChat's filter clearly had the ability to filter sensitive images regardless of scale so long as the aspect ratio had been preserved. We wanted to explore whether WeChat normalizes the dimensions of uploaded images to a canonical size, and, if so, how.

To answer these questions, we decided to test five different hypotheses: (1) Images are proportionally resized such that their width is some value such as 100. (2) Images are proportionally resized such that their height is some value such as 100. (3) Images are proportionally resized such that their largest dimension is some value such as 100. (4) Images are proportionally resized such that their smallest dimension is some value such as 100. (5) Both dimensions are resized according to two parameters to some fixed size and proportion such as 100×100.

If the last hypothesis is correct, then we would expect WeChat's filter to be robust to modifications to a sensitive image's aspect ratio, as any aspect ratio changes would be removed when the image is resized to a fixed aspect ratio. To test this hypothesis, we tested stretching the fifteen images from Figure A.5. We tested stretching each image 30% thinner and 30% shorter. We found that stretching the images was highly effective at evading the filter, as all of the images stretched shorter evaded filtering as well as all of the thinner images except for a drawing of Liu Xiaobo and his wife (Figure A.5 Image #11). This suggests that the last hypothesis is incorrect.

To test hypotheses 1 through 4, we made the following corresponding predictions: (1) If images are proportionally resized based on their width, then adding extra space to their width would evade filtering but adding it to their height would not. (2) If images are proportionally resized based on their height, then adding extra space to their height would evade filtering. (3) If images are proportionally resized based on their largest dimension, then adding extra space to that dimension would evade filtering. (4) If images are proportionally resized based on their smallest dimension, then adding extra space to that dimension would evade filtering.

To test these predictions, we chose a set of ten filtered images, five such that their height is no more than $\frac{2}{3}$ of their width, which we call the *wide* images, and five such that their width is no more than $\frac{2}{3}$ of their height, which we call the *tall* images (see Figure A.10). We then modified each of the images by adding blank black space the size of 50% of their width to their left and right sides (see Figure A.11 for an example) and again by adding black space the size of 50% of their height to their top and bottom sides. We repeated these again except by using 200% of the respective dimensions.

We found that wide images with space added to their width and tall images with space added to their height were always filtered. This is consistent with hypothesis 4, that WeChat resizes based on an uploaded image's shortest dimension, as this hypothesis predicts that adding space in this matter will not change the scale of the original image contents after the image is resized. We also found that 4 out of 5 wide images with space added to their height and 3 out of 5 tall images with space added to their width evaded filtering, suggesting that this caused the uploaded image to be further downscaled compared to the corresponding one on the blacklist.

The results between adding 50% and 200% extra space were fairly consistent, with only one fewer tall image being filtered. This consistency is to be expected, since according to the shortest dimension hypothesis, adding extra space past when the image has already become square will not affect its scaling.

It is not clear why some images—two tall images with extra width and one wide image with extra height—were still filtered. It is possible that WeChat's filtering algorithm has some robustness to changes in scale. However, it is also possible that variants of these images with extra space or some other border or content added in these areas are also on the blacklist. For example, the only wide image with extra height to still be filtered is the famous and highly reproduced Tank Man photo taken during the Tiananmen Square protests of 1989 (see Figure A.10 Wide Image #4). A reverse Google Image search found that there are many images with similar spacing added to them already in circulation on the Internet. Nevertheless, adding height to wide images or width to tall images was generally an effective strategy for evading filtering while preserving the image's visual appearance.

3.2.5 Sliding window

In the previous section, we tested how we could evade WeChat's filtering by extending the canvases of sensitive images in different ways depending on their aspect ratios. In the instances extending the canvas did not evade the filter, such as by adding height to a tall image or width to a wide image, this suggested that WeChat's filter exhibits translational invariance, *i.e.*, the ability for WeChat's filter to find an image if its canvas has been extended.

Translational invariance only requires that the filter recognize the image when the extended space is blank or black. What if the added content is not blank? Can that allow us to evade the filter? In this section we are concerned with whether the algorithm is not simply translationally invariant but whether it can find an image inside of another image regardless of its surrounding contents.

