Black-Box Audio Adversarial Attack Using Particle Swarm Optimization

HYUNJUN MUN, SUNGGWAN SEO, BAEHOON SON, AND JOOBEOM YUN
Department of Computer and Information Security and Covergence Engineering for Intelligent Drones, Sejong University, Seoul 05006, South Korea
Corresponding author: Joobeom Yun (jbyun@sejong.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2021-2020-0-01602; and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2021R1F1A1051251.

ABSTRACT The development of artificial neural networks and artificial intelligence has helped to address problems and improve services in various fields, such as autonomous driving, image classification, medical diagnosis, and speech recognition. However, this technology has raised security threats that are different from existing ones. Recent studies have shown that artificial neural networks can easily malfunction by adversarial examples. The adversarial examples operate the neural network model as intended by the adversary. In particular, adversarial examples targeting speech recognition models is an area that has been actively studied in recent years. Existing studies have focused more on white-box methods. However, most speech recognition services are provided online and involve black-box, making it difficult or impossible for adversaries to attack. Black-box attacks have several challenges. Typically, they have a low success rate and a high risk of detection. In particular, previously proposed genetic algorithm (GA)-based attacks are at a high risk of detection because they require numerous queries. Therefore, we propose an adversarial attack system using particle swarm optimization (PSO) algorithms to address these problems. The proposed system uses adversarial candidates as particles to obtain adversarial examples through iterative optimization. PSO-based adversarial attacks are more efficient in queries and have a higher attack success rate than the adversarial methods using GAs. In particular, our key function is that temporary particle generation maximizes query efficiency to reduce detection risk and prevent wastage of system resources. On average, our system exhibits 96% attack success rates with 1416.17 queries, indicating that 71.41% and 8% better in terms of query and success rates than existing GA-based attacks, respectively.

INDEX TERMS Adversarial example, automatic speech recognition, black-box attack, genetic algorithm (GA)-based attack, particle swarm optimization (PSO).

I. INTRODUCTION

The development of artificial neural networks [1] and artificial intelligence [2] has helped to address various problems in multiple areas, such as autonomous driving [3], image classification [4], medical diagnosis [5], and speech recognition [6]. However, recent studies [7], [8] have demonstrated that neural networks can malfunction owing to intentionally perturbed adversarial examples.

Previous studies on adversarial examples have mainly been conducted on image domains, but recently, studies have also been actively conducted in other domains [9]–[11]. For example, there have been studies on the effect of audio adversarial examples on automatic speech recognition using these examples, adversaries can fool smart devices to run the commands that they intend. For instance, under an adversarial attack, the listener hears the command correctly as “Alexa, turn off the light,” but the smart device runs the command “Alexa, turn on the gas stove.” Such attacks can cause significant damage to property or safety concerns. Figure 1 shows an outline of audio adversarial examples.

The adversarial attack has two categories depending on the accessibility to the neural network model: white-box and black-box. White-box attacks [8], [11]–[13] assume that adversaries are accessible to the internal states of the model such as gradient, parameters, and structures. Therefore, it is
The audio adversarial example is perturbed audio that causes misclassification of speech recognition models while maintaining human perception. For instance, the human hears the adversarial example as the original sound, but the speech recognition system hears something that the adversary intended.

In the audio domain, existing black-box attacks [17], [18] use genetic algorithms (GAs) [19]. However, these attacks are less efficient and less effective in terms of number of queries and performance. On average, they require thousands of queries for success, and the success rate and perturbation quality are also poor. To overcome this problem, we propose a novel audio black-box adversarial attack using particle swarm optimization (PSO) [20]. The PSO algorithm is a metaheuristic optimization method inspired by the behavior of biological groups. The PSO algorithm has several advantages: it does not require a gradient for optimization, is simple to implement, has low computational costs, and is faster than GA in global optimization. These advantages are suitable for black-box adversarial attacks, which should be performed with finite information and resources. In this study, adversarial example candidates are placed in the search space as particles and iteratively moved with a specific velocity based on the fitness score designed to find adversarial examples. If the particle reaches the global optimum, the particle is an adversarial example, and the accumulated velocity becomes a perturbation.

Although this approach has already been performed in the image domain [21], we optimized and developed several algorithmic components for appropriate application to the audio domain. In particular, we propose temporary particle generation based on GA operations, which are the key functions of our method. In practice, temporary particle generation helps the algorithm perform well with fewer queries or increases the success rate with additional queries. We aimed to derive the maximum adversarial attack performance using minimum queries. We achieved better efficiency and performance than existing methods based on GA [17].

Our contributions are as follows:

- **PSO and GA-based adversarial attacks**: We further developed existing adversarial attacks based on PSO algorithms [21] in the image domain, suggesting a novel audio black-box adversarial attack incorporating genetic algorithmic operations.

