arXiv:1709.01112v1 [cs.IT] 4 Sep 2017

Optimal deep neural networks for sparse recovery via Laplace techniques

Steffen Limmer and Sławomir Stańczak

***Abstract*—This paper introduces Laplace techniques to design an optimal neural network for estimation of simplex-valued sparse vectors from compressed measurements. To this end, we recast the problem of MMSE estimation (w.r.t. a prescribed uniform input distribution) as centroid computation of some polytope that is an intersection of the simplex and an affine subspace determined by the measurements. Due to a specific structure, the centroid can be computed analytically by extending a recent result by Lasserre that facilitates the volume computation of polytopes via Laplace transformations. Interestingly, the computation of volume and centroid can be performed by a classical deep neural network comprising threshold functions, rectified linear (ReLU) and rectified polynomial (ReP) activation functions. The proposed construction of a deep neural network for sparse recovery is completely analytical, which allows for bypassing time-consuming training procedures. Furthermore, we show that the number of layers in our construction is equal to the number of measurements which might enable novel low-latency sparse recovery algorithms for a larger class of signals than that assumed in this paper. To assess the applicability of the proposed uniform input distribution, we showcase the recovery performance on samples that are soft-classification vectors generated by two standard datasets. As both volume and centroid computation are known to be computationally hard, the network width grows exponentially in the worst-case. However, the width may be reduced by inducing sparse connectivity in the neural network via a well-suited basis of the affine subspace. Finally, we point out that our analytical construction may serve as a viable initialization to be further optimized and trained using particular input datasets at hand.**

***Index Terms*—sparse recovery, compressed sensing, deep neural networks, convex geometry**

I. Introduction

Deep neural networks have achieved significant improvements in a series of tasks ranging from image classification to speech recognition [1].While training the underlying networks is extremely time-consuming, real-time inference has been shown to be feasible, which has rendered successful commercial applications as self-driving cars and realtime machine translation possible [1]. These findings took many researchers by surprise as the underlying problems were believed to be both computationally and theoretically hard. Yet, theoretical foundations of successful applications remain thin and analytical insights have only recently appeared in the literature. In this context, results on the translation and deformation stability of convolutional neural networks [2], [3] deserve a particular attention. However, the interplay between sparsity of input signals and particular network architectures has been so far only addressed in numerical studies [4], [5], [6] or based on shallow architectures [7]. This paper is a contribution towards closing this important gap in theory. To this end, we consider a simple sparse recovery problem, which is shown to admit an optimal solution by an analytically-designed neural network.

More precisely, we address the problem of estimating a compressible vector from a set of dimension reduced noiseless measurements via an M-layer deep neural network. The real-valued input vector $\boldsymbol{x} \in \mathbb{R}^N$ is assumed to be distributed uniformly over the standard simplex and is to be estimated from $M < N$ linear measurements $\boldsymbol{y} \in \mathbb{R}^M$ given by

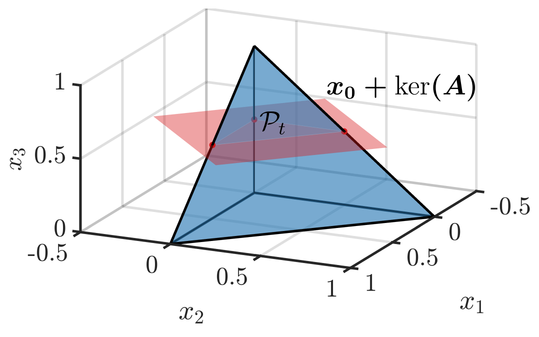
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}. \tag{1}$$

We refer the reader to Fig. 9 in Sec. VI for an illustration of a realization $\boldsymbol{x}$ of this stochastic process as well as markedly similar soft-classification vectors generated by standard datasets. Here and hereafter, the measurement matrix $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ is assumed to be an arbitrary fixed full-rank matrix. The problem of designing efficient algorithms for recovering $\boldsymbol{x}$ given $\boldsymbol{y}$ has been at the core of many research works with well-understood algorithmic methods, including ℓ_1-minimization [8] and iterative soft/hard-thresholding algorithms (ISTA / IHT) [8], [9]. Training structurally similar neural networks has been investigated in [6], [5], [4]; in particular, these works focused on the problem of fine-tuning parameters using stochastic gradient descent as a full search over all possible architectures (number of layers, activation function, size of the weight matrices) is infeasible. In contrast to these approaches, this paper introduces multidimensional Laplace transform techniques to obtain both the network architecture as well as the parameters in a fully analytical manner without the need for data-driven optimization. Interestingly, this approach reveals an analytical explanation for the effectiveness of threshold functions, rectified linear (ReLU) and rectified polynomial (ReP) activation functions. Moreover, we think that the resulting Laplace neural network can solve a larger class of sparse recovery problems than that assumed in this paper which is supported by a numerical study. To this end, the proposed Laplace neural network can be used as a viable initial model to be further optimized using conventional SGD-type techniques.

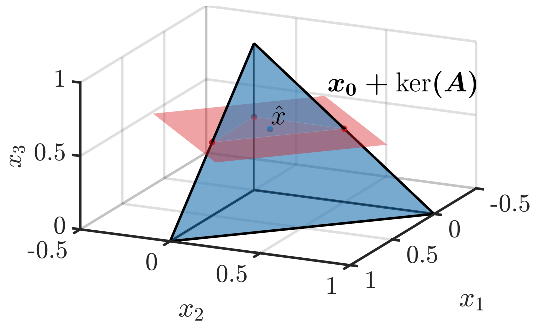
A. Notation and Some General Assumptions

Throughout the paper, the sets of reals, nonnegative reals and reals excluding the origin are designated by $\mathbb{R}$, $\mathbb{R}_+$ and $\mathbb{R}_{\neq 0}$, respectively. $\mathbb{S}^{N-1} \subset \mathbb{R}^N$ and $\Delta \subset \mathbb{R}^N$ denote the N-dimensional unit sphere and the standard simplex defined as $\Delta := \{\boldsymbol{x} \in \mathbb{R}_+^N : \sum_{n=1}^N x_n \leq 1\}$. We use lowercase, bold lowercase and bold uppercase serif letters x, $\boldsymbol{x}$, $\boldsymbol{X}$ to denote scalars, vectors and matrices, respectively. Sans-serif letters are

S. Limmer and S. Stańczak are with the Department of Network-Information Theory, TU Berlin.



(a) Intersection polytope $\mathcal{P}_{\boldsymbol{t}}$



(b) Centroid $\hat{\boldsymbol{x}}$

Fig. 1. Intersection polytope and centroid for $\boldsymbol{t} = 0.5$ and $\boldsymbol{A} = \boldsymbol{V}_s^T = [0, 0, 1]$.

used to refer to scalar, vector and matrix random variables x, $\boldsymbol{\mathsf{x}}$, $\boldsymbol{\mathsf{X}}$. Throughout the paper, all random vectors are functions defined over a suitable probability measure space $(\Omega, \mathcal{A}, p)$ where Ω is a sample space, $\mathcal{A}$ is a σ-algebra, and $p : \mathcal{A} \to [0, 1]$ a probability measure on $\mathcal{A}$.[1]. We use $p(\mathcal{X}) = p(\boldsymbol{x} \in \mathcal{X})$ where $\mathcal{X} \in \mathcal{A}$ and $\mathcal{U}(\mathcal{X})$ to denote the uniform distribution over the set $\mathcal{X} \in \mathcal{A}$; finally, functions of random vectors are assumed to be measurable functions, and we further assume that random vectors $\boldsymbol{\mathsf{x}}$ are absolute continuous with probability density function (pdf) $p_{\boldsymbol{\mathsf{x}}}(\boldsymbol{x})$ and expectation $\mathbb{E}[\boldsymbol{\mathsf{x}}] < \infty$. We use $\boldsymbol{0}$, $\boldsymbol{1}$, $\boldsymbol{e}_n$ and $\boldsymbol{I}$ to denote the vectors of all zeros, all ones, n-th Euclidean basis vector, and the identity matrix, where the size will be clear from the context. The i-th column, resp. j-th row, resp. submatrix of i_1-th to i_2-th column and j_1-th to j_2-th row, of a matrix is designated by $\boldsymbol{A}_{:,i}$, $\boldsymbol{A}_{j,:}$ and $\boldsymbol{A}_{i_1:i_2,j_1:j_2}$. $\mathrm{tr}\{\cdot\}$, $\odot$, $\oslash$ and $\mathbb{1}_{\mathcal{X}} : \boldsymbol{x} \to \{0, 1\}$ denote the trace of a matrix, entrywise product, entrywise division and the indicator function defined as $\mathbb{1}_{\mathcal{X}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{X}$ and 0 otherwise. $\mathbb{1}_{+}(x)$ is the Heaviside function and $(x)_+ := \max(0, x)$ is the rectified linear function.

II. Optimal Reconstruction by Centroid Computation of Intersection Polytopes

We assume that the sought vector $\boldsymbol{x} \in \mathbb{R}_+^N$ is a realization of a non-negative random vector $\boldsymbol{\mathsf{x}}$ drawn from a joint distribution with a pdf denoted by $p_{\boldsymbol{\mathsf{x}}}(\boldsymbol{x})$. Throughout the paper, we have the following assumption:

[1]In particular, if the random vectors are in $\mathbb{R}_+^N$, then $\Omega = \mathbb{R}_+^N$ and $\Delta \in \mathcal{A}$ with $\mathcal{A}$ being the power set of $\mathbb{R}_+^N$.