In our testing we found that WeChat's server-side image compression would increase compression for larger images and that the resulting compression artifacts could cause uploaded images to evade filtering. We carefully designed our experiment to control for this. Taking our five wide and five tall images used in the previous section (see Figure A.10), we extended their canvases on each side in their largest dimension by $i \cdot n$, for a total of $2 \cdot i \cdot n$, for each i in $\{1, 2, \ldots, 5\}$, where n is the size of the largest dimension. We first extended their canvases with blackness. As many image operations such as thresholding and edge detection are sensitive to an image's distribution of pixel intensities, to control for this, we also extended their canvases with a duplicate copy of the image itself so that the distribution of pixel intensities is not affected (see Figure A.12 for an illustration). To account for WeChat's compression, for any image we generated, if it evades filtering, we download the image and crop out the extended canvas, restoring it to its original size. If this image still evades filtering when uploaded, then we conclude that this is from the additional compression artifacts and not necessarily from the contents of the extended canvas.

We found that images extended with their own duplicates evaded filtering after a sufficiently large number of duplicates were added, and none of these evasions could be explained by image compression (see Figure A.13 for full results). Conversely, in all but one test, images extended with blank canvases were either filtered or their evasion could be explained by image compression. These results suggest that, even when we add additional contents to an uploaded image such that its distribution of pixel intensities do not change, these contents affect the ability of WeChat to recognize the uploaded image as sensitive. This finding suggests that WeChat may not use a sliding window approach that ignores contents outside of that window to compare images. Instead, the images appear to be compared as a whole and that adding complex patterns outside of a blacklisted image's original canvas can evade filtering.

3.2.6 Summary of visual-based filtering findings

In this section we have described a number of characteristics of WeChat's visual-based filtering. Our understanding of some of its mechanics has informed multiple strategies for evading the filter. However, the entire algorithm used by the filter is still unclear and understanding more about how the algorithm works may inform additional evasion strategies.

Our finding that the filter exhibits translational invariance may be the most informative clue in understanding the complete algorithm used by the filter. The use of template matching [2] would explain translational invariance. However, it is typically used to find images inside of other images as opposed to image comparison in itself. Moreover, template matching typically finds matches in a sliding window, which is incompatible with our finding that adding complex content outside of the window can evade the filter.

Perceptual hashing is a technique to reduce an image to a hash such that similar images have either equal [20] or similar [15] hashes to facilitate efficient comparison. It is used by many social media companies such as Facebook, Microsoft, Twitter and YouTube [8] to filter illegal content. Spectral methods can be used to achieve a hash exhibiting translational invariance. The popular open source implementation pHash [15] computes a hash using the discrete cosine transform, which is not translationally invariant. However, an alternative spectral computation that would exhibit translational invariance would be to calculate the image's amplitude spectrum by computing the absolute magnitude of the discrete Fourier transform of the image, as translation only affects the phase, not the magnitude, of the image's frequencies [11, page 126]. The use of a hash based on this computation would be consistent with our findings, but more work is needed to test if this technique, in possible combination with other image processing algorithms, is used.

4 Conclusion

An effective image filter evasion strategy is one that modifies a sensitive image so that it (1) no longer resembles a blacklisted image to the filter but (2) still resembles a blacklisted image to people reading it. The two ways we present to evade WeChat's OCR filter, via its implementations of grayscale conversion and blob merging, meet both of these requirements, as the resulting text is still easily legible to people but baffles the filter.

Regarding evading the visual-based filter, we discovered multiple evasion techniques, each with tradeoffs. Since the filter has no semantic understanding of the image, simple transformations like mirroring the image evade filtering. However, these may reverse text or otherwise fail to preserve meaningful details of the image. In our tests, edges appeared to be important features to WeChat's filter, as blurring would quickly evade the filter. However, edges are perceptually important to people too, so blurring may be undesirable. Based on how the filter resizes images to a canonical size, we found the conditions under which one may extend the canvas with a black border and have the image evade filtering. This option generally preserves the important perceptual qualities of an image except for adding a black border to it in one or more dimensions. Finally, we found that if the border is not black but contains some sufficiently complex patterns, then it does not matter which side(s) the border is added to. Generally though, it is simpler to add a black (or other simple) border according to the conditions we identified based on how WeChat resizes images.

