High-performance Evolutionary Algorithms for Online Neuron Control

Binxu Wang
Harvard Medical School
Boston, MA, USA
binxu_wang@hms.harvard.edu

Carlos R. Ponce
Harvard Medical School
Boston, MA, USA
carlos@hms.harvard.edu

Figure 1: Cholsky-CMAES Excelled in Activation Maximization both in silico and in vivo. A. Schematics of the XDream Evolution experiment. B. in vivo optimization trajectory from 12 paired Evolution experiments in two monkeys. Thin curves show the trajectories for individual experiments; shaded thick curves show the mean and standard error (SEM) of trajectories across experiments. Experiments that terminated earlier were extrapolated by constant (dashed line) to match the generation number for mean and SEM calculation. C. in silico optimization trajectory comparison for unit 1 in fc6 layer of CaffeNet, mean activation per generation is plotted.

ABSTRACT
Recently, optimization has become an emerging tool for neuroscientists to study neural code. In the visual system, neurons respond to images with graded and noisy responses. Image patterns eliciting highest responses are diagnostic of the coding content of the neuron. To find these patterns, we have used black-box optimizers to search a 4096d image space, leading to the evolution of images that maximize neuronal responses. Although genetic algorithm (GA) has been commonly used, there haven’t been any systematic investigations to reveal the best performing optimizer or the underlying principles necessary to improve them.

Here, we conducted a large scale in silico benchmark of optimizers for activation maximization and found that Covariance Matrix Adaptation (CMA) excelled in its achieved activation. We compared CMA against GA and found that CMA surpassed the maximal activation of GA by 66% in silico and 44% in vivo. We analyzed the structure of Evolution trajectories and found that the key to success was not covariance matrix adaptation, but local search towards informative dimensions and an effective step size decay. Guided by these principles and the geometry of the image manifold, we developed SphereCMA optimizer which competed well against CMA, proving the validity of the identified principles.

CCS CONCEPTS
• Mathematics of computing → Genetic programming; • Applied computing → Systems biology; • Computing methodologies → Neural networks.

KEYWORDS
visual neuroscience, differential geometry, activation maximization

1 INTRODUCTION
An essential goal in sensory neuroscience is to define how the neurons respond to natural stimuli and extract useful information to guide behavior. To a first approximation, visual neurons emit high rates of electrical signals to stimuli with certain visual attributes, so their outputs can be interpreted as conveying the presence of such features (e.g. “face neurons”[15, 25]). So to study the visual selectivity of neurons, it is crucial to choose highly activating stimuli.

Traditionally, researchers have used intuition[10] or limited theoretical frameworks to choose a fixed set of stimuli, i.e., simple images, such as circles and rings for studying lateral geniculate

This work is licensed under a Creative Commons Attribution International 4.0 License.

GECCO ’22, July 9–13, 2022, Boston, MA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9237-2/22/07.
https://doi.org/10.1145/3512290.3528725
nucleus cells, oriented bars for V1 neurons, hyperbolic gratings for V2 neurons [14], curved shapes for V4 neurons [23], or select categories such as faces for inferotemporal cortex neurons [7]. The desired property of these stimuli is their ability to drive neuronal activity. But as visual neurons become more selective along the posterior-anterior anatomical axis — responding to more complex visual attribute combinations — it becomes more difficult to choose highly activating stimuli.

To tackle this problem, an alternative approach is to use adaptive-search methods to find highly activating stimuli. The idea is to treat neuronal responses as a function of visual stimuli, and to iteratively search for variants that maximize this function. As images evolve under this search, they acquire visual attributes informative of the neuron’s intrinsic tuning, independent of bias in human intuition.

**Problem Formulation.** Formally, neurons can be conceptualized as noisy scalar functions \( f \) over the image space \( \mathcal{I} \). One common objective is to maximize the function \( f(I) \) — also known as activation maximization, commonly used for interpreting the coding of visual attribute combinations — it becomes more difficult to choose highly activating stimuli.

For multiple

\[
E(z) = \arg \max_{I} f(I) \tag{1}
\]

While this approach can be easily generalized to other objectives, we focus on activation maximization since making neurons fire highly activating stimuli. Usually, activation maximization, commonly used for interpreting the coding of visual attribute combinations — it becomes more difficult to choose highly activating stimuli.

Formally, neurons can be conceptualized as noisy scalar functions \( f \) over the image space \( \mathcal{I} \). One common objective is to maximize the function \( f(I) \) — also known as activation maximization, commonly used for interpreting the coding content of neurons or CNN hidden units [22]:

\[
r = f(I) + \epsilon, f : \mathcal{I} \to \mathbb{R} \tag{1}
\]

\[
l^* = \arg\max_{I} f(I) \tag{2}
\]

While this approach can be easily generalized to other objectives, we focus on activation maximization since making neurons fire strongly has been the most common goal in sensory neuroscience studies over the decades [15].

In the context of *in vivo* experiments, the process amounts to recording a set of neuronal activations \( \{r_i\} \) in response to a set of randomly sampled images \( \{i_i\} \) (each image displayed for a short duration, e.g. 100 ms, Fig. 1A). In the next step, the optimizers update their state and propose a new set of images. By repeating this process for dozens of iterations, the images begin to acquire visual attributes that drives the neuron’s activity highly. Usually, in one of these so-called Evolution experiments, the number of image presentations is limited to 1000-3000, taking 20-40 min. This mandates a high sample efficiency of the optimizer.

