RIEMANNIAN ADVERSARIAL ATTACKS ON SYMMETRIC POSITIVE DEFINITE MATRICES

Dimitri Timoz[⋆] *Thibault de Surrel*[†] *Florian Yger*[⋆]

[⋆] LITIS, INSA Rouen-Normandy, France
[†] LAMSADE, CNRS, PSL Univ. Paris-Dauphine, France

ABSTRACT

In this paper, we study adversarial vulnerabilities when inputs lie on the manifold of symmetric positive definite (SPD) matrices by proposing a Riemannian Projected Gradient Descent (R-PGD) attack. This attack performs updates along the affine-invariant geometry and projects using a geodesic budget. We also give a reconstruction procedure that maps adversarial SPD matrices back to the original signals while enforcing spectral constraints. On Brain Computer Interface datasets and SPDNet, R-PGD is more effective than Euclidean PGD and yields geometrically tailored perturbations that remain adversarial after pre-processing. Our results motivate robustness analyses and defenses for manifold deep models.

Index Terms— Adversarial attacks, Symmetric Positive Definite matrices, Riemannian geometry

1. INTRODUCTION

Covariance matrices are increasingly used in the analysis of multivariate signals due to their ability to capture statistical dependencies, particularly correlations, between different signal channels. Unlike raw signals, covariance matrices are often more robust to noise and offer a compact representation that retains essential structural information. This makes them highly valuable in applications such as medical imaging [1] or brain–computer interfaces (BCIs) [2], where signals are often high-dimensional and noisy. In fields like medical imaging, where errors can have serious consequences, evaluating the robustness of models is essential. In these contexts, covariance-based approaches have improved performances in tasks such as classification [3]. Moreover, the mathematical properties of covariance matrices such as their symmetric positive-definite (SPD) nature enable the use of advanced techniques grounded in Riemannian geometry. These approaches leverage the non-Euclidean structure of the space of SPD matrices, allowing for more accurate methods.

With the growing popularity of SPD matrices in signal processing and machine learning, several deep learning architectures have been specifically designed to operate on such structured data. A popular example is the SPDNet [4] that will be used in the following. At the same time, it has become well established that deep neural networks are vulnerable to adversarial attacks, due to their high capacity and sensitivity to small, carefully crafted perturbations [5, 6, 7]. These vulnerabilities can be especially concerning when the models operate on geometrically structured inputs like SPD matrices, where preserving the intrinsic properties of the data is crucial. In [8], the authors attack general neural networks (e.g., for images) by assuming that data are distributed on some underlying manifold and adapting adversarial perturbations accordingly. In our case, the data naturally lie on a known and fixed manifold (the manifold of SPD matrices), so our approach falls within the setting of learning on manifolds. Such attacks have been studied on structured data such as the hyperbolic manifold [9] or graph data [10]. To our knowledge, no such work has been conducted for SPD matrices.

Our objective is to construct a perturbation of a SPD matrix such that the semantic (i.e., Riemannian) distance to the original sample is minimal, while the prediction outcome by a SPDNet is completely altered (i.e., misclassified).

There are two main types of attacks: black-box attacks, where the model's parameters and architecture are unknown but predictions can still be obtained, and white-box attacks, where the model's parameters are known and gradients can be computed. Since a model that is robust to white-box attacks is also generally robust to black-box attacks, we focus on white-box attacks.

The main contributions of this paper are:

- A Riemannian version of Projected Gradient Descent (PGD) adapted to the geometry of the SPD manifold;
- A reconstruction method that maps adversarial SPD matrices back to time-domain signals;
- A thorough evaluation of SPDNet [4] trainied on BCI datasets, showing robustness and effectiveness.

2. LEARNING ON SPD MATRICES

2.1. The manifold of SPD matrices

The space of $d \times d$ Symmetric Positive Definite (SPD) matrices, denoted $\mathcal{S}_d^{++}$, forms a smooth Riemannian manifold.

