CLOAK: Computer Learning for Obfuscating Automobile Knowledge

Sean Chen
Northwestern University
seanchen2024@u.northwestern.edu

Ryan Newkirk
Northwestern University
ryannewkirk2024@u.northwestern.edu

Jeffrey Wu
Northwestern University
jeffreywu2024@u.northwestern.edu

Bill Yin
Northwestern University
billyin2024@u.northwestern.edu

Daniel Zhao
Northwestern University
danielzhao2024@u.northwestern.edu

Figure 1: Obfuscated image of a car's license plate with CLOAK

ABSTRACT

As photography becomes easier to access and social media becomes more popular, sharing images has become effortless. Consequently, many street-level photos now openly reveal vehicle license plates, carrying unique personal information. Through use of easily accessible online tools, these license plates can be used to easily discover the plate owner's identity and location. As digital image sharing becomes increasingly prevalent, it's become increasingly important to protect the privacy of people, and preventing unauthorized access to personal license plate information would greatly assist in the protection of privacy. To address these issues, we've developed a tool to automatically detect license plates in an image and apply an appropriate amount of blur to the license plate to protect the drivers' privacy while maintaining image utility. This preserves the image's quality while also concealing any potentially identifying information.

1 INTRODUCTION

In recent years, the barrier to taking photographs has fallen significantly as high quality digital cameras, in the form of smartphones, are being considered a daily necessity by billions. Additionally, the ability to share these high resolution images online with millions of viewers has become largely trivial, with billions of photos being uploaded online each day. As the prevalence of online photography increases, as well as high-speed access to the internet, it's become increasingly important to maintain the privacy of the subjects of photography. While it may be considered the responsibility of the uploader to make sure their photo maintains the privacy of its subjects, many users fail to consider privacy in their social media posts. Therefore, the responsibility to maintain the privacy of people involved in a photograph may fall onto the services where photos are uploaded.

Of the photos being uploaded, a number of them are street side photos. Being street side, there is a high likelihood that a car may be captured containing personally identifiable information from the license plate. This presents a potential privacy risk, as there are many easily accessible online tools which can be used to lookup a license plate's information, which would leak information about the driver and owner. Additionally, license plates can be re-identified to track the vehicle's position over time, which also presents a significant privacy risk [1].

Therefore, our plan was to develop a tool that can take an uploaded image and hide the license plate, maintaining the privacy of the user. In doing so, we aimed to align our tool with the principles of usable security, particularly by implementing license plate obfuscation methods that minimize user effort in safeguarding private data within their own images [2].

However, it's also important to maintain the utility of the image. While a fully censored and blurred image would be private for the subjects of the photo, it's obviously unusable. Therefore, we wish to maintain as much of the structure of the image as possible with minor visual artifacts. This leads us to the goal of ensuring that the license plates are illegible, while maintaining the quality of the image by not altering the image's license plate too greatly. Therefore,

we wish to maximize the amount of privacy, while minimizing the level of utility loss in the image.

1.1 Research Goals

Our main research goals for this software are:

(1) Obfuscate license plates
(2) Maximize driver privacy
(3) Maintain image utility

In short, the solution we develop should be capable of automatically detecting license plates, and obfuscating them to a satisfactory degree. The obfuscated license plate should be recognizable as a license plate, but should be unreadable to both automatic systems and human readers. This should be accomplished while maintaining the overall image's quality.

1.2 Background and Related Works

Much research has been done on approaching personal privacy preservation from the context of surveillance [3–5], and research specifically in license plate de-identification began during the emergence of convolutional neural networks (CNNs). Ling et al. (2011) explore the principle of using inhomogeneous principal component blur (IPCB) to preserve license plate privacy, and it stands out in its adaptive approach of blurring pixels such that there is minimal damage to image quality and utility [6]. This work provided a foundation for subsequent advancements in blurring technology and introduced the importance of balancing privacy protection with image quality preservation.

In subsequent years, researchers have leveraged image obfuscation to enhance privacy with new deep learning approaches. One important shift is in object detection, allowing for real-time detection of objects in images. Research by Redmon et al. (2016) introduced the You Only Look Once (YOLO) model as a convolutional neural network based model that can process images in a single pass. This faster and more accurate method of identifying objects could lead to better object detection methods, and development on this technology has continued to push state of the art benchmarks in object detection as one of the fastest and most accurate object detection models [7].