For most optimization algorithms, stimuli need to be represented in and generated from a “vector space”. We consider a smooth parametrization of images \( G : \mathbb{R}^d \to \mathcal{I}, z \mapsto I \) using a lower-dimensional vector space. In this implementation, the mapping from vector space to images is instantiated by an image generative neural network (DeePSim Generative Adversarial Network (GAN) [8]), which maps 4096d latent vectors to images. Thus the problem is about searching for images in the latent space such that it maximizes the response of the neuron.

\[
z^* = \arg\max_z \mathbb{E}[f(G(z))] \tag{3}
\]

This optimization approach has been applied to neurons in visual areas V4 and inferotemporal cortex (IT) [4, 16, 24, 28, 30, 33]. Image search was effectuated by classic genetic algorithms (GAs), acting in the space of parametrized 3D shapes or GANs. Though the use of GAs was successful in this domain [32], it has not been tested comprehensively against modern optimizers, which motivated us to determine if we could improve performance on this front.

This problem features a unique set of challenges, for example, search dimensionality is very high (e.g. \( d = 4096 \) in [24, 28, 32]), and the function \( f(.) \) must be evaluated with considerable noise in neuronal responses. Further, the total number of function evaluations \( N_f \) is highly limited (\( N_f < d \)), thus the dimensions could not be exhaustively explored.

In this project, we worked to improve the performance of optimizer in this domain and to extract the underlying principles for designing such optimizer. The main contributions are as follows:

- We conducted two *in silico* benchmarks, establishing the better performance of CMA-type optimizers over other optimizers including commonly used genetic algorithms (GA).
- We validated the performance increase of the CholeskyCMA algorithm, with a focused contrast to classic GA in vivo.
- We found that the CMA search trajectories exhibited the spectral signature of high-dimensional guided random walks, preferentially traveling along the top eigen-dimensions of the underlying image manifold.
- We found one reason that CMA succeeded was the decrease of angular step size, thanks to the spherical geometry of image space and the increased vector norms in Evolution.
- Guided by image space geometry, we built in these mechanisms to develop a SphereCMA algorithm, which outperformed the original CMA in *silico*.

## 2 Survey of Black Box Optimizers

Because *in vivo* tests of optimizer performance can be costly and time-consuming, we began by screening a large set of algorithms *in silico* using convolutional neural network (CNN) units as proxies for visual neurons, then tested the top performing algorithms with actual neurons in a focused comparison.

### 2.1 Large Scale *in silico* Survey

To simulate neuronal tuning function \( f \) that an optimizer might encounter in a biological setting, we used units from pre-trained CNNs as models of visual neurons [19, 32]. For the *in vivo* Evolution experiments, we aimed for optimizers that performed well with neurons across visual areas (including V1, V4, IT) and across different levels of signal-to-noise and single-neuron isolation. Thus we designed the benchmark ‘problem set’ to include units from multiple CNN architectures, layers, and noise levels, testing the overall performance for each optimizer.

We chose AlexNet and a deeper, adversarially trained ResNet (ResNet50-robust) as models of ventral visual cortex. The latter was chosen because it exhibits visual representations similar to the brain (high rank on Brain-Score [29]). For each network, we picked 5 layers of different depths. It has been noted that units from shallow-to-deep layers prefer features of increasing complexity [22], similar to that in the ventral stream cortical hierarchy [28]. For detailed information about these networks and their layers, see Sec. A.1. In the context of *in vivo* recordings, single-presentation neuronal responses are highly noisy [5]. To simulate the Poisson-like noisy response \( r \), we added Gaussian noise with zero mean and standard deviation proportional to the raw response \( ar \) (ratio \( a \) represented the noise level). We tested three noise levels for each objective function: no noise \( (a = 0) \), low noise \( (a = 0.2) \), and high noise \( (a = 0.5) \).

\[
r = \max(0, (1 + a\epsilon) r), \quad \epsilon \sim \mathcal{N}(0, 1) \tag{3}
\]
In the first round, we chose 12 gradient-free optimizers as implemented or interfaced by nevergrad[27]: NGOpt, DE, TwoPointsDE, ES, CMA[13], DiagonalCMA[1], RescaledCMA, SQPCMA, PSO, OnePlusOne, TBPSA, and RandomSearch. Here we compared these algorithms by their default hyper-parameter settings. Note that RandomSearch just sampled random vectors from an isotropic Gaussian distribution, finding the vector with the highest score, which formed the naive baseline (for a short introduction to these algorithms, see [27]). Each algorithm ran with a budget of 3000 function evaluations and three repetitions per condition.

Among the 12 optimizers, we found that Covariance Matrix Adaptation Evolution Strategy (CMA) and DiagonalCMA were the top two algorithms in achieving maximal activation (Fig. 2). Since the upper bound of the activation of a unit in CNN was arbitrary, we divided the raw activations by the empirical maximal activation achieved by that given unit, across all algorithms and repetitions. By this measure, when pooling all conditions, CMAES achieved $6.299 \pm 0.004$, DiagonalCMA achieved $6.677 \pm 0.004$, as a reference Random Search baseline achieved $0.139 \pm 0.002$ (mean $\pm$ std; $N = 1500$, Tab. A.2, Fig. 2). This difference of CMA-driven performance vs. any other optimizer was significant per a two-sample t-test: $t > 60$ for all comparisons, except for CMA vs DiagonalCMA, where $t = 3.80, p = 1.4 \times 10^{-4}$. We found the same result held consistently for units across CNN models, layers and noise levels (see comparison in Tab. A.2). The optimization score traces and the most activating images found for an example ResNet unit are shown in Fig. A.1 and Fig. A.3.

