Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures

Hershey, J.R.; Le Roux, J.; Weninger, F.

TR2014-117 August 2014

Abstract

Model-based methods and deep neural networks have both been tremendously successful paradigms in machine learning. In model-based methods, we can easily express our problem domain knowledge in the constraints of the model at the expense of difficulties during inference. Deterministic deep neural networks are constructed in such a way that inference is straightforward, but we sacrifice the ability to easily incorporate problem domain knowledge. The goal of this paper is to provide a general strategy to obtain the advantages of both approaches while avoiding many of their disadvantages. The general idea can be summarized as follows: given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network. We then de-couple the model parameters across layers to obtain novel neural-network-like architectures that can easily be trained discriminatively using gradient-based methods. The resulting formula combines the expressive power of a conventional deep network with the internal structure of the model-based approach, while allowing inference to be performed in a fixed number of layers that can be optimized for best performance. We show how this framework can be applied to the non-negative matrix factorization to obtain a novel non-negative deep neural network architecture, that can be trained with a multiplicative back-propagation-style update algorithm. We present experiments in the domain of speech enhancement, where we show that the resulting model is able to outperform conventional neural network while only requiring a fraction of the number of parameters. We believe this is due to the ability afforded by our framework to incorporate problem level assumptions into the architecture of the deep network.

arXiv.org

Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures

John R. Hershey
MERL
Cambridge, MA, USA
hershey@merl.com

Jonathan Le Roux
MERL
Cambridge, MA, USA
leroux@merl.com

Felix Weninger
TUM
Munich, Germany
weninger@tum.de

Abstract

Model-based methods and deep neural networks have both been tremendously successful paradigms in machine learning. In model-based methods, we can easily express our problem domain knowledge in the constraints of the model at the expense of difficulties during inference. Deterministic deep neural networks are constructed in such a way that inference is straightforward, but we sacrifice the ability to easily incorporate problem domain knowledge. The goal of this paper is to provide a general strategy to obtain the advantages of both approaches while avoiding many of their disadvantages. The general idea can be summarized as follows: given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network. We then de-couple the model parameters across layers to obtain novel neural-network-like architectures that can easily be trained discriminatively using gradient-based methods. The resulting formula combines the expressive power of a conventional deep network with the internal structure of the model-based approach, while allowing inference to be performed in a fixed number of layers that can be optimized for best performance. We show how this framework can be applied to the non-negative matrix factorization to obtain a novel non-negative deep neural network architecture, that can be trained with a multiplicative back-propagation-style update algorithm. We present experiments in the domain of speech enhancement, where we show that the resulting model is able to outperform conventional neural network while only requiring a fraction of the number of parameters. We believe this is due to the ability afforded by our framework to incorporate problem level assumptions into the architecture of the deep network.

1 Introduction

Two of the most successful general approaches to machine learning are model-based methods and deep neural networks (DNNs). Each offers important well-known advantages and disadvantages. The goal of this paper is to provide a general strategy to obtain the advantages of both approaches while avoiding many of their disadvantages. The general idea can be summarized as follows: given a model-based approach that requires an iterative inference method, we *unfold* the iterations into a layer-wise structure analogous to a neural network. We then *de-couple* the model parameters across layers to obtain novel neural-network-like architectures that can easily be trained discriminatively using gradient-based methods. The resulting formula combines the expressive power of a conventional deep network with the internal structure of the model-based approach, while allowing inference to be performed in a fixed number of layers that can be optimized for best performance. We call this approach *deep unfolding*.

A chief advantage of model-based approaches, such as probabilistic graphical models, is that they allow us to use prior knowledge and intuition to reason at the *problem level* in devising inference algorithms, or what David Marr called the "computational theory" level of analysis [1, 2]. Important assumptions about problem constraints can often be incorporated into a model-based approach in a straightforward way. Examples include constraints from the world, such as the linear additivity of signals, visual occlusion, three-dimensional geometry, as well as more subtle statistical assumptions such as conditional independence, latent variable structure, sparsity, low-rank covariances, and so on. By hypothesizing and testing different problem-level constraints, insight into the nature of the problem can be gained and used as inspiration to improve the modeling assumptions [2]. Unfortunately, inference in probabilistic models can be both mathematically and computationally intractable. Approximate methods, such as belief propagation and variational approximations, allow us to derive iterative algorithms to infer the latent variables of interest. However, despite greatly improving the situation, such iterative methods are still often too slow for time-sensitive applications. In such cases, rigorous discriminative optimization of such models can be challenging because they may involve bi-level optimization, where the parameter optimization depends in turn on an iterative inference algorithm [3].

Deterministic deep neural networks, which have recently become the state of the art in many applications, are formulated such that the inference is defined as a finite closed-form expression, organized into *layers* which are typically executed in sequence. Discriminative training of the networks can be used to optimize the speed versus accuracy trade-off, and has become indispensable in producing systems that perform very well in a particular application. However, a well-known disadvantage is that conventional DNNs are closer to mechanisms than problem-level formulations, and can be considered essentially "black-box" methods. It is difficult to incorporate prior knowledge about the world or the problem. Moreover, even when one has a working DNN system, it is not clear how it actually achieves its results, and so discovering how to modify its architecture to achieve better results could be considered as much an art as a science.

The proposed methodology addresses these difficulties by bringing the problem-level formulation of model-based methods to the task of designing deep neural network architectures. Each step of the process can be solved by well-known methods: deriving iterative inference methods for a given probabilistic model follows a long tradition that makes use of many well-known tools, and unfolding the iterations and applying the chain rule for gradient-based training is also relatively straightforward.

First we discuss the general framework, as well as its application in specific inference algorithms. Then we present experiments in the domain of speech enhancement, using models based on non-negative matrix factorization (NMF) [4, 5, 6]. In this domain, the NMF model embodies the problem-level assumption that signals mix linearly and therefore their power spectra are approximately additive. Despite the apparent simplicity of the NMF model, it has no known closed-form solution and relies on iterative inference methods, typically formulated as multiplicative updates. A novel deep network architecture results from unfolding the iterations and decoupling the parameters. This architecture can be more powerful than the original model-based NMF method, while still incorporating the basic additivity assumption from its problem-level analysis. Moreover it takes the form of a non-negative deep network, for which multiplicative updates can be derived that preserve non-negativity, without the need for constrained optimization. This is significant because it would otherwise be difficult to arrive at the NMF-based architecture by reasoning directly from a conventional deep network. By reasoning at the problem level with the model-based approach, our methodology allows one to derive inference architectures and training methods that otherwise would be very difficult to obtain.

Main contributions of this paper: The novel contributions of this paper include: a framework for deriving novel deep network architectures from problem-level modeling constraints; a novel non-negative deep network with non-negative parameters; multiplicative updates for training non-negative deep networks; finally, experiments showing the benefit of this approach in the domain of speech enhancement.

2 General formulation of deep unfolding

In the general setting we consider models for which inference is an optimization problem. One example is variational inference, where a lower bound on the data likelihood is optimized to estimate approximate posterior probabilities, which can then be used to compute conditional expected values of hidden variables. As another example, loopy belief propagation is an iterative algorithm that enforces local consistency constraints on marginal posteriors. When it converges, the fixed points correspond to stationary points of the Bethe free energy [7]. Finally, non-negative matrix factorization is a non-negative basis function expansion model, whose objective function can be optimized by simple multiplicative update rules.

Here, we present a general formulation based on a model, determined by *parameters* θ, that specifies the relationships between *hidden* quantities of interest y_i and the *observed* variables x_i for each data instance i. At test time, estimating these quantities of interest involves optimizing an inference objective function $\mathcal{F}_\theta(x_i, \phi_i)$, where ϕ_i are intermediate variables (considered as vectors) from which y_i can be computed: [1]

$$\hat{\phi}_i(x_i|\theta) = \arg\min_{\phi_i} \mathcal{F}_\theta(x_i, \phi_i), \quad \hat{y}_i(x_i|\theta) = g_\theta(x_i, \hat{\phi}_i(x_i|\theta)), \tag{1}$$

where g_θ is an estimator for y_i. For many interesting cases, this optimization cannot be easily done and leads to an iterative inference algorithm. In probabilistic generative models, $\mathcal{F}$ might be an approximation to the negative log likelihood, y_i could be taken to represent hidden variables and ϕ_i to represent an estimate of their posterior distribution. For example, in variational inference algorithms, ϕ_i could be taken to be the variational parameters. In sum-product loopy belief propagation, the ϕ_i would be the posterior marginal probabilities. On the other hand, for the non-probabilistic formulation of NMF, ϕ_i can be taken as the activation coefficients of the basis functions that are updated at inference time. Note that the x_i, y_i can all be sequences or have other underlying structure, but here for simplicity we ignore their structure.