- **Maximizing query efficiency and performance**: Our method reduced the required queries to the extent possible and improved attack success rates to overcome the drawbacks of black-box adversarial attacks. Consequently, The most remarkable performance was a 96% success rate in 1416.17 queries. The queries decreased 71.41% compared to the GA-based method, and the attack success rate was comparable to a white-box attack.

- **Analysis and optimization of audio black-box adversarial attacks**: We analyzed the mechanism through which black-box adversarial attacks can be combined with PSO algorithms through various experiments.

The remainder of this paper is organized as follows: Section II covers existing studies that form the basis of our system. In Section III, we explain the proposed system. In particular, we clearly show the contribution of PSO and GA to adversarial attacks. In Section IV, we evaluate the proposed system. Section V discusses the limitations and future work. Finally, Section VI summarizes the overall contents of this paper.

II. RELATED WORK

A. ADVERSARIAL EXAMPLE ON IMAGE

In this section, we describe existing adversarial example techniques and PSO. Szegedy et al. [7] introduced an adversarial example for the first time. They used the box-constrained limited L-BFGS method to perform an adversarial attack. Furthermore, they showed that adversarial examples can be applied to different models, even when trained with a different dataset. Transferability shows the universality of the adversarial examples.

Goodfellow et al. [8] claimed that the high-dimensional linearity of deep neural networks is vulnerable. They proposed the fast gradient sign method (FGSM) that adds imperceptible perturbations to the input. The perturbation is computed based on the gradient sign calculated by backpropagation and it increases the loss. FGSM is simple, fast, and exhibits excellent performance.

Carlini and Wagner [12] proposed an efficient and powerful method similar to that of Szegedy et al. [7] under a white-box situation. They devised a hinge-like objective function that allows them to have high target class confidence and small perturbation by balancing adversarial confidence and perturbation size. The proposed method exhibits a high success rate for targeted attack and neutralizes defensive distillation [22].
White-box adversarial attacks are easy to generate but impractical because they assume full access to the model. By contrast, the black-box attack assumes that is performed only with knowledge regarding the input to the model and the following output. In other words, it has an advantage that is more practical because it uses only information that can be obtained realistically to perform adversarial attacks. Therefore, black-box attacks generally replace internal information with query information. For instance, using query information, substitute models are trained and then, the original network is attacked using transferability [16], or adversaries optimize adversarial examples directly [23], [24]. However, it also has disadvantages such as detection risk or poor performance.

Chen et al. [14] proposed a black-box attack using zeroth-order-optimization (ZOO) that can directly attack neural networks without the substitute model. They optimized the hinge-like objective function inspired by Carlini and Wagner attacks [12] and estimated gradient and Hessian using a symmetric difference quotient. A black-box attack requires many queries and a substantial computational cost for estimating the gradient. Therefore, they randomly select a part of the inputs rather than all of them and update adversarial examples. In the experiments they conducted, ZOO achieved comparable performance to those of Carlini and Wagner attacks [12].

Su et al. [15] proposed a one-pixel attack that uses only the input of the model and the output accordingly. A One-pixel attack finds optimal adversarial perturbations using differential evolution, a type of evolutionary algorithm. Despite changing only one or a few pixels, the one-pixel attack exhibits satisfactory good achievement in the targeted attack scenario.

Black-box adversarial attacks have the advantage that adversaries can generate adversarial examples even though they do not know the gradient of the model; however, these attacks require numerous queries and high computational costs. To address this problem, Mosli et al. [21] proposed a black-box adversarial attack using the PSO algorithm [20] known as AdversarialPSO. They randomly placed the images in the search space as particles and thereafter moved them to find adversarial examples. Compared to ZOO, the queries decreased by 99% while maintaining a high non-targeted attack success rate. In addition, a higher attack success rate, fewer $L_2$ perturbations, and fewer queries were achieved compared to other black-box adversarial attacks based on GA [19].

Gong et al. [30] proposed an audio adversarial example based on a gradient. Utilizing iterative FGSM [8], they generated audio adversarial examples against paralinguistic neural network models, such as gender, emotion, and speaker recognition models.

Du et al. [31] attacked speech recognition models using modified PSO algorithms instead of the standard one. They modified the PSO algorithm to maintain traceability by adding a swarm-wide best position to the seed every epoch. In this study, we approached from a different perspective than Du et al. [31]. We assumed that the characteristics of particles sharing information with each other help explore fast adversarial examples. Therefore, we focused on increasing particle diversity, not the best position. This approach is shown in Section III-C.

Carlini and Wagner [11] performed targeted adversarial attacks on the speech recognition model DeepSpeech [6]. They merged connectionist temporal classification (CTC) loss [32] into an objective function similar to their previous work on image adversarial attacks [12]. The generated adversarial examples translated the targeted phrase intended by the adversary, but not the original phrase. The generated examples also had minimal quantitative distortion; thus, having no significant effect on auditory cognition. The authors showed that the generated audio adversarial example, like the image, also had transferability.

