SIMPLE TRANSPARENT ADVERSARIAL EXAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

There has been a rise in the use of Machine Learning as a Service (MLaaS) Vision APIs as they offer multiple services including pre-built models and algorithms, which otherwise take a huge amount of resources if built from scratch. As these APIs get deployed for high-stakes applications, it's very important that they are robust to different manipulations. Recent works have only focused on typical adversarial attacks when evaluating the robustness of vision APIs. We propose two new aspects of adversarial image generation methods and evaluate them on the robustness of Google Cloud Vision API's optical character recognition service and object detection APIs deployed in real-world settings such as sightengine.com, picpurify.com, Google Cloud Vision API, and Microsoft Azure's computer vision API. Specifically, we go beyond the conventional "small-noise" adversarial attacks and introduce *secret embedding* and *transparent adversarial examples* as a simpler way to evaluate robustness. These methods are so straightforward that even non-specialists can craft such attacks. As a result, they pose a serious threat where APIs are used for high-stakes applications. Our transparent adversarial examples successfully evade state-of-the-art object detection APIs such as Azure Cloud Vision (attack success rate 52%) and Google Cloud Vision (attack success rate 36%). 90% of the images have a secret embedded text that successfully fools the vision of time-limited humans but is detected by Google Cloud Vision API's optical character recognition. Complementing to current research, our results provide simple but unconventional methods on robustness evaluation.

1 INTRODUCTION

Deep neural networks have found to be vulnerable to adversarial attacks (Szegedy et al., 2014; Biggio et al., 2013; Goodfellow et al., 2015), where it is possible to add perturbations to the image that result in misclassification. These perturbations are such that the perturbed image still looks semantically similar to the original image to humans. These perturbed samples are called as adversarial examples. As neural networks get deployed for high-stakes applications, adversarial attacks pose huge security risks. The current methods for generating adversarial images require the attacker to have specific knowledge about the victim model (e.g. the input gradient used in white-box attack) or excessive number of prediction evaluations (e.g. gradient estimation or substitute model training with model queries in black-box attacks). They do not closely resemble the real-world attack scenarios where the attacker might not have machine learning and programming skills to carefully craft adversarial examples and the budget to carry out computationally expensive black-box attacks.

In this work, we argue that though the current approaches of generating adversarial examples are important and well studied with respect to a defined threat model (e.g. ℓ_p norm bounded perturbation), they do not provide a complete spectrum of robustness evaluation using semantically similar adversarial examples. Specifically, we propose new and straightforward manipulations for Optical Character Recognition (OCR) and object detection APIs under a new type of simple black-box attack called *Simple Transparent Adversarial Examples*, where anyone without any machine learning expertise can very easily create simple prediction-evasive and semantically similar examples and successfully fool deep neural networks powered vision APIs. They also do not need any computational power and so they are not computationally expensive contrary to the current adversarial examples generation methods. Therefore, our simple attack is more realistic in real-world settings considering the safety of publicly deployed AI systems and potentially more dangerous than the current black-box attacks. We hope that this work will motivate the researchers to also consider such simple attacks while evaluating the robustness of deep neural networks.

We also show that our attack is adversarial in two folds: (1) It's prediction evasion (i.e. the transparent adversarial examples evade object detection of the computer vision APIs) (2) The adversarial images generated by secret embedding approach carry information (text) that is detected by Google Cloud Vision API but evades the vision of time-limited humans. Figure 5 in A.9 briefly illustrates *Simple Transparent Adversarial Examples.*

2 Simple Transparent Adversarial Examples

The current methods of generating adversarial examples require the attacker to have a certain level of machine learning expertise (e.g. first-order or zeroth-order optimization for handling adversarial attacks with perturbation constraints) to craft these attacks (Szegedy et al., 2014; Goodfellow et al., 2015; Madry et al., 2019; Papernot et al., 2016; Brown et al., 2017; Carlini & Wagner, 2017; Papernot et al., 2017; Chen et al., 2017; Hosseini & Poovendran, 2018). Thus, only the attacker who has the required domain knowledge can attack deep neural networks. We propose a new type of simple attack that doesn't require the attacker to have any machine learning and programming knowledge to attack deep neural networks based cloud vision APIs. We talk about how our attack is different than some of the recently proposed simple manipulations in A.1