At training time, we may optimize the parameters θ using a discriminative objective function,

$$\mathcal{E}_\theta \overset{\text{def}}{=} \sum_i \mathcal{D}(y_i^*, \hat{y}_i(x_i|\theta)), \tag{2}$$

where $\mathcal{D}$ is a loss function and y_i^* a reference value. In some settings, we can also consider a discriminative objective $\mathcal{D}(y_i^*, \hat{\phi}_i(x_i|\theta))$ which computes an expected loss. In the general case, (2) is a *bi-level* optimization problem since $\hat{y}_i(x_i|\theta)$ is itself determined by an optimization problem (1) that depends on the parameters θ.

We assume that the intermediate variables ϕ_i in (1) can be optimized iteratively using update steps $k \in \{1 \dots K\}$ of the form [2]

$$\phi_i^k = f_\theta(x_i, \phi_i^{k-1}), \tag{3}$$

beginning with $\phi_i^{(0)}$. Note that, although all steps are assumed to use the same f_θ, it may be composed of smaller steps, each of which are different. This can occur in loopy belief propagation, when different messages are passed in each step, or in variational inference, when different variational parameters are updated in each step.

Rather than considering this iteration as an algorithm, let us consider unfolding it as a sequence of layers in a neural network-like architecture, where the iteration index is now interpreted as an index to the neural network layer. The intermediate variables $\phi^1, \dots, \phi^K$ are the nodes of layers 1 to K and (3) determines the transformation and activation function between layers. Finally, the y_i^K are the nodes of the output layer, and are obtained by $y_i^K = g_\theta(x_i, \phi_i^K)$.

At this point, the parameters θ, which are tied across layers, could be discriminatively trained using gradient-based methods with the chain rule, as in the back-propagation algorithm for neural networks. This is reminiscent of "back-propagation in time" for neural networks, where the weight matrices are the same for each un-folded time step. This kind of approach has recently attracted interest in the context of several particular models, such as sparse coding [8, 9] and non-negative matrix factorization [10, 11]. Hinton et al. [12] also showed that restricted Boltzmann machine inference can be seen as an infinite directed belief network with tied weights.

However, in the deep unfolding framework, we recognize that using the same parameters from layer to layer is not strictly necessary in the context of discriminative training, and may even be

[1]We arbitrarily formulate it as a minimization, as in the case of energy minimization, but equivalently it could be a maximization as in the case of probabilities.

[2]Indices k in superscript always refer to iteration index (similarly for l defined later as the source index).

detrimental to performance. Therefore, we explicitly *untie* the parameters across layers to form a more powerful deep neural network. Besides allowing the network to fit more complex functions, we speculate that this untying may also reduce the susceptibility to local minima. Of course the cost of untying is the possibility of over-fitting, but this is the same situation for all deep neural networks, and can be handled in similar ways.

To formulate this untying, we define parameters $\theta \overset{\text{def}}{=} \{\theta^k\}_{k=0}^K$ for each layer, so that $\phi_i^k = f_{\theta^{k-1}}(x_i, \phi_i^{k-1})$ and $y_i^K = g_{\theta^K}(x_i, \phi_i^K)$. Then we can compute the derivatives recursively as in back-propagation,

$$\frac{\partial \mathcal{E}}{\partial \phi_i^K} = \frac{\partial \mathcal{D}}{\partial y_i^K}\frac{\partial y_i^K}{\partial \phi_i^K}, \qquad \frac{\partial \mathcal{E}}{\partial \theta^K} = \sum_i \frac{\partial \mathcal{D}}{\partial y_i^K}\frac{\partial y_i^K}{\partial \theta^K}, \tag{4}$$

$$\frac{\partial \mathcal{E}}{\partial \phi_i^k} = \frac{\partial \mathcal{E}}{\partial \phi_i^{k+1}}\frac{\partial \phi_i^{k+1}}{\partial \phi_i^k}, \qquad \frac{\partial \mathcal{E}}{\partial \theta^k} = \sum_i \frac{\partial \mathcal{E}}{\partial \phi_i^{k+1}}\frac{\partial \phi_i^{k+1}}{\partial \theta^k}, \tag{5}$$

where $k < K$, and we sum over all the intermediate indices of the derivatives. The specific derivations will of course depend on the form of f, g and $\mathcal{D}$, for which we give examples below.

3 Deep discriminative non-negative matrix factorization

Here we apply the proposed deep unfolding framework to the non-negative matrix factorization (NMF) model [4], which can be applied to any non-negative signal. Although NMF can be applied in many domains, here we focus on the task of single-channel source separation, which aims to recover source signals from mixtures. In this context it encompasses the simple problem-level assumptions that power or magnitude spectra of different sources approximately add together, and that each source can be described as a linear combination of non-negative basis functions.

NMF operates on a matrix of F-dimensional non-negative spectral features, usually the power or magnitude spectrogram of the mixture, $\mathbf{M} = [\mathbf{m}_1 \cdots \mathbf{m}_T]$, where T is the number of frames and $\mathbf{m}_t \in \mathbb{R}_+^F$, $t = 1, \ldots, T$ are obtained by short-time Fourier transformation of the time-domain signal. With L sources, a set of R_l non-negative basis vectors $\mathbf{w}_1^l, \cdots, \mathbf{w}_{R_l}^l$ is assumed for each source $l \in \{1, \ldots, L\}$, and concatenated into matrices $\mathbf{W}^l = [\mathbf{w}_1^l \cdots \mathbf{w}_{R_l}^l]$. A column-wise normalized $\widetilde{\mathbf{W}}^l$ can be used to avoid scaling indeterminacy. The basic assumptions can then be written as

$$\mathbf{M} \approx \sum_l \mathbf{S}^l \approx \sum_l \widetilde{\mathbf{W}}^l \mathbf{H}^l = \widetilde{\mathbf{W}}\mathbf{H}. \tag{6}$$

The β-divergence, D_β, is an appropriate cost function for this approximation [13], which casts inference as an optimization of $\hat{\mathbf{H}}$,

$$\hat{\mathbf{H}} = \arg\min_{\mathbf{H}} D_\beta(\mathbf{M} \mid \widetilde{\mathbf{W}}\mathbf{H}) + \mu|\mathbf{H}|_1. \tag{7}$$

For $\beta = 1$, D_β is the generalized KL divergence, whereas $\beta = 2$ yields the squared Euclidean distance. An L1 sparsity constraint with weight μ is added to favor solutions where only few basis vectors are active at a time.

The following multiplicative updates minimize (7) subject to non-negativity constraints [13],

$$\mathbf{H}^{k+1} = \mathbf{H}^k \circ \frac{\widetilde{\mathbf{W}}^T(\mathbf{M} \circ (\mathbf{\Lambda}^k)^{\beta-2})}{\widetilde{\mathbf{W}}^T(\mathbf{\Lambda}^k)^{\beta-1} + \mu}, \tag{8}$$

for iteration $k \in 1, \ldots, K$, where $\circ$ denotes element-wise multiplication, the matrix quotient is element-wise, $\mathbf{\Lambda}^k := \widetilde{\mathbf{W}}\mathbf{H}^k$. $\mathbf{H}^0$ is initialized randomly.

After K iterations, to reconstruct each source, typically a Wiener filtering-like approach is used, which enforces the constraint that all the source estimates $\mathbf{W}^l\mathbf{H}^l$ sum up to the mixture:

$$\widetilde{\mathbf{S}}^{l,K} = \frac{\widetilde{\mathbf{W}}^l \mathbf{H}^{l,K}}{\sum_{l'} \widetilde{\mathbf{W}}^{l'} \mathbf{H}^{l',K}} \circ \mathbf{M}. \tag{9}$$

While in general, NMF bases are trained independently on each source before being combined, the combination is not trained discriminatively for good separation performance from a mixture. Recently, discriminative methods have been applied to sparse dictionary based methods to achieve better performance in particular tasks [14]. In a similar way, we can discriminatively train NMF bases for source separation. The following optimization problem for training bases, termed *discriminative NMF* (DNMF) was proposed in [6, 11]:

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \sum_l \gamma_l D_\beta \left(\mathbf{S}^l \mid \mathbf{W}^l \hat{\mathbf{H}}^l(\mathbf{M}, \mathbf{W}) \right), \tag{10}$$

$$\text{where } \hat{\mathbf{H}}(\mathbf{M}, \mathbf{W}) = \arg\min_{\mathbf{H}} D_\beta(\mathbf{M} \mid \widetilde{\mathbf{W}}\mathbf{H}) + \mu|\mathbf{H}|_1, \tag{11}$$

and γ_l are weights accounting for the application-dependent importance of the source l; for example, in speech de-noising, we focus on reconstructing the speech signal. The first part (10) minimizes the reconstruction error given $\hat{\mathbf{H}}$. The second part ensures that $\hat{\mathbf{H}}$ are the activations that arise from the test-time inference objective. Given the bases $\mathbf{W}$, the activations $\hat{\mathbf{H}}(\mathbf{M}, \mathbf{W})$ are uniquely determined, due to the convexity of (11). Nonetheless, the above remains a difficult bi-level optimization problem, since the bases $\mathbf{W}$ occur in both levels.

