7. Appendix

Algorithm 1: Non-Dominating Sorting

Input: combined population P, objective vectors L

$\mathcal{F} \leftarrow \{\}$ // set of fronts
for $p \in P$ **do**
  $S_p \leftarrow \{\}$ //set of p dominated solutions
  $n_p \leftarrow 0$ // domination counter of p
  for $q \in P$ **do**
    if p *dominates* q **then**
      $S_p \leftarrow S_p \cup \{q\}$
    else if q *dominates* p **then**
      $n_p \leftarrow n_p + 1$
  if $n_p == 0$ **then**
    p_{rank} = 1 // p belongs to the first front
    $\mathcal{F}_1 \leftarrow \mathcal{F}_1 \cup \{p\}$
  $i \leftarrow 1$ // initialize front counter
  while $\mathcal{F}_i \neq \emptyset$ **do**
    $Q \leftarrow \emptyset$ // store solutions of the next front
    for $p \in \mathcal{F}_i$ **do**
      for $q \in S_p$ **do**
        $n_q \leftarrow n_q - 1$
        if $n_q == 0$ **then**
          // q belongs to the next front
          $q_{rank} \leftarrow i + 1$
          $Q \leftarrow Q \cup \{q\}$
    $i \leftarrow i + 1$
    $\mathcal{F}_i \leftarrow Q$
  return $\mathcal{F}$

Algorithm 1 described the non-dominated sorting method proposed by Deb et al. [13] for sorting solutions with multiple objectives. The method first determines solutions in the first front (not dominated by any solutions) then iteratively constructs the remaining fronts.

Algorithm 2: Sparse-RS p_m Selection Method

Input: conducted function evaluations i, budget N, initial mutation size α_{init}

$t \leftarrow \text{int}(\frac{i}{N} \cdot 10000)$
$c \leftarrow \{0, 50, 200, 500, 1000, 2000, 4000, 6000, 8000\}$
$j \leftarrow$ index of c that is closest to t
$\beta \leftarrow \{2, 4, 5, 6, 8, 10, 12, 15, 20\}$
return α_{init}/β_j

Algorithm 2 reduces the mutation size as the number of classifiers queries increases. The method linearly re-scales the current number of models i with the assumption of $N = 10000$.

Algorithm 3: Adapted Sparse-RS Attack

Input: objective vector F, input $\mathbf{x} \in \mathcal{X}$, sparsity k, zero-sampling pr_0, initial mutation size α_{init}, budget N

$M \leftarrow k$ random pixel indices to be perturbed
$\Delta \leftarrow$ values of the perturbation to be applied
$L \leftarrow F(\mathbf{x}; \{M, \Delta\})$
for $i \leftarrow 0; i < N; i + 1$ **do**
  $p_m \leftarrow selection(\alpha_{init})$ // refer to Algorithm 2
  $M', \Delta' \leftarrow mutation(\{M, \Delta\}, p_m)$
  $L' \leftarrow F(\mathbf{x}; \{M', \Delta'\})$
  if $\{M', \Delta'\}$ *dominates* $\{M, \Delta\}$ **then**
    $M \leftarrow M', \Delta \leftarrow \Delta', L \leftarrow L'$
return $\{M, \Delta\}$

Algorithm 3 outlines the Sparse-RS algorithm proposed by Croce at al. [10] adapted to the multi-objective scenario. *dominates* is corresponds to Definition 3.1.

