Cocktail Universal Adversarial Attack on Deep Neural Networks

Shaoxin Li[1,2], Xiaofeng Liao[1], Xin Che[2], Xintong Li[2], Yong Zhang[3], and Lingyang Chu[2]⋆

[1] Chongqing University, Chongqing 400044, China
{shaoxin.li,xfliao}@cqu.edu.cn
[2] McMaster University, Hamilton L8S 4L8, Canada
{chex5,lix268,chul9}@mcmaster.ca
[3] Huawei Technologies Canada Co., Ltd, Canada
yong.zhang3@huawei.com

Abstract. Deep neural networks (DNNs) for image classification are known to be susceptible to many diversified universal adversarial perturbations (UAPs), where each UAP successfully attacks a large but substantially different set of images. Properly combining the diversified UAPs can significantly improve the attack effectiveness, as the sets of images successfully attacked by different UAPs are complementary to each other. In this paper, we study this novel type of attack by developing a cocktail universal adversarial attack framework. The key idea is to train a set of diversified UAPs and a selection neural network at the same time, such that the selection neural network can choose the most effective UAP when attacking a new target image. Due to the simplicity and effectiveness of the cocktail attack framework, it can be generally used to significantly boost the attack effectiveness of many classic single-UAP methods that use a single UAP to attack all target images. The proposed cocktail attack framework is also able to perform real-time attacks as it does not require additional training or fine-tuning when attacking new target images. Extensive experiments demonstrate the outstanding performance of cocktail attacks.

Keywords: Universal adversarial perturbation · Cocktail attack

1 Introduction

Deep neural networks (DNNs) for image classification are well known to be susceptible to universal adversarial perturbations (UAPs) [23]. A UAP is an adversarial perturbation that can be directly added to many target images to change a victim DNN's predictions on them [39]; and it often generalizes well in attacking different target images and transfers well in attacking different victim DNNs [7,19,23,34,39]. Many classic single-UAP methods [7,19,23,27,33,34,39]

⋆ This work is done by Shaoxin Li during his visit at McMaster University when supervised by the corresponding author Lingyang Chu.

train a single UAP to perform adversarial attacks in real time, which requires no further training or fine-tuning on the UAP when attacking new target images. As a result, a well-trained UAP posts a great potential threat to the security of DNNs.

To make it worse, a DNN image classifier is susceptible to a large number of UAPs [23, 33, 38, 40, 41]. This implies the existence of many diversified UAPs, each of which can successfully attack a large but substantially different set of images. Since the sets of images successfully attacked by the diversified UAPs are complementary to each other, properly combining multiple diversified UAPs should significantly boost the attack effectiveness. This may potentially pose an even bigger threat to the security of DNN-based image classifiers.

In order to verify and study the above security threat, we propose a novel framework named **cocktail universal adversarial attack**, or **cocktail attack** for short. The key idea is to find a set of K diversified UAPs and attack each new target image by the most effective UAP in the set. Here, the term “diversified” means that the K UAPs focus on attacking different sets of images with large discrepancies. We name the proposed attack by “cocktail” because it uses K diversified UAPs to perform strong attacks. When attacking a new target image, the most effective UAP is selected by a selection neural network that is trained simultaneously with the UAPs. The cocktail attack framework significantly improves the attack effectiveness because, if each UAP selected by the selection neural network can successfully attack the corresponding target image, then the cocktail attack will successfully attack the union of the images that are successfully attacked by each of the K diversified UAPs. Meanwhile, a cocktail attack can also be conducted in real time, because no additional training or fine-tuning is required when attacking a new target image. Comparing to classic single-UAP methods [7, 19, 23, 27, 33, 34, 39] that train a single UAP to attack all target images, the only extra overhead of conducting a cocktail attack is a forward pass of the target image through the selection neural network, which takes about 2.4 milliseconds on an NVIDIA RTX 3060 GPU.

As far as we know, the cocktail attack is a novel framework of universal adversarial attack that has not been systematically studied in the literature. As discussed later in Sec. 2, most existing single-UAP methods [7,19,23,27,33,34,39] focus on finding a single UAP that successfully attacks the largest group of images. These methods pay less attention on finding and combining diversified UAPs, thus missing the opportunity to harness the power of cocktail universal adversarial attacks.

In this paper, we propose a novel cocktail attack framework against DNN-based image classification models. The proposed attack framework can be generally applied on single-UAP methods to significantly boost the attack effectiveness while achieving a fast attack speed in real time. We make the following contributions. **First**, in order to obtain the set of K diversified UAPs used to conduct cocktail universal adversarial attacks, we formulate the task of finding diversified universal adversarial perturbations (FDUAP) as a clustering problem that is shown to be NP-hard. The goal of FDUAP is to find K clusters of images while generating the corresponding UAP for each of the clusters, such that the number of successfully attacked images is maximized. **Second**, we use a selection neural network to relax the clustering problem from a discrete optimization problem to a continuous optimization problem. The key idea is to substitute the discrete variable indicating the membership of each target image to a cluster of images by the probability of cluster membership predicted by the selection neural network. **Third**, we solve the continuous optimization problem by a gradient-based method, which simultaneously trains the set of K diversified UAPs and the selection neural network in an end-to-end manner. After training, the selection neural network can efficiently select one of the K diversified UAPs to attack a new target image, which makes it possible to perform fast cocktail attacks in real time. **Last**, we conduct extensive experiments to demonstrate the outstanding performance of the proposed cocktail attack framework, which significantly boosts the attack effectiveness of four representative single-UAP methods [27, 34, 39, 40] while keeping the attacks efficient in real time.

2 Related Works

How to conduct cocktail universal adversarial attacks on DNNs is a novel problem that has not been systematically studied in the literature. It is broadly related to the following two categories of existing adversarial attacks on DNNs.

The **image-dependent adversarial attacks** [2, 4, 8, 22, 24, 29–31, 33] aim to attack a target image by adding a specifically customized perturbation to the image, such that the victim DNN makes a false prediction on the perturbed image.

One subcategory of image-dependent adversarial attacks is the **optimization-based methods** [2,4,8,22,24,31]. These methods train each perturbation from scratch for every single target image by solving an optimization problem, which is computationally expensive to produce one perturbation per target image. Therefore, these methods cannot achieve real-time attacks on new target images [38, 40].

Another subcategory of image-dependent adversarial attacks is the **generator-based methods** [29,30,33]. These methods generate a perturbation much faster than the optimization-based methods. The key idea is to train a generator network that maps each new target image into a unique perturbation to attack the target image. Generating a perturbation for a new target image is done by a single forward pass of the target image through the generator network. Therefore, the generator-based methods can achieve real-time attacks on new target images.

The **single-UAP methods** [3, 5, 23, 26, 38, 40] are well known for their ability to conduct real-time attacks. This is achieved by training a single UAP to successfully attack as many target images as possible, such that the trained UAP can be directly used to attack new target images without any additional training or fine-tuning.

The existence of UAP was first discovered by [23]. Afterwards, many single-UAP methods were developed for different application contexts. The data-efficient method [17] generated a UAP with a small amount of training data. The data-free methods [20, 21, 25, 26, 28, 39, 40] went further to generate a UAP without accessing the training dataset of victim DNNs. The gradient-based methods [7, 34, 39, 40] achieved superior attack effectiveness by directly optimizing the UAP by stochastic gradients. The class-discriminative methods [1,13,38] attempted to craft a UAP specific to images of a chosen group of classes, while having limited influence on the other classes. The double-targeted method [3] extended the class-discriminative methods by shifting images of certain classes to a different class of choice. The GAN-based methods [14,27,33] implicitly modelled the distribution of UAPs for a victim DNN by training a generative adversarial network (GAN) [11]. The above methods focus on performing real-time attacks by using a single trained UAP. However, the number of images that can be successfully attacked by a single UAP is often limited, which reduces the attack effectiveness of these methods [40].

In our work, we discovered that a DNN is usually susceptible to many diversified UAPs, each of which can successfully attack a substantially different set of images. The sets of images successfully attacked by the diversified UAPs are often complementary to each other. Based on these insights, we develop a novel cocktail attack framework for universal adversarial attacks, which finds a set of K diversified UAPs and attacks a new target image by the most effective UAP chosen by a well-trained selection neural network.

As demonstrated by the extensive experiments in Sec. 4, the cocktail attack framework significantly improves the attack effectiveness of the classic single-UAP methods [27, 34, 38, 40], and the resulting cocktail attacks even achieve superior attack effectiveness than the image-dependent generator-based attack methods [29, 33]. Meanwhile, the cocktail attacks can also be conducted in real time since no additional training or fine-tuning is required when attacking new target images.

3 Cocktail Universal Adversarial Attack

In this section, we introduce how to find a set of diversified UAPs and then use them to launch fast cocktail universal adversarial attacks. Following the prior works [19, 23, 27, 32, 38, 39], we focus on attacking DNN models for image classification task.