To demonstrate this attack, we focus on attacking machine learning as a service (MLaaS) cloud vision APIs such as sightengine.com[1], PicPurify[2], Google Cloud Vision API[3], and Microsoft Azure's computer vision API[4]. We query these APIs via the web interface available through the demo. We test our attack on two of the most popular services offered by these visions APIs: optical character recognition (by Google Cloud Vision API) and object detection. *Simple Transparent Adversarial Examples* have dual definitions in our work: (1) Examples that are straightforward to craft and don't need any machine learning and programming knowledge (i.e. simple and transparent to craft) and (2) Examples modified with transparent white patche and secret invisible text embedding.

2.1 Simple Transparent Adversarial Examples to Evade Object Detection

Recent black-box adversarial attack methods such as (Papernot et al., 2017; Chen et al., 2017) and adversarial attack methods for object detectors and Google Cloud Vision API such as (Liu et al., 2019; Li et al., 2019; Chen et al., 2019; Song et al., 2018; Hosseini et al., 2017; Goodman, 2020) are small noise adversarial attacks that require the attacker to have machine learning and programming knowledge to carefully design such attacks. We propose a simple attack method where the attacker can add transparent white patches to an image using any publicly available online transparency tool to successfully evade state-of-the-art object detection APIs or misclassify the labels. Since anyone even without any machine learning and programming knowledge can easily fool the publicly deployed APIs, these type of attacks pose a higher security risk than the current typical adversarial attacks.

Moreover, the current black-box adversarial attack methods (Papernot et al., 2017) require thousands of queries to design adversarial examples that can be very expensive. In our proposed attack, the attacker can successfully attack object detection APIs with just a few queries (See Table 1). Hence, we argue that our black-box attack is cheaper and query efficient than the majority of current black-box attack methods with a different modification (perturbation) constraint.

2.1.1 Attack creation

For any image x_0, we perturb x_0 by introducing a white transparent patch *p* with a modification constraint ϵ, such that the resultant modified image *x* is either misclassified or evades the classification entirely. The value of ϵ is such that the modified image still remains unambiguous and class-preserving

[1]`https://sightengine.com/detect-weapons-alcohol-drugs`

[2]`https://www.picpurify.com/demo-gun.html`

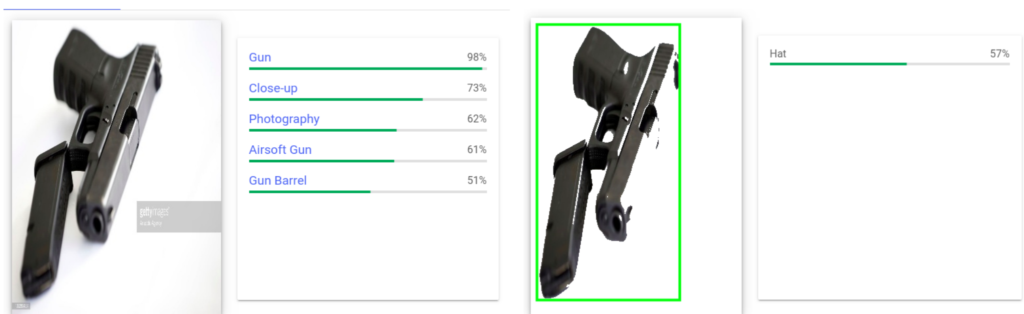
[3]`https://cloud.google.com/vision`

[4]`https://azure.microsoft.com/en-in/services/cognitive-services/computer-vision/`

