Music Signal Processing Using Vector Product Neural Networks

Zhe-Cheng Fan[*1], Tak-Shing T. Chan[†2], Yi-Hsuan Yang[‡2], and
Jyh-Shing R. Jang[§1]

[1]Dept. of Computer Science and Information Engineering, National Taiwan University, Taiwan
[2]Research Center for Information Technology Innovation, Academia Sinica, Taiwan

We propose a novel neural network model for music signal processing using vector product neurons and dimensionality transformations. Here, the inputs are first mapped from real values into three-dimensional vectors then fed into a three-dimensional vector product neural network where the inputs, outputs, and weights are all three-dimensional values. Next, the final outputs are mapped back to the reals. Two methods for dimensionality transformation are proposed, one via context windows and the other via spectral coloring. Experimental results on the iKala dataset for blind singing voice separation confirm the efficacy of our model.

Keywords: Deep learning, deep neural networks, vector product neural networks, dimensionality transformation, music source separation.

1 Introduction

In recent years, deep learning has become increasingly popular in the music information retrieval (MIR) community. For MIR problems requiring clip-level predictions or frame-by-frame predictions, such as genre classification, music segmentation, onset detection, chord recognition and vocal/non-vocal detection, many existing algorithms are based on convolutional neural networks (CNN) and recurrent neural networks (RNN). For audio regression problems which require an estimate for each time-frequency (t-f) unit over a spectrogram, such as source separation [Zhang and Wang, 2016], more algorithms are based on deep neural networks (DNN). This is because for such problems both input and output are matrices of the same size and therefore the neural network cannot involve operations that may reduce the spatial resolution. Existing DNN models for such problems usually consider each t-f unit as a real value and take

---

[*]lambert.fan@mirlab.org

[†]takshingchan@citi.sincia.edu.tw

[‡]yang@citi.sincia.edu.tw

[§]jang@mirlab.org

---

that as input for the network [Huang et al., 2014, Roma et al., 2016]. The question we want to address in this paper is whether we can achieve better results by applying some transformation methods to enrich the information for each t-f unit.

A widely used approach to enrich the information of each t-f unit is to add temporal context [Zhang and Wang, 2016]. For instance, in addition to the current frame, we add the previous-k and subsequent-k frames to compose a real-valued matrix and take it as the input of the neural network. But in this way, the interaction between different dimensions cannot be well modeled. To address this issue, we find it promising to consider a $(2k+1)$-dimensional neural network [Nitta, 2007]. As a first attempt, we implement this using vector product neural network (VPNN) [Nitta, 1993], a three-dimensional neural network that has only been tested on a simple XOR task in the literature. That is to say, k in this work is set to 1 (i.e. considering only the two neighboring frames). Each t-f unit is projected to a three-dimensional vector via this dimensionality transformation method. In VPNN, the input, output, weight and bias of each neuron are all three-dimensional vectors. While the main operation in DNN is matrix multiplication, it is the vector product in VPNN.

The goal of the paper is three-fold. First, we renovate VPNN with modern optimization techniques [Ruder, 2016] and test it on MIR problems instead of a simple XOR problem. Second, we test and compare the conventional context-enriched DNN structure using real values and the context-enriched VPNN structure for the specific task of blind singing voice separation from monaural recordings, which is a type of source separation problem. Third, we implement the idea of spectral coloring as another way to convert a real-valued matrix to a three-dimensional vector-valued matrix and evaluate this method again for blind singing voice separation. Our experiments confirm the efficacy of VPNN and both dimensionality transformation methods.

2 Vector Product Neural Network

In VPNN, the input data, weights, and biases are all three-dimensional vectors. Suppose there is an L-intermediate-layer VPNN, the input $\mathbf{z}_i^l$ of activation function in each neuron at the l-th layer is:

$$\mathbf{z}_i^l = \sum_{j=1}^{J} \mathbf{w}_{ij}^l \times \mathbf{a}_j^{l-1} + \mathbf{b}_i^l, \tag{1}$$

where $\times$ denotes vector product, $\mathbf{w}_{ij}^l$ stands for the weight connecting neurons j and i at l-th layer, $\mathbf{a}_j^{l-1}$ the input signal coming from neuron j at $(l-1)$-th layer, $\mathbf{b}_i^l$ the bias of neuron i at l-th layer. If $\mathbf{x}=[x_1 \ x_2 \ x_3]$ and $\mathbf{y}=[y_1 \ y_2 \ y_3]$, the result of vector product operation is $\mathbf{x} \times \mathbf{y} = [x_2y_3 - x_3y_2, \ x_3y_1 - x_1y_3, \ x_1y_2 - x_2y_1]$. As each element of the output vector receives contributions from all other dimensions, the vector product can capture all possible interactions among the three dimensions. This is not possible with real-valued neural network models

