When George Clooney Is Not George Clooney: Using GenAttack to Deceive Amazon's and Naver's Celebrity Recognition APIs

Keeyoung Kim, Simon S. Woo

To cite this version:

Keeyoung Kim, Simon S. Woo. When George Clooney Is Not George Clooney: Using GenAttack to Deceive Amazon's and Naver's Celebrity Recognition APIs. 33th IFIP International Conference on ICT Systems Security and Privacy Protection (SEC), Sep 2018, Poznan, Poland. pp.355-369, 10.1007/978-3-319-99828-2_25 . hal-02023746

HAL Id: hal-02023746
https://inria.hal.science/hal-02023746

Submitted on 21 Feb 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

When George Clooney is not George Clooney: Using *GenAttack* to Deceive Amazon's and Naver's Celebrity Recognition APIs

Keeyoung Kim[1,2,3] and Simon S. Woo[1,2] ⋆

[1] The State University of New York, Korea (SUNY-Korea), Incheon, S. Korea
[2] Stony Brook University, Stony Brook, NY, USA
[3] Artificial Intelligence Research Institute (AIRI), Seongnam, S. Korea
`{kykim,simon.woo}@sunykorea.ac.kr`

Abstract. In recent years, significant advancements have been made in detecting and recognizing contents of images using Deep Neural Networks (DNNs). As a result, many companies offer image recognition APIs for use in diverse applications. However, image classification algorithms trained with DNNs can misclassify adversarial examples, posing a significant threat to critical applications. In this work, we present a novel way to generate adversarial example images using an evolutionary genetic algorithm (GA). Our algorithm builds adversarial images by iteratively adding noise to the original images. Unlike DNN based adversarial example generations by other researchers, our approach does not require GPU resources and access to the target DNNs' parameters. We design, *GenAttack*, a simple yet powerful attack algorithm to create adversarial examples using complex celebrity images and evaluate those with real-world celebrity recognition APIs from Amazon and Naver. With our attack, we successfully deceive Amazon's and Naver's APIs with a success probability of 86.6% and 100%, respectively. Our work demonstrates the practicability of generating adversarial examples and successfully fooling the state-of-the-art commercial image recognition systems.

Keywords: Adversarial Example, Black-box attack, Genetic Algorithm

1 Introduction

Deep learning algorithms have been revolutionary in improving the performance of a wide range of applications, including computer vision, speech processing, and natural language processing. In particular, Convolutional Neural Networks (CNNs) have been extremely successful in detecting and recognizing the content of images [22, 20, 8]. Due to the success of deep learning, many companies including Amazon [1] and Naver [2] have unveiled image recognition and analysis APIs to be used for various applications. However, as Szegedy et al. [23] and Goodfellow et al. [7] showed that an imperceptible small perturbation to an input image can arbitrarily change the prediction of a deep learning-based classifier. These examples are referred to as *adversarial examples*, which optimize perturbations to maximize prediction errors. Moreover, Goodfellow et al. [7] showed

⋆ Corresponding Author

that these adversarial examples are not difficult to generate, and are robust and generalizable. Therefore, the robustness and stability of DNNs when facing adversarial examples have recently drawn the attention of many researchers [23, 25, 6, 7]. In particular, adversarial examples can be a serious threat to image processing applications such as airport security systems, self-driving cars, and user identification for financial transaction systems.

In this work, unlike other DNN-based attack methods [6], we propose an alternative approach to generate adversarial images using an evolutionary genetic algorithm (GA) to deceive the DNN based state-of-the-art image recognition APIs. We perform *GenAttack*, a simple yet powerful practical black-box attack using our GA to fool commercial APIs, and show that those commercial APIs are easily fooled with a high probability of success. Our contributions are summarized below:

1. We propose *GenAttack*, an attack algorithm using GA to generate adversarial images. We test GenAttack against larger, and more complex realistic images ranging from 200×300 to 2,100×2,800 pixels, unlike other research that utilizes the small image sizes. *GenAttack* adopts the heuristic optimization method so that it can easily deal with large number of pixels in parallel.
2. We evaluate our attacks with state-of-the-art commercial celebrity detection APIs from Amazon [1] and Naver [2] as representative test cases. Our approach effectively creates adversarial images and deceives Amazon and Naver APIs with **86.6%** and **100%** success rate.
3. We also show *transfer learning* of an adversarial image. We demonstrate that an adversarial example successfully fools one classifier (e.g. Naver) can be used to fool another classifier (e.g. Amazon), which could not be deceived originally. Therefore, transfer learning can be maliciously used to fool a classifier more effectively.

This paper is organized as follows. We discuss related work of adversarial examples in Section 2. We explain our GA and *GenAttack* in Section 3, and describe our experiment in Section 4. Section 5 presents the results of our evaluation of GenAttack. In Section 6, additional experiment for transfer learning is presented. We provide the possible defense mechanism, discussion, and limitations in Section 7. Finally, Section 8 offers conclusions.