Assumption 1. *The standard simplex Δ is the support of $p_{\boldsymbol{\mathsf{x}}}(\boldsymbol{x})$ so that we have $p_{\boldsymbol{\mathsf{x}}}(\Delta) = p_{\boldsymbol{\mathsf{x}}}(\boldsymbol{x} \in \Delta) = 1$.*

This assumption imposes some compressibility on $\boldsymbol{x}$, which is often referred to as soft sparsity. In practice, input signals $\boldsymbol{x} \in \Delta$ appear freqeuently in applications that involve discrete probability vectors including softmax classification in machine learning [1] as illustrated in Fig. 7 and 8, multiple hypothesis testing in statistics [1] or maximum likelihood decoding in communications [10]. The problem of compressing and recovering such vectors applies in particular to distributed decision making problems, where the decisions are distributed among different decision makers and need to be fused to improve an overall decision metric. We note that the set of discrete probability vectors of length $N + 1$ generates the dihedral face of a simplex in dimension $N + 1$ that can be embedded naturally into the (solid) simplex of dimension N by removing one arbitrary component. We refer the reader to Sec. VI for a more detailed description as well as numerical results of the proposed recovery method on soft-classification vectors generated by two standard datasets.

As a result of Assumption 1, given a vector $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, the set of feasible solutions is restricted to a polytope

$$\mathcal{P}_{\boldsymbol{y}} := \Delta \cap \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}\} = \Delta \cap (\boldsymbol{x}_0 + \ker(\boldsymbol{A})),^{2} \tag{2}$$

where $\boldsymbol{x}_0$ is an arbitrary solution to $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ (e.g. the Moore-Penrose pseudo inverse $\boldsymbol{x}_0 = \boldsymbol{A}^\dagger \boldsymbol{y}$) and $\ker(\boldsymbol{A})$ denotes the $N - M$-dimensional kernel of $\boldsymbol{A}$. In other words, upon observing $\boldsymbol{y} \in \mathbb{R}^M$, the support of the conditional pdf $p_{\boldsymbol{\mathsf{x}}|\boldsymbol{\mathsf{y}}}(\boldsymbol{x}|\boldsymbol{y})$ is equal to $\mathcal{P}_{\boldsymbol{y}}$. To simplify the subsequent derivations, let $\boldsymbol{V}_0$ and $\boldsymbol{V}_s$ constitute a basis of $\ker(\boldsymbol{A})$ and $\ker^\perp(\boldsymbol{A})$ as obtained, for instance, by the singular-value decomposition (SVD)

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix}\boldsymbol{U}_s\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_s & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_s^T \\ \boldsymbol{V}_0^T\end{bmatrix}. \tag{3}$$

Here, $\boldsymbol{U}_s \in \mathbb{R}^{M\times M}$, $\boldsymbol{\Sigma}_s \in \mathbb{R}^{M\times M}$, $\boldsymbol{V}_s \in \mathbb{R}^{N\times M}$ and $\boldsymbol{V}_0 \in \mathbb{R}^{N\times N-M}$. Then, we apply (3) to (2) and multiply $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ from left by $\Sigma_s^{-1}\boldsymbol{U}_s^T$ (note that $\boldsymbol{U}_s^T\boldsymbol{U}_s = \boldsymbol{I}_M$) to obtain an equivalent description for the same polytope given by

$$\mathcal{P}_{\boldsymbol{t}} := \Delta \cap \{\boldsymbol{x} : \boldsymbol{V}_s^T\boldsymbol{x} = \boldsymbol{t}\}. \tag{4}$$

This description uses the equivalent measurement vector

$$\boldsymbol{t} := \boldsymbol{\Sigma}_s^{-1}\boldsymbol{U}_s^T\boldsymbol{y} = \boldsymbol{V}_s^T\boldsymbol{x} \tag{5}$$

and defines the intersection polytope in terms of $\ker^\perp(\boldsymbol{A})$ via the orthogonal basis $\boldsymbol{V}_s$. We point out that as $\boldsymbol{A}$ is assumed to be real, so are also $\boldsymbol{U}$ and $\boldsymbol{V}$. Fig. 1(a) depicts an example of the polytope $\mathcal{P}_{\boldsymbol{t}}$ for $M = 1$ and $N = 3$.

Lemma 1 (Optimality of centroid estimator under uniform distribution)**.** *Assume $\boldsymbol{\mathsf{x}} \sim \mathcal{U}(\Delta)$ and suppose that a compressed realization $\boldsymbol{t} = \boldsymbol{V}_s^T\boldsymbol{x}$ (5) has been observed. Let $\rho_{\mathcal{P}}$ be the Lebesgue measure on the $(N - M)$-dimensional affine subspace that contains $\mathcal{P}_{\boldsymbol{t}}$ (see [11, Sec. 2.4]) and assume*

[2]For two sets $\mathcal{X}$ and $\mathcal{Y}$ we denote their Minkowski Sum by $\mathcal{X} + \mathcal{Y} := \{\boldsymbol{x} + \boldsymbol{y} : \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}\}$. Here, one set is the singleton $\boldsymbol{x}_0$.

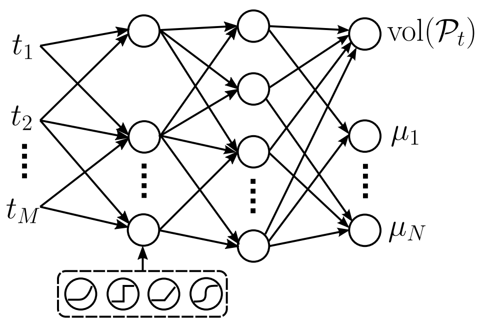


Fig. 2. Neural network to solve problem (P1) and (P2).

that $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) := \int_{\mathcal{P}_{\boldsymbol{t}}} 1 \ d\varrho_{\mathcal{P}} > 0$. *Then, the conditional mean estimator* $\hat{\boldsymbol{x}}$ *of* $\boldsymbol{x}$ *given* $\boldsymbol{t}$ *has the MMSE property*

$$\mathbb{E}\left[\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2\right] = \inf_f \mathbb{E}\left[\|\boldsymbol{x}-f(\boldsymbol{t})\|_2^2\right] \tag{6}$$

and is obtained by

$$\hat{x}_n = \mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})^{-1} \int_{\mathcal{P}_{\boldsymbol{t}}} x_n \ d\varrho_{\mathcal{P}}, \ n \in \{1,\ldots,N\}, \tag{7}$$

i.e., the centroid of the intersection polytope $\mathcal{P}_{\boldsymbol{t}}$*, cf.* (4)*.*

Proof. The proof is a standard result in estimation theory [12]. □

The remaining part of the paper is devoted to the following problem given a fixed measurement matrix $\boldsymbol{A}$:

Problem 1. *Design a neural network composed of only elementary arithmetic operations and activation functions such that if the input to the network is* $\boldsymbol{t}$*, then its output is*
(P1) the volume $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$*, and*
(P2) the moments $\boldsymbol{\mu} = \int_{\mathcal{P}_{\boldsymbol{t}}} \boldsymbol{x} \ d\varrho_{\mathcal{P}}$*.*

The sought neural network is depicted in Fig. 2

III. A review of Lasserre's Laplace techniques

To make this paper as self-contained as possible and highlight the theoretical contribution of the original works [13], [14], we start this section with a review of Laplace techniques that will be used in Sec. IV and V for computing the volume and moments of full- and lower-dimensional polytopes. To this end, we introduce a set of definitions restated from [15], [16].

Definition 1 (Def. 1.4.3 [16])**.** *The (one-sided) Laplace transform (LT) of a function* $f : \mathbb{R}_+ \to \mathbb{R}$ *is the function* $F : \mathbb{C} \to \mathbb{R}$ *defined by*

$$F(\lambda) = \mathcal{L}_t(f(t)) := \int_0^\infty f(t)\exp(-\lambda t) \ dt \tag{8}$$

provided that the integral exists.

Here and hereafter, we refer to t as transform variable and λ as Laplace variable and conveniently designate LT transform pairs by $f(t)\circ\!\!-\!\!\bullet F(\lambda)$.

Remark 1. *We assume input functions can be written in terms of the Heaviside function as* $\mathbb{1}_+(t)f(t)$ *where* $\mathbb{1}_+$ *will usually be omitted. Thereby, expression of* $f(t)$ *apply for* $t \geq 0$ *and* $f(t) = 0$ *for* $t < 0$*.*

Lemma 2 (Def. 1.4.4, Th. 1.4.8 [16])**.** *Let* f *be locally integrable and assume there exists a number* $\gamma \in \mathbb{R}$ *such that*

$$\int_0^\infty |f(t)|\exp(-\gamma t) \ dt < \infty. \tag{9}$$

Then, the Laplace integral (8) *is absolutely and uniformly convergent on* $\{\lambda \in \mathbb{C} : \mathrm{Re}(\lambda) \geq \gamma\}$ *and*

$$\gamma_{\mathrm{ac}} := \inf\left\{\gamma \in \mathbb{R} : \int_0^\infty |f(t)|\exp(-\gamma t) \ dt < \infty\right\} \tag{10}$$

is called the abscissa of absolute convergence.

Lemma 3 (Th. 1.4.12)**.** *[16] Let* f *be as in Lemma 2 and (possibly piecewise) continuous. Then in points* t *of continuity the inverse Laplace transform (ILT) is given by*

$$f(t) = \mathcal{L}_\lambda^{-1}(F(\lambda)) := \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} F(\lambda)\exp(\lambda t) \ d\lambda, \ c > \gamma_{\mathrm{ac}} \tag{11}$$

and in points of discontinuity we obtain the arithmetic mean

$$\frac{f(t^+)+f(t^-)}{2} = \mathcal{L}_\lambda^{-1}(F(\lambda)). \tag{12}$$

As we assume that $f(t) = 0$ for $t < 0$ given expressions for $f(t)\circ\!\!-\!\!\bullet F(\lambda)$ are to be understood as $\mathbb{1}_+(t)f(t)$.

