

AlphaDog: No-Box Camouflage Attacks via Alpha Channel Oversight

Qi Xia

Department of Electrical and Computer Engineering
University of Texas at San Antonio
qi.xia@my.utsa.edu

Qian Chen

Department of Electrical and Computer Engineering
University of Texas at San Antonio
guenevereqian.chen@utsa.edu

Abstract—Traditional black-box adversarial attacks on computer vision models face significant limitations, including intensive querying requirements, time-consuming iterative processes, a lack of universality, and low attack success rates (ASR) and confidence levels (CL) due to subtle perturbations. This paper introduces **AlphaDog**, an Alpha channel attack, the first universally efficient targeted no-box attack, exploiting the often overlooked Alpha channel in RGBA images to create visual disparities between human perception and machine interpretation, efficiently deceiving both. Specifically, **AlphaDog** maliciously sets the RGB channels to represent the desired object for AI recognition, while crafting the Alpha channel to create a different perception for humans when blended with a standard or default background color of digital media (thumbnail or image viewer apps). Leveraging differences in how AI models and human vision process transparency, **AlphaDog** outperforms existing adversarial attacks in four key ways: (i) as a no-box attack, it requires zero queries; (ii) it achieves highly efficient generation, taking milliseconds to produce arbitrary attack images; (iii) **AlphaDog** can be universally applied, compromising most AI models with a single attack image; (iv) it guarantees 100% ASR and CL. The assessment of 6,500 **AlphaDog** attack examples across 100 state-of-the-art image recognition systems demonstrates **AlphaDog**'s effectiveness, and an IRB-approved experiment involving 20 college-age participants validates **AlphaDog**'s stealthiness. **AlphaDog** can be applied in data poisoning, evasion attacks, and content moderation. Additionally, a novel pixel-intensity histogram-based detection method is introduced to identify **AlphaDog**, achieving 100% effectiveness in detecting and protecting computer vision models against **AlphaDog**. Demos are available on the **AlphaDog** website [1] (<https://sites.google.com/view/alphachannelattack/home>).

I. INTRODUCTION

Artificial intelligence (AI) plays a crucial role in computer vision tasks, enabling machines to comprehend visual data with precision. These AI models analyze diverse image formats, ranging from the conventional RGB format of JPEG to the enriched domain introduced by PNG, TIFF, HEIF, WebP, and GIF, which incorporate the Alpha Channel. This transparent layer revolutionizes image integration, fostering

creativity in graphic design and web development by seamlessly blending opaque and translucent elements.

Challenges of Traditional Adversarial Attacks. Despite the advancements, traditional black-box adversarial attacks [2]–[8] pose formidable challenges. These attacks, leveraging subtle manipulations of benign image RGB channels, confront four significant limitations such as (i) Intensive Querying: Crafting adversarial examples demands repeated querying of victim AI models, hindered by API limits and the risk of detection due to excessive queries. (ii) Low Efficiency: Generating attack examples through iterative perturbation tuning consumes substantial time, hindering operational agility. (iii) Model Specificity: Adversarial examples are often tailored to specific AI models, limiting their applicability across diverse architectures. (iv) Low Success Rate and Confidence: Constrained perturbations at imperceptible levels lead to low Attack Success Rates (ASR) and Confidence Levels (CL), challenging the efficacy of deceiving robust AI models.

Exploiting Alpha Channel Oversight: A Universal AlphaDog Approach. Modern image recognition platforms and computer vision models frequently overlook the Alpha channel during the **input/output (I/O) processing** stage, exposing a critical vulnerability that is exploited by our **AlphaDog**, an Alpha channel attack. Exploiting this vulnerability, **AlphaDog** efficiently generates attack images, regardless of the victim model's architecture or design, making it a universally effective attack method. By manipulating transparency, **AlphaDog** generates attack images that look normal to human observers but are recognized as malicious by AI models, creating a gap between human perception and machine interpretation. This vulnerability primarily impacts pure grayscale images and grayscale areas within color images, as the RGBA format manages only a single Alpha channel, leaving color regions unaffected.

Efficient AlphaDog Attack Image Generation. To achieve this deception, the RGB channels of **AlphaDog** attack images I_{Atk} are relevant to those of the malicious target image (I_{AI}). When AI models process the input attack image, they disregard its Alpha channel and focus solely on the information from I_{AI} 's RGB channels. Meanwhile, meticulous calculation of Alpha channel values ensures that when I_{Atk} is viewed through digital image media, humans perceive it as I_{Eye} . It is important to note that both I_{AI} and I_{Eye} are

arbitrary grayscale images or grayscale regions within color images. For further details, refer to Section IV.

Challenges in Designing and Executing AlphaDog Attacks. As outlined earlier, AlphaDog exploits a vulnerability in modern AI models’ input/output (I/O) libraries, where the Alpha channel of the input image is often ignored. This allows the creation of AlphaDog attack images that consistently trick AI models into identifying I_{Att} as I_{AI} , achieving a 100% Attack Success Rate (ASR). However, for AlphaDog attacks to be truly effective and covert, it is critical that human observers perceive I_{Att} as a benign, normal image I_{Eye} . This perception is influenced by the background color of the digital media displaying the image. For example, in most common image viewer applications like web browsers, the default background color is typically *white*. On macOS, the default image viewer app is *Preview*, and on Windows, it is *Photos*, both of which use a default *gray* background. The calculated Alpha channel values for I_{Att} are influenced by the background colors, as the attack image is blended with them.

A challenge arises when an AlphaDog image intended for a white background is viewed in an App with a gray background. In such cases, artifacts from I_{AI} or unintended visual elements may become noticeable, potentially alerting attentive users. To address this, we recommend reducing the intensity of I_{AI} to minimize the visibility of these artifacts and ensure that the attack remains undetected. Further details on addressing this challenge are provided in Section IV-C.

AlphaDog Threats Across Critical Sectors. Critical infrastructure that relies on AI technology is highly vulnerable to AlphaDog attacks. In image-based systems, the insertion of a single AlphaDog attack image, such as one depicting two political figures, can disrupt facial recognition systems. In the realm of telehealth and medical imaging, AlphaDog presents a serious threat by potentially leading to misdiagnoses in grayscale images like X-rays, MRIs, and CT scans. AlphaDog can endanger patient safety and also open the door to fraud, such as manipulating insurance claims. Even in everyday scenarios like traffic signs, AlphaDog can alter grayscale elements of road signs, potentially misleading autonomous vehicles and posing a significant risk to road safety. The wide-reaching implications of AlphaDog make it a critical concern across multiple sectors.

AlphaDog Real-World Practical Scenarios. We generated 6,500 AlphaDog attack images and tested them across 100 AI models, including 80 open-source systems and 20 cloud-based AI platforms. AlphaDog achieved a 100% Attack Success Rate (ASR) and 100% Classification Loss (CL). Figure 1 illustrates four examples of AlphaDog attacks and their evaluations. These examples reflect real-world scenarios in which AlphaDog manipulates grayscale elements within color images, such as a grayscale “20” speed limit sign in a color street scene, a grayscale image of a cat, a grayscale photo of President Biden, and an X-ray of a healthy hand.

The results presented in Figure 1 demonstrate AlphaDog’s effectiveness in deceiving cloud-based image recognition systems with predefined malicious targets. For instance, Chat-

GPT4 [9] incorrectly interprets the speed limit sign as indicating 75 miles per hour, while IMAGERecognize [10] misidentifies the grayscale cat image as a Canine/Dog/Husky with a confidence score of 100%. Similarly, Amazon Rekognition [11] mistakenly identifies President Biden’s image as President Obama with a confidence score of 99.9%. Furthermore, Google Bard [12] (later renamed to Gemini) incorrectly classifies the X-ray of a healthy hand as showing a clear proximal dislocation. For more demonstrations and details, visit the AlphaDog website [1], which provides an extensive overview of AlphaDog attack examples. These examples illustrate AlphaDog’s ability to compromise various AI models, highlighting the significant discrepancy between machine interpretation and human observation.

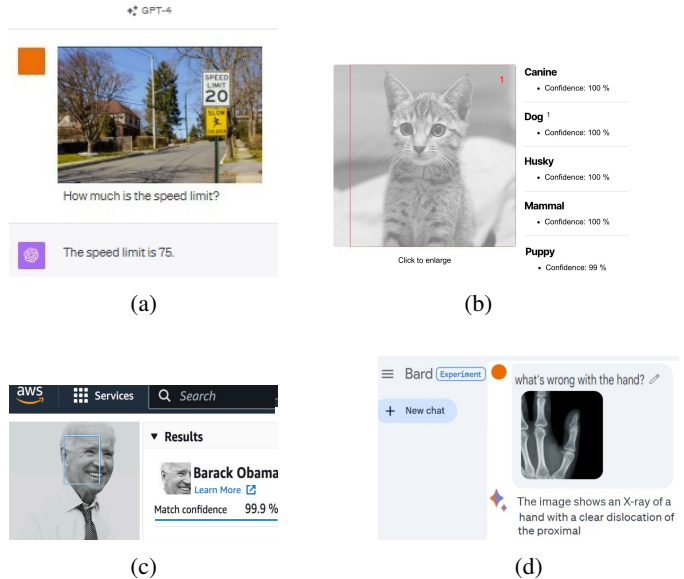


Fig. 1: Illustration of AlphaDog and four out of 6,500 tested attack image examples using the Google Chrome web browser as the image media. (a) The attack image is interpreted by ChatGPT-4 Vision [9] as “75 mph,” but visualized by human observers as “20 mph”. (b) The attack image is visualized as a cat by human observers, but an online AI Image Recognition System [10] identifies it as a husky dog. (c) The attack image is visualized as President Biden, while Amazon Rekognition [11] recognizes it as President Obama’s image. (d) The attack image is an X-ray of a healthy hand, but Bard [12] interprets it as a dislocated hand. AlphaDog demos are available here [1].

Contributions. AlphaDog presents several significant contributions to the field of adversarial attacks in computer vision:

- **First Targeted No-box Attacks:** AlphaDog stands as the first no-box attack, requiring **zero queries** and **no prior access** to AI models. This is possible due to a common behavior among targeted AI models: the removal of the input image’s Alpha channel. This innovation facilitates data poisoning, image alteration, and content removal attacks without requiring queries or responses between attackers and the targeted AI models.

- **First Universal Attack:** A single AlphaDog attack can effectively compromise all AI models that process the Alpha channel in a similar way. AlphaDog achieves a 100% ASR, validated across 100 state-of-the-art open-source and commercial AI models using 6,500 AlphaDog attack images derived from two arbitrary images. Of these models, 98 remove the Alpha channel from RGBA images, while only two replace the Alpha channel with either a black or white background. This universal approach eliminates the need for attackers to craft tailored adversarial examples for individual AI systems, representing a major leap forward in the adversarial threat landscape.
- **Guarantees 100% ASR, CL, and Stealthiness:** Unlike perturbation-based attacks, which often suffer from low adversarial success rates (ASR) and confidence levels (CL), AlphaDog delivers **clear target images** to AI models, ensuring 100% ASR and CL. An IRB-approved experiment with 20 participants from a large U.S. university campus confirmed the stealthiness of AlphaDog attack images.
- **Efficient and Automated Attack Example Generation:** Unlike traditional adversarial attacks, which require extensive queries and time, AlphaDog efficiently produces attack examples. The time complexity of automatic attack image generation is $O(MN)$ (Quadratic Time), enabling the generation of a $1,000 \times 1,000$ attack image in milliseconds, even on a standard Intel i3 CPU laptop.
- **Root Cause Analysis:** This study performs a root cause analysis on 20 off-the-shelf cloud-based image recognition systems and 80 open-source AI models. It identifies the vulnerability of AlphaDog stemming from the overlooked Alpha channel of input RGBA images by AI models.
- **Broad Applicability:** AlphaDog demonstrates effectiveness across various image contexts, compromising transparency in both grayscale and color images with grayscale regions. Experimental results highlight AlphaDog’s efficiency and disruptive potential in critical sectors such as transportation, medical imaging, telehealth, and online image-based surveys, all achieved without prior knowledge of or access to victim AI models.
- **Defense Mechanism Development:** A novel intensity histogram-based detector is crafted to effectively protect AI models from AlphaDog attacks, achieving a 100% detection rate.
- **AlphaDog Adversarial Attack Dataset:** A dataset containing 6,500 AlphaDog attack examples of arbitrary images is established, aiming to bridge the perceptual gap between human perception and machine interpretation. This dataset serves as a valuable resource for designing state-of-the-art AI models and enhancing their resilience against AlphaDog attacks.