In recent years, significant research has been devoted to the identification of license plate information [8, 9] and even the deanonymization of blurred images using artificial neural networks [10]. Following these advancements, many services have emerged specifically for people to find license plate data, such as online look-up services like *FaxVin* and *FindByPlate*. Part of the goal of our work is to address the recent gaps in privacy tools and research surrounding license plate privacy. We also aim to develop new methodologies of protecting the privacy of vehicles and drivers while taking advantage of new machine learning methods.

2 DATASET AND METHODOLOGY

2.1 Dataset

The dataset used is the Car License Plate Detection dataset hosted on Kaggle [1]. This dataset contains 433 images with bounding box annotations of the car license plates within the image. Annotations are provided in the PASCAL VOC format (Figure 2).

[1]https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

```
<annotation>
<folder>images</folder>
<filename>Cars0.png</filename>
<size>
      <width>500</width>
      <height>268</height>
      <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
      <name>licence</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <occluded>0</occluded>
      <difficult>0</difficult>
      <bndbox>
            <xmin>226</xmin>
            <ymin>125</ymin>
            <xmax>419</xmax>
            <ymax>173</ymax>
      </bndbox>
</object>
</annotation>
```

Figure 2: PASCAL VOC format xml

2.2 Models Used

2.2.1 Convolutional Neural Network (CNN) Model. As part of an early effort to develop this tool with customized models, a classification CNN model was attempted to recognize license plates in an image.

The early model consisted of the following layers, which did not yield a desirable and usable model.

```
# Create CNN model
class ConvNeuralNet(nn.Module):
    def __init__(self):
        super(ConvNeuralNet, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3,
             out_channels=16, kernel_size=3, stride=1,
             padding=1)
        self.conv2 = nn.Conv2d(in_channels=16,
             out_channels=32, kernel_size=3, stride=1,
             padding=1)

        self.fc1 = nn.Linear(in_features=32 * 150 * 150,
             out_features=200)
        self.fc2 = nn.Linear(200, 4)

    def forward(self,x):
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
```

```
        x = F.max_pool2d(F.relu(self.conv2(x)), (2, 2))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

The primary issue with this approach was the model that we constructed was structured for classification tasks and not for object recognition tasks, which caused the loss of the model to be anomalous. After some experimentation, this idea was discarded and replaced with pre-trained models.

2.2.2 PyTesseract Model. In order to extract text from the processed license plates, the Pytesseract library from Python was used. Pytesseract was used as part of the pipeline to evaluate the success of blurring on the extracted license plate images, specifically to assess if the images still contained machine recognizable text.

For this tool, the following configuration was used.

```
    # Use Tesseract OCR to extract text
    text = pytesseract.image_to_string(gray, config='--psm
        7 --oem 1')
```

When initializing Pytesseract, the detailed configuration in the config parameter was selected to extract the most amount of characters within our license plates. The procedure consisted of optimization through trial and error.

2.2.3 YOLOv5 Model. As another license plate detection method, a pre-trained model called YOLOv5 [7], developed by keremberke, was used to computationally extract the location of license plates. The parameters used were slightly adjusted from the default parameters specified in their HuggingFace model documentation to only allow a maximum of 1 detection per image. This allows the current tool to focus on a single license plate in an image, but this could be extended to detect multiple license plates in future developments.