In this work we presented experiments uncovering implementation details of WeChat's image filter and multiple effective evasion strategies. While the focus of this work has been WeChat, due to common implementation details between image filter implementations, we hope that our methods will serve as a road map for future research studying image censorship on other platforms.

Acknowledgments

This research was supported by the Open Society Foundations. We would like to thank our reviewers for insightful feedback. Image testing data and source code is available at `https://github.com/citizenlab/chat-censorship/tree/master/wechat/image-filtering`.

References

[1] Bamman, D., O'Connor, B., and Smith, N. A. Censorship and deletion practices in Chinese social media. *First Monday 17*, 3 (2012).

[2] Brunelli, R. *Template Matching Techniques in Computer Vision: Theory and Practice*. John Wiley & Sons, 2009.

[3] Cairns, C., and Plantan, E. Why autocrats sometimes relax online censorship of sensitive issues: A case study of microblog discussion of air pollution in china. In *Midwest Political Science Association Annual Conference, Chicago, IL, April* (2016), vol. 8.

[4] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (2017), ACM, pp. 15–26.

[5] Crandall, J. R., Crete-Nishihata, M., Knockel, J., McKune, S., Senft, A., Tseng, D., and Wiseman, G. Chat program censorship and surveillance in China: Tracking TOM-Skype and Sina UC. *First Monday 18*, 7 (6 2013).

[6] Crete-Nishihata, M., Knockel, J., Miller, B., Ng, J. Q., Ruan, L., Tsui, L., and Xiong, R. Remebering Liu Xiaobo: Analyzing censorship of the death of Liu Xiaobo on WeChat and Weibo. Tech. rep., Citizen Lab, University of Toronto, 2017. Available at `https://citizenlab.ca/2017/07/analyzing-censorship-of-the-death-of-liu-xiaobo-on-wechat-and-weibo/`.

[7] Deng, I. Tencent's WeChat hits 1 billion milestone as Lunar New Year boosts monthly active users. Available at `http://www.scmp.com/tech/apps-gaming/article/2135690/tencents-wechat-hits-1-billion-milestone-lunar-new-year-boost`, 2018.

[8] Facebook. Partnering to Help Curb Spread of Online Terrorist Content. Available at `https://newsroom.fb.com/news/2016/12/partnering-to-help-curb-spread-of-online-terrorist-content/`, 2016.

[9] Fu, K.-w., Chan, C.-h., and Chau, M. Assessing Censorship on Microblogs in China: Discriminatory Keyword Analysis and the Real-Name Registration Policy. *IEEE Internet Computing 17*, 3 (2013), 42–50.

[10] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* (2014).

[11] Hall, E. *Computer Image Processing and Recognition*. Elsevier, 1979.

[12] Hardy, S. Asia Chats: Investigating Regionally-based Keyword Censorship in LINE. Tech. rep., Citizen Lab, University of Toronto, 2013. Available at `https://citizenlab.ca/2013/11/asia-chats-investigating-regionally-based-keyword-censorship-line/`.

[13] Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box Adversarial Attacks with Limited Queries and Information. *arXiv preprint arXiv:1804.08598* (2018).

[14] King, G., Pan, J., and Roberts, M. How Censorship in China Allows Government Criticism but Silences Collective Expression. *American Political Science Review 107*, 2 (2013), 326–343.

[15] Klinger, E., and Starkweather, D. pHash: The open source perceptual hash library. Available at `http://phash.org/`, 2008.

[16] Knight, W. Twitter's Artificial Intelligence Knows What's Happening in Live Video Clips. Available at `https://www.technologyreview.com/s/601284/twitters-artificial-intelligence-knows-whats-happening-in-live-video-clips/`.

[17] Knockel, J., Crandall, J. R., and Saia, J. Three researchers, five conjectures: An empirical analysis of TOM-Skype censorship and surveillance. In *FOCI'11 (USENIX Workshop on Free and Open Communications on the Internet)* (2011).

[18] Knockel, J., Crete-Nishihata, M., Ng, J. Q., Senft, A., and Crandall, J. R. Every Rose Has Its Thorn: Censorship and Surveillance on Social Video Platforms in China. In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 15)* (2015).

