Design of a Non-negative Neural Network to Improve on NMF

Filip Wen-Fwu Tsai, Alireza M. Javid, and Saikat Chatterjee
Division of Information Science and Engineering
School of Electrical Engg and Computer Sc., KTH Royal Institute of Technology, Sweden
ftsai@kth.se, almj@kth.se, sach@kth.se

***Abstract*—For prediction of a non-negative target signal using a non-negative input, we design a feed-forward neural network to achieve a better performance than a non-negative matrix factorization (NMF) algorithm. We provide a mathematical relation between the neural network and NMF. The architecture of the neural network is built on a property of rectified-linear-unit (ReLU) activation function and a convex optimization layer-wise training approach. For an illustrative example, we choose a speech enhancement application where a clean speech spectrum is estimated from a noisy spectrum.**

***Index Terms*—Neural networks, non-negative matrix factorization, speech enhancement**

I. Introduction

A non-negative neural network (NNN) has non-negative input and output. Our goal is to design an NNN which provides better performance than non-negative matrix factorization (NMF). To design and train the NNN using a suitable cost function, we formulate our technical objectives and questions, as stated below.

1) Theoretical guarantee: The training cost of NNN is less than or equal to an NMF algorithm. How do we ensure that?
2) Avoiding over-fitting: We wish to achieve a better test performance than the NMF, even with a limited amount of training data. How do we avoid over-fitting using regularization?

We endeavor to address the objectives and questions by combining the following strategies: (a) We use the output of an NMF algorithm to form the input of NNN. This brings a clear mathematical relation between NNN and NMF. (b) We use an analytically motivated structure and regularization of weight matrices in the proposed NNN.

In pursuit of the objectives and questions, our *main contribution* is the theoretical development where we provide sufficient conditions for the number of layers in NNN, the number of nodes in a layer, and the lower bounds on the regularization parameters. The techniques for this theoretical development are motivated by the work of [1].

Literature review: We begin with a brief literature review on NMF [2]–[4]. Due to a good performance, NMF has been used for various applications, for example, astronomy [5], bioinformatics [6], and data mining [7]. In the gamut of machine learning, NMF interpretation is continued to be generalized and related among other to K-means clustering [8] and probabilistic graphical models [2]. There are also several studies on NMF for speech enhancement [9], [10]. We will explore speech enhancement as an illustrative application example in this article. Therefore, our literature review will tend to mention works in speech enhancement applications.

Neural networks receive significant attention today for speech enhancement. Examples of deep-neural-network-based speech enhancement methods in supervised learning strategy are studied in [11], [12]. The work [13] considered a perceptual objective measure in designing appropriate cost function for training a deep neural network. The works [14] and [15] have considered speech enhancement in spectrum domain using deep neural networks.

Relation with relevant literature: There are endeavors to bring together NMF and neural networks and explore their use in speech enhancement. A relevant work is [16] where iterative learning of NMF parameters is unfolded to bring an interpretation of layer-wise signal flow in a multi-layer neural network. In contrast, the work of [17] builds on a non-negative autoencoding development based on a single iteration of NMF parameter learning. The work of [18] performs encoding of vector estimation for target signal extraction where the estimation of encoding vector from noisy data is learned by a deep neural network. While these works successfully bring combinations of NMF and neural network, none of them provides sufficient mathematical justification for a better performance achievement vis-a-vis NMF. Our article is an endeavor to address this shortcoming.

Let us discuss the use of neural networks for end-to-end spectrum estimation in speech enhancement. The use of a deep neural network as a deep autoencoder for end-to-end spectrum estimation was explored in [19]. The algorithm of [19] estimates mel frequency cepstral coefficients of clean speech from noisy speech. A deep neural network with restricted Boltzman machine pretraining was explored in [20] for estimation of log power spectrum of clean speech from noisy speech. There are several other methods for the use of deep neural networks in speech enhancement in combination with some domain-specific aspects [21], for example, masking usage, feature usage, and perceptual objective measure [13]. An overview survey on the deep neural network for speech enhancement is recently published in [22]. The works mentioned above and several works cited in the overview article [22] are experimentally driven. We instead use a mathematically driven approach.

