LARGE-SCALE INDEPENDENT COMPONENT ANALYSIS BY SPEEDING UP LIE GROUP TECHNIQUES

Matthias Hermann *Georg Umlauf* *Matthias O. Franz*

HTWG Konstanz, Konstanz, Germany

ABSTRACT

We are interested in computing a mini-batch-capable end-to-end algorithm to identify statistically independent components (ICA) in large scale and high-dimensional datasets. Current algorithms typically rely on pre-whitened data and do not integrate the two procedures of whitening and ICA estimation. Our online approach estimates a whitening and a rotation matrix with stochastic gradient descent on centered or uncentered data. We show that this can be done efficiently by combining Batch Karhunen-Löwe-Transformation [1] with Lie group techniques. Our algorithm is recursion-free and can be organized as feed-forward neural network which makes the use of GPU acceleration straight-forward. Because of the very fast convergence of Batch KLT, the gradient descent in the Lie group of orthogonal matrices stabilizes quickly. The optimization is further enhanced by integrating ADAM [2], an improved stochastic gradient descent (SGD) technique from the field of deep learning. We test the scaling capabilities by computing the independent components of the well-known ImageNet challenge (144 GB). Due to its robustness with respect to batch and step size, our approach can be used as a drop-in replacement for standard ICA algorithms where memory is a limiting factor.

Index Terms— ICA, Lie group, ADAM

1. INTRODUCTION

Independent component analysis (ICA) is a statistical signal processing technique for identifying statistically independent linear components [3]. Compared to principal component analysis, which only takes second-order statistics into account, ICA also incorporates higher-order moments such as skewness and kurtosis. Performing ICA in large-scale scenarios is of particular interest as more and more large high-resolution datasets become available such as, e.g., ImageNet or LiDar or RGBD-Video [4]. In this work we are targeting the case where the data $\mathbf{X}$ does not fit into memory, and stochastic optimization for an efficient GPU-based neural network implementation is required. Here, stochastic optimization means that the data $\mathbf{X}$ is processed in mini-batches during gradient descent instead of computing the gradient over the entire dataset. This use case is of special interest during data exploration when the number of extracted components is large and the redundancy in the data is unknown. Current ICA algorithms based on Lie-Group techniques use L-BFGS (Broyden–Fletcher–Goldfarb–Shanno optimization) and rely on a costly full-batch line search, which makes them very accurate but slow in practice [5]. On the other hand, algorithms based on Infomax [6], also referred to as maximum likelihood estimation of the ICA model, have good convergence rates [7], but are known to be hard to optimize in online scenarios [8]. Both aspects prevent current ICA algorithms from computing ICA on large high-dimensional datasets, so there is a need for a better optimization scheme for the entire ICA pipeline. Our contributions to that pipeline (see Figure 1) are: (1) showing the importance of orthogonality constraints in large-scale ICA and the pre-whitening requirement for stability, (2) improving the geodesic flow update rule by using the ADAM optimizer in combination with the Caley approximation for the matrix exponential, and (3) demonstrating the resulting scaling capabilities by computing the first 484 independent components of the ImageNet challenge. By using b-sized mini-batches the space complexity of the entire pipeline for k components is limited to $\mathrm{O}(d(k+b))$.

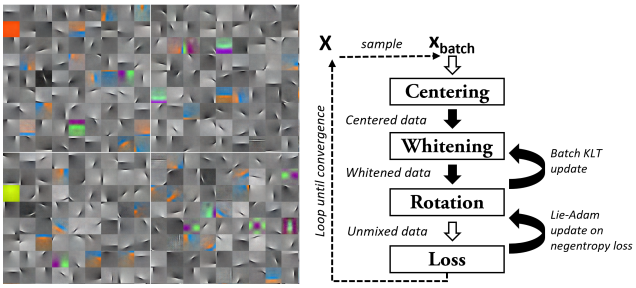

Fig. 1: Examples of the first 484 independent components estimated from the ImageNet dataset ($1.2 \cdot 10^6$ examples) (left). Every tile represents a single column of the mixing matrix which is reshaped to $3 \times 200 \times 200$ for illustration purpose. In Lie-ADAM, we used a learning rate of 0.01 and a batch size of 484. The model was trained with three runs through the dataset which took $3h$ on standard hardware with a single GPU. A schematic overview of the algorithm is shown on the right.

