CAPatch: Physical Adversarial Patch against Image Captioning Systems

Shibo Zhang
USSLAB, Zhejiang University

Yushi Cheng
BNRist, Tsinghua University

Wenjun Zhu
USSLAB, Zhejiang University

Xiaoyu Ji*
USSLAB, Zhejiang University

Wenyuan Xu
USSLAB, Zhejiang University

Abstract

The fast-growing surveillance systems will make image captioning, *i.e.*, automatically generating text descriptions of images, an essential technique to process the huge volumes of videos efficiently, and correct captioning is essential to ensure the text authenticity. While prior work has demonstrated the feasibility of fooling computer vision models with adversarial patches, it is unclear whether the vulnerability can lead to incorrect captioning, which involves natural language processing after image feature extraction. In this paper, we design CAPatch, a physical adversarial patch that can result in mistakes in the final captions, *i.e.*, either create a completely different sentence or a sentence with keywords missing, against multi-modal image captioning systems. To make CAPatch effective and practical in the physical world, we propose a detection assurance and attention enhancement method to increase the impact of CAPatch and a robustness improvement method to address the patch distortions caused by image printing and capturing. Evaluations on three commonly-used image captioning systems (Show-and-Tell, Self-critical Sequence Training: Att2in, and Bottom-up Top-down) demonstrate the effectiveness of CAPatch in both the digital and physical worlds, whereby volunteers wear printed patches in various scenarios, clothes, lighting conditions. With a size of 5% of the image, physically-printed CAPatch can achieve continuous attacks with an attack success rate higher than 73.1% over a video recorder.

1 Introduction

With the proliferation of cameras, at least 2.5 petabytes of images are generated daily, manually searching for useful information from such a vast amount of images is almost impossible. To facilitate image retrieval, captioning systems automatically convert the content of an image utilizing computer vision and natural language processing [11, 24]. For instance, captioning systems can help to identify evidence from the huge volumes of surveillance videos efficiently [6, 33] or can help those with visual impairments by transforming images/audios into texts and communicating via text-to-speech technology [13, 14, 17, 31, 41]. As a result, correctly captioning amid a dedicated adversary is important to ensure the security of those applications. If the adversary can manipulate the captioning results of the video surveillance systems by wearing an adversarial patch, *e.g.*, altering *"A man is opening the door"* into *"A bird is flying over a body of water"* or *"The wind is blowing the door"*, she can cause undesired consequences, as shown in Fig. 1.

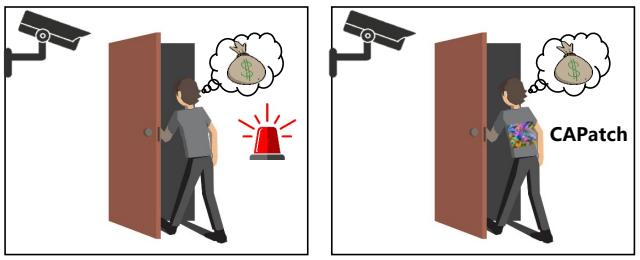


Figure 1: CAPatch attacks. The adversary fools the video surveillance systems by wearing a CAPatch on the body to enable further malicious behaviors such as break-in.

Since captioning systems utilize AI-based computer vision algorithms to detect objects in the images, it is natural to ask whether the adversarial patch designed purely for computer vision algorithms, *e.g.*, image classifiers [5, 18, 39], object detectors [9, 20, 28, 32], or face recognition models [1, 34, 38], *etc*., can output a caption irrelevant to the image content at all. Our preliminary analysis show that the patches optimized to fool computer vision can at most cause partial caption modification, but can hardly alter enough keywords to hide malicious behaviors, *e.g.*, converting the original caption of "*A group of people on skis in the snow*" to "*A person is sitting on a snowy hill with a cat in the background*". This is because the captioning systems consist of not only a feature extractor to identify objects in the images, but also a description generator to output captions utilizing natural language processing algo-

*Xiaoyu Ji is the corresponding author.
†Source code & demo: https://github.com/USSLab/CAPatch

rithms. Existing patches against computer vision algorithms can at most incorrectly identify the objects in the first step yet may not affect the final output of the description generator.

Existing work cannot affect natural language processing of the description generator for the following reasons. First, state-of-the-art image captioning systems utilize Faster R-CNN to detect objects and extract features of the regions with salient objects, which are in turn fed into the description generator. If a patch is not located within the region chosen by the Faster R-CNN model, the generated caption will not be affected. Yet, the patch is supposed to be placed anywhere in the images. Second, even if the patch is always located within the chosen region, the description generator utilizes an attention mechanism to assign higher weights to the region of interest. A patch may not have much impact if it does not overlap with the region of high attention. Third, granted that a patch contains features of target objects, which may be insufficient to affect the entire output of the description generator and hide the rest of the original objects in the image.

In this paper, we propose `CAPatch`, an adversarial patch to be worn outside a jacket, which can cause the image captioning system to output a caption that has nothing to do with the original image (*i.e.*, caption makeover attacks) or a caption with important keywords modified (*i.e.*, keyword hiding attacks), by exploiting vulnerabilities from both the image feature extractor and description generator. To overcome the aforementioned challenges, `CAPatch` is designed with detection assurance, attention enhancement, caption alteration, and robustness improvement mechanisms. As such, `CAPatch` can make Fast R-CNN based object detection algorithms select a collection of overlapping regions with each containing `CAPatch`. As a result, it increases the impact of `CAPatch` in terms of extracted features and causes the attention module to output high attention levels over the regions overlapping with `CAPatch`, regardless of where it is placed. Moreover, to output a chosen caption or to hide keywords, `CAPatch` is optimized based on a caption loss target function that incorporates the vulnerabilities of both the feature extractor and description generator. Finally, to make `CAPatch` effective in practice, we employ color smoothing and expectation over transformation to overcome the noises and distortion introduced during the image capturing process. We validate the effectiveness of `CAPatch` by conducting both simulation and real-world experiments on three popular image captioning systems, *i.e.*, Show-and-Tell, Self-critical Sequence Training: Att2in, and Bottom-up Top-down. The evaluation involves volunteers attaching a printed `CAPatch` on their jackets, and achieves continuous attacks with a success rate of 73.1% for caption makeover attacks and 92.4% for keyword hiding attacks, with a patch size of 5% of the images.

In summary, our contributions include the points below:

- To the best of our knowledge, this is the first work on the physical adversarial patch against image captioning systems with the goal to output a chosen caption or to hide keywords.
- We design `CAPatch` that utilizes the workflow of captioning systems, *e.g.*, feature extractors and description generators, such that the patch will affect both stages and output a chosen caption irrelevant to the original images or videos.
- We evaluate the performance of `CAPatch` with three image captioning systems (Show-and-Tell, Self-critical Sequence Training: Att2in, and Bottom-up Top-down) in both digital and physical worlds. The results demonstrate that physically-printed `CAPatchs` can work with various light conditions, distances, cameras, resolutions, etc.

2 Background and Related Work

In this section, we introduce image captioning systems and present the related work in adversarial attacks against image captioning systems as well as physical adversarial patches.

2.1 Image Captioning

An image captioning system is a deep learning-based system that aims to generate a text description for an image. As shown in Fig. 2, the image captioning system usually adopts an encoder-decoder architecture since image captioning is a multi-modal task based on both computer vision for image feature extraction and natural language processing for description generation. The computer version side. *a.k.a.*, encoder, is a feature extractor that encodes the input image into an intermediate representation. The natural language processing side, *a.k.a.*, decoder, is a language model that decodes the extracted image features into a descriptive sentence as the output. In the following, we introduce the feature extractor and the language model in detail.

2.1.1 Feature Extractor (Encoder)

The feature extractor aims to extract intermediate representation features from the input image, which are then used as the inputs of the language model for description generation. Common feature extractors include two types: (1) Convolutional Neural Network (CNN) based ones and (2) Faster R-CNN based ones. The CNN-based feature extractor outputs a feature map corresponding to all positions of the input image by utilizing a set of convolutional layers, *e.g.*, ResNet-101 [18], VGGNet [39], *etc*. However, it pays equal attention to each region of the input image, resulting in limited special localization with semantic meanings. To address it, the Faster R-CNN based feature extractor [2] utilizes a Faster R-CNN [36] to detect regions that contain salient objects or backgrounds with high probabilities, and then extracts feature vectors from the detected regions. Thus, the feature extractor focuses on regions relevant to the content of the image, which helps to generate human-like captions and answer questions.

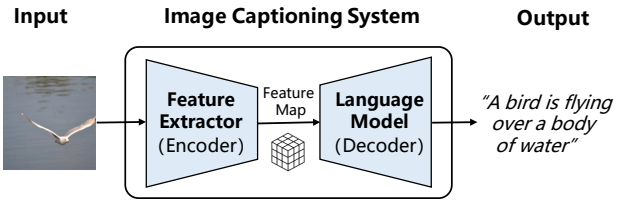


Figure 2: Image captioning system. An input image is encoded to intermediate representation features by a feature extractor (*e.g.*, CNN) and then decoded to an output caption by a language model (*e.g.*, LSTM).

2.1.2 Language Model (Decoder)

The language model generates a descriptive sentence, *i.e.*, a caption, with the extracted image features. State-of-the-art language models are usually based on a Long Short-Term Memory (LSTM) model with an attention module. An attention module generates a weighted feature map from each receptive field in the encoded feature map, and feeds the map along with the previously generated word to the LSTM model, which then generates the next word to form a caption. Since language models with attention modules exhibit better performance compared with the ones without attention modules, they have become the mainstream decoders for image captioning systems nowadays.

To investigate the vulnerability of image captioning systems in terms of adversarial patches, we study three popular image captioning systems that cover encoders and decoders: (1) a Show-Attend-and-Tell [46] model that utilizes a ResNet-101 as the encoder and an LSTM with attention as the decoder, (2) a Self-critical Sequence Training: Att2in model that uses a modified attention model and achieves better performance by employing reinforcement learning [37], compared with Show-Attend-and-Tell, and (3) a Bottom-up Top-down [2] model that uses a Faster RCNN in conjunction with a ResNet-101 as the encoder and a two layers attention LSTM as the decoder.