As for time efficiency, we measured the total run time taken by optimizing the objective function with a budget of 3000 evaluations. With units from AlexNet as objective, CMA had a longer runtime of $104.5 \pm 30.0$ s (mean $\pm$ std); DiagonalCMA accelerated the runtime by roughly five-fold ($23.0 \pm 4.5$ sec), although at a slightly compromised score (6.1%). As a reference, the baseline algorithm RandomSearch had an average runtime of $17.6 \pm 2.2$ sec. Same trends held for ResNet50 units, though the runtime values were generally longer because ResNet50 is deeper (Tab. A.2). In conclusion, we found CMA and DiagonalCMA algorithms were both well-suited for this domain, while DiagonalCMA achieved a good trade-off between speed and performance.

2.2 Comparison of CMA-type Algorithm with GA in silico

Given the general success of CMA and DiagonalCMA algorithm, we were motivated to test other types of CMA algorithms in the second round, comparing runtime values and achieved activations.

As described in [11], the CMA algorithm maintains and updates a Gaussian distribution $N(m, \sigma C)$ in the $\mathbb{R}^d$ space, with the mean vector $m$ and step size $\sigma$ initialized by the user; the covariance matrix $C$ is initialized as identity matrix $I$. In each step, the algorithm samples a population of codes from this exploration distribution, $z_1 \sim N(m, \sigma C)$. After evaluating these codes, it updates the mean vector $m_t$ by a weighted combination of the highest ranking codes. Note that $\sigma_t$ controls both the spread of samples in a generation and the average step size of the mean vector update.

These optimizers were tested on single core machine with V100 GPU without batch processing of images or activations.

Figure 2: CMAES Excelled in Large Scale Benchmark of Activation Maximization. Normalized activation were pooled across all conditions. Mean runtime across all condition were annotated below each optimizer. The activation maximizing image found by the optimizers for an CNN unit (AlexNet-fc8-unit003 tiger shark unit) were showed above the scores.

$$\|m_t - m_2\|.$$ This step size $\sigma_t$ is adapted based on the accumulated path length in the last few steps. Moreover, CMA updates the covariance matrix $C_t$ by a few low-rank matrices to adapt the shape of the exploration distribution.

In the original CMAES algorithm, after each covariance update, the covariance matrix $C_t$ needed to be eigen-decomposed, in order to get $C^{-1}$. For a high dimensional space $d = 4096$, it is costly to compute this decomposition at each update. Using the diagonal covariance matrix approximation, the inversion step could be simplified from $O(n^2)$ operation to $O(n)$ operation, which makes the DiagonalCMA [1] much faster (Fig. 2). This inspired us to use modified covariance update rules to accelerate the optimizer.

We found Cholesky-CMA-ES [21] which was proposed as a large scale variant of CMAES. By storing the Cholesky factor $A$ and its inverse of the covariance matrix, it could update these factors directly, without factorizing the covariance matrix at each update. An additional parameter $\text{update frequency}$ could be adjusted to tune the frequency of this update.

We implemented the $(\mu/\mu_\text{pop}, \lambda)$-Cholesky-CMA-ES algorithm (CholeskyCMA) [21] and compared it against the CMA and DiagonalCMA implemented in pycma library [1, 12] and the Genetic Algorithm (GA), classically used in this domain [24, 30]. For Cholesky-CMA, the hyper-parameters $\sigma_P$ and update frequency of Cholesky factor $\lambda$ were tuned and fixed $\sigma_P = 3.0$. $\text{update frequency} = 10$. For GA, we used the code and hyperparameters from [24]. We allowed for 3000 function evaluations per optimizer, which was 75 generations with a population size of 40.

We slightly modified the in silico benchmark: we chose 7 layers from AlexNet (conv2 to fc8), 10 units from each layer, with 3 noise levels ($\alpha = 0.0, 0.2, 0.5$). Each optimizer were run with 5 repetitions
in each condition, totaling 1050 runs. We evaluated the clean score, i.e. the highest noise-free activation of the given unit and the runtime for each optimizer. Here we also used the empirical maximal clean score achieved for each unit to get the normalized clean score and to calculate statistics.

The results were summarized in Table 1 and Figures A.4,A.5. We found that all CMA algorithms outperformed Genetic Algorithm by a large margin: if we pooled all conditions, the mean normalized score of CMA was 166.7% of that of GA optimizer. Noise deteriorated the score for all optimizers, and there the performance gap between GA and CMA type algorithm was narrowed but persisted: the mean normalized score for CMA was 196.2% of GA in noise free scenario; 151.9% for $\alpha = 0.2$; 146.4% for $\alpha = 0.5$ (Fig. A.4). The overall performance values of the three CMA-type optimizers were not statistically different, and all surpassed GA. (1-way ANOVA, $F_{2,3147} = 1.70, p = 0.18$).

When we examined the performance per layer, we found the relative performance of optimizers had a significant interaction with the source layer of the unit. DiagonalCMA was more effective than Cholesky and original CMA in the earlier layers, but performed less well in deeper layers (Tab. 1, Fig. A.5). We tested this interaction with ANOVA on a linear model: activation $\sim$ optimizer + noiselevel + layer + layer * optimizer + noiselevel * optimizer; in which, “optimizer” was modelled as a categorical variable (CholeskyCMA, CMA, DiagonalCMA), "noiselevel" and "layer" were continuous variable. All the factors except optimizer had statistically significant main effect; and interaction term layer * optimizer had $F_{2,3141} = 14.98, P = 3.34 \times 10^{-7}$ (see Tab. A.1). This curious interaction would be interpreted below in the context of different covariance matrix adaption mechanisms (Sec. 3.1).