```
    # set model parameters
    model.conf = 0.4 # NMS confidence threshold
    model.iou = 0.45 # NMS IoU threshold
    model.agnostic = False # NMS class-agnostic
    model.multi_label = False # NMS multiple labels per box
    model.max_det = 1 # maximum number of detections per image
```

2.3 Tool Procedure and Implementation

The tool we developed has 5 major steps in its procedure after a user inputs an image:

(1) Identify the bounding box of a license plate with a license plate detection model
(2) Crop out the license plate with the identified bounding box
(3) Apply levels of Gaussian blurring to the cropped license plate image
(4) Determine optimal blur level with a text recognition model
(5) Insert the optimally blurred license plate back into the original image

To identify the bounding box of a license plate in the inputted image, we utilize the pre-trained YOLOv5 license plate detection model. The outputted bounding box from this model provides the x_min, y_min, x_max, and y_max values of the image section that appears to be a license plate (Figure 3). Using the bounding box of the license plates, the tool extracts the cropped image of the license plate (Figure 4).

Figure 3: YOLOv5 model bounding box of license plate

Figure 4: Cropped license plate from original car image

After retrieving the cropped license plate image, various levels of Gaussian blurring is applied to this image (Figure 5). The choice of Gaussian blurring is because it is a common and effective image blurring technique that can't be stably reversed according to Hummel et al. (1986) [11]. In the experiments, 10 Gaussian blur levels were applied to each image where blur level 0 is the most amount of blurring and blur level 9 is no blurring. To create each blur level, the kernel size of the Gaussian blurring method was varied. The calculation of the blur levels is shown below:

$$kernel\_size\_unrounded = \left[5,\ \frac{W}{3} * \left(1 - \frac{b}{B-1}\right)\right] \quad (1)$$

where W is the pixel width of the cropped license plate image, b is the blur level, and B is the total number of blur levels. $B = 10$ is used for the 10 blur levels. Additionally, since the PyTorch gaussian_blur method only accepts odd values for the kernel dimensions (so there is a single center pixel), the value of $\frac{W}{3} * \left(1 - \frac{b}{B-1}\right)$ is rounded up to the next odd number. This means the kernel width will be incremented by 1 if it is even or remain the same if it is already odd.

Given the kernel size equation above, a size of 5 pixels is used for the kernel height. Meanwhile, the kernel width will vary between 0 (when $b = B - 1$) and $\frac{W}{3}$ (when $b = 0$). The equation also indicates that kernel width decreases in equal increments of $\frac{W}{3(B-1)}$ as the blur level increases (decreasing blur).

Once the various levels of Gaussian blurring are applied to the cropped license plate images, these images are inputted into the pre-trained PyTesseract text recognition model. Starting from the image with the most blur applied ($b = 0$), the images are inputted

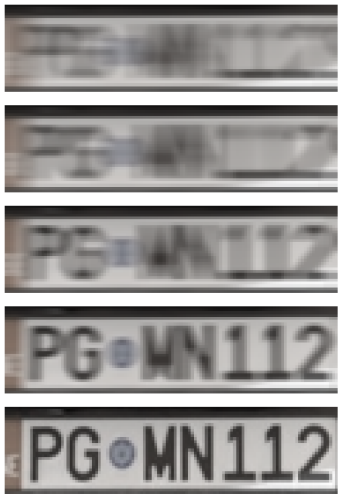

Figure 5: Amounts of Gaussian blur applied to license plate (blur levels 5 to 9 from top to bottom)

Figure 6: Final obfuscated image with blur level 5

into the PyTesseract model and the identified text is outputted. From here, there are 4 possible cases.

Case 1: If this model can only identify 1 or fewer alphanumeric characters in the image, the next image with less blurring (blur level incremented by 1) is attempted.

Case 2: If the model can identify 2 or more alphanumeric characters in the image, the tool stops attempting images with less blur. Instead, the tool selects the previously attempted blurred license plate image (blur level decremented by 1) since it was the last image where only 1 or fewer alphanumeric characters could be identified.

Case 3: If the model is immediately able to identify 2 or more alphanumeric characters at a blur level of 0 (maximum blur), then a blur level of 0 is selected.

Case 4: If the model still identifies 1 or fewer alphanumeric characters after reaching the image with a blur level of 9 (no blur), then the model selects the image with a blur level of 0 (maximum blur). The likely cause of this case is from a license plate image that is already very difficult to read or the text is oriented in a way that is difficult for the PyTesseract model to identify characters (e.g. text is slanted at odd angles that the PyTesseract model isn't trained to read). This means that the text could be either very difficult to read for people and the model, or the text could be very easy to read for people and difficult to read for the model. Since the tool wouldn't know which situation has occurred, we chose to have the model apply the maximum amount of blur. This decision ensures that user privacy is always protected and that utility is optimized whenever possible.

In these experiments, a threshold of 1 identified alphanumeric character was defined because license plates generally have more than 1 alphanumeric characters. However, this threshold could be modified in future experimentation to vary the balance between image privacy and image utility.

Finally, the selected blurred license plate image is inserted back into the original image and the tool outputs the updated image (Figure 6).

2.4 Technologies Used

To implement our methodology, we utilized Python libraries and models such as Pytorch, CV2, YOLOv5, and PyTesseract.

3 EVALUATION, EXPERIMENTS, RESULTS

3.1 Evaluation and Experiments