SUPPORTED BY FEDERAL MINISTRY OF EDUCATION AND RESEARCH (BMBF GRANT NO: 13N14540, 01IS19083A)

2. LARGE-SCALE ICA

In short, Independent component analysis (ICA) computes the matrix decomposition $\mathbf{X} = \mathbf{A}\,\mathbf{S}$, with mixing matrix $\mathbf{A}$ and independent sources arranged as columns of $\mathbf{S}$. The following section describes the relevant algorithmic decisions for conducting ICA in large-scale scenarios on CPU and GPU-hardware using as few hyperparameters as possible.

2.1. Optimizing independence

ICA is typically done in two steps: first, the data are whitened, and second, an orthogonal transformation $\mathbf{R}$ is chosen such as to maximize the independence between the components in $\mathbf{S}$ [3]. In general, the independence between several random variables $s_1, \ldots, s_k$ is measured by the mutual information $I(s_1, \ldots, s_k)$. A property of orthogonal transformations is that they do not change the shape and consequently not the differential entropy of a distribution. Hence, instead of the multivariate mutual information, we can minimize the differential entropy of the individual decorrelated components [9], [10]. Negentropy approximations $J(s)$ [3] are based on the maximum entropy principle and measure non-Gaussianity [5], [11]. The mutual information is then approximated by

$$I(s_{1:k}) \propto -\sum_i J(s_i) \propto \sum_i (\mathbb{E}[\mathrm{G}(s_i)] - \mathbb{E}[\mathrm{G}(\mathbf{z})])^2 \quad (1)$$

It measures the difference between the expectation of the whitened component s_i and a Gaussian variable $z \sim \mathcal{N}(0, 1)$ under the specified source model $\mathrm{G}(s_i)$. Here, we chose the traditional generalized Kurtosis measure [5], [11] for $\mathrm{G}(\cdot)$

$$\mathrm{G}(s_i) = -\frac{1}{0.99} \exp(-0.99 \frac{s_i^2}{2}) \quad (2)$$

As $\mathbb{E}[\mathrm{G}(\mathbf{z})]$ is a constant, optimization is achieved by either maximizing or minimizing $\mathbb{E}[\mathrm{G}(s_i)]$. The desired direction depends on the statistics of the components s_i and the underlying source model $\mathrm{G}(s_i)$. If unknown, automatically switching between maximizing or minimizing $\mathbb{E}[\mathrm{G}(s_i)]$ is needed. Extended Infomax [12] and Picard-O [7], for instance, achieve this by computing the component-wise kurtosis K_i based on a generic stability analysis. For simplicity we use the component-wise kurtosis K_i for estimating the sign [13].

2.2. Parallelism and GPU-hardware

For using highly parallel GPU-hardware, we need to reduce non-parallel computing steps in the algorithm. In existing algorithms, there are two major steps involved that complicate parallelism. The first is line search which is part of L-BFGS, for instance, as this includes a single-threaded loop iterating over the gradient direction. Second, greedy approaches as in Incremental PCA [14] and Robust ICA [15] are inherently recurrent and therefore suboptimal for parallelism. This is because, after extracting a single component, the orthogonality constraint must be enforced in order to prevent the algorithm from extracting the same components multiple times.

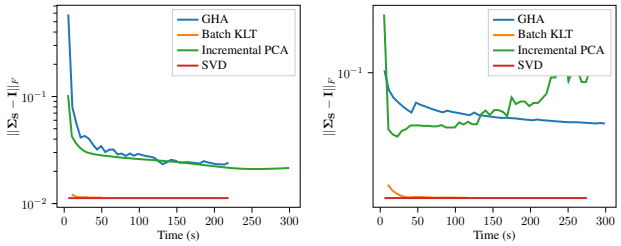

Fig. 2: Convergence of the 100×100 (left) and 1000×1000 (right) covariance matrix $\mathbf{\Sigma_S}$ pertaining to the largest eigenvalues to the identity matrix as measured by the Frobenius norm $||.||_F$ on the CIFAR10 dataset (50.000 examples of size $3 \times 32 \times 32$). As baseline, we show Singular Value Decomposition (SVD) which runs offline. Batch size was set to 100 and 1000, respectively.