In [11] the bi-level problem was approached by directly solving for the derivatives of the lower level problem after convergence. In [6] the problem was approached by untying the bases used for reconstruction in (10) from the analysis bases used in (11), and training only the reconstruction bases. In addition, (9) was incorporated into the discriminative criteria as

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \sum_l \gamma_l D_\beta \left(\mathbf{S}^l \mid \widetilde{\mathbf{S}}^{l,K}(\mathbf{M}, \mathbf{W}) \right), \tag{12}$$

This model can be considered a first step toward the proposed approach in the context of NMF.

Here, based on our framework, we unfold the entire model as a deep non-negative neural network, and we untie the parameters across layers as $\mathbf{W}^k$ for $k \in \{1 \dots K\}$. We call this new model *deep discriminative NMF* (DDNMF). We cast this into our general formulation by defining

$$i = t, \quad x = \mathbf{m}, \quad y^* = \mathbf{S}^l, \quad \phi^k = \mathbf{H}^k, \quad y^K = \widetilde{\mathbf{S}}^{l,K}, \quad \theta^k = \mathbf{W}^k.$$

We identify the inference objective and estimator (1) with (11) and (9), the discriminative objective (2) with (12), and the iterative updates (3) with (8).

In order to train this network while respecting the non-negativity constraints, we derive recursively defined multiplicative update equations by back-propagating a split between positive and negative parts of the gradient. Indeed, in NMF, multiplicative updates are often derived using a heuristic approach which uses the ratio of the negative part to the positive part as a multiplication factor to update the value of that variable of interest:

$$\mathbf{W}^{k+1} = \mathbf{W}^k \circ \frac{[\nabla_{\mathbf{W}^k} \mathcal{E}]_-}{[\nabla_{\mathbf{W}^k} \mathcal{E}]_+}. \tag{13}$$

To propagate the positive parts (and similarly the negative parts, with appropriate changes), we use:

$$\left[\frac{\partial \mathcal{E}}{\partial h^k_{r_k,t}}\right]_+ = \sum_{r_{k+1}} \left(\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_k,t}}\right]_+ \left[\frac{\partial h^{k+1}_{r_k,t}}{\partial h^k_{r_k,t}}\right]_+ + \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- \left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_- \right), \tag{14}$$

$$\left[\frac{\partial \mathcal{E}}{\partial w^k_{f,r}}\right]_+ = \sum_t \sum_{r_{k+1}} \left(\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{f,r}}\right]_+ \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_+ + \left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{f,r}}\right]_- \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- \right). \tag{15}$$

4 Experiments

The DDNMF method was evaluated along with competitive models on the 2nd CHiME Speech Separation and Recognition Challenge corpus [3]. The task is speech enhancement in reverberated

[3] http://spandh.dcs.shef.ac.uk/chime_challenge/ – as of June. 2014

noisy mixtures ($S = 2$, $l = 1$: speech, $l = 2$: noise). The background is mostly non-stationary noise sources such as children, household appliances, television, radio, and so on, recorded in a home environment. Training, development, and test sets of noisy mixtures along with noise-free reference signals are created from the Wall Street Journal (WSJ-0) corpus of read speech and a corpus of training noise recordings. The dry speech recordings are convolved with time-varying room impulse responses estimated from the same environment as the noise. The training set consists of 7 138 utterances at six SNRs from -6 to 9 dB, in steps of 3 dB. The development and test sets consist of 410 and 330 utterances at each of these SNRs, for a total of 2 460 / 1 980 utterances. By construction of the WSJ-0 corpus, our evaluation is speaker-independent. The background noise recordings in the development and test set are different from the training noise recordings, and different room impulse responses are used to convolve the dry utterances. In this paper, we present results on the development set. To reduce complexity we use only 10 % of the training utterances for all methods. Our evaluation measure for speech separation is source-to-distortion ratio (SDR) [15].

4.1 Feature extraction

Each feature vector concatenates $T = 9$ consecutive frames of left context, ending with the target frame, obtained as short-time Fourier spectral magnitudes, using 25 ms window size, 10 ms window shift, and the square root of the Hann window. This leads to feature vectors of size TF where $F = 200$ is the number of frequencies. Similarly to the features in $\mathbf{M}$, each column of $\hat{\mathbf{S}}^l$ corresponds to a sliding window of consecutive reconstructed frames. Only the last frame in each sliding window is reconstructed, which leads to an on-line algorithm. For the NMF-based approaches, we use the same number of basis vectors for speech and noise ($R^1 = R^2$), and consider $R^l = 100$ and $R^l = 1000$. We denote the total as $R = \sum_l R^l$. We look at two regimes of maximum iterations, $K = 4$ for which NMF-based approaches still have significant room for improvement in performance, and $K = 25$ for which, based on preliminary experiments, they are close to asymptotic performance.

4.2 Baseline 1: Deep Neural Network

To compare our DDNMF architecture with standard K-layer deep neural networks, we used the following setting. The feed-forward DNNs have $K-1$ hidden layers with hyperbolic tangent activation functions and an output layer with logistic activation functions. Denoting the output layer activations for time index t by $\mathbf{y}_t = (y_{1,t}, \ldots, y_{F,t})^T \in [0,1]^F$, the DNN computes the deterministic function

$$\mathbf{y}_t = \sigma(\mathbf{W}^K \tanh(\mathbf{W}^{K-1} \cdots \tanh(\mathbf{W}^1 \mathbf{x}_t))),$$

where $\mathbf{x}_t$ are the input feature vectors and σ and $\tanh$ denote element-wise logistic and hyperbolic tangent functions. As in the DDNMF experiments, $T = 9$ consecutive frames of context are concatenated together, but here the vectors $\mathbf{x}_t$ are logarithmic magnitude spectra. Thus, the only difference in the input feature representation with respect to DDNMF is the compression of the spectral amplitudes, which is generally considered useful in speech processing, but breaks the linearity assumption of NMF.

Previous attempts with DNNs have focused on direct estimation of the clean speech without taking into account the mixture in the output layer, or on direct estimation of a masking function without considering its effect upon the speech estimate. Here, based on our experience with model-based approaches, we train the masking function such that, when applied to the mixture, it best reconstructs the clean speech, which was also proposed in [16]. This amounts to optimizing the following objective function for the DNN training:

$$E = \sum_{f,t} (y_{f,t} m_{f,t} - s^l_{f,t})^2 = \sum_{f,t} (\tilde{s}^l_{f,t} - s^l_{f,t}),$$

where m are the mixture magnitudes and s^l are the speech magnitudes. Thus, the sequence of output layer activations $\mathbf{y}_t$ can be interpreted as a time-frequency mask in the magnitude spectral domain, similar to the 'Wiener filter' in the output layer of DDNMF (12). In our experiments, this approach leads to 1.5 dB improvements relative to mask estimation. Although this comes from the model-based approach, we include it here so that the DNN results are comparable solely on the context of the deep architecture and not the output layer.

Table 1: DNN source separation performance on the CHiME development set for various topologies.

SDR [dB]	Input SNR [dB]							
Topology	-6	-3	0	3	6	9	Avg	# params
3x256	3.71	5.78	7.71	9.08	10.80	12.75	8.31	644 K
1x1024	5.10	7.12	8.84	10.13	11.80	13.58	9.43	2.0 M
2x1024	5.14	7.18	8.87	10.20	11.85	13.66	9.48	3.1 M
3x1024	4.75	6.74	8.47	9.81	11.53	13.38	9.11	4.1 M
2x1536	5.42	7.26	8.95	10.21	11.88	13.67	9.57	5.5 M

Our implementation is based on the open-source software CURRENNT[4]. During training, the above objective function is minimized on the CHiME training set, using back-propagation, stochastic gradient descent with momentum, and discriminative layer-wise pre-training. Early stopping based on cross-validation with the CHiME development set, and Gaussian input noise (standard deviation 0.1) are used to prevent aggressive over-optimization on the training set. Unfortunately, our current experiments for DDNMF do not use cross-validation, but despite the advantage this gives to the DNN, as shown below DDNMF nevertheless performs better.

We investigate different DNN topologies (number of layers and number of hidden units per layer) in terms of SDR performance on the CHiME development set. Results are shown in Table 1.