[19] Knockel, J., Ruan, L., and Crete-Nishihata, M. Measuring Decentralization of Chinese Keyword Censorship via Mobile Games. In *7th USENIX Workshop on Free and Open Communications on the Internet (FOCI 17)* (2017).

[20] Kozat, S. S., Venkatesan, R., and Mihçak, M. K. Robust perceptual image hashing via matrix invariants. In *Image Processing, 2004. ICIP'04. 2004 International Conference on* (2004), vol. 5, IEEE, pp. 3443–3446.

[21] MacKinnon, R. China's "Networked Authoritarianism". *Journal of Democracy 22*, 2 (2011), 32–46.

[22] MathWorks. Convert RGB image or colormap to grayscale. Available at `https://www.mathworks.com/help/matlab/ref/rgb2gray.html`.

[23] Narodytska, N., and Kasiviswanathan, S. P. Simple Black-Box Adversarial Perturbations for Deep Networks. *arXiv preprint arXiv:1612.06299* (2016).

[24] Ng, J. Q. Blocked on Weibo – Search result logs and full list of banned words. Available at `http://blockedonweibo.tumblr.com/post/12729333782/search-result-logs-and-full-list-of-banned-words`, 2012.

[25] Ng, J. Q. Tracking Censorship on WeChat's Public Accounts Platform. Tech. rep., Citizen Lab, University of Toronto, 2015. Available at `https://citizenlab.org/2015/07/tracking-censorship-on-wechat-public-accounts-platform/`.

[26] Open Source Computer Vision library. Miscellaneous Image Transformations. Available at `https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html`.

[27] Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE transactions on systems, man, and cybernetics 9*, 1 (1979), 62–66.

[28] Reporters Without Borders. China: Journey to the heart of Internet censorship. Tech. rep., Reporters Without Borders, 2007. Available at `http://www.rsf.org/IMG/pdf/Voyage_au_coeur_de_la_censure_GB.pdf`.

[29] Robinson, S. Filtering inappropriate content with the Cloud Vision API. Available at `https://cloud.google.com/blog/big-data/2016/08/filtering-inappropriate-content-with-the-cloud-vision-api`.

[30] Ruan, L., Knockel, J., and Crete-Nishihata, M. We (Can't) Chat: "709 Crackdown" Discussions Blocked on Weibo and WeChat. Tech. rep., Citizen Lab, University of Toronto, 2017. Available at `https://citizenlab.ca/2017/04/we-cant-chat-709-crackdown-discussions-blocked-on-weibo-and-wechat/`.

[31] Ruan, L., Knockel, J., Ng, J. Q., and Crete-Nishihata, M. One App, Two Systems: How WeChat uses one censorship policy in China and another internationally. Tech. rep., Citizen Lab, University of Toronto, 2016. Available at `https://citizenlab.ca/2016/11/wechat-china-censorship-one-app-two-systems/`.

[32] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[33] Tencent. "微信"影响力报告:用数据读懂微信五大业务. Available at `http://tech.qq.com/a/20160321/030364.htm`, 2016.

[34] The Economist. WeChat's world. Available at `https://www.economist.com/business/2016/08/06/wechats-world`, 2016.

[35] Ullrich, J. B. Why Does Emperor Xi Dislike Winnie the Pooh and Scrambled Eggs? Available at `https://isc.sans.edu/forums/diary/Why+Does+Emperor+Xi+Dislike+Winnie+the+Pooh+and+Scrambled+Eggs/23395/`.

[36] Van Gumster, J., and Shimonski, R. *GIMP Bible*, vol. 616. John Wiley and Sons, 2011.

[37] Yahoo. Open nsfw model. Available at `https://github.com/yahoo/open_nsfw`.

[38] Zhu, T., Phipps, D., Pridgen, A., Crandall, J. R., and Wallach, D. S. The Velocity of Censorship: High-fidelity Detection of Microblog Post Deletions. In *Proceedings of the 22nd USENIX Conference on Security* (2013), USENIX Association, pp. 227–240.