2.2 Adversarial Examples

Existing adversarial examples against image captioning systems mainly focus on adding noises at the pixel level in the digital world to alter the output caption at either the word or sentence levels. For the word-level alteration, Show-and-Fool [8] is the first work that can insert a few targeted words into the generated caption and the follow-up work by Xu et al. [49] enhances such an attack and can insert keywords in specific locations by using a structured output learning with latent variables. Instead of inserting, Ji et al. [21] generate adversarial examples to remove targeted words while remaining the produced caption accurate. For sentence-level alteration, both Show-and-Fool [8] and Xu et al. [48] can change the output caption to a targeted sentence. Xu et al. [48] consider image captioning as a sequential recognition task, and propose an optimization-based attacking algorithm for sequential recognition models by learning adversarial perturbations from the derived gradients of each word in the sequence.

Instead of creating adversary examples by adding noises to the digital images, this paper aims at generating adversary patches that can be physically printed and worn, such that they can induce incorrect captions. In addition, `CAPatch` works for image captioning systems with CNN-based or Faster R-CNN based encoders while existing methods mainly work for the former.

2.3 Adversarial Patch

The adversarial patch is one type of adversarial example that has been shown to be effective in the physical world [5]. Different from previous pixel-wise digital adversarial examples [16], the adversarial patch appears in a form of perturbations within a small area and can be physically printed to attach to an existing object or to become a stand-alone image, *e.g.*, a poster. Typically, the adversarial patches are trained to be universal [5], *i.e.*, the attack can be successful for any images attached with the patch, which makes the attack easy to launch in the real world. Thus, the adversarial patch has been widely investigated and prior work [4, 9, 40] has studied physical adversarial patch attacks on computer vision models such as image classifiers and object detectors.

However, it is unknown whether existing patch generation methods are effective against a multi-modal image captioning system, which consists of both computer vision models and the natural language models, which may correct the mistakes produced by the computer vision models. In this paper, we investigate the problem and explore the feasibility of attacking the image captioning systems in the physical world with an adversarial patch. Particularly, we exploit the vulnerability of the language model to amplify the misclassification produced by prior work in computer vision.

3 Threat Model

3.1 Attack Goal

The goal of the attacker is to cause the image captioning system used in scenarios such as intelligent surveillance [6, 33] or blind assistance [13, 17, 31, 41] to output an incorrect caption by placing an adversarial patch at the scenes. For example, future intelligent (unattended) video surveillance systems may utilize image captioning systems to automatically generate surveillance logs for the sake of lowering manpower costs. An adversary tries to *bypass a video surveillance system for home security by wearing an adversarial patch outside her jacket to trick the image caption systems to output benign logs* such that she can break-in. To achieve such goals, the attacker may perform two types of attacks: (1) Caption Makeover Attacks (CMA), whereby the image captioning system outputs a completely irrelevant caption, and (2) Keyword Hiding Attacks (KHA), whereby the image captioning system fails to output a decisive keyword describing the image, *e.g.*, a person.

3.2 Adversary Capability

To achieve the aforementioned attack goals, we assume the adversary has the following capabilities:

Target Model Access. We assume the adversary may have white-box or black-box access to the target image captioning system. In white-box attacks, the adversary has prior knowledge of the image captioning model used in the victim surveillance system, including but not limited to their architecture, parameters, *etc*. In black-box attacks, the adversary has no prior information about the victim system but can train a `CAPatch` using a customized white-box model. In both cases, the adversary need not acquire the captured images or the captions generated by the victim system since `CAPatch` is designed to be applicable regardless of the scenes.

Camera Location Awareness. We assume the adversary can acquire the location of the surveillance camera by observation, based on which she can place the patch facing the camera and ensure that it is integrally captured.

3.3 Design Requirement

In addition, to make `CAPatch` practical, it shall meet the following design requirements:

Workable across Various Scenes. The adversary may conduct attacks at various scenes and the surveillance camera may switch views from time to time. Meanwhile, it is impractical to obtain the images captured by the camera in real-time. Thus, `CAPatch` should be workable across various scenes.

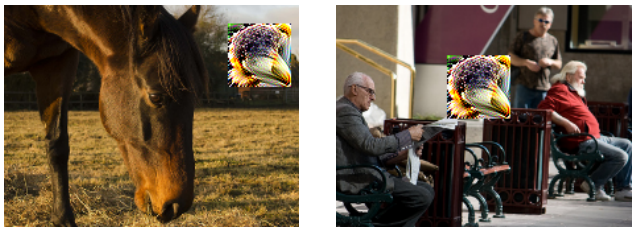
Workable across Various Locations. Similarly, the location of `CAPatch` in the captured image is unknown, lack of the real-time view of the camera. Therefore, `CAPatch` shall be workable across various locations instead of a fixed location.

4 Preliminary Analysis

In this section, we first present that fooling feature extractors is not enough to guarantee a caption makeover or keyword hiding attack against image captioning systems. Then, we introduce the basic idea of `CAPatch`.

4.1 Fooling Feature Extractors is Not Enough

Since adversarial patches have been proved to be effective against CNN and Faster R-CNN models in terms of classification and recognition, a naive trial is to utilize such a patch to attack the image captioning system. To investigate it, we conduct a feasibility test by generating adversarial patches against the feature extractors, *i.e.*, the CNN models. In particular, we generate 2 adversarial patches against a CNN-based feature extractor, *i.e.*, ResNet-101, using the patch optimization method from [5]. The purpose of the patches is to alter the classification results of the images into four targeted classes respectively: (1) *bus* and (2) *bird*. Then, we attach each of the generated patches to 1000 images randomly selected from the ImageNet [10] dataset respectively, and feed those patched images into the Show-Attend-and-Tell image captioning system, with the same ResNet-101 as the encoder and an attention-based LSTM as the decoder, to see whether the patches can modify the captions.



(a) *"A horse is eating grass in a field."* → *"A horse is eating a large bird in the grass."*

(b) *"A group of people sitting on a bench and a man in a red jacket."* → *"A man sitting on a bench with a bird on a bench."*

Figure 3: Captions for images with patches targeted *bird* class are inserted with corresponding keywords.

From the results, we find that adversarial patches against the CNN-based encoder can change the captions but with limited capability. The results show that the adversarial patches can induce the name of the targeted class into the generated caption or hide human-related keywords existing in the original caption with limited attack success rates, *i.e.,* 40.4% and 32.1% for inducing *bus* and *bird* respectively, and 11.0% and 10.7% for hiding keywords such as "person" and "man" with the aforementioned four patches. Fig. 3(a) shows that the generated captions can be illogical in semantics, *e.g.*, *" A horse is eating a large bird in the grass"*. Fig. 3(b) shows that though the targeted class is induced into the generated caption, the keywords "man" and "person" are not hidden. The reason is that the adversarial patches are aiming at classifications instead of a wrong caption. Therefore, simply using adversarial patches against the encoders of an image captioning system is not enough.

4.2 Basic Idea

To enhance the capability of the adversarial patch, our idea is to exploit the vulnerabilities from the system level instead of a single model level. To this end, we carefully analyze the architecture of the image captioning system and find that it consists of three key parts: (1) an encoder for feature extraction, (2) an attention module for image region weighting, and (3) a decoder for caption generation. To attack such a system, `CAPatch` shall meet the following requirements.

❶ **Detection Assurance.** `CAPatch` shall be detected by the feature extractor no matter where it is placed such that its attack effects can be passed to the caption generator in a form of feature vectors. Different from CNN-based encoders that extract features with a uniform grid of equally-sized image regions, Faster R-CNN based encoders first detect regions with salient objects or backgrounds with boxes and then ex-

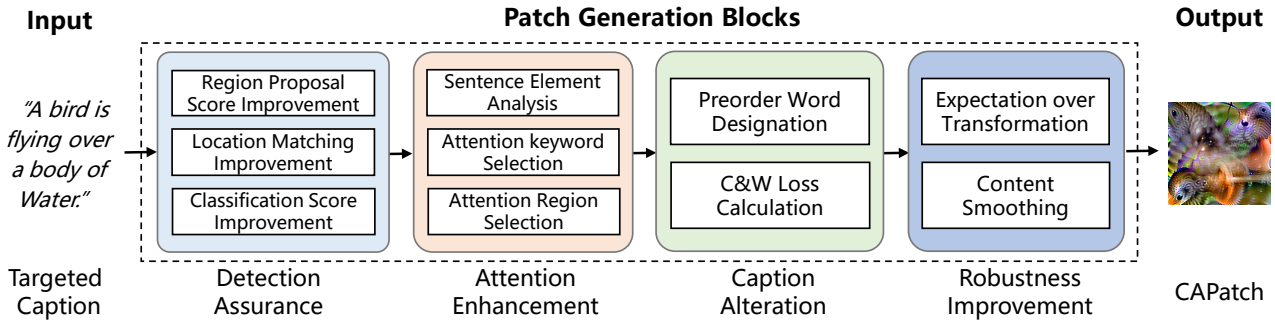


Figure 4: Overview of `CAPatch` generation. Based on the targeted caption, the adversary generates a `CAPatch` by first increasing the possibility that the patch is detected by the image captioning system, next to letting the system pay more attention to the patch when generating key elements of the targeted caption, then altering the generated caption into the targeted one integrally, and finally enhancing the robustness of the patch to make it more practice in the physical world. The generated `CAPatch` can then be attached to any objects to launch targeted attacks.

tract feature vectors from the detected regions. As a result, for image captioning systems with Faster R-CNN based encoders, *e.g.*, Bottom-up Top-down, we shall make sure that the region containing `CAPatch` is detected with as many boxes as possible by the Faster R-CNN model such that the feature vectors extracted from the patch can be passed to the caption generator to take effect.

❷ **Attention Enhancement.** `CAPatch` shall be weighted more by the attention module such that its attack effects can be strengthened. The image captioning system utilizes an attention module to pre-process the feature vectors extracted by the encoder before feeding them into the LSTM model to generate sentences. The goal of the attention module is to put more weights on feature vectors extracted from key image regions, to generate sentences rich in semantics. To strengthen the attack effects of `CAPatch`, we shall render the attention module to put more weights on its feature vectors no matter where it is placed.