4.3 Baseline 2: sparse NMF

Sparse NMF (SNMF) [17] is used as a baseline, by optimizing the training objective,

$$\overline{\mathbf{W}}^l, \overline{\mathbf{H}}^l = \underset{\mathbf{W}^l, \mathbf{H}^l}{\arg\min}\, D_\beta(\mathbf{S}^l \mid \widetilde{\mathbf{W}}^l \mathbf{H}^l) + \mu |\mathbf{H}^l|_1, \tag{16}$$

for each source, l. NMF and which we shall denote by SNMF. A multiplicative update algorithm to optimize (16) for arbitrary $\beta \geq 0$ is given by [18]. During training, we set $\mathbf{S}^1$ and $\mathbf{S}^2$ in (16) to the spectrograms of the concatenated noise-free CHiME training set and the corresponding background noise in the multi-condition training set. This yields SNMF bases $\overline{\mathbf{W}}^l$, $l = 1, 2$. As initial solution for $\overline{\mathbf{W}}$, we use exemplar bases sampled at random from the training data for each source. For the sparsity weight we use $\mu = 5$, which performed well for SNMF and DNMF algorithms for both $R^l = 100$ and $R^l = 1000$ in the experiments of [6]. In the SNMF experiments, the same basis matrix $\overline{\mathbf{W}}$ is used both for determining $\hat{\mathbf{H}}$ according to (7) and for reconstruction using (9).

4.4 Deep Discriminative NMF

In the DDNMF experiments, the KL divergence ($\beta = 1$) is used for the update equations, but we use squared Euclidean distance ($\beta = 2$) in the discriminative objective (12) since this corresponds closely to the SDR evaluation metric, and this combination performed well in [6]. In all the DDNMF models we initialize the basis sets for all layers using the SNMF bases, $\overline{\mathbf{W}}$, trained as described in Section 4.3. We then consider the C last layers to be *discriminatively trained*, for various values of C. This means that we decouple the bases for the final C layers (counting the reconstruction layer and analysis layers), and we train the bases $\mathbf{W}^k$ for k such that $K - C + 1 \leq k \leq K$ using the multiplicative back-propagation updates described in Section 3. Thus $C = 0$ corresponds to SNMF, $C \geq 1$ corresponds to DDNMF, with the special case $C = 1$ previously described as DNMF [6].

In the experiments, the $K - C$ non-discriminatively trained layers use the full bases $\overline{\mathbf{W}}$, which contain multiple context frames. In contrast the C discriminatively trained layers are restricted to a single frame of context. This is because the network is being trained to reconstruct a single target frame, whereas using the full context in $\mathbf{W}^k$ and $\mathbf{M}$ would enforce the additivity constraints across reconstructions of the full context in each layer. Instead, $\mathbf{W}^{k>K-C}$ is of size $(F \times R)$, and is initialized to the first F rows of $\overline{\mathbf{W}}$ and the matrix $\mathbf{M}'$, consisting of the first F rows of $\mathbf{M}$, is used in place of $\mathbf{M}$. For DDNMF, the fixed basis functions $\overline{\mathbf{W}}$ contain $D_{\mathrm{F}} = TFR$ parameters that are not trained, whereas the final C layers together have $P_{\mathrm{D}} = CFR$ discriminatively trained parameters, for a total of $P = (T + C)FR$.

[4]http://currennt.sf.net/

Table 2: DDNMF source separation performance on CHiME Challenge (WSJ-0) development set.

SDR [dB]	Input SNR [dB]								
$R^l = 100$	-6	-3	0	3	6	9	Avg.	P_D	P
$K = 4,\ \ C = 0$ (SNMF)	2.03	4.66	7.08	8.76	10.67	12.74	7.66	-	360 K
$K = 4,\ \ C = 1$ (DNMF)	2.91	5.43	7.57	9.12	10.97	13.02	8.17	40 K	400 K
$K = 4,\ \ C = 2$	3.19	5.68	7.78	9.28	11.09	13.07	8.35	80 K	440 K
$K = 4,\ \ C = 3$	3.22	5.69	7.79	9.28	11.09	13.05	8.35	120 K	480 K
$K = 4,\ \ C = 4$	3.32	5.76	7.84	9.31	11.11	13.05	8.40	160 K	520 K
$K = 25, C = 0$ (SNMF)	4.16	6.46	8.51	9.90	11.61	13.40	9.01	-	360 K
$K = 25, C = 1$ (DNMF)	4.92	7.09	8.90	10.24	12.02	13.83	9.50	40 K	400 K
$K = 25, C = 2$	5.16	7.28	9.05	10.36	12.12	13.89	9.64	80 K	440 K
$K = 25, C = 3$	5.30	7.38	9.14	10.43	12.18	13.93	9.73	120 K	480 K
$K = 25, C = 4$	5.39	7.44	9.19	10.48	12.22	13.95	9.78	160 K	520 K
$R^l = 1000$	-6	-3	0	3	6	9	Avg.	P_D	P
$K = 4,\ \ C = 0$ (SNMF)	1.79	4.45	6.94	8.66	10.61	12.76	7.54	-	3.6 M
$K = 4,\ \ C = 1$ (DNMF)	2.94	5.45	7.60	9.15	11.00	13.06	8.20	400 K	4 M
$K = 4,\ \ C = 2$	3.14	5.62	7.74	9.26	11.10	13.12	8.33	800 K	4.4 M
$K = 4,\ \ C = 3$	3.36	5.80	7.89	9.37	11.19	13.18	8.47	1.2 M	4.8 M
$K = 4,\ \ C = 4$	3.55	5.95	8.01	9.48	11.28	13.23	8.58	1.6 M	5.2 M
$K = 25, C = 0$ (SNMF)	4.39	6.60	8.67	10.06	11.82	13.67	9.20	-	3.6 M
$K = 25, C = 1$ (DNMF)	5.74	7.75	9.55	10.82	12.55	14.35	10.13	400 K	4 M
$K = 25, C = 2$	5.80	7.80	9.59	10.86	12.59	14.39	10.17	800 K	4.4 M
$K = 25, C = 3$	5.84	7.82	9.62	10.89	12.61	14.40	10.20	1.2 M	4.8 M

5 Discussion

Results in terms of SDR are shown for the experiments using DNNs in Table 1, and for the DDNMF family in Table 2, for a range of topologies. The first thing to note is that the DDNMF framework yields strong improvements relative to SNMF. Comparing the DNN and DDNMF approaches, we can first see that the best DDNMF topology achieves an SDR of 10.20 dB, outperforming the best DNN result of 9.57 dB, for a comparable number of parameters (4.8M for DDNMF versus 5.5M for the DNN). Looking more closely at the trade-off between model size and performance shows that the smallest DDNMF topology that outperforms the best DNN topology, is obtained for $R^l = 100, K = 25, C = 2$, and achieves an SDR of 9.64 dB using at least an order of magnitude fewer parameters (only 440K parameters overall, only 80K of which are discriminatively trained). We take this as strong evidence for the benefit of using a model-based approach to derive the deep learning architecture.

Analyzing further the effect of topology on performance for DDNMF, we can see that discriminatively training of the first layer gives the biggest improvement, but training more and more layers consistently improves performance, especially in low SNR conditions, while only adding a modest number of parameters per layer. Moving from $R^l = 100$ to $R^l = 1000$ does not lead to as much gain as one might expect despite the huge increase in parameter size. This could be because we are currently only training on 10 % of the data, and used a fairly conservative convergence criterion. For the same model size, using $K = 25$ layers leads to large gains in performance without increasing training time and complexity. However, it comes at the price of an increased computation cost at inference time. Intermediate topology regimes need to be further explored to get a better sense of the best speed/accuracy trade-off.

As potential further work in the speech enhancement domain, we could consider investigating the application of our framework to models with continuity constraints or factorial structure [19, 20, 21, 22]. More general future research directions within the deep unfolding paradigm include the unfolding of other classes of inference algorithms, such as loopy belief propagation for Markov random fields or variational inference algorithms for intractable generative models.

We have introduced a general framework that allows model-based approaches to guide the exploration of the universe of deep network architectures, which would otherwise be difficult to navigate. We hope that this framework will inspire a new generation of novel deep network architectures suitable for tackling difficult problems that require high-level domain insights.

References

[1] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*, H. Freeman, San Francisco, 1982.

[2] J. R. Hershey, "Perceptual inference in generative models," *Ph.D. Thesis, University of California, San Diego*, 2005.

[3] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of operations research*, vol. 153, no. 1, pp. 235–256, 2007.

[4] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2001, pp. 556–562.

[5] P. Smaragdis, B. Raj, and M. Shashanka, "Supervised and semi-supervised separation of sounds from single-channel mixtures," in *Proc. ICA*, 2007, pp. 414–421.

[6] F. Weninger, J. Le Roux, J. R. Hershey, and S. Watanabe, "Discriminative NMF and its application to single-channel source separation," in *Proc. of ISCA Interspeech*, Sep. 2014.

[7] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Info. Theory*, vol. 51, no. 7, pp. 2282–2312, 2005.

[8] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *ICML*, 2010, pp. 399–406.