to humans. We use publicly available onlinejpgtools.com's transparency maker tool [5] to introduce white transparent patches in the image. We use a readily available online tool to demonstrate that anyone can easily craft such examples to attack APIs deployed in high-stakes applications, hence these type of attacks pose a higher security risk than the typical adversarial attacks where the attacker is required to have machine learning expertise to design attacks. We use Kaggle's Weapons Dataset [6] to generate adversarial images and test them on weapon detection APIs (PicPurify and Sightengine.com) and general object detection APIs (Azure vision and Google Cloud Vision). For adding the transparent patch on the image, we follow two approaches: 1) We don't select the region(s) to be patched (i.e. we let the tool decide it) 2) We randomly select the region to be patched. We propose attack algorithms for the first approach in A.2.1 and for the second approach in A.2.2.

2.1.2 Experiments and Observations

We find that Simple Transparent Adversarial Examples successfully evade the weapon/object detection by sightengine.com, PicPurify, Google Cloud Vision API and Microsoft Azure's computer vision API. Table 1 in A.4.1 shows the performance of each API against these examples. Figure 1 illustrates this attack.



(a) Original image

(b) Modified image

Figure 1: (a) The original image that gets detected successfully by Google Cloud Vision API. (b) Modified image having 55 % transparency intensity and is misclassified as a **hat** by Google Cloud Vision API.

2.2 Simple Transparent Adversarial Examples for Optical Character Recognition (OCR)

We propose a new method called *secret embedding approach* to perturb images by embedding text in them. We call such images as *Simple Transparent Adversarial Examples* for OCR as they are straightforward to craft and do not require any machine learning or programming skills. We evaluate these images on Google Cloud Vision API's OCR feature[7]. We find that the images created by this method carry information (text) that is only recognizable by Google Cloud Vision API's OCR whereas it evades the vision of time-limited humans. In this sense, we go beyond the traditional definition of adversarial examples where they fool deep neural networks and not humans. The closest line of work to this is by (Elsayed et al., 2018), where the adversarial examples fool both computer vision and time-limited humans. We find that our examples also successfully evade the vision of both computer vision (i.e. Google Cloud Vision API) and time-limited humans at a specific font size and color (see A.7).

2.2.1 Attack creation

For an image x_0, we find an image x with an embedded text t, such that t evades the vision of time-limited humans but is detected by Google Cloud Vision API. To embed the text, we use an online tool imagecolorpicker.com [8] to find RGB value of the region where we plan to embed the text,

[5] https://onlinejpgtools.com/make-jpg-transparent
[6] https://www.kaggle.com/ar5p1edy/weapons-datasets
[7] https://cloud.google.com/vision/docs/ocr
[8] imagecolorpicker.com/en/

and then we embed the text using an online editing tool Photopea [9]. We use the publicly available online tools to show that such attacks can be easily desgined by anyone, even without any machine learning knowledge. We use images from Caltech 101 [10] and Caltech-256 [11] dataset for this attack. We propose a simple algorithm for *secret embedding approach* in A.3.

2.2.2 OBSERVATIONS

We find that in some cases there is a trade-off between the font size of the embedded text and the *RGB* difference. To elaborate, text with higher *RGB* difference and smaller font size ($\leq$15 px) or text with smaller *RGB* difference and relatively larger font size (>15 px) is favorable to craft these adversarial examples. However, smaller *RGB* difference and smaller font size are the ideal conditions. Figure 2 illustrates this attack. In our experiments we test 40 images. 90% of those images fooled the vision of time-limited humans and were recognized by Google Cloud Vision OCR. However, we also observe that it's possible to achieve near 100% evasion rate in the case of examples that fool both time-limited humans and OCR with secret text of very small font size size and *RGB* difference set to 0 (For example see A.7). The importance of this attack and it's possible risks and applications are highlighted in A.5 and A.6 in the appendix. A.3.1 shows the exact location in the image where the secret invisible text is embedded.

