Decomposer Networks: Deep Component Analysis and Synthesis

Mohsen Joneidi
mohsen.joneidi@outlook.com

Abstract

We propose the *Decomposer Networks* (DecompNet), a semantic autoencoder that factorizes an input into multiple interpretable components. Unlike classical autoencoders that compress an input into a single latent representation, the Decomposer Network maintains N parallel branches, each assigned a residual input defined as the original signal minus the reconstructions of all other branches. By unrolling a Gauss–Seidel style block-coordinate descent into a differentiable network, DecompNet enforce explicit competition among components, yielding parsimonious, semantically meaningful representations. We situate our model relative to linear decomposition methods (PCA, NMF), deep unrolled optimization, and object-centric architectures (MONet, IODINE, Slot Attention), and highlight its novelty as the first semantic autoencoder to implement an *all-but-one residual update rule*.

1 Introduction

Human creativity often begins with decomposition: breaking complex experiences into their essential parts. A skilled chef separates flavors, a painter distinguishes tones and textures, a musician isolates harmonies, and a mathematician dissects structures into simpler forms. This ability to analyze the whole through its components is at the heart of deep understanding. In mathematics, the singular value decomposition—introduced almost 150 years ago—embodied this principle, providing a powerful way to separate a matrix into fundamental elements with elegant and useful properties. Today, as we enter the era of artificial intelligence, the challenge is to equip machines with a comparable capacity for structured, component-wise reasoning. To this end, we introduce Decomposer Networks, a neural architecture designed to extend the spirit of SVD into the nonlinear and semantic domain of AI.

Decomposition of data into semantic components is a longstanding goal in signal processing and representation learning. Classical methods such as PCA and NMF provide additive factorization but are restricted to linear settings. Autoencoders and variational autoencoders capture nonlinear structure, yet entangle semantics within a single latent vector. Object-centric models introduce multi-slot representations, but rely on masking and attention rather than residual explain-away.

We propose DecompNet, a semantic autoencoder that assigns each branch a residual view of the input, enforcing specialization and interpretability. This architecture bridges the gap between explain-away principles from sparse coding and modern deep neural factorization. Related work are listed below:

Linear and Shallow Decomposition. Classical approaches such as PCA, ICA, and NMF [Jolliffe, 2002, Hyvärinen and Oja, 2000, Lee and Seung, 1999] provide additive decompositions but remain linear.

Deep Unrolled Factorization. Works such as LISTA [Gregor and LeCun, 2010], ADMM-Net [Yang et al., 2016], and deep NMF [Trigeorgis et al., 2016] unroll optimization into neural updates. They lack the residual competition mechanism we propose.

Object-Centric Scene Decomposition. Models such as MONet [Burgess et al., 2019], IODINE [Greff et al., 2019], and Slot Attention [Locatello et al., 2020] decompose inputs into slots using masking and attention. Our method instead employs residual subtraction to enforce explain-away dynamics.

Residual Factorization in Networks. Factorized residual units [Chen et al., 2017] improve efficiency, but focus on parameter sharing rather than semantic decomposition.

2 Relation to Classic Decompositions

To highlight the connection between Decomposer Networks and classical linear factorization, we consider a simplified setting in which each branch is a purely linear operator, i.e. $F_i(r) = W_i r$ and $S_i(y_i) = V_i y_i$, with $W_i \in \mathbb{R}^{k \times d}$

and $V_i \in \mathbb{R}^{d \times k}$. The overall reconstruction after one sweep is therefore

$$\hat{x} = \sum_{i=1}^{N} V_i W_i r_i, \tag{1}$$

where the residual r_i is defined as x minus the reconstructions of the other components.

Rank-one initialization. Assume each branch is initialized as a rank-one linear operator:

$$V_i W_i = u_i v_i^\top,$$

where $u_i, v_i \in \mathbb{R}^d$ are drawn at random with unit norm. Thus, each branch initially captures a one-dimensional projection of the input.

Iteration dynamics. During Gauss–Seidel sweeps, branch i is updated on the residual

$$r_i^{(t)} = x - \sum_{j \neq i} u_j v_j^\top x,$$

which ensures that $r_i^{(t)}$ lies in the orthogonal complement of the subspace spanned by $\{u_j v_j^\top\}_{j \neq i}$ up to reconstruction error. Applying $v_i^\top$ extracts the dominant direction in that residual, and updating u_i aligns it with $r_i^{(t)}$.