[9] P. Sprechmann, R. Litman, T. B. Yakar, A. M. Bronstein, and G. Sapiro, "Supervised sparse analysis and synthesis operators," in *NIPS*, 2013, pp. 908–916.

[10] T. B. Yakar, R. Litman, P. Sprechmann, A. Bronstein, and G. Sapiro, "Bilevel sparse models for polyphonic music transcription," in *ISMIR*, Nov. 2013.

[11] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Supervised non-euclidean sparse NMF via bilevel optimization with applications to speech enhancement," in *HSCMA*, May 2014.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the Itakura-Saito divergence: with application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, March 2009.

[14] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[15] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, July 2006.

[16] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *ICASSP*, May 2014.

[17] J. Eggert and E. Körner, "Sparse coding and NMF," in *Proc. Neural Networks*, 2004, vol. 4, pp. 2529–2533.

[18] P. D. O'Grady and B. A. Pearlmutter, "Discovering convolutive speech phones using sparseness and non-negativity," in *Proc. ICA*, pp. 520–527. 2007.

[19] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation.," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, pp. 1118–1133, 2012.

[20] G. J. Mysore and M. Sahani, "Variational inference in non-negative factorial hidden markov models for efficient audio source separation," in *ICML*, 2012.

[21] C. Févotte, J. Le Roux, and J. R. Hershey, "Non-negative dynamical system with application to speech and audio," in *ICASSP*, May 2013.

[22] U. Simsekli, J. Le Roux, and J. R. Hershey, "Non-negative source-filter dynamical system for speech enhancement," in *ICASSP*, May 2014.

Supplementary material

A Chain rule for the general formulation

We assume in general that the architecture is made of layers where h_i^{k+1} is a function of w^k, h_i^k, and the input x_i. We assume that all variables are vectorized.

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_q^k} &= \sum_i \frac{\partial \mathcal{D}(x_i, y_i)}{\partial w_q^k} \\ &= \sum_i \sum_{p_{k+1}} \left(\sum_{p_{k+2}} \left(\cdots \left(\sum_{p_K} \frac{\partial \mathcal{D}(x_i, y_i)}{\partial h_{i,p_K}^K} \frac{\partial h_{i,p_K}^K (w^{K-1}, h_i^{K-1})}{\partial h_{i,p_{K-1}}^{K-1}} \right) \cdots \right.\right. \\ &\qquad \left.\left. \cdots \right) \frac{\partial h_{i,p_{k+2}}^{k+2}(w^{k+1}, h_i^{k+1})}{\partial h_{i,p_{k+1}}^{k+1}} \right) \frac{\partial h_{i,p_{k+1}}^{k+1}(w^k, h_i^k)}{\partial w_q^k}\end{aligned}$$

or

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_q^k} &= \sum_i \sum_{p_{k+1}} \frac{\partial h_{i,p_{k+1}}^{k+1}(w^k, h_i^k)}{\partial w_q^k} \left(\sum_{p_{k+2}} \frac{\partial h_{i,p_{k+2}}^{k+2}(w^{k+1}, h_i^{k+1})}{\partial h_{i,p_{k+1}}^{k+1}} \left(\cdots \right.\right. \\ &\qquad \left.\left. \cdots \left(\sum_{p_K} \frac{\partial h_{i,p_K}^K (w^{K-1}, h_i^{K-1})}{\partial h_{i,p_{K-1}}^{K-1}} \frac{\partial \mathcal{D}(x_i, y_i)}{\partial h_{i,p_K}^K} \right) \cdots \right) \right)\end{aligned}$$

B Derivations for deep discriminative NMF

B.1 Chain rule for DDNMF

In the case of DNMF, the architecture is defined as follows, with $\mathbf{H}^0$ initialized, e.g., randomly:

$$\mathbf{H}^{k+1} = f(\mathbf{W}^k, \mathbf{M}, \mathbf{H}^k) \tag{17}$$

$$\hat{\mathbf{S}} = g(\mathbf{W}^K, \mathbf{M}, \mathbf{H}^K) \tag{18}$$

The objective function is given by:

$$\mathcal{E} = \sum_l \gamma_l D_\beta(\mathbf{S}^l | \widehat{\mathbf{S}}^l) \tag{19}$$

We assume that the sets of basis functions for all sources are stacked horizontally, and their activations are stacked vertically. In order to be able to assume different numbers of basis functions R_l for each source, the stacked structures are indexed using a single index r, such that the elements corresponding to source l are those for which $r \in I_l$, where we defined the set $I_l = [\![i_l, i_{l+1} [\![$ of integers r such that $i_l \leq r < i_{l+1}$, with $i_1 = 1$ and $i_{l+1} = i_l + R_l$.

We compute the gradient of $\mathcal{E}$ with respect to the parameters in the k-th layer, $w_{n,l,r}^k$.

For $k = K$, the gradient $\frac{\partial \mathcal{E}}{\partial w_{n,r}^K}$ is the one obtained for the reconstruction basis functions in Discriminative NMF.

For $k < K$, we use the chain rule to get:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{n,r}^k} &= \sum_{l'} \gamma_{l'} \frac{\partial D_\beta(\mathbf{S}^{l'} | \hat{\mathbf{S}}^{l'})}{\partial w_{n,r}^k} \\ &= \sum_t \sum_{r_{k+1}} \left(\cdots \left(\sum_{r_K} \left(\sum_{n'} \left(\sum_{l'} \gamma_{l'} \frac{\partial D_\beta(S_{n',t}^{l'} | \hat{S}_{n',t}^{l'})}{\partial \hat{S}_{n',t}^{l'}} \right) \frac{\partial g_{n',t}^l(\mathbf{W}^K, \mathbf{M}, \mathbf{H}^K)}{\partial h_{r_K,t}^K} \right) \frac{\partial f_{r_K,t}(\mathbf{W}^{K-1}, \mathbf{M}, \mathbf{H}^{K-1})}{\partial h_{r_{K-1},t}^{K-1}} \right) \cdots \right. \\ &\qquad \left. \cdots \frac{\partial f_{r_{k+2},t}(\mathbf{W}^{k+1}, \mathbf{M}, \mathbf{H}^{k+1})}{\partial h_{r_{k+1},t}^{k+1}} \right) \frac{\partial f_{r_{k+1},t}(\mathbf{W}^k, \mathbf{M}, \mathbf{H}^k)}{\partial w_{n,r}^k}\end{aligned}$$

This can be rewritten slightly more clearly as:

$$\frac{\partial \mathcal{E}}{\partial w^k_{n,r}} = \sum_t \sum_{r_{k+1}} \frac{\partial f_{r_{k+1},t}(\mathbf{W}^k, \mathbf{M}, \mathbf{H}^k)}{\partial w^k_{n,r}} \left(\sum_{r_{k+2}} \frac{\partial f_{r_{k+2},t}(\mathbf{W}^{k+1}, \mathbf{M}, \mathbf{H}^{k+1})}{\partial h^{k+1}_{r_{k+1},t}} \left(\cdots \right.\right.$$

$$\left.\left. \cdots \sum_{r_K} \frac{\partial f_{r_K,t}(\mathbf{W}^{K-1}, \mathbf{M}, \mathbf{H}^{K-1})}{\partial h^{K-1}_{r_{K-1},t}} \left(\sum_{n'} \left(\sum_{l'} \gamma_{l'} \frac{\partial D_\beta(S^{l'}_{n',t} | \hat{S}^{l'}_{n',t})}{\partial \hat{S}^{l'}_{n',t}} \right) \frac{\partial g^l_{n',t}(\mathbf{W}^K, \mathbf{M}, \mathbf{H}^K)}{\partial h^K_{r_K,t}} \right) \right) \cdots \right)$$

Once the values of h^k have been computed in a forward pass, the gradient can be computed in a backward pass, starting from the top layer, which is the most inner term in the sum above.

We give the expression for all terms in the sum, starting from the inside.

B.2 Top (K-th) layer derivative w.r.t. $\mathbf{H}^K$ with Least-Squares divergence measure

Even though we typically use KL-divergence ($\beta = 1$) to train the analysis basis functions and estimate the activations $\mathbf{H}$, nothing prevents us from using a different discrepancy measure at the final layer. If we want to optimize the signal-to-noise ratio (SNR) in the case where features are magnitude spectra, we can minimize the Euclidean distance $D^l_2 := D_2(\mathbf{S}^l | \hat{\mathbf{S}}^l)$ between the (true) magnitude spectrum of the source l, $\mathbf{S}^l$, and that of its reconstruction, $\hat{\mathbf{S}}^l$. This indeed directly corresponds to maximizing the SNR, neglecting the difference between noisy and oracle phases (we thus optimize for an upper-bound of the actual SNR). Note that similar updates can be obtained for $\beta = 1$ in the last layer as well. We are also only interested in the quality of the reconstruction of a single source, typically speech in a speech enhancement scenario.