2.3. High input dimensionality

High input dimensionality impacts the ICA algorithm heavily. Standard algorithms compute a *square* mixing matrix, which in the case of ImageNet would require storing 150528^2 parameters requiring 90 Gb in single precision for $224 \times 224 \times 3$ RGB image input. This is especially a limiting factor in GPU-computing, where the maximum available memory is comparatively small. Such large parameter matrices also considerably impact the stability of the algorithms. For mitigation, data dimension is typically reduced beforehand by applying PCA which is also needed for whitening [11]. Dimension reduction allows for *non-square* unmixing, in which fewer independent components than the number of input dimensions are unmixed. There are several algorithms for performing PCA in online scenarios available [16]. Here, we focus on algorithms that also work with mini-batches: (1) Batch Karhunen-Loewe-Transform (Batch KLT) [17], (2) Incremental PCA [14], and (3) the Generalized Hebbian Algorithm (GHA) [18]. As whitening is a mandatory step also in our ICA procedure, we compared the convergence of these algorithms on the well-known CIFAR10 dataset. Figure 2 shows clearly that Batch KLT is superior for stochastic whitening in this scenario. The learning rate in the GHA is a critical parameter and we found that a value of 10^{-5} works well.

2.4. High output dimensionality

When the input dimensionality is large, the output dimensionality of the algorithm can be large, too. This impacts the stability of the algorithm as the gradient update is done in a very high-dimensional parameter space. We are targeting scenarios where the number of ICs k is between 500 and 1000

components as this is a typical layer width in deep learning. Algorithms based on Infomax perform gradient update in the entire parameter space of square invertible matrices (the general linear group $GL(k)$) by maximizing the output entropy [19]. Optimization in higher dimensions causes problems as this space is not compact and includes diverging series [20]. For improving stability, orthogonality constraints are integrated [7], [21], and the search space is limited to the special orthogonal group $SO(k)$ [22]. This group has three appealing properties for our scenario: first, the number of parameters is reduced to $k(k-1)/2$; second, the special orthogonal group is compact, and hence there are no diverging series; third, $SO(k)$ is a Lie group with a tangent space, called the Lie algebra that can be used for gradient updates. The so-called *gradient flow* [22], [23] makes use of this and computes ICA with orthogonality constraints using Lie group techniques[1]. The corresponding Lie algebra is called $\mathfrak{so}(k)$. It consists of all skew-symmetric matrices and becomes the space for gradient descent. Every skew-symmetric matrix $\mathbf{\Theta}$ can be uniquely parameterized by a vector $\mathbf{r}$ of dimension $k(k-1)/2$ giving rise to a vector space. The components of that vector are called Plücker coordinates. Further, every skew-symmetric matrix $\mathbf{\Theta}$ can by related to an orthogonal matrix $\mathbf{R}$ by

$$\mathbf{R} = \exp(\mathbf{\Theta}), \tag{3}$$

where $\exp(\cdot)$ is the matrix exponential. The gradient of the loss function $\nabla_{\mathbf{R}_i} I$ represents an infinitesimal rotation and hence an element of $\mathfrak{so}(k)$. However, in order to compute a valid gradient step beyond the neighborhood of $\mathbf{R}_i$, the gradient direction needs to be expressed by the Lie bracket. We refer to Plumbley [22] for the full derivation of the relation between the two gradient expressions using the commutator

$$\nabla_{\mathbf{\Theta}_{\mathbf{r}_i}} I = (\nabla_{\mathbf{R}_i} I)^T \mathbf{R}_i - \mathbf{R}_i^T (\nabla_{\mathbf{R}_i} I)\,. \tag{4}$$