Definition 2. *The (one-sided)* M*-dimensional LT (*M*D-LT) of a function* $f : \mathbb{R}_+^M \to \mathbb{R}$ *is the function* $F : \mathbb{C}^M \to \mathbb{C}$ *defined by the concatenation of LTs*

$$F(\boldsymbol{\lambda}) = \left(\circ_{m=1}^M \mathcal{L}_{t_m}\right)(f(\boldsymbol{t})) \tag{13}$$

provided that all integrals exist. The order of integration in (13) *is arbitrary provided that the individual LTs converge absolutely and uniformly [15].*

Definition 3. *The* M*-dimensional ILT (*M*D-ILT) of a function* $F(\boldsymbol{\lambda}) : \mathbb{C}^M \to \mathbb{C}$ *is defined by the concatenation of ILTs*

$$f(\boldsymbol{t}) = \left(\circ_{m=1}^M \mathcal{L}_{\lambda_m}^{-1}\right)(F(\boldsymbol{\lambda})) \tag{14}$$

provided that all integrals exist. The order of integration is again arbitrary provided the individual ILTs converge and given expressions for $f(\boldsymbol{t})\circ\!\!-\!\!\bullet F(\boldsymbol{\lambda})$ *are to be understood as* $(\prod_{m=1}^M \mathbb{1}_+(t_m))f(\boldsymbol{t})$*.*

Remark 2. *To make this article accessible for a broader audience we omit a more detailed exposition of operational properties as well as existence conditions of the (multidimensional) Laplace operators and refer the interested reader to [15], [16]. We note that the transforms appearing in this article are obtained by combining standard transform pairs summarized in Tab. I.*

Now we are in a position to introduce the general idea of Lasserre for polyhedral volume computation [13], [14] that consists in exploiting the identity

$$\begin{aligned} f(\boldsymbol{t}) &= \left(\circ_m \mathcal{L}_{\lambda_m}^{-1}\right)\left(\circ_m \mathcal{L}_{t_m}\right)(f(\boldsymbol{t})) \\ &= \big(\underbrace{\mathcal{L}_{\lambda_M}^{-1} \circ \ldots \circ \mathcal{L}_{\lambda_1}^{-1}}_{M-\text{times}} \circ \underbrace{\mathcal{L}_{t_M} \circ \ldots \circ \mathcal{L}_{t_1}}_{M-\text{times}}\big)(f(\boldsymbol{t})) \end{aligned} \tag{15}$$

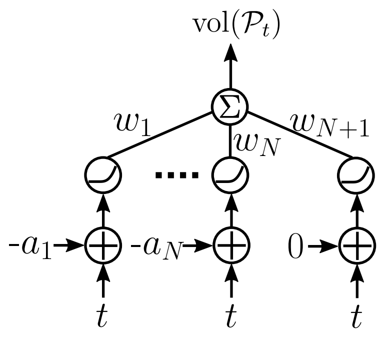


Fig. 3. Volume computation network with *rectified polynomial* layers for input $t \in \mathbb{R}$.

for functions $f(\boldsymbol{t})$ admitting (multidimensional) Laplace transforms according to Lemma 2, 3 and Def. 2, 3.

Interestingly, this approach allows for evaluating complicated functions as $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$ without resorting to costly multidimensional numerical integration. For the particular case of (P1) and a single measurement (see illustration in Fig. 1) with

$$f(t) = \mathrm{vol}(\Delta \cap \{\boldsymbol{x} : \boldsymbol{a}^T\boldsymbol{x} = t\}) \tag{16}$$

the corresponding result of Lasserre [14] is given below.

Lemma 4 (Volume of a simplex slice [14])**.** *Let $\mathcal{S} = \{\boldsymbol{s}_1, \ldots \boldsymbol{s}_{N+1}\} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N, \boldsymbol{0}\}$ denote the vertices of the standard simplex and $\boldsymbol{a} \in \mathbb{S}^{N-1}$ be such that $\langle \boldsymbol{a}, \boldsymbol{s}_n \rangle \neq \langle \boldsymbol{a}, \boldsymbol{s}_{n'} \rangle$ for any pair of distinct vertices $\boldsymbol{s}_n$, $\boldsymbol{s}_{n'}$. Then, the volume of the simplex slice at any point $t \in \mathbb{R}$ is given by*

$$\mathrm{vol}\,(\mathcal{P}_t) = \frac{1}{(N-1)!}\sum_{n=1}^{N+1} \frac{(t - \langle \boldsymbol{a}, \boldsymbol{s}_n \rangle)_+^{N-1}}{\prod_{n' \neq n}(\langle \boldsymbol{a}, \boldsymbol{s}_{n'} \rangle - \langle \boldsymbol{a}, \boldsymbol{s}_n \rangle)}. \tag{17}$$

Proof. The lemma is a restatement of Thm. 2.2 in [14]. □

On closer inspection of (17), we find that $\mathrm{vol}\,(\mathcal{P}_t)$ can be efficiently implemented given some fixed $\boldsymbol{a}$ and an input $t \in \mathbb{R}$ by a 1-layer (shallow) neural network via a set of *rectified polynomial* activation functions with shift $\{-a_1, \ldots, -a_N, 0\}$ and weights $\boldsymbol{w}_n = \prod_{n' \neq n}(\langle \boldsymbol{a}, \boldsymbol{s}_{n'} \rangle - \langle \boldsymbol{a}, \boldsymbol{s}_n \rangle)^{-1}$ as illustrated in Fig. 3. Even though the *rectified polynomial* activation function $(t - a_n)_+^{N-1}$ is currently not implemented in common Deep Learning libraries, it can be efficiently computed via a concatenation of a conventional ReLU $(t - a_n)_+$ followed by a polynomial activation function $(t - a_n)^{N-2}$ (both supported in e.g. the Caffe or Tensorflow framework [17], [18]).

Moreover, it was shown in [14] that (17) also holds when some $a_n < 0$ with $f(t) > 0$ for some $t < 0$ obstructing the direct application of Laplace transform techniques (see Rem. 1). Informally, the necessary extension was achieved by introducing the translation operator $S_{t^*}(\cdot)$ defined by

$$S_{t^*}(f(t)) := f(t - t^\star) \tag{18}$$

and a translation identity in conjunction with (15) given by

$$f(t) = (S_{t^*}S_{-t^*})\,(f(t)) = \left(S_{t^*}\mathcal{L}_\lambda^{-1}\mathcal{L}_t S_{-t^*}\right)(f(t)). \tag{19}$$

Here, the shift $t^\star \in \mathbb{R}$ has to be chosen such that the identity

$$S_{-t^*}(f(t)) = \left(\mathcal{L}_\lambda^{-1}\mathcal{L}_t S_{-t^*}\right)(f(t)) \tag{20}$$

#	$f(t)$	$F(\lambda)$
(LT1)	$c_1f_1(t) + c_2f_2(t)$	$c_1F_1(\lambda) + c_2F_2(\lambda)$
(LT2)	$\frac{t^{n-1}}{(n-1)!}$	$\frac{1}{\lambda^n}$
(LT3)	$\exp(at)$	$\frac{1}{\lambda - a}$
(LT4)	$t\exp(at)$	$\frac{1}{(\lambda - a)^2}$
(LT5)	$\frac{\exp(at) - \exp(bt)}{a - b}$ $(a \neq b)$	$\frac{1}{(\lambda - a)(\lambda - b)}$
(LT6)	$\mathbb{1}_+(t-a)f(t-a)$ $(a \geq 0)$	$\exp(-a\lambda)F(\lambda)$
(LT7)	$f'(t)$	$\lambda F(\lambda) - f(0^+)$

TABLE I
TABLE OF LAPLACE TRANSFORMS

is admissible and the LT acts on a function f vanishing for $t < 0$.

Remark 3. *To avoid the obfuscation connected with a multidimensional extension of the shifted LT identity* (20) *and involved analysis of admissibility conditions we consider in the following only a special case where $\boldsymbol{A}$ admits a non-negative orthogonal basis $\boldsymbol{V}_s \in \mathbb{R}_+^{N \times M}$ for $\ker^\perp(\boldsymbol{A})$. We show by numerical simulations that similar to Lemma 4 the neural networks to be introduced in the following indeed apply for every orthogonal basis $\boldsymbol{V}_s$.*

IV. Volume computation network

A. Theoretical Foundation

The goal of this section is to solve (P1), i.e., to obtain a neural network computing $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$ given some $\boldsymbol{t} \in \mathbb{R}^M$. To this end, we will first consider the special case $\boldsymbol{V}_s \in \mathbb{R}_+^{N \times M}$, $\boldsymbol{V}_s^T\boldsymbol{V}_s = \boldsymbol{I}$ (see Remark 3). Using this assumption, we may easily verify that $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) = 0$ whenever some $t_m < 0$ and Laplace techniques such as the identity (15) are admissible. To compute $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$, we start with a Lemma on exponential integrals over the simplex that will be used later on.

Lemma 5. *Let $\boldsymbol{l}$ be a linear form such that $\langle \boldsymbol{l}, \boldsymbol{s}_n \rangle \neq \langle \boldsymbol{l}, \boldsymbol{s}_{n'} \rangle$ for any pair of distinct vertices $\boldsymbol{s}_n$, $\boldsymbol{s}_{n'}$ of the simplex Δ. Then we have*

$$\int_\Delta \exp(-\langle \boldsymbol{l}, \boldsymbol{x} \rangle)\; d\boldsymbol{x} = \sum_{n=1}^{N+1} \frac{\exp(-\langle \boldsymbol{l}, \boldsymbol{s}_n \rangle)}{\prod_{n' \neq n}\langle \boldsymbol{l}, \boldsymbol{s}_{n'} - \boldsymbol{s}_n \rangle}. \tag{21}$$

Proof. The lemma follows from [19, Cor. 12] by changing the sign of the linear form and noting that the (full-dimensional) simplex has volume $(N!)^{-1}$. □

We use this result to compute the inner MD-LT in (15). To this end, let $\mathcal{T}_{\boldsymbol{t}} := \Delta \cap \{\boldsymbol{x} : \boldsymbol{V}_s^T\boldsymbol{x} \leq \boldsymbol{t}\}$ denote the intersection of the simplex and M halfspaces and consider

$$F(\boldsymbol{\lambda}) = F(\lambda_1, \ldots, \lambda_M) = \big(\underbrace{\mathcal{L}_{t_M} \circ \ldots \circ \mathcal{L}_{t_1}}_{M-\text{times}}\big)(\mathrm{vol}(\mathcal{T}_{\boldsymbol{t}}))\,. \tag{22}$$

