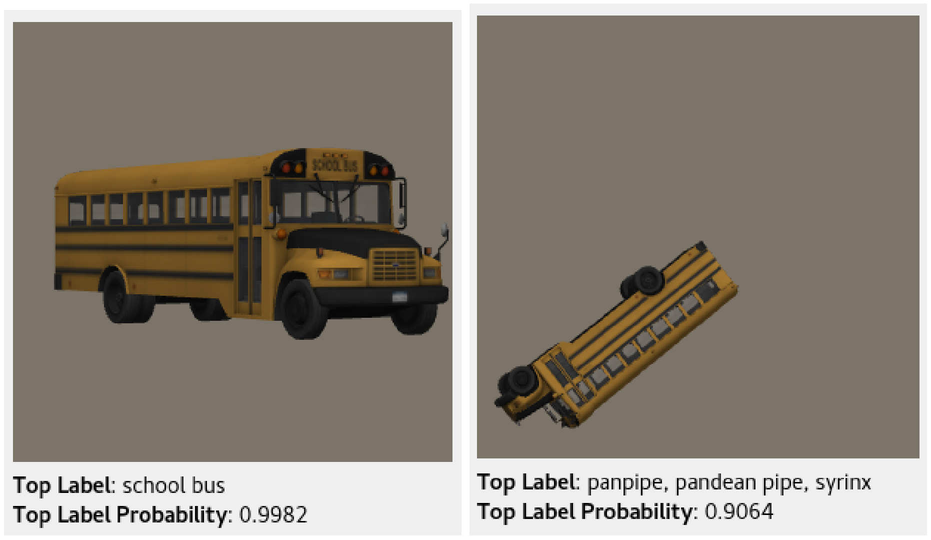
Using Covariance Matrix Adaptation Evolution Strategy to Discover Adversarial Poses for Artificial Neural Networks

Michael A. Alcorn Chinedu Eleh Qi Li

1 Introduction

A flurry of studies over the past several years have revealed the myriad of ways artificial neural networks are vulnerable to "adversarial examples" [1], i.e., examples that appear benign to a human but cause a neural network to fail. Typically, adversarial examples are generated by using gradient information from a target neural network to update the pixels of an input image. Essentially, this procedure adds human-imperceptible "noise" to an image in a way that causes a neural network to make a specific, incorrect prediction. Using 3D graphics, [2] showed neural networks are also easily confused by "natural adversarial" examples, i.e., unmanipulated images of the real world that are easily recognized by a human but cause a neural network to fail (e.g., a flipped over school bus; see Fig. 1).

Because the rendering process is non-differentiable, black box optimization techniques are necessary for discovering adversarial poses of 3D objects. [2] used finite differences (FD) to approximate the partial derivatives of a neural network's loss with respect to 3D pose parameters, and showed that gradient descent using these approximated partial derivatives could more effectively generate targeted adversarial poses than an informed random search approach. However, the pose loss landscape appears to be highly nonconvex and may possess many plateaus and saddle points (as is often the case with neural networks [4]), so derivative-free techniques may be better suited for this task. Evolutionary algorithms are a class of derivative-free techniques inspired by evolutionary biology that have been successfully used in a number of applications. Broadly, evolutionary algorithms consist of "populations" of candidate solutions, a "mutation" mechanism (i.e., a mechanism for


Top Label: school bus
Top Label Probability: 0.9982
Top Label: panpipe, pandean pipe, syrinx
Top Label Probability: 0.9064

(a) (b)

Figure 1: The Google Inception-v3 neural network [3] correctly recognizes a 3D object of a school bus in a "canonical pose" (a) but is confused by the same object in an adversarial pose (b).

randomly changing parameter values), a notion of "fitness" (typically defined in relation to a cost function), and a "natural selection" process (i.e., a way for "fitter" candidate solutions to reproduce and "weaker" candidate solutions to be removed). In this report, we compare the effectiveness of covariance matrix adaptation evolution strategy (CMA-ES) [5, 6], an evolutionary algorithm that has shown promise in reinforcement learning settings [7, 8], to FD for adversarial pose discovery.