2 Related Work

Adversarial examples [23] are examples, which machine learning models misclassify, where those examples are only slightly different from correctly classified examples. Applying an imperceptible perturbation to a test image can produce an adversarial example. Adversarial examples were first discussed and used against conventional machine learning algorithms by Barreno et al. [3] to evade handcrafted features. In particular, Biggio et al. [4] created adversarial examples for a linear classifier, SVM, and neural network using a gradient-based method. Szegedy et al. [23] first introduced the adversarial examples for the deep neural networks by adding small perturbations on the input images. They used the white-box L-BFGS method to generate adversarial examples using MNIST, ImageNet, AlexNet, and QuocNet with high probability. Since L-BFGS uses an expensive linear search, Fast Gradient Sign Method (FGSM) was proposed by Goodfellow et al. [7], which can be computed using back-propagation. RAND-FGSM [24] is proposed to add randomness during the gradient update. Papernot et al. [18] presented an efficient saliency adversarial map (JSMA). Their approach can find the input features that make the most significant change to the output so that a small portion of features can fool DNNs. DeepFool was proposed by Moosavi-Dezfooli et al. [15], which determines the closest distance from original input to the decision boundary, and performs an iterative attack to approximate the perturbation. Moreover, Carlini and Wager [6] showed that the back propagation method with DNNs can generate adversarial examples more effectively, and demonstrated that existing defense methods are not effective. Papernot et al. [17] introduced the practical black-box attack approach. Their approach consists of training a local model to substitute for the target DNN, using inputs synthetically generated by an adversary and labeled by the target DNN. But, their evaluation results are based on the trivial MNIST dataset. Nguyen et al. [16] implemented the evolutionary algorithm to generate images that humans cannot recognize but DNNs can. In addition, Vidnerov and Neruda [25] showed that the evolutionary method can generate adversarial examples from random noise. But they only tested classifiers to detect the trivial 0-9 digit images. Hosseini et al. [10] showed that Google's Cloud Vision API can be deceived by images added with random noise. Our approach is more sophisticated than the approach by Hosseini et al. [10] which simply adds uniform noise. In our approach, the GA locally optimizes the noise level at each iteration. Our advantage is that we generate adversarial images more effectively. We provide the comparison between our and random noise distribution in Section 5. Network distillation was proposed by Papernot et al. [19] to reduce the size of DNNs by extracting knowledge from DNNs to improve the robustness by 0.5% to 5% on MNIST and CIFAR10 dataset, respectively. Goodfellow et al. [7] and Huang et al. [11] introduce the *adversarial training*, an approach to include adversarial examples in the training stage. They incorporated adversarial examples in training sets, and showed it improved the robustness. Tramer et al. [24] proposed Ensemble Adversarial Training method to train a defense model with adversarial examples generated from multiple sources. However, they found that it is difficult to anticipate specific adversarial examples and include those during the training stage. Madry et al. [14] proved that adversarial training with large network capacity can defend the first-order adversary. Also, adversarial detection [13, 21, 5] have been proposed by others to detect adversarial examples during testing.

3 Design of Our Approach

First, we define the adversarial example problem and the objective of our approach. Next, we present the details of the GA to generate adversarial examples, and *GenAttack* to deceive commercial APIs.

3.1 Adversarial Examples for Image Classification

Szegedy et al. [23] shows the existence of *targeted adversarial examples* as follows: given a valid input image $\mathbb{I}$, and a target $t \neq C^*(\mathbb{I})$, it is possible to find a similar input $\mathbb{I}'$ such that $C^*(\mathbb{I}') = t$, yet $\mathbb{I}$ and $\mathbb{I}'$ are close according to some distance metric. In *Untargeted adversarial examples*, attacks only search for an input $\mathbb{I}'$ so that $C(\mathbb{I}) \neq C^*(\mathbb{I}')$ and $\mathbb{I}$ and $\mathbb{I}'$ are close. Then, finding adversarial examples can formulated as follows similar to [23, 26]:

$$\begin{aligned} &\min_{\mathbb{I}'} ||\mathbb{I}' - \mathbb{I}|| \\ &s.t. \quad C(\mathbb{I}) \neq C^*(\mathbb{I}'), \end{aligned} \tag{1}$$

where $||\cdot||$ is the distance between two samples, and C is a trained deep learning image classifier. The goal is to find the input $\mathbb{I}'$ minimizes the distance between $\mathbb{I}$ with small perturbations. We aim to find adversarial examples for an untargeted case, where we find an image, $\mathbb{I}'$ that C misclassifies from $\mathbb{I}$ to $\mathbb{I}'$.

3.2 Creating Adversarial Examples using Genetic Algorithm (GA)

In order to perform a black-box attack, we develop GA to effectively generate adversarial images against commercial APIs without access to any of their DNN model parameters, and do not require any GPU resources. The goal of our GA is to inject a small amount of optimum noise to an original image so that commercial APIs misclassify the original image, while humans can still easily recognize the original celebrity as shown in Fig. 1. We formulate our GA as follows:

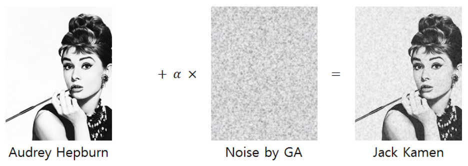


Fig. 1: Amazon API misclassifies the noise-added Audrey Hepburn image $\mathbb{I}'$ to Jack Kamen, while it correctly classifies the original image $\mathbb{I}$ to Audrey Hepburn