Alzantot et al. [17] first applied a GA to generate audio adversarial examples. They generated an adversarial population from the perturbed original audio. The population was bred using crossover and mutation, and candidates were selected with the highest target class probability. The targeted attack on the speech command classification model was successful, but a quantitative distortion evaluation was not performed.

Black-box adversarial attacks using GAs achieve good performance when exploring large search spaces quickly, but there is a disadvantage of poor accuracy in finding global optimizations of GA [33]. Taori et al. [18] combined the GA and gradient estimation to overcome this disadvantage. First, they generated initial adversarial examples through a GA while the edit distance [34] was more than two between the adversarial and the targeted phrase. Then, they fine-tuned the adversarial examples using gradient estimation. Consequently, they succeeded in generating adversarial examples that were completely consistent with the target sentence with a success rate of 35% even in a black-box situation.

Audio adversarial examples almost lose adversarial properties when physically playing owing to ambient noise, distortion by I/O devices, and room impulse. The over-the-air transmission problem is one of the main challenges in audio adversarial examples. Previous studies [28], [29], [35]–[37] proposed robust audio adversarial examples to maintain adversarial properties in over-the-air transmission. They mainly pre-captured and incorporated the core of distortion (input, ambient noise, reflection, and echo) primarily.
during physical playing into the adversarial example optimization process.

Most commercial speech recognition systems, such as Amazon Alexa and Apple Siri, provide only transcripts, not prediction scores. Since most black-box adversarial attacks perform optimization based on prediction scores, they do not apply to these limited black-box models, which is the most challenging scenario. Zheng et al. [38] defined the generation of audio adversarial examples for limited black-box as discontinuous optimization. To solve the complex optimization problem, they divided it into sub-problems and solved them by cooperatively optimizing them.

C. PARTICLE SWARM OPTIMIZATION

Kennedy and Eberhart [20] and Shi [39] first proposed PSO, a metaheuristic optimization method inspired by biological group behavior such as bird flocking. Figure 2 shows an overview of the PSO algorithm. For each iteration, \( n \) particles in a swarm \( P = \{x_1, x_2, \ldots, x_n\} \) move with their respective positions \( x_i \) and velocities \( v_i \) to the search space. The initial positions and velocities were randomly initialized. The next velocity reflects not only the current velocity but also the information of the swarm and is formulated as follows:

\[
v_i^{t+1} = v_i^t + r_1(P_i - x_i^t) + r_2(P_g - x_i^t)
\]

where \( r \) denotes a random value, \( P_i \) indicates the best position each particle has experienced, and \( P_g \) indicates the best position throughout the swarm. Figure 3 shows the geometric analysis of the velocity calculation in the PSO. The strategy of using swarm-wide information for individual particle optimization gives the advantage of global optimization. Even if individual particles fall into a local optimum, swarm information helps them to escape from there and easily reach the global optimum [40].

The newly updated position \( x_i^{t+1} \) calculates the score through the objective function which is designed for the optimization problem, and if Equation 2 is satisfied, update \( P_i = x_i^{t+1} \) because the newly found position is closer to the global optimum than the previously experienced best position.

\[
f(x_i^{t+1}) > f(P_i)
\]

(2)

\[
f(P_i) > f(P_g)
\]

(3)

In addition, when Equation 3 is satisfied, the swarm-wide best position is updated. When the evaluation is completed by Equation 3, the algorithm checks whether \( P_g \) is the optimal solution. If not, the algorithm returns to Equation 1 and repeats the steps of the velocity calculation, position update, and score calculation.

We aim to optimize the black-box audio attack, a more practical attack scenario, among the numerous problems associated with the successful audio adversarial attacks. Recently performed black-box audio adversarial attacks are mostly GA-based and require thousands of queries with low success rates. Therefore, we propose the use of a PSO algorithm in generating adversarial examples. The PSO algorithm is simple, and optimization does not require a differential (that is gradient). In addition, according to previous studies [33] that compared GA and PSO algorithms, PSO algorithms have lower computational costs and better global exploration capabilities than GA. These advantages of PSO algorithms are suitable for the black-box adversarial examples. Our proposed method explores the search space more efficiently than before and generates better quality adversarial examples.