2 Background

2.1 Formal problem statement

Much of the following optimization problem description is reproduced from [2]. Let f be an image classifier that maps an image $\mathbf{q} \in \mathbb{R}^{H \times W \times C}$ onto a softmax probability distribution over 1,000 output classes. Let R be a 3D renderer that takes as input a set of parameters ϕ and outputs a render, i.e., a 2D image $R(\phi) \in \mathbb{R}^{H \times W \times C}$. Typically, ϕ is factored into mesh vertices V, texture images T, a background image B, camera parameters C, and lighting parameters L, i.e., $\phi =$

$\{V, T, B, C, L\}$. To change the 6D pose of a given 3D object, a 3D rotation and 3D translation is applied to the original vertices V yielding a new set of vertices $\hat{V}$. Specifically, to rotate an object with geometry defined by a set of vertices $V = \{v_i\}$, the linear transformation in (1) is applied to each vertex $v_i \in \mathbb{R}^3$:

$$v_i^R = R_y R_p R_r v_i \tag{1}$$

where R_y, R_p, and R_r are the 3×3 rotation matrices for yaw, pitch, and roll, respectively. The rotated object is then translated by adding a vector $T = \begin{bmatrix} x_\delta & y_\delta & z_\delta \end{bmatrix}^\top$ to each vertex:

$$v_i^{R,T} = T + v_i^R \tag{2}$$

Here, we wish to estimate only the pose transformation parameters x (while keeping all parameters in ϕ fixed) such that the rendered image $R(x; \phi)$ causes the classifier f to assign the highest probability (among all outputs) to an incorrect target output at index t. Formally, we attempt to solve the below optimization problem:

$$x^* = \arg\max_x (f_t(R(x, \phi))) \tag{3}$$

In practice, we minimize the cross-entropy loss $\mathcal{L}$ for the target class. Because radians have a circular topology (i.e., a rotation of 0 radians is the same as a rotation of 2π radians, 4π radians, etc.), we parameterized each rotation angle θ_i as $(\cos(\theta_i), \sin(\theta_i))$—a technique commonly used for pose estimation and inverse kinematics—which maps the Cartesian plane to angles via the $atan2$ function. Therefore, we optimized in a space of $3 + 2 \times 3 = 9$ parameters.

2.2 Finite differences baseline

As our baseline, we used the FD gradient descent approach described in [2]. Specifically, the partial derivative of the loss function with respect to the various pose parameters was approximated as:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\mathcal{L}(x_i + \frac{h}{2}) - \mathcal{L}(x_i - \frac{h}{2})}{h} \tag{4}$$

The approximate gradient $\nabla\mathcal{L}$ obtained from (4) served as the gradient in our gradient descent optimization procedure. We used the vanilla gradient descent assignment:

$$x \coloneqq x - \gamma \nabla \mathcal{L}(x) \tag{5}$$

to update the parameters.

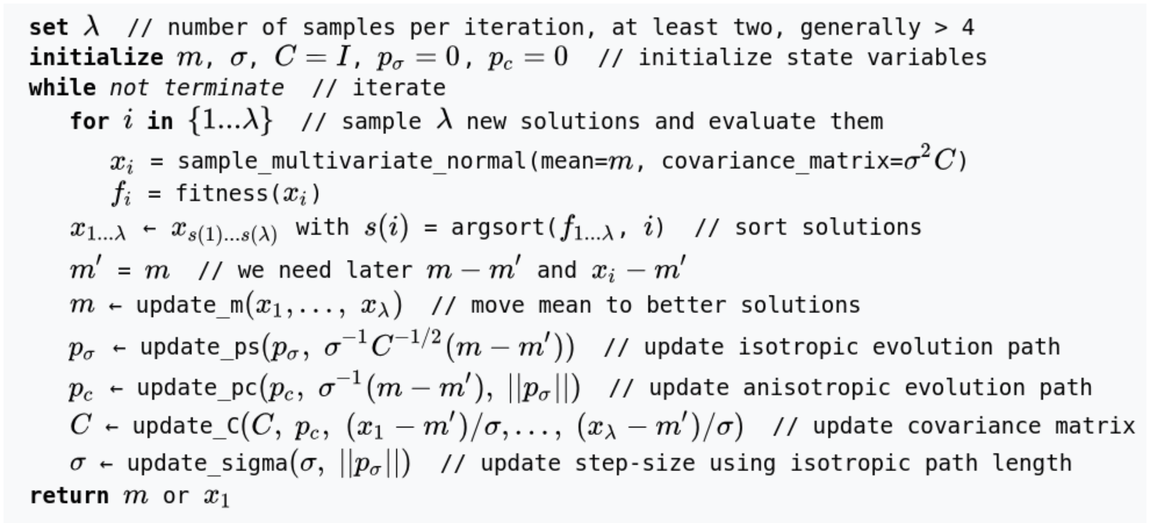
3 Covariance matrix adaptation evolution strategy