(a) Original image (b) Modified image (c) Original image (d) Modified image

Figure 2: (b) & (d) are modified images that fool time-limited humans but not Google Cloud Vision API (i.e. they are recognized by OCR). (b) has an embedded text "Hello World" of font size 11 px inside a rectangular region formed by the x & y coordinates (11, 167), (86, 167), (11, 195), (86, 195) and *RGB* difference 30 and (d) has an embedded text "Hello World" of font size 9 px and *RGB* difference 30 inside a rectangular region formed by the x & y coordinates (90, 62), (147, 62), (90, 82), (147, 82). This adversarial examples fool the vision of time-limited humans but not Google Cloud Vision's OCR.

3 CONCLUSION

In order to deploy deep neural networks in safety-critical areas and high-stakes applications, it's very important that the robustness evaluation is thorough, efficient, and covers the whole spectrum of semantically similar examples. Lately, there has been a huge surge in the amount of research in terms of creating adversarial attacks and defenses for deep neural networks, which has provided some great insights on evaluating the robustness. Though important, we argue that the current research in this field has not focused on simple unconventional methods on evaluating the robustness. This needs the attention because simple methods can be used any anyone to attack deep neural networks, whereas the current conventional methods can only be used by attacker having a machine learning expertise. Therefore, there's a high chance that attackers might adopt such simple and cheap methods to attack in the real-world and thus posing a huge security concerns. Specifically, we propose simple transparent adversarial examples and illustrate their novel insights to complement current adversarial example generation pipeline on several image-based APIs. We hope that our unconventional route to highlight such simple attacks will motivate the research in this field to also consider the serious threats posed by such simple attacks to build more inclusive and broad robust defenses for machine learning systems in real-world settings. We also propose some potential future directions in A.8.

[9] https://www.photopea.com/
[10] http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[11] https://authors.library.caltech.edu/7694/

REFERENCES

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. *Lecture Notes in Computer Science*, pp. 387–402, 2013. ISSN 1611-3349. doi: 10.1007/978-3-642-40994-3_25. URL http://dx.doi.org/10.1007/978-3-642-40994-3_25.

Tom Brown, Dandelion Mane, Aurko Roy, Martin Abadi, and Justin Gilmer. Adversarial patch. 2017. URL https://arxiv.org/pdf/1712.09665.pdf.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

Lu Chen and Wei Xu. Attacking optical character recognition (ocr) systems with adversarial watermarks, 2020.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. *Lecture Notes in Computer Science*, pp. 52–68, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-10925-7_4. URL http://dx.doi.org/10.1007/978-3-030-10925-7_4.

Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3910–3920. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7647-adversarial-examples-that-fool-both-computer-vision-and-time-limited-humans.pdf.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations, 2019. URL https://openreview.net/forum?id=BJfvknCqFQ.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Dou Goodman. Transferability of adversarial examples to attack cloud-based image classifier service, 2020.

Hossein Hosseini and Radha Poovendran. Semantic adversarial examples, 2018.

Hossein Hosseini, Baicen Xiao, and Radha Poovendran. Google's cloud vision api is not robust to noise, 2017.

Yuezun Li, Xiao Bian, Ming ching Chang, and Siwei Lyu. Exploring the vulnerability of single shot module in object detectors via imperceptible background patches, 2019.

Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pp. 372–387, 2016. doi: 10.1109/EuroSP.2016.36.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pp. 506–519, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349444. doi: 10.1145/3052973.3053009. URL `https://doi.org/10.1145/3052973.3053009`.

Congzheng Song and Vitaly Shmatikov. Fooling ocr systems with adversarial text images, 2018.

Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, August 2018. USENIX Association. URL `https://www.usenix.org/conference/woot18/presentation/eykholt`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6199`.