The tangent space of $\mathcal{S}_d^{++}$ at a point Σ is denoted as $\mathcal{T}_\Sigma\mathcal{S}_d^{++}$ and consists of all symmetric matrices of dimension d. Equipping $\mathcal{S}_d^{++}$ with an appropriate Riemannian metric allows for meaningful notions of distance and geodesics. This geometric framework enables operations that respect the intrinsic curvature of the space, leading to more robust algorithms operating on $\mathcal{S}_d^{++}$. In this work we use the Affine Invariant Riemannian Metric (AIRM) [1, 11] defined on the tangent space $\mathcal{T}_\Sigma\mathcal{S}_d^{++}$ at Σ by:

$$\langle A, B\rangle_\Sigma = \mathrm{Tr}(\Sigma^{-1}A\,\Sigma^{-1}B), \quad A, B \in \mathcal{T}_\Sigma\mathcal{S}_d^{++}.$$

This metric allows us to define a distance on the manifold $\mathcal{S}_d^{++}$, denoted δ_r and given by:

$$\delta_r(\Sigma_1, \Sigma_2) = \|\log(\Sigma_1^{-1/2}\Sigma_2\Sigma_1^{-1/2})\|_F, \quad \Sigma_1, \Sigma_2 \in \mathcal{S}_d^{++}, \tag{1}$$

where $\log$ denotes the matrix logarithm and $\|\cdot\|_F$ the Frobenius norm. To map a SPD matrix X onto the tangent space at Σ, we use the logarithmic mapping:

$$\mathrm{Log}_\Sigma(X) = \Sigma^{\frac{1}{2}}\log\left(\Sigma^{-\frac{1}{2}}X\Sigma^{-\frac{1}{2}}\right)\Sigma^{\frac{1}{2}}, \quad X \in \mathcal{S}_d^{++}.$$

Vice versa, to project from the tangent space $\mathcal{T}_\Sigma\mathcal{S}_d^{++}$ back onto $\mathcal{S}_d^{++}$, we use its inverse, the exponential mapping:

$$\mathrm{Exp}_\Sigma(V) = \Sigma^{\frac{1}{2}}\exp\left(\Sigma^{-\frac{1}{2}}V\Sigma^{-\frac{1}{2}}\right)\Sigma^{\frac{1}{2}}, \quad V \in \mathcal{T}_\Sigma\mathcal{S}_d^{++}.$$

For $f\colon \mathcal{S}_d^{++} \to \mathbb{R}$ a smooth function, let ∇f be its Euclidean gradient. Then, its Riemannian gradient $\mathrm{grad}\, f$ can be computed as (see [12] for more details)

$$\mathrm{grad}\, f(\Sigma) = \Sigma\left(\nabla f(\Sigma)\right)_{\mathrm{sym}}\Sigma,$$

with $(M)_{\mathrm{sym}} = \frac{1}{2}(M + M^\top)$ is the symmetrized of Σ.

2.2. The SPDNet

SPDNet is a neural network architecture designed to operate on SPD matrices, presented in [4]. This model introduces three new layers: the *BiMap*, the *ReEig* and the *LogEig* layers that preserve the intrinsic geometry of SPD matrices. Let us describe them:

BiMap Layer. The BiMap layer applies an affine transformation to the input SPD matrix using a bilinear projection, defined as:

$$f_{\text{BiMap}}(\Sigma) := W\Sigma W^\top \tag{2}$$

where the trainable weights W belong to the compact Stiefel manifold of matrices in $\mathbb{R}^{i\times o}$ with orthonormal columns. Here, o and i denote the output and input dimensions, respectively, with $o \leq i$. We can stack multiple BiMap layers, each dedicated to processing a different channel.