Ethical Issue. We have engaged with key stakeholders, including Google, Amazon, and Microsoft, regarding the AlphaDog issue and received responses indicating their proactive efforts to address this concern. Notably, AlphaDog has been recognized and selected as a candidate for the Microsoft Bug Bounty Award 2024, highlighting the industry’s

recognition of the critical need to resolve this vulnerability.

Paper Outline. Section II describes preliminary knowledge. Section III presents the threat model. Section IV explains the AlphaDog math foundation. Section V demonstrates AlphaDog real-world feasibility. Section VI analyzes the factors affecting AlphaDog. Section VII investigates the defense strategy against AlphaDog. Section VIII discusses the limitations of the study. Section IX reviews related prior studies. Section X concludes the paper.

II. BACKGROUND AND PRELIMINARIES

AlphaDog introduces an innovative technique exploiting vulnerabilities in the Alpha channel specific to **RGBA** image formats. This method seamlessly combines an AlphaDog attack image with a background color, creating distinctions between normal and malicious target images that can be perceived by human observers and interpreted by AI models. This section provides an overview of essential concepts, covering the RGBA image format and transparency, the portrayal of background colors in image media, and the Alpha Compositing technique. Additionally, we explore how modern AI models handle transparency information from input RGBA image files and discuss the latest advancements in blackbox settings for adversarial image attacks.

A. RGBA Image Format.

The widely used RGBA image format has Red (R), Green (G), and Blue (B) color channels and an Alpha channel (A). Alpha channel values range from 0 (fully transparent) to 1 (fully opaque) for each pixel. When an RGBA image is composited onto a background, the Alpha channel controls the level of transparency. Introducing a background color influences the overall appearance of the image. For example, adding a pure white background causes pixels with higher Alpha values to appear more opaque, while pixels with lower Alpha values permit more of the white background to show through, resulting in varying levels of transparency.

B. Digital Image Media and Background Colors

Digital image media has two types: Image Viewer (IV) and Thumbnails (TB). IVs like Chrome, Adobe PDF Viewer, Windows Photo Viewer, Mac Preview, and iPhone Photo display images in their original size. The scaled-down versions provided by operating system TB tools, like Windows File Explorer and Mac Finder, enable users to preview image content without opening it through IVs. Image media comes with either white or gray background colors. When visualizing RGBA images, images get composited onto these apps’ backgrounds. This can cause variations in the same RGBA image because pixel transparency is notably influenced by the image medium’s background color.

The grayscale background of image media enables color normalization from black to white on a scale of 0 to 1. In this scale, 0 signifies black, and 1 represents white. The intermediate shades of gray are used to convey various levels of transparency in the composited image. Table I summarizes

TABLE I: Background Colors of Image Media Apps. * indicates default Apps for the Operating System.

Composition Strategy	Thumbnail (Reduced-Size Image Display)	Viewer (Full-Size Image Display)
White Background	Google Chrome, Safari, Mozilla Firefox, Microsoft Edge, Microsoft IE, macOS Finder*, iPhone Photos*, Win10 file explorer*, Ubuntu file explorer*	Google Chrome, Safari, Mozilla Firefox, Microsoft Edge, Microsoft IE, Adobe PDF viewer, Adobe Photoshop, Ubuntu Image viewer*, iPhone Photos*
Gray Background	N/A	Win10 Photos* (R:64 G:64 B:64), Mac Preview* (R:150,G:150,B:150)

commonly used image media and their background colors. Most default to a white background, except for Windows 10 Photos (gray), and Mac Preview (gray).

C. Handling of RGBA Images by AI Models

The AI models' I/O often involves processing only the RGB channel values while disregarding the Alpha channel of input images. This common practice among these systems typically entails either discarding the Alpha channel entirely or introducing and merging a default white or black background color with the input image before proceeding to the image processing stage or pipeline. Table II provides a breakdown of 100 widely utilized computer vision models, comprising 80 open-source models categorized based on their training datasets, alongside 20 commercial cloud-based image recognition systems. Notably, among these models, 98 remove the Alpha channel from the input image. However, Google Bard [12] and Google Cloud Vision API [13] deviate from this norm by adding a black or white background.

D. Element-wise Operations

Element-wise matrix operations are mathematical functions and algorithms utilized in computer vision, operating on individual pixels of an image. When two image matrices A and B share the same dimensions, all four basic operations of addition (+), subtraction (-), multiplication (\odot), and division (\oslash) between these matrices can be performed. Detailed operation can be seen in Appendix B. Element-wise operations adhere to the same rules as operations on numbers, including the order of precedence, associative property, and commutative rules. In Section IV-B, to determine the intensity range of target images, we leverage the multiplication property of division (i.e., Equations 5 and 6), which can be expressed as:

$$A \oslash B = C \iff A = B \odot C \quad (1)$$

E. Alpha Compositing in Computer Graphics

Alpha Compositing, also known as Alpha Blending, is a technique utilized in computer graphics to merge an image

TABLE II: Computer vision models for Alpha channel treatment in input images. "Open" denotes open-source models, and "Cloud" signifies commercial cloud-based image recognition systems. Only the underscored cloud-based systems with w (Google Cloud Vision Api) and b (Bard) add a white background to the input image; others remove the input image's Alpha channel.

Training Dataset (Model Category)	100 Computer Vision Models 80 (Open) and 20 (Cloud)
COCO [33] (Open)	Mask RCNN [14]–[17], YOLOv3 [18]–[32], YOLOv4 [34], YOLOv5 [35], [36], RetinaNet [37]–[40], CenterNet [41]–[44], EfficientDet [45]–[49], Cascade RCNN [50], [51]
Pascal VOC [52] (Open)	Faster RCNN VGG [53]–[58], YOLOv1 [59], YOLOv2 [60]
ImageNet [72] (Open)	AlexNet [61]–[65], ResNet [66]–[71], EfficientNet [73]–[76], InceptionV3 [77], InceptionV4 [78], GoogLeNet [79]
MNIST [80] (Open)	LeNet-5 [81], [82]
FDDB [83] (Open)	Facenet [84], Cascade CNN [85]
KITTI [86] (Open)	MonoDepth [87]–[90]
BDD100k [91] (Open)	YOLOv3 [92], [93], YOLOv5 [94]
Wider Face [95] (Open)	Facenet [96], YOLOv3 [97], [98], YOLOv2 [99], [100]
CIFAR-10 [101] (Open)	ResNet [102]
Dataset Unknown (Cloud)	<u>$Bard^b$ [12]</u> <u>$GoogleCloudVisionApi^w$ [13]</u> Amazon Rekognition [11] GeminiProVisionAPI [103] Baidu Image [104], Baidu API [105] IMAGERecognize [10], TeachableMachine [106], Nyckel [107], Labelbox [108], ChatGPT4 [9], Wolfram [109], Vue.ai [110], Microsoft Azure [111], AliYunVision [112], Tencent Vision [113], Landing Lens [114], Clarifai [115], Imagga [116], ANYLINE [117]

with a background, replicating partial or complete transparency. The term "Alpha" pertains to the Alpha channel of RGBA images. A grayscale RGBA image I_{Atk} comprises a 3-D matrix of size $m \times n \times 2$, consisting of two 2-D matrices of the same size ($m \times n$): an RGB channel intensity matrix (I_{IN}) and an Alpha Channel Matrix (A). Therefore, the AlphaDog attack image I_{Atk} can be expressed as the concatenation of I_{IN} and A as follows:

$$I_{Atk} = \text{Concat}(I_{IN}, A). \quad (2)$$

When this RGBA image I_{Atk} is superimposed on a background color BKG of digital image media, it blends to produce a final composite image I_{Eye} , perceived by human eyes. The Alpha compositing formula is given by:

$$I_{Eye} = A \circ I_{IN} + (1 - A) \cdot BKG. \quad (3)$$

Here, BKG is technically a $m \times n$ matrix as well, but it is often a constant matrix with all elements having the same value. Thus, we can treat BKG as a scalar for simplicity (e.g., $BKG = 1$ for a white background) and calculate it by scalar multiplication with another matrix. Therefore, the operation between matrix I_{IN} and matrix A is element-wise multiplication, while the operation between matrix $(1 - A)$ and constant number BKG is scalar multiplication.

Application of Alpha Compositing for AlphaDog Design. Alpha Compositing (Equation 3) serves as the foundational theoretical framework for generating AlphaDog attack images. In the process, when AI models process I_{Atk} , its Alpha channel A is disregarded, leading the AI models to perceive I_{AI} as equivalent to I_{IN} . Conversely, human observers perceive the composite image I_{Eye} , resulting in a perceptual disparity between AI models and human interpretation.

To illustrate how AlphaDog generates targeted perceptual disparity, we provide a 5×5 compositing example where we design I_{Atk} to be identified by AI models as the letter “Z” (I_{AI}), while our goal is for human observers to perceive the letter “K” (I_{Eye}) as depicted in Figure 2. Since we have predetermined I_{AI} and I_{Eye} , we can calculate A using Equation 3. It is important to note that all matrices are normalized with intensities ranging from 0 to 1. Additionally, $BKG = 1$ indicates a white background, which is common in most widely used image viewers as observed in Table I. We have “Z” (I_{AI}) and “K” (I_{Eye}) as:

$$I_{AI} = I_{IN} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$I_{Eye} = A \circ I_{IN} + 1 - A = \begin{bmatrix} 1 & 0.5 & 1 & 1 & 0.5 \\ 1 & 0.5 & 1 & 0.5 & 1 \\ 1 & 0.5 & 0.5 & 1 & 1 \\ 1 & 0.5 & 1 & 0.5 & 1 \\ 1 & 0.5 & 1 & 1 & 0.5 \end{bmatrix}$$

, and therefore, we obtain A as:

$$A = \begin{bmatrix} 0 & 0.5 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0.5 & 0 \\ 0 & 1 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 1 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 \end{bmatrix}.$$

During the practical AlphaDog attack image generation process, the attacker can **freely/arbitrarily** select two target images I_{AI} ($I_{IN} = I_{AI}$) and I_{Eye} to compute A using Equation 3. With the calculated A and I_{IN} , the attacker can then create I_{Atk} , making it an exceptionally efficient one-step procedure. Refer to Section IV for further elaboration.

F. Black-box Adversarial Image

A black-box attack, as defined by Papernot et al. and Costa et al. [118], [119], is characterized by the constraint that

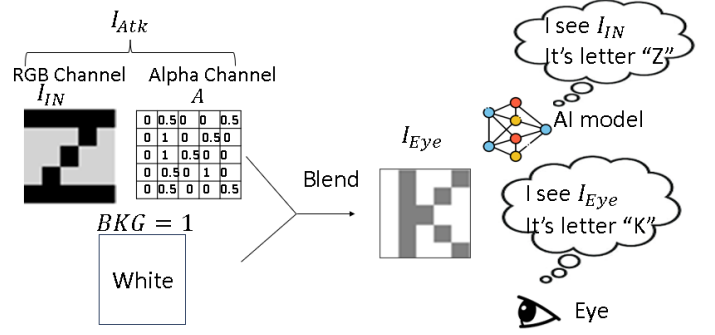


Fig. 2: An example of visual disparity by AlphaDog, where an AI model removes Alpha channel from I_{Atk} and sees only I_{IN} as “Z”, while human eyes sees the blending result I_{Eye} as “K”.

attackers can only access samples from an oracle or pairs consisting of an AI model’s input and output. In practical terms, attackers introduce imperceptible perturbations to an otherwise benign image. They then submit this altered image as queries to the target AI model to elicit responses. Based on these responses, attackers adjust the perturbations incrementally. This process is iterative and can involve hundreds or even thousands of iterations. Through this method, attackers aim to subtly manipulate the image until the AI model misclassifies it. However, such attacks face several limitations, including the need for excessive queries, low efficiency due to iterative tuning, model-specificity, and low success rates and confidence levels. To address these challenges, this paper introduces a targeted no-box threat model for AlphaDog, overcoming the limitations of traditional black-box adversarial attacks.

III. ALPHADOG: TARGETED NO-BOX THREAT MODEL

This section performs a root cause analysis of AlphaDog, highlighting vulnerabilities in both open-source and proprietary AI models. We then establish the AlphaDog threat model, delineating attack objectives and associated no-box settings, investigating potential attacker capabilities, and examining real-world AlphaDog attack scenarios.