3.1 The Task of Finding Diversified UAPs

In this subsection, we define the task of finding diversified UAPs, as finding a set of diversified UAPs is the first step to launch successful cocktail universal adversarial attacks.

Denote by $\mathcal{X} \subseteq \mathbb{R}^D$ a distribution of images, and by $X = \{x_i\}_{i=1}^N \subset \mathcal{X}$ a set of N training images with a set of class labels, denoted by $\mathcal{C} = \{1, 2, \ldots, C\}$.

The victim model to attack is a DNN, denoted by f, which is trained on X to perform a classification task. For a target image $x_i \in X$, denote by $l_f(x_i) \in \mathcal{C}$ the class label of x_i predicted by f. The goal of the attacker is to add an adversarial perturbation $\delta \in \mathbb{R}^D$ to the target image x_i, such that $l_f(x_i) \neq l_f(x_i + \delta)$. The process of adding δ to x_i is referred to as an **attack** [3, 27, 34, 38]. If $l_f(x_i) \neq l_f(x_i + \delta)$, then we say x_i is **successfully attacked** by δ. We refer to a UAP as a perturbation when the context is clear.

Now, we define the task of finding diversified UAPs (FDUAP) as follows.

Definition 1. *Given a victim DNN f to be attacked, the training dataset $X = \{x_i\}_{i=1}^N$ used to train f, and a real-valued perturbation magnitude $\xi > 0$. The task of **finding diversified universal adversarial perturbations (FDUAP)** is to find a set of K perturbations, denoted by $P = \{\delta_1, \ldots, \delta_K\}$, and a set of K non-overlapping clusters of images in X, denoted by $\mathcal{S} = \{S_1, \ldots, S_K\}$, such that*

1. *$S_1 \cup \ldots \cup S_K = X$ and $S_i \cap S_j = \emptyset$ when $i \neq j$;*
2. *in each cluster $S_i \in \mathcal{S}$, the number of images successfully attacked by $\delta_i \in P$ is maximized; and*
3. *for each $\delta_i \in P$, $\|\delta_i\|_\infty \leq \xi$.*

The key idea of FDUAP is to learn a set of K non-overlapping clusters of X, denoted by $\mathcal{S} = \{S_1, \ldots, S_K\}$, and use each cluster $S_i \in \mathcal{S}$ to train a perturbation $\delta_i \in P$. Meanwhile, we also limit the infinity norm of δ by a small perturbation magnitude ξ such that δ is visually imperceptible to humans [27, 32, 37, 38]

Since each perturbation is trained on a different cluster of images, the sets of images successfully attacked by different perturbations are likely to be different. Therefore, we regard these perturbations as diversified. We also simultaneously optimize the perturbations in P and the clusters of images in $\mathcal{S}$, such that, in each cluster S_i, the number of images successfully attacked by δ_i is maximized. According to the previous works [3, 23, 34, 38, 40], maximizing the number of training images successfully attacked by δ_i will improve the generalizability of δ_i in successfully attacking many other images. Therefore, these perturbations in P are universally applicable when attacking the images in their corresponding clusters.

3.2 Formulate the FDUAP Task

To formulate the FDUAP task into an optimization problem, we first model the set of clusters $\mathcal{S}$ by a binary matrix $A \in \{0,1\}^{N \times K}$ with N rows and K columns. The entry $A_{i,j} \in \{0,1\}$ at the i-th row and the j-th column of A defines whether the i-th image $x_i \in X$ belongs to the j-th cluster $S_j \in \mathcal{S}$. That is, $A_{i,j} = 1$ when $x_i \in S_j$, and $A_{i,j} = 0$ when $x_i \notin S_j$. Since $S_1 \cup \ldots \cup S_K = X$ and $S_i \cap S_j = \emptyset$ when $i \neq j$, the matrix A should satisfy $\sum_{j=1}^K A_{i,j} = 1, \forall i \in \{1, \ldots, N\}$.

To train the perturbations in P, a loss function $L_f(x_i, \delta_j)$ is required to measure the similarity between the predictions of an image x_i in the victim

DNN f before and after the perturbation δ_j is added to x_i [19, 32]. A smaller value of $L_f(x_i, \delta_j)$ indicates f is more likely to change its prediction on x_i, thus the attack launched by adding δ_j to x_i is more likely to be successful [19, 32, 38].

Many effective loss functions have been proposed by different single-UAP methods in the literature [42]. It is beyond the scope of this work to develop a new loss function. Instead, we focus on developing the cocktail attack framework, which can be generally applied on different single-UAP methods to significantly boost their attack effectiveness.

We formulate the FDUAP task as the following **mixed integer programming (MIP) problem**.

$$\begin{aligned}
\min_{A,P} \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} A_{i,j} L_f(x_i, \delta_j), \\
\text{s.t.} \quad & A \in \{0,1\}^{N\times K}, \\
& \sum_{j=1}^{K} A_{i,j} = 1, \forall i \in \{1, \ldots, N\}, \\
& \|\delta_j\|_\infty \le \xi, \forall j \in \{1, \ldots, K\},
\end{aligned} \tag{1}$$

where $A_{i,j}$ ensures each perturbation $\delta_j \in P$ is trained on the corresponding cluster $S_j \in \mathcal{S}$. Minimizing the objective function in Eq. (1) will maximize the number of successfully attacked images in each cluster, which produces a solution to the FDUAP task. We prove the MIP problem is NP-hard in the appendix.

Directly solving the MIP problem in Eq. (1) faces **two issues** that prevent the development of a successful cocktail attack. **First**, we cannot solve the MIP problem by standard gradient-based methods in an end-to-end manner, since it requires to optimize the discrete binary matrix A. This causes an issue because the perturbations in P are often optimized by a gradient-based optimization method to achieve good attack effectiveness [7, 34, 39, 40]. **Second**, the binary matrix A cannot assign a new target image x_{new} that is not contained in X to any cluster in $\mathcal{S}$, which means we cannot use A to select the most effective perturbation from P to attack x_{new}.

Next, we introduce our solution to tackle these issues by introducing a neural network to relax the MIP problem to a continuous optimization problem.

3.3 Our Solution

Our solution to relax the MIP problem is to parametrize each entry $A_{i,j}$ in the matrix A by a neural network g_θ named **selection neural network**. Specifically, g_θ maps a target image x_i to a K-dimensional vector $\mathbb{P}_\theta(x_i)$, where the j-th entry, denoted by $\mathbb{P}_\theta(x_i)_j$, is the predicted value of $A_{i,j}$. In plain words, g_θ predicts the probability $\mathbb{P}_\theta(x_i)_j$ for a target image x_i to be a member of a cluster S_j in $\mathcal{S}$. By substituting each entry $A_{i,j}$ with $\mathbb{P}_\theta(x_i)_j$, we convert the MIP problem into the following **continuous optimization (CO) problem**.

$$\begin{aligned} \min_{\theta, P} \quad & \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{P}_\theta(x_i)_j L_f(x_i, \delta_j), \\ \text{s.t.} \quad & \|\delta_j\|_\infty \leq \xi, \forall j \in \{1, \ldots, K\}. \end{aligned} \tag{2}$$

The CO problem successfully tackles the first issue of the MIP problem since it allows us to train the perturbations in P by gradient-based methods in an end-to-end manner. In this work, we propose a training algorithm to alternately optimize the selection neural network g_θ and the set of perturbations in P until the objective function in Eq. (2) becomes stable. Specifically, we update g_θ by the ADAM optimizer [18] since it is a classic method to train neural networks. We update the perturbations in P by the projected gradient descend (PGD) to handle the convex constraints $\|\delta_j\|_\infty \leq \xi$ in Eq. (2), which ensures the feasibility of each perturbation $\delta_j \in P$.

Algorithm 1 summarizes the details of our proposed training algorithm, which trains the selection neural network g_θ and the set of perturbations in P by stochastic gradient descent.

The CO problem also tackles the second issue of the MIP problem. Because, for a new target image x_{new}, the K entries in $\mathbb{P}_\theta(x_{new})$ give the probabilities for x_{new} to belong to each of the K clusters in $\mathcal{S}$. Therefore, we can first assign x_{new} to the cluster with the largest probability to contain x_{new}, and then select the corresponding perturbation in P to attack x_{new}. Algorithm 2 summarizes the details of conducting a cocktail attack on a new target image x_{new} using the K trained perturbations in P and the trained selection neural network g_θ.

The incorporation of g_θ also allows us to conduct cocktail attacks in real time, because selecting the perturbation to attack a new target image x_{new} only requires feeding x_{new} into g_θ for a single forward pass, which only takes around 2.4 milliseconds on an NVIDIA RTX 3060 GPU.

4 Experiments

In this section, we investigate the performance of the cocktail attack framework in boosting the attack effectiveness of single-UAP methods. Due to the limit of space, we discuss the following in the appendix: 1) the dominated classes of diversified UAPs; 2) the attack robustness against typical defense methods; 3) the training efficiency of our method; and 4) the hyperparameter analysis on K.