II. Non-negative Neural Network

Let $\mathbf{x} \in \mathbb{R}_+^P$ and $\mathbf{t} \in \mathbb{R}_+^P$ denote input and target output, respectively. Here, we use $\mathbb{R}_+$ to denote non-negative real line. We use $\mathbf{h}(\mathbf{x})$ to denote a known function of $\mathbf{x}$. Based on feed-forward neural network architectures, we construct the following construction for the proposed NNN comprising of L layers:

$$\tilde{\mathbf{t}} = \mathbf{W}_{L+1}\,\mathbf{g}_L(\mathbf{W}_L \ldots \mathbf{g}_2(\mathbf{W}_2\,\mathbf{g}_1(\mathbf{W}_1\,\mathbf{h}(\mathbf{x})))). \qquad (1)$$

For l'th layer, $\mathbf{W}_l$ denotes weight matrix and $\mathbf{g}_l(.)$ denotes the non-linear operation that uses scalar-wise activation functions. At the l'th layer the feature vector is

$$\mathbf{y}_l = \mathbf{g}_l(\mathbf{W}_l \ldots \mathbf{g}_2(\mathbf{W}_2\,\mathbf{g}_1(\mathbf{W}_1\,\mathbf{h}(\mathbf{x})))). \qquad (2)$$

The weight matrix $\mathbf{W}_l$ has $n_l \times n_{l-1}$ dimension. The dimension of the feature vector $\mathbf{y}_l$ is the number of nodes n_l in the l'th layer. Let us assume that the activation functions that we use in the NNN are fixed. Then, the NNN is represented as a function $\mathbf{f}$ as $\tilde{\mathbf{t}} = \mathbf{f}(\mathbf{h}(\mathbf{x}), \{\mathbf{W}_l\})$, in which $\{\mathbf{W}_l\}_{l=1}^L$ are the parameters to train.

A. Theoretical motivation

Let us assume to have a J-size training dataset $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathbf{t}^{(j)})\}_{j=1}^J$, where $(\mathbf{x}^{(j)}, \mathbf{t}^{(j)})$ is j'th observation-target data pair. In case of spectrum-based speech enhancement, $\mathbf{x}^{(j)}$ is the spectrum of noisy speech signal for j'th time window and $\mathbf{t}^{(j)}$ is the spectrum of clean speech signal in the corresponding window.

Using the training set $\mathcal{D}$, we use mean-square-error (MSE) cost for optimization due to analytical tractability. The MSE cost is

$$\begin{aligned} \mathcal{C}_{\text{NNN}} &= \textstyle\sum_{j=1}^J \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}^{(j)}\|_2^2 \\ &= \textstyle\sum_{j=1}^J \|\mathbf{t}^{(j)} - \mathbf{f}(\mathbf{h}(\mathbf{x}^{(j)}), \{\mathbf{W}_l\})\|_2^2. \end{aligned} \qquad (3)$$

The optimization problem for training with non-negative constraint is as follows:

$$\underset{\{\mathbf{W}_l\}}{\arg\min}\; \mathcal{C}_{\text{NNN}} \text{ s.t. } \left\{ \begin{array}{ll} \|\mathbf{W}_l\|_{\text{F}}^2 \leq \rho_l, & l = 1, 2, \ldots, L, \\ \tilde{\mathbf{t}}^{(j)} \geq \mathbf{0}, & j = 1, 2, \ldots, J. \end{array} \right. \qquad (4)$$

The overall optimization problem is non-convex due to non-linear activation functions in $\mathbf{g}_l(.)$. The constraint $\|\mathbf{W}_l\|_{\text{F}}^2 \leq \rho_l$ is a standard regularization that helps to avoid over-fitting when the amount of training data is limited. The constraint $\tilde{\mathbf{t}}^{(j)} \geq \mathbf{0}$ ensures non-negativity of output.

Next, let us assume that we have an NMF method that is trained using the (same) training dataset $\mathcal{D}$. We denote the output of NMF as $\tilde{\mathbf{t}}_{\text{NMF}}$ for the input $\mathbf{x}$; $\tilde{\mathbf{t}}_{\text{NMF}}$ is non-negative and a function of $\mathbf{x}$. Then, to fairly compare costs from different methods, the output of NMF is formulated with Euclidean distance [3]. Therefore, the optimal cost for the NMF of using the training dataset $\mathcal{D}$ is

$$C_{\text{NMF}} = \sum_{j=1}^J \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}_{\text{NMF}}^{(j)}\|_2^2, \qquad (5)$$

where $\tilde{\mathbf{t}}_{\text{NMF}}^{(j)}$ is the output of (optimal) NMF, in Euclidean distance, [3] for the data point $\mathbf{x}^{(j)}$ in the training dataset $\mathcal{D}$.
Requirements: Our main design criterion is how to construct NNN that follows the feed-forward structure (1) and addresses the following requirements:

1) The output $\tilde{\mathbf{t}}$ is non-negative for an arbitrary non-negative test data point $\mathbf{x}$.
2) The NNN training cost follows: $C_{\text{NNN}} \leq C_{\text{NMF}}$.
3) Analytical setting of regularization parameter ρ_l, $\forall l$.