By the definition of the MD-LT (13) and the fact that $\mathbb{R} \to \mathbb{R}_+ : x_k \to \exp(x_k)$ is non-negative the Fubini-Tonelli theorem [20, Th. 9.11] implies that for $\mathrm{Re}(\boldsymbol{\lambda}) > \mathbf{0}$ we have

$$\begin{aligned} F(\boldsymbol{\lambda}) &= \int_{\mathbb{R}_+^M} \exp(-\langle \boldsymbol{\lambda}, \boldsymbol{t} \rangle) \left(\int_{\Delta \cap \{\boldsymbol{x} : \boldsymbol{V}_s^T \boldsymbol{x} \leq \boldsymbol{t}\}} 1 \, d\boldsymbol{x} \right) d\boldsymbol{t} \\ &= \int_\Delta \left(\int_{[\boldsymbol{v}_1^T \boldsymbol{x}, \infty) \times \ldots \times [\boldsymbol{v}_M^T \boldsymbol{x}, \infty)} \exp(-\langle \boldsymbol{\lambda}, \boldsymbol{t} \rangle) \, d\boldsymbol{t} \right) d\boldsymbol{x} \\ &= \frac{1}{\prod_{m=1}^M \lambda_m} \int_\Delta \exp(-\langle \boldsymbol{V}_s \boldsymbol{\lambda}, \boldsymbol{x} \rangle) \, d\boldsymbol{x}. \end{aligned} \tag{23}$$

Using the Laplace identity (15) we see that

$$\mathrm{vol}(\mathcal{T}_{\boldsymbol{t}}) = \left(\circ_m \mathcal{L}_{\lambda_m}^{-1} \right) \left(\frac{1}{\prod_m \lambda_m} \int_\Delta \exp(-\langle \boldsymbol{V}_s \boldsymbol{\lambda}, \boldsymbol{x} \rangle) \, d\boldsymbol{x} \right), \tag{24}$$

whenever the MD-ILT (14) on the RHS exists. In this case, the RHS is a function of the transform variable $\boldsymbol{t}$ and the inner simplex integral may be evaluated via Lemma 5 whenever the corresponding condition holds. To obtain $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$, we use (23) to establish the following proposition.

Proposition 1. *Let $\boldsymbol{V}_s$ be a non-negative orthogonal basis ($\boldsymbol{V}_s \in \mathbb{R}_+^{N \times M}$, $\boldsymbol{V}_s^T \boldsymbol{V}_s = \boldsymbol{I}_M$) and $\mathcal{P}_{\boldsymbol{t}} := \Delta \cap \{\boldsymbol{x} : \boldsymbol{V}_s \boldsymbol{x} = \boldsymbol{t}\}$. Then, we have*

$$\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) = \left(\circ_{m=1}^M \mathcal{L}_{\lambda_m}^{-1} \right) \left(\int_\Delta \exp(-\langle \boldsymbol{V}_s \boldsymbol{\lambda}, \boldsymbol{x} \rangle) \, d\boldsymbol{x} \right) \tag{25}$$

provided that the integrals on the RHS exist.

Proof. The proof is deferred to Appendix A. □

Let us now turn to the numerical evaluation of $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$ via the MD-ILT (14). To this end, we first evaluate the inner integral over the simplex using Lemma 5 to obtain

$$\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) = \left(\circ_{m=1}^M \mathcal{L}_{\lambda_m}^{-1} \right) \left(\sum_{n=1}^N \frac{\exp(-\langle \boldsymbol{V}_s \boldsymbol{\lambda}, \boldsymbol{s}_n \rangle)}{\prod_{n' \neq n} \langle \boldsymbol{V}_s \boldsymbol{\lambda}, \boldsymbol{s}_{n'} - \boldsymbol{s}_n \rangle} \right). \tag{26}$$

Given that the argument of the MD-ILT in (26) is a sum of *exponential-over-polynomial* (*exp-over-poly*) functions we continue with a Lemma on corresponding one-dimensional transform pairs.

Lemma 6 (ILT of an *exp-over-poly* function)**.**

1) *Let $M = 1$, $a \geq 0$, $\boldsymbol{b} \in \mathbb{R}_{\neq 0}^N$. Then we have the transform pair ($F(\lambda) \bullet\!\!-\!\!\circ f(t)$)*

$$\frac{\exp(-a\lambda)}{\prod_{n=1}^N (b_n \lambda)} \overset{a \geq 0}{\bullet\!\!-\!\!\circ} \frac{(t-a)_+^{N-1}}{(N-1)! \prod_{n=1}^N b_n}. \tag{27}$$

2) *Let $M \geq 2$, $\boldsymbol{a} \in \mathbb{R}^M$ and $\boldsymbol{B}_{:,1} \in \mathbb{R}_{\neq 0}^N$, $\boldsymbol{B}_{:,2:M} \in \mathbb{R}^{N \times M-1}$ with pairwise linearly independent rows. Then we have the transform pair $F(\lambda_1, \ldots, \lambda_M) \bullet\!\!-\!\!\circ f(t_1, \lambda_2, \ldots, \lambda_M)$*

$$\frac{\exp(-\langle \boldsymbol{a}, \boldsymbol{\lambda} \rangle)}{\prod_{n=1}^N [\boldsymbol{B}\boldsymbol{\lambda}]_n} \overset{a_1 \geq 0}{\bullet\!\!-\!\!\circ} \frac{\mathbb{1}_+(t_1 - a_1)}{\prod_n B_{n,1}} \sum_{n=1}^N \frac{\exp(-\langle \boldsymbol{a}^{(n)}, \boldsymbol{\lambda}_{2:M} \rangle)}{\prod_{n'=1}^{N-1} [\boldsymbol{B}^{(n)} \boldsymbol{\lambda}_{2:M}]_{n'}}. \tag{28}$$

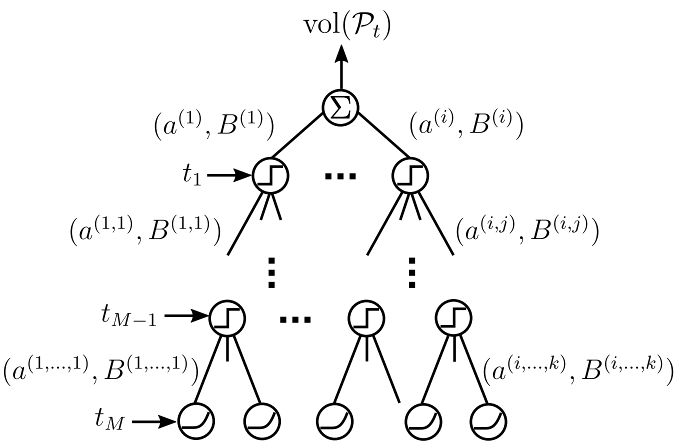


Fig. 4. Volume network with *rectified polynomial* and *threshold* layers for input $\boldsymbol{t} \in \mathbb{R}^M$.

Setting $\boldsymbol{C} := \boldsymbol{B} \oslash (\boldsymbol{B}_{:,1} \mathbf{1}^T)$ we obtain $\boldsymbol{a}^{(n)} \in \mathbb{R}^{M-1}$ and $\boldsymbol{B}^{(n)} \in \mathbb{R}^{N-1 \times M-1}$ ($n \in \{1, \ldots, N\}$) by

$$\boldsymbol{a}^{(n)} = \boldsymbol{a}_{2:M} + (t_1 - a_1) \boldsymbol{C}_{n,2:M}, \tag{29}$$

$$\boldsymbol{B}^{(n)} = \boldsymbol{C}_{1:N \setminus n, 2:M} - \mathbf{1} \boldsymbol{C}_{n,2:M}. \tag{30}$$

Remark 4. *We highlight that in case $B_{k,1} = 0$ for some k we can treat the corresponding factor $\boldsymbol{B}_{k,:}\boldsymbol{\lambda} = \boldsymbol{B}_{k,2:M} \boldsymbol{\lambda}_{2:M}$ as a constant w.r.t. the ILT $\mathcal{L}_{\lambda_1}^{-1}(\cdot)$. Accordingly, we can apply* (28) *using the truncated matrix $\tilde{\boldsymbol{B}} = \boldsymbol{B}_{1:N \setminus k,:}$, where $\tilde{\boldsymbol{B}}_{:,1} \in \mathbb{R}_{\neq 0}^{N-1}$, and obtain the truncated output $\tilde{\boldsymbol{B}}^{(n)} \in \mathbb{R}^{N-1 \times M-1}$. For the subsequent ILT we have to include the corresponding factor $\boldsymbol{B}_{k,2:M} \boldsymbol{\lambda}$ again by applying the matrix concatenation*

$$\boldsymbol{B}^{(n)} = \begin{bmatrix} \tilde{\boldsymbol{B}}^{(n)} \\ \boldsymbol{B}_{k,2:M} \end{bmatrix}. \tag{31}$$

B. Structure of the network

Now we are in a position to present a computation network for the MD-ILT (25) by an algorithm that can be described by a computational tree (see [13]). Due to the particular form of RHS in (21) the root of this tree is the MD-ILT of $N+1$ *exp-over-poly*-functions

$$(\boldsymbol{a}^{(i)}, \boldsymbol{B}^{(i)}) \to \frac{\exp(-\langle \boldsymbol{a}^{(i)}, \boldsymbol{\lambda} \rangle)}{\prod_{n'} [\boldsymbol{B}^{(i)} \boldsymbol{\lambda}]_{n'}}. \tag{32}$$

1) In layer $m = 1$, each node computes the ILT $F(\lambda_1, \ldots, \lambda_M) \bullet\!\!-\!\!\circ f(t_1, \lambda_2, \ldots, \lambda_M)$ of an *exp-over-poly*-function with parameters $(\boldsymbol{a}^{(i)}, \boldsymbol{B}^{(i)})$ inherited from the root node. If the branch is active, i.e. $\mathbb{1}_+(t_1 - a_1^{(i)})(\prod_n B_{n,1}^{(i)})^{-1} \neq 0$, it applies transform (28) and passes the corresponding N exp-over-poly functions $(\boldsymbol{a}^{(i,j)}, \boldsymbol{B}^{(i,j)})$, $(i,j) \in \{1, \ldots, N+1\} \times \{1, \ldots, N\}$ to its children nodes.
2) In layer $m = 2$, each node computes the ILT $f(t_1, \lambda_2, \ldots, \lambda_M) \bullet\!\!-\!\!\circ f(t_1, t_2, \lambda_3, \ldots, \lambda_M)$ of an *exp-over-poly*-function inherited from its parent node. If the branch is active, i.e. $\mathbb{1}_+(t_2 - a_1^{(i,j)})(\prod_n B_{n,1}^{(i,j)})^{-1} \neq 0$,