4.1 Experimental Settings

Datasets. Following the routine of the prior works [23, 27], we use a subset of ImageNet [6] to conduct our experiments. Both the training and testing datasets contain 1,000 categories of images. The training dataset is sampled from the original training dataset of ImageNet. It contains 10,000 images, that is, 10

Algorithm 1: Solving the CO problem

```
   Inputs : The victim DNN $f$, the training dataset $X$, the number of
            perturbations $K$, and the perturbation magnitude $\xi$.
   Output: The trained perturbations in $P$ and the selection neural network $g_\theta$.
 1 Initialize each perturbation in $P$ by zeros.
 2 Initialize the parameters $\theta$ of $g_\theta$.
 3 do
 4     $X_B \leftarrow$ Sample a batch of images from $X$.
 5     Compute the average loss on $X_B$ by Eq. (2).
 6     Compute the gradients with respect to $P$.
 7     Update $P$ by the projected gradient descent [10].
 8     Compute the average loss on $X_B$ by Eq. (2).
 9     Compute the gradients with respect to $\theta$.
10     Update $\theta$ by the ADAM optimizer [18].
11 while not converge;
12 return $P$ and $g_\theta$.
```

Algorithm 2: Conducting a cocktail attack

```
   Inputs : The $K$ trained perturbations in $P$, the trained selection neural
            network $g_\theta$ and a new image $x_{new}$.
   Output: The perturbed image $x'_{new}$
 1 Compute: $\mathbb{P}_\theta(x_{new}) = g_\theta(x_{new})$.
 2 Select: $j^* \leftarrow \text{argmax}_j \, \mathbb{P}_\theta(x_{new})_j$.
 3 Perturb: $x'_{new} \leftarrow x_{new} + \delta_{j^*}$.
 4 Clip the pixel values of $x'_{new}$ to the range of $[0, 255]$.
 5 return Perturbed image $x'_{new}$.
```

images per category. The testing dataset is exactly the validation dataset of ImageNet, which contains 50,000 images. The training dataset is used for the training of attack methods and the testing dataset is used to evaluate the attack performance of different attack methods.

Victim models and selection neural network. We evaluate the attack performance of different attack methods against six victim DNNs, including ResNet-50 [15], VGG-16 [35], Inception-V3 [36], SqueezeNet [16], ViT [9] and LeViT [12]. All the victim models are pre-trained on ImageNet, and they are not modified during the training of attack methods. We adopt SqueezeNet [16] as the architecture of the selection neural network due to its lightweight structure [16].

Baselines. We adopt four representative single-UAP methods, including UAT [34], DF-UAP [39] written in short as DF, Cosine-UAP [40] written in short as COS, and NAG [27], to verify the performance of the proposed cocktail attack framework in boosting the attack effectiveness. We also compare with two image-dependent generator-based attack methods, including GAP [33] and TDA [29].

We apply the proposed cocktail attack framework to each of the above single-UAP methods in the following two different ways. **Cocktail attack type 1 (CK1)**. We first run each single-UAP method to generate a set of K UAPs. For UAT, DF and COS, we independently run each method for K times on the training dataset to obtain the set of K UAPs. Since NAG trains a UAP generator to produce UAPs, we generate K UAPs by independently sampling a UAP using the generator for K times. After obtaining the K UAPs, we regard them as a constant set of perturbations, denoted by P, and solve the CO problem in Eq. (2) to train the selection neural network g_θ. We do not execute the step 7 to update P when calling Algorithm 1. Therefore, CK1 can be viewed as an ablated version of the proposed cocktail attack framework since it does not train P and g_θ together. **Cocktail attack type 2 (CK2)**. This is the complete version of the proposed cocktail attack framework as described in Algorithm 1, which solves the CO problem in Eq. (2) to train P and g_θ together.

When applying the cocktail attack types CK1 and CK2 to a single-UAP method, we adopt the same loss function $L_f(x_i, \delta_j)$ as the single-UAP method when solving the CO problem in Eq. (2). Comparatively, the single-UAP method only finds a single UAP and uses it to attack all target images, but CK1 and CK2 find K diversified UAPs to conduct cocktail attacks. We denote by UAT-CK1, DF-CK1, COS-CK1, and NAG-CK1 the cocktail attack methods when applying CK1 to each of the single-UAP methods; and by UAT-CK2, DF-CK2, COS-CK2, and NAG-CK2 the cocktail attack methods when applying CK2. When the context is clear, we omit the names of the single-UAP methods and directly use CK1 and CK2 to refer to the corresponding cocktail attack methods of a single-UAP method.

Implementation details. For the baseline methods, we adopt the default hyperparameters used in their implementations. For the proposed cocktail attack framework, we use $K = 5$ if not otherwise specified. For both CK1 and CK2, we use a batch size of $\lceil \frac{64}{K} \rceil$. For both the ADAM optimizer and the PGD, we use a learning rate of 10^{-3}. The number of training epochs is set to 100. The architecture of the selection neural network g_θ is SqueezeNet [16], which has 18 layers and we use K output neurons in the last layer. The implementations are realized using Pytorch version 1.11.0 with CUDA version 11.3. All experiments are conducted on a server with a NVIDIA 3060 GPU, 32GB main memory, and an Intel(R) Core(TM) i9-10900F CPU @ 2.80GHz.

Evaluation metrics. Following the prior works [23, 27, 34], we adopt the *fooling ratio (FR)* on the testing dataset to evaluate the attack effectiveness. It is computed as the ratio between the successfully attacked images and the total number of attacked images. A larger FR value indicates a better attack effectiveness. We also evaluate the *average attack time (AAT)* of each attack method by measuring the average time cost of generating the perturbed image of a target image. It is defined as AAT $= \frac{1}{|T|} \sum_{x_i \in T} t(x_i)$, where T is the set of testing images, $|T|$ is the number of testing images in T, and $t(x_i)$ is the time cost to generate a perturbed image for x_i. A smaller value of AAT implies a faster attack speed. We report AAT in milliseconds by default.

Table 1: The FR of all the attack methods when $K = 5$ and $\xi \in \{2, 6, 10\}$. Bold and underlined numbers show the best and runner up performance.

$K = 5$	ξ	GAP	TDA	(UAT, CK1, CK2)	(DF, CK1, CK2)	(COS, CK1, CK2)	(NAG, CK1, CK2)
ResNet-50	2	25.5	<u>27.8</u>	(19.0, 21.1, 24.7)	(21.6, 24.2, **29.0**)	(21.2, 24.6, 28.3)	(18.5, 24.5, 27.6)
	6	84.2	<u>84.4</u>	(73.6, 77.6, **87.6**)	(72.7, 76.6, 87.0)	(66.6, 71.8, 80.5)	(68.5, 75.7, 79.2)
	10	96.5	<u>98.4</u>	(94.1, 96.7, 98.1)	(94.6, 96.8, **99.0**)	(92.3, 94.6, 97.7)	(91.1, 96.3, 97.9)
VGG-16	2	<u>41.0</u>	40.7	(36.2, 38.5, **44.0**)	(35.3, 37.9, 42.7)	(32.6, 35.3, 42.6)	(35.1, 39.1, 42.9)
	6	<u>89.1</u>	87.5	(83.8, 87.5, 93.0)	(85.6, 89.4, 94.5)	(87.1, 90.4, 95.3)	(82.5, 91.5, **95.8**)
	10	98.7	<u>99.0</u>	(95.3, 96.2, 98.5)	(94.1, 96.7, **99.4**)	(95.4, 97.3, 99.2)	(92.2, 97.2, 99.1)
Inception-V3	2	34.2	<u>35.4</u>	(23.6, 26.9, 34.7)	(26.4, 29.5, **38.6**)	(26.4, 29.9, 37.0)	(24.0, 30.2, 37.0)
	6	81.7	<u>84.9</u>	(73.3, 78.6, **85.4**)	(72.2, 76.7, 85.1)	(70.0, 74.4, 83.8)	(70.3, 78.8, 84.3)
	10	95.4	<u>97.8</u>	(94.4, 97.5, **99.5**)	(93.7, 95.5, 99.0)	(92.8, 95.4, 98.3)	(90.3, 96.3, 99.0)
SqueezeNet	2	33.3	<u>36.5</u>	(26.0, 29.9, **36.9**)	(25.3, 29.0, 35.3)	(27.5, 30.0, 35.4)	(24.2, 33.7, 34.3)
	6	87.5	<u>89.0</u>	(80.0, 83.7, 90.6)	(79.3, 82.2, **91.2**)	(77.5, 81.8, 89.2)	(76.3, 83.2, 88.3)
	10	98.3	<u>98.6</u>	(94.9, 96.3, 98.8)	(94.3, 96.1, **99.5**)	(93.8, 95.0, 99.4)	(91.0, 95.3, 99.3)
ViT	2	35.5	<u>36.1</u>	(25.3, 28.6, 35.1)	(29.5, 32.4, **40.3**)	(25.6, 31.8, 39.2)	(27.9, 30.4, 37.9)
	6	83.7	<u>84.4</u>	(76.5, 80.4, **88.9**)	(75.6, 79.3, 88.3)	(73.4, 78.1, 86.2)	(73.6, 80.3, 86.5)
	10	96.7	<u>97.9</u>	(95.2, 98.6, **99.6**)	(95.1, 98.3, 99.5)	(94.9, 97.8, 99.1)	(94.3, 98.7, 99.2)
LeViT	2	34.9	<u>35.8</u>	(24.6, 27.7, 35.0)	(27.4, 31.2, **40.2**)	(26.1, 30.5, 39.0)	(25.7, 31.4, 36.7)
	6	81.3	<u>83.6</u>	(74.2, 78.1, **87.6**)	(73.5, 78.5, 85.9)	(73.8, 76.6, 84.3)	(71.2, 79.6, 84.4)
	10	96.3	<u>98.2</u>	(94.6, 98.2, 99.1)	(94.2, 98.0, **99.6**)	(93.9, 97.6, 98.9)	(93.8, 98.0, 99.0)

4.2 Attack Effectiveness

In this subsection, we analyze the attack effectiveness of each attack method when the training and the testing are conducted to attack the same victim model.