Proposition 1 (Sufficient condition)**.** *We can construct NNN as a feed-forward neural network of L-layers to fulfill the requirements mentioned above, under the following conditions:*

1) Input $\mathbf{h}(\mathbf{x})$ is formed as $\mathbf{h}(\mathbf{x}) = [\tilde{\mathbf{t}}_{\text{NMF}}^\top\, \mathbf{x}^\top]^\top \in \mathbb{R}_+^{2P}$, where $\tilde{\mathbf{t}}_{\text{NMF}}$ is output of the NMF for input $\mathbf{x}$.
2) The NNN is comprised of at-least two layers, i.e., $L \geq 2$.
3) Activation function used in NNN is ReLU.
4) The number of nodes for the l'th layer follows:

$$\left\{ \begin{array}{ll} n_l \geq 4P, & l = 1, \\ n_l \geq 2P, & l = 2, 3, \ldots, L-1, \\ n_l = \;\; P, & l = L. \end{array} \right. \qquad (6)$$

5) Regularization parameters can be set analytically as follows:

$$\epsilon_l = \left\{ \begin{array}{ll} 4Pm_l, & l = 1, \\ 4P^2 m_{l-1}, & l = 2, \\ 2Pm_{l-1}, & l = 3, 4, \ldots, L, \end{array} \right. \qquad (7)$$

where $m_l = \frac{n_l}{2}$ and the relation between the regularization parameters ϵ_l and ρ_l is $\rho_l = \beta_l \epsilon_l$, where

$$\beta_l = \left\{ \begin{array}{ll} 2Pm_l, & l = 2, 3, \ldots, L-1, \\ 1, & \text{otherwise}. \end{array} \right. \qquad (8)$$

6) $\mathbf{W}_{L+1} = \mathbf{I}_P$ where $\mathbf{I}_P$ denotes identity matrix of size P.

Proof: The proposition is proved using examples of existence. We will design architecture of NNN in the next subsection to show existence examples.

Remark 1. *A minimal size NNN has $L = 2$ layers; the first layer has $n_1 = 4P$ nodes and the second layer has $n_2 = P$ nodes.*

B. Architecture design of NNN

In this subsection, we design the architecture of NNN and also prove the proposition 1. First, we design a two-layer NNN for simplicity; thereafter, we show that the design can be extended for an NNN comprised of more than two layers.

A two-layer NNN has the following signal flow relation:

$$\tilde{\mathbf{t}} = \mathbf{W}_3\,\mathbf{g}_2(\mathbf{W}_2\,\mathbf{g}_1(\mathbf{W}_1\,\mathbf{h}(\mathbf{x}))). \qquad (9)$$

We use ReLU activation function $\max(0, x)$ in nodes of NNN and we set $\mathbf{h}(\mathbf{x}) = [\tilde{\mathbf{t}}_{\text{NMF}}^\top\, \mathbf{x}^\top]^\top \in \mathbb{R}_+^{2P}$. To design the two-layer NNN and eventually prove the proposition 1, we require to use a property related to a single-layer feed-forward neural network (SLFN) that uses ReLU activation function. This property is called loss-less flow property (LFP), which is mentioned in [1]. The main point of LFP is that a signal can pass through a non-linear system exactly. Let us use the notation $\mathbf{h} \triangleq \mathbf{h}(\mathbf{x})$ for notational clarity. The LFP shows that for an SLFN of (9) we have

$$\mathbf{h} = \mathbf{R}_1^{\dagger} \mathbf{U}_{m_1} \mathbf{g}_1(\mathbf{V}_{m_1} \mathbf{R}_1 \mathbf{h}), \tag{10}$$

where $\mathbf{R}_1 \in \mathbb{R}^{m_1 \times 2P}$ is a full column-rank matrix (i.e. $m_1 \geq 2P$) and $\dagger$ denotes pseudoinverse such that $\mathbf{R}_1^{\dagger}\mathbf{R}_1 = \mathbf{I}_{2P}$. Here, $\mathbf{V}_{m_1}$ and $\mathbf{U}_{m_1}$ are two deterministic matrices as follows:

$$\mathbf{V}_{m_1} = \begin{bmatrix} \mathbf{I}_{m_1} \\ -\mathbf{I}_{m_1} \end{bmatrix} \text{ and } \mathbf{U}_{m_1} = [\mathbf{I}_{m_1} \quad -\mathbf{I}_{m_1}], \tag{11}$$

where $\mathbf{I}_{m_1}$ is an m_1-dimensional identity matrix. Note that $\mathbf{R}_1\mathbf{h}$ may not be a non-negative signal even though $\mathbf{h}$ is a non-negative signal. By using LFP, we now go back to the two-layer NNN construction as per (9).