A. Root Cause Analysis of AlphaDog

The vulnerability exploited by AlphaDog originates from how contemporary computer vision models process input RGBA images. Rather than utilizing information from all four RGBA channels in the input images, these models selectively read data solely from the RGB channels. Frequently, the Alpha channel values of the AI models’ input image are disregarded by either eliminating this channel or, in occasional instances, blending the input image with a white or black background.

1) *Analysis of open-source AI models:* We analyzed the I/O source code of 80 widely-used open-source computer vision models available on GitHub to examine how they process the Alpha channel of input images, confirming and validating the presence of the AlphaDog vulnerability. Table II provides the complete list of examined open-source AI models.

OpenCV [120], TensorFlow I/O [121], and Pillow [122] are the primary libraries used by these models to read input images. OpenCV, through the `imread(imageFilePath)` function, defaults to extracting image data only from the RGB channels, excluding Alpha channel information. While TensorFlow I/O and Pillow can consider every channel when reading a PNG image, our analysis indicates that most programmers, if not all, typically trim the RGB channels, neglecting the Alpha channel values. A code sample demonstrating this behavior is provided in Figure 7 in Appendix A. This analysis of 80 open-source AI models’ I/O source code confirms that the most popular open-source computer vision models remove Alpha channel information from input images.

2) *Analysis of cloud-based image recognition systems: proprietary AI models:* We analyze 20 prominent commercial cloud-based image recognition systems, chosen for their prominence and accessibility on the internet. These systems and analysis results which include ChatGPT-4.0 [9] and Google’s Gemini Pro API [103], are listed in Table II. Our investigation revealed that out of the 20 proprietary systems tested, only Google’s Bard [12] and Google Cloud Vision API [13] handle the Alpha channel differently by blending a black or white background color with the input image. In contrast, similar to many open-source AI models, the remaining 18 cloud-based image recognition systems discard the Alpha channel information from the input image.

B. Attack Objective: Targeted Attack

AlphaDog is a **targeted** attack that manipulates image transparency to deceive computer vision AI models. Its goal is to cause these models to misclassify the AlphaDog attack image as a predetermined **malicious target image** or class. Concurrently, the attack aims to ensure that human observers perceive the manipulated image differently, recognizing it as a specific **normal target image** without detecting any anomalies or suspicious information. Grayscale regions are vulnerable to AlphaDog, while colorful regions remain immune since the RGBA image format’s Alpha channel can modulate a pixel’s overall intensity but cannot alter the intensity ratio between the red (R), green (G), and blue (B) channels. Based on the goals of the AlphaDog attack, the following definitions are set up.

- 1) *Human Observer/Victim User:* Refers to AI model designers, developers, or users who are humans and visualize images using digital image media apps.
- 2) *Targeted/Victim AI (or Computer Vision) Models:* Denotes open-source AI models or cloud-based image recognition systems capable of interpreting images and classifying or predicting their contents.
- 3) *Normal Target Image:* Refers to the specific target image intended for recognition by human observers.
- 4) *Malicious Target Image:* Denotes the particular target image that the victim AI model should interpret.
- 5) *AlphaDog Attack Images:* Represents the source or input images fed into victim AI models, intending to deceive the models into interpreting the attack as the malicious target

image. Simultaneously, human observers should recognize it as a normal target image without discerning any noise or suspicious information.

C. No-box Settings

Based on the root cause analysis outlined in Section III-A, AlphaDog emerges as highly effective in a **No-box** attack scenario since almost all cutting-edge AI models demonstrate the behavior of discarding the Alpha channel. In this context, **No-box** signifies that the attack does not require access to the victim AI model (or the “box”).

- *Query:* No query. The attacker assumes that all victim AI models remove the input image’s Alpha channel.
- *AI Model Output Response:* No response from the victim AI models.
- *Human Visualized Image:* Normal target images I_{Eye} .

D. Attacker Capability

The attackers possess the following two capabilities or requirements for the design and implementation of AlphaDog.

- **Selection of malicious-normal target image pairs for AI models and human observers:** The attacker strategically chooses a pair of target images aligned with their attack objectives and the critical sectors they aim to compromise. One image is the malicious target for AI model interpretation, while the other appears normal for human observation.
- **Restricted access and no knowledge of victim AI models.** Attackers face restrictions on accessing or understanding targeted AI models, especially in their developmental stages when models remain inaccessible. Despite this limitation, attackers can still craft AlphaDog attack images under the assumption that these new models adhere to the prevalent practice of disregarding the input image’s Alpha channel. These crafted attack images might be illicitly integrated into training datasets, potentially resulting in the deployment of unreliable and untrustworthy AI models for the public use.

E. Attack Scenario

Real-world targeted no-box AlphaDog attack scenarios can be implemented as follows.

- 1) **Exploiting Grayscale Areas:** AlphaDog excels at targeting grayscale regions within images, enabling attackers to compromise the integrity of both purely grayscale images and colored images containing such regions.
- 2) **Evasion Attacks:** Malicious AlphaDog images can lead to the misidentification of crucial components within the normal target image when analyzed by AI models.
- 3) **Data Poisoning:** Attackers insert malicious AlphaDog images into the training dataset, leading to models learning incorrect behaviors or creating exploitable vulnerabilities.

IV. ALPHADOG DESIGN FOUNDATION

A. AlphaDog Attack Image Generation

An adversary constructs the AlphaDog attack image I_{Atk} using a pair of arbitrary target images, determined by the adversary’s attack purpose: I_{AI} , designated as the malicious

target image for interpretation by AI models, and $I_{E_{ye}}$, intended as the normal target for human observation. As outlined in Section II-E, the construction of I_{Atk} involves obtaining two 2-D matrices: the RGB channel matrix I_{IN} and the Alpha channel matrix A . The RGB channel matrix of I_{Atk} is set equal to I_{AI} , ensuring that AI models disregarding the Alpha channel perceive it as I_{AI} . Additionally, A is calculated using Equation (Eq.) 4 to ensure that when I_{Atk} is viewed on digital image media with a background, it appears as $I_{E_{ye}}$. Algorithm 1 outlines the pseudocode for automatically generating AlphaDog attack images.

Algorithm 1 AlphaDog Attack Image Generation

```

function IMAGEGENERATION( $I_{AI}$ ,  $I_{E_{ye}}$ ,  $m$ ,  $n$ )
   $I_{Atk} \leftarrow$  empty 3D array with dimensions  $m \times n \times 2$ 
   $A \leftarrow$  empty 2D array with dimensions  $m \times n$ 
   $I_{AI} \leftarrow$  PREPROCESS( $I_{AI}$ ,  $m$ ,  $n$ )  $\times 0.8 + 0.2$ 
   $I_{E_{ye}} \leftarrow$  PREPROCESS( $I_{E_{ye}}$ ,  $m$ ,  $n$ )  $\times 0.2$   $\triangleright$  Eq. 9
  for  $i \leftarrow 1$  to  $m$  do
    for  $j \leftarrow 1$  to  $n$  do
       $A[i][j] \leftarrow \frac{1 - I_{E_{ye}}[i][j]}{1 - I_{AI}[i][j]}$   $\triangleright$  Eq. 4
    end for
  end for
   $I_{Atk} = [I_{AI}, A]$   $\triangleright$  concatenate RGB and Alpha
  return  $I_{Atk} \times 255$   $\triangleright$  return 8-bit attack image
end function

function PREPROCESS( $I$ ,  $m$ ,  $n$ )
   $I \leftarrow$  remove  $I$ 's alpha channel
   $I \leftarrow$  resize( $I$ ,  $m$ ,  $n$ )  $\triangleright$  Resize the image to  $n \times m$ 
   $I \leftarrow I/255$   $\triangleright$  Normalize the 8-bit image
  return  $I$ 
end function

```

1) *Preprocessing*: In this step, adversaries arbitrarily select a pair of target images: I_{AI} and $I_{E_{ye}}$. The RGB matrices of these target images are known to the attackers, and they may be of any image type, including RGB or RGBA. If the target images are RGBA images, their Alpha channel values are discarded. These grayscale image pairs are normalized before processing, ensuring that the generated I_{Atk} is also normalized, with intensities ranging from 0 to 1. Normalization is achieved by dividing each pixel value by its maximum intensity value, typically 255 for an 8-bit image. Additionally, all images are resized to a standardized size of $m \times n$. The resulting I_{Atk} comprises two 2-D matrices: the RGB channel intensity matrix (I_{IN}) and the Alpha matrix (A). I_{Atk} is represented as a 3D matrix of size $m \times n \times 2$, formed by concatenating I_{IN} and A along the third dimension. Equation 2 illustrates this concatenation. In Algorithm 1, this step corresponds to the **PREPROCESS** function.

2) *Creating I_{Atk}* : Generating the AlphaDog attack image I_{Atk} involves calculating its Alpha channel matrix A , which depends on $I_{E_{ye}}$ and the background color of the digital image media. For illustration, we assume a white background color ($BKG = 1$), commonly seen in web browsers and thumbnails

(see Table I). The steps for creating I_{Atk} are as follows, referring to lines 6-11 of Algorithm 1:

- (a) Initialize $I_{IN} = I_{AI}$, considering that most advanced AI models remove the Alpha Channel (A) and process only the intensity matrix I_{IN} .
- (b) When I_{Atk} blends with digital image media containing a white background (intensity = 1), according to Equation 3, calculate the Alpha matrix:

$$A = (1 - I_{E_{ye}}) \odot (1 - I_{AI}) \quad (4)$$

- (c) According to Equation 2, concatenating I_{IN} and A to obtain $I_{Atk} = \text{Concat}(I_{AI}, (1 - I_{E_{ye}}) \odot (1 - I_{AI}))$

Insight 1: Crafting an I_{Atk} image is an efficient process with a time complexity of $O(MN)$, where M and N are the dimensions of the image.

B. Achieving Stealthy AlphaDog Attack: Target Image Intensity Histogram Features Analysis

To ensure the stealthiness of I_{Atk} in AlphaDog attacks, the histograms of I_{AI} and $I_{E_{ye}}$ must exhibit high separation, preventing any discernible noise or malicious target image been observed by human eyes. Given that I_{Atk} and its Alpha matrix A are normalized, namely $0 \leq A \leq 1$, according to Equation 4 the following element-wise inequality holds:

$$0 \leq (1 - I_{E_{ye}}) \odot (1 - I_{AI}) \leq 1. \quad (5)$$

To address this inequality, we apply the multiplication property of the division rule (Equation 1) by multiplying both sides by the denominator $(1 - I_{AI})$. Consequently, we establish that the overall intensity of I_{AI} must be lower than that of $I_{E_{ye}}$:

$$0 \leq I_{AI} \leq I_{E_{ye}} \leq 1 \quad (6)$$

This analysis confirms that the histograms of I_{AI} and $I_{E_{ye}}$ exhibit a distinct separation, with I_{AI} on the left side and $I_{E_{ye}}$ on the right side. Consequently, **no histogram overlapping between I_{AI} and $I_{E_{ye}}$** is allowed; otherwise, Equation 6 is violated. If $I_{E_{ye}} < I_{AI}$, plugging this into the expression for A in Equation 4 yields $A > 1$. As the upper bound of A is 1 due to normalization, values greater than 1 are unexpectedly ‘‘clipped’’ to the maximum allowable value of 1. In regions where A is clipped (A_{clip}), a large area with $A_{clip} = 1$ exists. Utilizing Equation 3, we can derive $I_{E_{ye}}$ in this clipped area as:

$$\begin{aligned}
 I_{E_{ye}} &= A_{clip} \odot I_{IN} + (1 - A_{clip}) \times 1 \\
 &= 1 \times I_{IN} + (1 - 1) \\
 &= I_{IN} = I_{AI}
 \end{aligned} \quad (7)$$

This implies that in the clipped area of A (A_{clip}), I_{AI} becomes visible to human eyes, compromising AlphaDog stealthiness. Figure 3 illustrates how a large histogram overlap reveals I_{AI} (the dog silhouette) to human eyes.