Table 1 reports the FR of all the attack methods. We can see that the FR of CK2 consistently outperforms CK1. This is because the set of UAPs used by CK1 are separately trained from the selection neural network g_θ, where each UAP is trained by an independent run of a single-UAP method. The UAPs generated in this way usually do not have good diversity, which reduces the FR of CK1. For CK2, the set of UAPs are trained together with g_θ. These UAPs tend to have high diversity, as each of the UAPs is trained to attack the images in a separate cluster S_j in $\mathcal{S}$, which results in a much better FR of CK2 over CK1. We will analyze the diversity of the UAPs trained by the proposed cocktail attack framework in Sec. 4.5.

We can also see that the FR of the single-UAP methods, such as UAT, DF, COS and NAG, are inferior to the image-dependent generator-based methods such as GAP and TDA. This phenomenon has also been observed by the existing works [30, 33]. The reason for the inferior FR of the single-UAP methods is that they use a single UAP to attack all the target images, while GAP and TDA produce image-dependent attacks that customize a unique perturbation to attack each target image.

However, when the proposed cocktail attack framework is applied, the attack effectiveness of the single-UAP methods is significantly improved. We can see that DF-CK2 achieves an outstanding FR that is almost always higher than GAP and TDA. Moreover, as shown by the numbers marked in bold in each row of Tab. 1, the highest FR is always achieved by a CK2 attack method.

This demonstrates the outstanding performance of the proposed cocktail attack framework in improving the attack effectiveness of the single-UAP methods.

How could CK2 achieve a better FR than GAP and TDA by using only a small set of $K = 5$ *perturbations?* We believe it is due to the following reasons. **First**, each of the perturbations used by CK2 is a strong UAP that is well trained on the training dataset to successfully attack a large and diversified set of images. Therefore, the selection neural network $g_{\boldsymbol{\theta}}$ is only required to perform a simple 5-class classification task to select the most promising UAP from the 5 UAPs. This lowers the difficulty of designing and training an effective selection neural network to conduct high performance cocktail attacks. Therefore, CK2 can achieve an outstanding FR by simply implementing $g_{\boldsymbol{\theta}}$ as a lightweight SqueezeNet [16]. **Second**, for each of GAP and TDA, an adversarial perturbation is generated by a generator network, which is required to predict every pixel value of a unique perturbation for each target image. Comparing to developing the selection neural network $g_{\boldsymbol{\theta}}$ for a simple 5-class classification task, designing and training a generator network to generate a large number of unique perturbations is more difficult. This could be a reason for the inferior FR of GAP and TDA.

4.3 Attack Efficiency

In this subsection, we analyze the attack efficiency of different methods when conducting the previous experiments in Sec. 4.2.

Table 2 reports the AAT of all the attack methods when $K = 5$ and $\xi = 6$. The ATT results of $\xi = 2$ and $\xi = 10$ are similar to those when $\xi = 6$. Therefore, we do not report these AAT results to save the redundancy. As shown in Tab. 2, the AAT of all the compared methods are less than 10.3 milliseconds. This demonstrates the superior efficiency of universal adversarial attacks and generator-based attack methods in launching real-time attacks.

The AAT of each single-UAP method is less than 0.1 milliseconds since it produces a perturbed image by directly adding the UAP to the target image. However, the fast attack speed of the single-UAP methods is gained at the cost of only using a single UAP to attack all target images, which has been demonstrated to be the major bottleneck of their attack effectiveness by the previous experiments in Tab. 1.

The AAT of CK1 and CK2 are about 2.4 milliseconds. Comparing to the single-UAP methods, the major overhead of CK1 and CK2 is caused by calling the selection neural network g_θ to make a selection in the small set of $K = 5$ diversified UAPs. Since g_θ is a lightweight SqueezeNet [16] and making the selection only requires a forward pass through g_θ, the additional time cost induced by calling g_θ is only about 2.4 milliseconds. Considering the significant improvement of attack effectiveness achieved by the cocktail attack framework in Tab. 1, an additional time cost of 2.4 milliseconds is almost ignorable when conducting real-time adversarial attacks.

The AAT of GAP and TDA are around 10.1 milliseconds, which is slightly slower than CK1 and CK2. The major overhead of GAP and TDA is caused by the complicated generator network that maps a target image into a perturbation.

Table 2: The AAT in milliseconds of all the attack methods when $K = 5$ and $\xi = 6$.

$K = 5$, $\xi = 6$	GAP	TDA	(UAT, CK1, CK2)	(DF, CK1, CK2)	(COS, CK1, CK2)	(NAG, CK1, CK2)
ResNet-50	10.1	10.1	(<0.1, 2.3, 2.3)	(<0.1, 2.4, 2.3)	(<0.1, 2.3, 2.3)	(<0.1, 2.3, 2.3)
VGG-16	10.2	10.1	(<0.1, 2.4, 2.3)	(<0.1, 2.4, 2.3)	(<0.1, 2.4, 2.4)	(<0.1, 2.4, 2.4)
Inception-V3	10.2	10.1	(<0.1, 2.3, 2.4)	(<0.1, 2.5, 2.3)	(<0.1, 2.4, 2.4)	(<0.1, 2.3, 2.4)
SqueezeNet	10.1	10.1	(<0.1, 2.3, 2.3)	(<0.1, 2.3, 2.3)	(<0.1, 2.4, 2.3)	(<0.1, 2.4, 2.4)
ViT	10.3	10.2	(<0.1, 2.2, 2.3)	(<0.1, 2.3, 2.3)	(<0.1, 2.3, 2.3)	(<0.1, 2.3, 2.2)
LeViT	10.1	10.2	(<0.1, 2.3, 2.4)	(<0.1, 2.4, 2.4)	(<0.1, 2.4, 2.5)	(<0.1, 2.4, 2.4)

Table 3: The Trans_{f_i} of all the attack methods when $K = 5$ and $\xi \in \{2, 6, 10\}$. Bold and underlined numbers show the best and runner up performance.

$K = 5$	f_i	GAP	TDA	(UAT, CK1, CK2)	(DF, CK1, CK2)	(COS, CK1, CK2)	(NAG, CK1, CK2)
$\xi = 2$	ResNet-50	13.0	14.5	(9.6, 11.3, 14.5)	(9.7, 11.9, 15.0)	(9.9, 12.3, **15.2**)	(8.8, 12.0, 13.9)
	VGG-16	16.4	19.7	(15.6, 17.9, **21.5**)	(14.7, 16.2, 20.1)	(14.4, 16.9, 20.2)	(13.8, 17.3, 20.3)
	Inception-V3	19.3	20.8	(15.6, 17.0, **21.8**)	(16.6, 18.3, 21.7)	(15.9, 18.0, 20.6)	(15.6, 19.2, 21.1)
	SqueezeNet	16.8	20.1	(14.0, 16.7, 19.7)	(14.5, 16.6, 20.5)	(15.1, 17.4, **21.5**)	(13.9, 18.3, 20.2)
	ViT	8.8	9.7	(7.4, 8.1, 11.6)	(8.0, 9.3, **11.9**)	(7.5, 8.3, 11.7)	(7.1, 7.8, 11.0)
	LeViT	9.5	10.2	(8.0, 9.6, 12.0)	(8.4, 10.3, **12.4**)	(7.8, 9.6, 11.9)	(7.6, 8.6, 11.5)
$\xi = 6$	ResNet-50	52.8	59.3	(46.8, 51.1, 60.8)	(48.4, 52.7, **61.1**)	(45.1, 49.5, 58.4)	(46.2, 52.8, 57.0)
	VGG-16	39.1	44.0	(37.1, 40.3, **45.2**)	(35.5, 38.5, 42.2)	(34.6, 37.3, 42.3)	(33.9, 39.6, 43.9)
	Inception-V3	55.7	60.0	(51.9, 55.0, **61.0**)	(49.6, 52.6, 59.6)	(49.4, 52.5, 58.5)	(50.0, 54.3, 59.4)
	SqueezeNet	48.0	53.4	(44.1, 51.2, **55.8**)	(40.8, 42.6, 50.9)	(42.4, 46.0, 52.3)	(42.3, 45.7, 51.4)
	ViT	31.6	32.8	(27.4, 31.5, 35.2)	(27.7, 32.2, **35.8**)	(26.9, 28.4, 31.7)	(27.0, 30.8, 34.8)
	LeViT	33.9	36.0	(30.2, 34.0, 38.8)	(30.8, 35.2, **39.6**)	(29.4, 32.5, 36.5)	(29.6, 33.2, 37.5)
$\xi = 10$	ResNet-50	66.9	71.6	(66.4, 67.2, 72.5)	(65.6, 68.3, **72.9**)	(62.0, 64.1, 70.6)	(66.4, 70.0, 71.7)
	VGG-16	50.2	54.6	(46.2, 49.0, **56.3**)	(47.9, 50.2, 56.1)	(47.3, 49.1, 55.4)	(46.8, 50.5, 54.6)
	Inception-V3	71.4	75.3	(68.9, 72.5, **77.7**)	(67.4, 70.3, 75.1)	(66.5, 69.4, 76.0)	(66.3, 71.1, 76.4)
	SqueezeNet	59.0	65.0	(54.1, 57.1, 64.0)	(55.3, 58.1, **65.2**)	(52.3, 55.3, 63.8)	(56.0, 61.2, 65.0)
	ViT	39.3	43.4	(37.3, 41.6, 46.3)	(35.8, 42.0, **47.2**)	(36.4, 40.9, 46.2)	(36.6, 41.7, 46.5)
	LeViT	45.0	48.2	(42.3, 46.5, 50.5)	(44.6, 47.9, **51.0**)	(41.6, 44.8, 49.7)	(42.5, 45.9, 49.2)