To construct the two-layer NNN (9) that follows proposition 1, we use the following steps:

1) We use ReLU as activation function in all layers.
2) We set $\mathbf{h}(\mathbf{x}) = [\tilde{\mathbf{t}}_{\text{NMF}}^{\top} \, \mathbf{x}^{\top}]^{\top} \in \mathbb{R}_+^{2P}$.
3) We set $\mathbf{W}_1 = \mathbf{V}_{m_1}\mathbf{R}_1$, where $\mathbf{R}_1$ is a $m_1 \times 2P$-dimensional randomly chosen column-orthonormal matrix instance, that is $\mathbf{R}_1^{\top}\mathbf{R}_1 = \mathbf{I}_{2P}$. Once we choose the $\mathbf{R}_1$ matrix instance, which happens after attaining a satisfying prediction of the target (see the next point), we fix it for further use. Using equation (11), the regularization coefficient ϵ_1 for $\|\mathbf{W}_1\|_{\text{F}}^2 \leq \epsilon_1$ can be set as $\epsilon_1 = \|\mathbf{V}_{m_1}\|_{\text{F}}^2 \|\mathbf{R}_1\|_{\text{F}}^2 = 2m_1 \times 2P = 4Pm_1$.
4) In the first layer, the feature vector $\mathbf{y}_1 = \mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x}))$. We can use the feature vector $\mathbf{y}_1$ to predict the target $\mathbf{t}$ as $\tilde{\mathbf{t}}_1 = \mathbf{O}_1\mathbf{y}_1$. Then, for the j'th data point in the training dataset $\mathcal{D}$, we have
$$\tilde{\mathbf{t}}_1^{(j)} = \mathbf{O}_1\mathbf{y}_1^{(j)} = \mathbf{O}_1\mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x}^{(j)})). \tag{12}$$
For the two-layer NNN, we set $\mathbf{W}_2 = \mathbf{O}_1$, where $\mathbf{O}_1$ is a $P \times 2m_1$-dimensional matrix found by the following optimization problem:
$$\begin{aligned} &\underset{\mathbf{O}_1}{\arg\min} \; \textstyle\sum_{j=1}^{J} \|\mathbf{t}^{(j)} - \mathbf{O}_1\mathbf{y}_1^{(j)}\|_2^2 \\ &\text{s.t.} \begin{cases} \|\mathbf{W}_2\|_{\text{F}}^2 = \|\mathbf{O}_1\|_{\text{F}}^2 \leq \epsilon_2, \\ \tilde{\mathbf{t}}_1^{(j)} = \mathbf{O}_1\mathbf{y}_1^{(j)} \geq \mathbf{0}, \quad j = 1, 2, \ldots, J, \end{cases} \end{aligned} \tag{13}$$
where $\epsilon_2 = P\|\mathbf{R}_1^{\top}\|_{\text{F}}^2 \|\mathbf{U}_{m_1}\|_{\text{F}}^2 = 4P^2m_1$. The above optimization problem is convex for the parameter $\mathbf{O}_1$. We will explain the choice of ϵ_2 after the next point.
5) The feature vector $\mathbf{y}_2 = \mathbf{g}_2(\mathbf{W}_2 \, \mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x})))$, where $\mathbf{W}_2 = \mathbf{O}_1$ and the optimum $\mathbf{O}_1$ is found from (13). The number of nodes in the second layer is n_2, which is the dimension of $\mathbf{y}_2$. We set $n_2 = P$ and $\mathbf{W}_3 = \mathbf{I}_P$. This construction provides two aspects. First, the output of the two-layer NNN is guaranteed to be non-negative as $\tilde{\mathbf{t}} = \mathbf{W}_3 \, \mathbf{g}_2(\mathbf{W}_2 \, \mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x}))) \geq \mathbf{0}$. This address the first requirement. Second, it is also ensured that $\tilde{\mathbf{t}}^{(j)} = \tilde{\mathbf{t}}_1^{(j)}$ for $j = 1, 2, \ldots, J$ as we use ReLU and $\tilde{\mathbf{t}}_1^{(j)} \geq \mathbf{0}$. This in turn provides
$$\mathcal{C}_{\text{NNN}} = \sum_{j=1}^{J} \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}^{(j)}\|_2^2 = \sum_{j=1}^{J} \|\mathbf{t}^{(j)} - \mathbf{O}_1\mathbf{y}_1^{(j)}\|_2^2, \tag{14}$$
where the optimum $\mathbf{O}_1$ is found from (13).

While we have addressed the first requirement, we are yet to answer about the second and third requirements. The answer lies in the optimization problem (13). To prove $\mathcal{C}_{\text{NNN}} \leq \mathcal{C}_{\text{NMF}}$ we show an existence proof using LFP. Let us choose $\mathbf{O}_1$ as follows:

$$\mathbf{O}_1 = [\mathbf{I}_P \;\; \mathbf{0}_P]\mathbf{R}_1^{\top}\mathbf{U}_{m_1}, \tag{15}$$

where $\mathbf{0}_P$ is a P-dimensional square zero matrix. Using LFP, we have

$$\begin{aligned} \mathbf{O}_1\mathbf{y}_1 &= [\mathbf{I}_P \;\; \mathbf{0}_P]\mathbf{R}_1^{\top}\mathbf{U}_{m_1} \, \mathbf{g}_1(\mathbf{V}_{m_1}\mathbf{R}_1\mathbf{h}) \\ &= [\mathbf{I}_P \;\; \mathbf{0}_P]\mathbf{h} = [\mathbf{I}_P \;\; \mathbf{0}_P] \begin{bmatrix} \tilde{\mathbf{t}}_{\text{NMF}} \\ \mathbf{x} \end{bmatrix} \\ &= \tilde{\mathbf{t}}_{\text{NMF}}. \end{aligned} \tag{16}$$