Connection to SVD. This procedure is mathematically equivalent to *deflation methods* for singular value decomposition. Classical deflation iteratively subtracts rank-one approximations from a matrix or signal until convergence, with each step converging to the next singular component [Golub and Van Loan, 2013]. In our setting, each Gauss–Seidel update performs an analogous step: the first branch converges to the dominant singular component (u_1, v_1), the second to the next (u_2, v_2), and so forth. After sufficient iterations, the collection of branches recovers the SVD of the input up to scaling and ordering.

Implication. Therefore, Decomposer Networks can be viewed as a *nonlinear extension* of SVD. In the linear case with rank-one subnetworks, they reduce to classical singular value decomposition via iterative deflation. In the general nonlinear case, they retain the explain-away residual dynamics but extend beyond linear manifolds, enabling decomposition into semantic components that need not be orthogonal or linear.

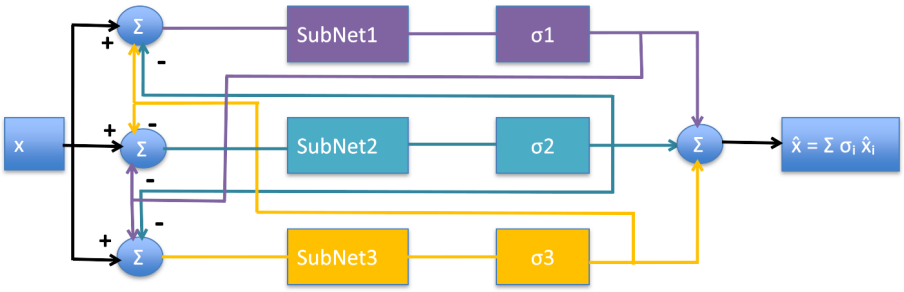

Figure 1: Decomposer Networks (3 components). Each residual summer adds x and subtracts the *other* branches' scaled reconstructions ($-\sigma_j \hat{x}_j$). Each color shows one component and colored arrows show components data; gains σ_i feed both the final sum and the residual feedback. Each SubNet can be as simple as a rank-1 multiplication or as deep as a multi layer auto-encoder.

3 Model and Cost Function

Given input $x \in \mathbb{R}^d$, the network learns N components $\{y_i\}_{i=1}^N$ via branch-specific encoders F_i and decoders S_i:

$$\hat{x}_i = S_i(y_i), \qquad \hat{x} = \sum_{i=1}^{N} \hat{x}_i. \tag{2}$$

Each branch i receives as input a residual defined by the reconstructions of all other branches:

$$r_i^{(t)} = x - \sum_{j \neq i} \hat{x}_j^{(t)}. \tag{3}$$

The branch update is then:

$$y_i^{(t)} = F_i(r_i^{(t)}), \qquad \hat{x}_i^{(t)} = S_i(y_i^{(t)}). \tag{4}$$

Iterating over $i = 1, \ldots, N$ for K sweeps yields a Gauss–Seidel style residual refinement. Training minimizes a composite loss:

$$\mathcal{L} = \|x - \hat{x}\|^2 + \lambda_s \sum_i \|y_i\|_1 + \lambda_\perp \sum_{i \neq j} \langle \hat{x}_i, \hat{x}_j \rangle^2,$$

with optional semantic heads to align each component to supervised labels.

Decomposer Networks are the first semantic autoencoders to implement an explicit *all-but-one residual update rule*. Each branch is forced to model what the others cannot, producing semantic disentanglement by design.

Compared to deep unrolled methods, our updates are residual-conditioned and sequential; compared to object-centric models, our decomposition arises from residual explain-away rather than attention masks.