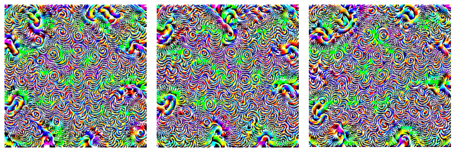
4.4 Effectiveness of Transfer Attacks

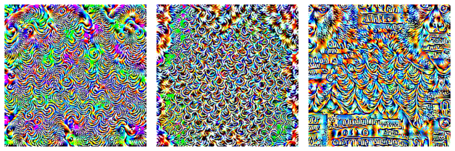
In this subsection, we analyze the transfer attack effectiveness of each method when the training is conducted to attack one victim model, named **source model**, and the testing is conducted to attack a different victim model, named **target model**.

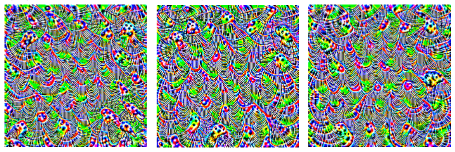
Denote by $\text{FR}_{i,j}$ the transfer attack effectiveness when transferring from a source model f_i to a target model f_j ($f_i \neq f_j$). For each method, we compute $\text{FR}_{i,j}$ in the following steps: 1) conduct the training on the training dataset to attack f_i; 2) generate the perturbed testing images on the testing dataset; 3) use the perturbed testing images to attack f_j and compute $\text{FR}_{i,j}$ as the FR of f_j.

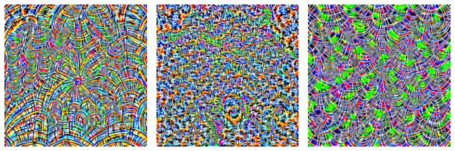
Denote by $Q = \{f_1, \ldots, f_6\}$ the six victim models targeted in our experiments. For each attack method, we measure its **average transfer attack effectiveness** when transferring from a source model $f_i \in Q$ to the other victim models $f_j \in Q \setminus f_i$ by $\text{Trans}_{f_i} = \frac{1}{5}\sum_{f_j \in Q \setminus f_i} \text{FR}_{i,j}$, which is the average of $\text{FR}_{i,j}$ when transferring from the source model f_i to each of the other target models. A higher Trans_{f_i} means a better transfer attack effectiveness.

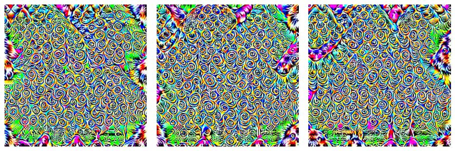
Table 3 reports the average transfer attack effectiveness of all the methods when $K = 5$ and $\xi \in \{2, 6, 10\}$. We can see that the Trans_{f_i} of CK2 consistently outperforms CK1, and both CK1 and CK2 achieve better Trans_{f_i} than the corresponding single-UAP method. Moreover, as shown by the bold numbers

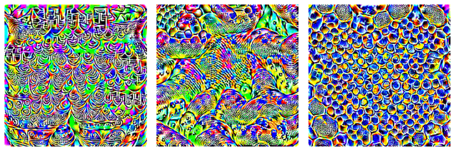

(a) UAT-CK1, FR=(74.2, 73.9, 73.4), AD=0.06

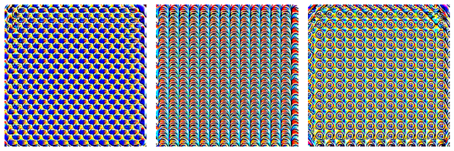

(b) UAT-CK2, FR=(73.3, 71.5, 70.9), AD=0.30


(c) DF-CK1, FR=(72.5, 72.3, 72.0), AD=0.08


(d) DF-CK2, FR=(71.5, 69.8, 67.8), AD=0.29


(e) COS-CK1, FR=(67.2, 66.8, 66.7), AD=0.12


(f) COS-CK2, FR=(66.7, 66.0, 65.9), AD=0.32


(g) NAG-CK1, FR=(68.8, 68.7, 68.5), AD=0.17

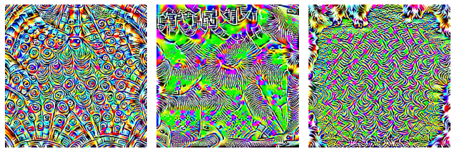

(h) NAG-CK2, FR=(69.3, 67.4, 65.2), AD=0.28

Fig. 1: The UAPs generated by CK1 and CK2, respectively, on ResNet-50 when $K = 3$ and $\xi = 6$. (a), (c), (e) and (g) show the UAPs produced by CK1. (b), (d), (f) and (h) show the UAPs produced by CK2. AD is the average diversity of the three UAPs. FR show the fooling ratio of the three UAPs in the same subfigure.

in Tab. 3, the highest Trans_{f_i} is always achieved by a CK2 attack method. These results demonstrate the outstanding performance of the proposed cocktail attack framework in improving the transfer attack effectiveness of the single-UAP methods.

We explain the superior transferability of CK2 as follows. **First**, due to the good transferability of UAP [19, 23, 39], each of the K UAPs transfers well from attacking the source model to the other target models. **Second**, since the K UAPs are diversified, they tend to successfully transfer-attack different sets of images. **Third**, g_θ also transfers well in selecting the most effective UAP to successfully attack different models. **In sum**, CK2 achieves superior transferability because it successfully attack the union of the images successfully transfer-attacked by each of K UAPs. The better transferability of CK2 over CK1 is due to the higher diversity of the UAPs trained by CK2.

4.5 The Diversity of UAPs: A Case Study

In this subsection, we conduct a case study in Fig. 1 to visualize the UAPs produced by CK1 and CK2, respectively. For both CK1 and CK2, we adopt

$K = 3$ and $\xi = 6$, and use ResNet-50 as the victim model. Each of CK1 and CK2 produces a set of K UAPs, denote by P, where every UAP in P is visualized in the same way as [23, 40].

We measure the diversity of the K UAPs by the **average diversity (AD)**, $\text{AD} = \frac{1}{\binom{K}{2}} \sum_{i \neq j} D_{i,j}$, where $D_{i,j}$ is the diversity between the pair of UAPs δ_i and δ_j in P on the testing dataset. We compute $D_{i,j}$ by $D_{i,j} = 1 - \frac{|V_i \cap V_j|}{|V_i \cup V_j|}$, where V_i and V_j are the sets of testing images successfully attacked by δ_i and δ_j, respectively. A larger $D_{i,j}$ means a greater difference between V_i and V_j, which implies a greater diversity between δ_i and δ_j, thus producing a higher AD.

In Fig. 1, each set of UAPs produced by CK1 shows similar visual patterns and they have a small value of AD. Take Fig. 1a as an example, the three UAPs generated by UAT-CK1 are very similar in both global and local patterns. Figure 1c and Fig. 1e show similar results for the UAPs generated by DF-CK1 and COS-CK1, respectively. Such results are also observed in some previous works [14, 23, 27, 40], which indicated that multiple independent runs of a single-UAP method usually generate similar UAPs that tend to successfully attack similar sets of images. This is the major cause for the small AD of the UAPs generated by CK1. For NAG-CK1, since many different UAPs can be sampled from the generator trained by NAG, the UAPs in Fig. 1g show a similar grid-like global pattern but they have dissimilar local patterns. Therefore, NAG-CK1 achieves a higher AD compared to the other CK1 attack methods.