Let us thus assume that $\gamma_l = 1, \gamma_{l', l' \neq l} = 0, \beta = 2$ in the reconstruction layer and Wiener filter is used for reconstruction (and that fact is included in the optimization), i.e., we optimize to reconstruct a single source l (e.g., speech), using the L2 norm as the error measure, and assume that the source is reconstructed using

$$g^l(\mathbf{W}^K, \mathbf{M}, \mathbf{H}^K) = \frac{\mathbf{W}^{K,l}\mathbf{H}^{K,l}}{\mathbf{W}^K \mathbf{H}^K} \circ \mathbf{M},$$

where $\mathbf{H}^{K,l} = [\mathbf{h}^K_{i_l}; \cdots ; \mathbf{h}^K_{i_{l+1}-1}]$ (we use the notation $[\mathbf{a}; \mathbf{b}]$ for $[\mathbf{a}^\intercal \mathbf{b}^\intercal]^\intercal$) and $\mathbf{W}^{K,l}$ is defined accordingly. Let us denote $\mathbf{\Lambda} = \mathbf{W}^K \mathbf{H}^K = \sum_l \mathbf{W}^{K,l}\mathbf{H}^{K,l}$, $\mathbf{\Lambda}^l = \mathbf{W}^{K,l}\mathbf{H}^{K,l}$, and $\mathbf{\Lambda}^{\bar{l}} = \mathbf{\Lambda} - \mathbf{\Lambda}^l$. Then we have:

$$[\nabla_{\mathbf{H}^{K,l}} \mathcal{E}]_+ = \mathbf{W}^{K,l\intercal} \frac{\mathbf{M}^2 \circ \mathbf{\Lambda}^l \circ \mathbf{\Lambda}^{\bar{l}}}{\mathbf{\Lambda}^3}, \tag{20}$$

$$[\nabla_{\mathbf{H}^{K,l}} \mathcal{E}]_- = \mathbf{W}^{K,l\intercal} \frac{\mathbf{M} \circ \mathbf{S}^l \circ \mathbf{\Lambda}^{\bar{l}}}{\mathbf{\Lambda}^2}, \tag{21}$$

$$\left[\nabla_{\mathbf{H}^{K,\bar{l}}} \mathcal{E}\right]_+ = \mathbf{W}^{K,\bar{l}\intercal} \frac{\mathbf{M} \circ \mathbf{S}^l \circ \mathbf{\Lambda}^l}{\mathbf{\Lambda}^2}, \tag{22}$$

$$\left[\nabla_{\mathbf{H}^{K,\bar{l}}} \mathcal{E}\right]_- = \mathbf{W}^{K,\bar{l}\intercal} \frac{\mathbf{M}^2 \circ (\mathbf{\Lambda}^l)^2}{\mathbf{\Lambda}^3}, \tag{23}$$

where $\mathbf{H}^{K,\bar{l}} = [\mathbf{H}^{K,1}; \cdots ; \mathbf{H}^{K,l-1}; \mathbf{H}^{K,l+1}; \cdots ; \mathbf{H}^S]$, and $\mathbf{W}^{K,\bar{l}}$ is defined accordingly.

B.3 Intermediate ($k+1$-th) layer derivative w.r.t. $\mathbf{H}^k$ and $\mathbf{W}^k$, $k < K$

Even though we may be using any β divergence for the last layer, and we in fact use $\beta = 2$, i.e., the L2 distance, we use $\beta = 1$, i.e., the KL divergence, for the intermediate layers. We noticed that this combination leads to better results than using the same β all the way through. This may be due to the KL divergence being better suited to decomposing mixtures, as was shown in previous evaluations.

To simplify the update equations for $\mathbf{W}^k$, we assume that $\mathbf{W}^k$ is not normalized in the final layers where it is optimized. However, we do assume that it is normalized in lower layers which are not optimized: the analysis basis functions were indeed trained under that assumption to avoid obtaining trivial solutions regarding sparsity simply by rescaling $\mathbf{W}$ and $\mathbf{H}$. Actually, any layer where $\mathbf{W}^k$ is not optimized can use the normalized versions of the updates without any change.

For the layers where $\mathbf{W}^k$ is not normalized, the update equation for $\mathbf{H}^{k+1}$, which determines the structure of the intermediate layers, is given by:

$$h^{k+1}_{r_{k+1},t}(\mathbf{W}^k, \mathbf{M}, \mathbf{H}^k) = h^k_{r_{k+1},t} \frac{\sum\limits_n w^k_{n,r_{k+1}} \frac{m_{n,t}}{\sum_{r'} w^k_{n,r'} h^k_{r',t}}}{\sum\limits_n w^k_{n,r_{k+1}} + \mu}. \tag{24}$$

Let $\alpha^k_r = \sum_n w^k_{n,r}$ and $\Lambda^k_{n,t} = \sum_{r'} w^k_{n,r'} h^k_{r',t}$. The derivative of the intermediate layers is obtained as:

$$\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}} = \frac{1}{\alpha^k_{r_k} + \mu} \sum_n w^k_{n,r_k} \frac{m_{n,t}}{\Lambda^k_{n,t}} - \frac{h^k_{r_k,t}}{\alpha^k_{r_k} + \mu} \sum_n (w^k_{n,r_k})^2 \frac{m_{n,t}}{(\Lambda^k_{n,t})^2}, \quad \text{for } r_{k+1} = r_k \tag{25}$$

$$\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}} = - \frac{h^k_{r_{k+1},t}}{\alpha^k_{r_{k+1}} + \mu} \sum_n w^k_{n,r_{k+1}} w^k_{n,r_k} \frac{m_{n,t}}{(\Lambda^k_{n,t})^2}, \quad \text{for } r_{k+1} \neq r_k \tag{26}$$

and

$$\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n_0,r_k}} = \frac{h^k_{r_k,t}}{\alpha^k_{r_k} + \mu} \left(1 - \frac{w^k_{n_0,r_k} h^k_{r_k,t}}{\Lambda^k_{n_0,t}}\right) \frac{m_{n_0,t}}{\Lambda^k_{n_0,t}} - \frac{h^k_{r_k,t}}{(\alpha^k_{r_k} + \mu)^2} \left(\sum_n w^k_{n,r_k} \frac{m_{n,t}}{\Lambda^k_{n,t}}\right), \quad \text{for } r_{k+1} = r_k$$

$$\frac{\partial h^{k+1}_{k+1,t}}{\partial w^k_{n_0,r_k}} = - \frac{h^k_{r_k,t}}{\alpha^k_{r_{k+1}} + \mu} \frac{w^k_{n_0,r_{k+1}} h^k_{r_{k+1},t}}{\Lambda^k_{n_0,t}} \frac{m_{n_0,t}}{\Lambda^k_{n_0,t}}, \quad \text{for } r_{k+1} \neq r_k \tag{27}$$

The split in positive and negative parts is as follows for $\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}$:

For $r_{k+1} = r_k$:

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_+ = \frac{1}{\alpha^k_{r_k} + \mu} \sum_n w^k_{n,r_k} \frac{m_{n,t}}{\Lambda^k_{n,t}}, \tag{28}$$

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_- = \frac{h^k_{r_k,t}}{\alpha^k_{r_k} + \mu} \sum_n (w^k_{n,r_k})^2 \frac{m_{n,t}}{(\Lambda^k_{n,t})^2}. \tag{29}$$

For $r_{k+1} \neq r_k$:

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_+ = 0, \tag{30}$$

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_- = \frac{h^k_{r_{k+1},t}}{\alpha^k_{r_{k+1}} + \mu} \sum_n w^k_{n,r_{k+1}} w^k_{n,r_k} \frac{m_{n,t}}{(\Lambda^k_{n,t})^2}. \tag{31}$$

The split in positive and negative parts is as follows for $\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n_0,r_k}}$:

For $r_{k+1} = r_k$:

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n_0,r_k}}\right]_+ = \frac{h^k_{r_k,t}}{\alpha^k_{r_k} + \mu} \left(1 - \frac{w^k_{n_0,r_k} h^k_{r_k,t}}{\Lambda^k_{n_0,t}}\right) \frac{m_{n_0,t}}{\Lambda^k_{n_0,t}}, \tag{32}$$

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n_0,r_k}}\right]_- = \frac{h^k_{r_k,t}}{(\alpha^k_{r_k} + \mu)^2} \left(\sum_n w^k_{n,r_k} \frac{m_{n,t}}{\Lambda^k_{n,t}}\right). \tag{33}$$

For $r_{k+1} \neq r_k$:

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n_0,r_k}}\right]_+ = 0, \tag{34}$$

$$\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n_0,r_k}}\right]_- = h^k_{r_k,t} \frac{m_{n_0,t}}{(\Lambda^k_{n_0,t})^2} \frac{w^k_{n_0,r_{k+1}} h^k_{r_{k+1},t}}{\alpha^k_{r_{k+1}} + \mu}. \tag{35}$$

As we will see in Section B.4.2, thanks to back-propagation, we never have to store these quantities in full.