ReEig Layer. To ensure the output of the BiMap layer remains an SPD matrix, the ReEig layer enforces positive eigenvalues by performing spectral decomposition: $\Sigma = U\Lambda U^\top$ followed by a rectification step:

$$f_{\text{ReEig}}(\Sigma) := U\max(\epsilon I, \Lambda)U^\top \tag{3}$$

where $\epsilon > 0$ is a small constant (usually $\epsilon = 10^{-6}$) ensuring numerical stability and maintaining the positive-definiteness of the matrix. Although this layer can also serve as an activation function, in practice, we observe that the model does not significantly exploit this capability. For practical purposes, it is advisable to add a small random perturbation $\epsilon + \mu$ to each eigenvalue. This helps avoid ill-conditioning errors when computing the eigen decomposition. We rely on the initial work on SPDNet [4] where the authors exploit the matrix generalization of back propagation to compute the gradients of the ReEig layer.

LogEig Layer. The final layer is designed to apply Euclidean operations on SPD output. To achieve that, the authors of [4] define a LogEig layer by applying a $\log(\cdot)$ operation.

$$f_{\text{LogEig}}(\Sigma) := U\log(\Lambda)U^\top \tag{4}$$

This layer is typically placed at the end of the network, before some classical Euclidean fully connected layers.

The Network. SPDNet is built by stacking BiMap and ReEig layers in order to propagate information directly on the SPD manifold while preserving positive-definiteness. The resulting representations are then projected to a tangent space (via a LogEig layer) before being processed by standard fully-connected (FC) layers for classification.

3. RIEMANNIAN ADVERSARIAL ATTACKS

3.1. The multi-channel attack

One method behind adversarial attacks on deep learning models is to find a way to increase the loss function J while respecting a budget constraint ϵ. This can be achieved efficiently using a gradient ascent algorithm applied to the input data. One of the most common and effective attack on Euclidean inputs is the *Projected Gradient Descent* (PGD) attack [13]. It consists of iterations of Riemannian Ascent Gradient method where the perturbation is allowed to leave the constraint set, typically defined as a ball or a sphere with respect to the Riemannian distance given at Eq. 1 before being projected back onto it for each iteration. This allows the perturbation to "travel" more freely at the border of the sphere defined by the budget, which helps correct the direction of the gradient if needed.

We propose the following Riemannian PGD attack:

$$\Sigma_{k+1} = \Pi_{\mathcal{B}_\epsilon(\Sigma_0)}(\exp_{\Sigma_k}(\alpha\,\mathrm{grad}\, J(\Sigma_k, Y)))$$

where Y is the target of the model. To ensure Σ_{k+1} is still on the manifold $\mathcal{S}_d^{++}$, we apply an exponential or retraction map in the direction of the Riemannian gradient. Then, to project back onto the ball $\mathcal{B}_\epsilon(\Sigma_0) = \{\Sigma \in \mathcal{S}_d^{++} \mid \delta_r(\Sigma_0, \Sigma) \leq \epsilon\}$, we use the following projection:

$$\Pi_{\mathcal{B}_\epsilon(\Sigma_0)}(\Sigma) = \begin{cases} \exp_{\Sigma_0}\left(\frac{\epsilon}{\delta_r(\Sigma_0,\Sigma)} \log_{\Sigma_0}(\Sigma)\right) & \text{if } \delta_r(\Sigma_0, \Sigma) > \epsilon, \\ \Sigma & \text{otherwise.} \end{cases}$$

This projection ensure that the attack respect the budget and is still on the manifold $\mathcal{S}_d^{++}$. We can initialize Σ_0 at a random position in the ball using a wrapped Gaussian on $\mathcal{S}_d^{++}$ [14]. Running the attack multiple times and keeping the best adversarial sample often leads to better results. The Riemannian PGD attack has the advantage of defining the attack budget using a Riemannian distance, which limits the loss of geometric information caused by the attack and helps preserve the semantics of the original sample.

3.2. The reconstruction of the attacked signal

Once we have generated covariance matrices that fool the model we want to be able to come back to our signals. We have to keep in mind that they will be filtered by a pre-processing step. The algorithm must be able to find signals resistant to them. We aim to construct a perturbation $E \in \mathbb{R}^{d \times t}$ such that, when added to a clean multichannel signal $X \in \mathbb{R}^{d \times t}$, the perturbed signal $\tilde{X} = X + E$ has a target covariance matrix Σ_{target}.