Therefore, for the choice $\mathbf{O}_1 = [\mathbf{I}_P \;\; \mathbf{0}_P]\mathbf{R}_1^{\top}\mathbf{U}_{m_1}$, we have $\tilde{\mathbf{t}}_{\text{NNN}} = \tilde{\mathbf{t}}_{\text{NMF}}$, and hence, $\mathcal{C}_{\text{NNN}} = \mathcal{C}_{\text{NMF}}$. Observe that $\|\mathbf{O}_1\|_{\text{F}}^2 \leq \|[\mathbf{I}_P \;\; \mathbf{0}_P]\|_{\text{F}}^2 \|\mathbf{R}_1^{\top}\|_{\text{F}}^2 \|\mathbf{U}_{m_1}\|_{\text{F}}^2 = 4P^2m_1 = \epsilon_2$, and hence, the existence choice $\mathbf{O}_1 = [\mathbf{I}_P \;\; \mathbf{0}_P]\mathbf{R}_1^{\top}\mathbf{U}_{m_1}$ is in the feasible set of the optimization problem (13). Note also that the optimization problem (13) is convex for parameter $\mathbf{O}_1$ and hence, $\mathcal{C}_{\text{NNN}} \leq \mathcal{C}_{\text{NMF}}$ due to (14). This address the second and third requirements. Therefore, a minimal size NNN has the following architecture:

$$\tilde{\mathbf{t}} = \mathbf{g}_2(\mathbf{O}_1 \, \mathbf{g}_1(\mathbf{V}_{2P}\mathbf{R}_1 \, \mathbf{h}(\mathbf{x}))), \tag{17}$$

where $\mathbf{R}_1$ is an instance of $2P$-dimensional orthogonal random matrix and $\mathbf{O}_1$ is found from (13).

1) Extension to three layers and more: We can now design a three-layer NNN based on the optimized two-layer NNN. A three-layer NNN has the signal flow relation

$$\tilde{\mathbf{t}} = \mathbf{W}_4 \, \mathbf{g}_3(\mathbf{W}_3 \, \mathbf{g}_2(\mathbf{W}_2 \, \mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x})))). \tag{18}$$

The parameters are set as follows:

1) Previously, we showed $\mathbf{h}(\mathbf{x}) = [\tilde{\mathbf{t}}_{\text{NMF}}^{\top} \, \mathbf{x}^{\top}]^{\top}$ and $\mathbf{W}_1 = \mathbf{V}_{m_1}\mathbf{R}_1$ in the two-layer NNN. Therefore, we use them directly in the three-layer NNN.
2) We set $\mathbf{W}_2 = \mathbf{V}_{m_2}\mathbf{R}_2\mathbf{O}_1$, where $\mathbf{R}_2$ is an $m_2 \times P$-dimensional randomly chosen column-orthonormal matrix, that is $\mathbf{R}_2^{\top}\mathbf{R}_2 = \mathbf{I}_P$, and $\mathbf{O}_1$ is found in the two-layer NNN optimization using (13). Once we choose the $\mathbf{R}_2$ matrix instance, similar to $\mathbf{R}_1$, we fix it for further use. Note, $\epsilon_2 \neq \rho_2$ as $\|\mathbf{W}_2\|_{\text{F}}^2 = \|\mathbf{V}_{m_2}\mathbf{R}_2\mathbf{O}_1\|_{\text{F}}^2 \leq \|\mathbf{V}_{m_2}\|_{\text{F}}^2\|\mathbf{R}_2\|_{\text{F}}^2\epsilon_2 = 2Pm_2\epsilon_2 = \beta_2\epsilon_2 = \rho_2$.
3) The feature vector $\mathbf{y}_2 = \mathbf{g}_2(\mathbf{W}_2 \, \mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x})))$ corresponds to the second layer. We can use the feature vector $\mathbf{y}_2$ to predict the target $\mathbf{t}$ as $\tilde{\mathbf{t}}_2 = \mathbf{O}_2\mathbf{y}_2$. Then, for the j'th data point in the training dataset $\mathcal{D}$, we have
$$\tilde{\mathbf{t}}_2^{(j)} = \mathbf{O}_2\mathbf{y}_2^{(j)} = \mathbf{O}_2\mathbf{g}_2(\mathbf{W}_2 \, \mathbf{g}_1(\mathbf{W}_1 \, \mathbf{h}(\mathbf{x}^{(j)}))). \tag{19}$$
Now, for the three-layer NNN, we set $\mathbf{W}_3 = \mathbf{O}_2 \in \mathbb{R}^{P \times n_2}$, where $\mathbf{O}_2$ is found by the following optimization problem:
$$\begin{aligned} &\underset{\mathbf{O}_2}{\arg\min} \; \textstyle\sum_{j=1}^{J} \|\mathbf{t}^{(j)} - \mathbf{O}_2\mathbf{y}_2^{(j)}\|_2^2 \\ &\text{s.t.} \begin{cases} \|\mathbf{W}_3\|_{\text{F}}^2 = \|\mathbf{O}_2\|_{\text{F}}^2 \leq \epsilon_3, \\ \tilde{\mathbf{t}}_2^{(j)} = \mathbf{O}_2\mathbf{y}_2^{(j)} \geq \mathbf{0}, \quad j = 1, 2, \ldots, J, \end{cases} \end{aligned} \tag{20}$$