❸ **Caption Alteration.** `CAPatch` shall be optimized with a target caption for both attack goals. For CMA, this is necessary to achieve a sentence-level attack. For KHA, this can help suppress other words. To achieve it, we shall optimize `CAPatch` based on the vulnerabilities of both the feature extractors and the caption generator.

❹ **Robustness Improvement.** To further make `CAPatch` practical in the physical world, it shall be resistant to image distortions commonly occurring in both the printing and photographing processes. We shall consider those factors in optimization and enhance the robustness of `CAPatch`.

5 Design

5.1 Overview

Based on the aforementioned idea, we design `CAPatch` with four modules, as shown in Fig. 4. The input of the patch generation blocks is targeted caption and the output is a `CAPatch` that can achieve both CMA and KHA. The ❶ **Detection Assurance** module designs a detection loss considering both the region proposal probabilities and the classification results, to make Faster R-CNN based encoders propose as many boxes as possible on any region that `CAPatch` appears. The ❷ **Attention Enhancement** module first analyzes the elements of the targeted caption, then selects keywords that need extra attention, and finally enhances the weights of `CAPatch` related regions when generating those selected words. The ❸ **Caption Alteration** module exploits the vulnerabilities of both the computer vision model and the LSTM model, and designs a caption loss to make it output the targeted sentence. The ❹ **Robustness Improvement** module improves the robustness of `CAPatch` in the physical world by addressing the patch distortions caused by both non-ideal placements and photographing. In the following subsections, we present the details of each module respectively.

5.2 Detection Assurance

`CAPatch` appears in a form of an image patch. To make it effective against the multi-modal image captioning system, we first ensure its attack effect can be passed from the computer vision side to the natural language processing side.

For image captioning systems with CNN based encoders, such a transfer is feasible since CNN based encoders extract features with a uniform grid of equally-sized image regions, and pass all the extracted features to the decoder, as shown in Fig. 5. For image captioning systems with Faster R-CNN based encoders, however, they utilize a Faster R-CNN model to detect regions with salient objects or backgrounds, and only transfer feature vectors extracted from those detected regions. As a result, for those systems, we shall try to ensure that `CAPatch` is detected no matter where it is placed.

The Faster R-CNN based encoder extracts features in two stages. In the first stage, a Region Proposal Network (RPN) outputs several regions that are most likely to contain objects using pre-defined reference anchors, with each region proposed with a probability. Then, a regressor and a classifier

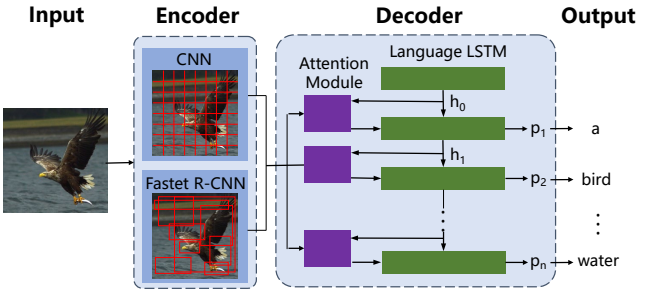


Figure 5: Basic architecture of an image captioning system, where CNN based encoders extract features with a uniform grid of equally-sized regions while Faster R-CNN based encoders extract features from proposed boxes.

output a bounding box and a classification probability for each proposal region, respectively. Finally, feature vectors from bounding boxes with classification probabilities higher than a threshold are transferred to the decoder.

Different from prior work that deceives Faster R-CNN with a clear target in both location and class, our goal is to render Faster R-CNN to propose more valid boxes over the location of `CAPatch` no matter where it is placed. To achieve it, we consider the following three steps: (1) increasing the possibility that the model proposes regions over the location of `CAPatch`, (2) increasing the classification scores of the proposed bounding boxes to exceed the predefined threshold thus making them valid, and (3) increasing the matching degrees of the valid bounding boxes and the location of `CAPatch`.

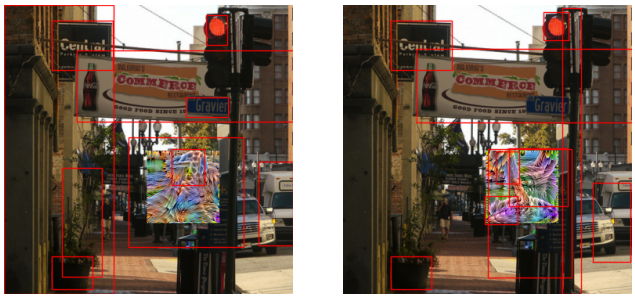
Region Proposal Score Improvement. In the first step, we try to make Faster R-CNN propose as many regions as possible over the location of `CAPatch`. To achieve it, we first search for all the reference anchors whose centers are in the region of `CAPatch`, and denote the searched anchor set as $\boldsymbol{A}$. Then, for each anchor a in $\boldsymbol{A}$, we calculate the RPN probability p_{rpn} of the region determined by a. Finally, we maximize the sum of all the calculated RPN probabilities as the first part of the detection loss:

$$\mathcal{L}_{\text{det}1} = \sum_{a \in \boldsymbol{A}} -\log p^{a}_{rpn} \tag{1}$$

Location Matching Improvement. Each proposed region is then detected as a bounding box with a classification score. With those bounding boxes, we increase their matching degrees with `CAPatch`. In other words, we try to make those bounding boxes have larger overlaps with `CAPatch` such that a large amount of image features from `CAPatch` can be transferred to the decoder. To achieve it, we calculate the Intersection over Area (IoA) of each bounding box and `CAPatch`, and maximize their sum as the second part of the detection loss:

$$\mathcal{L}_{\text{det}2} = \sum_{box \in \boldsymbol{B}} -IoA(box, box_{patch}) \tag{2}$$

where $\boldsymbol{B}$ is the set of bounding boxes, box_{patch} is the location of `CAPatch`, $IoA(\cdot)$ is the intersection area between box and box_{patch} over box_{patch}'s area.



(a) `CAPatch` is contained by only two boxes without the detection loss.

(b) `CAPatch` is contained by four boxes with detection loss.

Figure 6: The number of proposed boxes that contain `CAPatch` is increased from two to four with the detection loss.

Classification Score Improvement. A bounding box is considered to be valid and the features can be passed to the caption generator if its classification score is significant. To make those boxes close to `CAPatch` valid, we need to maximize their classification scores. We first select the box out of the patch according to IoA and then increase the scores of their corresponding categories to avoid their disappearance in further optimization.

$$\begin{aligned} \boldsymbol{B'} &= \{box | IoA(box, box_{patch}) > \delta\}, box \in \boldsymbol{B} \\ \mathcal{L}_{\text{det}3} &= \sum_{box \in \boldsymbol{B'}} -\log p^{b}_{cls} \end{aligned} \tag{3}$$

where $\boldsymbol{B}$ is the set of bounding boxes, $\boldsymbol{B'}$ is the selected bounding boxes, δ is the selection threshold, p^{b}_{cls} is the classification probability of the bounding box b. In practice, we set $\delta = 0.7$.

The finally detection loss is consisted of the aforementioned three parts as follows:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{det}1} + \lambda_1 \mathcal{L}_{\text{det}2} + \lambda_2 \mathcal{L}_{\text{det}3} \tag{4}$$

where λ_1, λ_2, are the weights of $\mathcal{L}_{\text{det}2}$ and $\mathcal{L}_{\text{det}3}$, respectively.

With the detection loss, we ensure that the encoder detects `CAPatch`, as shown in Fig. 6. Then, the encoder extracts a set of feature vectors from the input image. Each feature vector v_i is a D-dimensional vector, and we denote the feature vector set as:

$$\boldsymbol{V} = \{v_1, v_2, \ldots, v_m\}, v_i \in \mathbb{R}^D \tag{5}$$

where m is the number of feature vectors.

5.3 Attention Enhancement

The feature vectors V extracted from the input image are then fed into the decoder for caption generation. The encoder, *i.e.*, the language model, is usually composed of an attention module and a language LSTM. When generating each word of the output caption, the attention module selects feature vectors (*i.e.*, image regions) that need special attention and then the language LSTM translates those selected feature vectors into a

(a) Benign image (b) *"A group of people walking down a street with umbrellas."* (c) *"A man is flying a kite in the water."* (d) *"A bird is flying over a body of water."* (e) *"A bird is flying over a bird."*

Figure 7: Illustrations of the effects of the attention loss. (a) is benign image, (b) shows the caption and attention heatmap of the benign image, and (c-e) show the captions and attention heatmaps with a patch targeted *"A bird is flying over a body of water"* but enhanced with no attention loss, an appropriate attention loss, and an excessive attention loss receptively.

word. At each time step t (*i.e.*, each word of the sentence), the feature vectors under attention can be expressed as a weighted sum of the exacted feature vector set $\boldsymbol{V}$ as follows:

$$\hat{v}_t = \sum_{i=1}^{m} \alpha_{(i,t)} v_i \tag{6}$$

where $\alpha_{(i,t)}$ is the weight of each feature vector v_i. We denote $\boldsymbol{\alpha}_t = \{\alpha_{(1,t)}, \alpha_{(2,t)}, ..., \alpha_{(m,t)}\}$ as the weight vector, which can be further expressed as:

$$\begin{aligned} \boldsymbol{\alpha}_t &= \text{softmax}\left(e_t\right) \\ \boldsymbol{e}_t &= f_{\text{att}}\left(\boldsymbol{V}, h_{t-1}\right) \end{aligned} \tag{7}$$

where $\boldsymbol{e}_t$ is the unnormalized weight vector calculated by an alignment model $f_{att}(\cdot)$ based on the input feature vector set $\boldsymbol{V}$ and the hidden state of the last word h_{t-1}.

To strengthen the attack effects of `CAPatch`, we try to render the attention module to put more weights on the feature vectors extracted from the regions of `CAPatch` when generating captions. A natural question is, shall such an operation be performed on each word of the caption or only one or several keywords? To investigate, we analyze the elements of a caption.