III. AUDIO ADVERSARIAL ATTACK USING PSO

In this section, we explain an audio adversarial attack using the PSO algorithm. Figure 4 shows an overview of the proposed system. The system is divided into three steps: initialization, optimization, and temporary particle generation. First, initialization places the perturbed original audio
Algorithm 1 Initialization

Input: input audio $x$, particle array $P$
Output: $P$, $SBP$, $SBF$

1. $SBF \leftarrow -\infty$ # SwarmBestFitness
2. $SBP \leftarrow x$ # SwarmBestPosition
3. for $p$ in $P$ do
4.  $\epsilon \leftarrow$ random noise
5.  $p.Position \leftarrow x + \epsilon$
6.  $p.BestFitness \leftarrow$ GetFitness($p$)
7.  $p.BestPosition \leftarrow p.Position$
8.  if $p.BestFitness > SBF$ then
9.   $SBF \leftarrow p.BestFitness$
10. $SBP \leftarrow p.BestPosition$
11. end if
12. end for
13. return $P$, $SBP$, $SBF$

Algorithm 2 Optimization

Input: input audio $x$, particle array $P$, maximum iteration $t_{max}$, number of temporary particles $n$
Output: $SBP$

1. for $t = 1, 2, \ldots, t_{max}$ do
2.  if $SBF > 0$ then
3.    return $SBP$ #Found Adv, stop early
4.  end if
5.  for $p$ in $P$ do
6.    $v \leftarrow$ CalculateVelocity
7.    $p.Position \leftarrow$ UpdatePosition($p$, $v$)
8.    Fitness $\leftarrow$ GetFitness($p$)
9.    if Fitness $> p.BestFitness$ then
10.     $p.BestFitness \leftarrow$ Fitness
11.     $p.BestPosition \leftarrow p.Position$
12. end if
13. if $p.BestFitness > SBF$ then
14.    $SBF \leftarrow p.BestFitness$
15.    $SBP \leftarrow p.BestPosition$
16. end if
17. end for
18. if No SBF improve then
19.    $P = GenerateTemporaryParticles(P, n)$
20. end if
21. end for
22. return $SBP$

as initial particles in a swarm. Second, optimization explores adversarial examples by updating and evaluating the particles. Finally, temporary particle generation is a function that improves query efficiency and increases short-term searchability through temporary additional particles when the particles fall into the local optimum.

A. INITIALIZATION

The generation of adversarial examples using the PSO algorithm begins by initializing the swarm, as described in Algorithm 1. The initialization begins by adding white noise $\epsilon$ to the original audio $x$ as much as the swarm size. The swarm size sets the number of particles and affects both the attack success rate and query; therefore, it should be decided carefully. The algorithm thereafter evaluates the fitness score of the initial particles. The fitness score is the output value of an objective function used to generate the adversarial examples. We designed the objective function such that the PSO algorithm can intuitively find adversarial examples, and formulated it as follows:

$$Fitness\_Score = \{Z(x'_T) - \arg \max (Z(x'_i)_{i \neq T})\}$$  \hspace{1cm} (4)

where $x'$ denotes the position of the particle, $T$ is the target label and $i$ represents the others. The term $Z(x)$ denotes the logit, which is the input of the softmax layer. It is used rather than the probability output to cope with adversarial probability changes sensitively. To obtain high adversarial confidence in black-box settings, we set the logit of the target class to be higher than others, the particles are more rewarded by a higher fitness score. In other words, when the fitness score is greater than zero, the swarm best particle is an adversarial example. This approach was inspired by the work of Carlini and Wagner [11] and Mosli et al. [21].

The objective function rewards the particles with a higher fitness score when the probability of the target label is higher than that of the other labels. The fitness score starts with a negative number when the particles are not adversarial examples, then increases through optimization, and subsequently exceeds zero when the algorithm finds adversarial examples. In other words, the algorithm solves a type of maximization problem.

B. OPTIMIZATION

Optimization is the process of finding a better fitness through iterated particle movements and is described in Algorithm 2. In every iteration, the new position of the particle is updated using the velocity. Velocity indicates the direction in which the particle moves using the velocity vector $X$ is an n-dimensional vector $V = (v_1, v_2, \ldots, v_n)$. Each particle moves to a new position using its n-dimensional velocity vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$.

In every iteration, the new position of the particle is updated using the velocity. Velocity indicates the direction in which the particle moves using the velocity vector $X$ is an n-dimensional vector $V = (v_1, v_2, \ldots, v_n)$.

According to iteration $t$, the next position is formulated as follows:

$$x_{i^t}^{t+1} = x_{i^t}^t + v_{i^t}^{t+1}$$  \hspace{1cm} (5)

$$v_{i^t}^{t+1} = wv_{i^t}^t + c_1r_1(P_i - x_{i^t}^t) + c_2r_2(P_B - x_{i^t}^t)$$  \hspace{1cm} (6)

Equation 6 determines the direction in which the particle moves using the sum of the three terms. The first term refers to inertia, which regulates the influence of current velocity on the next velocity and is weighted by the inertia weight $w$. The second term is memory, which is influenced by the best position that each particle has experienced and is denoted as $P_i$. Additionally, it is weighted with constants $c_1$ to determine
the extent to which memory affects the velocity. It is also randomized with a uniform distribution number $r_1$ to provide the search process with randomness. The third term is referred to as information, which is influenced by the swarm-wide best position denoted as $P_c$. For the same reason as that for the second term, the third term includes the weight $c_2$ and the uniform distributed random number $r_2$.