In CMA-ES, candidate solutions are sampled from a multivariate normal distribution where the mean vector m, covariance matrix C, and a scalar σ (which scales C and is referred to as the "step size" or "path length control") are updated through an evolutionary process (see pseudocode in Fig. 2). CMA-ES also maintains two vectors, p_c, referred to as the "anisotropic evolution path", and p_σ, referred to as the "isotropic evolution path", which contain information on the history of updates. In the following sections, we provide detailed descriptions of how m, C, and σ are updated and further explain the roles of p_c and p_σ.

3.1 Updating m

Like all evolutionary algorithms, the first step of CMA-ES is to generate a population of individuals. To do so, CMA-ES samples λ candidate solutions (also referred to as "children") from the multivariate normal distribution specified by the values of m, σ, and C for the current generation g, i.e.:

$$x \ \sim \ \mathcal{N}(m^{(g)}, (\sigma^{(g)})^2 C^{(g)})$$



```
set $\lambda$  // number of samples per iteration, at least two, generally > 4
initialize $m$, $\sigma$, $C = I$, $p_\sigma = 0$, $p_c = 0$  // initialize state variables
while not terminate  // iterate
   for $i$ in $\{1...\lambda\}$  // sample $\lambda$ new solutions and evaluate them
      $x_i$ = sample_multivariate_normal(mean=$m$, covariance_matrix=$\sigma^2 C$)
      $f_i$ = fitness($x_i$)
   $x_{1...\lambda}$ ← $x_{s(1)...s(\lambda)}$ with $s(i)$ = argsort($f_{1...\lambda}$, $i$)  // sort solutions
   $m'$ = $m$  // we need later $m - m'$ and $x_i - m'$
   $m$ ← update_m($x_1, \ldots, x_\lambda$)  // move mean to better solutions
   $p_\sigma$ ← update_ps($p_\sigma$, $\sigma^{-1} C^{-1/2}(m - m')$)  // update isotropic evolution path
   $p_c$ ← update_pc($p_c$, $\sigma^{-1}(m - m')$, $||p_\sigma||$)  // update anisotropic evolution path
   $C$ ← update_C($C$, $p_c$, $(x_1 - m')/\sigma, \ldots, (x_\lambda - m')/\sigma$)  // update covariance matrix
   $\sigma$ ← update_sigma($\sigma$, $||p_\sigma||$)  // update step-size using isotropic path length
return $m$ or $x_1$
```

Figure 2: Pseudocode for the CMA-ES algorithm (image source: Wikipedia).

Each x[1] is then assigned a fitness score[2]. To update the population mean vector m, CMA-ES simply takes a weighted average of the top μ individuals (also referred to as "parents") as determined by the fitness function:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i \, x_{i:\lambda}^{(g)} \tag{6}$$

which is equivalent to:

$$m^{(g+1)} = m^{(g)} + \sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(g)} \tag{7}$$

where $w_1 \geq w_2 \geq \cdots \geq w_\mu > 0$ sum to one, $y_{i:\lambda}^{(g)} = x_{i:\lambda}^{(g)} - m^{(g)}$ (i.e., a step direction), and $x_{i:\lambda}^{(g)}$ is the ith ranked candidate solution.

[1]In our case, x contains the pose parameters.
[2]In our case, the fitness function is derived from the neural network loss.