To evaluate tool efficacy, we conducted an experiment on a random sample of 50 car images from the Car License Plate Detection dataset. In this experiment, we inputted the 50 images into the tool and recorded the resulting blur levels used for each image. These blur levels were then plotted on a histogram to show the distribution of blur levels used from 0 through 9. Additionally, the average blur level across the sample of car images was calculated. With this distribution and average, we can assess how well the tool varies its blur level for different images.

For the tool to consider an image sufficiently blurred, the image must be illegible to the text recognition model, which is defined as only being capable of identifying 1 or fewer alphanumeric characters. More details about the methodology and blur level calculation can be found in the Tool Procedure and Implementation section above.

3.2 Results

From the experiment on 50 sample images, the average the blur level was 3.02. However, as seen in Figure 7, it is a bimodal distribution of the blur levels, one centered around 0 (strongest blur) and one centered around 6 (medium blur). It's reasonable to expect all images to require degree of blurring, as each image in the sample set contains a detectable license plate. Therefore, the overall mean of the experiment is not unexpected. However, the bimodal distribution was unexpected, and may be evidence for a discrepancy in the blurring levels required by different images of license plates.

Images with a more obscure license plate that is not directly facing the camera may require less blurring as they are inherently harder to identify and read from, while those that are more conspicuous are likely to need more. Therefore we believe that level of blurring is not a universal value but rather context-dependent.

Examples of the tool's blur results can be seen in Figure 8.

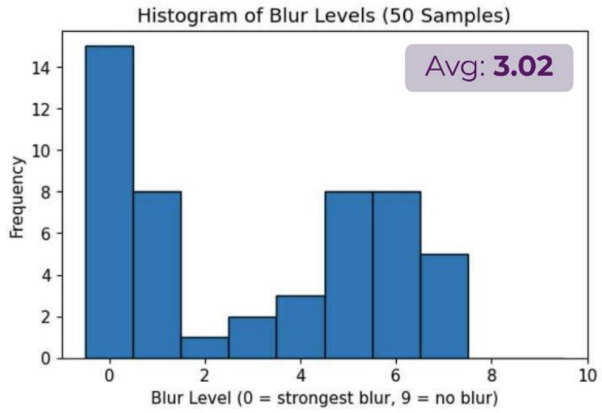

Figure 7: Blur level distribution

Figure 8: Original image input (left) and resulting blurred image output (right) of tool

4 DISCUSSION

4.1 Implications and Uses

With billions of images and thousands of hours of videos [12] of the world around us shared daily, the need for privacy rises for the many sources of possible personal information. Our license plate privatization process is invaluable in many domains, especially in the context of images captured outdoors on roads or streets. The versatility extends to automating the removal of license plates in publicly released images and videos shared by law enforcement, news, and journalists, as well as protecting privacy of vehicles and people on social media posts. This tool also allows images captured in public spaces to preserve the visual utility and aesthetic essence sought by the photographer while safeguarding the privacy of any vehicle information incidentally present in the images.

The YOLOv5 model's real-time object detection capabilities means the same process of image obfuscation we've created can be used in real-time applications as well. For example, it can facilitate the real-time monitoring of traffic and surveillance to aid in efficient traffic management/analysis without compromising individual privacy or protect the privacy of vehicle/location information in live streams.

Our technology can also be adapted to different domains. While the process focuses on car license plate detection, there are many other sources of sensitive information that can appear in images. This includes street signs and address plaques, which can leak information of the photographer's location as well as draw unwanted eyes on the people who might live in the background.

4.2 Challenges

At the start of our project we attempted to create our own CNN model to recognize license plates from photos. However, we saw two large issues: the images in the dataset were not uniform shape and the core of our method is not classification. We needed the model to perform object detection rather than image classification, which was an area we were not familiar with.

We solved the uniform shape issue by manually padding the images with black borders to the right and bottom of our images. This would allow us to maintain the same x and y values for the borders given in the dataset, as well as not modifying the image in any way (it is trivial to crop out the borders after finishing). After consulting with our peer mentor about our challenges, we agreed that this was a good method to standardize image sizes.

Our peer mentor also suggested the use of a pre-trained model to detect and locate the license plate. As we were already looking into pre-trained models, such as YOLOv5, we agreed that using it would be best. Together, we developed the idea to put the license plates back into a license plate or text recognition model in order to evaluate the quality of the privatization. This inspired us to make the varied Gaussian blur levels (Figure 5) and bring in a text recognition model to evaluate them.

4.3 Limitations

The final model has a few key limitations. We focused on just picking out one license plate from the image and privatizing that. The one license plate would be the highest value detected by YOLO. However, this means that in a picture with multiple license plates, only one will be blurred. This can be adjusted by the YOLOv5 model parameters and a few modifications to the current tool.