To explicitly control the spectral content of E, we define it in the frequency domain. For each channel i, we construct the raw signal $e^{(i)}$ through its Fast Fourier Transform (FFT):

$$s^{(i)} = \text{FFT}(e^{(i)}), \quad S = [s^{(1)}, \ldots, s^{(d)}] \in \mathbb{R}^{d \times f} \tag{5}$$

where $f = t//2 + 1$ is the number of positive frequency bins. We optimize E to match the target covariance, by minimizing:

$$L_{\text{cov}}(E, \Sigma_{\text{target}}) = \left\| (X + E)(X + E)^\top - \Sigma_{\text{target}} \right\|_F^2 \tag{6}$$

To promote band frequencies, we use this regularizer:

$$L_{\text{freq}}(S) = \frac{1}{d|F|} \sum_{i=1}^{d} \sum_{k \in F} |s^{(i)}[k]| \tag{7}$$

where F is the set of reduced frequencies by the pre-processing.

The total loss is:

$$L(E) = L_{\text{cov}}(E) + \lambda L_{\text{freq}}(\text{FFT}(E)) \tag{8}$$

where λ a hyperparameter (usually 10^{-3}) that balances the priority between the MSE and the frequencies regularizer.

We optimize E using L-BFGS [15] for 120 epochs with a learning rate of 0.5 to produce a perturbation that shapes the signal's covariance while concentrating its energy within a specific bandwidth. We avoid constrained optimization because it does not always guarantee an admissible solution.

4. EXPERIMENTS

4.1. Generating adversarial SPD matrices

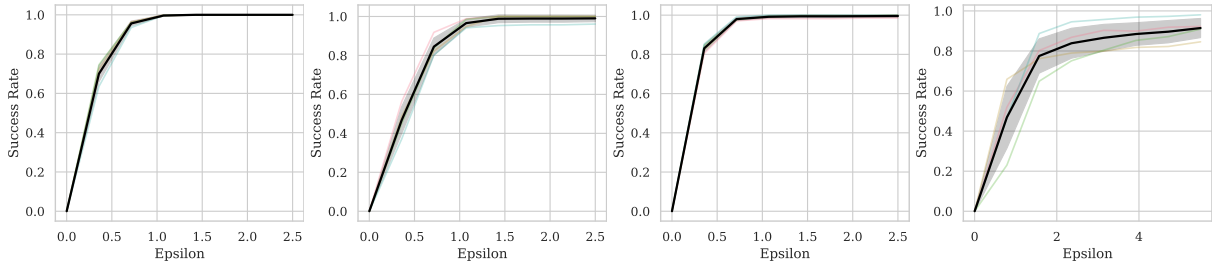
We evaluate our adversarial attacks on a SPDNet trained on BCI datasets. The model contains 2 BiMap and ReEig layers of one channel (22×22, 22×16) such that we have intermediate matrices respectively of size 22×22 and 16×16, followed by a LogEig and FC layer leading to 2 output classes. A RiemannianAdam optimizer [16] was used for training.

We used the motor imagery datasets BNCI2014-001 [17] and Zhou2016 [18] from MOABB [19] for our experiments. These datasets consist of several subjects and several sessions for each subjects, all with balanced classes. We start by applying a band-pass filter with range $[7; 35]$ Hz for each dataset. The models were trained in a inter-subject setup, by excluding a subject from the training set and evaluating on the same held subject. For both datasets we attacked the model only for the 4 best subjects. The R-PGD attack had 40 iterations and always converged before the end with a step size α of 0.1.

We also tried to apply Euclidean attacks to our architectures, using a L_2 norm budget. Some of these attacks generated perturbations that were not admissible on the manifold of SPD matrices $\mathcal{S}_d^{++}$, i.e., they produced matrices that were no longer SPD. Additionally, Euclidean attacks do not use the same distance metric as Riemannian attacks, making it difficult to compare them in terms of perturbation magnitude. The Riemannian distance can differ significantly from the Euclidean one and better captures the intrinsic geometry of the manifold $\mathcal{S}_d^{++}$, which makes it more appropriate and meaningful in the context of adversarial attacks.