[39] 陈波. 图片检测系统及方法. Available at `https://encrypted.google.com/patents/CN101303734A?cl=tr`, Nov. 12 2008. CN Patent App. CN 200,810,125,798.

A Supplementary material

This section includes supplementary visualizations and data. These figures are referenced in the paper to help illustrate different image transformations and to provide the complete results of our experiments.

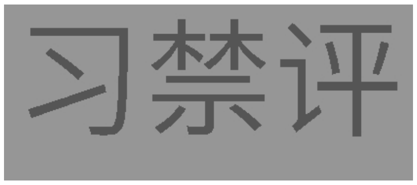
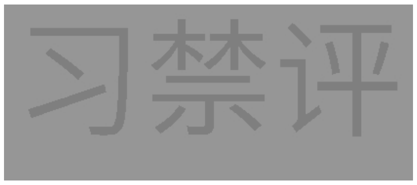
Algorithm	Result
Original	
Average	
Lightness	
Luminosity	

Figure A.1: An image with green text and a background color of gray with the same shade as the text according to the luminosity formula for grayscale and how the text would appear to an OCR algorithm according to three different grayscale algorithms. If the OCR algorithm uses the same grayscale algorithm that we used to determine the intensity of the gray background, then the text effectively becomes invisible to the algorithm.[2]

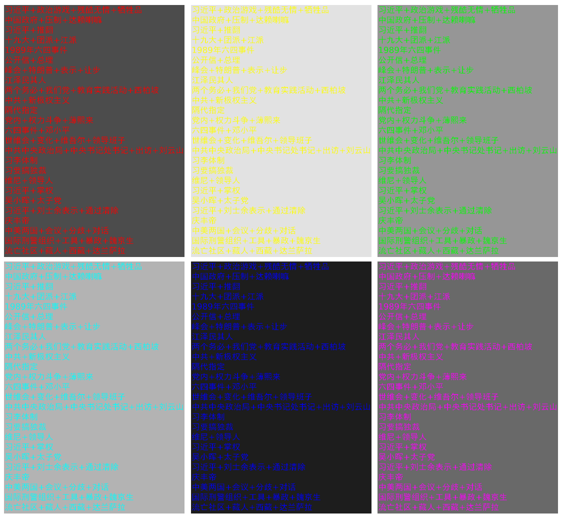


Figure A.2: Each of the six colors of text tested. Here the background color of each of the above images was chosen according to the luminosity of the text's color.[2]

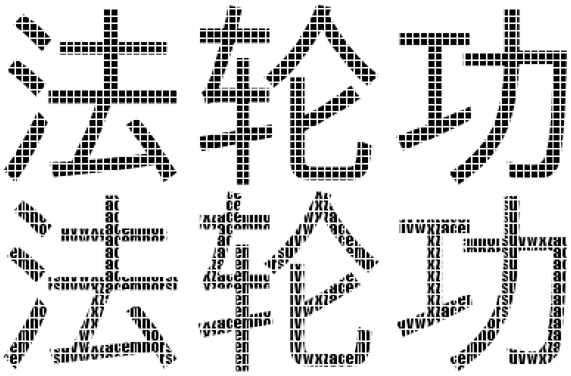


Figure A.3: Text patterned using squares and letters.

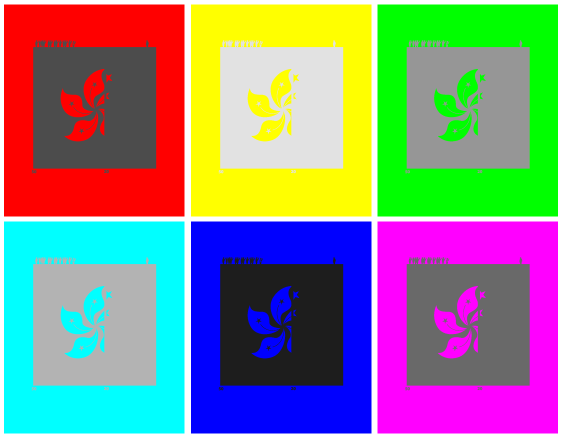


Figure A.4: Each of the six colors tested. Here the intensity of the gray background of each image was chosen according to the luminosity of the foreground color.[2]

[2]The figures in this section are generally intended to be displayed in color. If displayed in grayscale, then the image features in the color examples in this figure may disappear depending on how the display or printer converts color to grayscale, and the color examples may appear as solid gray rectangles. This is not unlike how these image's features are hidden from WeChat's filter when it converts them to grayscale.