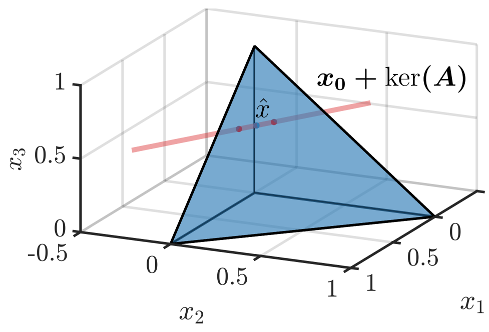


Fig. 5. Intersection polytope (line) and centroid for $\boldsymbol{t} = [0.5; 0.933]$ and $\boldsymbol{A} = \boldsymbol{V}_s^T = [0, 0, 1; 0.5, 0.866, 0]$.

it applies transform (28) and passes the corresponding $N-2$ exp-over-poly functions $(\boldsymbol{a}^{(i,j,k)}, \boldsymbol{B}^{(i,j,k)})$, $(i,j,k) \in \{1,\dots,N+1\} \times \{1,\dots,N\} \times \{1,\dots,N-1\}$ to its children nodes.

3) ...
4) In the last layer $m = M$, each node computes the ILT $f(t_1,\dots,t_{M-1},\lambda_M) \bullet\!\!-\!\!\circ f(t_1,\dots,t_M)$ of an *exp-over-poly*-function inherited from its parent node. If the branch is active, i.e. $\mathbb{1}_+(t_M - a_1^{(i,j,\dots)})((N-1)!(\prod_n B_{n,1}^{(i,j,\dots)}))^{-1} \neq 0$, it applies transform (27) and obtains a numerical value.
5) The final result is obtained by summing and weighting all values of the last layer.

The described computational tree is equivalent to a deep neural network with M layers, weights $\prod_n B_{n,1}^{(i,j,\dots)}$ and *threshold* as well as *rectified polynomial* activation functions (see Fig. 4).

C. Volume computation example

In the following we will present a numerical example for the computation of $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$ via the described computation network and repeated application of Lemma 6.

Example 1 (Volume computation)**.** *Assume $M = 2$, $N = 3$ and let $\mathcal{P}_{\boldsymbol{t}}$ be defined by the measurements (see corresponding illustration in Fig. 5)*

$$\boldsymbol{t} = \boldsymbol{V}_s^T \boldsymbol{x}$$

$$\begin{bmatrix} 0.5 \\ 0.0933 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0.866 & 0 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.05 \\ 0.5 \end{bmatrix}. \tag{33}$$

The exp-over-poly functions at the root node are obtained by (21) *and defined by the parameters*

$$\boldsymbol{a}^{(1)} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}, \ \boldsymbol{a}^{(2)} = \begin{bmatrix} 0 \\ 0.866 \end{bmatrix}, \ \boldsymbol{a}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ \boldsymbol{a}^{(4)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{B}^{(1)} = \begin{bmatrix} 0 & 0.366 \\ 1 & -0.5 \\ 0 & -0.5 \end{bmatrix}, \ \boldsymbol{B}^{(2)} = \begin{bmatrix} 0 & -0.366 \\ 1 & -0.866 \\ 0 & -0.866 \end{bmatrix},$$

$$\boldsymbol{B}^{(3)} = \begin{bmatrix} -1 & 0.5 \\ -1 & 0.866 \\ -1 & 0 \end{bmatrix}, \ \boldsymbol{B}^{(4)} = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.866 \\ 1 & 0 \end{bmatrix}$$

The next step is to compute the ILT with respect to λ_1, i.e.,

$$\mathcal{L}_{\lambda_1}^{-1}\left(\sum_{i=1}^{4} \frac{\exp(-\langle \boldsymbol{a}^{(i)}, \boldsymbol{\lambda} \rangle)}{\prod_{n=1}^{N} [\boldsymbol{B}^{(i)} \boldsymbol{\lambda}]_n}\right). \tag{34}$$

Applying (28) *and using truncation and concatenation when required (see Rem. 4) yields*

$$\boldsymbol{a}^{(1,1)} = 0.25,\ \boldsymbol{a}^{(2,1)} = 0.433,\ \boldsymbol{a}^{(3,1)} = 0.25,$$
$$\boldsymbol{a}^{(3,2)} = 0.433,\ \boldsymbol{a}^{(3,3)} = 0,\ \boldsymbol{a}^{(4,1)} = 0,$$
$$\boldsymbol{B}^{(1,1)} = \begin{bmatrix} 0.366 \\ -0.5 \end{bmatrix}, \ \boldsymbol{B}^{(2,1)} = \begin{bmatrix} -0.366 \\ -0.866 \end{bmatrix}, \ \boldsymbol{B}^{(3,1)} = \begin{bmatrix} -0.366 \\ 0.5 \end{bmatrix},$$
$$\boldsymbol{B}^{(3,2)} = \begin{bmatrix} 0.366 \\ 0.866 \end{bmatrix}, \ \boldsymbol{B}^{(3,3)} = \begin{bmatrix} -0.5 \\ -0.866 \end{bmatrix}, \ \boldsymbol{B}^{(4,1)} = \begin{bmatrix} 0.5 \\ 0.866 \end{bmatrix}.$$

Accordingly, the outputs of the last layer in the computational tree are given by

$$f^{(i,j)} = \frac{(t_2 - a_1^{(i,j)})_+}{\prod_n B_{n,1}^{(i,j)}} \cdot \frac{\mathbb{1}_+(t_1 - a_1^{(i)})}{\prod_{n'} B_{n',1}^{(i)}}$$
$$f^{(1,1)} = 0,\ f^{(2,1)} = 0,\ f^{(3,1)} = 0$$
$$f^{(3,2)} = 0,\ f^{(3,3)} = 0,\ f^{(4,1)} = 0.2155,$$

where index $n' \in \mathrm{supp}(\boldsymbol{B}_{:,1}^{(i)})$ and the final result is given by $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) = 0.2155$.

V. Moment computation network

The final step towards the optimal estimator of Lem. 1 is to compute the moment vector $\boldsymbol{\mu} = \int_{\mathcal{P}_{\boldsymbol{t}}} \boldsymbol{x}\, d\varrho_{\mathcal{P}}$. To this end, we introduce an extension of Lemma 5 as well as a conjecture that was verified numerically but currently lacks a rigorous proof.

Lemma 7. *Let $\boldsymbol{l}$ and Δ be as in Lemma 5. Then we have*

$$\int_\Delta x_k \exp(-\langle \boldsymbol{l}, \boldsymbol{x} \rangle)\, d\boldsymbol{x}$$
$$= \frac{\exp(-\langle \boldsymbol{l}, \boldsymbol{s}_k \rangle)}{\prod_{n \neq k} \langle \boldsymbol{l}, \boldsymbol{s}_n - \boldsymbol{s}_k \rangle} + \sum_{\substack{n=1 \\ n \neq k}}^{N+1} \frac{\exp(-\langle \boldsymbol{l}, \boldsymbol{s}_n \rangle)}{\langle \boldsymbol{l}, \boldsymbol{s}_k - \boldsymbol{s}_n \rangle \prod_{n' \neq n} \langle \boldsymbol{l}, \boldsymbol{s}_{n'} - \boldsymbol{s}_n \rangle}$$
$$- \sum_{\substack{n=1 \\ n \neq k}}^{N+1} \frac{\exp(-\langle \boldsymbol{l}, \boldsymbol{s}_k \rangle)}{\langle \boldsymbol{l}, \boldsymbol{s}_n - \boldsymbol{s}_k \rangle \prod_{n' \neq k} \langle \boldsymbol{l}, \boldsymbol{s}_{n'} - \boldsymbol{s}_k \rangle} \tag{35}$$

Proof. The proof is deferred to Appendix C. □

Conjecture 1. *Let $\mathcal{P}_{\boldsymbol{t}}$ and $\boldsymbol{V}_s$ be as in Prop. 1. Then, the moment $\mu_k := \int_{\mathcal{P}_{\boldsymbol{t}}} x_k\, d\varrho_{\mathcal{P}} = \mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})\hat{x}_k$ (see* (7)*) is given by*

$$\mu_k = \left(\circ_m \mathcal{L}_{\lambda_m}^{-1}\right)\left(\int_\Delta x_k \exp(-\langle \boldsymbol{V}_s \boldsymbol{\lambda}, \boldsymbol{x} \rangle)\, d\boldsymbol{x}\right), \tag{36}$$

provided that the integrals on the RHS exist.