3.2 Updating C

Suppose at generation $g = 0, 1, 2, \cdots$ we obtain the vectors $x_1^{(g)},\ x_2^{(g)}, \cdots,\ x_\lambda^{(g)}$. The empirical covariance matrix $C^{(g)}$ can be defined in terms of rank-one matrices:

$$C^{(g)} = \frac{1}{\lambda - 1} \sum_{i=1}^{\lambda} \left(x_i^{(g)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} x_j^{(g)} \right) \left(x_i^{(g)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} x_j^{(g)} \right)^T \tag{8}$$

where $\frac{1}{\lambda} \sum_{j=1}^{\lambda} x_j^{(g)} = m^{(g)}$ is the mean of the random vectors. By using only the rank-one matrices from the fittest individuals, e.g.:

$$C_\mu^{(g+1)} = \sum_{i=1}^{\mu} w_i \left(x_{i:\lambda}^{(g+1)} - m^{(g)} \right) \left(x_{i:\lambda}^{(g+1)} - m^{(g)} \right)^T \tag{9}$$

we may be able to produce a covariance matrix that generates better candidate solutions in the subsequent generation.

Notably, substituting the point $m^{(g)}$ in (9) with a point $m^{(g+1)}$ from the convex hull of $m^{(g)}$ and the mean of the μ "fittest" individuals produces the Estimation of Multivariate Normal Algorithm (EMNA) covariance matrix:

$$C_{\text{EMNA}}^{(g+1)} = \sum_{i=1}^{\mu} w_i \left(x_{i:\lambda}^{(g+1)} - m^{(g+1)} \right) \left(x_{i:\lambda}^{(g+1)} - m^{(g+1)} \right)^T \tag{10}$$

For the simple case where $w_i = \frac{1}{\mu}$, (10) reduces to

$$C_{\text{EMNA}}^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \left(x_{i:\lambda}^{(g+1)} - m^{(g+1)} \right) \left(x_{i:\lambda}^{(g+1)} - m^{(g+1)} \right)^T \tag{11}$$

Further, multiplying (11) by $\frac{\mu}{\mu-1}$ gives the unbiased estimator of the covariance matrix for the parent vectors. Algorithms using $\frac{\mu}{\mu-1} C_{\text{EMNA}}^{(g+1)}$ can converge very fast; however, in cases where the initial sampling does not bracket the optimum (which is likely for a small μ), convergence is often premature [9], which is why CMA-ES uses $m^{(g)}$ instead (see Fig. 3).

An alternative way to update the covariance matrix is to average all of the covariance matrices given in (9) from previous generations, i.e.:

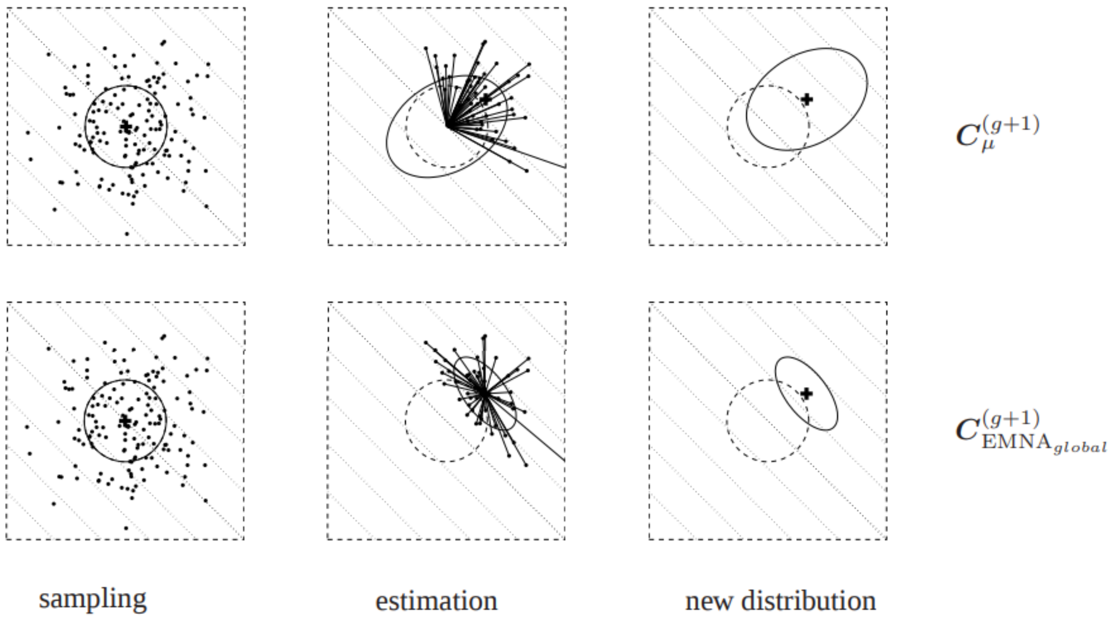


Figure 3: The distribution of candidate solutions when using the covariance matrix update from (9) (top row) or (11) (bottom row). Notice how (11) *decreases* the variance of candidate solutions in the direction that improves the cost function (image source: [5]).