A Appendix

A.1 How are simple transparent adversarial examples different from the recently proposed simple transformations

Simple transformation such as translations and rotation (Engstrom et al., 2019) and the recent non-programmatic Typographic attack (Goh et al., 2021) have been also found to fool deep neural networks but they require machine learning knowledge to design adversarial perturbation (such as manually looking through the multimodal model's neurons to find text snippets for the typographic attack). Our attack methods are simple to craft without needing any kind of machine learning expertise.

A.2 Attack Algorithms for object detection

A.2.1 first approach

For the first approach where we don't select the region to be patched ourselves, we propose the following algorithm:

- **Step 1**: Using the transparency tool, we select some percentage of transparency to start with (10%) and apply transparent patch on the image.
- **Step 2**: We then query the object detection API with the transparently modified image.
- **Step 3**: If the modified image evades object detection, we then check for lesser transparency percentages (in this case < 10%) till we get the image with the minimum transparency such that it evades the object detection but remains unambiguous and class-preserving to humans.
- **Step 4**: If the transparently modified image doesn't evade the object detection, we keep increasing the transparency (in this case > 10%) and querying the API till we get a modified sample such that evades object detection but remains unambiguous and class-preserving to humans. We kept increasing the transparency by 5-10% for the successive images in our experiments.

A.2.2 second approach

For the second approach where we select specific region/color to be patched, we propose the following algorithm:

- **Step 1**: We select any random region in the image to apply the transparent patch and we get the RGB color values of that region using any publicly available tool.
- **Step 2**: We then select some percentage of transparency to start with (5%) and provide those RGB color values to the transparency tool so that it would apply patch on that region(s)

- **Step 3:** We then query the object detection API with the transparently modified image.
- **Step 4**: If the modified image evades object detection, we then check for lesser transparency percentages (in this case < 5%) till we get the image with the minimum transparency such that it evades the object detection but remains unambiguous and class-preserving to humans.
- **Step 5**: If the transparently modified image doesn't evade the object detection, we keep increasing the transparency (in this case > 5%) and querying the API till we get a modified sample such that evades object detection but remains unambiguous and class-preserving to humans. We kept increasing the transparency by 1-2% for the successive images in our experiments.

A.3 ATTACK ALGORITHM FOR OCR

- **Step 1**: Based on our experiments, we start with a font size of 15 px and a difference of 30 between the *RGB* values of the region where we want to embed the text and *RGB* values of font color of the text we want to embed. We call this difference as *RGB* difference. Further, we test the embedded image on Google Cloud Vision API's OCR service.
- **Step 2**: If OCR is able to recognize the embedded text, we further keep reducing the *RGB* difference or the font size or both up to the extent till which text can be recognized by OCR but evades the vision of time-limited humans.
- **Step 3**: If OCR is not able to recognize the text, we steadily increase the *RGB* difference keeping the font size as 15 px till the extent it gets recognized by OCR but evades the vision of time-limited humans.

A.3.1 LOCATION OF THE SECRET TEXT EMBEDDING

(a) Modified image with invisible text embedding (b) Modified image with black text embedding

Figure 3: (a) is the modified image that has secret invisible text of font size 9px and RGB values (160, 155, 157) in it and it evades the vision of time-limited humans but is detected by Google Cloud Vision API's OCR. To show exactly where it is placed, we change the font color to black and border it with a red rectangle that results in (b). The actual dimensions of the image are 648 px (width) and 430 px (height).

A.4 RESULTS AND OBSERVATIONS FOR SIMPLE TRANSPARENT ADVERSARIAL EXAMPLES FOR OBJECT DETECTION

A.4.1 PERFORMANCE OF APIS AGAINST SIMPLE TRANSPARENT ADVERSARIAL EXAMPLES

Table 1 summarizes the performance of all four APIs against simple transparent adversarial examples for object detection. Even though our attack has a lower attack success rate than the current norm-constrained black-box attack methods, our attack has more realistic scenario and very likely to occur in real-world settings since it's very straightforward and query efficient to carry out. Picpurify and Sightengine APIs look more robust because they are specifically designed to detect weapons in images, whereas Azure Cloud Vision and Google Cloud Vision APIs are general object detection