4 Optimization and Learning

Setup. We are given a dataset $\mathcal{D} = \{x^{(n)}\}_{n=1}^{B}$ (mini-batch size B). The Decomposer Network contains N *autoencoders* (AEs). AE i has an encoder $E_i(\cdot;\theta_i)$ and decoder $D_i(\cdot;\phi_i)$ producing a component reconstruction

$$\hat{x}_i \;=\; D_i\big(E_i(r_i)\big),$$

where r_i is the *residual input* to AE i defined by

$$r_i \;=\; x \;-\sum_{j\neq i} \sigma_j\, \hat{x}_j. \tag{5}$$

The final reconstruction is the scaled sum

$$\hat{x} \;=\; \sum_{i=1}^{N} \sigma_i\, \hat{x}_i, \tag{6}$$

where $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]^\top$ are *per-sample* nonnegative scalars (analogous to singular values in SVD). We optionally perform K Gauss–Seidel sweeps over i to refine $\{\hat{x}_i\}$ (weights tied across sweeps unless otherwise noted).

4.1 Objective

For a mini-batch, we minimize

$$\mathcal{L} \;=\; \frac{1}{B}\sum_{n=1}^{B} \underbrace{\Big\|x^{(n)} - \sum_{i} \sigma_i^{(n)}\, \hat{x}_i^{(n)}\Big\|_2^2}_{\text{reconstruction}}$$

$$+\,\lambda_s \sum_i \|\mathbf{z}_i\|_1 \;+\; \lambda_\perp \sum_{i\neq j} \big\langle \hat{x}_i, \hat{x}_j \big\rangle^2 \;+\; \sum_i \lambda_{\text{sem},i}\, \mathcal{L}_{\text{sem},i}, \tag{7}$$

where $\mathbf{z}_i = E_i(r_i)$ are AE codes (sparsity promotes parsimony), the orthogonality/independence penalty $\lambda_\perp$ reduces component overlap, and $\mathcal{L}_{\text{sem},i}$ are optional semantic heads if supervision is available. All inner products, norms, and losses are computed per-sample then averaged over the batch.

4.2 Per-sample scaling coefficients $\boldsymbol{\sigma}$

For a fixed set of component reconstructions $\{\hat{x}_i\}_{i=1}^N$ (produced by the current AEs), the optimal scaling $\boldsymbol{\sigma}$ for *each* sample x solves a small nonnegative least-squares (NNLS):

$$\min_{\boldsymbol{\sigma}\geq 0} \ \big\|x - \mathbf{H}\,\boldsymbol{\sigma}\big\|_2^2, \qquad \text{with} \quad \mathbf{H} = [\hat{x}_1\ \hat{x}_2\ \cdots\ \hat{x}_N] \in \mathbb{R}^{d\times N}. \tag{8}$$

When nonnegativity is not enforced, the closed-form is

$$\boldsymbol{\sigma}^\star \ = \ (\mathbf{H}^\top\mathbf{H} + \varepsilon\mathbf{I})^{-1}\mathbf{H}^\top x, \tag{9}$$

with a tiny Tikhonov $\varepsilon > 0$ for stability. With NNLS, a fast projected gradient or active-set solver suffices because N is small. Importantly, we compute (9) (or NNLS) *independently for each sample* in the batch.

4.3 Alternating training (residual coordinate descent)

We alternate between updating the per-sample scalars $\boldsymbol{\sigma}$ and updating AE weights $\{\theta_i, \phi_i\}$. Each outer iteration uses one mini-batch.

Step A: Update $\boldsymbol{\sigma}$ (for each sample). Hold AE weights fixed. For each $x^{(n)}$, compute current components $\hat{x}_i^{(n)}$ by (5)–(6) (one or more Gauss–Seidel sweeps), form $\mathbf{H}^{(n)}$, then solve (8) (NNLS) or (9) to obtain $\boldsymbol{\sigma}^{(n)}$.

Step B: Update AE weights (one or more sweeps). Hold $\{\boldsymbol{\sigma}^{(n)}\}$ fixed. For each sweep $t = 1, \ldots, K$ and branch $i = 1, \ldots, N$:

$$r_i^{(n,t)} = x^{(n)} - \sum_{j\neq i} \sigma_j^{(n)}\,\hat{x}_j^{(n,t)} \quad \text{(Gauss–Seidel uses latest } \hat{x}_j\text{)}, \tag{10}$$

$$\mathbf{z}_i^{(n,t)} = E_i\big(r_i^{(n,t)};\theta_i\big), \qquad \hat{x}_i^{(n,t)} = D_i\big(\mathbf{z}_i^{(n,t)};\phi_i\big). \tag{11}$$