Population and Individuals: A population is a set of individuals, and they are defined as uniform noise matrices, where their size is the same as the original input celebrity image. To produce the noise-added adversarial images from the noise matrices, we use the modified method based on Carlini and Wagner [6], as follows:

$$\mathbb{X} = \tanh(\tanh^{-1}(\frac{\mathbb{I}}{\mathbb{I}_{max}} - 0.5) + \alpha \times \mathbb{N}) \tag{2}$$

$$\mathbb{I}' = \frac{(\mathbb{X} - \min(\mathbb{X}))}{(\max(\mathbb{X}) - \min(\mathbb{X}))} \times \mathbb{I}_{max} \tag{3}$$

In Eq. 2, we transform an original (target) image $\mathbb{I}$ to $\tanh^{-1}$ space, and map it from -0.5 to 0.5 range by dividing by $\mathbb{I}_{max}$ and subtracting 0.5, where $\mathbb{I}_{max}$ is the maximum RGB pixel value. Next, we add $\mathbb{I}$ with a noise matrix $\mathbb{N}$ multiplied by the coefficient α. Then, we re-transform the noise added image back to the original space to obtain the adversarial example $\mathbb{I}'$ in Eq. 3. As shown in Fig. 1, α adjusts a noise level in generating an adversarial image, and α is searched from multiplying by 2 or subtracting 0.05 in [0.0, 0.9] interval. Generally, a higher α increases the success rate of our attack, however, it will produce a very noisy image. Hence, minimize the noise amount, α, using the following fitness function.

Fitness function: We use the following L_1 loss as a distance measure between the original image $\mathbb{I}$, and the adversarial image $\mathbb{I}'$:

$$L_1 \ = \frac{1}{n} \sum |(\mathbb{I} - \mathbb{I}')|, \tag{4}$$

where n is the number of pixels in the image $\mathbb{I}$. Then, we define the fitness function f as follows in Eq. 5:

$$f = P_o - P_d + \gamma \times L_1, \tag{5}$$

where P_o is the predicted probability for the original label and P_d is the predicted probability for any other wrong labels. We can obtain either one of P_o or P_d and set the other to zero, because the commercial APIs only return the highest probability of one of P_o or P_d. Next, we formulate our GA as a minimization problem with the value of the fitness function in Eq. 5. to produce the best individual which has high P_d, and the low P_o, and L_1 values. In Eq. 5, γ is another coefficient to balance the noise amount to deceive APIs by guiding a GA to find adversarial images with the least amount of noise, where γ can be chosen from 0.01 to 0.1 in this work. Also, we automatically choose γ, which is inversely proportional to α, because P_o and P_d always have the values between 0 to 1. In a default setting, we run 5 epochs to generate an adversarial example for a target image after fixing α, where α requires from several ten to three hundred steps. The number of steps in one epoch – children generated by crossover and then accepted to inherit to the next generation – is the same to the number of populations. The number of API calls per each step in the algorithm will be affected by the chance how much mutation will be called.

Selection: We implement a tournament selection, where we set the tournament size to four. Then, two of four individuals in one tournament will be selected. In our design, the more fit has 80% chance to win, and the less fit has 20% chance to maintain a good variety in the population and explore wider search areas to find a global optimum. After selection, two chosen parents move to the next crossover stage.

Crossover and Inheritance: Crossover permutes two selected parents. We design a simple crossover for 2D matrix as shown in Fig. 2. First, we obtain a random point $(x,\ y)$ in the noise matrices of two selected individuals. We use this point as an origin point to start. Next, we throw a tetrahedron die. If we get N, between 1 to 4, the quadrant N of the noise matrices will be exchanged between two individuals.

Then, the newly generated children are chosen to inherit to the next generation, if they have a better fitness than their parents. To conserve the best fit individuals and not to lose them, we also added the following inheritance heuristics: if the best individual in the current generation is better than any individuals in the next generation, we copy the best individual in the current generation to the next generation.

Mutation: Mutation aims to reduce the noise level of adversarial images. We design two mutation methods based on the class labels of the newly produced individuals. The first mutation method is used, when the noise added image is still classified into the original class. Then, we add a small amount of random noise to individuals to produce more variations. We use the second mutation method, when a noise injected image is classified into another class. In this case, we try to reduce the noise slightly by using the following local optimization technique: We randomly choose 2% of the pixels in the noise matrix, and reduce their magnitude by 30%. If successful, we repeat the same process for up to 5 more times. In this local optimization step, we only accept mutated individuals with improved fitness values.

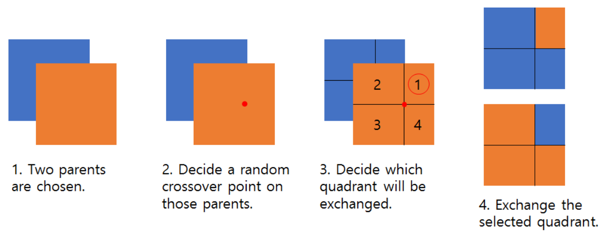

1. Two parents are chosen.
2. Decide a random crossover point on those parents.
2
1
3
4
3. Decide which quadrant will be exchanged.
4. Exchange the selected quadrant.