As for runtime\(^6\), DiagonalCMA was still the fastest, with average runtime 6.6 ± 0.1 sec (mean± sem, $N = 1050$ same below.), while the second fastest optimizer was GA with runtime 16.6 ± 0.2 sec. In contrast, the classic CMA algorithm taking 97.0 ± 0.8 was the slowest among the four, while the CholeskyCMA took 42.4 ± 0.4 sec per run. Indeed, updating Cholesky factors made it run faster without reducing performance.

### 2.3 CMA outperformed GA in vivo

After the second round of in silico benchmarks, we were ready to compete the top candidates with previous state-of-the-art Genetic Algorithm in vivo. We chose CholeskyCMA algorithm and compared it against the classic GA, since its speed and performance were both good across noise level and visual hierarchy. For detailed methods of in vivo experiments, we refer readers to Sec. A.2, but briefly, we recorded electrophysiological activity in two animals using chronically implanted arrays, placed within V1, V4 and IT.

We conducted 14 experiments. Two out of fourteen (2/14) experiments did not result in a significant increase in firing rate of the neuron for either optimizer (per criterion $P < 0.001$, for t-test between firing rates in first two and last two generation), which we excluded from the further analysis. From the 12 experiments where at least one optimizer increased the firing rate, we normalized the firing rate of the neuron to the highest generation-averaged firing rate for the CholeskyCMA optimizer (Fig. 1 B). The normalized final generation activation for CholeskyCMA was $0.908 \pm 0.018$ comparing to $0.628 \pm 0.045$ for GA (paired t-test, $t = 6.69, p = 3.4 \times 10^{-3}, df = 11$. Raw firing rates for each experiments are shown in Fig. A.6). Thus, the maximal activation of CholeskyCMA surpassed GA by 44%, which was comparable to the activation gain in the high-noise condition (46.2%, $\alpha = 0.5$) in silico.

From this, we concluded that CholeskyCMA outperformed classic GA algorithm both in vivo and in silico, becoming the preferred algorithm for conducting activation maximization.

### 3 THE ANALYSIS OF CMA EVOLUTION

Why did CMA-type algorithms perform so well? Was it the covariance updates, adaptation of step size, or a fortuitous match between the geometric structure of the latent space and the algorithm? We were motivated to find which component contributed to its success. We reasoned the optimizers should work best when they conform to the geometry of the generative image manifold $G$, and the geometry of neuronal tuning function $f$ on the manifold. So we focused on analyzing the geometry of the search trajectory with respect to the geometry of image space $[31]$. When available, we analyzed the in silico and in vivo evolutions back-to-back to validate the effect.

### 3.1 The "Dysfunction" of Covariance Matrix Adaptation

First, we noticed that in a high-dimensional space, the covariance matrix updates were impaired in the original ($\mu/\mu_W, \lambda$) CMA or...
CholeskyCMA algorithm. In the default setting4, the covariance learning rates $c_1, c_2 \propto 1/d^2$, which were exceedingly small at $d = 4096$. Thus, the covariance matrix was updated negligibly and could be well approximated by an identity matrix. Recently, this was also pointed out in [1] (Sec.4.5) and the authors proposed to increase the learning rate in DiagonalCMA. We tested the effectiveness of this modification.

Empirically, we validated this for the original, Cholesky, and Diagonal CMAES algorithms. We quantified this by measuring the condition number of the covariance matrix $\kappa(C)$ and its relative distance to identity matrix $\Lambda(C)$.

\[
\kappa(C) = \frac{\lambda_{\text{max}}(C)}{\lambda_{\text{min}}(C)} \quad \Lambda(C) = \frac{\|C - I\|_F^2}{\|C\|_F^2} \tag{4}
\]

We found that the final generation condition number $\kappa(C)$ of the CholeskyCMA was $1.000175 \pm 0.000004$ (mean ± std, $N = 175$, same below), while the relative distance to identity $\Lambda(C)$ was $8.05 \pm 0.31 \times 10^{-12}$. In comparison, condition number $\kappa(C)$ for the original CMAES was $1.002739 \pm 0.000046$, and for the DiagonalCMA, $1.124728 \pm 0.034774$. The relative distances $\Lambda(C)$ to the identity matrix were $7.13 \pm 0.01 \times 10^{-5}$ and $1.63 \pm 0.21 \times 10^{-4}$ for original CMA and DiagonalCMA algorithms. Though as designed, DiagonalCMA updated the covariance matrix more effectively than the other two CMA algorithms, its final covariance was still quite isotropic. On the other hand, for the original and CholeskyCMA algorithm, we could safely approximate the exploration distribution $p(z_{t+1}^{(i)} | m_t)$ by an isotropic Gaussian scaled by the step size $\sigma_t$

\[
p(z_{t+1}^{(i)} | m_t) \sim \mathcal{N}(m_t, C) \propto \mathcal{N}(m_t, \sigma_t I) \tag{5}
\]

This isotropic exploration throughout the Evolution experiments simplified subsequent analyses of the algorithm.