On the contrary, **benign RGBA images typically exhibit consistent histograms** (lack of separation) irrespective of the background with which they are blended. This disparity

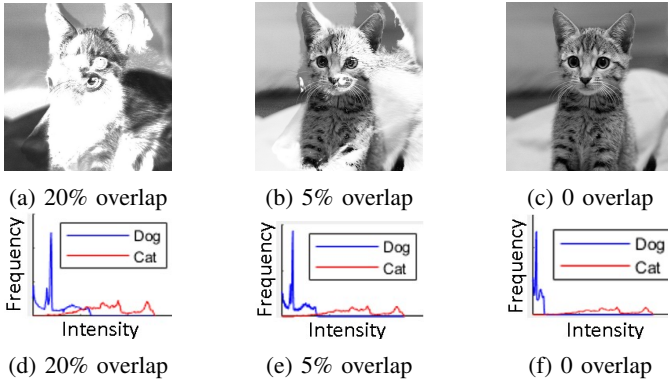


Fig. 3: Illustration demonstrating how intensity histogram overlap between I_{AI} (dog) and I_{Eye} (cat) can expose I_{AI} (dog) to human eyes. The attack image I_{Atk} is viewed using Chrome with a white background blended as I_{Eye} : (a) 20% histogram overlap reveals a clear view of the dog; (b) 5% histogram overlap reveals a silhouette of the dog (Husky); (c) No histogram overlap effectively conceals the dog. Subfigures (d), (e), and (f) depict the histograms of the target images under different levels of overlapping.

in histogram separation between I_{Atk} and benign images forms the basis of the novel defense mechanism introduced in Section VII, enabling the detection of AlphaDog attack images with a 100% detection rate.

Insight 2: The distinct separation of histograms between I_{AI} and I_{Eye} is crucial to maintain AlphaDog stealthiness. This characteristic is utilized to design a defense to detect AlphaDog attack images from benign images, as discussed in Section VII.

C. Addressing the AlphaDog Design Challenge: Ensuring I_{Atk} Stealthiness by Reducing the Intensity of I_{AI}

To wage AlphaDog successfully, AlphaDog remains stealthy when the victim opens the attack image using any image viewer listed in Table I. A key challenge arises when blending I_{Atk} with a gray background, as seen in applications like Windows Photo or Mac Preview, where I_{AI} might be inadvertently revealed, compromising AlphaDog stealthiness. This occurs because Equation 3, designed to ensure that I_{Eye} is displayed when blended with a white background ($BKG = 1$), cannot account for the possibility of a gray background (BKG_{gr}). When I_{Atk} is blended with a gray background, I_{Eye2} is determined by:

$$I_{Eye2} = A \circ I_{AI} + (1 - A) \cdot BKG_{gr} \quad (8)$$

Since $BKG_{gr} \neq 1$, $I_{Eye2} \neq I_{Eye}$, potentially revealing parts of I_{AI} and jeopardizing AlphaDog stealthiness. To mitigate this risk, we propose reducing the intensity of I_{AI} . Lowering I_{AI} intensity minimizes its contribution to I_{Eye2} , making it nearly invisible. After testing 6,500 examples, we found that setting 0.2 as the upper bound for I_{AI} intensity achieves both

100% ASR and stealthiness. Thus, Equation 6 can be updated as:

$$0 \leq I_{AI} \leq 0.2 \leq I_{Eye} \leq 1 \quad (9)$$

Once Equation 9 is ensured, a stealthy I_{Atk} can be created. It is important to note that **the value 0.2 serves as a universal threshold, independent of the specific target images (I_{AI} or I_{Eye}) or AI models.** For a detailed mathematical proof, please refer to Appendix D. The evaluation of stealthiness is thoroughly discussed in Section VI.

Furthermore, it is worth highlighting that this section introduces an additional constraint (Equation 9), which does not conflict with any existing equations or inequalities in Sections IV-A and IV-B. As a result, I_{Atk} remains displayed as I_{Eye} when blended with a white background. This is particularly significant because thumbnails predominantly utilize a white background, ensuring the preservation of stealthiness when displayed on a web browser.

Insight 3: By reducing the intensity of I_{AI} to 0.2, the leakage of I_{AI} is effectively prevented when observed by victim users (human eyes) in I_{Atk} , ensuring the stealthiness of the AlphaDog attack.

D. AlphaDog Design for AI Models with Differential Treatment of Input Image Alpha Channels

In Table II, it is evident that only two cutting-edge AI models deviate from the norm by utilizing a solid white/black background instead of simply removing the Alpha channel from input images. To effectively target these exceptional models, we leverage Equation 3 to compute the A and I_{IN} matrices to create successful adversarial images I_{Atk} . We provide an in-depth exploration of this methodology in Appendix D, demonstrating its effectiveness in crafting such specialized attack images. Additionally, we confirm that Insight 2 remains pertinent even for these outliers, thus ensuring that the defensive strategies outlined in Section VII are equally applicable in mitigating attacks targeting these models.

V. ALPHADOG REAL-WORLD APPLICATIONS

A. Automatic Generation of AlphaDog Attack Images

In real-world scenarios, attackers can leverage AlphaDog for various malicious purposes. One significant application is the automatic generation of a large dataset of AlphaDog attack images. To achieve this, attackers start by selecting arbitrary grayscale target image pairs I_{AI} and I_{Eye} , which are then used to craft the attack image I_{Atk} following the procedure outlined in Algorithm 1. The algorithm's time complexity is $O(MN)$, ensuring efficiency in generating attack images.









When targeting grayscale regions within colorful images, attackers should initially isolate and crop these grayscale areas, designating them as I_{Eye} . Subsequently, they can choose a suitable I_{AI} and preprocess it using Algorithm 1 to generate an attack image I_{Atk} of the corresponding size. Finally, the attacker should substitute the original grayscale area in the image with the newly created attack image I_{Atk} . This streamlined process empowers attackers to automatically generate

AlphaDog attack images tailored to their specific objectives, enabling them to compromise the integrity of both grayscale and colorful images containing grayscale regions.

B. Practical Applications of AlphaDog and Examples of Attack Images

The practical impact of AlphaDog in real-world scenarios highlights its potency and potential for harm, extending from deceptive practices in digital document fraud to potential risks in autonomous vehicles, telehealth, and content moderation. Table III presents a variety of applications, demonstrating AlphaDog can create visual disparities between machine interpretation and human perception.

TABLE III: Four examples of AlphaDog attack images out of 6,500. The left column shows AI-visualized malicious target images (I_{AI}), while the right column displays human-observed normal target images (I_{Eye}). Download examples of AlphaDog attack images from [1].

Target image I_{AI} seen by AI models	Target image I_{Eye} seen by eyes	Description
		Pure grayscale images
		
		
		Speed limit: A grayscale region of a color image

1) *Deceptive Manipulation by leveraging inconsistencies between displays:* AlphaDog attack images are created by exploring different image media apps' different background colors. An attacker can create an image document that shows different content when blended with different background colors. The attack can then get two parties to share the same document. If they use different image viewer with different background color, the content being displayed will be different. This inconsistency can become the basis of potential financial fraud activities

2) *Threats to AI Trustworthiness: Implications for Healthcare and Autonomous Vehicles:* Beyond camouflage, AlphaDog also functions as an image alteration attack, posing risks to AI reliability by inducing false predictions, data poisoning, and inaccurate diagnoses. For instance, consider the scenario where an AI model falsely diagnoses a healthy finger as having a bone fracture due to AlphaDog manipulation. This example highlights the potential impact of AlphaDog on patient safety and the risk of insurance fraud within telehealth platforms, as depicted in the second AlphaDog attack image example in Table III.

The manipulation of grayscale regions within color images, particularly through adversarial attacks, poses a significant risk for severe traffic accidents by poisoning the underlying AI models of Autonomous Vehicles (AVs). The last example in Table III illustrates how a speed limit of 20 miles per hour is falsely detected as 75 miles per hour, causing AVs to accelerate beyond the appropriate speed.

3) *Threats to Content Moderation Integrity: Exploiting AlphaDog in Online Environments:* The utilization of computer vision for content moderation is widespread across numerous online social media platforms and services, including Google [13], Amazon AWS [123], and Microsoft Azure [111]. However, the emergence of AlphaDog presents a potential threat to these systems by compromising the availability and integrity of their AI models. For instance, consider the case of Moderate Content [124], a real-time image content moderation API utilized by numerous websites to filter inappropriate content. AlphaDog can deceive these AI models, causing them to perceive offensive content such as hate speech, pornography, graphic violence, and defamatory material as acceptable. This deceptive capability of AlphaDog poses significant challenges and may contribute to the spread of harmful content within online communities.

VI. ALPHADOG EFFECTIVENESS

This section empirically evaluates the effectiveness and universality of AlphaDog as a targeted no-box attack. Additionally, it investigates the influence of image formats on the success rate of AlphaDog.

Test Image Dataset: Our dataset encompasses the following, and is publicly accessible at [1].

- 1) 2,000 AlphaDog attack images in PNG format, each sized 256×256 , under the assumption that AI models remove the input Alpha channel.
- 2) 4,000 attack images of identical dimensions but in alternative formats (TIFF, WebP, SVG, and GIF), with 1,000 examples per format.
- 3) 500 attack images each specifically tailored for testing the Bard and Gemini AI models, known for their deviations from the norm.

A. AlphaDog Attack Image Generation Efficiency Analysis

As detailed in Section V-A, the time complexity for generating an attack image of size $M \times N$ is $O(MN)$. To assess practical performance, we conducted experiments on a Dell

Intel Core i3 8th Gen and 4GB RAM, implementing the algorithm described in Section V-A using Matlab. The process of generating the 2,000 PNG images, each sized 256×256, required a total of 26 seconds, averaging approximately 13 milliseconds per image.

Insight 4: The generation process for AlphaDog images is highly efficient, with each image processed in milliseconds.

B. AlphaDog Attack Image Stealthiness Evaluation

The stealthiness of AlphaDog attack images is evaluated both mathematically and through an IRB-approved experiment.

For the mathematical verification, we employ the mean squared error (MSE) method. After normalizing I_{Eye} and I_{Eye2} , we compute MSE as follows:

$$MSE = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (I_{Eye}[i, j] - I_{Eye2}[i, j])^2$$

All testing images in Test Image Dataset-1) exhibit an MSE value of 0, indicating perfect stealthiness. This alignment with I_{Eye} is essential for maintaining absolute stealthiness, particularly in thumbnail images and most image viewers with white backgrounds. In the analysis of Test Image Dataset-3 on gray backgrounds, such as those in Windows Photo and Mac Preview, all MSE values are less than 0.08. This further confirms the stealthiness of the attack image datasets, even when viewed against gray backgrounds.

In addition, we conducted an IRB-approved experiment involving 20 participants recruited from a large U.S. university campus. Recruitment methods included distributing project flyers on campus and disseminating digital flyers via email and social media platforms. The participants, aged between 18 and 45 years (with a mean age of 25.18 and a standard deviation of 6.8), comprised 50% females. Ethnic composition among participants was 50% Hispanic, 45% Caucasian, and 5% Asian, African American, and other ethnicities. All participants correctly identified images from Test Image Dataset-1) as I_{Eye} without detecting any anomalies, affirming the stealthiness of AlphaDog attack images.

C. Experimental Validation of AlphaDog Universality

To assess the universal effectiveness of AlphaDog in compromising 98 state-of-the-art open-source computer vision AI models and cloud-based image recognition systems, all capable of removing the input image’s Alpha channel, we conduct experiments on two sets of 1,000 randomly selecting target images from Test Image Dataset-1), labeling them as Set_1 and Set_2. The goal is to evaluate whether identical AlphaDog attack images could successfully compromise different AI models, thus validating AlphaDog universality.

Our findings reveal that all attack images achieve an exceptional 100% ASR, surpassing the performance of any existing black-box attack methods documented in related literature. This remarkable success underscores the unique approach of AlphaDog, which diverges from traditional adversarial

attacks. Unlike conventional methods that introduce perturbations to images in an attempt to mislead computer vision models by crossing decision boundaries, AlphaDog directly presents clear, unmodified target images to the models, ensuring both 100% ASR and 100% CL.

D. Influence of Different RGBA Image Formats

This experiment assesses the compatibility of AlphaDog attack images with five modern RGBA image formats: PNG, TIFF, WebP, SVG, and GIF. The objective is to evaluate how these formats are handled by computer vision models and to ensure the successful execution of AlphaDog attacks when using attack images saved in these formats. We conduct the assessment using 4,000 attack examples of Test Image Data-2), across the five different formats and evaluated them against 100 computer vision models.

Among the 80 open-source AI models examined, we observe that they utilize a variety of frameworks and I/O libraries such as OpenCV, TensorFlow, Pillow, Imageio, and SimpleTik for reading input images. Upon inspecting the source code of these models, we find that only the PNG format is universally accepted. Conversely, the SVG format is unsupported by most models. Additionally, TIFF, WebP, and GIF formats exhibit varying levels of support across different I/O libraries, as summarized in Table IV. Models that support a particular image format achieve a 100% ASR, while those that do not simply respond with an **“input format error”**.