Sentence Element Analysis. We find that a caption can be regarded as composed by two types of elements [15]: (1) content words generated by the information from the image, and (2) function words generated without much image information. Typical content words include (1) nouns such as "cat", "bog", and "bus", (2) verbs such as "fly", "sit", and "stand", (3) adjectives such as "large", "green", and "dirty" (4) adverbs such as "highly" and "beautifully", and (5) numerals such as "one", "two", *etc*. Function words include prepositions, pronouns, conjunctions, auxiliary verbs, and articles, *e.g.*, "in", "out", "I", "and", "of", "are", "the", *etc*. When generating function words, the language LSTM utilizes more semantic information of the previous word instead of the features extracted from the image [29].

Attention Word Selection. Inspired by this observation, we design to render the system attend to the feature vectors extracted from the regions of `CAPatch` when generating content words rather than every word of the caption. The reason for such a design is two-fold: (1) the attack efficiency can be improved by freeing function words, and (2) the attack success rate can be increased since too much attention will cause the system to focus on a specific word and generate captions such as "A bird is flying over a bird", as shown in Fig. 7.

Attention Region Selection. Then, we select the regions (feature vectors) that the attention module shall attend to. As shown in Fig. 5, the CNN based encoder extracts a feature vector from each equal-sized grid region of the image, while the Faster R-CNN based encoder extracts a feature vector from each detected bounding box. In both cases, `CAPatch` has the possibility to be contained by several regions but it's hard for them to overlap exactly. To ensure that the attention module attends to `CAPatch` as much as possible while paying little attention to other regions, we select regions that have significant overlaps with `CAPatch`:

$$\boldsymbol{B}_{att} = \{box | IoA(box, box_{patch}) > \delta\}, box \in \boldsymbol{B}_{reg} \tag{8}$$

where $\boldsymbol{B}_{att}$ is the set of selected regions, $\boldsymbol{B}_{reg}$ is the set of regions with feature vector, δ is the region selection threshold, we set the same δ as Eq. 3.

Attention Loss. With the selected words and regions, we design the attention loss $\mathcal{L}_{att}$ as follows:

$$\mathcal{L}_{att} = \sum_{t \in T} \max\{-\sum_{i=1}^{n} \alpha_{(i,t)}, -\varepsilon_1\} \tag{9}$$

where T is the set of selected keywords, ε_1 is the maximum of the attention weight for each word. In practice, $\varepsilon_1 = 0.35$.

5.4 Caption Alteration

The language LSTM then generates the final caption by outputting each word in sequence. At each time step t, the output of the language LSTM is determined by the hidden state of the last word h_{t-1}, the embedding of the last word w_{t-1}, and the weighted feature vector $\hat{v}_t$ as follows:

$$\begin{aligned} \boldsymbol{z_t} &= LSTM(h_{t-1}, \hat{v}_t, w_{t-1}) \\ \boldsymbol{p_t} &= softmax(\boldsymbol{z_t}) \end{aligned} \tag{10}$$

where $LSTM(\cdot)$ represents the LSTM cell, $\boldsymbol{z_t}$ is a set of unnormalized probabilities that indicate the possibility of each word in the vocabulary list W appearing at the current position, and the final output $\boldsymbol{p_t}$ is the normalized version of $\boldsymbol{z_t}$. Then, the word with the maximal probability is usually selected as the output word w_t.

Joint Probability Estimation. To generate the target caption integrally, we first estimate the joint probability for the system to generate it at present. Denote the target caption as $S = \{w_0, \cdots, w_k\}$, where k is the number of words in the target caption. Then, the joint probability of generating S can be calculated as follows:

$$\log P\left(S|I'\right) = \sum_{t=2}^{k} \log P\left(w_t \mid I', w_1, \ldots, w_{t-1}\right) \tag{11}$$

where I' is the input image attached with `CAPatch`. Our goal is to enhance $\log P(S|I')$ and ensure it attains the maximum value among all possible captions.

C&W Loss Calculation. To achieve it, we employ the commonly-used C&W loss [7] as the caption loss, which utilizes the unnormalized probabilities $\boldsymbol{z_t}$, *i.e.*, logits, and has been proved to be effective on various adversarial attack tasks:

$$\mathcal{L}_{cap} = \sum_{t=1}^{k} \max\{\max_{i \neq index(w_t)} \{z_t^i\} - z_t^{index(w_t)}, -\varepsilon_2\} \tag{12}$$

where ${z_t}^i$ is logit of the i^{th} word in the vocabulary list W, $index(w_t)$ is the index of the word w_t in W, ε_2 is a constant value to avoid over-optimizing a specific word. In practice, we set $\varepsilon_2 = 1$.

5.5 Robustness Improvement

With the aforementioned detection loss, attention loss, and caption loss, the generated `CAPatch` shall be able to fool the system to output a target caption. To further make it robust in the physical world, we then perform the robustness enhancement. The challenges of a robust `CAPatch` mainly lie in two aspects: (1) the detailed texture of `CAPatch` can be lost during both the printing and photographing processes due to the limited resolution or non-ideal exposure, and (2) the captured `CAPatch` can be distorted due to the unparallel photographing.

Color Smoothing. For the first challenge, the optimized `CAPatch` in the digital world shall be printed and then captured by the camera to be effective, where both processes can introduce pixel errors due to the limited resolution and the sampling noises. As a result, extreme differences between adjacent pixels in the perturbation are unlikely to be accurately captured by cameras [38]. To address it, we smoothen the color of `CAPatch` by exploiting a Total Variation (TV) loss:

$$\mathcal{L}_{tv} = \sum_{i,j} \sqrt{\left((p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2\right.} \tag{13}$$

where p denotes the patch, and $p_{i,j}$ is the pixel at (i, j).

The final loss of `CAPatch` is consisted of a detection loss, an attention loss, a caption loss and a TV loss, as follows:

$$\mathcal{L} = \mathcal{L}_{cap} + \alpha\mathcal{L}_{det} + \beta\mathcal{L}_{att} + \gamma\mathcal{L}_{tv} \tag{14}$$

where α, β, γ are the weights of those losses, respectively.

Expectation over Transformation. For the second challenge, since the captured image only stands for a certain snapshot of the real scenario, and will differ from the original image when the perspective of shooting varies even if the `CAPatch` in the view remains the same. Factors commonly considered to affect photo shooting include lighting conditions, shooting distances, and angles. To address it, we use the Expectation over Transformation (EoT) [5], which augments the training of `CAPatch` with random transformations to overcome various situations in the physical world.

To deal with the transformation of `CAPatch`, we augment the patch with four dimensions, *i.e.*, resize, rotation, brightness, and contrast. We design the transformation as a preprocessing of `CAPatch`, and perform four different transformations simultaneously with a uniform distribution to randomize the degree of each transformation, as follows:

$$\widehat{p} = \arg\min_{p} \mathbb{E}_{x \sim X, t \sim T, l \sim L}[\mathcal{L}(A(p, x, l, t); S] \tag{15}$$

where $\widehat{p}$ is the optimized `CAPatch`, X is the training dataset, L is a distribution over locations in the image, T is a distribution over transformations of the patch. The function $A(\cdot)$ refers to adding the transformed patch to the image x at the location l.

The whole training process of `CAPatch` is shown in Algorithm 1 in Appendix. A.

6 Evaluation

In this section, we evaluate the performance of `CAPatch` against image captioning systems. We consider two sets of evaluations in this paper: (1) simulation evaluation, where digital images attached with `CAPatch` are fed into image captioning systems directly, and (2) real-world evaluation, where the images with `CAPatch` are printed first and captured by cameras. In summary, we highlight the key results as follows:

- In the simulation evaluation, `CAPatch` can achieve attack success rates of up to 98.4% for CMA and 98.8% for KHA. The average success rates are 80.1% for CMA and 86.0% for KHA on three popular image captioning models under 6 targeted captions.
- In the real-world evaluation, `CAPatch` can achieve continuous attacks with an overall attack success rate of 73.1% for CMA and 92.4% for KHA under different cloth materials and various camera resolutions.
- `CAPatch` can achieve attacks across various locations and scenes, and is robust to the image rotation, image resizing, light condition change and camera resolutions.

Table 1: Overall Performance of CAPatch

Target Caption	Attack Success Rate					
	SAT		SCST		Up-Down	
	CMA	KHA	CMA	KHA	CMA	KHA
Caption 1: A bird is flying over a body of water	87.8%	93.2%	90.2%	93.3%	72.5%	79.3%
Caption 2: A cat is sitting on a wooden bench	98.4%	98.8%	94.4%	97.0%	71.5%	80.0%
Caption 3: A black dog holding a frisbee in its mouth	95.9%	97.3%	87.8%	90.3%	87.1%	89.3%
Caption 4: A couple of elephants are standing in the glass	84.5%	86.6%	60.1%	97.9%	41.7%	64.9%
Caption 5: A fire hydrant sitting on the side of a street	93.4%	96.9%	82.0%	91.0%	52.4%	54.3%
Caption 6: A bathroom with a sink and a mirror	81.5%	77.4%	94.2%	96.7%	66.7%	63.8%

6.1 Setup

Image Captioning Models. We choose three representative image captioning models, *i.e.*, Show-Attend-and-Tell (SAT) [46], Self-critical Sequence Training: Att2in (SCST) [37], and Bottom-up Top-down (Up-Down) [2] that contain different encoders and decoders mentioned in Sec. 2. Specifically, the encoder of SAT and SCST is ResNet-101 trained on ImageNet [10] while the encoder of Up-Down is Faster R-CNN trained on the Visual Genome dataset [23]. The decoder of all the three models is LSTM [19], and they are all trained on the training set of Microsoft COCO 2014 (MS COCO) [27], which is a commonly used benchmark database for image captioning.

Datasets. We utilize the *Karpathy* [22] validation and test datasets extracted from the MS COCO dataset, which is along with other image captioning studies [8, 49]. The *Karpathy* datasets contain 5,000 validation images and 5,000 test images respectively. Each image is with 5 human-annotated captions and resized to 600 $\times$ 600 pixels as required by the image captioning models. We utilize the validation dataset for CAPatch training and the test dataset for the evaluation.

Target Captions and Words. In accord with the attack scenario, we select 6 target sentences for CMA as shown in Tab. 1, which describe irrelevant animals, indoor and outdoor scenes respectively. For KHA, we select 11 words related to humans as the target hidden words, including person, people, man, woman, men, women, girl, boy, girls, boys and player. For each tested image captioning model, we select images that can output these words in the original test dataset as the KHA test dataset ($\sim$2,000 images).