Also, the model struggles with slanted license plates. This is due to the usage of text recognition to judge how blurred the image should be. The text recognition model often fails to identify text from slanted or sideways images, so when the pipeline sees that the PyTessaract model has not outputted any text, it may choose a weaker blur than needed. Additionally, both the bounding boxes given by the dataset (Figure 2) and the bounding boxes found by YOLO are all upright oriented rectangles. This is not ideal since slanted or sideways license plates will include areas within the box that are not part of the actual license plate (Figure 9). This can cause areas not part of the license plate to be blurred, which hampers image utility. To address these limitations for slanted license plates, a better text recognition model and a more accurate license plate detection model (adjusts bounding box shape to match the shape of the license plate) may be used or created.

Figure 9: Bounding box of slanted license plate

Finally, we struggled with creating evaluation metrics that were robust. We ended up using a soft metric where we use another model to evaluate the correct amount of blur. However, as mentioned before, this means that the output is also dependant on the ability and accuracy of these separate models. At the moment, there is no robust accuracy metric to evaluate the output of the tool.

4.4 Future Work

Currently, the tool's blurring procedure is not streamlined to the point that can be released for public use. To reach this point, having this application running on a server with a working frontend is a useful extension so anyone can use the tool.

Additionally, creating a better and more robust metric to evaluate the privacy level of the modified image could engender a more robust blurring model. Currently, the method of determining blur level includes a naïve PyTessseract text extraction model, which is usable but could be improved to be more robust in order to better optimize the utility of the image while maintaining license plate privacy. A better text extraction model would be an auspicious step to have a better tool. This extension could potentially give rise to new blurring methods that can be compared using a universal and reliable blur level metric.

In terms of data, having a larger dataset can be helpful to eliminate the bi-modal distribution seen in the results section, granting a more in-depth understanding on what blur is necessary for each image.

Finally, as mentioned previously, our license plate obfuscation process can be extended to more applications such as street signs, home addresses, and many more privacy "breaches" throughout everyday photography. The tool can be modified to also apply in these domains by finding the corresponding image datasets and building a more generalized tool.

5 CONCLUSION

In this paper, we introduced CLOAK, a solution for maintaining the privacy of drivers by obfuscating vehicle license plates in images. We successfully developed a tool that can automatically detect the location of a license plate and hide it, while maintaining maximum utility of the image. On an image being loaded, our solution uses a pre-trained YOLOv5 model to automatically detect and locate a license plate, which is then Gaussian blurred using various kernel sizes. Our experimental results demonstrate the effectiveness of our tool in creating a guarantee of privacy relative to the PyTesseract model's capability of reading license plate text, such that the levels of blur can be dynamically adjusted to ensure privacy without sacrificing image quality.

CLOAK's applications extend beyond license plates to safeguarding other sensitive information in images, presenting a potential solution for privacy protections in various scenarios. As image sharing becomes more prevalent, CLOAK addresses the critical need to balance privacy and utility, providing a practical tool for users and organizations alike.

REFERENCES

[1] Jing Gao, Lijun Sun, and Ming Cai. Quantifying privacy vulnerability of individual mobility traces: A case study of license plate recognition data. *Transportation Research Part C: Emerging Technologies*, 104:78–94, 2019.

[2] Peter Leo Gorski, Luigi Lo Iacono, and Matthew Smith. Eight lightweight usable security principles for developers. *IEEE Security amp;amp; Privacy*, 21(1):20–26, 2023.

[3] Andrew Senior. Privacy protection in a video surveillance system. *Protecting Privacy in Video Surveillance*, page 35–47, 2009.

[4] M Upmanyu, A M Namboodiri, K Srinathan, and C V Jawahar. Efficient privacy preserving video surveillance. *2009 IEEE 12th International Conference on Computer Vision*, 2009.

[5] Thomas Winkler and Bernhard Rinner. A systematic approach towards user-centric privacy and security for smart camera networks. *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*, 2010.

[6] Liang Du and Haibin Ling. Preservative license plate de-identification for privacy protection. *2011 International Conference on Document Analysis and Recognition*, Sep 2011.

[7] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, Imyhxy, , Lorna, (Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, Tkianai, YxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - yolov5 sota realtime instance segmentation, 2022.

[8] Rohith Polishetty, Mehdi Roopaei, and Paul Rad. A next-generation secure cloud-based deep learning license plate recognition for smart cities. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, December 2016.

[9] Shan Du, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325, February 2013.

[10] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. Defeating image obfuscation with deep learning, 2016.

[11] Robert A. Hummel, B. Kimia, and Steven W. Zucker. Deblurring gaussian blur. *Computer Vision, Graphics, and Image Processing*, 38(1):66–80, April 1987.

[12] T.J. Thomson, Daniel Angus, and Paula Dootson. 3.2 billion images and 720,000 hours of video are shared online daily. can you sort real from fake?, Nov 2023.