Table 1: Performance of APIs against Simple Transparent Adversarial Examples

API	ASR (%)	Samples Tested	Avg ϵ_1	Queries for ϵ_1	Avg ϵ_2	Queries for ϵ_2
Azure Cloud Vision	52%	50	39.37%	4	6.87%	3
Google Cloud Vision	36%	50	47.08%	5	6.71%	4
Sightengine	9%	50	43.25%	4	6%	2
Picpurify	2%	50	30%	3	NA	NA

Table 1: ASR denotes the attack success rate. Samples Tested show the number of modified images tested. Avg ϵ_1 refers to the average modification constraint (i.e. transparency percentage) when we don't select any region ourselves to be patched and Avg ϵ_2 refers to the average modification constraint when we select region(s) ourselves to be patched. Queries for ϵ_1 and Queries for ϵ_2 denote the average number of queries requires to fool the APIs when the modification constraints are ϵ_1 and ϵ_2. NA indicates that the modified images did not evade the object detection of that specific API.

APIs that detect different types of objects in an image. We also observe that in the cases where attack is not successful, it does significantly degrade the accuracy.

A.4.2 Limitations

One of the limitations of this attack is that if the object (for example a gun) to be patched in the image is very small than the overall size of the image, it might get difficult to apply a transparent patch on it. Even if the patch is applied, the object might look totally ambiguous to humans. Another limitation is that in the case where we select the region to be patched, sometimes we might need to try different regions in the object to see which one optimizes for the minimum transparency percentage. Though as per our experiments, it should take only a couple of attempts to find the appropriate region.

A.4.3 Possible risk of this attack

Weapon detection APIs are deployed in various applications to filter out images containing weapons that might look violent and shocking or to simply detect weapons in images for high-stakes applications. Our results suggest potential risks that the attacker can very easily apply transparent patch to fool the weapon detection APIs, resulting in safety and security concerns.

A.5 Importance of the OCR attack

The current work on evaluating the robustness of deep neural networks based OCR systems revolves mostly around traditional adversarial examples (Song & Shmatikov, 2018; Chen & Xu, 2020). We go beyond the traditional adversarial examples and introduce *Simple Transparent Adversarial Examples* to evaluate the robustness of OCR systems. As vision APIs offering OCR services get deployed in high-stakes applications and safety-critical areas, such type of attacks can cause a tremendous loss. Moreover, as these type of adversarial images can be easily crafted by anyone with just online tools that are publicly available, they pose a higher security risk than the current typical adversarial examples.

A.6 Possible Applications and Risks of OCR attack

We show some ways in which the attacker can use Simple Transparent Adversarial Examples for OCR to attack high-stakes applications.

1. **Breaking Blind Review**: anonymized submissions can be broken by adding author names to the figures such that they evade the vision of time-limited humans but are detected by OCR applications.
2. **Breaking Check Scanner Systems**: A lot of check scanner APIs are available in the market that use OCR to process the checks. The attacker can embed a secret text, such as manipulating the amount or the name, which can have a major security threat.
3. **Steganography**: Steganography is the method of hiding secret data within a file to avoid detection. *Secret embedding approach* can be used as a simple technique to perform

steganography, where image with secret invisible text can sent safely to its destination without the actual content being getting revealed.

A.7 Images that fool both time-limited humans and Google Cloud Vision OCR

(a) Original image (b) Modified image (c) Original image (d) Modified image

Figure 4: (a) and (c) are the original images. (b) and (d) are modified in such a way that they fool both time-limited humans as well as machines (Google Cloud Vision's OCR). (b) has a secret embedded text "Hello World" of font size 5 px and *RGB* difference 0 inside a rectangular region formed by the x & y coordinates (4, 13), (27, 13), (4, 28), (27, 28) and (d) has a secret embedded text "Hello World" of font size 5 px and *RGB* difference 0 inside a rectangular region formed by the x & y coordinates (18, 32), (77, 32), (18, 59), (77, 59).