where $\epsilon_3 = \|\mathbf{R}_2^\top\|_\mathrm{F}^2 \|\mathbf{U}_{m_2}\|_\mathrm{F}^2 = 2Pm_2$.

4) $\mathbf{W}_4 = \mathbf{I}_P$.

Similarly, the training cost for three-layer NNN is less than or equal to the training cost for two-layer NNN. This becomes clear from setting the regularization coefficient $\epsilon_3 = \|\mathbf{R}_2^\top\|_\mathrm{F}^2 \|\mathbf{U}_{m_2}\|_\mathrm{F}^2 = 2Pm_2$. If we choose an existence of $\mathbf{O}_2 = \mathbf{R}_2^\top \mathbf{U}_{m_2}$ which is in the feasible set of the optimization problem (20), then, $\mathbf{O}_2\mathbf{y}_2 = \mathbf{R}_2^\top \mathbf{U}_{m_2} \mathbf{g}_2(\mathbf{V}_{m_2}\mathbf{R}_2\mathbf{O}_1\mathbf{y}_1) = \mathbf{O}_1\mathbf{y}_1$ using LFP. This leads to the result that the training cost for three-layer NNN is equal to the training cost for two-layer NNN. As the optimization problem (20) is convex for the $\mathbf{O}_2$ parameter, we guarantee that the training cost for three-layer NNN is less than or equal to the training cost for two-layer NNN. Naturally, the three-layer NNN can be further extended to more layers where layers are added one-by-one. For more than three layers, we consider similar line of argument and solve a similar optimization problem like (20), where we solve each new matrix $\mathbf{O}_{l-1}$ with its corresponding regularization coefficient $\epsilon_l = 2Pm_{l-1}$ for $l = 4, 5, \ldots, L$.

C. NNN as the post-processing for NMF

We now investigate the use of NNN as a post-processing method for NMF to achieve a better performance than the NMF. In the post-processing scheme, we use the output of NMF $\tilde{\mathbf{t}}_\mathrm{NMF} \triangleq \tilde{\mathbf{t}}_\mathrm{NMF}(\mathbf{x})$ as the input to the NNN post-processing as follows:

$$\tilde{\mathbf{t}} = \mathbf{W}_{L+1}\, \mathbf{g}_L(\mathbf{W}_L \ldots \mathbf{g}_2(\mathbf{W}_2\, \mathbf{g}_1(\mathbf{W}_1\, \tilde{\mathbf{t}}_\mathrm{NMF}))). \quad (21)$$

There is no direct use of $\mathbf{x}$ in NNN post-processing, instead the use of $\mathbf{x}$ comes via the use of $\tilde{\mathbf{t}}_\mathrm{NMF}$. Like the theoretical argument shown in the previous subsection for NNN, we can show that the training cost of NNN post-processor is less than or equal to the training cost of NMF. In this case, we can set the number of nodes per layer $n_l \geq P$ for $l = 1, 2, \ldots, L-1$ and the regularization coefficients $\epsilon_1 = 2Pm_1$ and $\epsilon_l = 2Pm_{l-1}$ for $l = 2, 3, \ldots, L-1$. Note, the l'th layer NNN post-processing has the same construction condition as the l'th layer NNN for $l \geq 3$.

III. Experiments

A. Generation of datasets for experiments

The clean speech signals used are from the TIMIT dataset. We took the speech samples with the New England dialect (DR1 folder) of the TIMIT dataset. The total duration of the clean speech signal is around 2 minutes and 40 seconds. This total dataset is divided into three parts: training dataset (1 min 20 sec), validation dataset (40 sec), and test dataset (40 sec). This clean speech is collected from 32 speakers, 16 males and 16 females. The speech samples between the training dataset, the validation dataset, and the test dataset have no overlap. Further, each utterance (sentence) is unique in this total clean speech dataset. Note that this amount of dataset is limited in comparison to the current practice of using a large amount of speech data in neural network training for speech enhancement application. For example, the work of Kolbæk et al. [13] considered 37 hours of training data and 4 hours of validation data. Therefore, a valid question is why do we use a data-limited training scenario for our speech enhancement application. There are many practical applications where data is not in abundance. It is important to evaluate a machine learning method for those data-limited application scenarios. Therefore, as a practical example, we deliberately investigate the scope of NNN for a data-limited speech enhancement scenario.