Remark 5. *Conj. 1 originates from Prop. 1 and its numerical correctness was verified by extensive comparison with integration by simplicial decomposition [21]. However, a current gap in a rigorous proof consists in relating the moment of a polyhedron to the moment of an $(N-M)$-dimensional face*

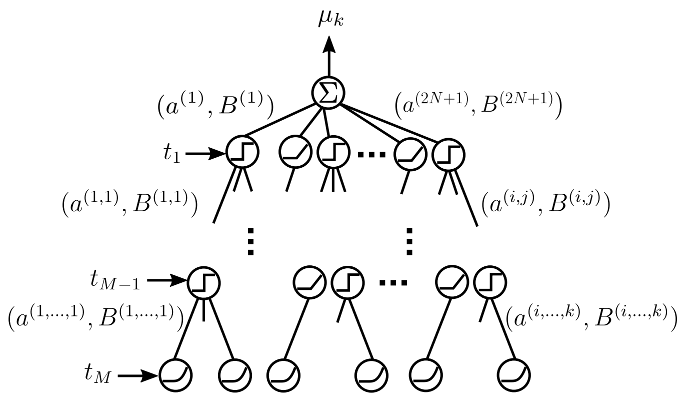


Fig. 6. Moment network for $\boldsymbol{\mu}_k$ with *rectified linear*, *rectified polynomial* and *threshold* layers for $\boldsymbol{t} \in \mathbb{R}^M$.

in the spirit of (44) *[22, Prop. 3.3]. We will assume the conjecture to be valid in what follows in the hope that our network construction will prove useful to other researchers.*

To evaluate the MD-ILT in (36) we inspect the terms in the sum (35) and note that the first term contains only simple poles permitting its Laplace transformation via Lemma 6. In fact, the corresponding term already appears in (21) and accordingly the MD-ILT is already computed as part of the volume computation network. However, this does not apply to the remaining $2N$ terms as they contain quadratic terms $\langle \boldsymbol{l}, \boldsymbol{s}_k - \boldsymbol{s}_n\rangle$ and $\langle \boldsymbol{l}, \boldsymbol{s}_n - \boldsymbol{s}_k\rangle$. Accordingly, for $M \geq 2$ two rows of the denominator matrix $\boldsymbol{B}^{(i)}$, $2 \leq i \leq 2N+1$ will be linearly dependent violating the assumptions of Lemma 6. A modified result applicable for ILTs of *exp-over-poly*-functions with one double pole (resp. two linearly dependent rows of $\boldsymbol{B}$) is provided in the following Lemma.

Lemma 8 (ILT of *exp-over-poly* function with one double pole)**.** *Let* $M \geq 2$*,* $N \geq 3$*,* $\boldsymbol{B}_{:,1} \in \mathbb{R}^N_{\neq \boldsymbol{0}}$ *and assume w.l.o.g. that the first and second row of* $\boldsymbol{B}$ *are equal, i.e.,* $\boldsymbol{B}_{1,:} = \boldsymbol{B}_{2,:}$*. Then, we have the transform pair* $F(\lambda_1, \dots, \lambda_M) \bullet\!\!-\!\!\circ f(t_1, \lambda_2, \dots, \lambda_M)$

$$\frac{\exp(-\langle \boldsymbol{a}, \boldsymbol{\lambda}\rangle)}{\prod_{n=1}^N [\boldsymbol{B}\boldsymbol{\lambda}]_n} \overset{a_1 \geq 0}{\bullet\!\!-\!\!\circ} \frac{(t_1 - a_1)_+}{\prod_{n=1}^N B_{n,1}} \frac{\exp(-\langle \boldsymbol{a}^{(1)}, \boldsymbol{\lambda}_{2:M}\rangle)}{\prod_{n'=1}^{N-1} [\boldsymbol{B}^{(1)}\boldsymbol{\lambda}_{2:M}]_{n'}} + \frac{\mathbb{1}_+(t_1 - a_1)}{\prod_{n=1}^N B_{n,1}} \sum_{n=2}^{N-1} \Bigg(\frac{\exp(-\langle \boldsymbol{a}^{(n)}, \boldsymbol{\lambda}_{2:M}\rangle)}{\prod_{n'=1}^{N-1} [\boldsymbol{B}^{(n)}\boldsymbol{\lambda}_{2:M}]_{n'}} - \frac{\exp(-\langle \boldsymbol{a}^{(1)}, \boldsymbol{\lambda}_{2:M}\rangle)}{\prod_{n'=1}^{N-1} [\boldsymbol{B}^{(n)}\boldsymbol{\lambda}_{2:M}]_{n'}} \Bigg). \tag{37}$$

Setting $\boldsymbol{C} := \boldsymbol{B} \oslash (\boldsymbol{B}_{:,1}\boldsymbol{1}^T)$ *we obtain* $\boldsymbol{a}^{(n)} \in \mathbb{R}^{M-1}$ *and* $\boldsymbol{B}^{(n)} \in \mathbb{R}^{N-1 \times M-1}$ *(*$n \in \{1, \dots, N-1\}$*) by*

$$\boldsymbol{a}^{(n)} = \begin{cases} \boldsymbol{a}_{2:M} + (t_1 - a_1)\boldsymbol{C}_{1,2:M}, & n = 1 \\ \boldsymbol{a}_{2:M} + (t_1 - a_1)\boldsymbol{C}_{n+1,2:M}, & n \geq 2 \end{cases} \tag{38}$$

$$\boldsymbol{B}^{(n)} = \begin{cases} \boldsymbol{C}_{3:N,2:M} - \boldsymbol{1}\boldsymbol{C}_{1,2:M}, & n = 1 \\ \begin{bmatrix} \boldsymbol{C}_{n+1,2:M} - \boldsymbol{C}_{1,2:M} \\ \boldsymbol{C}_{3:N,2:M} - \boldsymbol{1}\boldsymbol{C}_{n+1,2:M} \end{bmatrix}, & n \geq 2. \end{cases} \tag{39}$$

Proof. The proof is deferred to Appendix D. □

Using Lemma 8 we can readily build a computational tree (resp. neural network) to compute the moment μ_k ($1 \leq k \leq N$) comprising $\mathcal{O}(N^M)$ nodes as depicted in Fig. 6. It can be shown that the individual networks for volume and moments perform a set of identical computations, i.e., the networks share particular subnetworks, which results in a subadditive number of nodes for the combined network computing $\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$ and $\boldsymbol{\mu}$. However, the resulting number of nodes in the network still grows as $\mathcal{O}(N^M)$ in the worst case.

Remark 6. *While the worst case growth of* $\mathcal{O}(N^M)$ *applies to a fully connected network, a numerical investigation revealed that large subnetworks were never activated by the corresponding activation functions independently of the particular network input* $\boldsymbol{t} \in \{\boldsymbol{V}_s^T\boldsymbol{x} : \boldsymbol{x} \in \Delta\}$*. In addition, the size of deactivated subnetworks changes with the chosen set of orthogonal basis vectors* $\boldsymbol{V}_s$ *of the subspace* $\ker^{\perp}(\boldsymbol{A})$ *and applying orthogonal transformations resulted in much smaller or much larger active subnetworks. The choice of a favorable (or even optimal) basis of the affine subspace is beyond the scope of this paper but poses an interesting question to be addressed in future works.*

VI. Numerical Example: Compressing soft-classification vectors

To assess the performance of the proposed network we consider the problem of *soft-decision compression* for *distributed decision making*. In the presumed setting, nodes reduce data traffic by transmitting only compressed versions of their local soft-classification vectors and recovery at a fusion center allows for enhanced algorithms to improve an overall decision metric. One possible application in this regard is multi-view image classification. To this end, we investigate the compressibility of softmax-outputs of deep learning classifiers on MNIST handwritten digits and CIFAR-10 images obtained using MatConvNet (www.vlfeat.org/matconvnet/) and assess the fitness of the uniform simplex distribution for practical datasets. The outputs of the trained classifiers are vectors $\tilde{\boldsymbol{x}} \in \mathbb{R}^{10}_+$, where each entry measures the estimated class-membership probability corresponding either to occurrence of digits $\{0, \dots, 9\}$ for MNIST, or occurrence of classes $\{$*plane, car, bird, cat, deer, dog, frog, horse, ship, truck*$\}$ for CIFAR-10. These vectors are preprocessed by removing one uninformative component (e.g. $\tilde{x}_{10} = 1 - \sum_{n=1}^9 \tilde{x}_n$) which ensures that

$$\boldsymbol{x} := \tilde{\boldsymbol{x}}_{1:9} \in \Delta. \tag{40}$$

Examples of a realization $\boldsymbol{x}$ of $\boldsymbol{x} \sim \mathcal{U}(\Delta)$ as well as input images and outputs of standard classifiers are given in Fig. 7, 8 and 9. We adjust the confidence levels to match the measured accuracy via the *temperature*-parameter of the *softmax*-output such that the top-entry is on average 0.9748 for MNIST and 0.7987 for CIFAR-10 (for the default parameter almost all decisions are made with unduly high confidence levels). We measure the empirical mean-square error

$$\mathrm{eMSE} := \frac{1}{N_i} \sum_{i=1}^{N_i} \|\boldsymbol{x}^{(i)} - \hat{\boldsymbol{x}}^{(i)}\|_2^2 \tag{41}$$

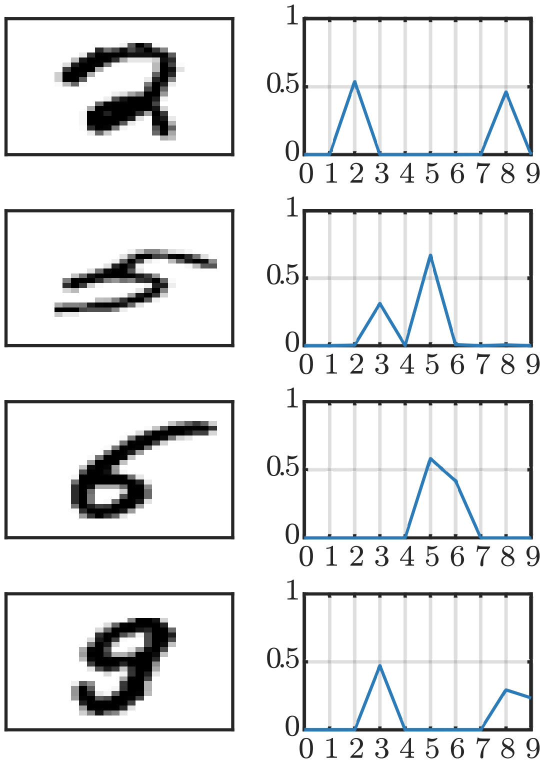


Fig. 7. MNIST images and softmax-classifications for low-confidence examples. True labels are $\{2, 5, 6, 9\}$ and estimated labels are $\{2, 5, 5, 3\}$.