Accumulate the batch loss (7) with $\hat{x} = \sum_i \sigma_i^{(n)}\hat{x}_i^{(n,t)}$ and update $\{\theta_i, \phi_i\}$ by backpropagation (any first-order optimizer). Optionally use *relaxation* (damping) to improve stability:

$$\hat{x}_i^{(n,t)} \leftarrow (1-\alpha)\,\hat{x}_i^{(n,t-1)} + \alpha\,\hat{x}_i^{(n,t)}, \qquad \alpha \in (0,1].$$

Jacobi vs. Gauss–Seidel. Jacobi updates compute all r_i from the previous sweep (parallelizable on GPUs); Gauss–Seidel consumes freshest neighbors (often faster empirical convergence). Both are differentiable end-to-end.

4.4 Algorithm

Algorithm 1: Alternating training for Decomposer AEs with per-sample $\boldsymbol{\sigma}$
Input: batch $\{x^{(n)}\}_{n=1}^{B}$, AEs $\{E_i, D_i\}$ with weights $\{\theta_i, \phi_i\}$, sweeps K, damping α.
Repeat until convergence:

1. **(Forward components)** For each n: initialize $\hat{x}_i^{(n,0)} = 0$.
   For $t = 1..K$, for $i = 1..N$: form $r_i^{(n,t)}$ by (5), compute $\hat{x}_i^{(n,t)} = D_i(E_i(r_i^{(n,t)}))$, optionally relax with α.
2. **(Per-sample scales)** For each n: form $\mathbf{H}^{(n)} = [\hat{x}_1^{(n,K)}, \ldots, \hat{x}_N^{(n,K)}]$ and solve (8) (or (9)) for $\boldsymbol{\sigma}^{(n)}$.
3. **(Backprop AEs)** With $\{\boldsymbol{\sigma}^{(n)}\}$ fixed, recompute residual sweeps and minimize (7) by SGD/Adam w.r.t. $\{\theta_i, \phi_i\}$.

Output: trained $\{\theta_i, \phi_i\}$ and, at inference, per-sample $\boldsymbol{\sigma}$ via Step 2.

4.5 Gradients and practical notes

Backprop through residuals. In Step B, $\boldsymbol{\sigma}$ is held fixed; gradients flow through the residual construction (5) and through each AE. Because r_i depends on $\hat{x}_j$, a branch update indirectly influences others, which is precisely the desired competitive coupling.
Nonnegativity and normalization. Enforce $\sigma_i \geq 0$ either by NNLS, by a softplus parameterization $\sigma_i = \text{softplus}(\tau_i)$, or by projecting negative values to zero after (9). To avoid trivial rescalings, apply weight normalization to decoders or constrain $\|\hat{x}_i\|_2$ (e.g., divide by its norm inside $\mathbf{H}$ and absorb scale into σ_i).
Stability. Use small K initially (e.g., $K = 1$), then increase to 3–5. Damping $\alpha \in [0.3, 0.7]$ reduces "ping–pong" between branches. Orthogonality/independence penalties ($\lambda_\perp$) curb duplicate explanations.
Permutation symmetry. To prevent slot swapping, bias branches slightly differently (e.g., distinct receptive fields or weak semantic heads), or add mild diversity priors.
Inference. At test time, compute components via K sweeps and estimate $\boldsymbol{\sigma}$ per sample by (8); report both $\{\hat{x}_i\}$ and $\{\sigma_i\}$.

Potential use cases of DecompNet include:

- Time-series decomposition (trend, oscillatory modes, noise)

- Radar/communications (clutter vs. target vs. multipath separation)
- Images (structure vs. texture vs. illumination)
- Biomedical signals (e.g., ECG/EEG component separation)

5 Experimental Results

5.1 Dataset

All experiments were conducted on the **AT&T Faces Dataset** (formerly known as the ORL database) Samaria and Harter [1994]. The dataset contains 400 grayscale images of 40 subjects, each with variations in facial expression, pose, and illumination. Each image has an original resolution of 112×92 pixels, which was optionally downsampled to 56×46 for computational efficiency. All images were standardized to zero mean and unit variance per feature prior to training.