How is this related to the performance of the algorithm? This relates to the interaction between the unit layer and optimizer, noted above (Sec. 2.2, Tab. 1, Tab. A.1): the DiagonalCMA outperformed original CMA in earlier layers of CNN but not in deeper layers. It seems the faster update of diagonal covariance matrix was only beneficial for units in earlier layers. The diagonal covariance was designed for a separable, ill-conditioned functional landscape. We noticed that although all units had highly ill-conditioned landscapes, the units in shallower layers had tuning for fewer dimensions than units in deeper layers (Sec. C). In other words, the units in earlier layers had a larger invariant space, and a diagonal covariance might suit these landscape better than the more complex ones for deeper units.

In short, we conclude that the effectiveness of CMA-type algorithm was not in its adaptation of the exploratory distribution shape, and we postulate that it could work with fixed covariance.

### 3.2 Evolution Trajectories Showed a Sinusoidal Structure, Characteristic of Random Walks

Next, we investigated the geometric structure of the Evolution trajectories, through the lens of Principal Component Analysis (PCA), a linear dimension-reduction algorithm. For each experiment, we computed the mean latent vector for each generation $\{z_t\}, t = 1..T$, and applied PCA to this $T$-by-$d$ matrix of mean vectors ($T = 75$).

We found a pervasive sinusoidal structure to the typical trajectory. When a given trajectory was projected onto the top principal components (PC), the projection resembled cosine waves (Fig. 3A). On the $k$th component, the projected trajectories were well-fit by cosine functions of frequency $k \pi / T$ (Fig. D.1A). If we allowed the phase and frequency to be fit, $A \cos(2\pi \omega (t/T + \phi))$, the fit $R^2$ was above 0.80 for the top 16 PCs for 95% of trajectories (Fig. D.1B). As a result, projecting the mean trajectory onto the top PCs will result in Lissajous curves (Fig. D.3). Further, we analyzed the PCs of the in vivo Evolution experiments ($N = 264$), and found the same sinusoidal structure (Fig. 3C). Even for control evolutions driven by noise, this sinusoidal structure persisted (Fig. D.2).

This intriguing structure was reminiscent of the PCA of another type of high-dimensional optimization trajectory: that of neural-network parameters during training [20]. This structure was later observed and analytically described for high-dimensional random walks or discrete Ornstein-Uhlenbeck (OU) processes guided by potential [2]. To test this connection, we examined the explained variance of each principal components (Fig. 3B). We found that the explained variance $\rho(k)$ of the $k$th PC was well approximated by Eq. 6, which was derived as the theoretical limit for PCA of random walk with $T$ steps in high-dimensional space $\mathbb{R}^d$[2]. If we take the step number $T \to \infty$ to infinity in Eq. 6, the explained variance of $k$th PC scales with $k^{-2}$ (Eq. 6), showing a fast decay:

\[
\rho(k) = \frac{4}{\pi^2 k^2} \left[1 - \cos\left(\frac{\pi k}{T}\right)\right]^{-1} \approx \frac{6}{\pi^2 k^2} \tag{6}
\]

Indeed, both in theory and in our data, the first principal component explained more than 60% variance of mean evolution trajectory, and the top five PC explained close to 90% of the variance (Fig. 3B). In this view, the evolution trajectory of mean latent code $z$ of CMAES could be regarded as a (guided) random walk in high dimensional space, with its main variance residing in a low-dimensional space.

How are evolutionary trajectories related to random walks? In the extreme, when the response is pure noise, the randomly weighted average of the candidates $\{z_t^{(i)}\}$ will be isotropic; thus each step $p(m_{t+1} | m_t)$ is taken isotropically. Intuitively, this is close to a random walk. In our evolutions, the selection of candidates was biased by the visual-attributed tuning of units or neurons, thus their associated trajectories were random walks guided by given potentials. As derived in [2], for random walks guided by quadratic potentials, the structure of trajectory should be dominated by the dimensions with the smallest driving force (curvature). As we will see below (Sec. 3.3, Sec. B,C), the latent space had a large portion of dimensions to which the units were not tuned. This may explain why our evolution trajectories looked like random walks via PCA. In summary, we interpreted this as an intriguing and robust geometric property of high-dimensional curves [2], but its full significance remains to be defined.

### 3.3 Evolution Trajectories Preferentially Travel in the Top Hessian Eigenspace

Given that individual trajectories had a low-dimensional structure, we asked if there was a common subspace that these Evolution

4See the default setting of $c_1, c_2$ in Tab. 2 of [12].
The Hessian Eigenspace. The first 800th eigenvalue plotted in dashed line. Regression lines plotted against the log eigenvalue. We found that the collection of observed visual attributes of different complexity [28], it was possible that the preferred visual features populated certain subspaces in the latent space of our generator.

We first collected the in silico Evolution trajectories of N = 1050 runs, across all conditions, and represented each trajectory by the mean code from the final generation, i.e. the evolution direction \( \zeta_j \). We shuffled the entries of each vector \( \{ \zeta_j \} \), \( j = 1...N \) to form a control collection of evolution directions \( \{ \zeta_{shf} \} \), preserving their vector norms. We found that the collection of observed Evolution directions was lower-dimensional than its shuffled counterpart: by PCA, the explained variance of top 7 PCs was larger than the shuffled counterpart \( p < 0.005 \) comparing to 500 shuffles, Fig. A.7.

Next, we examined the relationship between the collection of trajectories and the global geometry of the space (Sec. B). Previously, we reported that the latent space of the generator exhibits a highly anisotropic geometry, as quantified by the Riemannian metric tensor \( H \) of this image manifold [31]. The bilinear form \( \sigma^T H \sigma \) defined by this Riemannian metric tensor represented the degree of image change when moving in direction \( \sigma \). Consequently, moving in top eigen-directions changes the generated image much more than other directions, while moving in the bottom eigenspace scarcely affects the image (Fig. B.1.B.C). Moreover, this structure is homogeneous in the space, thus similar directions will cause rapid image change at different positions in the latent space. Thus, there exist global dimensions that affect the image a lot, and dimensions that do not. We hypothesized that the optimization trajectories might preferentially travel in some part of this eigenspace.