In Figure 1, we observe that the Euclidean PGD attack encounters increasing difficulty in completely degrading the model's performance beyond a certain perturbation budget. This phenomenon arises primarily because, at higher budgets, the attack tends to move outside the SPD manifold. Consequently, the mandatory projection step onto the SPD manifold significantly reduces the effective perturbation magnitude. On the other hand, for R-PGD we clearly observe that the attack keeps progressing steadily until it eventually reaches 100% accuracy. When comparing Euclidean PGD and Riemannian PGD on the dataset Zhou2016, we observe that R-PGD does not exhibit a saturation phase, whereas the Euclidean version struggles to achieve 100% success.

This behavior can be intuitively understood from Figure 2: Euclidean-based attacks must follow trajectories that are intrinsically curved due to the geometry of the manifold, yet they attempt to approximate these paths using straight-line updates. The resulting frequent projections back onto the manifold decrease the effectiveness of each update, explaining the observed saturation in performance degradation.



(a) R-PGD on BNCI2014-001 (b) R-PGD on Zhou2016 (c) PGD on BNCI2014-001 (d) PGD on Zhou2016

Fig. 1: Riemannian and Euclidean PGD Attacks success rate (i.e. misclassified samples after perturbation over the amount of sample) on BCI models trained on inter-session with Motor Imagery paradigm on all the subject except one evaluation subject.

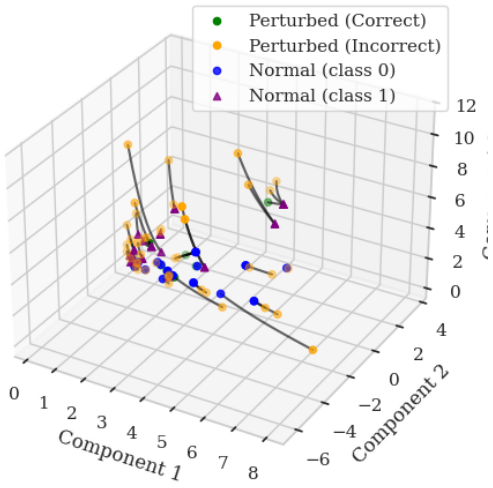


Fig. 2: Riemannian 2×2 t-SNE [20] visualization of adversarial perturbations on SPD matrices. Each point corresponds to a sample on the SPD manifold: normal data (blue for class 0, purple for class 1) and perturbed data (green if still correctly classified, orange if misclassified). Black curves indicate the affine-invariant geodesics between original samples and their perturbations. We can clearly distinguish clusters for each class, with perturbations tending to move in opposite directions across the manifold.

4.2. Back to the signals

In practice, an adversarial SPD matrix is of limited use if it cannot be traced back to a corresponding perturbation signal. To assess the effectiveness of our method, we attempt to reconstruct the perturbation signal that led to the previously obtained covariance matrices that have fooled the model.

As we trained our model, we applied a pre-processing step on the perturbed signals by filtering out frequencies outside the $[7; 35]$ Hz band before evaluating the quality of our attacks. In Table 1, we observe that most of our Riemannian PGD attacks are successfully reconstructed on both *Zhou2016* and *BNCI2014_001*, with attack budgets ϵ ranging from 0.3 to 2.5. The pre-processing step only slightly decreases the attack quality. This can be explained by the fact that the perturbed signal leverages the clean signal to produce new covariance matrices. The more severe the filtering, the more the regularizer reduces construction and pre-processing error.

	Zhou2016	BNCI2014_001
With filtering	99.93%	99.32%
Without filtering	100%	99.87%

Table 1: Attack that still are fooling the model after signal reconstruction and the whole model pre-processing step.

5. CONCLUSION

In this work, we introduced a novel Riemannian PGD attacks tailored to the manifold of SPD matrices. We also proposed a signal reconstruction method that successfully fools the model, demonstrating resilience even under typical pre-processing filters used in BCI pipelines.