We generated training, validation, and test datasets comprised of observation-target data pair $(\mathbf{x}, \mathbf{t})$. Here, $\mathbf{x}$ is a noisy spectrum of a speech frame that we observed and $\mathbf{t}$ is the spectrum of the same frame in clean condition. Note, $\mathbf{t}$ is only known to the system during the training phase. We used an additive model for noisy speech generation (addition of speech and noise in waveform domain). Several types of noise are added in varying signal-to-noise ratio (SNR). We have used six noise types for our experiments. Five noises are from NOISEX-92 database, which is factory, babble, machine-gun, buccaneer, and F16, while the sixth noise source is from ITU-T recommendation P.501 [23], which is a cafeteria noise. Factory, babble, and machine-gun noises are used for training. As there are six noise types for testing, we had two experimental conditions: matched noise and mismatched noise. In the matched noise condition, the methods are trained and tested on the same noise types. In the mismatched condition, testing is performed on unseen noise types: buccaneer, F16, and cafeteria.

We used signals in 16 kHz sampling rate with appropriate up- and downsampling. From noisy signals, we computed spectrum vectors using a 1024-size short-time Fourier transform (STFT). The window length is 32 ms with 75% overlap in window shift. We used Hann windows to find the STFT-based spectrum vectors. The dimension of a spectrum vector is $P = 513$; the observation vector $\mathbf{x}$ and the target vector $\mathbf{t}$ are 513-dimensional vectors. The training dataset $\mathcal{D}$ of spectrum vectors had $J = 10000$ samples. The validation dataset and the testing dataset of spectrum vectors had 5000 samples each.

B. Experimental results

For our speech enhancement application, we used several performance measures: normalized-mean-square-error (NMSE) in dB scale, source-to-distortion-ratio (SDR) in dB scale, source-to-interference-ratio (SIR) in dB scale, source-to-artifacts-ratio (SAR) in dB scale [24], and perceptual-evaluation-of-speech-quality (PESQ) [25]. Among these five performance measures, NMSE is the measure where a low value is good. On the other hand, high values are good for the other four measures.

In our experiments[1], we work with NNNs of different sizes (layer numbers and number of nodes). Due to the brevity of space, we report the results for minimal size NNNs, mentioned in (17). Our first experiment considers the hypothesis that the NNN-based post-processing provides better performance

[1]MATLAB codes are available in https://sites.google.com/site/filiptsai/ and https://sites.google.com/site/saikatchatt/.

TABLE I: Performance of NMF, NNN post-processing, and NNN. The maximum standard deviation of a value is at most $\pm 1.1\%$, while the mean of all standard deviations is $\pm 0.42\%$.

SNR/	Matched noise condition					Mismatched noise condition				
[dB]	0	5	10	15	20	0	5	10	15	20
NMF										
NMSE	−4.76	−5.73	−6.08	−6.20	−6.22	−4.02	−5.45	−5.99	−6.17	−6.21
SDR	3.19	6.69	8.84	9.83	10.19	1.43	5.40	8.15	9.56	10.08
SIR	5.30	10.27	15.15	19.93	24.59	2.92	7.92	12.88	17.75	22.48
SAR	8.45	9.59	10.13	10.33	10.36	8.60	9.62	10.15	10.35	10.36
PESQ	2.08	2.35	2.63	2.87	3.07	1.76	2.08	2.38	2.66	2.91
NNN post-processing										
NMSE	−6.09	−7.78	−8.51	−8.80	−8.90	−5.02	−7.23	−8.31	−8.75	−8.89
SDR	4.75	7.81	9.54	10.35	10.68	4.25	7.58	9.49	10.37	10.70
SIR	8.05	12.91	17.47	21.84	26.09	24.82	28.36	32.02	35.77	39.32
SAR	8.13	9.64	10.38	10.70	10.82	4.31	7.62	9.52	10.39	10.71
PESQ	2.12	2.36	2.57	2.74	2.86	1.87	2.16	2.41	2.62	2.79
NNN										
NMSE	−8.46	−11.62	−13.78	−15.00	−15.56	−6.05	−10.00	−12.94	−14.66	−15.44
SDR	6.58	10.59	13.78	16.03	17.34	5.35	9.90	13.51	15.99	17.37
SIR	9.37	14.06	18.41	22.56	26.63	24.49	27.66	30.76	34.10	37.73
SAR	10.30	13.34	15.68	17.14	17.89	5.42	9.98	13.60	16.06	17.41
PESQ	2.28	2.58	2.84	3.08	3.26	1.98	2.31	2.62	2.90	3.14

than NMF on the test dataset. The NNN-based post-processing was mentioned in section II-C. The performance of NMF and NNN post-processing are shown in Table I. We observe that NNN post-processing leads to better performance than NMF, for most of the performance measures. It is encouraging to find that the performance for the mismatched noise condition is not significantly poorer than the matched noise condition. This result shows the efficiency of NNN in mismatched noise conditions as well as data-limited training. The success of NNN over NMF is attributed to the explicit regularization.