We can also see from Fig. 1 that each set of UAPs produced by CK2 shows substantially different appearances and they have a much larger AD than the UAPs generated by CK1. This is because CK2 generates a set of diversified UAPs by training each UAP to attack a separate cluster of images $S_j \in \mathcal{S}$.

As shown in the captions of subfigures in Fig. 1, the FR of the UAPs generated by CK1 is slightly larger than those generated by CK2. This can be explained by the fact that each UAP generated by CK1 is trained on the entire training dataset, but a UAP generated by CK2 is trained on a cluster of the training dataset, which contains less training images. However, since the UAPs generated by CK2 have a much larger AD than the UAPs of CK1, each UAP in P generated by CK2 is able to successfully attack a large number of new images that cannot be successfully attacked by the other UAPs in P. This explains the experimental results in Sec. 4.2, Sec. 4.4, where the attack effectiveness of CK2 consistently outperforms CK1.

5 Conclusion

In this paper, we propose a novel cocktail attack framework that can be generally applied to existing single-UAP methods to significantly boost their attack effectiveness while achieving a fast attack speed. The key idea is to simultaneously train a set of diversified UAPs and a selection neural network, and use the selection neural network to choose the most effective UAP when attacking a new target image. Extensive experiments show the superior performance of the proposed cocktail attack framework.

Acknowledgements

This work is supported in part by the start-up funding of McMaster University, in part by National Natural Science Foundation of China (Grant no. 61932006 and 62272073), in part by the Natural Science Foundation of Chongqing (Innovation and Development Joint Fund) under grant CSTB2023NSCQ-LZX0149, in part by the Fundamental Research Funds for the Central Universities under grant 2023CDJKYJH019, in part by the scholarship from China Scholarship Council.

References

1. Akhtar, N., Jalwana, M.A., Bennamoun, M., Mian, A.: Label universal targeted attack. arXiv preprint arXiv:1905.11544 (2019)
2. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: International Conference on Machine Learning. pp. 274–283 (2018)
3. Benz, P., Zhang, C., Imtiaz, T., Kweon, I.S.: Double targeted universal adversarial perturbations. In: Proceedings of the Asian Conference on Computer Vision (2020)
4. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy. pp. 39–57 (2017)
5. Chaubey, A., Agrawal, N., Barnwal, K., Guliani, K.K., Mehta, P.: Universal adversarial perturbations: A survey. arXiv preprint arXiv:2005.08087 (2020)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
7. Deng, Y., Karam, L.J.: Universal adversarial attack via enhanced projected gradient descent. In: IEEE International Conference on Image Processing. pp. 1241–1245 (2020)
8. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9185–9193 (2018)
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
10. Goldstein, A.A.: Convex programming in hilbert space. Bulletin of the American Mathematical Society **70**(5), 709–710 (1964)
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in Neural Information Processing Systems **27** (2014)
12. Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., Douze, M.: Levit: a vision transformer in convnet's clothing for faster inference. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12259–12269 (2021)
13. Gupta, T., Sinha, A., Kumari, N., Singh, M., Krishnamurthy, B.: A method for computing class-wise universal adversarial perturbations. arXiv preprint arXiv:1912.00466 (2019)
14. Hayes, J., Danezis, G.: Learning universal adversarial perturbations with generative models. In: IEEE Security and Privacy Workshops. pp. 43–49 (2018)

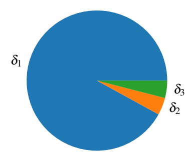
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
16. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and$<$ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
17. Khrulkov, V., Oseledets, I.: Art of singular vectors and universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8562–8570 (2018)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Li, M., Yang, Y., Wei, K., Yang, X., Huang, H.: Learning universal adversarial perturbation by adversarial example. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 1350–1358 (2022)
20. Li, Y., Bai, S., Xie, C., Liao, Z., Shen, X., Yuille, A.: Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. In: Proceedings of the European Conference on Computer Vision. pp. 795–813 (2020)
21. Liu, H., Ji, R., Li, J., Zhang, B., Gao, Y., Wu, Y., Huang, F.: Universal adversarial perturbation via prior driven uncertainty approximation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2941–2949 (2019)
22. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
23. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1765–1773 (2017)
24. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2574–2582 (2016)
25. Mopuri, K.R., Ganeshan, A., Babu, R.V.: Generalizable data-free objective for crafting universal adversarial perturbations. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(10), 2452–2465 (2018)
26. Mopuri, K.R., Garg, U., Babu, R.V.: Fast feature fool: A data independent approach to universal adversarial perturbations. arXiv preprint arXiv:1707.05572 (2017)
27. Mopuri, K.R., Ojha, U., Garg, U., Babu, R.V.: Nag: Network for adversary generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 742–751 (2018)
28. Mopuri, K.R., Uppala, P.K., Babu, R.V.: Ask, acquire, and attack: Data-free uap generation using class impressions. In: Proceedings of the European Conference on Computer Vision. pp. 19–34 (2018)
29. Nakka1, K.K., Salzmann, M.: Learning transferable adversarial perturbations. Advances in Neural Information Processing Systems **34**, 13950–13962 (2021)
30. Naseer, M.M., Khan, S.H., Khan, M.H., Shahbaz Khan, F., Porikli, F.: Cross-domain transferability of adversarial perturbations. Advances in Neural Information Processing Systems **32** (2019)
31. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy. pp. 372–387. IEEE (2016)
32. Park, S.M., Wei, K.A., Xiao, K., Li, J., Madry, A.: On distinctive properties of universal perturbations. arXiv preprint arXiv:2112.15329 (2021)

33. Poursaeed, O., Katsman, I., Gao, B., Belongie, S.: Generative adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4422–4431 (2018)
34. Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L.S., Goldstein, T.: Universal adversarial training. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5636–5643 (2020)
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
36. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
37. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. arXiv preprint arXiv:1805.12152 (2018)
38. Zhang, C., Benz, P., Imtiaz, T., Kweon, I.S.: Cd-uap: Class discriminative universal adversarial perturbation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 6754–6761 (2020)
39. Zhang, C., Benz, P., Imtiaz, T., Kweon, I.S.: Understanding adversarial examples from the mutual influence of images and perturbations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14521–14530 (2020)
40. Zhang, C., Benz, P., Karjauv, A., Kweon, I.S.: Data-free universal adversarial perturbation and black-box attack. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7868–7877 (2021)
41. Zhang, C., Benz, P., Karjauv, A., Kweon, I.S.: Universal adversarial perturbations through the lens of deep steganography: Towards a fourier perspective. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 3296–3304 (2021)
42. Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., Kweon, I.S.: A survey on universal adversarial attack. arXiv preprint arXiv:2103.01498 (2021)

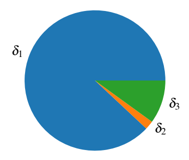
Appendix

A The Dominated Classes of Diversified UAPs

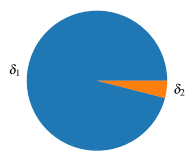
In this subsection, we use the three diversified UAPs generated by UAT-CK2 as an example to investigate the distribution of images attacked by each of the diversified UAPs. The pie chart in each subfigure of Fig. 2 shows the results for a specific class. In each pie chart, the three regions in blue, orange and green show the proportion of images attacked by the diversified UAPs δ_1, δ_2 and δ_3, respectively. The diversified UAPs δ_1, δ_2 and δ_3 are visualized in Fig. 1b from left to right.



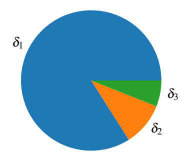
(a) Toaster



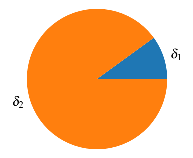
(b) Dumbbell



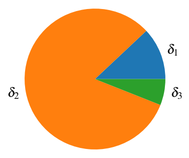
(c) Planetarium



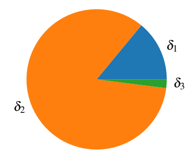
(d) Car wheel



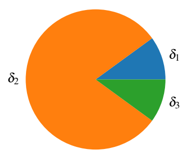
(e) Custard apple



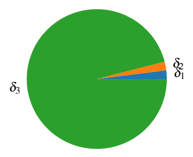
(f) Knee pad



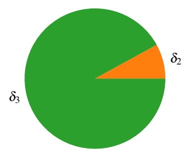
(g) Ocarina



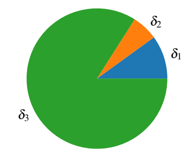
(h) Tabby cat



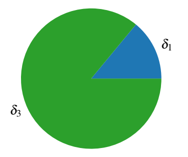
(i) French loaf



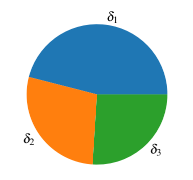
(j) Lorikeet



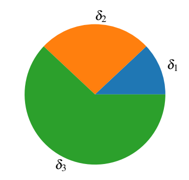
(k) Seatbelt



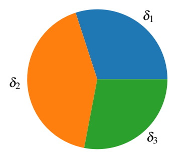
(l) Flute



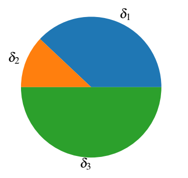
(m) Pomeranian



(n) Horizontal bar



(o) Ring-binder



(p) Head cabbage

Fig. 2: The proportion of images in the same class attacked by each of the diversified UAPs. $\delta_1, \delta_2, \delta_3$ are the three UAPs in Fig. 1b (from left to right). The caption of each subfigure is the name of an image class. The area of each coloured region shows the proportion of images attacked by one UAP.