From there, we compute the corresponding parameter vector $\mathbf{r}_i$ by taking the upper triangular matrix of $\nabla_{\mathbf{\Theta}_{\mathbf{r}_i}} I$ corresponding to the Plücker coordinates. The *gradient flow* update rule is then given by

$$\mathbf{R}_{i+1}^T = \exp(-\eta\, \mathbf{\Theta}_{\mathbf{r}_i})\, \mathbf{R}_i^T, \tag{5}$$

with step size η and the skew-symmetric matrix $\mathbf{\Theta}_{\mathbf{r}_i}$ parametrized by $\mathbf{r}_i$ at the i-th iteration step. Unfortunately, rotation matrices for $k > 2$ are not commutative. Hence we cannot make additive steps of descent in $\mathfrak{so}(k)$ and need to map between the Lie algebra and the manifold in every iteration. This procedure is not for free, as we need to evaluate the matrix exponential between every iteration. Additionally, in order to update the solution at the current iterate $\mathbf{r}_i$, we need to *translate* the geodesic gradient direction by matrix multiplication. The accuracy and speed of the method depends critically on the computation of the matrix exponential and the overall number of iterations. In principle, the exponential can be computed by spectral decomposition and computing the exponential of the eigenvalues. Instead, the Padé-algorithm [24] can be used, which gives a significant speedup over spectral decomposition. Furthermore, we also tested the Cayley approximation, which is considerably faster and almost as accurate for most practical problems with $k < 1000$, and therefore our choice for being included in the algorithm (see Fig. 3)

[1] An improved method integrates L-BFGS for acceleration [5] .

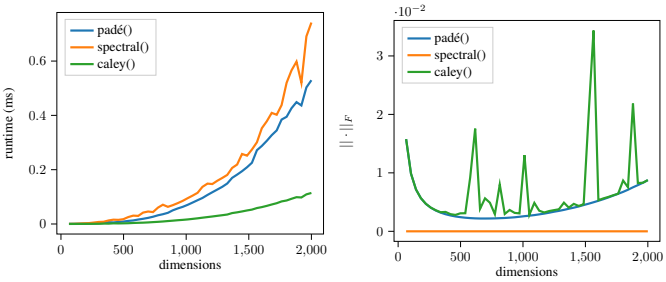

Fig. 3: Runtime and precision of the matrix exponential methods comparison between spectral $\exp(\mathbf{M})$, cayley$(\mathbf{M})$, padé$(\mathbf{M})$ for $\mathbf{M} \sim \mathcal{N}(0, 0.1)$ using PyTorch. Interestingly both the Cayley approximation cayley$(\mathbf{M}) \approx (\mathbf{I} - \frac{\mathbf{M}}{2})^{-1}(\mathbf{I} - \frac{\mathbf{M}}{2})$ and the Padé algorithm give similar accuracies (10^{-4}) up to 500 dimensions. However, for larger dimensions the Caley approximation shows large peaks.

2.5. Large datasets

As the loss function is computed as a sum over all input examples, performing parameter updates by gradient descent is very costly. As a consequence, we cannot use offline or full-batch algorithms and need to switch to online or mini-batch processing. Majorization-minimization ICA (MM) [25] is the most recent algorithm for estimating ICA online that guarantees convergence through an expectation-maximization (EM) scheme. Mini-batch processing induces a source of stochasticity impacting the gradient estimation and its variance. This is problematic for methods that rely on computing optimal step sizes per iteration by line-search as they become strongly sub-optimal. Momentum-based methods estimate curvature information by averaging over past mini-batches, which is more robust and allows a significant speedup [26]. Such methods arose from deep learning, where both the number of training examples and the number of parameters are large. Scarpiniti, Scardapane, Comminiello, *et al.* [26] already integrated the well-known ADAM algorithm into Infomax and improved its speed for low-dimensional problems (d <= 5). This kind of optimization is especially attractive in non-convex problems or when only a stochastic estimate of the gradient is available and hence line-search techniques are unreliable. However, in our experiments this approach did not convergence for high dimensional data. As already mentioned, we are optimizing

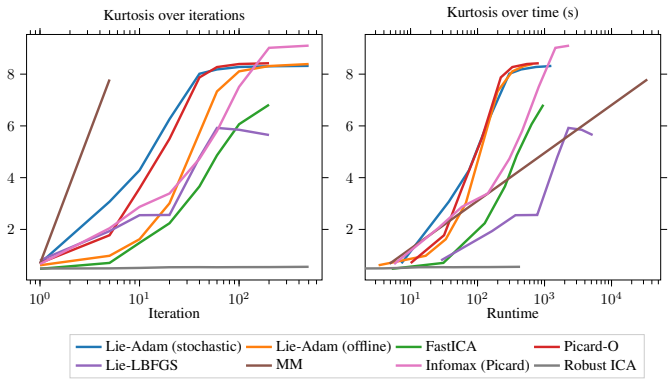

Fig. 4: The first plot shows the evolution of kurtosis of the computed ICs measured over iterations, the second plot over runtime for the STL10 dataset (10^5 examples of size $3 \times 96 \times 96$). The learning rates are 0.01 in the offline scenario, and 0.001 for the stochastic scenario. The batch size and k is 484.