The second experiment considers the hypothesis that the NNN proposed in section II-B that considers input $\mathbf{h}(\mathbf{x}) = [\tilde{\mathbf{t}}_{\text{NMF}}^\top \ \mathbf{x}^\top]^\top$ is better than NNN-based post-processing mentioned in section II-C. We compare two-layer structures for a fair comparison. The performance of NNN post-processing and NNN are shown in Table I. We observe that the use of $\tilde{\mathbf{t}}_{\text{NMF}}$ and $\mathbf{x}$ together in NNN is typically better than the only use of $\tilde{\mathbf{t}}_{\text{NMF}}$ in the NNN-based post-processing.

IV. Conclusions

We conclude that it is possible to include non-negativity constraints for neural networks and relevant optimization problems are mathematically tractable. Under some technical conditions, the non-negative architectures can be proved to be better than non-negative matrix factorization (NMF), in Euclidean distance, in the sense of optimized training cost. The technical conditions help to decide the structure of the neural network with analytical regularization. For testing conditions, our experiments find that the proposed neural networks provide significant improvement for speech enhancement compared to NMF, even for a limited amount of training data.

References

[1] S. Chatterjee, A. M. Javid, M. Sadeghi, S. Kikuta, P. P. Mitra, and M. Skoglund, "Ssfn: Self size-estimating feed-forward network and low complexity design," *Arxiv*, 2019.

[2] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[3] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *In NIPS*, pp. 556–562, MIT Press, 2000.

[4] Y. Wang and Y. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 1336–1353, Jun 2013.

[5] M. R. Blanton and S. Roweis, "K-corrections and filter transformations in the ultraviolet, optical, and near-infrared," *The Astronomical Journal*, vol. 133, pp. 734–754, Jan 2007.

[6] H. Kim and H. Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, pp. 1495–1502, May 2007.

[7] C. Thurau, K. Kersting, and C. Bauckhage, "Convex non-negative matrix factorization in the wild," in *2009 Ninth IEEE International Conference on Data Mining*, pp. 523–532, Dec 2009.

[8] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the 2005 SIAM international conference on data mining*, pp. 606–610, SIAM, 2005.

[9] E. M. Grais and H. Erdogan, "Single channel speech music separation using nonnegative matrix factorization and spectral masks," in *2011 17th International Conference on Digital Signal Processing (DSP)*, pp. 1–6, Jul 2011.

[10] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, pp. 2140–2151, Oct 2013.

[11] Y. Xu, J. Du, L. Dai, and C. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 7–19, Jan 2015.

[12] E. W. Healy, S. E. Yoho, J. Chen, Y. Wang, and D. Wang, "An algorithm to increase speech intelligibility for hearing-impaired listeners in novel segments of the same noise type," *J. Acoust. Soc. Am.*, vol. 138, no. 3, pp. 1660–1669, 2015.

[13] M. Kolbæk, Z. Tan, and J. Jensen, "Monaural speech enhancement using deep neural networks by maximizing a short-time objective intelligibility measure," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5059–5063, Apr 2018.

[14] S. Rim Park and J. Lee, "A fully convolutional neural network for speech enhancement," in *INTERSPEECH*, pp. 1993–1997, 2017.

[15] T. Kounovsky and J. Malek, "Single channel speech enhancement using convolutional neural network," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pp. 1–5, May 2017.

[16] J. Le Roux, J. R. Hershey, and F. Weninger, "Deep nmf for speech separation," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 66–70, Apr 2015.

[17] P. Smaragdis and S. Venkataramani, "A neural network alternative to non-negative audio models," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 86–90, Mar 2017.

[18] T. G. Kang, K. Kwon, J. W. Shin, and N. S. Kim, "Nmf-based target source separation using deep neural network," *IEEE Signal Processing Letters*, vol. 22, pp. 229–233, Feb 2015.

[19] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising auto-encoder," *Proc. Interspeech*, pp. 436–440, Jan 2013.

[20] Y. Xu, J. Du, L. Dai, and C. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, pp. 65–68, Jan 2014.

[21] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1562–1566, May 2014.

[22] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, pp. 1702–1726, Oct 2018.

[23] ITU-T, "Test signals for use in telephonometry – Recommendation ITU-T P.501," Mar 2017.

[24] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1462–1469, Jul 2006.

[25] ITU-T, "Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs – Recommendation ITU-T P.862 (Amendment 2)," Nov 2005.