The 1,000 classes of images in our dataset produce 1,000 pie charts, which cannot be shown in the paper due to the limit of space. However, we discovered that these classes generally fall into four typical categories, thus we show some examples in each category in Fig. 2.

The first category of classes, such as "Toaster", "Dumbbell", "Planetarium" and "Car Wheel", are shown in the first row of Fig. 2. For these classes, the UAP δ_1 is selected by the selection neural network g_θ to attack most of the images. Such classes are dominated by δ_1, because δ_1 is likely the most effective in attacking the images in these classes. Similarly, the second category of classes shown in the second row of Fig. 2, such as "Custard apple", "Knee pad", "Ocarina" and "Tabby cat", are dominated by δ_2. The third category of classes shown in the third row of Fig. 2 are dominated by δ_3. The fourth category of classes shown in the fourth row of Fig. 2 are not significantly dominated by any UAP, because the proportions of images attacked by each of the three UAPs tend to be balanced. This is because all the UAPs are comparably effective in attacking the images in these classes, thus the selection neural network g_θ makes a more balanced selection when attacking images in such classes.

The above results demonstrate an interesting class specific property of the diversified UAPs trained by our cocktail attack framework. That is, each diversified UAP is highly effective in dominating the attack of certain classes, and the sets of classes dominated by different UAPs are different. Such a property is the key to the outstanding performance of the proposed cocktail attack framework, because, when the selection neural network g_θ chooses the most effective UAP to attack each new image, the final set of successfully attacked images will be the union of the images successfully attacked by every UAP.

B Attack Robustness

In this section, we investigate the robustness of different attack methods by evaluating their attack effectiveness in the presence of different defenses. Specifically, we consider four image transformations-based defenses presented in [13], including JPEG, Crop and rescale, total variance (TV) minimization and bit depth reduction. Each of these defenses is applied on every perturbed image before feeding it to the victim model. We adopt the default hyperparameters of these defenses used in [13]. When applying a defense, a higher FR achieved by an attack method indicates that the defense is less effective to the attack method, which implies its higher attack robustness.

Table 4 reports the results on ResNet-50 and ViT when $K = 5$ and $\xi = 6$. Comparing to the FR when no defence is applied, the FR of all the attack methods drops when a defense is applied. Nevertheless, CK2 still consistently achieves the best FR in all cases. This is because, although the effectiveness of every UAP is reduced by the defense, the set of images successfully attacked by different UAPs are still diversified, thus the key mechanism of cocktail attack still works to effectively improve the attack effectiveness.

Table 4: The FR of all the attack methods achieved on ResNet-50 and ViT in the presence of different defenses. We set $K = 5$ and $\xi = 6$. Bold number marks the highest FR in each row. Underlined number shows the runner up performance of the baseline methods.

$K = 5$, $\xi = 6$	Defenses	GAP	TDA	(UAT, CK1, CK2)	(DF, CK1, CK2)	(COS, CK1, CK2)	(NAG, CK1, CK2)
ResNet-50	No defence	84.2	84.4	(73.6, 77.6, **87.6**)	(72.7, 76.6, 87.0)	(66.6, 71.8, 80.5)	(68.5, 75.7, 79.2)
	JPEG	46.5	44.7	(36.3, 42.8, **47.7**)	(34.2, 40.5, 46.8)	(29.6, 34.4, 40.9)	(31.6, 37.6, 42.3)
	Crop and rescale	28.8	25.3	(22.2, 26.7, 30.4)	(22.4, 27.4, **31.7**)	(16.3, 22.9, 26.0)	(18.4, 23.8, 28.5)
	TV minimization	35.9	37.3	(29.2, 32.1, 38.6)	(28.5, 32.2, **39.2**)	(23.3, 26.4, 30.1)	(25.2, 28.8, 33.2)
	Bit depth reduction	45.7	45.3	(35.2, 40.4, 46.7)	(36.8, 42.3, **47.5**)	(27.6, 33.6, 38.2)	(30.7, 37.0, 42.4)
ViT	No defence	83.7	84.4	(76.5, 80.4, **88.9**)	(75.6, 79.3, 88.3)	(73.4, 78.1, 86.2)	(73.6, 80.3, 86.5)
	JPEG	49.6	50.2	(40.5, 45.0, **52.3**)	(38.1, 45.8, 50.2)	(37.6, 43.5, 48.0)	(34.0, 41.4, 47.1)
	Crop and rescale	32.3	31.8	(27.8, 29.5, **34.0**)	(25.2, 28.3, 33.2)	(25.0, 28.1, 32.9)	(24.4, 28.7, 31.3)
	TV minimization	40.8	41.6	(35.2, 39.7, **43.4**)	(33.3, 38.0, 43.0)	(34.6, 38.2, 42.1)	(32.9, 37.3, 40.0)
	Bit depth reduction	47.3	49.2	(39.7, 43.6, **51.4**)	(37.2, 44.0, 49.5)	(36.3, 41.3, 47.5)	(33.3, 39.8, 45.2)

C Training Efficiency

In this section, we discuss the training efficiency of the proposed cocktail attack framework. Different from the classic single-UAP methods that train a single UAP, our cocktail attack framework aims to train K UAPs and the selection neural network g_θ. Since g_θ is a lightweight SqueezeNet, the training of g_θ is not an efficiency bottleneck.

However, the value of K affects the efficiency of our training. This is because, as shown in our formulation in Eq. (2), for each training image we need to compute the value of the loss function $L_f(x_i, \delta_j)$ K times by adding each of the K perturbations to that image and then feeding the perturbed K images to the victim model. As a result, for the proposed cocktail attack methods, the number of images fed to the victim model in one training iteration is K times of the batch size. Compared to the batch size b used by the single-UAP methods, we set the batch size for cocktail attack methods to $\lceil \frac{b}{K} \rceil$ due to the limited GPU memory. In this way, the number of images used for forward-pass and back-propagation in one batch of training iteration of our methods is approximately the same as b, thus the training time of our methods for each batch is approximately the same as the single-UAP methods. However, the total number of batches per epoch for our methods is about K times of the corresponding single-UAP methods, which causes the training time of CK2 to be approximately K times of the corresponding single-UAP method when the same number of training epochs are used.

The training time of CK1 is about $2K$ times of the corresponding single-UAP method. Denote by E the training time of a single-UAP method, CK1 first trains K UAPs by independently running the single-UAP method K times, which costs $K \times E$ time. Then, CK1 takes another $K \times E$ time to solve the optimization problem in Eq. (2). Therefore, the overall training time of CK1 is about $2K \times E$, which is $2K$ times of the training time of the corresponding single-UAP method.

Table 5 reports the training time of different single-UAP methods and the corresponding CK1 and CK2 attack methods when attacking ResNet-50 with

Table 5: The training time (minutes) on ResNet-50 when $\xi = 6$ and $K \in \{1, 3, 5, 7, 9\}$.

K	(UAT, CK1, CK2)	(DF, CK1, CK2)	(COS, CK1, CK2)	(NAG, CK1, CK2)
1	(43, 95, 54)	(45, 98, 56)	(42, 94, 55)	(166, 221, 57)
3	(43, 268, 143)	(45, 277, 150)	(42, 261, 141)	(166, 305, 144)
5	(43, 425, 230)	(45, 435, 239)	(42, 430, 226)	(166, 391, 232)
7	(43, 603, 315)	(45, 615, 328)	(42, 598, 311)	(166, 469, 319)
9	(43, 774, 405)	(45, 794, 414)	(42, 757, 395)	(166, 556, 406)

$\xi = 6$ and $K \in \{1, 3, 5, 7, 9\}$. We can observe that the training time for CK1 and CK2 are approximately $2K$ and K times that of the corresponding single UAP method, such as UAT, DF and COS. This validates our analysis above.

We can also see in Tab. 5 that the training time of NAG-CK1 and NAG-CK2 does not align with our analysis. Because, the training time of NAG is dominated by the time cost to train a generator of UAP. However, NAG-CK1 does not independently train K generators; instead, it only trains a single generator to sample K UAPs. Therefore, the training time of NAG-CK1 is less than $2K$ times of NAG's training time. The training time of NAG-CK2 is also much less than K times of NAG's training time. This is because NAG costs a lot of time to train a generator, instead, NAG-CK2 only utilizes its fooling loss [29] as $L_f(x_i, \delta_j)$ to train g_θ and UAPs.