in the Lie algebra of skew-symmetric matrices, which are parametrized by the Plücker coordinate vector $\mathbf{r}$. The main advantage of its vector space structure is the fact that we can optimize the components of $\mathbf{r}$ independently of each other. We leverage this property and use ADAM [2] to control the learning rate per vector component. Hence, the update rule becomes

$$\mathbf{R}_{i+1}^T = \exp(-\boldsymbol{\Theta}_{\hat{\mathbf{r}}_i})\,\mathbf{R}_i^T, \tag{6}$$

where $\hat{\mathbf{r}}_i = \mathbf{H}_{\text{adam}}\mathbf{r}_i$ is the new coordinate vector scaled by the diagonal matrix $\mathbf{H}_{\text{adam}}$ estimated by ADAM. The diagonal terms represent estimated curvature information.

The final optimization objective problem is

$$\text{argmin}_{\mathbf{R}}\, I(\mathbf{S}) = \sum_j^N \sum_i^k \text{sign}(K_i)\text{G}(s_{ij}), \tag{7}$$

$$\text{where } \mathbf{S} = (\mathbf{X}\,\mathbf{W}\,\mathbf{R})^T, \tag{8}$$

where $\mathbf{X}$ is the centered data matrix and $\mathbf{W}$ is the whitening matrix estimated by Batch KLT. In scenarios without access to the whole dataset column-wise centered input data is sometimes impossible. To account for this, we compute the column-wise mean μ incrementally for every example x_i during the first epoch by using the well-known formula

$$\mu_i = \mu_{i-1} + \frac{x_i - \mu_{i-1}}{i} \tag{9}$$

and keep it fixed afterwards. In the following, we refer to our approach as *Lie-Adam*.

3. EXPERIMENTS

We compared Lie-Adam[2] with the following ICA algorithms: For FastICA [11], we used the parallel implementation of *scikit-learn*, for L-BFGS with Lie-Group techniques [5] we used Lie-Adam and replaced ADAM by L-BFGS, for Picard-O [7] and Majorization-minimization ICA (MM) [25], we used the Python packages provided by the authors, for Infomax we used the Picard-O implementation [26] and Lie-Adam with the Infomax update rule. We ran all experiments with single-precision on standard hardware consisting of an Intel i7 7800k, 32 GB RAM and a 16 GB GeForce 1080Ti. Figure 4 shows the kurtosis on validation data for comparison. When looking at the plots, we see the superiority of Lie-Adam, both in offline and stochastic mode, and Picard-O. Figure 1 shows the computed ICs for the ImageNet dataset. Note that this is not trivial as this requires to store and process 1.2 million data samples, which requires 144 GB of RAM. In Lie-Adam, we used a learning rate $lr = 0.01$ and batch size $bs = 484$ and trained the model for three runs through the dataset, which took $3h$ on our hardware.

[2]Lie-Adam is provided as Python package on Github: https://github.com/matherm/Lie-Adam.git

4. CONCLUSION

We investigated the task of computing independent components in large scale scenarios using accelerated Lie group techniques in combination with Batch KLT. The combination of fast whitening, a stable gradient descent and good convergence rates is a huge improvement for scaling up current Lie-group-based ICA techniques. The proposed Lie-Adam approach offers two modes: in offline mode, it works very similar to L-BFGS-based algorithms like Picard-O [7]. However, in stochastic mode, it offers fast convergence rates while maintaining highly accurate solutions. We showed that the computation can be done efficiently without computing the matrix-exponential explicitly. The derivatives for gradient descent are computed in the corresponding Lie-algebra of the special orthogonal group, which turned out to be well-suited for the use of ADAM [2] in combination with mini-batches. Our approach allows us to formulate the ICA update equations as a standard neural network and leveraging recent approaches in deep learning. For proving the capabilities of our algorithm, we extracted independent components of ImageNet which is a 144 GB Dataset on standard hardware. To our knowledge, we are the first who present ICs of this popular large-scale dataset. The ability to efficiently compute ICA enables further research directions: First, ICA for large-scale datasets like video and neural network data. Second, as our model is a fully-featured neural network, it makes the integration of recent autoencoder techniques for source modeling straightforward.