B.4 Multiplicative back-propagation updates

In NMF, multiplicative updates are often derived using a heuristic approach which splits the gradient of the objective function with respect to the variable of interest into positive and negative parts, and uses the ratio of the negative part to the positive part as a multiplication factor to update the value of that variable of interest, e.g.,

$$\mathbf{W}^{k+1} = \mathbf{W}^k \circ \frac{[\nabla_{\mathbf{W}}\mathcal{E}]_-}{[\nabla_{\mathbf{W}}\mathcal{E}]_+} \tag{36}$$

The ratio and multiplication operations are performed term by term.

In order to obtain multiplicative updates in our setting, we need to recursively compute the positive and negative parts of the gradient with respect to each variable.

B.4.1 General case

The whole gradient can be split into positive and negative terms recursively. For example, for

$$\frac{\partial \mathcal{E}}{\partial h^k_{p_k,i}} = \sum_{p_{k+1}} \frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}} \frac{\partial h^{k+1}_{p_{k+1},i}}{\partial h^k_{p_k,i}} \tag{37}$$

for each sample i, then

$$\left[\frac{\partial \mathcal{E}}{\partial h^k_{p_k,i}}\right]_+ = \sum_{p_{k+1}} \left(\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_+ \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial h^k_{p_k,i}}\right]_+ + \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_- \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial h^k_{p_k,i}}\right]_- \right), \tag{38}$$

$$\left[\frac{\partial \mathcal{E}}{\partial h^k_{p_k,i}}\right]_- = \sum_{p_{k+1}} \left(\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_+ \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial h^k_{p_k,i}}\right]_- + \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_- \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial h^k_{p_k,i}}\right]_+ \right). \tag{39}$$

For $\frac{\partial \mathcal{E}}{\partial w^k_{q_k}}$, the expression involves $\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}$ and $\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial w^k_{q_k}}$, with an extra summation over i:

$$\frac{\partial \mathcal{E}}{\partial w^k_{q_k}} = \sum_i \sum_{p_{k+1}} \frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}} \frac{\partial h^{k+1}_{p_{k+1},i}}{\partial w^k_{q_k}}. \tag{40}$$

The split becomes:

$$\left[\frac{\partial \mathcal{E}}{\partial w^k_{q_k}}\right]_+ = \sum_i \sum_{p_{k+1}} \left(\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_+ \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial w^k_{q_k}}\right]_+ + \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_- \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial w^k_{q_k}}\right]_- \right), \tag{41}$$

$$\left[\frac{\partial \mathcal{E}}{\partial w^k_{q_k}}\right]_- = \sum_i \sum_{p_{k+1}} \left(\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_+ \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial w^k_{q_k}}\right]_- + \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{p_{k+1},i}}\right]_- \left[\frac{\partial h^{k+1}_{p_{k+1},i}}{\partial w^k_{q_k}}\right]_+ \right). \tag{42}$$

Note that this general formulation can be applied to any model with non-negative parameters, even though here we use the NMF variable names.

B.4.2 Details for DDNMF

Splitting the gradient for H:

$$\frac{\partial \mathcal{E}}{\partial w^k_{n,r}} = \sum_t \sum_{r_{k+1}} \frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}} \frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}} \tag{43}$$

then:

$$\left[\frac{\partial \mathcal{E}}{\partial w^k_{n,r}}\right]_+ = \sum_t \sum_{r_{k+1}} \left(\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}}\right]_+ \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_+ + \left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}}\right]_- \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- \right)$$

$$
\begin{aligned}
=&\sum_t \left[\frac{\partial h^{k+1}_{r,t}}{\partial w^k_{n,r}}\right]_+ \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_+ + \sum_t \sum_{r_{k+1}} \left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}}\right]_- \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- \\
=&\sum_t \frac{h^k_{r,t}}{\alpha^k_r+\mu}\left(1-\frac{w^k_{n,r}h^k_{r,t}}{\Lambda^k_{n,t}}\right)\frac{m_{n,t}}{\Lambda^k_{n,t}}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_+ \\
&+\sum_t\sum_{r_{k+1}} h^k_{r,t}\frac{m_{n,t}}{(\Lambda^k_{n,t})^2}\frac{w^k_{n,r_{k+1}}h^k_{r_{k+1},t}}{\alpha^k_{r_{k+1}}+\mu}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- \\
&-\sum_t h^k_{r,t}\frac{m_{n,t}}{(\Lambda^k_{n,t})^2}\frac{w^k_{n,r}h^k_{r,t}}{\alpha^k_r+\mu}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_- \quad \text{(to compensate for the term } r_{k+1}=r_k\text{)} \\
&+\sum_t \frac{h^k_{r,t}}{(\alpha^k_r+\mu)^2}\left(\sum_{n'} w^k_{n',r}\frac{m_{n',t}}{\Lambda^k_{n',t}}\right)\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_- \quad \text{(same)} \\
=&\sum_t \frac{m_{n,t}}{\Lambda^k_{n,t}}\left(\frac{h^k_{r,t}}{\alpha^k_r+\mu}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_+\right) - w^k_{n,r}\sum_t \frac{m_{n,t}}{(\Lambda^k_{n,t})^2}\left(\frac{(h^k_{r,t})^2}{\alpha^k_r+\mu}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_+\right) \\
&+\sum_t h^k_{r,t}\left\{\frac{m_{n,t}}{(\Lambda^k_{n,t})^2}\sum_{r_{k+1}}\frac{w^k_{n,r_{k+1}}}{\alpha^k_{r_{k+1}}+\mu}\left(h^k_{r_{k+1},t}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_-\right)\right\} \\
&-w^k_{n,r}\sum_t \frac{m_{n,t}}{(\Lambda^k_{n,t})^2}\left(\frac{(h^k_{r,t})^2}{\alpha^k_r+\mu}\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_-\right) \\
&+\sum_t \frac{h^k_{r,t}}{(\alpha^k_r+\mu)^2}\left(\sum_{n'} w^k_{n',r}\frac{m_{n',t}}{\Lambda^k_{n',t}}\right)\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_- \qquad (44)
\end{aligned}
$$

It is very important to carefully compute these quantities to avoid having to consider computations or storage involving tensors. It turns out that the above expression can be reformulated using only matrix operations. Altogether, we get:

$$
\begin{aligned}
[\nabla_{\mathbf{W}^k}\mathcal{E}]_+ =& \frac{\mathbf{M}}{\mathbf{\Lambda}^k}\left(\frac{\mathbf{H}^k}{(\mathbf{W}^k)^\intercal \mathbf{1}_{N\times T}+\mu}\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\right)^\intercal \\
&+\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\frac{\mathbf{W}^k}{(\mathbf{W}^k)^\intercal \mathbf{1}_{N\times R}+\mu}\left(\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)\right)\right\}(\mathbf{H}^k)^\intercal \\
&-\mathbf{W}^k\circ\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\left(\frac{(\mathbf{H}^k)^2\circ\left([\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+ + [\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)}{(\mathbf{W}^k)^\intercal \mathbf{1}_{N\times T}+\mu}\right)^\intercal\right\} \\
&+\frac{\mathbf{1}_{R\times T}\left(\mathbf{H}^k\circ\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)^\intercal}{\left((\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu\right)^2}, \qquad (45)
\end{aligned}
$$

where $\mathbf{1}_{a\times b}$ is an $a\times b$ matrix of all 1. This matrix is only used here for notation purposes, and in practice, we use MATLAB's bsxfun to avoid having to explicitely create it.