Fig. 2: Description of the Crossover Process

3.3 *Genetic Algorithm-based Attack (GenAttack)*

We propose the GA-based attack, *GenAttack*, against commercial APIs, and present the details of our attack procedures. First, we test the commercial API with an original image, and check whether the API correctly recognizes the celebrity image from the returned initial output label, I.Label, and its confidence value. If it is correctly recognized, it becomes our target celebrity image to create an adversarial example. If not, we discard this image, since the API is wrong in the first place. Next, we initiate *GenAttack* and start querying each commercial API with the noise-added image. If the returned result produces an incorrect output label (i.e., some other celebrity), that means our attack is successful, and we successfully create an adversarial image. We label this output class as an adversarial label, A.Label. If it consistently returns the correct I.Label, we slightly increase and adjust α, and compute the fitness function, searching for the optimum noise combinations according to our GA. We iteratively repeat this process for several epochs. until we successfully force the API to produce an incorrect output (A.Label). Finally, if we can deceive an API, so that the API returns a different name from the I.Label, we declare the attack is successful. If we cannot deceive, or the API returns 'Unrecognized' (UNKR), the attack is not successful, and we fail to create an adversarial image. Our attack criteria is much stronger than prior research [9, 10], which includes UNKR as a success.

4 Experiment

The goal of our experiment is to evaluate generated adversarial examples, and further test the robustness of commercial APIs using GenAttack. We used Amazon Rekognizer [1], and Naver Clover Service [2] to provide a side-by-side attack success, and robustness comparison between these providers. In particular, we used the celebrity recognition APIs, which are offered from both providers, and celebrity images are relatively easy to find. Also, they are complex and realistic. Although Cloud Vision API by Google also provides the face image analysis information, and returns the top 20 relevant entities from the Internet, their returned labels are based on web search results, not images themselves. Hence, a side-by-side comparison to Amazon and Naver is difficult, therefore, we do not evaluate Google's API in this paper.

4.1 Dataset

We chose 119 famous celebrities (72 men and 47 women) as a dataset. Although we try to select celebrities that are both popular in America and Asia, we hypothesize that the Naver API based in Asia would be more optimized for Asian celebrities over American and European celebrities. Hence, we include several Asian celebrities to test, even though they may not be so well known in America or Europe. We use the practical image sizes ranging from 200×300 to $2{,}100\times2{,}800$ pixels. These are much larger than the small benchmark datasets such as MNIST (28×28), CIFAR-10 (32×32), and ImageNet (227×227) which have been used in prior research[6, 16]. Some sample celebrity images and names are shown in Fig. 3 and Table 1.

4.2 Experimental Setup

We run 5 epochs to get an adversarial example for all 119 target images, starting with α= 0.1. Then we automatically adjust α from 0.05 to 0.9 based on the attack success and confidence value returned from the API. We run 5 more epochs to generate an adversarial example for a target image after obtaining α from the GA. If we consecutively fail to produce an adversarial image in the next 10 steps, we increase α and repeat the process again. If we find an adversarial images in 10 consecutive steps, we decrease α to reduce the noise.

5 Results

In this section, we report the attack success rate and analyze generated noise in adversarial examples from GenAttack.

5.1 Attack Success Rate

Table 1. summarizes our attack results for several celebrity images. Due to space limitations, we only present celebrities whose original image was correctly recognized by both APIs. In Table 1, the fist column is the correct celebrity name for each image followed by its initial I.Label and I.Pr., where those indicate the