For the 20 cloud-based image recognition systems, since the frameworks are unknown, we conduct experiments to verify if AlphaDog attacks using different image formats could be successful. We craft efficient attack images for the cat-dog example and saved them in five RGBA formats. These images were then fed into the 20 cloud-based systems using the appropriate variants. Similar to open-source models, PNG images, being the most widely supported format, successfully compromised all cloud-based systems with a 100% success rate. The compatibility of different image formats is detailed in Table IV.

Insight 5: PNG emerges as the universally supported format and should be considered the primary option for attackers.

VII. DEFENSE

In Section III-A, we discussed that the root cause of AlphaDog lies in the neglect of Alpha channel information in input RGBA images by AI model designers, which typically either remove it entirely or blend the image uniformly with a white or black background color. To mitigate AlphaDog threats, AI designers have two options. Firstly, they can update their pre-processing code to include a pixel intensity histogram analysis of the input image after the image is read by the I/O library, ensuring its normalcy before further processing. Alternatively, they can develop a separate detector to identify AlphaDog attacks using the same approach of intensity histogram analysis through an independent I/O process. In this case, input images would first pass through the detector,

TABLE IV: Effects of RGBA image formats on AlphaDog success for open-source and cloud-based models. Checkmark (✓) indicates successful AlphaDog attacks with the corresponding format, while cross (×) indicates "format unsupported error".

Model	I/O Library or Model	RGBA Image Format				
		PNG	TIFF	Webp	SVG	GIF
80 open-sourced models shown in Table II	OpenCV	✓	✓	✓	×	✓
	TensorflowIO	✓	×	×	×	✓
	Pillow	✓	✓	✓	×	✓
	Imageio	✓	✓	✓	×	✓
	SimpleTik	✓	✓	×	×	×
20 Cloud-based systems	ChatGPT4	✓	✓	✓	✓	✓
	BaiduImage	✓	×	×	×	✓
	BaiduAPI	✓	×	×	×	×
	FreelImage	✓	×	×	×	×
	Wolfram	✓	✓	✓	✓	✓
	Google Api	✓	✓	✓	×	✓
	Gemini	✓	×	✓	×	×
	Teaching	✓	✓	×	×	✓
	LabelBox	✓	×	×	×	✓
	Nyckel	✓	✓	×	×	×
	MS Azure	✓	✓	✓	×	✓
	AliYun	✓	×	✓	×	✓
	Tencent	✓	×	×	×	×
	Amazon	✓	×	×	×	×
	Bard	✓	×	×	×	×
	GoogleCloud	✓	✓	✓	×	✓
	LandingLens	✓	×	×	×	×
Clarifai	✓	✓	✓	×	✓	
Imagga	✓	×	×	×	×	
Anyline	✓	×	×	×	×	
Vue.ai	✓	×	×	×	×	

and only normal images would proceed to the models' pre-processing phases for tasks like object detection or other computer vision tasks.

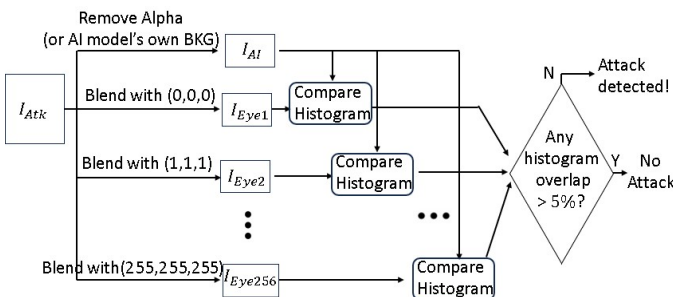


Fig. 4: Diagram illustrating intensity histogram-based detection.

A. Intensity Histogram-Based Detection Framework

Building upon Insight 2, we establish a foundation for detecting AlphaDog attack images using intensity histogram analysis. The defense strategy involves the reconstruction of I_{AI} and I_{Eye} from the input I_{Atk} , followed by the computation of their histogram overlap rate.

To restore I_{AI} , we remove the Alpha Matrix A from I_{Atk} , while I_{Eye} is restored by exhaustively blending I_{Atk} with all possible backgrounds. Since the standard 8-bit intensity level

is widely adopted, we blend I_{Atk} with 256 backgrounds spanning intensities from 0 to 255, yielding 256 I_{Eye} candidate images.

Once I_{Atk} and the 256 I_{Eye} candidates are identified, we conduct 256 pairwise comparisons to calculate the histogram overlap rate. Empirically, we find that an overlap rate lower than 5% indicates the input image is likely an AlphaDog attack image. Figure 4 illustrates the intensity histogram-based detection process, wherein 256 I_{Eye} candidates are generated from the grayscale image. Detection of AlphaDog attacks hinges on the percentage of intensity-overlapping pixels among these pairs, with the detection threshold set at 5% or lower intensity overlap.

It is crucial to note that **even if the attacker intentionally attempts to overlap the histograms of I_{AI} and I_{Eye} by more than 5%, they cannot circumvent our histogram-based detection method.** As explained in Section IV-B and demonstrated in Figure 3, a substantial histogram overlap makes I_{AI} visible to human eyes, undermining the stealthiness of AlphaDog. This inherent contradiction results in the failure of the attack, making it mutually exclusive.

B. Evaluation of AlphaDog Detection

The intensity histogram-based detector operates independently, requiring no modifications to the source code of computer vision AI models. To gauge its effectiveness, we randomly collected 1,000 benign PNG images from online sources like Google Images. For each benign image, we generated I_{AI} and 256 I_{Eye} candidates (I_{Eye1} to I_{Eye256}) by either removing their Alpha channel or creating 256 levels of grayscale background colors. Using the rule-based detection algorithm, all these images were correctly identified as "normal."

Similarly, we treated the 1,000 benign images as normal target images and randomly selected two different images of the same size from Google Images to serve as malicious target images. This process resulted in the creation of 1,000 AlphaDog attack images, with 800 categorized as AlphaDog removal and 200 with solid background attacks, respectively. For each of the 1,000 attack images, we generated I_{AI} and 256 I_{Eye} candidates. Our detector then analyzed the intensity histograms of these 256 image pairs, successfully flagging all of them as AlphaDog attacks.

Figure 5 illustrates the **minimum** pixel intensity overlap percentage among the 256 pairs of images for both the 1,000 benign images and the 1,000 AlphaDog attack examples. While all benign images surpassed the predefined thresholds, the overlaps in AlphaDog attack images were nearly 0%, significantly lower than the threshold. This demonstrates the effectiveness of our defense mechanism, the histogram-based detector, in identifying and mitigating potential AlphaDog attacks on AI models.

VIII. LIMITATIONS

A primary limitation of AlphaDog is its vulnerability confined to grayscale regions within an image. This limitation

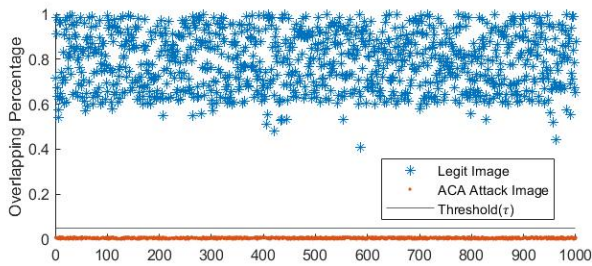


Fig. 5: Intensity histogram-based detection results for 1,000 AlphaDog attack images compared to 1,000 benign images.

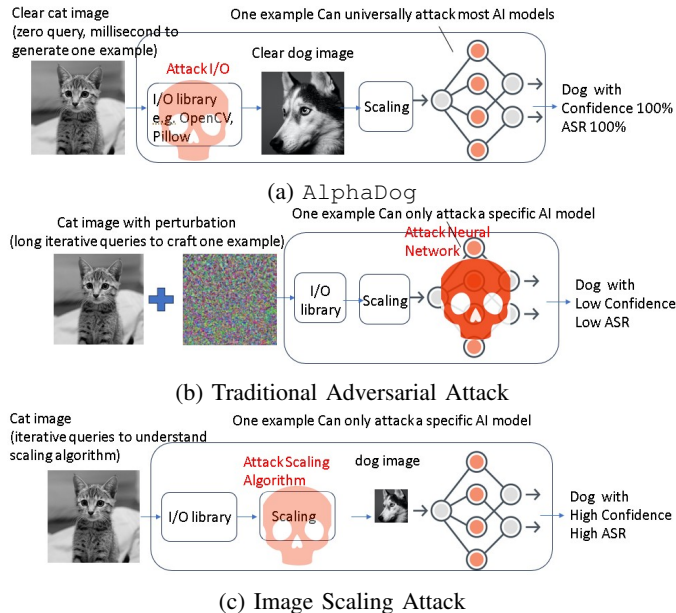


Fig. 6: Comparison of Three Different Attacks: (a) AlphaDog attacks I/O, requiring zero queries, (b) traditional attacks targeting Neural Networks, requiring intensive queries, and (c) Scaling Attacks targeting scaling algorithms, requiring queries to determine scaling algorithm type and scaling ratio.

arises because the Alpha channel can only adjust the overall intensity of each pixel, without affecting the intensity ratio between the RGB channels. However, it is worth noting that grayscale images are extensively utilized in various domains such as traffic signs, medical imagery, and face recognition, thereby ensuring the potential impact of AlphaDog despite this constraint.

IX. RELATED WORK

This section provides an overview of current black-box attacks aimed at compromising the security of computer vision AI models. Furthermore, we delve into AlphaDog, comparing it with other research, particularly the Image Scaling Attack. This comparison highlights AlphaDog universality, ease of deployment, and more potent method for attacking computer vision AI models. Figure 6 shows the comparison between AlphaDog and existing adversarial attacks.

A. Black-box Adversarial Attacks

Recent research has focused on developing image adversarial attacks within a black-box setting [2]–[8]. These attacks typically follow an iterative, query-based methodology, requiring the attacker to perform numerous queries with modified perturbations and have access to the model’s decision-making process for evaluating the impact of perturbations.

Square Attack [2] employs a randomized search scheme with a score-based black-box approach to enhance query efficiency, addressing the challenge of query budget. Boundary Attack [3] minimizes the need for hyperparameter tuning and reduces reliance on substitute models, making it competitive and efficient compared to traditional gradient-based attacks. Chen et al. [4] improve query efficiency by significantly reducing the number of model queries required, making it more efficient for attacking black-box machine-learning models. GenAttack [5] enhances query efficiency through the use of genetic algorithms and gradient-free optimization techniques. Moon et al. [6] introduce the triangle attack, utilizing geometric information to optimize perturbations in a low-frequency space, resulting in a higher attack success rate within a limited query budget. Ilyas et al. [7] leverage adaptive, optimization-focused algorithms such as multi-armed bandits and contextual bandits to efficiently manage exploration-exploitation trade-offs and reduce unnecessary queries across various domains. Simple black-box adversarial attacks (SimBA) [8] adopt a straightforward iterative process, randomly sampling vectors from a predefined orthonormal basis and adjusting them to the target image.

Comparative Analysis: Black-Box Attacks and AlphaDog. Despite advancements in improving query efficiency, conventional black-box adversarial attacks often necessitate a significant number of queries for generating a single adversarial instance. In practical scenarios, frequent queries and access to an online model’s decisions are often unfeasible. For instance, ChatGPT-4, with an average response time of 5 seconds per input image, imposes a limitation of only 25 queries every 3 hours [125]. This renders the attack preparation time-consuming and impractical. Furthermore, in situations where a new AI model or version is in the testing phase and accessibility is restricted to developers or a limited user base, attackers may only play the role of testers, contributing testing data to the development of the new AI models. In such cases, attackers lack access to model decisions, resulting in the failure of black-box attacks. Additionally, black-box adversarial examples are typically model- or framework-specific, meaning one attack instance can only target a particular model.

In contrast, AlphaDog introduces a strong targeted no-box approach, leveraging the shared behavior of AI models in removing the Alpha channel from RGBA images. This approach requires zero queries and does not rely on model outputs, making it simpler yet highly effective across various AI models. However, AlphaDog primarily targets grayscale regions of RGBA images, while recent black-box attacks

impose fewer requirements on image color and format.

B. Image Scaling Attacks

Image Scaling Attacks, demonstrated by Xiao et al. [126] and Qiring et al. [127], exploit vulnerabilities in image scaling algorithms to embed a smaller image within a larger benign image. This technique aims to deceive AI models by concealing the attack image within the benign one. While Image Scaling Attacks have shown effectiveness, they face several challenges.