Algorithm 4: SA-MOO Method

Input: objective vector F, input $\mathbf{x} \in \mathcal{X}$, query budget N, sparsity k, population size s, zero-sampling probability pr_0

// Initial Population
$P \leftarrow \{\{M_1, \Delta_1\}, \cdots, \{M_s, \Delta_s\}\}$
// Objective Evaluation
$L \leftarrow \{F(\mathbf{x}; \{M_1, \Delta_1\}), \cdots, F(\mathbf{x}; \{M_s, \Delta_s\})\}$
for $i \leftarrow 0;\ i < N;\ i \leftarrow i + s$ **do**
  // Uniformly Sample $s/2$ pairs of P indices
  $J \leftarrow \mathcal{U}(\{1, \cdots, s\})^{\frac{s}{2} \times 2}$
  $P_O \leftarrow \{\}$
  $L_O \leftarrow \{\}$
  for $j \in J$ **do**
    $O \leftarrow crossover(P_{j_0}, P_{j_1})$
    $M_1'', \Delta_1'' \leftarrow mutation(O_1)$
    $M_2'', \Delta_2'' \leftarrow mutation(O_2)$
    $P_O \leftarrow P_O \cup \{\{M_1'', \Delta_1\}, \{M_2'', \Delta_2\}\}$
    $L_O \leftarrow L_O \cup \{F(\mathbf{x}; \{M_1'', \Delta_1''\})\}$
    $L_O \leftarrow L_O \cup \{F(\mathbf{x}; \{M_2'', \Delta_2''\})\}$
  $P \leftarrow P \cup P_O$
  $L \leftarrow L \cup L_O$
  $P \leftarrow$ *non-dominated sorting*(P)
  $P \leftarrow P_{1:s}$ // Select lowest ranked solutions
  $L \leftarrow L_P$
return P // return population of solutions

	AT_1				AT_2			
Method	ASR	l_0	l_2	SSIM	ASR	l_0	l_2	SSIM
SA-MOO*	**84.40%**	**15.02**	**8.00**	**0.95**	**76.90%**	**15.28**	**8.54**	**0.95**
SA-MOO**	**84.40%**	**15.02**	**8.00**	**0.95**	**76.90%**	**15.28**	**8.54**	**0.95**
SA-MOO*	**44.20%**	18.39	11.93	**0.93**	**39.10%**	**18.81**	**12.90**	**0.93**
SA-MOO**	**44.20%**	**18.37**	**11.92**	**0.93**	**39.10%**	18.82	**12.90**	**0.93**

Table 4. Statistics of attack success rate, average ssim, and average l_0, l_2 distances of non-targeted (top) and targeted (bottom) attacks on the CIFAR-10 trained models AT_1 and AT_2. Where "SA-MOO" is the proposed method, ** refers to both *crossover* and *mutation* operators, * refers to only the *mutation operator.*

Algorithm 5: Crossover Operator

Input: pixel locations M_a, M_b, perturbation values Δ_a, Δ_b, crossover size p_c, sparsity k

$O \leftarrow \{\}$

for $r \in \{a, b\}$ *and* $e \in \{b, a\}$ **do**

  $U \leftarrow M_e \setminus (M_r \cap M_e)$

  $b \leftarrow \min\{p_c \cdot k, |U|\}$

  $A \leftarrow \mathcal{U}(M_r)^b$

  $B \leftarrow \mathcal{U}(U)^b$

  $M'_r \leftarrow (M_r \setminus A) \cup B$

  $\Delta'_r \leftarrow (\Delta_r \setminus \Delta_{r_A}) \cup \Delta_{e_B}$

  $O \leftarrow O \cup \{\{M'_r, \Delta'_r\}\}$

return O

Algorithm 6: Mutation Operator

Input: pixel locations M'_a, perturbation values Δ'_a, mutation size p_m, sparsity k, zero-sample probability pr_0

$U \leftarrow \{1, \cdots, h \cdot w\}$ // image height h and width w

$T \leftarrow U \setminus M'_a$

$A \leftarrow \mathcal{U}(M'_a)^{p_m \cdot k}$

$B \leftarrow \mathcal{U}(T)^{p_m \cdot k}$

$M''_a \leftarrow (M'_a \setminus A) \cup B$

// Sample with a zero-probability pr_0

$\Delta''_a \leftarrow (\Delta'_a \setminus \Delta'_{a_A}) \cup \mathcal{U}(\{-1, 0, 1\})^{(p_m \cdot k) \times 3}$

return $\{M''_a, \Delta''_a\}$