Note that we need to compute a lot of vector products between the layers in VPNN. In order to reduce training time, we propose to reformulate the vector product as matrix multiplication, which is more amenable to GPU acceleration. Suppose there are two vector-valued matrices, $\mathbf{P}$ and $\mathbf{Q}$. Their vector-valued matrix product $\otimes$ can be equivalently written as:

$$\mathbf{P} \otimes \mathbf{Q} = [\mathbf{p}_2\mathbf{q}_3 - \mathbf{p}_3\mathbf{q}_2, \mathbf{p}_3\mathbf{q}_1 - \mathbf{p}_1\mathbf{q}_3, \mathbf{p}_1\mathbf{q}_2 - \mathbf{p}_2\mathbf{q}_1], \tag{2}$$

where $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ are the matrices making up $\mathbf{P}$ and $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ are the matrices making up $\mathbf{Q}$. By applying Eq. (2), the output at hidden layer l can be defined as:

$$\mathbf{A}^l = \phi(\mathbf{W}^l \otimes \mathbf{A}^{l-1} + \mathbf{B}^l), \tag{3}$$

and the output $\mathbf{Y}$ of the VPNN, which is a vector-valued matrix, can be defined as:

$$\mathbf{Y} = \phi(\mathbf{W}^L ... \phi(\mathbf{W}^2 \otimes \phi(\mathbf{W}^1 \otimes \mathbf{A}^0 + \mathbf{B}^1) + \mathbf{B}^2) + ... \mathbf{B}^L), \tag{4}$$

where operator $\otimes$ denotes the vector-valued matrix product mentioned in Eq. (2). At the l-th layer, $\mathbf{A}^l$ is the hidden state, $\mathbf{W}^l$ is weight matrix, and $\mathbf{B}^l$ is bias matrix. All of them are vector-valued matrices. At the first layer, $\mathbf{A}^0$ is the input of the VPNN, consisting of vector-valued data from dimensionality transformation. The function ϕ is the sigmoid function. In order to achieve better performance, modern gradient optimization methods [Ruder, 2016] are implemented in our VPNN. Due to space constraints, we only report our results using Adam, a method for stochastic gradient descent.

3 Dimensionality Transformation and Objective Function

In this section, we elaborate two ideas for dimensionality transformation. One is based on adding temporal context described in Section 1. The other one is based on a novel technique called spectral coloring, which associates each t-f unit with a color in the RGB color space. Both ideas yield three-dimensional vectors as the input to VPNN.

3.1 Context-Windowed Transformation

To enrich the information for each t-f unit and improve the problem of interaction between different dimensions, we make current frame, previous and subsequent frames as a three-dimensional vector for each t-f unit. For ordinary NN, the input would be three real-valued matrices. For VPNN, the input is a three-dimensional matrix. In other words, the first dimension consists of previous frames, second dimension for current frames and third dimension for subsequent frames. Here we call this model as context-Window Vector Product Neural Network (WVPNN). After feeding the three-dimensional matrix into the VPNN, we get three-dimensional outputs and the second dimension is our predicted result.

3.2 Spectral Color Transformation

We can also map each t-f unit from a one-dimensional value into a three-dimensional vector, by using so-called spectral color transform, using for example the hot colormap in Matlab directly as a lookup table, where the forward and inverse maps are both computed by nearest neighbor searches. The hot colormap is associated with a resolution parameter which specifies the length of the colormap. As nearest neighbor interpolation will turn into piecewise linear interpolation when n tends to infinity, we can imitate the hot colormap with infinite resolution by the following piecewise linear functions instead:

$$\mathbf{v} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \max(\min(x/n, 1), 0) \\ \max(\min((x-n)/n, 1), 0) \\ \max(\min((x-2n)/(1-2n), 1), 0) \end{bmatrix}, \tag{5}$$

where $\mathbf{v}$ stands for the three-dimensional vector-valued vector, r, g and b the R, G, and B values respectively, x the magnitude of each t-f unit, and n a scalar to bias the generation of RGB values. We empirically set n to 0.0938 in this work. We call this model Colored Vector Product Neural Network (CVPNN). After feeding the RGB values into VPNN, we get RGB values as outputs. Each RGB value is then inversed-mapped to a magnitude at each t-f unit.

3.3 Target Function and Masking

During the training process of singing voice separation, given the predicted vocal spectra $\tilde{\mathbf{Z}}_1$ and predicted music spectra $\tilde{\mathbf{Z}}_2$, together with the original sources $\mathbf{Z}_1$ and $\mathbf{Z}_2$, the objective function J of the WVPNN and CVPNN can be defined as:

$$J = \|\tilde{\mathbf{Z}}_1 - \mathbf{Z}_1\|^2 + \|\tilde{\mathbf{Z}}_2 - \mathbf{Z}_2\|^2. \tag{6}$$

After getting the output from WVPNN and CVPNN, we can obtain the predicted spectra $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$. We smooth the results with a time-frequency masking technique called that soft time-frequency mask [Huang et al., 2015], and the magnitude spectra of the input frame can be transformed back to the time-domain by inverse STFT with the original phases.