One major challenge is accurately determining the resizing ratio required for embedding the attack image, which can be time-consuming and difficult, especially in black-box or cloud-based AI models where direct access to the downscaled image is unavailable. Additionally, if the scaling ratio is not large enough, the hidden image might become visible to the user, compromising the attack's stealth. The success of these attacks is also impacted by various image pre-processing actions associated with scaling methods, including cropping, filtering, affine transformations, and color adjustments. Moreover, Image Scaling Attacks are typically model-dependent, meaning crafted attack images can only target specific AI models, requiring attackers to gather basic information about the target model beforehand.

Comparative Analysis: Image Scaling and AlphaDog.

AlphaDog demonstrates distinct advantages over Image Scaling Attacks [126], [127], owing to its no-box approach and universality. Firstly, AlphaDog does not need a scaling step in the AI model's pre-processing pipeline, in contrast to Image Scaling Attacks, which heavily depend on such a step; without it, the attack becomes ineffective. Secondly, AlphaDog exhibits universal scalability across various AI models, allowing the AlphaDog attack image to be universally applied to compromise computer vision models without tailoring it to specific ones. In contrast, Image Scaling Attacks require customized attack images aligned with the scaling algorithm and ratio of each targeted AI model. Thirdly, the time-consuming process of reverse engineering scaling ratios, which poses a significant hindrance to Image Scaling Attacks, does not apply to AlphaDog due to its no-box nature. Lastly, AlphaDog offers enhanced stealth and applicability. AlphaDog is less prone to visibility issues, establishing itself as a more versatile and potent technique for adversarial image manipulation.

X. CONCLUSION

AlphaDog represents a groundbreaking advancement in adversarial attacks, adopting a targeted no-box methodology. By exploiting grayscale regions in RGBA images, AlphaDog deceives AI models into interpreting the attack image as malicious, while displaying a normal image to human observers. Notably, AlphaDog seamlessly operates in the targeted no-box scenario, eliminating the necessity for queries and responses from AI model outputs, thereby showcasing a streamlined yet remarkably effective strategy. The universal adaptability, coupled with its 100% confidence level and ASR

across a diverse array of AI models, underscores the versatility and potency of AlphaDog as a revolutionary technique for adversarial image manipulation.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their comments that guided us in revising the paper. This work was supported by the U.S. Department of Energy/National Nuclear Security Administration (DOE/NNSA) #DE-NA0003985. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency.

REFERENCES

- [1] "Alpha channel attack website." <https://sites.google.com/view/alphachannelattack/home>, 2023. Accessed: 2024-1-10.
- [2] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," in *European conference on computer vision*, pp. 484–501, Springer, 2020.
- [3] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.
- [4] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1277–1294, IEEE, 2020.
- [5] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *Proceedings of the genetic and evolutionary computation conference*, pp. 1111–1119, 2019.
- [6] S. Moon, G. An, and H. O. Song, "Parsimonious black-box adversarial attacks via efficient combinatorial optimization," in *International conference on machine learning*, pp. 4636–4645, PMLR, 2019.
- [7] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," *arXiv preprint arXiv:1807.07978*, 2018.
- [8] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *International Conference on Machine Learning*, pp. 2484–2493, PMLR, 2019.
- [9] "Openai chatgpt." chat.openai.com, 2023. Accessed: 2024-1-10.
- [10] "Free image recognition online." imagerecognize.com, 2023. Accessed: 2024-1-10.
- [11] "Amazon rekognition." aws.amazon.com/pm/rekognition/, 2024. Accessed: 2024-1-10.
- [12] "Bard vision." bard.google.com, 2023. Accessed: 2024-1-10.
- [13] "Google cloud vision api." cloud.google.com/vision, 2023. Accessed: 2024-1-10.
- [14] "Mask rcnn nvidia." github.com/NVIDIA/DeepLearningExamples, 2023. Accessed: 2024-1-10.
- [15] "Mask rcnn facebook." github.com/facebookresearch/maskrcnn-benchmark, 2023. Accessed: 2024-1-10.
- [16] "Mask rcnn matterport." github.com/matterport/Mask_RCNN, 2023. Accessed: 2024-1-10.
- [17] "Mask rcnn tusimple." github.com/TuSimple/mx-maskrcnn, 2023. Accessed: 2024-1-10.
- [18] "Yolov3 Alexeyab." github.com/AlexeyAB/yolo2_light, 2024. Accessed: 2024-1-10.
- [19] "Yolov3 died." github.com/died/YOLO3-With-OpenCvSharp4, 2024. Accessed: 2024-1-10.
- [20] "Yolov3 madhawav." github.com/madhawav/YOLO3-4-Py, 2024. Accessed: 2024-1-10.
- [21] "Yolov3 walktree." github.com/walktree/libtorch-yolov3, 2024. Accessed: 2024-1-10.
- [22] "Yolov3 Xiaochus." github.com/xiaochus/YOLOv3, 2024. Accessed: 2024-1-10.
- [23] "Yolov3 Yu-zhewen." github.com/Yu-Zhewen/Tiny_YOLO_v3_ZYNQ, 2024. Accessed: 2024-1-10.
- [24] "Yolov3 David8862." github.com/david8862/keras-YOLOv3-model-set, 2024. Accessed: 2024-1-10.

- [25] “Yolov3 Mystic123.” github.com/mystic123/tensorflow-yolo-v3, 2024. Accessed: 2024-1-10.
- [26] “Yolov3 Qidian213.” github.com/Qidian213/deep_sort_yolov3, 2024. Accessed: 2024-1-10.
- [27] “Yolov3 Spikeking.” github.com/SpikeKing/keras-yolo3-detection, 2024. Accessed: 2024-1-10.
- [28] “Yolov3 Dataxujing.” github.com/DataXujing/YOLO-V3-Tensorflow, 2024. Accessed: 2024-1-10.
- [29] “Yolov3 heartkilla.” github.com/heartkilla/yolo-v3, 2024. Accessed: 2024-1-10.
- [30] “Yolov3 jasonyip184.” github.com/jasonyip184/yolo, 2024. Accessed: 2024-1-10.
- [31] “Yolov3 Chenyingpeng.” github.com/ChenYingpeng/caffe-yolov3, 2024. Accessed: 2024-1-10.
- [32] “Yolov3 Ayooshkathuria.” github.com/ayoooshkathuria/pytorch-yolo-v3, 2024. Accessed: 2024-1-10.
- [33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.
- [34] “Yolov4 hunglc007.” github.com/hunglc007/tensorflow-yolov4-tflite, 2023. Accessed: 2024-1-10.
- [35] “Yolov5 Megvii.” github.com/Megvii-BaseDetection/YOLOX, 2023. Accessed: 2024-1-10.
- [36] “Yolov5 Dataxujing.” github.com/DataXujing/YOLO-v5, 2023. Accessed: 2024-1-10.
- [37] “Retinanet fizyr.” github.com/fizyr/keras-retinanet, 2023. Accessed: 2024-1-10.
- [38] “Retinanet nvidia.” github.com/NVIDIA/retinanet-examples, 2023. Accessed: 2024-1-10.
- [39] “Retinanet yhenon.” github.com/yhenon/pytorch-retinanet, 2023. Accessed: 2024-1-10.
- [40] “Retinanet kuangliu.” github.com/kuangliu/pytorch-retinanet, 2023. Accessed: 2024-1-10.
- [41] “Centernet Caowgg.” github.com/CaoWGG/TensorRT-CenterNet, 2023. Accessed: 2024-1-10.
- [42] “Centernet Duankaiwen.” github.com/Duankaiwen/CenterNet, 2023. Accessed: 2024-1-10.
- [43] “Centernet xingyizhou.” github.com/xingyizhou/CenterNet, 2023. Accessed: 2024-1-10.
- [44] “Centernet xingyizhou2.” github.com/xingyizhou/CenterNet2, 2023. Accessed: 2024-1-10.
- [45] “Efficientdet zylo117.” github.com/zylo117/Yet-Another-EfficientDet-Pytorch, 2024. Accessed: 2024-1-10.
- [46] “Efficientdet signatrix.” github.com/signatrix/efficientdet, 2024. Accessed: 2024-1-10.
- [47] “Efficientdet rwightman.” github.com/rwightman/efficientdet-pytorch, 2024. Accessed: 2024-1-10.
- [48] “Efficientdet xuannianz.” github.com/xuannianz/EfficientDet, 2024. Accessed: 2024-1-10.
- [49] “Efficientdet toandaominh1997.” github.com/toandaominh1997/EfficientDet.Pytorch, 2024. Accessed: 2024-1-10.
- [50] “Cascadernn haoweicai.” github.com/haoweicai/Detector-Cascade-RCNN, 2024. Accessed: 2024-1-10.
- [51] “Cascadernn ruoqianguo.” github.com/ruoqianguo/cascade-rcnn_Pytorch, 2024. Accessed: 2024-1-10.
- [52] X. Li, T. Wei, Y. P. Chen, Y.-W. Tai, and C.-K. Tang, “Fss-1000: A 1000-class dataset for few-shot segmentation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2869–2878, 2020.
- [53] “Fasterrcnnvgg16 fengkaibit.” github.com/fengkaibit/faster-rcnn_vgg16, 2024. Accessed: 2024-1-10.
- [54] “Fasterrcnnvgg16 jwyang.” github.com/jwyang/faster-rcnn.pytorch, 2024. Accessed: 2024-1-10.
- [55] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [56] “Vgg16 ry.” github.com/ry/tensorflow-vgg16, 2024. Accessed: 2024-1-10.
- [57] “Vgg16 machrisaa.” github.com/machrisaa/tensorflow-vgg, 2024. Accessed: 2024-1-10.
- [58] “Repvgg Dingxiaoh.” github.com/DingXiaoH/RepVGG, 2024. Accessed: 2024-1-10.
- [59] “Yolov1 abeardear.” github.com/abeardear/pytorch-YOLO-v1, 2024. Accessed: 2024-1-10.
- [60] “Yolov2 longcw.” github.com/longcw/yolo2-pytorch, 2024. Accessed: 2024-1-10.
- [61] “Alexnet kratzert.” github.com/kratzert/finetune_alexnet_with_tensorflow, 2024. Accessed: 2024-1-10.
- [62] “Alexnet Dynmi.” github.com/Dynmi/AlexNet, 2024. Accessed: 2024-1-10.
- [63] “Alexnet songhan.” github.com/songhan/Deep-Compression-AlexNet, 2024. Accessed: 2024-1-10.
- [64] “Alexnet udacity.” github.com/udacity/CarND-Alexnet-Feature, 2024. Accessed: 2024-1-10.
- [65] “Alexnet uoguelph-mlrg.” github.com/uoguelph-mlrg/theano_alexnet, 2024. Accessed: 2024-1-10.
- [66] “Resnet akamaster.” github.com/akamaster/pytorch_resnet_cifar10, 2024. Accessed: 2024-1-10.
- [67] “Resnet facebookarchive.” github.com/facebookarchive/fb.resnet.torch, 2024. Accessed: 2024-1-10.
- [68] “Resnet kenshohara.” github.com/kenshohara/3D-ResNets-PyTorch, 2024. Accessed: 2024-1-10.
- [69] “Resnet raghakot.” github.com/raghakot/keras-resnet, 2024. Accessed: 2024-1-10.
- [70] “Resnet ry.” github.com/ry/tensorflow-resnet, 2024. Accessed: 2024-1-10.
- [71] “Resnet tornadomeet.” github.com/tornadomeet/ResNet, 2024. Accessed: 2024-1-10.
- [72] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
- [73] “Efficientnet lukemelas.” github.com/lukemelas/EfficientNet-PyTorch, 2024. Accessed: 2024-1-10.
- [74] “Efficientnet qubvel.” github.com/qubvel/efficientnet, 2024. Accessed: 2024-1-10.
- [75] “Efficientnet rwightman.” github.com/rwightman/gen-efficientnet-pytorch, 2024. Accessed: 2024-1-10.
- [76] “Efficientnetv2 d-li14.” github.com/d-li14/efficientnetv2.pytorch, 2024. Accessed: 2024-1-10.
- [77] “Inceptionv3 smichalowski.” github.com/smichalowski/google_inception_v3_for_caffe, 2024. Accessed: 2024-1-10.
- [78] “Inception v4 Cadene.” github.com/Cadene/tensorflow-model-zoo_torch, 2024. Accessed: 2024-1-10.
- [79] “Googlenet inception conan7882.” github.com/conan7882/GoogLeNet-Inception, 2024. Accessed: 2024-1-10.
- [80] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [81] “Lenet5 changwoolee.” github.com/changwoolee/lenet5_hls, 2024. Accessed: 2024-1-10.
- [82] “Lenet ganyc717.” github.com/ganyc717/LeNet, 2024. Accessed: 2024-1-10.
- [83] V. Jain and E. Learned-Miller, “Fddb: A benchmark for face detection in unconstrained settings,” Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst, 2010.
- [84] “Facenet guanfuchen.” github.com/guanfuchen/facenet, 2024. Accessed: 2024-1-10.
- [85] “Face detection layumi.” github.com/layumi/2015_Face_Detection, 2024. Accessed: 2024-1-10.
- [86] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [87] “Monodepth haofengac.” github.com/haofengac/MonoDepth-FPN-PyTorch, 2024. Accessed: 2024-1-10.
- [88] “Monodepth mrharicot.” github.com/mrharicot/monodepth, 2024. Accessed: 2024-1-10.
- [89] “Monodepth nianticlabs.” github.com/nianticlabs/monodepth2, 2024. Accessed: 2024-1-10.
- [90] “Monodepth Oniroai.” github.com/OniroAI/MonoDepth-PyTorch, 2024. Accessed: 2024-1-10.
- [91] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, “Bdd100k: A diverse driving dataset for heterogeneous multitask learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2636–2645, 2020.
- [92] “Yolov3 sanwong15.” github.com/sanwong15/yolo-v3-with-bdd, 2024. Accessed: 2024-1-10.