5. REFERENCES

[1] A. Levy and M. Lindenbaum, "Sequential karhunen-loeve basis extraction and its application to images," in *Proceedings 1998 International Conference on Image Processing. ICIP98*, IEEE, vol. 2, 1998, pp. 456–460.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[3] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[4] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[5] S. E. Selvan, A. Mustatea, C. C. Xavier, and J. Sequeira, "Accurate estimation of ICA weight matrix by implicit constraint imposition using Lie group," *IEEE transactions on neural networks*, vol. 20, no. 10, pp. 1565–1580, 2009.

[6] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[7] P. Ablin, J.-F. Cardoso, and A. Gramfort, "Faster ICA under orthogonal constraint," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4464–4468.

[8] J. Montoya-Martínez, J.-F. Cardoso, and A. Gramfort, "Caveats with stochastic gradient and maximum likelihood based ICA for EEG," in *International Conference on Latent Variable Analysis and Signal Separation*, Springer, 2017, pp. 279–289.

[9] J Hérault and B ANS, "Circuits neuronaux á synapses modifiables: Décodage de messages composites par apprentissage non supervisé [neuronal circuits with modifiable synapses: Decoding composite messages by unsupervised learning]," *Comptes Rendus de l'Académie des Sciences*, vol. 299, pp. 525–528, 1984.

[10] J. Hérault, C. Jutten, and B. Ans, "Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé," in *10 Colloque sur le traitement du signal et des images, FRA, 1985*, GRETSI, Groupe d'Etudes du Traitement du Signal et des Images, 1985.

[11] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[12] T.-W. Lee and T. J. Sejnowski, "Independent component analysis for mixed subgaussian and super-gaussian sources," in *Technical report, Computational Neurobiology Lab, The Salk Institute, La Jolla*, Citeseer, 1998.

[13] M Girolami, "Self-organizing artificial neural networks for signal separation," *Unpublished Ph. D. dissertation, Paisley University, Scotland*, 1997.

[14] M. Brand, "Incremental singular value decomposition of uncertain data with missing values," in *European Conference on Computer Vision*, Springer, 2002, pp. 707–720.

[15] V. Zarzoso and P. Comon, "Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size," *IEEE Transactions on neural networks*, vol. 21, no. 2, pp. 248–261, 2009.

[16] R. Arora, A. Cotter, K. Livescu, and N. Srebro, "Stochastic optimization for PCA and PLS," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2012, pp. 861–868.

[17] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International journal of computer vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[18] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural networks*, vol. 2, no. 6, pp. 459–473, 1989.

[19] J. Cardoso, "Infomax and maximum likelihood for blind source separation," *IEEE Signal Processing Letters*, vol. 4, no. 4, pp. 112–114, 1997.

[20] S. Fiori and Y. Bengio, "Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial.," *Journal of Machine Learning Research*, vol. 6, no. 5, 2005.

[21] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[22] M. D. Plumbley, "Geometrical methods for non-negative ICA: Manifolds, Lie groups and toral subalgebras," *Neurocomputing*, vol. 67, pp. 161–197, 2005.

[23] Y. Nishimori, "Learning algorithm for independent component analysis by geodesic flows on orthogonal group," in *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, IEEE, vol. 2, 1999, pp. 933–938.

[24] A. H. Al-Mohy and N. J. Higham, "A new scaling and squaring algorithm for the matrix exponential," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 970–989, 2010.

[25] P. Ablin, A. Gramfort, J.-F. Cardoso, and F. Bach, "Stochastic algorithms with descent guarantees for ICA," *Proceedings of Machine Learning Research*, 2019.

[26] M. Scarpiniti, S. Scardapane, D. Comminiello, R. Parisi, and A. Uncini, "Effective blind source separation based on the ADAM algorithm," in *Multidisciplinary Approaches to Neural Computing*, Springer, 2018, pp. 57–66.