5.2 Experiment 1: Linear Decomposer Networks (Rank-1 Autoencoders)

In the first experiment, each subnetwork was parameterized by a *rank-1 projection operator* of the form $u_i u_i^T$. This model is equivalent to a shallow autoencoder with a single latent scalar coefficient. The Decomposer Network was trained on the standardized AT&T face dataset using the proposed iterative residual learning scheme and per-sample singular weights σ_i.

Despite being trained through gradient-based optimization, the learned projection directions $\{u_i\}$ converged to the principal directions of the dataset. This behavior is expected: under linearity and orthogonality constraints, the Decomposer Network minimizes the same objective as the Singular Value Decomposition (SVD) or Principal Component Analysis (PCA). As shown in Fig. 2, each component aligns with the dominant eigenvectors of the data covariance matrix, confirming that the architecture recovers PCA-like bases through unsupervised training. In three presented experiments, the first image is the original image and then five components are shown and the last image is combination of components.

5.3 Experiment 2: Unconstrained CNN Autoencoders

In the second experiment, the rank-1 restriction was removed and replaced with 3-layer convolutional autoencoders. Each subnetwork could now model

Figure 2: Experiment 1: Rank-1 linear subnetworks converge to PCA-like components on the AT&T dataset. The learned components resemble the top singular vectors of the data matrix.

nonlinear and spatially structured features. Without additional constraints, the subnetworks jointly learned overlapping but diverse reconstructions of the same input.

While the overall reconstruction $\hat{x} = \sum_i \sigma_i \hat{x}_i$ matched the input closely, the individual components $\hat{x}_i$ still exhibited global traces of the original face. This shows that, in the absence of explicit spatial or semantic disentanglement, the subnetworks collectively distribute the information but do not specialize in localized or interpretable features. The results, shown in Fig. 3, illustrate that the decomposition still captures multiple expressive modes of reconstruction even without orthogonality or region-based separation.

Figure 3: Experiment 2: CNN-based subnetworks without spatial constraints. Each component contributes differently to the reconstruction but all retain global image structure.

5.4 Experiment 3: Spatially Masked Decomposer Networks

To encourage spatial specialization and semantic disentanglement, the third experiment introduced fixed *Gaussian masks* before each autoencoder. The masks, defined over the image domain, were centered at random coordinates and designed such that each had a 0.5-level contour covering approximately half the image area. These masks modulated the input residual for each subnetwork, guiding each one to focus on a specific spatial region while preserving overlap at the boundaries.

This modification resulted in more interpretable decompositions: individual subnetworks captured localized facial attributes such as eyes, mouth, or shading patterns, while the aggregated reconstruction $\hat{x}$ remained faithful to the original. As shown in Fig. 4, the decomposition became semantically meaningful that represents a coherent spatial or textural region within the face. This suggests that fixed masking can impose structured priors that lead to human-interpretable subcomponents without explicit supervision.

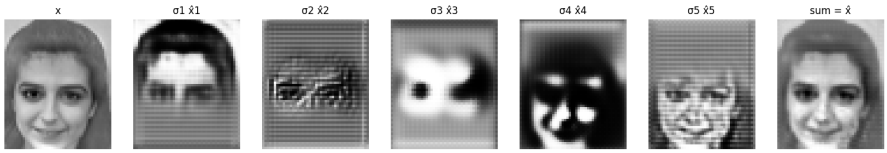

Figure 4: Experiment 3: Decomposer Networks with fixed Gaussian spatial masks. Each component captures a semantically meaningful subregion of the input image.

5.5 Summary

Across the three experiments, the proposed architecture demonstrated a progression from linear decomposition (recovering SVD/PCA) to nonlinear expressive components, and finally to semantically structured representations through spatial priors. This progression highlights the flexibility of Decomposer Networks as a unified framework bridging classic linear decomposition and modern deep feature factorization.

6 DecompNet for Synthesis and Control

Beyond analysis and decomposition, Decomposer Networks (**DecompNet**) naturally support *controlled synthesis*. Each input x is represented as a sum of learned semantic components, modulated by per-sample coefficients σ_i:

$$\hat{x} = \sum_{i=1}^{N} \sigma_i \hat{x}_i, \qquad \hat{x}_i = f_i(r_i),$$

where f_i denotes the ith subnetwork and r_i is its residual input. Since each component $\hat{x}_i$ corresponds to a coherent and interpretable substructure (spatial or conceptual), the coefficient σ_i can be interpreted as a *semantic control weight*. By modifying these weights after training, DecompNet can generate new samples that smoothly vary one semantic factor while keeping others fixed.