- [93] “Yolov3 yogeshgajjar.” github.com/yogeshgajjar/BDD100k-YOLOV3-tiny, 2024. Accessed: 2024-1-10.
- [94] “Yolov5 williamhyin.” github.com/williamhyin/yolov5_bdd100k, 2024. Accessed: 2024-1-10.
- [95] S. Yang, P. Luo, C. C. Loy, and X. Tang, “Wider face: A face detection benchmark,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [96] “Facenet davidsandberg.” github.com/davidsandberg/facenet, 2024. Accessed: 2024-1-10.
- [97] “yoloface.” github.com/sthanhg/yoloface, 2024. Accessed: 2024-1-10.
- [98] “yoloface2.” github.com/swdev1202/keras-yolo3-facedetection, 2024. Accessed: 2024-1-10.
- [99] “yoloface3.” github.com/zlmo/Face-Detection, 2024. Accessed: 2024-1-10.
- [100] “yoloface3.” github.com/azmathmoosa/azFace, 2024. Accessed: 2024-1-10.
- [101] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images.” Technical report, University of Toronto, 2009. Accessed: [Insert date of access].
- [102] “Resnet cifar-10 akamaster.” github.com/akamaster/pytorch_resnet_cifar10, 2024. Accessed: 2024-1-10.
- [103] “Gemini pro vision api.” ai.google.dev/docs, 2023. Accessed: 2024-1-10.
- [104] “Baidu graph recognition.” graph.baidu.com/, 2023. Accessed: 2024-1-10.
- [105] “Baidu vision api.” ai.baidu.com/tech/imagerecognition/fine_grained, 2024. Accessed: 2024-1-10.
- [106] “Teachable machine.” teachablemachine.withgoogle.com, 2023. Accessed: 2024-1-10.
- [107] “Nyckel.” nyckel.com, 2023. Accessed: 2024-1-10.
- [108] “Labelbox.” labelbox.com, 2023. Accessed: 2024-1-10.
- [109] “Wolfram image identification project.” <https://www.imageidentify.com/>, 2023. Accessed: 2024-1-10.
- [110] “vue.ai.” <https://vue.ai/>.
- [111] “Microsoft azure.” azure.microsoft.com/en-us/services/cognitive-services/content-moderator/, 2023. Accessed: 2024-1-10.
- [112] “Alibaba cloud vision api.” <https://vision.aliyun.com/>.
- [113] “Tencent vision.” <https://cloud.tencent.com/>.
- [114] “Landing lens.” landing.ai/platform, 2024. Accessed: 2024-1-10.
- [115] “Clarifai.” clarifai.com, 2024. Accessed: 2024-1-10.
- [116] “Imagga image recognition api.” imagga.com.
- [117] “Anyline.” <https://anyline.com/>.
- [118] 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, IEEE, 2016.
- [119] J. C. Costa, T. Roxo, H. Proença, and P. R. Inácio, “How deep learning sees the world: A survey on adversarial attacks & defenses,” *arXiv preprint arXiv:2305.10862*, 2023.
- [120] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [121] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Watenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems.” <https://www.tensorflow.org/>, 2015. Software available from tensorflow.org.
- [122] A. Clark, “Pillow (pil fork) documentation.” buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf, 2015.
- [123] “Mxnet.” docs.aws.amazon.com/sagemaker/latest/dg/image-classification.html, 2023. Accessed: 2024-1-10.
- [124] “ModerateContent.” moderatecontent.com.
- [125] “Chatgpt4 Cap Limit.” community.openai.com/t/less-than-40-requests-for-paid-service-is-not-acceptable-gpt-4/105424.
- [126] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, “Seeing is not believing: Camouflage attacks on image scaling algorithms,” in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 443–460, 2019.
- [127] E. Quiring, D. Klein, D. Arp, M. Johns, and K. Rieck, “Adversarial preprocessing: Understanding and preventing {Image-Scaling} attacks in machine learning,” in *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1363–1380, 2020.
- [128] “Sharpness aware minimization tensorflow.” github.com/sayakpaul/Sharpness-Aware-Minimization-TensorFlow/blob/main/SAM.ipynb, 2023. Accessed: 2024-1-10.
- [129] “Stack overflow use pillow to read a png.” stackoverflow.com/a/33618483.

APPENDIX

A. Supplementary figures for Section III-A1

Figure 7 shows code samples discussed in Section III-A1.

```
for file in os.listdir(DIRNAME):
    img = cv2.imread(f"{DIRNAME}/{file}")
    test_images_size_tups.append(img.shape)
```

(a)

```
def augment(image, label):
    image = tf.image.resize_with_crop_or_pad(image, 40, 40) # Add 8 pixels of padding
    image = tf.image.random_crop(image, sizes=[32, 32, 3]) # Random crop back to 32x32
    image = tf.image.random_brightness(image, max_delta=0.5) # Random brightness
    image = tf.clip_by_value(image, 0., 1.)
```

(b)

▲ Sounds like the image is being opened in grayscale mode. Try converting to RGB before accessing the pixel values.

11

```
img = Image.open(path).convert("RGB")
pixels = img.load()
```

🔖 Share Improve this answer Follow

answered Nov 9, 2015 at 21:18

Kevin

75.3k ● 12 ● 135 ● 166

(c)

Fig. 7: Illustration of Alpha channel neglect: (a) OpenCV default reading only RGB channels from a Kaggle OpenCV tutorial [120]; (b) TensorFlow code extracting only RGB channels from [128], the best benchmark for EffNET-L2 CIFAR-100; (c) Stack Overflow Q&A [129] suggesting developers use Pillow [122], ignoring the Alpha channel.

B. Element-wise Operation

A and B should be of the same size. Let

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

then the **element-wise multiplication** of A and B is given by

$$A \circ B = \begin{bmatrix} a_{11} \times b_{11} & a_{12} \times b_{12} \\ a_{21} \times b_{21} & a_{22} \times b_{22} \end{bmatrix}$$

element-wise division of A and B is given by

$$A \oslash B = \begin{bmatrix} a_{11}/b_{11} & a_{12}/b_{12} \\ a_{21}/b_{21} & a_{22}/b_{22} \end{bmatrix}$$

element-wise addition/subtraction of A and B is given by

$$A \pm B = \begin{bmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} \end{bmatrix}$$

, especially if A is a constant matrix with every element value be 1 ($A = 1$), then we can rewrite it as:

$$1 \pm B = \begin{bmatrix} 1 \pm b_{11} & 1 \pm b_{12} \\ 1 \pm b_{21} & 1 \pm b_{22} \end{bmatrix}$$

C. Investigate AI models' Alpha Channel Processing Strategy

As shown in Figure II, most AI models remove the Alpha channel. However, some outliers use backgrounds instead. To determine how an AI model processes the Alpha channel, an attacker can upload a transparent image with the RGB channel set to red. If the AI model responds with "Red," it indicates that the model removes the input image's Alpha channel values. If it responds with "White" or "Black," it implies the model blends a "White" or "Black" background color to the input image, respectively.

D. Utilizing Background Colors Difference to Develop Target-Adaptive AlphaDog Attack Images

As discussed earlier and outlined in Table II, modern AI models treat the Alpha channel of RGBA images in three ways: either by removing the Alpha channel or blending the image with a white or black background. This can be mathematically represented as:

$$BKG_{AI} = \{ \text{"Removal"}, \text{"White"}, \text{"Black"} \}, \quad (10)$$

where BKG_{AI} denotes how AI models treat the image's Alpha channel, also referred to as the AI model background color. Image media can be further classified into Thumbnails and Image Viewers. According to our analysis and survey shown in Table I, popular Thumbnails typically employ a white background:

$$BKG_{TB} = \{ \text{"White"} \}. \quad (11)$$

Image Viewers (BKG_{IV}) background colors vary. Most Image Viewers opt for a white background, with some using backgrounds of different gray levels, and a few employing a black background:

$$BKG_{IV} = \{ \text{"White"}, \text{"Gray"}, \text{"Black"} \} \quad (12)$$

It is critical that the background colors of the image media BKG_{Eye} , including both the Image Viewer (BKG_{IV}) and Thumbnails (BKG_{TB}), differ from those used by the AI model (BKG_{AI}), to ensure that the victim user perceives the normal target image I_{Eye} while the AI model interprets the malicious target image I_{AI} . Notably, Thumbnails and Image Viewers may not always have the same default background color. For example, Thumbnail macOS Finder has a white background, whereas Image Viewer Mac Preview has a gray background. The Thumbnail and Image Viewer backgrounds

are not the same as the AI background's mathematical representation, which is:

$$BKG_{AI} \cap (BKG_{TB} \cup BKG_{IV}) = \emptyset. \quad (13)$$

By considering Equation 13 and all feasible values for sets BKG_{AI} , BKG_{TB} , and BKG_{IV} , we enumerate combinations meeting the criteria, denoted by "Re" for "Removal," "W" for "White," Gr for "Gray," and "Bk" for "Black." Two distinct representatives emerge: Equation 14 mirrors real-world scenarios involving operating systems and default background colors of image media, detailed in Table I.

$$BKG_{AI} \times BKG_{TB} \times BKG_{IV} = \{ (Re, W, W), (Re, W, Gr), (Re, W, Bk), (Bk, W, W), (Bk, W, Gr) \} \quad (14)$$

Conversely, Equation 15 explores potential combinations, considering customized image media background colors such as "Black," as well as theoretically satisfied background color combinations not observed in real-world image media apps.

$$BKG_{AI} \times BKG_{TB} \times BKG_{IV} = \{ (Re, Gr, W), (Re, Gr, Gr), (Re, Bk, Bk), (Re, Gr, Bk), (Re, Bk, W), (Re, Bk, Gr), (W, Bk, Gr), (W, Gr, Gr), (W, Gr, Bk), (W, Bk, Bk), (Bk, Gr, W), (Bk, Gr, Gr) \} \quad (15)$$

1) *Creating Pure Grayscale AlphaDog Attack Images:* Attackers have the flexibility to choose both the normal target image (I_{Eye}) and the malicious target image (I_{AI}), ensuring they share dimensions of $M \times N$. To create the attack image I_{Atk} , its RGB color values (I_{RGB}) and the same-dimensional transparency matrix of the Alpha channel values (A) should be calculated. Figure 8 (a) illustrates I_{Atk} 's creation process.

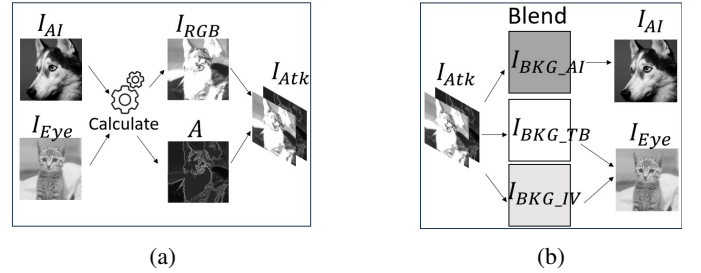


Fig. 8: Illustration of (a) attack Image I_{Atk} creation process, and (b) how I_{Atk} blends with three different backgrounds to achieve AlphaDog.