The inertia weight $w$ is a hyperparameter that significantly affects convergence. Additionally, it helps balance between memory and information of the PSO algorithm. The initial study used a fixed $w$; however, subsequent studies improved the performance of the PSO algorithm using various $w$s. In our implementation, we generated adversarial examples using linearly decreasing inertia weights $[41]$, which achieved the best performance at minimum error based on Bansal et al. $[42]$. This was selected for the following two reasons. First, the linear decreasing inertia causes particles to move more in the initial iteration and less afterward. This characteristic is suitable for generating adversarial examples that require precise perturbation control as the optimization progresses. Second, we expected that the minimum error would help generate more accurate adversarial examples. The linearly decreasing inertia weight $[41]$ is formulated as follows:

$$w = w_{\text{max}} - t \cdot \frac{w_{\text{max}} - w_{\text{min}}}{t_{\text{max}}} \quad (7)$$

To prevent excessive perturbation from being added to the audio, we limit the $L_2$ distance between the original and adversarial audio using the maximum change upper bound $B$. The $L_2$ distance measures the Euclidean distance between the two coordinates. We calculated the distance and clipped the particles using the following equations:

$$L_2(x', x) = \sqrt{\sum_{i} (x'_i - x_i)^2} \quad (8)$$

$$x'_i^{t+1} = clip(x'_i + v'_i^{t+1}, x - B, x + B) \quad (9)$$

Each iteration calculates the fitness of the updated particle using Equation 4. The algorithm continues to update the best fitness and position of the particle depending on the fitness score. In addition, it updates the swarm-wide best fitness and position (SBF and SBP) using the same rule. These updates indicate that the algorithm finds a new and better adversarial example candidate within the swarm.

As optimization continues, perturbations defined as velocity accumulate in the initial particles. Then, depending on the design of the objective function, the proposed method moves the particles to have a target class, and the accumulated perturbation becomes noise to obtain adversarial examples.

The algorithm ends when it reaches the maximum iteration (that is, it cannot find an adversarial example) or when the SBF is greater than zero (that is, the SBP is an adversarial example). Except for these cases, the algorithm proceeds to the temporary particle generation step when the SBF is no longer improved.

---

**Algorithm 3 Temporary Particles Generation**

**Input:** particle array $P$, number of temporary particles $N$

**Output:** $P$

1: for $i = 1, 2, \ldots, n$ do
2: Select $parent1$ from $P$
3: Select $parent2$ from $P$
4: $child \leftarrow \text{Mutate(Crossover}(parent1, parent2))$
5: $child.BestFitness \leftarrow \text{GetFitness}(child)$
6: $child.BestPosition \leftarrow child.\text{Position}$
7: if $child.BestFitness > SBF$ then
8: $SBF \leftarrow child.BestFitness$
9: $SBP \leftarrow child.BestPosition$
10: end if
11: $P.append(child)$
12: end for
13: return $P$

---

**C. TEMPORARY PARTICLES GENERATION**

One of the most important points when generating black-box adversarial examples through PSO is that attack performance and queries increase in proportion to swarm size. Increasing the swarm size achieves a better attack success rate; however, the risk of detection also increases dramatically owing to the abnormal number of queries. To address this problem, we propose temporary particle generation, which starts with a small swarm size and adds temporary particles as necessary. The temporary particle generation is one of the most important parts of our method, as expressed in Algorithm 3.

During the optimization process, when particles fall to the local optimum, the fitness score no longer improves. In PSO, swarm-wide information helps to avoid such local convergence, but it can still occur. Thus, we devised a method to provide explosive additional search capabilities on a swarm using temporary particles. The algorithm generates new children particles from existing particles using GA operations such as crossover and mutation. The crossover is an operation in which two selected parents are mixed at an arbitrary division point to create a new child. Mutations are operations that provide randomness to newly created children. We simply add the random noise. As in any other step, the generated temporary particles calculate, update the fitness, and finally add to the swarm.

Our approach has several advantages. First, rather than optimizing using several particles initially, the overall number of queries decreases because the particles are generated as needed. Second, if the SBP falls into the local optimum and the fitness is no longer improved, temporary particles can effectively escape by exploiting their search capabilities. Third, temporary particles generated by the dielectric algorithm have different directions and positions from existing particles, which allows them to extend their search field and reach a corner case.
D. EARLY TERMINATION
Temporary particle generation helps in finding adversarial examples away from the local optimum through additional search capabilities but is not always successful. A swarm with temporary particles may fall into another local optimum too. In this time, temporary particles can instead unintentionally increase the number of queries. Therefore, we implemented an early termination that terminates the algorithm if the fitness does not improve after temporary particle generation.