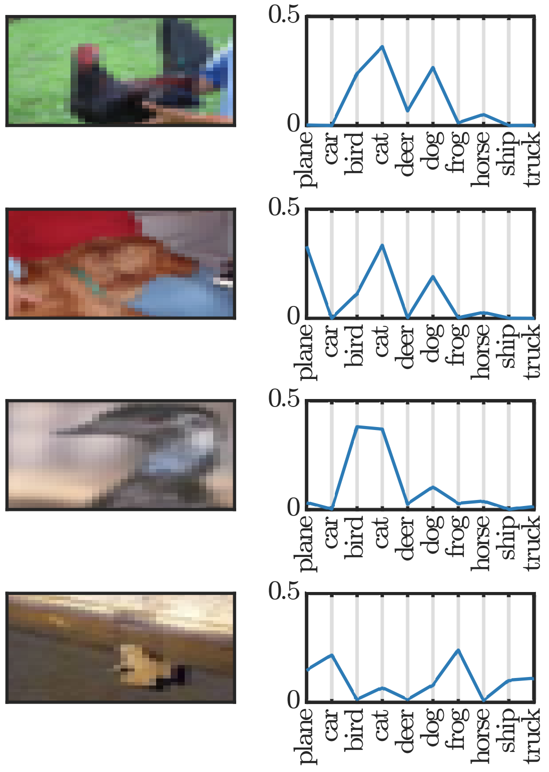


Fig. 8. CIFAR-10 images and softmax-classifications for low-confidence examples. True labels are $\{$*bird, dog, bird, frog*$\}$ and estimated labels are $\{$*cat, cat, bird, frog*$\}$.

over a testing set of cardinality $N_i = 500$ and compare the proposed estimator $\hat{\boldsymbol{x}}$ (7) with exact centroid computation by simplicial decomposition using *Qhull* [21] as well as solutions to the well-established non-negative ℓ_1*-minimization*

$$\hat{\boldsymbol{x}}_{\ell_1} = \underset{\boldsymbol{x} \in \mathbb{R}_+^N: \ \boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}}{\operatorname{argmin}} \quad \|\boldsymbol{x}\|_1 \tag{42}$$

and (simplex constrained) ℓ_2*-minimization*

$$\hat{\boldsymbol{x}}_{\ell_2} = \underset{\boldsymbol{x} \in \Delta: \ \boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}}{\operatorname{argmin}} \quad \|\boldsymbol{x}\|_2^2. \tag{43}$$

For the compression matrix $\boldsymbol{A}$ we use an i.i.d. random Gaussian matrix drawn once and set fixed for all simulations. The compressed soft-classification vector is given by $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^M$ where we vary the number of compressed measurements M. As a reference, we also showcase the results for input signals following the presumed uniform simplex distribution. in Fig. 10. It is interesting to see that for the prescribed uniform distribution the proposed centroid estimator outperforms the conventional ℓ_1- and ℓ_2-based methods by a factor of about 3 and 2 (see Fig. 10). For the MNIST and CIFAR-10 dataset the proposed method is on par with the well-established ℓ_1-minimization method (see Fig. 11 and 12). All simulations were run on a laptop with i7-2.9 GHz processor.[3] Of course, additional performance gains of the proposed network can be expected when fine-tuning the analytically obtained parameters based on given datasets is employed.

[3]In the spirit of reproducible research, the simulation code for the computation of volumes and centroids using Laplace techniques and simplicial decomposition is made available at https://github.com/stli/CentNet.

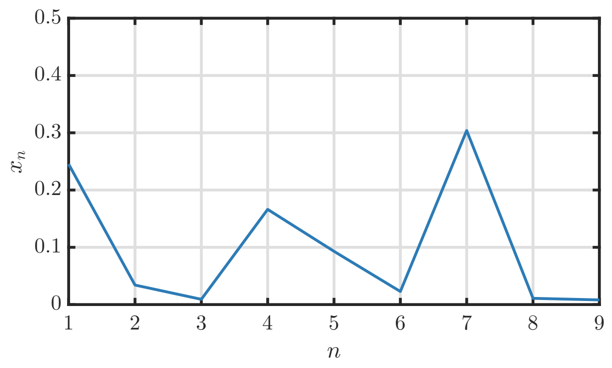


Fig. 9. Realization of $\boldsymbol{x} \sim \mathcal{U}(\Delta)$ for $N = 9$.

As a side note, the volumes of the intersection polytopes can be surprisingly small and in some cases were below numerical precision causing precision problems and numerical underflows. On the other hand, numerical instability is a well-known problem in the design of deep neural networks (see e.g. [1]) and appropriate numerical stabilization techniques, e.g., via logarithmic calculus, are often required. For the datasets at hand, numerical underflows occurred for a small number of MNIST examples, where the input $\boldsymbol{x}$ was close to a vertex $\boldsymbol{s}_i$ of the simplex Δ and the intersection volume becomes extremely small. We note, that these examples were removed for the evaluation of the neural network but kept for all other estimators. As this case is rather easy to solve using conventional ℓ_1-minimization and resulting estimation errors

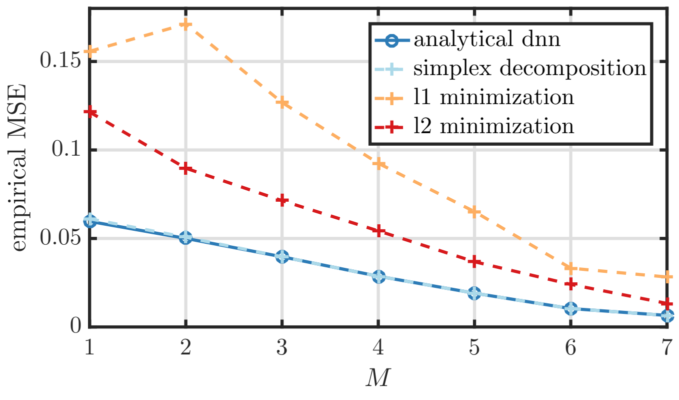


Fig. 10. Empirical MSE for $N = 9$, $\boldsymbol{x} \sim \mathcal{U}(\Delta)$, i.i.d. Gaussian matrix $\boldsymbol{A}$ and varying number of measurements M.

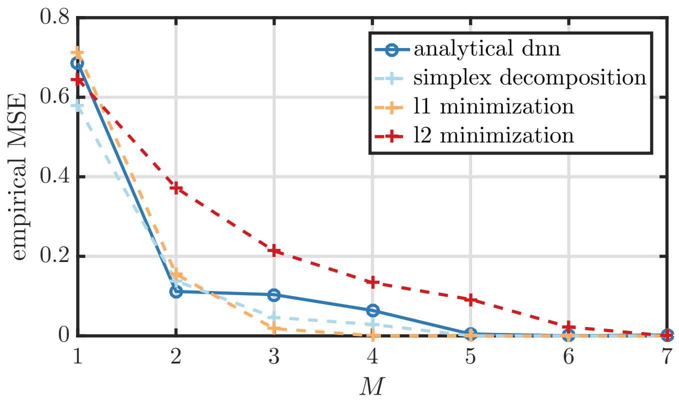


Fig. 11. Empirical MSE for MNIST dataset, i.i.d. Gaussian matrix $\boldsymbol{A}$ and varying number of measurements M.

are typical smaller than average the depicted results do not favour the proposed approach. A detailed numerical analysis and numerically redesigned network is beyond the scope of this paper.

VII. Conclusion

In this paper we proposed a novel theoretically well-founded neural network for sparse recovery. By using multidimensional Laplace techniques and a prescribed input distribution, we obtain a neural network in a fully analytical fashion. Interestingly, the obtained neural network is composed of weights as well as commonly employed threshold functions, rectified linear (ReLU) and rectified polynomial (ReP) activation functions. The obtained network is a first step to understanding the practical effectiveness of classical deep neural network architectures. To scale to higher dimensions, a main problem is to decrease the network width which may be achieved by deactivating maximally large subnetworks via a well-chosen basis of the affine subspace which poses an interesting problem for future works. In addition, it may be beneficial to investigate approximations of the constructed network by a smaller subnetwork which may yield a reasonable approximation of the centroid of interest.

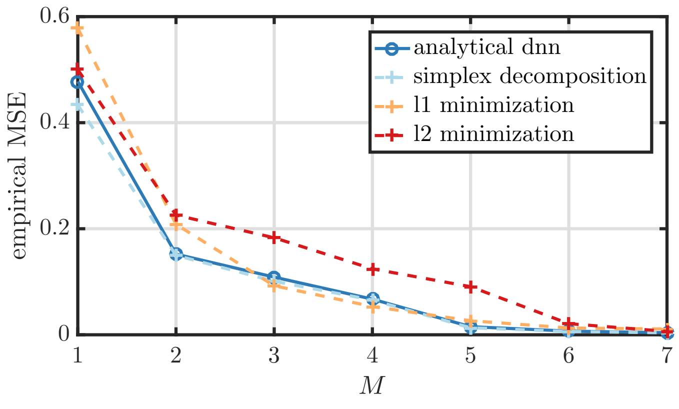


Fig. 12. Empirical MSE for CIFAR-10 dataset, i.i.d. Gaussian matrix $\boldsymbol{A}$ and varying number of measurements M.

Appendix A
Proof of Proposition 1

First note that $\mathcal{P}_{\boldsymbol{t}}$ is an $(N-M)$-dimensional face of $\mathcal{T}_{\boldsymbol{t}}$. Hence, repeated application of [22, Prop. 3.3] shows that

$$\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) = \|\boldsymbol{v}_1\|_2 \cdots \|\boldsymbol{v}_M\|_2 \frac{\partial^M}{\partial t_1 \cdots \partial t_M} \mathrm{vol}(\mathcal{T}_{\boldsymbol{t}}). \tag{44}$$

Next, applying $f(\boldsymbol{t}) = \mathrm{vol}(\mathcal{P}_{\boldsymbol{t}})$ to (15) yields

$$\begin{aligned}\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) &= \left(\circ_m \mathcal{L}^{-1}_{\lambda_m}\right)\left(\circ_m \mathcal{L}_{\lambda_m}\right)\mathrm{vol}(\mathcal{P}_{\boldsymbol{t}}) \\ &= \left(\circ_m \mathcal{L}^{-1}_{\lambda_m}\right)\left(\circ_m \mathcal{L}_{\lambda_m}\right)\frac{\partial^M}{\partial t_1 \cdots \partial t_M}\mathrm{vol}(\mathcal{T}_{\boldsymbol{t}}). \end{aligned}\tag{45}$$

By continuity of $\mathrm{vol}(\mathcal{T}_{\boldsymbol{t}})$ we have $\forall\ m \in \{1,\ldots,M\}$ that

$$\lim_{t_m \to 0}\ \mathrm{vol}(\mathcal{T}_{\boldsymbol{t}}) = 0 \tag{46}$$

and repeated application of the transform pair (LT7) yields the desired result.