We used the averaged metric tensor \( H \) from Wang and Ponce (2021, [31]). Its eigenvectors \( U = \{ u_1, u_2...u_d \} \) with eigenvalues \( \lambda_k \), \( k = 1...d \) became our reference frame for analyzing trajectories. We projected the collection of evolution directions \( \{ \zeta_j \} \) onto this basis, and examined the mean projection amplitude on each eigenvector \( \frac{1}{N} \sum_{j=1}^{N} |u_k^T \zeta_j| \) as a function of eigenvalue \( \lambda_k \) (Fig. 4A). Since for in silico experiments, the initial generation was the origin, this quantity was the average distance travelled along the eigenvector across runs. We expected the trajectories to travel further for eigen-dimensions where more CNN units exhibit "gradient".

We applied the Kolmogorov–Smirnov test to determine if the distribution of projection coefficient in the top and bottom eigen-dimensions were different. We found that the projection coefficients onto the top 800 eigen-dimensions as a distribution were significantly different from those onto bottom dimensions (KS statistics 0.223, \( P = 9.3 \times 10^{-18} \), Fig. 4A). Further, within the top 800 PC dimensions, the mean traveled distance strongly correlated with the log eigenvalues (Pearson correlation 0.738, \( P = 2.8 \times 10^{-138} \), Fig. 4A), i.e. the trajectories tended to travel further for dimensions with larger eigenvalues.

As for in vivo evolutions, we examined the collection of evolution trajectories (\( N = 264 \) from 2 monkeys, [28]) and projected them.
onto the Hessian eigenframe as above. We found that they preferentially aligned with the top eigenspace than the lower eigenspace (Fig. 4B): the distance traveled in the top 800 eigenspace correlated with the eigenvalues (Pearson correlation $0.319, p = 1.2 \times 10^{-15}$).

Since in vivo evolutions used a set of initial codes $\{z_0^{(i)}\}$ instead of the origin, we used noise-driven evolutions starting from these initial codes as control. As a baseline, the noise-driven evolutions had a lower correlation between the distance traveled and the eigenvalues ($r = 0.198$ for the noise evolution with matching generations; $r = 0.223$ for that with matching code norm). Our results were robust to the choice of cutoff dimension (600-1000).

In conclusion, this result showed us that the evolution trajectories traveled further in the top eigenspace, and the average distance traveled was positively correlated with the eigenvalue. How do we interpret this effect? Since the top eigen-dimensions change the images more perceptibly, the tuning functions of neurons and CNN units were more likely to exhibit a "gradient" in such subspace. As the lower eigenspace did not induce perceptible changes, those dimensions barely affected the activations of units or neurons. Thus, no signal could guide the search in lower eigenspace, inducing dynamics similar to pure diffusion. As the visually tuned units could exhibit gradient in the top eigenspace, the search would be a diffusion with a driving force, allowing the optimizers to travel farther in the top eigenspace.

### 3.4 The Spherical Geometry of Latent Space Facilitated Convergence

Finally, we noticed one geometric structure that facilitated the convergence of the CMA algorithm. For the DeepSim generator[8] with DCGAN architecture[26], the mapping $G$ was relatively linear. Namely, changing the scale of the input $z$ mainly affected the contrast of the generated image $G(z)$ (Fig. B.1). Thus, when the base vector $z_0$ has a larger norm, travelling the same euclidean distance $\Delta z$ will induce a smaller perceptual change; in contrast, travelling the same angular deviation $\Delta \theta$ will result in a similar pattern change regardless of the norm of the base vector $z_0$ (Fig. A.8). We reasoned that the angular distance in latent space would be a better proxy to the perceptual distance across generated images.

We validated this using the perceptual dissimilarity metric Learned Perceptual Image Patch Similarity (LPIPS, [34]). We measured image variability in each generation using the mean LPIPS distance between all pairs. We measured the code variability by the standard deviation (std) of latent codes $\tilde{\sigma}_t$ averaged across $d$ dimensions, which was an estimate of the step size $\sigma_t$ and was strongly correlated with the mean L2 distance among codes. We also measured the mean angular distance between all pairs of codes in each generation. We found that the std of latent codes $\tilde{\sigma}_t$ decreased from $\tilde{\sigma}_0 = 3.0$ (first generation) to $\tilde{\sigma}_T = 2.31 \pm 0.04$ ($N = 1050$, and hit the floor at around 25 generations (Fig. 6). In contrast, the perceptual variability kept decreasing till the end, which was better approximated by the trend of angular variability within each generation (from 1.57 rad to 0.50 rad, Fig. 6). This discrepancy of Euclidean and angular distance was mediated by the increasing vector norm during evolution: we found that the squared norm of the mean latent codes $\tilde{\sigma}_t$ scaled linearly with generation $\|\tilde{z}_t\|^2 \propto t$ ($r^2 = 0.9985 \pm 0.008, N = 1050$, Fig. D.2C), which is a classic property of a random walk. As a combined effect of the increasing vector norm and the decreasing step size $\sigma_t$, the angular variability within a generation decreased towards the end. Similarly, we found that across each given trajectory, angular distance was also a better proxy for the perceptual distance between images than L2 distance (Fig. 5). The mean-code images from late generations were more similar to each other than the ones from initial generations, as predicted by the angular distance between the mean vectors — but not L2 distance.