IV. EVALUATION
A. METRICS
- **Target model and Dataset:** We attacked a convolutional neural network (CNN)-based speech command classification model [43] provided by Tensorflow [44] and assumed a black-box situation in which the internal state of the model was inaccessible except for inputs and subsequent outputs. We trained the model using a speech command dataset [45] provided by Google and achieved 99% accuracy through fine-tuning. The speech command dataset [45] consists of 65,000 audio files that are one-second audio clips of single word audio, and we used the following ten classes: “no”, “left”, “down”, “on”, “up”, “right”, “off”, “yes”, “stop”, and “go”.
- **Evaluation metric:** We considered three criteria to evaluate our methods. The first is the success rate. It is the ratio of adversarial examples generated from the original audio and only counts if the original class has precisely changed to the target class. The second is the average number of iterations and queries. This indicates the number of average iterations and queries that need to be sent to the target model for an adversarial attack. The query is calculated once each time it interacts with the neural network model. Finally, we measured the perceptions of adversarial examples: quantitative evaluation and human perception.

B. EVALUATION SETUP
We selected 500 random audio clips (50 per label) from the dataset for the targeted adversarial attack experiments. In each audio clip, we generated targeted adversarial examples except for the original class. In other words, nine adversarial examples were generated from one audio clip. Therefore, we generated 4500 adversarial examples. We specified the maximum number of iterations as 500. The swarm size refers to the number of particles and is represented by a combination of P and numbers. We set the linearly decreasing inertia weight (Equation 7) to \( w_{\text{max}} = 1.0 \) and \( w_{\text{min}} = 0.0 \) to gradually decrease from 1 to 0 following the iteration. The maximum change in upper boundary B set the adversarial example not to change by more than 10% in the original audio. The algorithm ends in three cases. First, when the fitness score is greater than zero (that is, when the adversarial attack is successful). Second, when reaches the maximum iteration. Third, when the fitness score was no longer improved (that is, early termination).

C. RESULT USING PSO
Figure 5 shows the targeted adversarial attack success rate based on the swarm size. The average success rate achieved 89.53%, 94.08%, 95.68%, and 97% at P25, P50, P100, and P150, respectively. The PSO algorithm effectively generated adversarial examples for most labels. In addition, although the attack success rate for a particular label was slightly lower than others, it increased with the swarm size.

Table 1 summarizes the effect of swarm size on the success rate, average iteration, average query, and average \( L_2 \) distance. The algorithm achieves satisfactory performance in terms of the success rate and \( L_2 \) distance compared to the query consumed. For instance, P25 achieved a success rate of 89.53%, an \( L_2 \) distance of 6.98, and a query of 699.17. As the swarm size increases, the success rate, average query, and average \( L_2 \) distance increase simultaneously. However, the larger the swarm size, the smaller the average number of iterations. In the evaluation, a larger swarm finds adversarial examples better.

As shown in Figure 5 and Table 1, the swarm size directly affects the performance. However, queries increase more rapidly than improve success rates. Figure 6 shows the query efficiency according to the change in the swarm size. The horizontal and the vertical axes represent the swarm size, query efficiency, respectively. The query efficiency is the ratio of success rates per query. When the swarm is small (P25), the query efficiency is high at 12.81% because the algorithm generates adversarial examples with fewer queries. However, as the swarm increases (from P50 to P300), the query efficiency drops sharply, to approximately 0%. Both the success rate and the query increase with swarm size; however the success rate rarely increases in a large swarm, but the query increases significantly.

D. RESULTS OF TEMPORARY PARTICLE GENERATION
Through observations and results of the experimental process, we determined that inhibiting query efficiency was a local convergence problem. The swarm that fell in local optimum

| Swarm Size | Success Rate | Avg. Iteration | Avg. Query | Avg. \( L_2 \) Dist. |
|------------|--------------|----------------|------------|---------------------|
| 25         | 89.53%       | 26.92          | 699.17     | 6.98                |
| 50         | 94.08%       | 20.31          | 1064.94    | 7.36                |
| 100        | 95.68%       | 16.07          | 1708.93    | 7.52                |
| 150        | 96.73%       | 13.87          | 2231.53    | 7.57                |
| 200        | 97%          | 12.75          | 2752.15    | 7.60                |
| 300        | 97.42%       | 11.10          | 3633.73    | 7.64                |
continued to waste queries even though optimization was no longer underway. Therefore, we attempted to prevent this local convergence problem and wastage of computational costs by introducing temporary particle generation. The results applying the temporary particle generation are shown in Figure 7 and Table 2.