Appendix B
Proof of Lemma 6

For Lemma 6.1 with $M = 1$ the stated transform pair is obtained by using the transform pairs (LT6), (LT2) and linearity (LT1).

For Lemma 6.2 with $M \geq 2$ we first prove the transform pair ($\lambda_n^{(0)} \neq \lambda_{n'}^{(0)}$ for $n \neq n'$)

$$\frac{c\exp(-a\lambda)}{\prod_{n=1}^N(\lambda - \lambda_n^{(0)})} \overset{a\geq 0}{\bullet\!\!-\!\!\circ}\ \mathbb{1}_+(t-a)\sum_{n=1}^N \frac{c\exp(\lambda_n^{(0)}(t-a))}{\prod_{n'\neq n}(\lambda_n^{(0)} - \lambda_{n'}^{(0)})}. \tag{47}$$

To this end, we use the partial fraction expansion (see [23, (10) p.77])

$$\begin{aligned}\frac{1}{\prod_{n=1}^N(\lambda-\lambda_n^{(0)})} &= \sum_{n=1}^N \frac{1}{\lambda-\lambda_n^{(0)}} \frac{1}{\frac{d}{d\lambda}\prod_{n'=1}^N(\lambda-\lambda_{n'}^{(0)})|_{\lambda=\lambda_n^{(0)}}} \\ &= \sum_{n=1}^N \frac{1}{\lambda-\lambda_n^{(0)}} \frac{1}{\prod_{n'\neq n}(\lambda_n^{(0)}-\lambda_{n'}^{(0)})} \end{aligned}\tag{48}$$

in conjunction with the transform pair (LT3) to obtain the transform pair

$$\frac{1}{\prod_{n=1}^{N}(\lambda-\lambda_n^{(0)})} \bullet\!\!-\!\!\circ \sum_{n=1}^{N}\frac{\exp(\lambda_n^{(0)}t)}{\prod_{n'\neq n}(\lambda_n^{(0)}-\lambda_{n'}^{(0)})}. \tag{49}$$

Then, (47) follows from (49) by linearity and the transform pair (LT6). Finally, by assumption

$$\lambda_n^{(0)} := -\frac{1}{B_{n,1}}\boldsymbol{B}_{n,2:M}\boldsymbol{\lambda}_{2:M} \tag{50}$$

are pairwise distinct (we can assume $\boldsymbol{\lambda}_{2:M}$ is arbitrary but fixed) and the result (28) follows from (47) by setting $\lambda := \lambda_1$, $a := a_1$ and $c := (\prod_{n=1}^{N} B_{n,1})^{-1}\exp(-\langle \boldsymbol{a}_{2:M}, \boldsymbol{\lambda}_{2:M}\rangle)$.

Appendix C
Proof of Lemma 7

To obtain the desired integral assume $\boldsymbol{l}_{1:N\setminus k}$ is arbitrary but fixed and consider the function

$$f(\boldsymbol{x}, l_k) := \exp(-\langle \boldsymbol{l}, \boldsymbol{x}\rangle). \tag{51}$$

As f is jointly continuous in the variables $\boldsymbol{x}$, l_k and $\frac{\partial}{\partial l_k}f(\boldsymbol{x}, l_k)$ is continuous it holds that (see [24, Th. 8.11.2])

$$\int_\Delta x_k \exp(-\langle \boldsymbol{l}, \boldsymbol{x}\rangle)\, d\boldsymbol{x} = \int_\Delta -\frac{\partial}{\partial l_k} f(\boldsymbol{x}, l_k)\, d\boldsymbol{x}$$
$$= -\frac{\partial}{\partial l_k}\int_\Delta f(\boldsymbol{x}, l_k)\, d\boldsymbol{x} = -\frac{\partial}{\partial l_k}\sum_{n=1}^{N+1}\frac{\exp(-\langle \boldsymbol{l}, \boldsymbol{s}_n\rangle)}{\prod_{n'\neq n}\langle \boldsymbol{l}, \boldsymbol{s}_{n'}-\boldsymbol{s}_n\rangle}.$$

Carrying out the differentiation yields the desired result.

Appendix D
Proof of Lemma 8

First we obtain the transform pair $F(\lambda)\bullet\!\!-\!\!\circ f(t)$ ($\lambda_n^{(0)} \neq \lambda_{n'}^{(0)}$ for $n \neq n' \in \{1,\dots,N-1\}$)

$$\frac{\exp(a\lambda)}{(\lambda-\lambda_1^{(0)})\prod_{n=1}^{N-1}(\lambda-\lambda_n^{(0)})} \overset{a\geq 0}{\bullet\!\!-\!\!\circ} \frac{(t-a)_+\exp(\lambda_1^{(0)}(t-a))}{\prod_{n=2}^{N-1}(\lambda_1^{(0)}-\lambda_n^{(0)})}$$
$$+\mathbb{1}_+(t-a)\sum_{n=2}^{N-1}\frac{\exp(\lambda_n^{(0)}(t-a))-\exp(\lambda_1^{(0)}(t-a))}{(\lambda_n^{(0)}-\lambda_1^{(0)})\prod_{n'\neq n}(\lambda_n^{(0)}-\lambda_{n'}^{(0)})} \tag{52}$$

by using the partial fraction expansion (53)

$$\frac{1}{(\lambda-\lambda_1^{(0)})\prod_{n=1}^{N-1}(\lambda-\lambda_n^{(0)})} = \tag{53}$$

$$= \frac{1}{(\lambda-\lambda_1^{(0)})}\sum_{n=1}^{N-1}\frac{1}{\lambda-\lambda_n^{(0)}}\frac{1}{\prod_{n'\neq n}(\lambda_n^{(0)}-\lambda_{n'}^{(0)})} \tag{54}$$

in conjunction with the transform pairs (LT4), (LT5) and (LT6). By assumption, $\boldsymbol{B}_{1,:} = \boldsymbol{B}_{2,:}$ and $\boldsymbol{\lambda}_{2:M}$ is arbitrary but fixed so that

$$\lambda_n^{(0)} := \begin{cases} -\frac{1}{B_{1,1}}\boldsymbol{B}_{1,2:M}\boldsymbol{\lambda}_{2:M}, & n=1 \\ -\frac{1}{B_{n,1}}\boldsymbol{B}_{n+1,2:M}\boldsymbol{\lambda}_{2:M}, & n\in\{2,\dots,N-1\} \end{cases} \tag{55}$$

are pairwise distinct and (37) follows from (52) by setting $\lambda := \lambda_1$, $a := a_1$ and multiplying both sides by $c := (\prod_{n=1}^{N} B_{n,1})^{-1}\exp(-\langle \boldsymbol{a}_{2:M}, \boldsymbol{\lambda}_{2:M}\rangle)$.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[2] S. Mallat, "Understanding deep convolutional networks," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, pp. 20150203, 2016.

[3] T. Wiatowski and H. Bölcskei, "A mathematical theory of deep convolutional neural networks for feature extraction," *arXiv preprint arXiv:1512.06293*, 2015.

[4] B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang, "Maximal sparsity with deep networks?," in *Advances in Neural Information Processing Systems*, 2016, pp. 4340–4348.

[5] U. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 747–751, 2016.

[6] Z. Wang, Q. Ling, and T. Huang, "Learning deep l0 encoders," in *AAAI Conference on Artificial Intelligence*, 2016, pp. 2194–2200.

[7] S. Limmer and S. Stanczak, "Towards optimal nonlinearities for sparse recovery using higher-order statistics," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.

[8] M. Fornasier and H. Rauhut, "Compressive sensing," in *Handbook of mathematical methods in imaging*, pp. 187–228. Springer, 2011.

[9] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[10] D. Tse and P. Viswanath, *Fundamentals of wireless communication*, Cambridge university press, 2005.

[11] B. Makarov and A. Podkorytov, *Real analysis: Measures, integrals and applications*, Springer Science & Business Media, 2013.

[12] S. Kay, "Fundamentals of statistical signal processing, volume i: estimation theory," 1993.

[13] J. B. Lasserre and E. S. Zeron, "A laplace transform algorithm for the volume of a convex polytope," *Journal of the ACM*, vol. 48, no. 6, pp. 1126–1140, 2001.

[14] J. B. Lasserre, "Volume of slices and sections of the simplex in closed form," *Optimization Letters*, vol. 9, no. 7, pp. 1263–1269, 2015.

[15] Y. Brychkov, V. K. Tuan, H. J. Glaeske, and A. Prudnikov, "Multidimensional integral transformations," 1992.

[16] H. J. Glaeske, A. Prudnikov, and K. Skòrnik, "Operational calculus and related topics," 2006.

[17] Y. Jia et al., "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[18] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[19] V. Baldoni, N. Berline, J. De Loera, M. Köppe, and M. Vergne, "How to integrate a polynomial over a simplex," *Mathematics of Computation*, vol. 80, no. 273, pp. 297–325, 2011.

[20] G. Teschl, "Topics in real and functional analysis," 2014.

[21] B. Büeler, A. Enge, and K. Fukuda, "Exact volume computation for polytopes: a practical study," in *Polytopes, combinatorics and computation*. Springer, 2000, pp. 131–154.

[22] J. B. Lasserre, "An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbb{R}^n$," *Journal of optimization theory and applications*, vol. 39, no. 3, pp. 363–377, 1983.

[23] G. Doetsch, *Introduction to the Theory and Application of the Laplace Transformation*, Springer Science & Business Media, 2012.

[24] J. Dieudonne, *Foundations of Modern Analysis*, vol. 1, Academic Press, 1969.