For a visual neuron or a CNN unit, the perceptual distances across images matter more. Thus, when designing the new optimizer, we reasoned it should control the angular step size as a proxy to perceptual distance, instead of the Euclidean distance ($\sigma_t$) (Sec. 4).

### 4 PROPOSED IMPROVEMENT: SPHERECMA

Finally, we proposed an optimizer incorporating the findings above. We would test whether this new optimizer could perform as well as the original CMA algorithms.

This optimizer is designed to operate on a hypersphere, thus it was called SphereCMA (Alg. 1). This design was guided by the training of the image generator $G$. Many generative models (GAN) were trained to map an isotropic distribution of latent codes $\rho(z)$ (e.g. Gaussian [8, 17] or truncated Gaussian [3]) to a distribution of natural images. Thus, the generator $G$ has only learned to map latent codes $z$ sampled from this distribution. In a high-dimensional $d$ space, the density of standard normal distribution concentrates around a thin spherical shell with radius $R \approx \sqrt{d}$. Due to this, latent codes should be sampled from this hypersphere to obtain natural images. Thus, we enforced the optimizer to search on this sphere.
We tested this algorithm against other CMA-style algorithms in the same setup as in Sec. 2.2. Specifically, we tested two versions with different step-size-decay function $\mu_{\text{dec}}$, exponential (Sp Exp) and inverse function (Sp Inv). The change of angular step size through generations is shown in Fig. A.9, the parameters for the decay function were tuned by Bayesian optimization and fixed.

We found that, SphereCMA with exponential decay (Sp Exp) outperformed all other CMA algorithms when pooled across all noise levels and layers (Tab. 1, two-sample t-tests, $t_{2098} = 4.03, p = 5.7 \times 10^{-5}$ for CholeskyCMA, $t_{2098} = 4.92, p = 9.1 \times 10^{-7}$ for original CMA, $t_{2098} = 3.30, p = 1.0 \times 10^{-3}$ for DiagonalCMA, $N = 1050$). Compared to CholeskyCMA, SphereCMA surpassed its achieved activation by around 7%. Interestingly, when we compared the performance per noise level, we found that SphereCMA-Exp was the best performing algorithm under the noise-free condition ($\alpha = 0$) across the seven layers (two-sample t-test, $t_{2098} > 9, p < 1 \times 10^{-20}$ for all of CholeskyCMA, original CMA and DiagonalCMA; runs were pooled across 7 layers, $N = 350$, same below). But in noisier conditions, it performed comparably or sometimes worse than other CMA optimizers: at $\alpha = 0.2, 0.5$ all comparisons with other optimizers were not significant except for SphereCMA-Exp vs DiagonalCMA at $\alpha = 0.5$ ($t_{2098} = -3.08, p = 2.2 \times 10^{-3}$). For trajectory comparison, see Fig. A.2.

Overall, this result confirmed that the principles extracted from analyzing CMA-style algorithms were correct and crucial for their performance. By building these essential structures into our spherical optimizer, we could replicate and even improve the performance of CMA-ES algorithms. Finally, the interaction between the performance of SphereCMA-Exp and the noise level, provided us a future direction to find the source of noise resilience in the original CMA algorithm and improve the SphereCMA algorithm.

### 5 DISCUSSION

In this study, we addressed a problem common to both machine learning and visual neuroscience – the problem of defining the visual attributes learned by hidden units/neurons across the processing hierarchy. Neurons and hidden units are highly activated when the incoming visual signal matches their encoded attributes, so one guiding principle for defining their encoded information is to search for stimuli that lead to high activations. Since the brain does not lend itself to gradient descent, it is necessary to use evolutionary algorithms to maximize activations of visual neurons in vivo. Here, we identified a class of optimizers (CMA) that work well for this application and analyzed why they perform so well, and finally developed a faster and better optimizer (SphereCMA) based on these analysis. Here are some lessons we learned in this exploration. First, screening with comprehensive in silico benchmarks accelerates the algorithm development in vivo. Secondly, geometry of latent space matters. As a general message, when developing optimizers searching in the latent space of some generative models, we should pay attention to the distance structure i.e. geometry of the generated samples (e.g. images) instead of the latent space. Optimizers leveraging space geometry shall perform better. We hope our workflow can help the optimizer design in other domains, e.g. search for optimal stimuli in other sensory modalities and drug discovery in molecular space.
ACKNOWLEDGMENTS

We appreciate Mary Burkemper in her assistance in monkey managing and data collection. We thank Yunyi Shen, Kaining Zhang for reading the manuscript and providing valuable feedbacks. We are grateful to the BIS cluster and staff in Washington University in St Louis for their GPU resources and technical support.

B.W. is funded by the McDonnell Center for Systems Neuroscience in Washington University (pre-doctoral fellowship to B.W.). Additional research funding by the David and Lucile Packard Foundation and the E. Matilda Ziegler Foundation for the Blind (C.R.P.).

REFERENCES

[1] Y. Akimoto and N. Hansen. 2020. Diagonal Acceleration for Covariance Matrix Adaptation Evolution Strategies. Evolutionary Computation 28, 3 (sep 2020), 405–435. https://doi.org/10.1162/EVCO_A_00260 arXiv:1905.05885

[2] Joseph M Antognini and Jascha Sohl-Dickstein. 2018. PCA of high dimensional random walks with comparison to neural network training. arXiv preprint arXiv:1806.08895 (2018).