Figure 8 (b) presents how AlphaDog creates a perceptual contradiction between humans and machines, with AI models recognizing I_{AI} while humans perceive I_{Eye} . AlphaDog attack images I_{Atk} are blended with different BKG_{AI} colors based on the AI models they target. These attack images are blended with BKG_{TB} and BKG_{IV} background colors viewed by human observers (I_{Eye}) through Thumbnail or Image Viewer apps, denoted as I_{TB} and I_{IV} , respectively. The

three background color combinations must adhere to the conditions specified in Equations 14 and 15, resulting in blended attack images I_{BKG_AI} , I_{BKG_TB} , and I_{BKG_IV} , respectively.

Each pixel in images I_{AI} , I_{EYe} , I_{BKG_AI} , I_{BKG_TB} , and I_{BKG_IV} is represented by i_{AI} , i_{EYe} , i_{BKG_AI} , i_{BKG_TB} , and i_{BKG_IV} , respectively. Similarly, i_{Atk} denotes each pixel in the attack image I_{Atk} , comprising the RGB value i_{RGB} and the Alpha channel value α of each pixel. To ensure consistent perception of the same normal target image I_{EYe} by human observers, regardless of whether attack images are displayed by Thumbnails or Image Viewers, the following equation must be satisfied:

$$I_{EYe} = I_{TB} \times I_{IV}. \quad (16)$$

This condition is crucial to prevent AlphaDog attacks from being detected by vigilant users. It is worth noting that each pixel i of the input image has been normalized, indicating that $0 \leq i \leq 1$.

Pixel-level Alpha Compositing. As the AI model treats the input image's Alpha channel differently, the pixel-level value of the malicious target image i_{AI} varies. If the victim AI model removes the Alpha channel ($BKG_{AI} = Re$), the attack image's RGB value i_{RGB} and the malicious target image i_{AI} are identical:

$$i_{AI} = i_{RGB}. \quad (17)$$

If the victim AI models blend a background color to the input image ($BKG_{AI} = \{W, Bk\}$), i_{AI} is expressed as:

$$i_{AI} = \alpha \cdot i_{RGB} + i_{BKG_AI}(1 - \alpha). \quad (18)$$

The normal target image I_{EYe} always blends with the image media's background colors I_{BKG_EYe} , including both Thumbnails' and Image Viewers' (I_{BKG_EYe} represents either I_{BKG_TB} or I_{BKG_IV}), and each pixel i_{EYe} can be expressed as:

$$i_{EYe} = \alpha \cdot i_{RGB} + i_{BKG_EYe}(1 - \alpha). \quad (19)$$

By combining Equations 17 and 19, we can solve for α when the AI model removes the Alpha channel:

$$\alpha = \frac{i_{EYe} - i_{BKG_EYe}}{i_{AI} - i_{BKG_EYe}}. \quad (20)$$

Additionally, for AI models that blend a background color with the input image, we can determine the values of α and i_{RGB} for this case by combining Equations 18 and 19 as:

$$\alpha = 1 - \frac{i_{AI} - i_{EYe}}{i_{BKG_AI} - i_{BKG_EYe}} \quad (21)$$

$$i_{RGB} = \frac{i_{AI} \cdot i_{BKG_EYe} - i_{EYe} \cdot i_{BKG_AI}}{i_{BKG_AI} - i_{BKG_EYe} - i_{AI} + i_{EYe}}. \quad (22)$$

For a comprehensive overview, the values of i_{RGB} and α corresponding to all background combinations outlined in Equations 14 and 15 are provided in Table V.

In the following, we explore the mathematical process of crafting AlphaDog attack images for the strong no-box settings when BKG_{AI} is specified as "Re." In this scenario,

the background colors of Thumbnails or Image Viewers can be either the same or different. We illustrate both cases using the following two background color combinations as examples:

$$BKG_{AI} \times BKG_{TB} \times BKG_{IV} = (Re, W, W), \quad (23)$$

and

$$BKG_{AI} \times BKG_{TB} \times BKG_{IV} = (Re, W, Gr) \quad (24)$$

Case 1 (Equation 23). In this combination, the AI model removes the input image's Alpha channel information, and thumbnail shares the same background with the image viewer. Consequently, Equations 17 and 20 should be used. As both image media background colors are "W," indicating $i_{BKG_TB} = i_{BKG_IV} = 1$. Therefore, we can determine α as:

$$\alpha = \frac{i_{EYe} - 1}{i_{AI} - 1}. \quad (25)$$

Given that image transparency ranges between 0 and 1 ($0 \leq \alpha \leq 1$), according to Equation 25, we deduce that the malicious target image should be darker than the normal target image:

$$0 \leq i_{AI} \leq i_{EYe} \leq 1, \quad (26)$$

where 0 represents black and 1 signifies white. After acquiring $M \times N$ matrices (or pixels) of I_{RGB} and A , the attack image I_{Atk} can be created by combining I_{RGB} and A .

Case 2 (Equation 24). Similar to Case 1, each pixel's RGB color of the attack image is the same as the malicious target image's (refer to Equation 17). In contrast to Case 1, where image media backgrounds are identical, Case 2 introduces attack images blended and displayed with distinct background colors for Thumbnails and Image Viewers. This difference can potentially alert careful users to anomalies or even the presence of a malicious target image. To ensure that only the normal target image is perceptible to vigilant users, as demonstrated in Equation 16, we derive the following two equations according to Equation 19:

$$i_{EYe} = i_{TB} = \alpha \cdot i_{AI} + (1 - \alpha), \quad (27)$$

and

$$i_{EYe} \times i_{IV} = \alpha \cdot i_{AI} + i_{BKG_IV}(1 - \alpha). \quad (28)$$

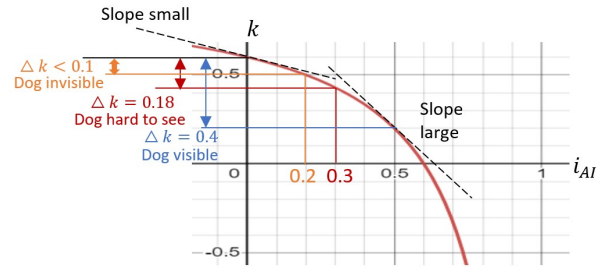


Fig. 9: Graph of k function.

We first ensure the satisfaction of Equation 27 to make the attack image viewed in Thumbnails I_{TB} identical to

TABLE V: Creation of attack image (I_{Atk}) with various background combinations of AI models, Thumbnails, and Image Viewers, as mentioned in Equations 14 and 15. Each cell displays the expression for calculating i_{RGB} , α , and how to adjust i_{AI} to ensure consistent visualization of the normal target image by different Thumbnails and Image Viewers.

BKG_{AI} $\times BKG_{TB}$ $\times BKG_{IV} =$	Attack Image		Additional constraint to ensure $i_{Eye} \propto i_{IV}$
	$i_{RGB} =$	$\alpha =$	
("Re", "W", "W")	i_{AI}	$\frac{i_{Eye}-1}{i_{AI}-1}$	N/A
("Re", "Bk", "Bk")		$\frac{i_{Eye}}{i_{AI}}$	N/A
("Re", "Gr", "Gr")		$\frac{i_{Eye}-i_{BKG_{TB}}}{i_{AI}-i_{BKG_{TB}}}$	$\frac{i_{AI}-i_{BKG_{IV}}}{i_{AI}-i_{BKG_{TB}}} \approx const$ $\frac{i_{AI} \cdot i_{BKG_{IV}} - i_{BKG_{TB}}}{i_{AI}-i_{BKG_{TB}}} \approx const$
("Re", "W", "Gr") ("Re", "Gr", "W")		$\frac{i_{Eye}-1}{i_{AI}-1}$	$\frac{i_{AI}-i_{BKG_{IV}}}{i_{AI}-1} \approx const$
("Re", "Bk", "Gr") ("Re", "Gr", "Bk")		$\frac{i_{Eye}}{i_{AI}}$	$1 - \frac{i_{BKG_{IV}}}{i_{AI}} \approx const$
("Re", "W", "Bk") ("Re", "Bk", "W")	Failure: Mathematics yields no solution. The only real-world scenario where AlphaDog deception may fail is if victim users utilize iPhone Photos as their image viewer while the targeted AI models remove the Alpha channel.		
("W", "Bk", "Bk")	$\frac{-i_{Eye}}{1-i_{AI}+i_{Eye}}$	$1 - i_{AI} + i_{Eye}$	N/A
("W", "Bk", "Gr") ("W", "Gr", "Bk")			$i_{AI} \cdot i_{BKG_{IV}} \approx const$
("W", "Gr", "Gr")	$\frac{i_{AI} \cdot i_{BKG_{TB}} - i_{Eye}}{1 - i_{BKG_{TB}} - i_{AI} + i_{Eye}}$	$1 - \frac{i_{AI} - i_{Eye}}{1 - i_{BKG_{TB}}}$	$\alpha \cdot i_{RGB} + i_{BKG_{IV}}(1-\alpha) \propto i_{Eye}$
("Bk", "W", "W")	$\frac{i_{AI}}{i_{Eye}-i_{AI}-1}$	$i_{AI} - i_{Eye}$	N/A
("Bk", "W", "Gr") ("Bk", "Gr", "W")			$(1 - i_{BKG_{IV}}) \cdot i_{AI} \approx const$
("Bk", "Gr", "Gr")	$\frac{i_{AI} \cdot i_{BKG_{TB}}}{-i_{BKG_{TB}} - i_{AI} + i_{Eye}}$	$1 + \frac{i_{AI} - i_{Eye}}{i_{BKG_{TB}}}$	$1 - \frac{i_{BKG_{IV}}}{i_{AI}} \approx const$

the normal target image I_{Eye} , which we obtain the same α expression as Case1 Equation 25. Subsequently, we replace α in Equation 28 by using Equation 25, and express i_{IV} in terms of i_{Eye} as:

$$i_{IV} = k(i_{Eye} - 1) + i_{BKG_{IV}}, \quad (29)$$

where

$$k = \frac{i_{AI} - i_{BKG_{IV}}}{i_{AI} - 1}. \quad (30)$$

We note that the term k (Equation 30) is not a constant. Therefore, the relationship between i_{IV} and i_{Eye} cannot be linear unless k is independent of changes in i_{AI} , i.e., $\Delta k \approx 0$. Examining the curve of k 's function plotted in Figure 9, we observe that the slope decreases when i_{AI} is closer to 0. Consequently, a slight decrease in i_{AI} significantly reduces Δk , making k approach a constant value.

By crafting attack images using different values and visualizing them through image media with different background colors, we determine that when $0 < i_{AI} < 0.2$, $\Delta k < 0.1$, and i_{AI} becomes invisible to human observers. Figure 10 (a-c) illustrates I_{IV} generated with different I_{AI} pixel intensity ranges, where the normal target image (Cat) is clearer, while the malicious target image (Dog) is concealed and cannot be visualized when $0 < i_{AI} < 0.2$.

Limitation in Deceiving Human Observers: Table V illustrates instances where AlphaDog fails to deceive hu-

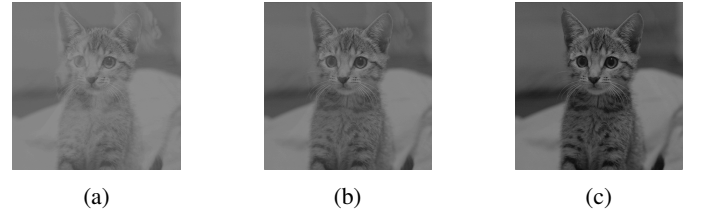


Fig. 10: Illustration of AlphaDog Case 2 attack image generation, concealing a malicious target image I_{AI} (dog) within a normal target image I_{Eye} (cat) to prevent detection by human observers. The attack image is viewed using macOS' default Image Viewer *Preview*, with a background color intensity set to 0.6 ("Gray"). Different I_{AI} intensity ranges are presented: (a) $0 \leq I_{AI} \leq 0.5$, revealing a silhouette of the dog (Husky); (b) $0 \leq I_{AI} \leq 0.3$, significantly reducing the visibility of the dog image; (c) $0 \leq I_{AI} \leq 0.2$, rendering the dog image completely invisible.

man observers due to the absence of mathematical solutions. Specifically, when the background color combinations are $BKG_{AI} \times BKG_{TB} \times BKG_{IV} = \{(\text{Re}, \text{W}, \text{Bk}), (\text{Re}, \text{Bk}, \text{W})\}$. A potential real-world scenario arises if users use customized backgrounds to view the images.