Our experiments on BCI datasets show that Riemannian attacks outperform Euclidean ones in attack success rate and geometric fidelity, and that these adversarial examples remain effective after reconstruction. This highlights the importance of robustness analysis in non-Euclidean learning and opens the door to future work on defense strategies for those models.

Despite the effectiveness of the proposed Riemannian attack, our approach comes with several limitations as our attack relies on full access to model gradients, which makes it inapplicable in black-box settings. Although white-box robustness often implies some black-box resistance, the performance of our approach in black-box scenarios remains unexplored. While we propose an effective attack pipeline, we do not explore potential defense mechanisms, such as Riemannian adversarial training or manifold regularization, which would provide a more complete robustness analysis.

6. REFERENCES

[1] Xavier Pennec, "Manifold-valued image processing with SPD matrices," in *Riemannian Geometric Statistics in Medical Image Analysis*, Xavier Pennec, Stefan Sommer, and Tom Fletcher, Eds., pp. 75–134. Academic Press, Jan. 2020.

[2] Florian Yger, Maxime Berar, and Fabien Lotte, "Riemannian approaches in brain-computer interfaces: A review," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 10, pp. 1753–1762, 2017.

[3] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten, "Multiclass brain–computer interface classification by riemannian geometry," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 4, pp. 920–928, 2012.

[4] Zhiwu Huang and Luc Van Gool, "A riemannian network for spd matrix learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2017, vol. 31.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin, "On evaluating adversarial robustness," 2019.

[6] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021.

[7] Xiao Zhang and Dongrui Wu, "On the vulnerability of cnn classifiers in eeg-based bcis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 5, pp. 814–825, 2019.

[8] Shufei Zhang, Kaizhu Huang, Rui Zhang, and Amir Hussain, "LEARNING ADVERSARIAL EXAMPLES WITH RIEMANNIAN GEOMETRY," 2019.

[9] Max van Spengler, Jan Zahálka, and Pascal Mettes, "Adversarial attacks on hyperbolic networks," in *Computer Vision – ECCV 2024 Workshops*, Alessio Del Bue, Cristian Canton, Jordi Pont-Tuset, and Tatiana Tommasi, Eds., Cham, 2025, pp. 363–381, Springer Nature Switzerland.

[10] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song, "Adversarial attack on graph structured data," *arXiv preprint arXiv:1806.02371*, 2018.

[11] Rajendra Bhatia, *Positive Definite Matrices*, Princeton University Press, USA, 2015.

[12] Nicolas Boumal, *An introduction to optimization on smooth manifolds*, Cambridge University Press, 2023.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.

[14] Thibault de Surrel, Fabien Lotte, Sylvain Chevallier, and Florian Yger, "Wrapped gaussian on the manifold of symmetric positive definite matrices," in *Forty-second International Conference on Machine Learning*, 2025.

[15] Dong C. Liu and Jorge Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989.

[16] Gary Bécigneul and Octavian-Eugen Ganea, "Riemannian adaptive optimization methods," 2019.

[17] Michael Tangermann, Klaus-Robert Müller, Ad Aertsen, Niels Birbaumer, Christoph Braun, Clemens Brunner, Robert Leeb, Carsten Mehring, Kai J. Miller, Gernot R. Müller-Putz, Guido Nolte, Gert Pfurtscheller, Hubert Preissl, Gerwin Schalk, Alois Schlögl, Carmen Vidaurre, Stephan Waldert, and Benjamin Blankertz, "Review of the bci competition iv," *Frontiers in Neuroscience*, vol. 6, pp. 55, 2012.

[18] Bangyan Zhou, Xiaopei Wu, Zhao Lv, Lei Zhang, and Xiaojin Guo, "A fully automated trial selection method for optimization of motor imagery based brain-computer interface," *PLOS ONE*, vol. 11, no. 9, pp. 1–20, 09 2016.