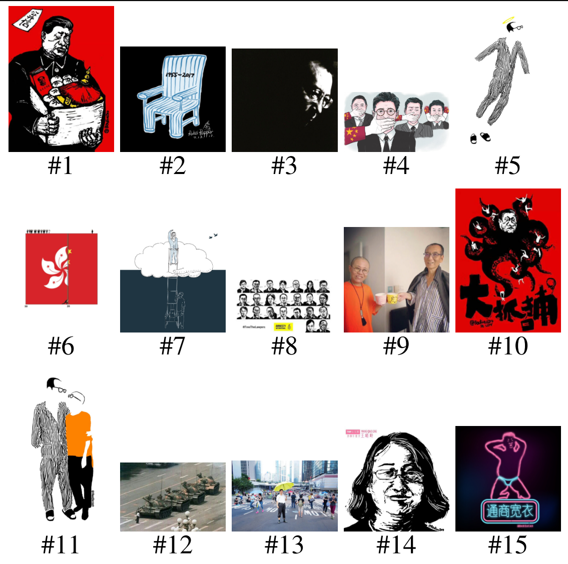


Figure A.5: The fifteen images we used for testing.

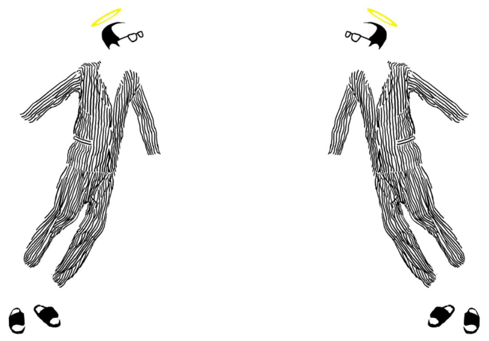


Figure A.6: Left, an image of Liu Xiaobo. Right, the mirrored image. Despite both images showing a depiction of the deceased Liu Xiaobo, only the original image on the left is filtered.

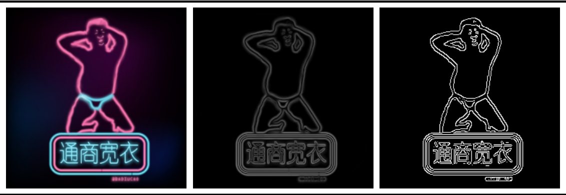


Figure A.7: Two kinds of edge detection. Left, the original image. Center, the image with a Sobel filter applied. Right, the Canny edge detection algorithm.

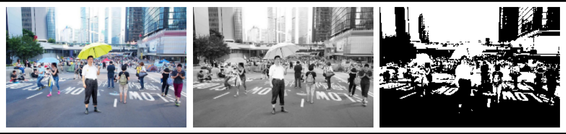


Figure A.8: Left, the original image. Center, the image in grayscale. Right, the image thresholded according to Otsu's method. All three images are filtered.

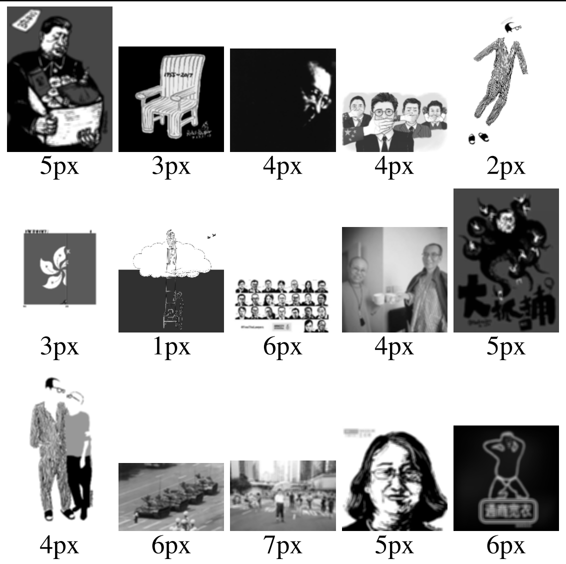


Figure A.9: The largest normalized box filter kernel size that can be applied to each image while still being filtered.