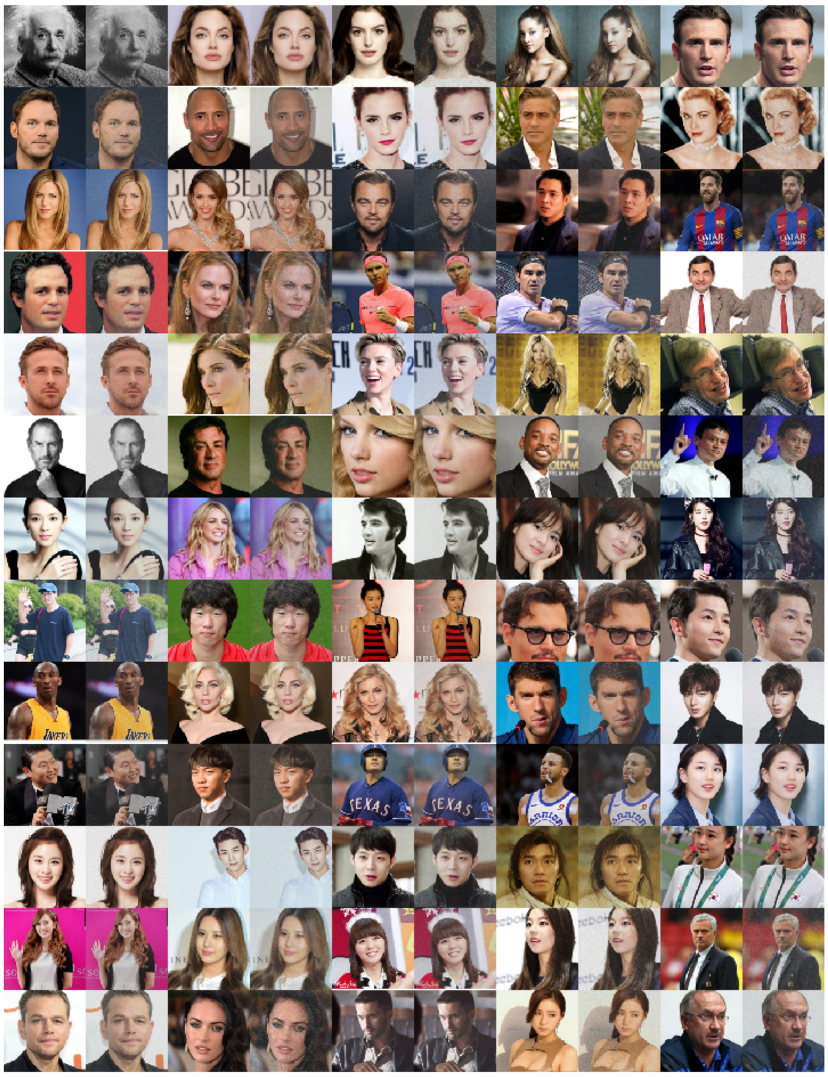


Fig. 3: Original vs. Generated Adversarial Images for 65 celebrities

original input label and its confidence probability returned by each API. And A.Label and A.Pr. are the output label and its confidence probability for adversarial images we generate with our GenAttack. 'UNKR' means that the original image is successfully recognized, while the noise-added image is unrecognized.

Table 1: Examples of attack result against Amazon and Naver APIs with each celebrity

Celeb	Amazon				Naver			
	I.Label	I.Pr.	A.Label	A.Pr.	I.Label	I.Pr.	A.Label	A.Pr.
Sohee	Sohee	1.00	Park Soo-jin	0.60	Ahn Sohee	1.00	Ahn Sohee	1.00
Alicia Keys	Alicia Keys	1.00	Cindy Bruna	0.79	Alicia	1.00	Alicia	0.28
Kim Yuna	Kim Yuna	0.99	UNKR	0.00	Yuna Kim	1.00	Choi Ja-hye	0.24
Kim Soo-hyun	Kim Soo-hyun	0.97	Kim Kiri	0.84	Kim Soo-hyun	1.00	Choi Yonggeun	0.74
Kate Mara	Kate Mara	1.00	UNKR	0.00	Kate Mara	1.00	G. Atkinson	0.59
Megan Fox	Megan Fox	1.00	Maimie McCoy	0.61	Megan Fox	1.00	G. Atkinson	1.00
Jun Ji-hyun	Jun Ji-hyun	1.00	Yang Lan	0.70	Jun Ji-hyun	0.24	Gong Hyeon-ju	0.25
Song Hye-kyo	Song Hye-kyo	0.99	Juri Ueno	0.77	Song Hye-kyo	0.87	Hirano Yuta	0.45
Park Ji-sung	Park Ji-sung	0.74	Park Chuyoung	0.86	Park Ji-sung	1.00	Hwang In-hoo	0.27
Im Yoon-ah	Im Yoon-ah	1.00	UNKR	0.00	Im Yoon-ah	1.00	Im Seong-eon	0.68
ShinSoo Choo	ShinSoo Choo	0.99	Y. Tsutsugo	0.59	ShinSoo Choo	0.58	J. Hyeonseok	0.21
Song Joong-ki	Song Joong-ki	1.00	Steven Ma	0.84	Song Joong-ki	1.00	Ji Jin-hee	0.16
Seohyun	Seohyun	1.00	J-Min	0.89	Seohyun	1.00	Jo Yoon-hee	0.50
Eric Mun	Eric Mun	1.00	Tao Lin	0.99	Eric Mun	1.00	Joo Sang-wook	0.34
Lee Min-ho	Lee Min-ho	1.00	Lee Joon-gi	0.72	Lee Min-ho	1.00	K. Min-hyeok	0.99
Hyun-jin Ryu	Hyun-jin Ryu	1.00	UNKR	0.00	Hyun-jin Ryu	1.00	Kim Dong-ju	0.46
Yoo Jae Suk	Yoo Jae Suk	0.92	Marshall Allen	0.92	Yoo Jae-suk	1.00	Kim Dong-yeon	1.00
Lee Seung-gi	Lee Seung-gi	1.00	Keisuke Koide	0.55	Lee Seung-gi	0.76	Kim Min-sang	0.73
Ok Taecyeon	Ok Taecyeon	0.95	Huang Jingyu	0.86	Ok Taecyeon	1.00	Kim Min-soo	0.44
Son Yeon-jae	Son Yeon-jae	1.00	Park So-youn	0.88	Son Yeon-jae	0.96	Kim Tae-ri	0.67
Kang Ho-dong	Kang Ho-dong	1.00	Tommy Chang	0.80	Kang Ho-dong	1.00	Kim Yeongseok	0.49
Kwon Yuri	Kwon Yuri	0.99	UNKR	0.00	Yuri	1.00	Lee Eun-jeong	0.21
Lionel Messi	Lionel Messi	0.98	Paul Anderson	0.72	Messi	1.00	Lee Il-woong	0.33
IU	IU	0.99	H. Jungeum	0.92	IU	0.97	Lee Ji-eun	0.76
Lee Byung-hun	Lee Byung-hun	1.00	Kim Byung-man	0.89	Lee Byung-hun	1.00	Lee Sang-woo	0.27
Matt Damon	Matt Damon	1.00	F. Marques	0.84	Matt Damon	1.00	Matt Damon	0.59
Mark Wahlberg	Mark Wahlberg	1.00	Shawn Hatosy	0.75	Mark Wahlberg	1.00	Oh Ji-myeong	0.32
Uli Stielike	Uli Stielike	1.00	D. Pleasence	0.99	Stielike	1.00	Oh Ji-myeong	0.24
Lily Collins	Lily Collins	1.00	S. Carpenter	0.84	Lily Collins	1.00	Oh Seo-woon	0.45
Tom Cruise	Tom Cruise	1.00	B. Daugherty	0.89	Tom Cruise	1.00	Olivier	0.61
Jessica Jung	Jessica Jung	1.00	Shin Bora	0.89	Jessica Jung	1.00	Park High	1.00
Lee Chungyong	Lee Chungyong	1.00	DongHyun Kim	0.69	Lee Chungyong	0.95	Park Se-jun	0.75
Tang Wei	Tang Wei	0.99	Soyou	0.61	Tang Wei	0.63	Ryeowon Jung	0.28
Shin Se-kyung	Shin Se-kyung	1.00	Akiko Suwanai	0.60	Shin Se-kyung	0.74	Sa Hee	0.23
Kim Tae-hee	Kim Tae-hee	0.96	Kang So-ra	0.73	Kim Tae-hee	1.00	Seo Yeong-hee	0.74
Taeyeon	Taeyeon	0.67	UNKR	0.00	Kim Taeyeon	1.00	Shihono Ryo	0.51
Sooyoung	Sooyoung	0.99	UNKR	0.00	Sooyoung	1.00	Sol Ji	0.83
Park Tae-hwan	Park Tae-hwan	1.00	UNKR	0.00	Park Tae-hwan	1.00	S. Changhwan	0.74
Fedor Emelianenko	Danny Wuerffel	0.60	UNKR	0.00	Fedor	1.00	Song Jae-ho	0.35
José Mourinho	José Mourinho	0.99	Nicolás Lúcar	0.78	Mourinho	1.00	Song Yongtae	0.33
Claudia Kim	Claudia Kim	1.00	Krystal Jung	0.92	Claudia Kim	1.00	Tae-im Lee	0.67
Olivier Martinez	Olivier Martinez	0.92	Álvaro Medrán	0.79	Olivier	0.55	Yu Oh-seong	0.45
Bae Suzy	Bae Suzy	0.99	Chae Soo-bin	0.91	Suzy	1.00	Yuu	0.33