$$C^{(g+1)} = \frac{1}{g+1} \sum_{i=0}^{g} \frac{1}{(\sigma^{(i)})^2} C_\mu^{(i+1)}$$

where $(\sigma^{(i)})^2$ is incorporated to make $C_\mu^{(i+1)}$ comparable for each i. However, this weighting scheme may place too much emphasis on covariance matrices from the distant past. To address this potential shortcoming, CMA-ES uses an exponential moving average update:

$$C^{(g+1)} = (1 - c_\mu) C^{(g)} + c_\mu \frac{1}{(\sigma^{(i)})^2} C_\mu^{(g+1)}. \tag{12}$$

where $c_\mu \in [0, 1]$ is a learning rate specifying how much information is retained versus learned between generations[3]. Intuitively, (12), which is referred to as the "rank-μ update", increases the likelihood of successful steps by following the *natural* gradient of the expected fitness [5, 9].

[3]For this reason, $\frac{1}{c_\mu}$ is known as the "backward time horizon".

Still another way to update C is by omitting explicit covariance matrix estimates altogether and instead relying on information contained in the history of changes to m, i.e., its "evolution path". The evolution path p_c is also updated as an exponential moving average:

$$p_c^{(g+1)} = (1 - c_c)p_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\,\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \tag{13}$$

where c_c is a learning rate and $\sqrt{c_c(2 - c_c)\mu_{\text{eff}}}$ is a carefully chosen normalization constant that ensures $p_c^{(g+1)} \sim \mathcal{N}(0, C^{(g)})$ under random selection.[4] $C^{(g)}$ can then be updated with:

$$C^{(g+1)} = (1 - c_1)C^{(g)} + c_1 p_c^{(g+1)} p_c^{(g+1)^T} \tag{14}$$

where c_1 is a learning rate. (14) is referred to as the "rank-one update" and contains information on correlations between generations, which makes it useful when dealing with small populations [9]. CMA-ES integrates both the population-level information contained in the rank-μ update (12) and the intergenerational information contained in the rank-one update (14) into a single update for C:

$$C^{(g+1)} = (1 - c_1 - c_\mu)C^{(g)} + c_1 p_c^{(g+1)} p_c^{(g+1)^T} + c_\mu \sum_{i=1}^{\mu} w_i \left(x_{i:\lambda}^{(g+1)} - m^{(g)}\right)\left(x_{i:\lambda}^{(g+1)} - m^{(g)}\right)^T \tag{15}$$

3.3 Updating σ

While the covariance matrix update controls the distribution of future step *directions*, it does not allow for explicit control of future step *sizes*. [5] provides two motivations for including step-size adaptation in the CMA-ES algorithm:

1. The covariance matrix update cannot approximate the optimal step size well.
2. The best learning rate for updating the covariance matrix is too slow for changing the overall step size effectively.

To update σ, CMA-ES uses what is referred to as "cumulative step length adaption" (CSA).

[4]See [5] for a detailed explanation.

Intuitively, if the distance of an evolution path from some starting point is short, CMA-ES must be backtracking (see Fig. 4), so the step size should be decreased. Alternatively, if the evolution path is long, CMA-ES must be consistently moving in the same direction, so the step size should be increased.[5] Lastly, if the evolution path is neither short nor long (i.e., when steps tend to be perpendicular), then CMA-ES is moving as fast as it can, so the step size should remain the same.