4 Experiments

The proposed models are evaluated by singing voice separation experiments on the iKala dataset [Chan et al., 2015]. Only 252 song clips are released as a public set for evaluation. Due to the limitation of GPU memory, we partition the public set into 63 training clips and 189 testing clips. To reduce computation, all clips are downsampled to 16000 Hz. For each song clip, we use STFT to yield magnitude spectra with a 1024-point window and a 256-point hop size. The performance is measured in terms of source to distortion ratio (SDR), source to interferences ratio (SIR), and source to artifact ratio (SAR), as calculated by the blind source separation (BSS) Eval toolbox v3.0 [Vincent et al., 2006]. The overall performance is reported via global NSDR (GNSDR), global SIR (GSIR), and global SAR (GSAR), which are the weighted means of the measures over all clips with a weighting proportional to the length of the clips. Higher numbers mean better performances.

In order to compare the performance of ordinary DNN and CVPNN, we construct two networks that both consist of 3 hidden layers and 512 neurons in each hidden layer, denoted as CVPNN and DNN$_1$, respectively, using Eq. (6) as the target function. The dimensionality of each t-f unit is 1 for ordinary DNN and 3 for CVPNN. A network structure similiar to this DNN$_1$ was used in [Roma et al., 2016]. As shown in Table 1, CVPNN performs better than DNN$_1$ in both GNSDR and GSIR. As CVPNN has three times of NN parameters (i.e. number of weights and bias) as compared with DNN$_1$, for fair comparison we further construct an ordinary DNN comprising 3 hidden layers and 1536 neurons in each hidden layer, denoted as DNN$_2$, so that both CVPNN and ordinary DNN have the same number of parameters. Table 1 shows that CVPNN still performs better. Besides, we also construct two architectures which have the same number of parameters and t-f units composed of context window size of 3 frames, denoted as WVPNN and DNN$_3$ respectively. Both of them are composed of 3 hidden layers. The number of neurons in each hidden layer is 512 for WVPNN, and 1536 for DNN. The difference of these two is the combination of input frames. The input frames is constructed

Table 1: Comparison of ordinary DNN, WVPNN and CVPNN.

Neural Networks.	Arch.	Context Window Size	GNSDR	GSIR	GSAR
DNN_1	512x3	1	8.16	11.88	12.11
DNN_2	1536x3	1	8.37	12.64	11.82
CVPNN	512x3	1	8.87	13.38	11.37
DNN_3	1536x3	3	8.85	12.59	12.52
WVPNN	512x3	3	9.01	13.82	11.97

as a three-dimensional vector-valued matrix for WVPNN and a two-dimensional valued matrix for DNN_3. Results in Table 1 show that WVPNN performs better than ordinary DNN_3.

5 Conclusion and Future Work

In this paper, we propose WVPNN and CVPNN for monaural singing voice separation, using two dimensionality transformation methods. We also propose modern gradient optimization methods on VPNN to attain better performance. Our evaluation shows that both proposed models are better than traditional DNN, with 0.16–0.85 dB GNSDR gain and 1.23–1.94 dB GSIR gain. Future work is to extend these models to CNN and RNN and apply them to other MIR problems, such as genre classification and music segmentation.

References

Tak-Shing Chan et al. Vocal activity informed singing voice separation with the iKala dataset. In *Proc. ICASSP*, pages 718–722, 2015.

Po-Sen Huang et al. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *Proc. ISMIR*, pages 477–482, 2014.

Po-Sen Huang et al. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE Trans. Audio, Speech and Language Processing*, 23(12): 2136–2147, 2015.

Tohru Nitta. A backpropagation algorithm for neural networks based an 3D vector product. In *Proc. IJCNN*, pages 589–592, 1993.

Tohru Nitta. N-dimensional vector neuron. In *Proc. IJCAI*, pages 2–7, 2007.

Gerard Roma et al. Singing voice separation using deep neural networks and f0 estimation. In *http://www.music-ir.org/mirex/abstracts/2016/RSGP1.pdf*, 2016.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Emmanuel Vincent et al. Performance measurement in blind audio source separation. *IEEE Trans. Audio, Speech and Language Processing*, 14(4):1462–1469, July 2006.

Xiao-Lei Zhang and DeLiang Wang. A deep ensemble learning method for monaural speech separation. *IEEE Trans. Audio, Speech and Language Processing*, 24(5):967–977, 2016.