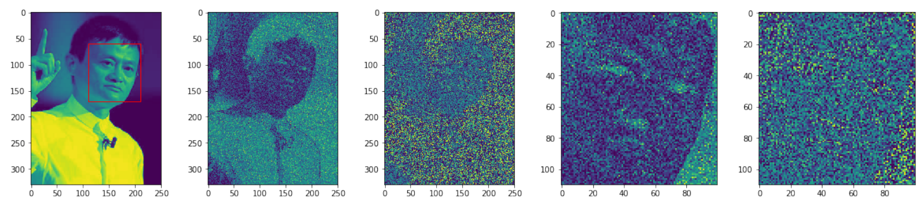
Amazon API correctly recognizes 112 images from the 119 input images. Our algorithm attacked those 112 images, and achieved the overall **86.61%** success rate, successfully creating 97 adversarial examples. We find that *GenAttack* effectively adds and improves noise from a predicted label with a low initial confidence value returned for its initial adversarial example generation attempt. From Table 1, we can observe that *GenAttack* guides noise to find a path from one output celebrity class to another celebrity class with fairly high confidence values (A.Pr.) in many cases shown in the 5th column in Table 1.

On the other hand, the Naver API correctly recognizes only 45 out of the 119 original images, misclassifying many original celebrity images from America and Europe. Hence, we validate Naver is more localized to Asian faces. Among those correctly recognized 49 images, *GenAttack* successfully creates the adversarial images for all 49 images, yielding **100.00%** success rate. Naver seems to generate different output labels for many Asian celebrities even with a small amount of added perturbations, and Naver is much easier to fool. However, their A.Pr. are generally lower than Amazon, which means Naver outputs the new label with smaller confidence value. With the Naver API, we observed that Tom Cruise was the most difficult one to find an adversarial example for. We hypothesize that Naver might not have many faces that are similar to Tom Cruises or have faces that are clearly distinctive. Therefore, Naver locks on to the features of

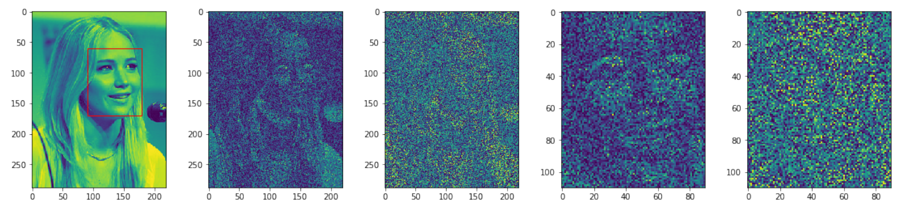
Tom Cruise and we think *GenAttack* could not easy to find other similar classes. In Fig. 3, we present 65 original celebrity images (left) and adversarial images (right) generated from *GenAttack* side-by-side for a comparison. As we can examine, the generated adversarial images are very close to the original images, and humans can trivially recognize the generated adversarial examples.