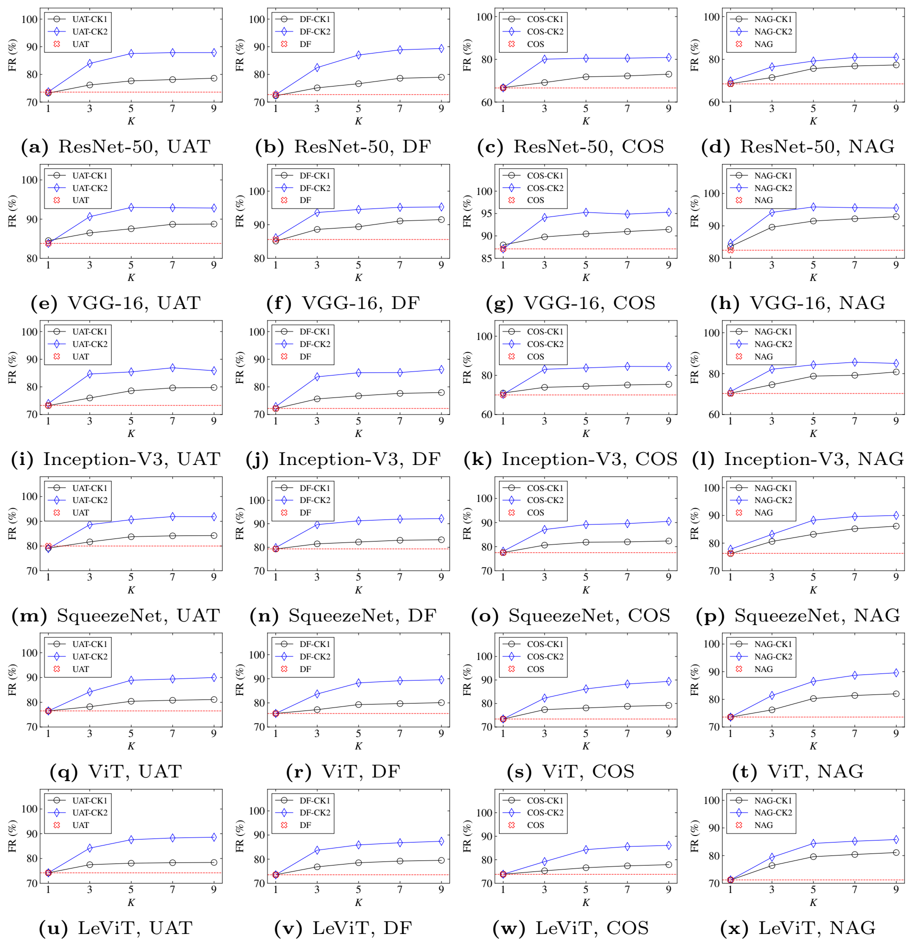
In conclusion, although a larger value of K makes the training of the cocktail attack framework less efficient, UAPs are often trained offline in the literature [25,36,41,45]. Therefore, the training time is not a big concern for attackers, who will benefit from the fast attack efficiency of the trained UAPs. Moreover, as shown by the experimental results in Fig. 3, using a small value of $K = 5$ is sufficient for CK2 to achieve significant attack effectiveness, indicating that the additional offline training time of CK2 is worth the effort.

D Hyperparameter Analysis

In this section, we investigate the effect of the number of perturbations K on the attack effectiveness of CK1 and CK2 when applied to each of the single-UAP methods. We adopt the settings in Sec. 4.2 to analyze the FR of each method when training and testing are conducted to attack the same victim model.

Figure 3 shows the results of attacking the six victim models with $\xi = 6$ and $K \in \{1, 3, 5, 7, 9\}$. We only report the FR of the single-UAP methods when $K = 1$ because each of them only uses a single UAP to attack all target images. As shown in Fig. 3, each single-UAP method and its corresponding cocktail attack methods achieve similar FR when $K = 1$. This is because, when $K = 1$, the cocktail attack launched by CK1 and CK2 degenerates to the same single-UAP attack as the corresponding single-UAP method.

We can also see that the FR of CK1 and CK2 improve significantly when K increases. The reason is that each newly added UAP will successfully attack more images that cannot be successfully attacked by the previous set of UAPs.



(a) ResNet-50, UAT **(b)** ResNet-50, DF **(c)** ResNet-50, COS **(d)** ResNet-50, NAG

(e) VGG-16, UAT **(f)** VGG-16, DF **(g)** VGG-16, COS **(h)** VGG-16, NAG

(i) Inception-V3, UAT **(j)** Inception-V3, DF **(k)** Inception-V3, COS **(l)** Inception-V3, NAG

(m) SqueezeNet, UAT **(n)** SqueezeNet, DF **(o)** SqueezeNet, COS **(p)** SqueezeNet, NAG

(q) ViT, UAT **(r)** ViT, DF **(s)** ViT, COS **(t)** ViT, NAG

(u) LeViT, UAT **(v)** LeViT, DF **(w)** LeViT, COS **(x)** LeViT, NAG

Fig. 3: The FR of the single-UAP methods and the corresponding cocktail attack methods when attacking the six victim models. We use $\xi = 6$ for all the experiments. We report the FR of CK1 and CK2 at $K \in \{1, 3, 5, 7, 9\}$. The dashed red line is an auxiliary line to show the FR of the single-UAP method at $K = 1$.

However, when $K \geq 5$, further increasing the value of K does not significantly improve the FR of CK1 and CK2. Since the set of successfully attacked images is already large when $K = 5$, the number of new images successfully attacked by adding more UAPs diminishes with the increase of K. In summary, we can conclude from Fig. 3 that the proposed cocktail attack framework can significantly improve the attack effectiveness of a single-UAP method by using only a small number of diversified UAPs.

E Proof of NP-hardness

Theorem 1. *The MIP problem in Eq. (1) is NP-hard.*

Proof. We prove the theorem by showing that an instance of the K-means clustering problem [23] can be reduced to an instance of the MIP problem in polynomial time.

An instance of the K-means clustering problem can be formulated as

$$\min_{\mathcal{S},\mu_1,\ldots,\mu_k} \quad \sum_{j=1}^{K}\sum_{x_i\in S_j} \|x_i-\mu_j\|^2, \tag{3}$$

where $\mathcal{S}=\{S_1,\ldots,S_K\}$ is a set of K non-overlapping clusters of a set of N images, denoted by $X=\{x_i\}_{i=1}^N$, and μ_j is the mean of the images in S_j.

Next, we construct an instance of the MIP problem in polynomial time and prove that its solution is exactly the same as the solution to the instance of the K-means problem. The key idea is to construct a loss function

$$L_f(x_i,\delta_j)=\|x_i-\delta_j\|^2, \tag{4}$$

which maps the tuple (x_i,δ_j) to the squared Euclidean distance between x_i and δ_j. We construct the loss function $L_f(x_i,\delta_j)$ in polynomial time as follows.

First, we compute the squared difference matrix

$$H=(x_i-\delta_j)\circ(x_i-\delta_j), \tag{5}$$

where the operator $\circ$ is the Hadamard product between two matrices. Second, we construct a DNN f that takes H as the input and outputs the sum of all the entries in H. This is done by first passing H through a convolutional layer that maps H to itself, then using a fully connected layer to compute the sum of the entries in H. By plugging $L_f(x_i,\delta_j)=\|x_i-\delta_j\|^2$ into Eq. (1), we can construct an instance of the MIP problem as

$$\begin{aligned}\min_{A,P} \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} A_{i,j}\|x_i-\delta_j\|^2,\\ \text{s.t.} \quad & A\in\{0,1\}^{N\times K},\\ & \sum_{j=1}^{K} A_{i,j}=1, \forall i\in\{1,\ldots,N\},\\ & \|\delta_j\|_\infty\le\xi, \forall j\in\{1,\ldots,K\},\end{aligned} \tag{6}$$

which can be rewritten as

$$\begin{aligned}\min_{\mathcal{S},\delta_1,\ldots,\delta_k} \quad & \sum_{j=1}^{K}\sum_{x_i\in S_j} \|x_i-\delta_j\|^2,\\ \text{s.t.} \quad & \|\delta_j\|_\infty\le\xi, \forall j\in\{1,\ldots,K\},\end{aligned} \tag{7}$$

because $A_{i,j} = 1$ means $x_i \in S_j$ and $A_{i,j} = 0$ means $x_i \notin S_j$.

By substituting δ_j with μ_j and setting $\xi = +\infty$, Eq. (7) will be the same as Eq. (3). In this case, the solution to the instance of the MIP problem is exactly the same as the solution to the instance of the K-means problem. In sum, the K-means problem can be reduced in polynomial time to the MIP problem. Since the K-means clustering problem is NP-hard [40], the MIP problem is also NP-hard.