Patch Training. We utilize the I-FGSM [25] method for CAPatch optimization as it is friendly to handle losses in different scales. To boost the effectiveness of I-FGSM, we adopt the learning rate decay strategy, which is commonly used in the training of neural networks. Empirically, we set the initial learning rate to 0.016 and finally decay it to 0.001.

6.2 Simulation Evaluation

In this section, we evaluate the attack effectiveness, robustness, and transferability of CAPatch against three image captioning models. We also conduct a comparison with related works.

(a) *"A horse is eating grass in a field."* $\rightarrow$ *"A bird is flying over a body of water."*

(b) *"A group of people sitting on a bench and a man in a red jacket."* $\rightarrow$ *"A bird is flying over a body of water."*

Figure 8: Captions for images with CAPatch targeted *"A bird is flying over a body of water."*. The patches are attached to the same images and locations as Fig. 3.

6.2.1 Attack Effectiveness

Overall Performance. To evaluate the overall performance of CAPatch against image captioning systems, we first train 18 CAPatchs (150 $\times$ 150 pixels by default) for the 6 target captions across the 3 image captioning models. Then, we attach the generated patches to each image in the test dataset and obtain the adversarial images. Finally, we feed those images into the three target systems and obtain the overall attack success rates with two attack goals as shown in Tab. 1.

For CMA, CAPatch can achieve an average attack success rate of 88.9% against SAT, 84.8% against SCST, and 65.3% against Up-Down. For KHA, CAPatch can achieve an average attack success rate of 90.5% against SAT, 94.4% against SCST, and 71.9% against Up-Down. Thus, CAPatch can achieve higher attack success rates on KHA than CMA. Among the three image captioning systems, SAT and SCST are more vulnerable to CAPatch attacks compared with Up-Down. The reason is that SAT and SCST utilize the CNN encoder while Up-Down utilizes the Faster R-CNN decoder. The former has more chances to pass the adversarial features embedded in the patches to the subsequent language models, resulting in a higher attack success rate.

Impact of Patch Sizes. A patch of a larger size may increase the attack success rate at the cost of a closer attack distance. To investigate the impact of patch sizes, we generate CAPatchs with various sizes ranging from 100 pixels to 200 pixels with a step of 20 pixels. Then, we feed the generated CAPatchs into the three image captioning systems (target

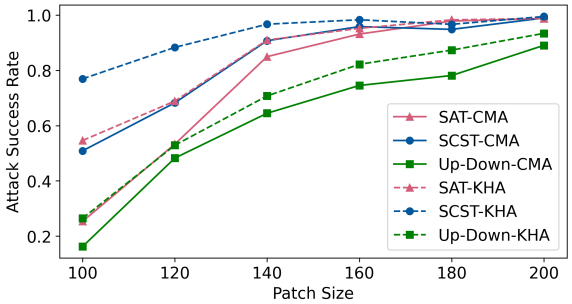


Figure 9: Attack success rates v.s. patch sizes.

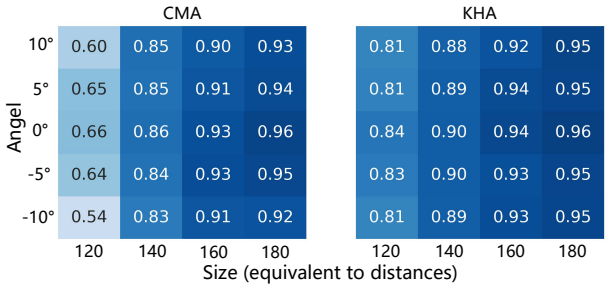


Figure 10: Attack success rates v.s. rotation angles and distances.

Caption 1 by default) and calculate the attack success rates as shown in Fig. 9.

From the results, we find that the attack success rates increase with the patch size for all three models. With a patch size of 100 × 100 pixels, CMA can achieve average attack success rates of 25.4%, 50.9%, and 16.2% against SAT, SCST, and Up-Down, and KHA can achieve average attack success rates of 54.6%, 76.9%, and 26.5%, respectively. When the patch size approaches 200 × 200 pixels, the attack success rates increase to 98.8%, 99.0%, and 89.1% for CMA, and 99.0%, 99.5%, and 93.4% for KHA. The reason for the better performance of a large patch is that more adversarial features can be extracted to generate the targeted captions.

Impact of Locations. `CAPatch` is designed to work with various locations. To validate it, we evaluate whether it is effective regardless of its location in the image. The results show that `CAPatch` can work in various locations with slight performance differences. More details can be seen in Appendix. B.

Impact of Scenes. Another factor that may affect the attack effectiveness is the background scene where `CAPatch` is attached to. In particular, we investigate whether the size and number of objects in the scene will impact the attack performance of CMA and whether the size of people will impact the attack performance of KHA. The results show that `CAPatch` can work in various scenes with some performance differences. More details can be seen in Appendix. C.

6.2.2 Attack Robustness

In Sec. 5.5, we propose a robustness enhancement method to make `CAPatchs` practical in the physical world. In this section, we evaluate the attack robustness of `CAPatch` in terms of its resistance to image rotation and resize. To investigate the impact of image rotation, we rotate the generated `CAPatchs` with an angle ranging from $-10°$ to $10°$ with a step of $5°$, before attaching them to random positions of 1000 randomly-selected test images. To investigate the impact of distances, we resized the original patch (150×150 pixels) to $120 \times 120, 140 \times 140, 160 \times 160, 180 \times 180$ pixels respectively before attaching them to the test images. Different from the experiments in Sec. 6.2.1, the patch size changes here is to emulate different distances between `CAPatch` and the camera. Therefore, in this set of experiments, we investigate the attack performance of `CAPatch` under certain shooting distance variations and angle errors that commonly happen in the real world.

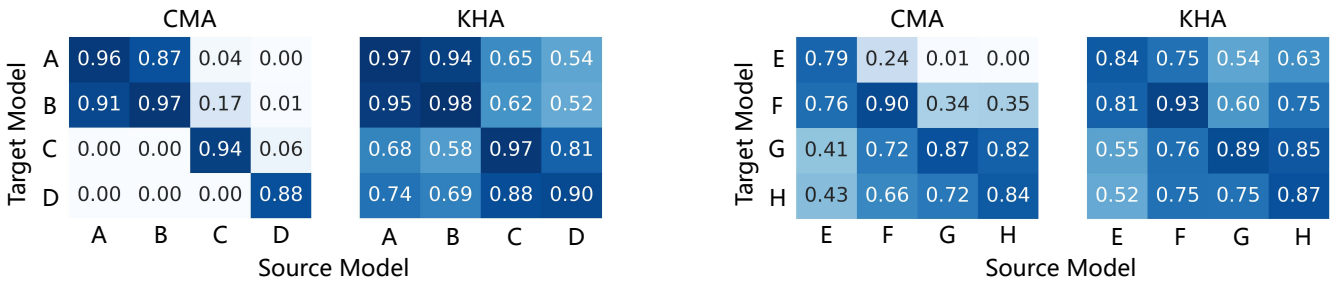
Table 2: Comparison with Related Works.

Method	Attack Success Rate					
	SAT		SCST		UP-Down	
	CMA	KHA	CMA	KHA	CMA	KHA
Show and Fool [8]	0.0%	23.1%	0.0%	5.4%	0.0%	11.7%
SSVM [49]	9.7%	44.5%	29.9%	48.1%	22.5%	28.2%
Random Noise	0.0%	2.8%	0.0%	3.1%	0.0%	4.3%
`CAPatch`	87.8%	93.2%	90.2%	93.3%	72.5%	79.3%

The results shown in Fig. 10 demonstrate that the attack success rate increases with the patch size but decreases with the rotation angle for both CMA and KHA. A trained patch with a default 150-pixel width can still achieve attack success rates of over 54% for CMA and over 81% for KHA after resized into 120-pixel, *i.e.*, suffered from around 36% pixels loss. The attack success rate can be improved when the patch is magnified but becomes saturated when the patch size reaches 160-pixel width. For image rotation, we find that a `CAPatch` larger than 140×140 pixels can achieve attack success rates of over 80% for both CMA and KHA when rotated by $10°$. Another finding is that large patches are more resistant to image rotation. For instance, a patch of 120-pixel width suffers from a success rate decrease of up to 12% for CMA while a 180-pixel width patch only suffers from a decrease of up to 4% when both rotated by $10°$. We assume it is because large patches contain more adversarial information and thus can be more robust to the distortions.

6.2.3 Comparison with Related Works

We also compare `CAPatch` with 2 most-related methods, which are (1) Show-and-fool [8], and (2) SSVM [48]. Since these two methods are digital adversarial example attacks designed for a single image, we use their optimization methods and train them with the same training data as `CAPatch`. Besides, we employ (3) random noise as the baseline of the comparison. From the results shown in Tab. 2, we can find that `CAPatch` shows higher attack success rates for both CMA and KHA. Compared with Show-and-fool and SSVM, `CAPatch`



(a) Results between models with CNN encoder

(b) Results between models with Faster R-CNN encoder

Figure 11: Transferability of `CAPatch` across various models.

achieves better performance since it employs the proposed detection assurance and attention enhancement methods. Compared with the random noises, `CAPatch` is more effective since it is specifically optimized for target goals by employing the caption loss.

6.2.4 Attack Transferability

In addition to the white-box attacks analyzed above, we investigate the black-box attack capability of `CAPatch`. Specifically, we conduct transfer-based black-box attacks by using a `CAPatch` trained based on a white-box model to attack another black-box model. To achieve it, we test the transferability of `CAPatch` across 8 models with different encoders, decoders and training methods as shown in Tab. 6 in Appendix. D, where model A, model D and model E correspond to SAT, SCST and Up-Down. We select caption 3 as our target caption and set the patch size to 150×150 pixels. We report the transferability between models with the same encoder since features are difficult to transfer between different encoders.