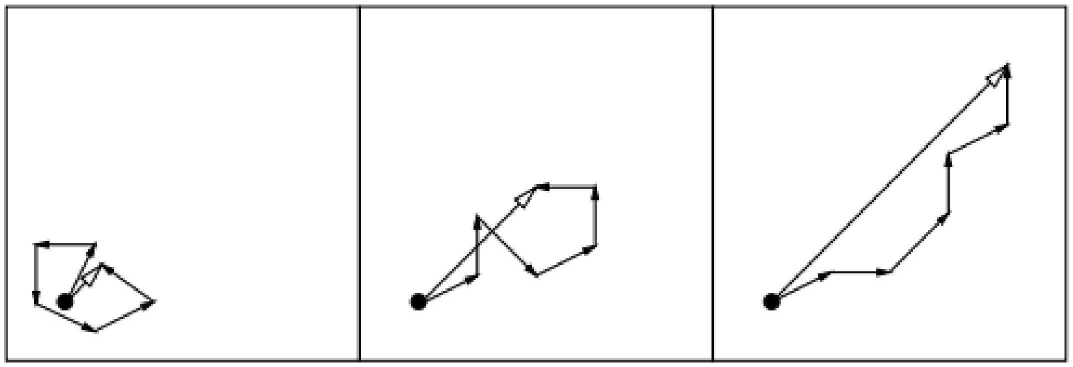


Figure 4: Example evolution paths of six steps (image source: [5]).

To determine whether the length of an evolution path is "short" or "long", we can compare the path's length with its expected length under random selection. Given a vector $p_\sigma^{(g+1)}$ summarizing an evolution path, CMA-ES uses the following update rule for σ:

$$\sigma^{(g+1)} = \sigma^{(g)} \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(g+1)}\|}{\mathbb{E}\|\mathcal{N}(0,I)\|} - 1\right)\right) \tag{16}$$

where c_σ^{-1} is the backward time horizon for σ, d_σ is a damping parameter, and $\mathbb{E}\|\mathcal{N}(0,I)\|$ is the average Euclidean norm for $\mathcal{N}(0,I)$ distributed vectors. When $\|p_\sigma^{(g+1)}\| \approx \mathbb{E}\|\mathcal{N}(0,I)\|$, the exponent in (16) is approximately zero and the step size remains approximately the same. Similarly, longer evolution paths will increase the step size while shorter evolution paths will decrease the step size.

Importantly, the length of p_σ is direction agnostic (which is not the case for p_c), which is why it is referred to as the "*conjugate* evolution path" or "*isotropic* evolution path". The update rule

[5]The conceptual similarities between CSA and methods like momentum should be apparent [9].

for p_σ:

$$p_\sigma^{(g+1)} = (1 - c_\sigma)\, p_\sigma^{(g)} + \sqrt{c_\sigma\,(2 - c_\sigma)\,\mu_{\text{eff}}}(C^{(g)})^{-\frac{1}{2}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \tag{17}$$

is nearly identical to (13) except for the inclusion of $(C^{(g)})^{-\frac{1}{2}}$, the square root of the inverse of $C^{(g)}$. $(C^{(g)})^{-\frac{1}{2}}$ makes the expected length of $p_\sigma^{(g+1)}$ independent of its direction and ensures $p_\sigma^{(g+1)} \sim \mathcal{N}(0, I)$ under random selection.[6]

4 Experiments and results

We compared the effectiveness of CMA-ES to FD for adversarial pose discovery by generating targeted adversarial poses for a 3D school bus object. We used the `pycma`[7] Python package for our CMA-ES implementation, and our FD implementation followed [2]. The code for our experiments can be found at: https://github.com/airalcorn2/strike-with-a-pose/blob/master/paper_code/optimizer_example.py. Because the FD approach used by [2] performed 18 forward passes through the neural network per iteration[8], we used $\lambda = 18$ for our CMA-ES experiments to make the computational demands comparable between the two algorithms (the vast majority of the computation for each iteration occurs in the forward pass).

For each optimization method, we ran 20 trials targeting each of the 20 most common classifications for the school bus object[9]. For each (target, trial) pair, we generated a starting point using the z_δ-constrained random search procedure described in [2], and we used this *same* starting point to initialize both FD and CMA-ES runs. For each (method, target, trial) triple, we recorded both the best target probability $p_{o,t,l}$[10] discovered by the method and whether the target misclassification occurred. Each experiment was allowed to run for 100 iterations.