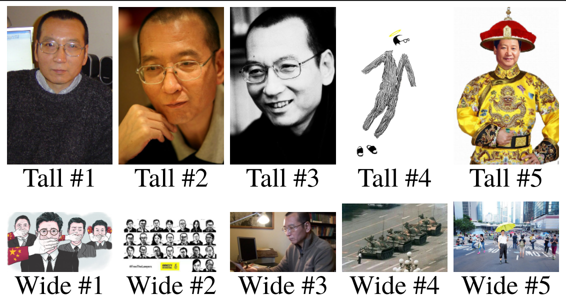


Figure A.10: The five *tall* and the five *wide* images we used for testing.

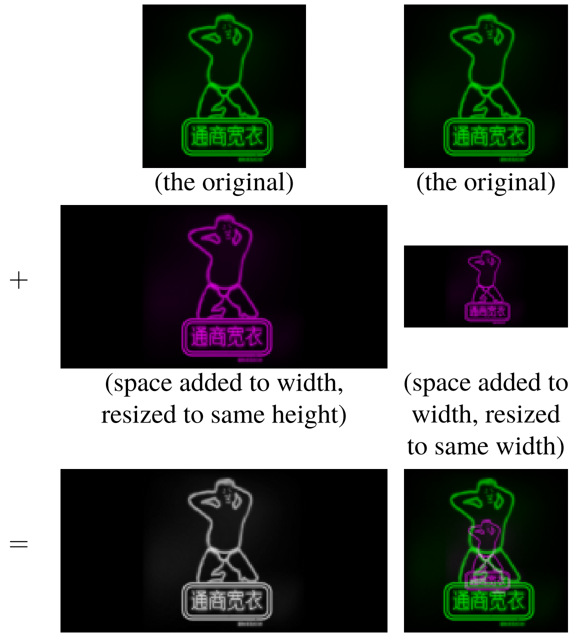

(the original)
(the original)
+
(space added to width, resized to same height)
(space added to width, resized to same width)
=

Figure A.11: Two different ways of resizing an image after extra space is added to its width. If resizing by its height (hypothesis 2) or by its shortest dimension (hypothesis 4), the scale of the image's contents are unchanged with respect to the original and there is complete overlap (white). If resizing by its width (hypothesis 1) or by its largest dimension (hypothesis 3), the image's original contents become smaller and do not overlap well with the original.

	Blank					Duplicated				
$i =$	1	2	3	4	5	1	2	3	4	5
	Y	C	Y	Y	Y	Y	Y	Y	Y	Y
	Y	C	C	Y	Y	Y	Y	Y	Y	Y
	C	C	C	C	C	Y	Y	Y	Y	Y
	Y	Y	Y	Y	Y	Y	Y	Y	N	N
	Y	Y	Y	Y	Y	Y	Y	N	N	N
	Y	Y	Y	Y	Y	Y	Y	N	N	N
	Y	C	C	Y	C	Y	Y	Y	N	N
	Y	Y	N	Y	Y	Y	Y	N	N	N
	Y	Y	Y	Y	Y	Y	Y	N	N	N
	Y	Y	Y	Y	Y	Y	Y	N	N	N

Figure A.13: After testing a wide and tall image by either extending it by i-many blank canvases or by i-many image duplicates, was it still filtered? Y = Yes, N = No, C = No due to compression artifacts. With one exception, all images extended with blank canvases that evaded filtering did so due to compression artifacts, whereas when extending an image with duplicates of itself, none of the filtering evasion can be explained by compression artifacts.

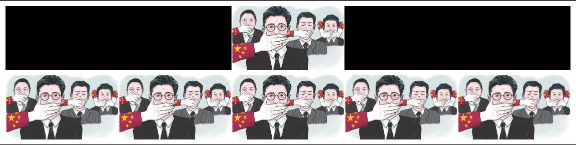


Figure A.12: Above, an image extended with $i = 2$ blank canvases to the left and right. Below, an image extended with $i = 2$ duplicates of itself.