From the results shown in Fig. 11, we find that for KHA, `CAPatch` can achieve transfer-based black-box attacks with an average success rate of 70.1% and shows better transferability between models with the CNN encoder. For CMA, `CAPatch` can achieve transfer-based black-box attacks with an average success rate of 31.3% and shows better transferability between models with the Faster R-CNN encoder. Thus, `CAPatch` can achieve attacks even without any prior information about the victim image captioning system. In addition, the adversary can increase the attack success rate by using `CAPatch` trained on various models at the same time.

6.3 Real-world Evaluation

In this section, we evaluate the attack effectiveness and robustness of `CAPatch` in the real world.

6.3.1 Experimental Setup

Physical Patch. We generate an adversarial patch with a target caption of "A bird is flying over a body of water", then print it on a paper with a size of $30cm \times 30cm$. The physical patch

(a) Man A (b) Man B (c) Man C (d) Man D

Figure 12: Four volunteers wearing different clothes attached with patches.

is then attached to the body of four volunteers with different clothes including (1) a black and white sweater, (2) a blue striped shirt, (3) a green hoody, and (4) a dark blue jacket. The four volunteers and their clothes are shown in Fig. 12. During the experiments, we ask the volunteers to raise and lay down their arms to investigate the impact of slight motions. In the default distance, `CAPatch` occupied 5% of the image. Our evaluation has got the approval of the Institutional Review Board (IRB).

Surveillance Camera. We use an Apple iPhone 12 smartphone as the surveillance camera. During the attacks, We record videos at 30 FPS with different resolutions ranging from 360P, 480P, 720P, 1080P, and 4K for each volunteer. Each video lasts for 5 seconds, *i.e.*, 150 frames. The default ambient light condition is 300 lux.

Image Captioning Model. We employ SCST as the victim model and feed the recorded videos into it to evaluate the performance of `CAPatch`. The real-world demos can be found at https://github.com/USSLab/CAPatch.

Table 3: Real-world performance of `CAPatch`.

Volunteer	Methods	Attack Success Rate				
		360P	480P	720P	1080P	4K
Man A in the black and white sweater	CMA	16.0%	72.7%	90.0%	84.7%	94.7%
	KHA	100.0%	100.0%	100.0%	100.0%	100.0%
Man B in the blue striped shirt	CMA	35.3%	92.7%	88.7%	93.3%	92.7%
	KHA	100.0%	100.0%	100.0%	100.0%	100.0%
Man C in the green hoody with letters	CMA	13.3%	62.7%	92.7%	100%	94.0%
	KHA	98.0%	98.7%	100.0%	100.0%	100.0%
Man D in the dark blue jacket	CMA	2.0%	47.3%	89.3%	100%	100.0%
	KHA	9.3%	47.3%	95.3%	100.0%	100.0%

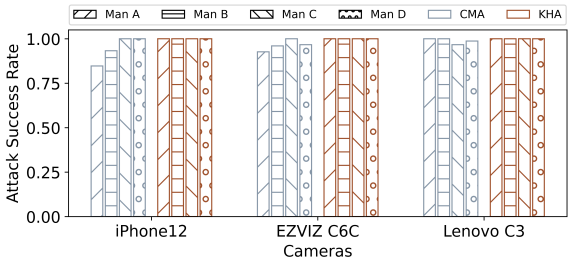


Figure 13: Attack success rates with different cameras.

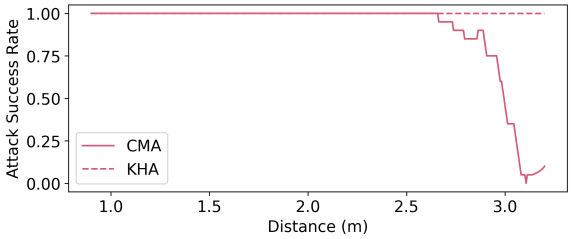


Figure 14: Attack success rates with different distances.

6.3.2 Attack Effectiveness

We first evaluate the overall performance of CAPatch by investigating its attack success rates for both CMA and KHA under different cloth materials and various camera resolutions.

The results shown in Tab. 3 demonstrate that with a default camera resolution of 1080P, a physically printed patch can achieve continuous attacks with an average success rate of 94.5% for CMA and 100% for KHA. In addition, CAPatch does not show significant performance variations when attached to different cloth materials, which provides encouraging signs of practical uses.

For video resolutions, we find that the average attack success rate has a correlation with the camera resolution. With the 4K resolution, the average attack success rate is 95.4% for CMA, slightly higher than that of 1080P and is maintained at 100% for KHA. Low camera resolutions, *e.g.*, 360P, decrease the performance of CAPatch since many details of the patch are lost during the photographing. Nevertheless, since surveillance cameras usually employ high resolutions ($\geq$ 720P), CAPatch can achieve an average attack success rate of over 90.2% for CMA and 98.8% for KHA in practice.

6.3.3 Attack Robustness

We then evaluate the attack robustness of CAPatch by investigating the impacts of light conditions, cameras, target models, and attack distances.

Impact of Light Conditions. For various scenes, the ambient light condition may have variations. To investigate its impacts, we conduct experiments in five scenes with different light conditions including two hallway scenes, two indoor scenes, and one outdoor scene. In each scene, we record a 5-second 1080p video. The results are shown in Tab. 4. For KHA, we find that CAPatch achieves attack success rates of over 95% in all light conditions. For CMA, CAPatch achieves an attack success rate $\geq$ 93% with an ambient light condition $\geq$ 130 lux. However, in the scenes with bad light conditions, *e.g.*, Hallway 1 (2 lux), the performance of CAPatch decreases since more noises are introduced during the photographing.

Table 4: Attack success rates under various light conditions.

Scene	Light Condition	Attack Success Rate	
		CMA	KHA
Hallway 1	2 lux	78.7%	100.0%
Hallway 2	130 lux	94.0%	95.3%
Indoor 1	300 lux	93.3%	100.0%
Indoor 2	400 lux	95.3%	100.0%
Outdoor 1	1100 lux	98.7%	98.7%

Impact of Cameras. To investigate the impact of cameras, in addition to the default recording device iPhone12, we conduct experiments with 2 representative surveillance cameras, which are (1) EZVIZ C6C and (2) Lenovo C3. From the results shown in Fig. 13, we can find that CAPatch can work with different cameras. Specifically, for CMA, CAPatch can achieve an average attack success rate of 94.5% with iPhone 12, 96.3% with EZVIZ C6C, and 98.9% with Lenovo C3. For KHA, CAPatch can achieve an average attack success rate of 100.0% with iPhone 12, 100.0% with EZVIZ C6C, and 100.0% with Lenovo C3.

Impact of Models. To investigate the impact of models, we conducted experiments with different models as shown in Tab. 5 demonstrate that CAPatch can successfully attack all three tested models. Specifically, for CMA, CAPatch can achieve an average attack success rate of 94.0% against SAT, 94.5% against SCST, and 91.8% against Up-Down. For KHA, CAPatch can achieve an average attack success rate of 100.0% against SAT, 100.0% against SCST, and 100.0% against Up-Down.

Impact of Distances. Another factor is the attack distance, *i.e.*, the distance of the volunteer to the camera. To investigate it, we ask the volunteer to move from 3.2 m to 0.9 m toward the camera. During moving, the patch size varies from 2.5% of the image to 25.0% of the image We record a 12-second 1080p video and use a slide window of 20 frames to calculate the average attack success rate of each distance. From the results shown in Fig. 14, we find that CAPatch can achieve successful attacks continuously with gradually-changing patch sizes. For CMA, CAPatch can achieve attack success rates over 80% when the volunteer is 0.9 m (CAPatch is 25.0% of the image) to 2.9 m (CAPatch is 2.9% of the image) from the camera.

Table 5: Attack success rates with different models.

Volunteer	Attack Success Rate					
	SAT		SCST		UP-Down	
	CMA	KHA	CMA	KHA	CMA	KHA
Man A	100.0%	100.0%	84.7%	100.0%	92.0%	100.0%
Man B	92.0%	100.0%	93.3%	100.0%	94.0%	100.0%
Man C	84.7%	100.0%	100.0%	100.0%	89.3%	100.0%
Man D	99.3%	100.0%	100.0%	100.0%	92.0%	100.0%

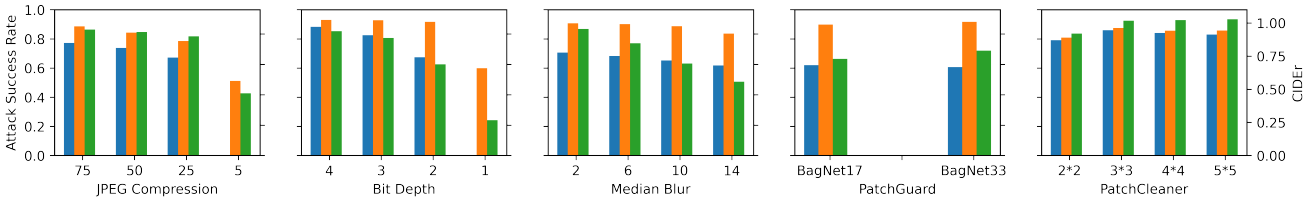


Figure 15: CMA SRs (blue bars), KHA SRs (orange bars) and CIDEr in benign scenarios (green bars) with different defenses. The x-axis represents different settings of defenses.

For KHA, CAPatch can achieve attack success rates over 80% when the volunteer is 0.9 m (CAPatch is 25.0% of the image) to 3.2 m (CAPatch is 2.5% of the image) from the camera. With a longer focal length, e.g., 78 mm, the attack distance can be increased to 8.7 m for CMA and 9.6 m for KHA.

7 DISCUSSION

7.1 Countermeasures

CAPatch exploit vulnerabilities from both computer vision and natural language processing models to mislead the output caption of image captioning systems. To the best of our knowledge, there are no defense mechanisms against adversarial examples for image captioning models yet. However, much work has proposed defense mechanisms against adversarial examples for image classification models and object detection models, which are encoders of image captioning models. We envision some of the aforementioned defense methods have the potential to be transferred to image captioning models, including: (1) modifying the inputs images such as JPEG comprehension [12], median blurring [47] and auto-encoder reformation [30]. Those mechanisms can disturb the perturbations of adversarial examples and thus may decrease the effect of CAPatch. (2) improving models by adversarial training [42] or gradients obfuscation [3], which may increase the difficulty of CAPatch attacks. (3) reducing the propagation of adversarial effects by modifying the model architecture, such as PatchGuard [44] and PatchCleaner [45].