5.2 Noise and Image Analysis

We carefully analyze the noise patterns of the adversarial images, where we add random noise in tanh space. Also, we compare our noise with uniform noise in tanh space to characterize differences.



(a) Original image: Jack Ma (b) Noise by GA (c) Random noise (d) Zoom in on a face area of (b) (e) Zoom in on a face area of (c)



(f) Original image: Jennifer Lawrence (g) Noise by GA (h) Random noise (i) Zoom in on a face area of (g) (j) Zoom in on a face area of (h)

Fig. 4: Comparison of noise distribution (*GenAttack* vs. Uniform Noise)

Figures 4.(a) and (f) are the adversarial examples we produced for Jack Ma and Jennifer Lawrence. In Fig. 4, brighter yellow represents a higher pixel value, and dark blue indicates a lower pixel value. Figures 4.(b) and (g) only show the generated L_2 noise, and Figs. (c) and (h) are the uniformly generated noise for the same images. As we compare these two sets of images, we can clearly observe that our GA tends to better capture face features of the input and injects noise, while random noise spreads over all pixels. In order to analyze the differences more clearly, we zoom in the face areas. As we can observe from Figs. 4.(d) and (e), and (i) and (j), we find noise generated from the GA more closely changes the face features so that the CNNs based classifier can more easily make a mistake and steer towards another celebrity. On the other hand, random noise is distributed uniformly over all pixels. Hence, we clearly observe the differing noise distribution, and our generated noise appears to better learn the face features with the GA to optimize noise to increase a classification error.

Noise Filtering Defense and Generated image sizes: Generally, pre-filtering can be an effective defense mechanism as proven in other research [9]. However, it is not in our case. We applied both Gaussian (linear) and Median (non-linear) filters to remove added noise from GA, where these filters have been a successful defense shown from other research [9]. In our case, noise filtering cannot prevent from generating adversarial examples for both Amazon and Naver, but generated adversarial images need slightly more noise than the non-filtered case. Also, we find that our approach effectively generates adversarial examples for any size of input celebrity images in our dataset, ranging from 200×300 to 2,100×2,800 in pixels. Hence, we demonstrate that our GA can generate almost size-invariant adversarial images without loss of any performance.

6 Transfer Learning for Attacks

We performed the *transfer learning* capability of our proposed method. If our algorithm can deceive one classifier, we hypothesize we can deceive another API. Hence, attackers can use this transfer learning for an attack, where adversarial features (noise matrices) learned from one DNNs (e.g. Naver) can be used to create an adversarial image for another classifier (e.g. Amazon), and vice versa. Among all 119 celebrity images, we obtained ten adversarial samples that successfully fool only one of the APIs, as shown in Table 2. In our attack, we query both Naver and Amazon APIs simultaneously, and calculate fitness as follows by extending the fitness function for the single API in Eq. 5:

$$f = P_o^{Amazon} - P_d^{Amazon} + P_o^{Naver} - P_d^{Naver} + \gamma \times L_1, \tag{6}$$

where P_o^{Amazon} and P_o^{Naver} are the predicted probability for original label from Amazon and Naver, and P_d^{Amazon} are P_d^{Naver} are the predicted probability for other label produced from Amazon and Naver similar to Eq. 5. When optimizing α in Eq. 2, we only consider the adversarial image generation success rate of the target API, which was originally unsuccessful. For example, if we want to find adversarial examples for the Amazon API with the help from the Naver API, we optimize α based on the success rate of the Amazon API.

We performed the transfer learning attack experiment for all available 10 test cases. Overall, 7 out of 10 transfer learning attack were successful, improving the most of UNKR (originally failed attacks by the single API) to other celebrities. Among those, where Amazon API fails to recognize 8 of celebrities initially, our algorithm successfully fools Amazon API with the help from Naver API.

As shown in Table 2, six of (Before) 'UNKR' were successfully classified to (After) other celebrities. However, creating adversarial images for Kate Maria, and Kim Yuna were unsuccessful, even using Naver API. On the other two cases, where our algorithm successfully attacked Amazon API but not Naver API initially, we performed the transfer learning attack on Naver with the help from Amazon. Naver was originally correct for "Sohee" but Amazon led Naver to misclassify the correct label "Sohee" to "Park Soo-jin" (the same adversarial label in Amazon, as shown in blue). Hence, this shows that *targeted attack* is possible via transfer learning, making other celebrity to a specific victim label (e.g. Park Soo-jin). Also, Naver successfully launched the targeted attack for "Sooyoung" to be "Solji" (shown in blue) in the same way. This demonstrates that the same fake label can be exactly transferred from one classifier to another classifier. Hence, noise generated from our algorithm is transferable between classifiers for generating adversarial examples. Hence, attackers can practically leverage transfer learning to improve his attacks against DNNs.