The negative part of the gradient is exactly the same, except that $\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_-$ and $\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_+$ are interchanged:

$$
\begin{aligned}
\left[\frac{\partial \mathcal{E}}{\partial w^k_{n,r}}\right]_- =& \sum_t\sum_{r_{k+1}}\left(\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}}\right]_+\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- + \left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}}\right]_-\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_+\right) \\
=&\sum_t \left[\frac{\partial h^{k+1}_{r,t}}{\partial w^k_{n,r}}\right]_+ \left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r,t}}\right]_- + \sum_t\sum_{r_{k+1}}\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial w^k_{n,r}}\right]_-\left[\frac{\partial \mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_+ \qquad (46)
\end{aligned}
$$

$$
\begin{aligned}
[\nabla_{\mathbf{W}^k}\mathcal{E}]_- =& \frac{\mathbf{M}}{\mathbf{\Lambda}^k}\left(\frac{\mathbf{H}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)^\intercal \\
&+\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\frac{\mathbf{W}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu}\left(\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\right)\right)\right\}(\mathbf{H}^k)^\intercal \\
&-\mathbf{W}^k\circ\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\left(\frac{(\mathbf{H}^k)^2\circ\left([\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+ + [\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)^\intercal\right\} \\
&+\frac{\mathbf{1}_{R\times T}\left(\mathbf{H}^k\circ\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\right)^\intercal}{((\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu)^2}
\end{aligned}
\tag{47}
$$

Splitting the gradient for $\mathbf{H}^k$: Now the gradient with respect to $\mathbf{H}^k$, which is used recursively:

$$
\begin{aligned}
\left[\frac{\partial\mathcal{E}}{\partial h^k_{r_k,t}}\right]_+ &= \left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_k,t}}\right]_+\left[\frac{\partial h^{k+1}_{r_k,t}}{\partial h^k_{r_k,t}}\right]_+ + \sum_{r_{k+1}}\left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_-\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_-, \\
&= \left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_k,t}}\right]_+\left[\frac{\partial h^{k+1}_{r_k,t}}{\partial h^k_{r_k,t}}\right]_+ + \sum_{r_{k+1}}\left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_-\frac{h^k_{r_{k+1},t}}{\alpha^k_{r_{k+1}}+\mu}\sum_n w^k_{n,r_{k+1}}w^k_{n,r_k}\frac{m_{n,t}}{(\Lambda^k_{n,t})^2} \\
&= \left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_k,t}}\right]_+\left[\frac{\partial h^{k+1}_{r_k,t}}{\partial h^k_{r_k,t}}\right]_+ + \sum_n w^k_{n,r_k}\left\{\frac{m_{n,t}}{(\Lambda^k_{n,t})^2}\left(\sum_{r_{k+1}}\left(\left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_- h^k_{r_{k+1},t}\right)\frac{w^k_{n,r_{k+1}}}{\alpha^k_{r_{k+1}}+\mu}\right)\right\}
\end{aligned}
\tag{48}
$$

where

$$
\left[\frac{\partial h^{k+1}_{r_k,t}}{\partial h^k_{r_k,t}}\right]_+ = \frac{1}{\alpha^k_{r_k}+\mu}\sum_n w^k_{n,r_k}\frac{m_{n,t}}{\Lambda^k_{n,t}}. \tag{49}
$$

This can also be rewritten using matrix computations:

$$
[\nabla_{\mathbf{H}^k}\mathcal{E}]_+ = \frac{[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\circ\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu} + (\mathbf{W}^k)^\intercal\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\mathbf{W}^k\left(\frac{[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\circ\mathbf{H}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)\right)\right\} \tag{50}
$$

The negative part is similar again, just interchanging $[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+$ and $[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-$:

$$
\left[\frac{\partial\mathcal{E}}{\partial h^k_{r_k,t}}\right]_- = \left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_k,t}}\right]_-\left[\frac{\partial h^{k+1}_{r_k,t}}{\partial h^k_{r_k,t}}\right]_+ + \sum_{r_{k+1}}\left(\left[\frac{\partial\mathcal{E}}{\partial h^{k+1}_{r_{k+1},t}}\right]_+\left[\frac{\partial h^{k+1}_{r_{k+1},t}}{\partial h^k_{r_k,t}}\right]_-\right) \tag{51}
$$

and in matrix notations:

$$
[\nabla_{\mathbf{H}^k}\mathcal{E}]_- = \frac{[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\circ\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu} + (\mathbf{W}^k)^\intercal\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\mathbf{W}^k\left(\frac{[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\circ\mathbf{H}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)\right)\right\} \tag{52}
$$

B.5 DDNMF gradient computation procedure

Let us put everything together for the case of single-source reconstruction with Wiener filter and minimization of L2 distance ($\beta = 2$) at the last layer, and KL-divergence NMF updates for $\mathbf{H}^k$ without normalization of $\mathbf{W}^k$ in the layers where we optimize $\mathbf{W}^k$ (but with normalization for all layers up to there). Assuming that a forward computation of $\mathbf{H}^k$ for $k = 1, \ldots, K$ as been performed, we compute positive and negative parts of the gradients as follows:

- Compute the gradient for the last layer with respect to $\mathbf{H}^K$.

$$[\nabla_{\mathbf{H}^{K,l}}\mathcal{E}]_+ = \mathbf{W}^{K,l\intercal}\frac{\mathbf{M}^2\circ\mathbf{\Lambda}^l\circ\mathbf{\Lambda}^{\bar{l}}}{\mathbf{\Lambda}^3}, \tag{53}$$

$$[\nabla_{\mathbf{H}^{K,l}}\mathcal{E}]_- = \mathbf{W}^{K,l\intercal}\frac{\mathbf{M}\circ\mathbf{S}^l\circ\mathbf{\Lambda}^{\bar{l}}}{\mathbf{\Lambda}^2}, \tag{54}$$

$$[\nabla_{\mathbf{H}^{K,\bar{l}}}\mathcal{E}]_+ = \mathbf{W}^{K,\bar{l}\intercal}\frac{\mathbf{M}\circ\mathbf{S}^l\circ\mathbf{\Lambda}^l}{\mathbf{\Lambda}^2}, \tag{55}$$

$$[\nabla_{\mathbf{H}^{K,\bar{l}}}\mathcal{E}]_- = \mathbf{W}^{K,\bar{l}\intercal}\frac{\mathbf{M}^2\circ(\mathbf{\Lambda}^l)^2}{\mathbf{\Lambda}^3}, \tag{56}$$

where $\mathbf{\Lambda}=\mathbf{\Lambda}^K$, $\mathbf{\Lambda}^l=\mathbf{\Lambda}^{K,l}$, $\mathbf{\Lambda}^{\bar{l}}=\mathbf{\Lambda}^{K,\bar{l}}$.

- Compute recursively the gradient for lower layers with respect to $\mathbf{H}^k$ using:

$$\begin{aligned}[\nabla_{\mathbf{H}^k}\mathcal{E}]_+ = {} & \frac{\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu} \\ & + (\mathbf{W}^k)^\intercal\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\frac{\mathbf{W}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu}\left(\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)\right)\right\}\end{aligned} \tag{57}$$

$$\begin{aligned}[\nabla_{\mathbf{H}^k}\mathcal{E}]_- = {} & \frac{\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu} \\ & + (\mathbf{W}^k)^\intercal\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\frac{\mathbf{W}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu}\left(\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\right)\right)\right\}\end{aligned} \tag{58}$$

- Compute the gradient with respect to $\mathbf{W}^k$ for those layers where we want to optimize $\mathbf{W}^k$. If $\mathbf{W}^k$ is tied across layers, the gradients at those layers can simply be summed.

$$\begin{aligned}[\nabla_{\mathbf{W}^k}\mathcal{E}]_+ = {} & \frac{\mathbf{M}}{\mathbf{\Lambda}^k}\left(\frac{\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)^\intercal \\ & + \left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\frac{\mathbf{W}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu}\left(\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)\right)\right\}(\mathbf{H}^k)^\intercal \\ & - \mathbf{W}^k\circ\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\left(\frac{(\mathbf{H}^k)^2\circ\left([\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+ + [\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)^\intercal\right\} \\ & + \frac{\mathbf{1}_{R\times T}\left(\mathbf{H}^k\circ\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)^\intercal}{((\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu)^2}\end{aligned} \tag{59}$$

$$\begin{aligned}[\nabla_{\mathbf{W}^k}\mathcal{E}]_- = {} & \frac{\mathbf{M}}{\mathbf{\Lambda}^k}\left(\frac{\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)^\intercal \\ & + \left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\circ\left(\frac{\mathbf{W}^k}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu}\left(\mathbf{H}^k\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\right)\right)\right\}(\mathbf{H}^k)^\intercal \\ & - \mathbf{W}^k\circ\left\{\frac{\mathbf{M}}{(\mathbf{\Lambda}^k)^2}\left(\frac{(\mathbf{H}^k)^2\circ\left([\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+ + [\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_-\right)}{(\mathbf{W}^k)^\intercal\mathbf{1}_{N\times T}+\mu}\right)^\intercal\right\} \\ & + \frac{\mathbf{1}_{R\times T}\left(\mathbf{H}^k\circ\left((\mathbf{W}^k)^\intercal\frac{\mathbf{M}}{\mathbf{\Lambda}^k}\right)\circ[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_+\right)^\intercal}{((\mathbf{W}^k)^\intercal\mathbf{1}_{N\times R}+\mu)^2}\end{aligned} \tag{60}$$

- From there, we can form multiplicative update equations for $\mathbf{W}^k$.

Note that only the $[\nabla_{\mathbf{H}^{k+1}}\mathcal{E}]_\pm$ for the layer $k+1$ need to be available when performing computations for $\mathbf{H}^k$ and $\mathbf{W}^k$, and they can thus be replaced iteratively as we descend through layers, limiting the memory requirements.