To understand the effectiveness of existing defenses on our attack, we test CAPatch against 5 popular defense methods, including (1) JPEG compression, (2) Bit-depth, (3) Median Blur, (4) PatchGuard, and (5) PatchCleaner. The first three methods transform the input images and can be directly used without retraining. The last two methods change the architecture of the model according to the characteristics of adversarial patches. We use the ASR to evaluate the effectiveness of the defense and use the Consensus-based Image Description Evaluation (CIDEr) [43] to evaluate its impact on the performance of the image captioning model. The result is shown in Fig. 15. For the first three input-transformed defenses, we find that with a large defense strength, the tested methods can defend our attacks to some extent but impair the performance of the image captioning model. For PatchGuard and PatchCleaner, both the benign performance and attack success rate decrease slightly. The results indicate that existing countermeasures employed on computer vision systems cannot easily defeat our attack. We analyze why these defense methods cannot work in Appendix. E and will explore new defense methods considering the characteristics of both the encoder and decoder of image captioning systems in the future.

8 Countermeasures

8.1 Limitation

CAPatch attacks have the following limitations at present. First, we require the white-box access of image captioning models to generate CAPatch at present, which may not always be available in practice. Therefore, one future direction is to improve the transferability of CAPatch. Second, although we have shown CAPatch's attack capability against three image captioning models, including two types of encoders, two types of decoders, and two types of training strategies, there remain other models to be investigated, *e.g.*, Yang et al. [50] utilize a graph-based encoder and Li et al. [26] utilize a transformer-based decoder. The attack effectiveness of CAPatch against those models shall be further studied and improved. Third, several parameters of the CAPatch optimization process are manually set yet and may not be optimal. We plan to employ parameter optimization methods such as Bayesian Optimization [35] to find better parameter combinations in the future.

9 Conclusion

In this paper, we investigate the possibility of deceiving multi-modal image captioning system with adversarial patches. We find that the vulnerability of the computer vision model to adversarial patches can be passed to the natural language processing side of the image captioning system, inducing mistakes in the generated captions. Based on this finding, we propose CAPatch, a physical adversarial patch aiming at targeted attacks against image captioning systems by exploiting vulnerabilities from both the computer vision and natural language processing models. Evaluations with three commonly-used image captioning systems (Show-and-Tell, Self-critical Sequence Training: Att2in, and Bottom-up Top-down) demonstrate the effectiveness of CAPatch in both the digital and physical worlds. This work serves as the first attempt on the adversarial patch against multi-modal artificial

intelligent systems. Further directions include investigating other multi-modal systems vulnerable to adversarial attacks.

Acknowledgement

We thank our shepherd and the other anonymous authors for their valuable comments and suggestions. We would like to thank our colleagues Zihao Dan, Haoxiang Zhang and Bo Yang for help with the experiments. This paper is supported by China NSFC Grant 62222114, 61925109, 62071428, 62271280 and China Postdoctoral Science Foundation Grant BX2021158.

References

[1] Brandon Amos, Bartosz Ludwiczuk, Mahadev Satyanarayanan, et al. Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science*, 6(2), 2016.

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, pages 6077–6086, 2018.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35nd International Conference on International Conference on Machine Learning (ICML'18)*, pages 274–283. PMLR, 2018.

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35nd International Conference on International Conference on Machine Learning (ICML'18)*, pages 284–293. PMLR, 2018.

[5] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[6] Brian Carle. Understanding ai in video surveillance, 2020. https://www.salientsys.com/news/understanding-ai-in-video-surveillance.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P'17)*, pages 39–57. IEEE, 2017.

[8] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*, pages 2587–2597, Melbourne, Australia, 2018.

[9] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Proceedings of the 2018 Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML'18)*, pages 52–68. Springer, 2018.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 248–255, 2009.

[11] Jianfeng Dong, Xirong Li, and Cees GM Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Transactions on Multimedia*, 20(12):3377–3388, 2018.

[12] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[13] Envision. An app that articulates everyday visual information into speech, 2022. https://www.letsenvision.com/app.

[14] Facebook. How does automatic alt text work on facebook?, 2021. https://www.facebook.com/help/216219865403298.

[15] Victoria Fromkin, Robert Rodman, and Nina Hyams. *An introduction to language*. Cengage Learning, 2018.

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[17] Danna Gurari, Yinan Zhao, Meng Zhang, and Nilavra Bhattacharya. Captioning images taken by people who are blind. In *Proceedings of the 2020 European conference on computer vision (ECCV'20)*, pages 417–434. Springer, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, pages 770–778, 2016.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Shahar Hoory, Tzvika Shapira, Asaf Shabtai, and Yuval Elovici. Dynamic adversarial patch for evading object detection models. *arXiv preprint arXiv:2010.13070*, 2020.

[21] Jiayi Ji, Xiaoshuai Sun, Yiyi Zhou, Rongrong Ji, Fuhai Chen, Jianzhuang Liu, and Qi Tian. Attacking image captioning towards accuracy-preserving target words removal. In *Proceedings of the 28th ACM International Conference on Multimedia (MM'20)*, pages 4226–4234, 2020.

[22] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 3128–3137, 2015.

[23] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[24] Ashwath Krishnan, Sudhanva Rajesh, and SS Shylaja. Text-based image retrieval using captioning. In *Proceedings of the 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT'21)*, pages 1–5. IEEE, 2021.

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[26] Guang Li, Linchao Zhu, Ping Liu, and Yi Yang. Entangled transformer for image captioning. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, pages 8928–8937, 2019.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the 2014 European conference on computer vision (ECCV'14)*, pages 740–755. Springer, 2014.

[28] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.

[29] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, pages 375–383, 2017.

[30] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security (CCS'17)*, pages 135–147, 2017.

[31] Microsoft. Add alternative text to a shape, picture, chart, smartart graphic, or other object., 2021. https://support.microsoft.com/en-US/office/add-alternative-text-to-a-shape-picture-chart-smartart-graphic-or-other-object-44989b2a-903c-4d9a-b742-6a75b451c669.

[32] Yisroel Mirsky. Ipatch: A remote adversarial patch. *arXiv preprint arXiv:2105.00113*, 2021.

[33] M Nivedita, Priyanka Chandrashekar, Shibani Mahapatra, Y Asnath Victy Phamila, and Sathish Kumar Selvaperumal. Image captioning for video surveillance system using neural networks. *International Journal of Image and Graphics*, page 2150044, 2021.

[34] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[35] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. Boa: The bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference (GECCO'99)*, pages 525–532. Citeseer, 1999.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, pages 91–99, 2015.

[37] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*, pages 7008–7024, 2017.

[38] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, pages 1528–1540, 2016.

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

[40] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *Proceedings of*

the 12th USENIX Workshop on Offensive Technologies (WOOT'18), 2018.

[41] TapTapSee. Assistive technology for the blind and visually impaired, 2021. https://taptapseeapp.com/.

[42] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[43] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 4566–4575, 2015.

[44] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security'21)*, pages 2237–2254, 2021.

[45] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. {PatchCleanser}: Certifiably robust defense against adversarial patches for any image classifier. In *Proceedings of the 31th USENIX Security Symposium (USENIX Security'22)*, pages 2065–2082, 2022.

[46] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15)*, pages 2048–2057. PMLR, 2015.

[47] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[48] Xing Xu, Jiefu Chen, Jinhui Xiao, Zheng Wang, Yang Yang, and Heng Tao Shen. Learning optimization-based adversarial perturbations for attacking sequential recognition models. In *Proceedings of the 28th ACM International Conference on Multimedia (MM'20)*, pages 2802–2822, 2020.

[49] Yan Xu, Baoyuan Wu, Fumin Shen, Yanbo Fan, Yong Zhang, Heng Tao Shen, and Wei Liu. Exact adversarial attack to image captioning via structured output learning with latent variables. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, pages 4130–4139, 2019.

[50] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, pages 10685–10694, 2019.

A Training Algorithm of `CAPatch`

Algorithm 1 `CAPatch` Generation

Input:
- X: Training set of images
- $S = \{w_0, w_1, \cdots, w_k\}$: Target output sentence
- $model$: Victim image captioning model
- α, β, γ: Hyper-parameters

Output: p: `CAPatch`

for *epoch* in *epochs* **do**
  for img x in X **do**
    $\mathcal{L}_{det} = 0$, $\mathcal{L}_{att} = 0$, $\mathcal{L}_{cap} = 0$, $\mathcal{L}_{tv} = TV(p)$
    $p' \leftarrow$ randomly-transform(p)
    $x' \leftarrow$ randomly-$A(p')$
    feature_set V, box_set $B = model.encoder(x')$
    if $model.encoder =$ Faster R-CNN **then**
      $\mathcal{L}_{det} =$ compute-detection-loss($\boldsymbol{V}, \boldsymbol{B}$)
    end if
    for time step t in N **do**
      weight $\boldsymbol{\alpha}_t = softmax(f_{\text{att}}(\boldsymbol{V}, h_{t-1}))$
      $\mathcal{L}_{att} +=$ compute-attention-loss($\boldsymbol{\alpha}_t$)
      $\boldsymbol{z_t}$ = LSTM($\boldsymbol{\alpha}_t, \boldsymbol{V}, S$)
      $\mathcal{L}_{cap} +=$ compute-caption-loss($\boldsymbol{z_t}, S$)
    end for
    $\mathcal{L} = \mathcal{L}_{cap} + \alpha\mathcal{L}_{det} + \beta\mathcal{L}_{att} + \gamma\mathcal{L}_{tv}$
    p = Optimizer(p, $\nabla_p \mathcal{L}$)
  end for
end for
return p;

B Impact of Locations.

To validate it, we divide an image into 6×6 blocks and test `CAPatch` when its center falls in each block. For each block, we randomly select 1,000 images from the test dataset and attach the generated patches of various sizes (ranging from 120 pixels to 180 pixels) to the corresponding location to conduct attacks. The attack success rates of `CAPatchs` attached to different locations are shown in Fig. 16 in the form of a heatmap.