In 94% of the trials, CMA-ES discovered a higher target probability than FD (see Fig. 5). Following [2], we present "median of medians" statistics where we first take the median of quantities

[6]See [5] for a detailed explanation.
[7]https://github.com/CMA-ES/pycma
[8]two forward passes for each of the nine pose parameters
[9]for a total of $2 \times 20 \times 20 = 800$ experiments
[10]o denotes the optimization method, t denotes the target, and l denotes the trial number

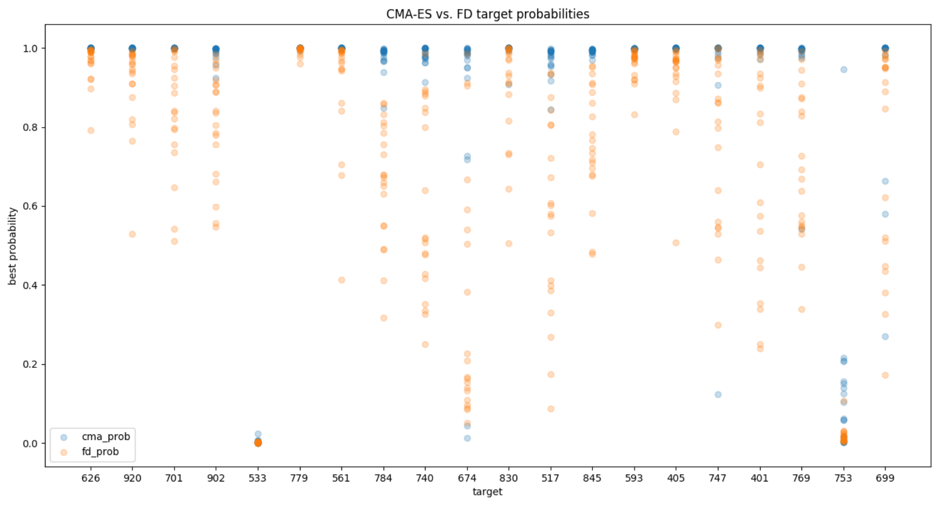


Figure 5: Target probabilities for the CMA-ES and FD algorithms. Each point represents a different trial.

of interest for a specific object, and then take the median of these median values across objects. For each (o, t) pair, we calculated the median $p_{o,t,l}$ across all trials and denote it $p_{o,t}$. We then calculated the difference of these median values between the optimization methods $p_t = p_{e,t} - p_{f,t}$ where e indicates CMA-ES and f indicates FD. The median p_t was 0.12, i.e., the median $p_{e,t}$ was often at least 0.12 higher than the median $p_{f,t}$. In fact, for all 20 targets, the median $p_{e,t}$ was greater than the median $p_{f,t}$. For 18 of the 20 target classes, there was no difference in the number of trials leading to a "hit" between the two methods. However, for two targets, CMA-ES had two and 11 more hits than FD. Combined, these results provide strong evidence that CMA-ES is much better suited for the adversarial pose discovery task than FD.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *International Conference on Learning Representations*, 12 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[2] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W.-S. Ku, and A. Nguyen, "Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects," *Conference on Computer Vision and Pattern Recognition*, 11 2019. [Online]. Available: http://arxiv.org/abs/1811.11553

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Conference on Computer Vision and Pattern Recognition*, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdfhttp://arxiv.org/abs/1512.00567

[4] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *Neural Information Processing Systems*, pp. 2933–2941, 2014. [Online]. Available: https://papers.nips.cc/paper/5486-identifying-and-attacking-the-saddle-point-problem-in-high-dimensional-non-convex-optimization

[5] N. Hansen, "The CMA Evolution Strategy: A Tutorial," 4 2016. [Online]. Available: http://arxiv.org/abs/1604.00772

[6] D. Ha, "A Visual Guide to Evolution Strategies," 2017. [Online]. Available: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

[7] D. Ha and J. Schmidhuber, "Recurrent World Models Facilitate Policy Evolution," *Neural Information Processing Systems*, pp. 2450–2462, 2018. [Online]. Available: https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution

[8] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," 3 2017. [Online]. Available: http://arxiv.org/abs/1703.03864

[9] N. Hansen, "Covariance Matrix Adaptation Evolution Strategy (CMA-ES)," Tech. Rep. [Online]. Available: http://www.cmap.polytechnique.fr/~nikolaus.hansen/copenhagen-cma-es.pdf