Figure 7 shows the success rate after applying the temporary particle generation method, which was performed under the same conditions in Section IV-B. The title of each subfigure represents the number of primary particles with additional particles. The attack success rate increased after applying temporary particle generation regardless of swarm size. The relationship between the success rates and swarm size for a particular label remained similar. In other words, the tendency toward generating adversarial examples was similar to the previous experiments. Overall, temporary particles help primary particles to explore adversarial examples.

However, this approach is meaningless if the increase in the number of temporary particles generates the query more than is necessary. For instance, 50+50 particles should use fewer queries than 100 particles. The results of applying the temporary particle generation are summarized in Table 2. In comparison with results, temporary particles increased the success rate with a minor effect on the queries. Algorithms with temporary particles reduced average queries by 18.75%, 24.90%, 36.54%, and 29.64% based on the total cluster size P50, P100, P150, and P200, respectively.
In particular, generating a large number of temporary particles simultaneously from the small swarm consumes fewer queries than otherwise (that is, generating a small number or several times). Our most remarkable achievement was also observed in this case. P50 + 100 achieved a 96% high success rate with 1416.17 queries. This result supports our argument that the purpose of temporary particle generation is to increase a short-term explosive search capability when the fitness score is stagnant. Temporary particle generation starts additional optimization from the time when the existing algorithm is stagnant. Therefore, the average iteration increased. Similarly, the distance also increased by a small amount because further optimization implies additional perturbation.

Figure 8 shows the effect of temporary particle generation on query and efficiency. After applying temporary particle generation, query efficiency increased by 20.40%, 32.01%, 55.33%, and 41.52% for P50, P100, P150, P200, respectively. In particular, query efficiency increased significantly in the large swarm than in the small swarm, showing that temporary particles operate according to their original purpose of maximizing success rates while minimizing query growth. Overall, P150 showed the best performance in both success rate, number of queries, and query efficiency.

**E. PERCEPTION OF ADVERSARIAL EXAMPLES**

Figure 9 shows overlapped waveforms visualizing the difference between adversarial and original audio. Visually, the generated adversarial example makes no significant difference compared to the original audio. Most of the noise was evenly spread throughout the waveform, and the volume was also small.

Table 3 evaluates the perception of adversarial examples using two criteria: quantitative evaluation and human perception. Quantitative evaluation analyzes the magnitude of the perturbation or noise numerically. The $L_2$ distance represents the smallest distance between the adversarial example $x'$ and the original audio $x$, also known as Euclidean distance, calculated in Equation 8. SNR represents the relative magnitude of the signal to noise and is expressed as follows:

$$\text{SNR}(x, \delta) = 10 \cdot \log_{10}\left(\frac{P_x}{P_\delta}\right) \quad (10)$$

where $x$ denotes the original audio, $\delta$ is the noise, and $P$ is the power. In other words, the larger the SNR, the more minor the noise compared to the audio. The quantitative evaluation averaged regardless of swarm size. The $L_2$ distance was 7.45 and the SNR was 20.23. The perturbation is quantitatively very small and achieves results comparable to those of the existing white-box method [46]–[48].

Human perception evaluates how physically reproduced adversarial examples actually sound and are perceived by people. Twenty participants listened to and classified different 20 adversarial examples that were randomly selected regardless of the swarm size. We recorded the mechanism through which the participants classified the adversarial examples. Participants classified 94% of the total 400 adversarial examples such that adversarial sounds were heard.
adversarial examples as labels of the original audio, and only 6% were misclassified as other labels.

Table 4 shows the results of human perception by label. Participants recorded to 'original' when an abnormality was not detected and accurately classified as the original label and recorded to 'others' when an abnormality was detected or classified as another label. Participants did not detect abnormalities in most adversarial examples except specific labels which have low attack success rates (i.e., 'up' and 'stop'). This result is because labels that are difficult to attack generally require more iterations and particle movement, which gradually increases perturbation.

F. COMPARISON WITH GA-BASED METHODS

Table 5 compares our method with the existing adversarial method [17] based on the GA. (P) indicates the evaluation of previous studies [17], whereas (E) indicates our experiments. We applied the same setting and evaluation criteria in our study. In other words, we used the same subset of audio clips for each experiment for accurate comparison. The number
TABLE 5. Comparison with existing GA-based method. (P) indicates the evaluation of previous studies [17], whereas (E) indicates our experiments.

| Method     | Success rate | Avg. query |
|------------|--------------|------------|
| P25        | 89.53%       | 699.17     |
| P50+100    | 96%          | 1416.17    |
| Genetic [17]| 86% (P)     | N/A (P)    |
|            | 88% (E)      | 4954.83 (E)|

FIGURE 10. Overlapped original audio clip waveforms (orange) with adversarial examples (blue) generated by GA-based method. The label of the original audio clip is “no,” whereas the label of the adversarial audio clip is “up.”

of iterations and the number of audio clips are the same as 500. The query was measured only when it directly interacted with the model. The evaluation part of [17] achieved an 86% probability of success and did not evaluate queries. In our experiment, [17] averaged 4954.83 queries and recorded an 88% success rate. Compared to [17], our method achieved better performance in terms of both query and success rate. Based on P25, the average query was 85.88% less, and the success rate was 1.53% higher. In addition, compared with P50+100, which had the best query efficiency among large swarms, the average query was 71.41% less, and the success rate exhibited a large difference of 8%. Also, as shown in Figure 10, the perturbation was also loud. This result demonstrates that PSO algorithms with faster global optimization capabilities compared to GA are more suitable for black-box adversarial examples that require minimal queries.