Table 2: Transfer Learning Attack, where one API assists in deceiving another API, which was originally unsuccessful

Initial Correct Label	**Succ. Naver Adversarial Label**	**(Before) $\Longrightarrow$ (After): Fooling Amazon with Naver**
Park Tae-hwan	Song Chang-hwan	UNKR $\Longrightarrow$ **Julio Cesar Ceodillo**
Hyun-jin Ryu	Kim Dong-ju	UNKR $\Longrightarrow$ **Niarn**
Sooyoung	Solji	UNKR $\Longrightarrow$ **Solji**
Kwon Yuri	Lee Eun-jeong	UNKR $\Longrightarrow$ **Yoo Ara**
Im Yoon-ah	Im Seong-eon	UNKR $\Longrightarrow$ **Lee Jin**
Taeyeon	Shihono Ryo	UNKR $\Longrightarrow$ **Jin Se-yeon**
Kate Mara	Gemma Atkinson	UNKR $\Longrightarrow$ UNKR
Kim Yuna	Choi Ja-hye	UNKR $\Longrightarrow$ UNKR

Initial Correct Label	**Succ. Amazon Adversarial Label**	**(Before) $\Longrightarrow$ (After): Fooling Naver with Amazon**
Sohee	Park Soo-jin	Sohee $\Longrightarrow$ **Park Soo-jin**
Alicia Keys	Cindy Bruna	Alicia $\Longrightarrow$ Alicia

7 Discussions, and Limitations

Robust DNNs and conservative reporting: One possible defense approach is to make DNNs more robust against noise via adversarial training with GA [7]. Also, it is better to be more conservative in reporting an output label, when a confidence value is low. For example, if the confidence value is below 70%, APIs can generate 'UNKR'. In this way, APIs do not provide any feedback to attackers, and adversarial example generation cannot be proceeded. Instead of attempting to make the best guess always, it is important to know "when APIs do not know." From the defense perspective, it is better to be conservative and even not to report any results when confused. However, clear trade-offs among customers' service needs, performance, and security requirements have to be considered to better design the overall defense mechanisms.

Network Level Rate Limiting and Noise Filtering: In order to create adversarial examples, several queries need to be made to obtain returned output labels and confidence values. The large number of API queries per sec. for the same or similar images can be a suspicious adversarial attack activity. Hence, various rate limiting techniques such as CAPTCHAs and network defense mechanisms can be employed. However, this cannot be effective for distributed *GenAttack* querying over multiple IPs or with slower rates. Also, we need a more sophisticated pre-filtering strategy to learn noise patterns generated from our GA, and remove those more effectively. Currently, we are investigating improved noise filtering techniques.

Limitations and future work: Even though GA searches for an optimum noise value, it is not guaranteed to find a global optimum noise. GA can resort on the local optimum, because of the nature of the evolutionary algorithm. Also, finding an optimum noise without access to DNN parameters is a challenging task. Further empirical experiments and theoretical analysis are needed to control different GA parameters to fine-tune the noise. For future work, we plan to compare *GenAttack* with other attack and defense mechanisms[7, 12, 14, 24].

8 Conclusion

We introduce a simple yet powerful method, *GenAttack*, to generate adversarial images, which does not require any knowledge about DNNs or use GPU resources. *GenAttack* optimizes noise using a iterative approach and can provide significant benefits over other complex gradient based estimation attacks. Further, we show that *GenAttack* is highly practical, and is transferable to attack other classifiers.

Acknowledgement

We would like to thank Adam Eisenstadt and anonymous reviewers for their valuable feedback. This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ICT Consilience Creative program(IITP-2017- R0346-16-1007) supervised by the IITP (Institute for Information & communications Technology Promotion), and by NRF funded by the MSIT(NRF-2017R1C1B5076474).

References

1. Amazon rekognition - deep learning-based image analysis. `https://aws.amazon.com/rekognition`. Accessed: 2017-12-30.
2. Naver - clova face recognition. `https://www.ncloud.com/product/aiService/cfr`. Accessed: 2017-12-30.
3. M. Barreno, B. Nelson, A. D. Joseph, and J. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.
4. B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.
5. J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.
6. N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.
7. I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
8. K. He, G. Gkioxari, P. DollÂŽar, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.
9. H. Hosseini, B. Xiao, and R. Poovendran. Deceiving google's cloud video intelligence api built for summarizing videos. *arXiv preprint arXiv:1703.09793*, 2017.
10. H. Hosseini, B. Xiao, and R. Poovendran. Googles cloud vision api is not robust to noise. *arXiv preprint arXiv:1704.05051*, 2017.
11. R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

12. A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv/1611.01236*, 2016.
13. J. Lu, T. Issaranon, and D. A. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *CoRR*, abs/1704.00103, 2017.
14. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
15. S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.
16. A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.
17. N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.
18. N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.
19. N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.
20. S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2016.
21. Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.
22. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.
23. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
24. F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.
25. P. Vidnerová and R. Neruda. Evolutionary generation of adversarial examples for deep and shallow machine learning models. In *Proceedings of the 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016*, page 43. ACM, 2016.
26. X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *arXiv preprint arXiv:1712.07107*, 2017.