From the results, we find that for `CAPatchs` of various sizes, the attack success rates of KHA are higher than those of CMA but show similar distributions across different locations. For SAT and SCST, `CAPatch` placed on various locations achieve similar attack success rates. It is because they utilize a CNN

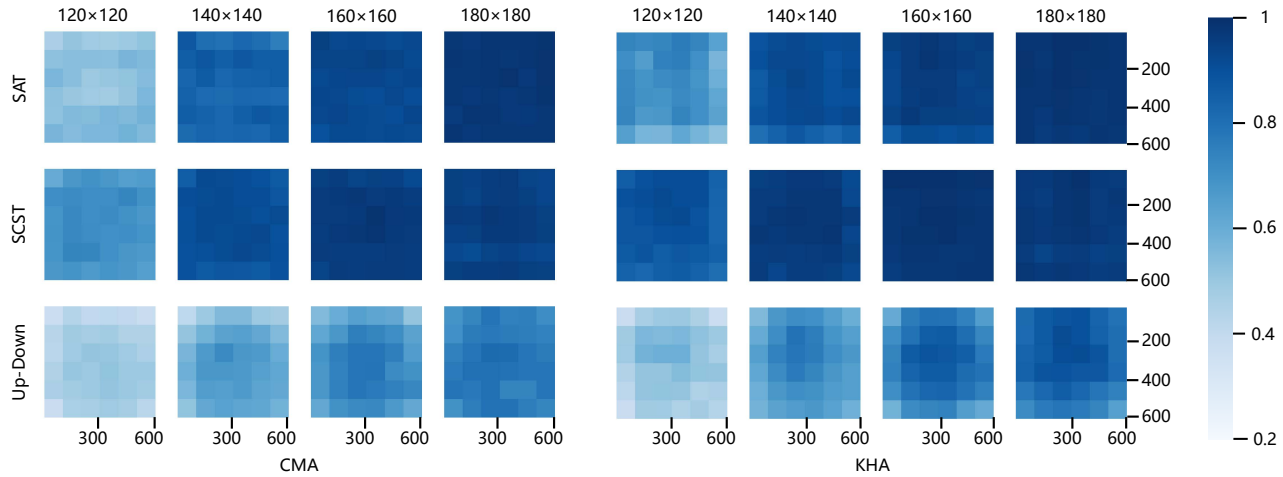


Figure 16: Attack success rates v.s. patch locations. The horizontal rows indicate different image captioning models, the vertical rows indicate different patch sizes, and the colors indicate the attack success rates.

based encoder, which divides images into uniform girds to extract features. For Up-Down, however, CAPatchs placed on the centers of the images achieve higher attack success rates compared with that placed on the edges. The reason lies in two aspects: (1) Up-Down utilizes the Faster R-CNN based encoder, which proposes regions for feature extraction based on anchors. Compared with the image center, the edges contain fewer anchors, rendering the patch more difficult to be detected and thus be less effective. (2) The main objects appearing in the original images are more likely to locate at the image centers. CAPatchs placed on the image centers can block them and thus reduce their impacts.

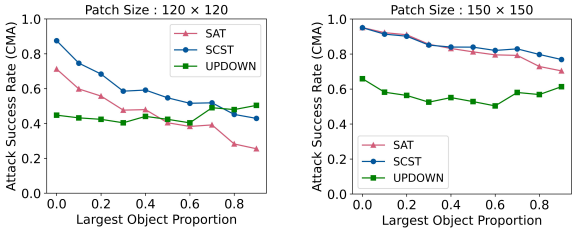
C Impact of Scenes.

To achieve it, we first generate CAPatchs of two sizes, *i.e.*, 120 $\times$ 120 pixels and 150 $\times$ 150 pixels, then divide an image into 10 $\times$ 10 blocks, and finally attach the generated CAPatchs to each block of each image in the test dataset to conduct attacks. We obtain the ground truths of the size of the largest object, the total number of objects, and the size of the largest person in the scenes from the instance annotation of the MS COCO dataset, based on which we plot the attack success rates as shown in Fig. 17.

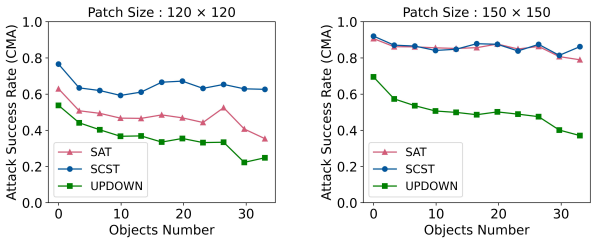
Fig. 17(a) shows the attack success rates of CMA against the proportion of the largest object in the original image. For SAT and SCST, the attack success rate decreases as the largest object proportion increases, and a larger patch is more likely to resist impacts from large objects in the original images. When against SCST, a CAPatch of 120 $\times$ 120 pixels can achieve attack success rates of 87.5% and 42.9% when the largest object proportion is within $[0, 0.1]$ and $[0.9, 1]$ while a CAPatch of 150 $\times$ 150 pixels can achieve attack success rates of 95.0% and 76.9% when the largest object proportion is within $[0, 0.1]$ and $[0.9, 1]$. Similar performances are observed when against SAT. For Up-Down, however, the attack success rate changes slightly with the increase of the largest object proportion. The performance variations between the three models come from their encoders. SAT and SCST utilize CNN based encoders that extract features from uniform grids. As a result, objects with large sizes are extracted with more features and thus have higher impacts on the captioning results. By contrast, Up-Down utilizes the Faster R-CNN based encoder that extracts an equal-sized feature vector from each of the proposed boxes and thus is less sensitive to the objects in the background scenes. Nevertheless, for any system, increasing the size of CAPatch is likely to enhance its resistance to the impacts from large objects in the original images.

Fig. 17(b) shows the attack success rates of CMA against the number of objects in the original images. From the results, we find that Up-Down suffers more impacts from the number of objects compared with SAT and SCST. For Up-Down, a CAPatch of 120 $\times$ 120 pixels can achieve attack success rates of 53.8% and 24.8% when the number of objects is within $[0, 3]$ and $[30, +\infty)$ respectively, while a CAPatch of 150 $\times$ 150 pixels can achieve attack success rates of 69.5% and 37.1% when the number of objects is within $[0, 3]$ and $[30, +\infty)$ respectively. The performance variations are also caused by the encoder. The Faster R-CNN encoder of Up-Down extracts more feature vectors when more objects appear in the image, which makes it difficult for CAPatch to have a high attention weight and thus impairs the attack effectiveness. Similarly, increasing the size of CAPatch can reduce the impact of object numbers for all three systems.

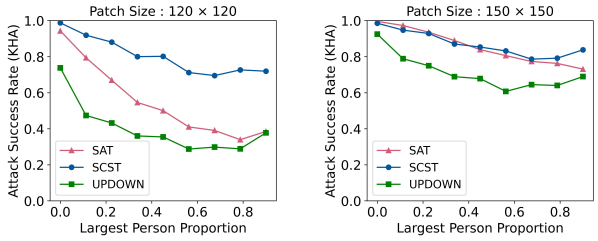
Fig. 17(c) shows the attack success rates of KHA against the proportion of the largest person in the original image, which decrease as the largest person proportion increases for all three models. For SAT and SCST, the results are similar to those of CMA against the proportion of the largest object in Fig. 17(a). For Up-Down, however, the results are different.



(a) CMA success rates v.s. the size of the largest object in the background.



(b) CMA success rates v.s. the number of objects in the background.



(c) KHA success rates v.s. the size of the largest person in the background.

Figure 17: Attack success rates v.s. background scenes.

The possible reason is that the Faster R-CNN encoder of Up-Down is trained on the Visual Genome dataset, which includes 1600 object classes and has many classes related to person such as hand, hair, glasses, dress, etc. A person with a larger proportion may be detected with more characteristics and generate more bounding boxes, increasing the difficulty of hiding.

D Models of Transferability Experiment

Table 6: Eight models used in transferability evaluation.

ID	Encoder	Decoder	Training Method
A	ResNet-101	SAT-decoder	Cross-Entropy
B	ResNet-101	SAT-decoder	Self-critical Sequence
C	ResNet-101	att2in	Cross-Entropy
D	ResNet-101	att2in	Self-critical Sequence
E	Faster R-CNN	Top Down	Cross-Entropy
F	Faster R-CNN	Top Down	Self-critical Sequence
G	Faster R-CNN	att2in	Cross-Entropy
H	Faster R-CNN	att2in	Self-critical Sequence

E Countermeasures

For the first three methods, i.e., JPEG Compression, Bit-depth and Media Blur, they cannot work since we design and employ the robustness improvement module during the generation of `CAPatch`. Specifically, we smooth the colors of `CAPatch` by using tv loss and enhance the effectiveness of `CAPatch` in different size, brightness and contrast by using Expectation over Transformation (EoT). Those robustness improvement methods render the first three defense methods ineffective.

For PatchGuard, it cannot work due to the different architecture between image classification models and image captioning models. PatchGuard is designed for image classification models, which uses a CNN with small receptive fields to decrease the effectiveness of adversarial patches and proposes a robust masking algorithm for secure feature aggregation. However, image captioning models use an LSTM to decode the extracted features instead of a simple fully connected layer and focus on a smaller area when outputting single word. These reasons lead to the reduced effectiveness of PatchGuard when defending `CAPatch` attacks.

For PatchCleaner, it cannot work because it removes the effects of adversarial patches based on the outputs' consistency. PatchCleaner is designed for the image classification system, which applies masks of various locations to the input image and evaluates the model prediction on every masked image. Different from image classification, the outputs of the image captioning system change slightly under different masks, which leads to inconsistent output and then changes the judgment of PatchCleaner. For a clean image, all one-mask predictions usually reach a unanimous agreement while for an adversarial image, a disagreement will occur between the benign prediction and malicious predictions since at least one mask can remove the patch and recover the benign prediction. Then, PatchCleanser performs a second round of masking and regards the consistent output as the correct label. However, for the image captioning model, its output changes with the position of the second-round mask, making it hard to get a uniform result. In this case, PatchCleanser will output the majority prediction of the first-round masking, leading to a failure of defense.