6.1 Semantic Factor Manipulation

In the linear case (Section 2), modifying σ_i scales the contribution of the corresponding principal component, akin to classic PCA synthesis. In the nonlinear and masked configurations, however, each $\hat{x}_i$ represents a learned nonlinear generator for a specific semantic attribute. For instance, one subnetwork may implicitly encode global illumination, another may capture facial expression, and a third may represent background shading. Adjusting σ_i for the "illumination" component allows us to *brighten* or *darken* the synthesized face without retraining the network or providing explicit attribute labels:

$$x_{\text{synth}} = \sum_i \tilde{\sigma}_i \hat{x}_i, \quad \tilde{\sigma}_j \neq \sigma_j \text{ for illumination factor } j.$$

This mechanism provides interpretable, low-dimensional control over the generated appearance while preserving image fidelity.

6.2 Relation to Controllable and Disentangled Generation

The concept of tuning σ_i aligns closely with efforts in disentangled representation learning and controllable generative modeling. In particular, DecompNet shares conceptual similarities with:

- **Disentangled VAEs** such as β-VAE Higgins et al. [2017] and FactorVAE Kim and Mnih [2018], which separate latent factors but rely on global latent variables rather than structured residual pathways.
- **GAN-based control models** like StyleGAN Karras et al. [2019], where latent style vectors modulate specific layers to affect semantic attributes such as color or lighting.
- **Object-centric generative models** such as MONet Burgess et al. [2019] and IODINE Greff et al. [2019], which iteratively reconstruct image regions and allow slot-wise manipulation. Unlike these models, DecompNet achieves component control without attention mechanisms or probabilistic inference; the control variable σ_i is explicitly interpretable and directly tied to the reconstruction weights.

6.3 Potential Applications

This controllable synthesis property enables DecompNet to serve as a semantic editing framework. After training on natural images, one could modify σ_i to:

- Adjust lighting or shading by tuning an "illumination" component.
- Manipulate expression intensity while keeping identity constant.
- Combine components from different images to create hybrid compositions (e.g., swapping background vs. facial texture).

Such controllable synthesis bridges classical linear component editing (as in PCA morphing) and modern interpretable generative modeling, offering a deterministic, explainable alternative to latent-space manipulation in VAEs and GANs.

6.4 Discussion

The synthesis behavior of DecompNet underscores its dual role as both an *analyzer* and a *synthesizer*. Because each subnetwork learns to reconstruct a specific residual aspect of the input, the learned $\{\hat{x}_i\}$ act as basis generators, while the coefficients $\{\sigma_i\}$ form a semantic coordinate system. In contrast to typical deep generative models, these coordinates are not latent abstractions but physically interpretable scaling factors of identifiable visual components. This property opens avenues for zero-shot semantic editing and for data-driven control in creative or scientific image synthesis applications.

7 Conclusion

We introduced **Decomposer Networks**, a semantic autoencoder based on residual all-but-one factorization. This model brings together the interpretability of classical decomposition and the expressiveness of deep neural networks, opening a new path for semantic disentanglement in complex domains. Decomposer Networks extend the concept of singular vectors and singular values to deep components and their contribution. As DecompNet becomes shallower, the components merge to principal components defined by SVD.

References

Christopher P. Burgess et al. Monet: Unsupervised scene decomposition and representation. In *Proc. NeurIPS*, 2019.

Yunpeng Chen, Jianan Li, Hanwang Hu, and Jiashi Wang. Sharing residual units through collective tensor factorization. In *Proc. AAAI*, 2017.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.

Klaus Greff, Raphael L. Kaufmann, et al. Iodine: Iterative object decomposition inference network. In *ICLR*, 2019.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proc. ICML*, 2010.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, et al. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2017.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *ICML*, 2018.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Francesco Locatello et al. Object-centric learning with slot attention. In *Proc. NeurIPS*, 2020.

F.S. Samaria and A.C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, pages 138–142. IEEE, 1994.

George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Björn W. Schuller. A deep semi-nmf model for learning hidden representations. In *Proc. ICML*, 2016.

Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. Deep admm-net for compressive sensing mri. In *Proc. NIPS*, 2016.