[19] Bruno Aristimunha, Igor Carrara, Pierre Guetschel, Sara Sedlar, Pedro Rodrigues, Jan Sosulski, Divyesh Narayanan, Erik Bjareholt, Quentin Barthelemy, Robin Tibor Schirrmeister, Reinmar Kobler, Emmanuel Kalunga, Ludovic Darmet, Cattan Gregoire, Ali Abdul Hussain, Ramiro Gatti, Vladislav Goncharenko, Jordy Thielen, Thomas Moreau, Yannick Roy, Vinay Jayaram, Alexandre Barachant, and Sylvain Chevallier, "Mother of all bci benchmarks," 2025.

[20] Thibault de Surrel, Sylvain Chevallier, Fabien Lotte, and Florian Yger, "Geometry-Aware visualization of high dimensional Symmetric Positive Definite matrices," *Transactions on Machine Learning Research Journal*, Feb. 2025.

A. APPENDIX SPD SCALING

$$\delta_r(\Sigma_0, \Sigma) = ||\log(\Sigma_0^{-1/2}\Sigma\Sigma_0^{-1/2})||_F$$

$$\Sigma_s = \exp_{\Sigma_0}(\alpha \log_{\Sigma_0}(\Sigma))$$

we look for α such that:

$$\delta_r(\Sigma_0, \Sigma_s) = \epsilon$$

$$\delta_r(\Sigma_0, \Sigma_s) = ||\log(\Sigma_0^{-1/2}\exp_{\Sigma_0}(\alpha \log_{\Sigma_0}(\Sigma))\Sigma_0^{-1/2})||_F$$

$$\delta_r(\Sigma_0, \Sigma_s) = ||\log(\Sigma_0^{-1/2}\Sigma_0^{1/2}\exp(\Sigma_0^{-1/2}\Sigma_0^{1/2}\alpha\log(\Sigma_0^{-1/2}\Sigma\Sigma_0^{-1/2})\Sigma_0^{1/2}\Sigma_0^{-1/2})\Sigma_0^{1/2}\Sigma_0^{-1/2})||_F$$

$$\delta_r(\Sigma_0, \Sigma_s) = ||\log(\exp(\alpha\log(\Sigma_0^{-1/2}\Sigma\Sigma_0^{-1/2}))||_F$$

$$\delta_r(\Sigma_0, \Sigma_s) = ||\alpha\log(\Sigma_0^{-1/2}\Sigma\Sigma_0^{-1/2})||_F$$

$$\delta_r(\Sigma_0, \Sigma_s) = |\alpha|\ ||\log(\Sigma_0^{-1/2}\Sigma\Sigma_0^{-1/2})||_F = |\alpha|\delta_r(\Sigma_0, \Sigma) = \epsilon$$

$$\alpha = \frac{\epsilon}{\delta_r(\Sigma_0, \Sigma)}$$

B. APPENDIX SYNTHETIC DATA

To better illustrate our attack, we first test it on synthetic data generated from wrapped Gaussian distributions. Each class is constructed as the sum of two wrapped Gaussians, with one Gaussian shared across all classes to create overlap. The centers of these Gaussians are obtained by generating random multivariate signals and computing their covariance matrices, which serve as the class prototypes. The variance of the common wrapped Gaussian is fixed at $\sigma = 50$. For the class-specific wrapped Gaussians, σ is varied in the range $[10, 100]$, and R-PGD attacks are computed for each setting.

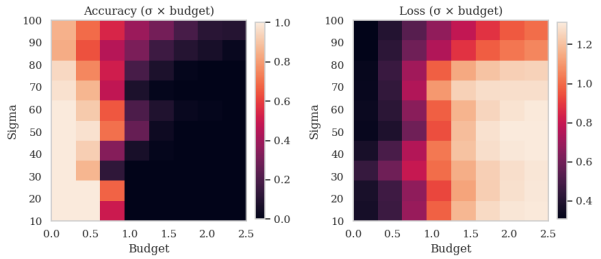


Fig. 3: Loss and accuracy metrics of the R-PGD attacks on synthetic data constructed

The figure 3 shows that more we increase the variances of the dataset, more the trained model will be robust. We clearly observe that higher the variance is, harder it will be to attack the model.