V. DISCUSSION

In this section, we discuss the limitations of our studies and further developments.

- Our method targets a one-word audio clip for just one second. However, most speech recognition models used in smart devices target sentences that consist of several words. Unlike a general classification model with a limited number of output labels, the speech recognition model has no limitation in length and has an uncountable output case. Therefore, it is difficult to calculate the fitness based on the probability of the label and generate targeted adversarial sentences or phrases. To address this problem, we consider a new objective function using CTC loss [32] to calculate the fitness scores. We can attack speech recognition models using CTC loss between the targeted and adversarial phrases rather than the probability in calculating the fitness score because CTC loss outputs a loss value for a variable sequence such as voice.

- Adversarial attacks that iteratively optimize populations such as GA and PSO are excellent for global optimization. However, if the candidates are closer to the global optimum then search capabilities decrease sharply. In the worst case, the search falls into a nearby local optimum. In addition, it is difficult to obtain a finely tuned minimum perturbation using such methods. In other words, it is difficult to generate high-quality adversarial examples because a black-box attack cannot obtain the gradient of the model. To address this problem, we consider gradient estimation through additional queries. Existing studies [8], [14], [18], [49], achieved satisfactory performance with only signs or approximations of the gradient. Therefore, we can fine-tune the adversarial examples after gradient estimation, even if the estimated gradient is not very accurate.

- Over-the-air transmission attacks that physically play audio adversarial examples and attack smart devices are one of the most important challenges. Over-the-air transmission requires more detailed and computationally intensive optimization to avoid losing its adversarial properties during physical playing. Consequently, previous studies [28], [35], [36] were mainly conducted with white-box attacks. However, black-box attacks are difficult to perform because of a lack of information. The same applies to our system, and the mechanism which enables physical transmission in black-box attacks should also be discussed.

VI. CONCLUSION

We proposed a novel black-box adversarial example that combines PSO and GA for an efficient audio adversarial example. The initial particles with white noise moved iteratively within the swarm and found adversarial examples. PSO has better global optimization capability and effectively generated adversarial examples with an improved number of queries and success rates compared with existing GA-based attacks. In particular, the key function of our system, temporary particle generation, helped improve the performance while preventing unnecessary query waste. In both quantitative evaluation and human cognitive experiments, it was confirmed that our adversarial examples are less perceived. However, our system has several limitations in attacking sentences or phrases and the inability to transmit over-the-air. We will continue to study audio adversarial attacks aimed at overcoming these shortcomings.
[47] P. Neekhara, S. Hussain, P. Pandey, S. Dubnov, J. McAuley, and F. Koushanfar, “Universal adversarial perturbations for speech recognition systems,” 2019, arXiv:1905.03828.

[48] L. Schönherr, T. Eisenhofer, S. Zeiler, T. Holz, and D. Kolossa, “Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems,” in Proc. Annu. Comput. Secur. Appl. Conf., Dec. 2020, pp. 5231–5240.

[49] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in Artificial Intelligence Safety and Security. London, U.K.: Chapman & Hall, 2018, pp. 99–112.

**HYUNJUN MUN** is currently pursuing the M.S. degree in computer and information security and convergence engineering for intelligent drones with Sejong University, Seoul, South Korea. His research interests include deep learning, adversarial example, and AI security.

**SUNGGWAN SEO** is currently pursuing the M.S. degree in computer and information security and convergence engineering for intelligent drones with Sejong University, Seoul, South Korea. His research interests include deep learning, adversarial example, digital forensics, and AI security.

**BAEHOON SON** is currently pursuing the M.S. degree in computer and information security and convergence engineering for intelligent drones with Sejong University, Seoul, South Korea. His research interests include deep learning, adversarial example, digital forensics, and AI security.

**JOOBEOM YUN** received the B.S. degree in computer science and engineering from Korea University, Seoul, South Korea, in 1999, the M.S. degree in computer engineering from Seoul National University, Seoul, in 2001, and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012. He is currently an Associate Professor with the Department of Computer and Information Security, Sejong University, Seoul. His research interests include software security, artificial intelligence (AI) security, and network security.

***