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018).

[4] Eric T. Carlson, Russell J. Rasquinha, Kechen Zhang, and Charles E. Connor. 2014. A sparse object coding scheme in area V4. Current biology. CB 21 (2014), 288–293. Issue 4. https://doi.org/10.1016/J.CUB.2011.01.013

[5] Gabriela Czanner, Sridevi V. Sarma, Demba Ba, Uri T. Eden, Wei Wu, Emad Eskandar, Hubert H. Lim, Simona Temereanca, Wendy A. Suzuki, and Emery N. Brown. 2015. Measuring the signal-to-noise ratio of a neuron. Proceedings of the National Academy of Sciences of the United States of America 112, 23 (jun 2015), 7141–7146. https://doi.org/10.1073/pnas.1505545112

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition. Ieee, 248–255.

[7] R. Desimone, T. D. Albright, C. G. Gross, and C. Bruce. 1984. Stimulus-selective properties of inferior temporal neurons in the macaque. The Journal of neuroscience : the official journal of the Society for Neuroscience 4, 8 (aug 1984), 2051–62. https://doi.org/10.1523/JNEUROSCI.04-08-02051.1984

[8] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In Advances in neural information processing systems. 658–666.

[9] Jeremy Freeman and Eero P Simoncelli. 2011. Metamers of the ventral stream. Nature neuroscience 14, 9 (2011), 1195–1201.

[10] Charles G. Gross. 1994. How inferior temporal cortex became a visual area. Cerebral Cortex 4, 5 (1994), 455–469. https://doi.org/10.1093/cercor/4.5.455

[11] Nikolaus Hansen. 2016. The CMA Evolution Strategy: A Tutorial. (apr 2016). arXiv:1604.00772 http://arxiv.org/abs/1604.00772

[12] Nikolaus Hansen and Anne Auger. 2014. Principled Design of Continuous Stochastic Search: From Theory to Practice. (2014). https://doi.org/10.1007/978-3-642-33206-7_8

[13] N. Hansen and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies: Evolutionary computation 9, 2 (2001), 159–195. https://doi.org/10.1109/44.902669

[14] J. Hegdé and D. C. Van Essen. 2000. Selectivity for complex shapes in primate visual area V2. The Journal of neuroscience : the official journal of the Society for Neuroscience 20, 5 (2000). https://doi.org/10.1523/JNEUROSCI.20-05-J0010.2000

[15] David H Hubel and Torsten N Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. The Journal of physiology. 160, 1 (1962), 106.

[16] Chia Chun Hung, Eric T. Carlson, and Charles E. Connor. 2012. Medial Axis Shape Coding in Macaque Inferotemporal Cortex. Neuron 74 (6 2012), 1099–1113. Issue 6. https://doi.org/10.1016/J.NEURON.2012.04.029

[17] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition. 4401–4410.

[18] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In Advances in neural information processing systems. 3387–3395.

[19] A Pasupathy and C E Connor. 1999. Responses to contour features in macaque area V4. Journal of neurophysiology 82, 5 (nov 1999), 2490–502. http://www.ncbi.nlm.nih.gov/pubmed/10563421

[20] David H. Hubel and Torsten N. Wiesel. 2019. Invariant visual representation by simple neurons in the human brain. Nature 435, 7045 (2005), 1102–1107.

[21] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015).

[22] J. Rapin and O. Tremain. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Neversgrad.

[23] Olivia Rose, James Johnson, Binxu Wang, and Carlos R. Ponce. 2021. Visual prototypes in the ventral stream are attuned to complexity and gaze behavior. Nature Communications 2021 12:1 (12 2021), 1–16. Issue 1. https://doi.org/10.1038/s41467-021-27027-8

[24] Martin Schrimpf, Jonas Kuhlbusch, Ha Hong, Najib Majaj, Rishi Rajalingham, Elia Issa, Kojiit Kus, Pouya Bashivan, Jonathan Prescott-Roy, Franziska Geiger, Kaiyln Schmidt, Daniel Yamins, and James DiCarlo. 2018. Brain-Score: Which Artificial Neural Network for Object Recognition is most Brain-Like? bioRxiv (jan 2018), 407007. https://doi.org/10.1101/407007

[25] Ramanujan Srinath, Alexanderia Emonds, Qingyang Wang, Augusto A. Lempel, Erika Dunn-Weiss, Charles E. Connor, and Kristina J. Nielsen. 2021. Early Emergence of Solid Shape Coding in Natural and Deep Network Vision. Current Biology 31 (1 2021), 51–55.e5. Issue 1. https://doi.org/10.1016/J.CUB.2020.09.076

[26] Binxu Wang and Carlos R. Ponce. 2021. The Geometry of Deep Generative Image Models and its Applications. (1 2021). https://arxiv.org/abs/2101.06006

[27] Will Xiao and Gabriel Kreiman. 2020. XDream: Finding preferred stimuli for visual neurons using generative networks and gradient-free optimization. PLOS Computational Biology 16, 6 (jun 2020), e1007973. https://doi.org/10.1371/JOURNAL.PCBI.1007973

[28] Yukako Yamane, Eric T Carlson, Katherine C Bowman, Zhihong Wang, and Charles E. Connor. 2008. A neural code for three-dimensional object shape in macaque inferotemporal cortex. Nature Neuroscience 11, 11 (nov 2008), 1352–1360. https://doi.org/10.1038/nn2202

[29] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE conference on computer vision and pattern recognition. 586–595.