A.8 Future directions

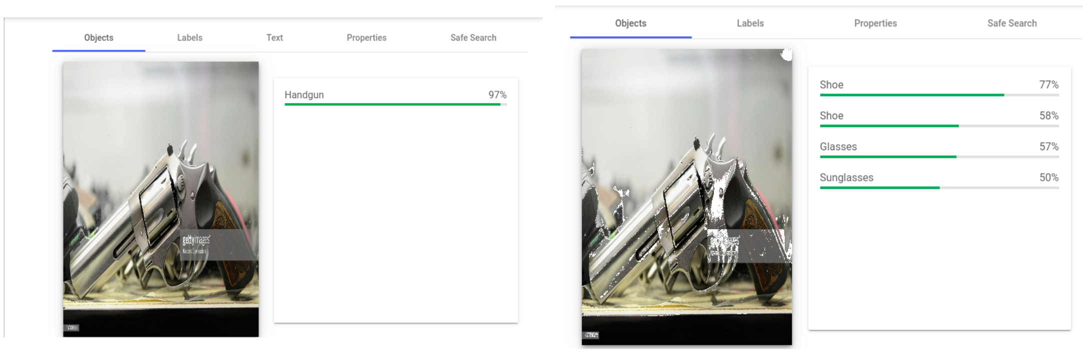
There are also some future interesting directions. First, can we effectively apply the transparent patch on other objects apart from Guns such that they still remain unambiguous to humans but evade object detection. Second, it would be also interesting to see how robust are the pre-trained models to these attacks. Our primary focus has been on evaluating the robustness of publicly deployed APIs since the attacker can very easily query them through the web interface in the real-world.

A.9 More results

(a) (b) (c) (d) (e) (f) (g) (h) (i)

(j) (k) (l) (m) (n) (o) (p) (q) (r)

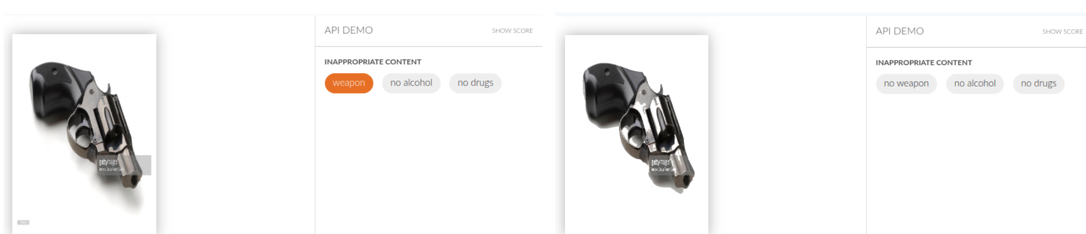
Figure 5: (a) to (i) are original unmodified images. (j) to (m) are modified by adding secret embedding "Hello World" with varying font sizes and font colors. They evade vision of time-limited humans but are detected by Google Cloud Vision's OCR. (n) to (r) are modified by adding white transparent patches using an online tool. They fool object detection APIs such as sightengine.com, PicPurify, Google Cloud Vision, and Microsoft Azure's computer vision API either by evading detection or getting misclassified. (p), (q), and (o) are misclassified as packaged goods, camera, and grooming trimmer by Google Cloud Vision API, (n), (o), and (r) evade sightengine.com, (o) and (r) evade PicPurify.



(a) Original image

(b) Modified image

Figure 6: (a) is the original image that gets detected successfully by Google Cloud Vision API. (b) is modified image having 6 % transparency intensity and is misclassified as a **shoe** with 77% confidence by Google Cloud Vision API.



(a) Original image

(b) Modified image

Figure 7: (a) is the original image having gun that is detected successfully by sightengine.com. (b) is the image modified by adding transparent patch, which evades the sightengine.com weapon detection API.