



SEPTEMBER 15 2023

Efficient design of neural networks for the classification of acoustic spectra

Vlad S. Paul  ; Philip A. Nelson 



JASA Express Lett. 3, 094802 (2023)

<https://doi.org/10.1121/10.0020990>



View
Online



Export
Citation

CrossMark

Related Content

Small angle scattering of diblock copolymers profiled by machine learning

J. Chem. Phys. (April 2022)

Application of wavelet transformation and adaptive neighborhood based modified backpropagation (ANMBP) for classification of brain cancer

AIP Conference Proceedings (August 2017)



Advance your science and career as a member of the
Acoustical Society of America

[LEARN MORE](#)

Efficient design of neural networks for the classification of acoustic spectra

Vlad S. Paul  and Philip A. Nelson 

Institute of Sound and Vibration Research, University of Southampton, Southampton, SO17 1BJ, United Kingdom

vsp1g18@soton.ac.uk, p.a.nelson@soton.ac.uk

Abstract: A previous paper by Paul and Nelson [(2021). *J. Acoust. Soc. Am.* **149**(6), 4119–4133] presented the application of the singular value decomposition (SVD) to the weight matrices of multilayer perceptron (MLP) networks as a pruning strategy to remove weight parameters. This work builds on the previous technique and presents a method of reducing the size of a hidden layer by applying a similar SVD algorithm. Results show that by reducing the neurons in the hidden layer, a significant amount of training time is saved compared to the algorithm presented in the previous paper while no or little accuracy is being lost compared to the original MLP model. © 2023 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

[Editor: David C Swanson]

<https://doi.org/10.1121/10.0020990>

Received: 24 February 2023 **Accepted:** 28 August 2023 **Published Online:** 15 September 2023

1. Introduction

The trend of applying machine learning techniques to solve problems in acoustics has increased rapidly in recent years. A number of different applications are discussed in Bianco *et al.* (2019), where the authors present a detailed review of machine learning models and their use in acoustics. Similarly, Purwins *et al.* (2019) presents a summary of deep learning approaches applied to audio signal processing. More recently, Grumiaux *et al.* (2021) presented a review of deep learning approaches focused for audio source localisation applications. Drawing from these reviews, among other sources, it becomes clear that there has been a consistent trend of building network models of increasing complexity to handle more difficult tasks. Notably, as the network model size escalates, it inherently demands a higher computational power, thereby leading to longer training times.

The so-called “pruning” of neural networks is a very active research topic (Blalock *et al.*, 2020; Choudhary *et al.*, 2020) because the use of computational power has become an issue in the use of high dimensional networks with large numbers of neurons. It has been presented in Denil *et al.* (2013) that a large number of weight parameters in the usual network architectures can be observed as redundant, suggesting that network layers are usually over-parameterized. This redundancy implies that numerous weight values encode similar or indistinguishable patterns, thereby offering opportunities for optimization and compression of the network without compromising its overall performance. There are various proposed techniques to implement network pruning (e.g., Augasta and Kathirvalavakumar, 2013; Blalock *et al.*, 2020; Suzuki *et al.*, 2018), and some of these methods are based on a low-rank approximation of the weight matrices, where, for example, a weight matrix is approximated by two or more low-rank matrices (e.g., Jaderberg *et al.*, 2014; Shmalo *et al.*, 2023). In Yang *et al.* (2020), for instance, the authors propose the use of the singular value decomposition (SVD) on the weight matrix at the beginning of the training, decomposing the matrix into two smaller matrices and performing a full-rank SVD training before removing small singular values on the trained model. Similarly, Shmalo *et al.* (2023) use the same low-rank approximation of the weight matrix using two smaller matrices with the aim of reducing the overfitting of networks and improving the accuracy. In acoustics, some previous work (Cai *et al.*, 2014; Xue *et al.*, 2013) used SVD-based approaches to reduce the training parameters of feed-forward networks. In Xue *et al.* (2013), the authors computed two small matrices from the SVD matrices of the weights at one point during training and showed that for a speech recognition task, they can remove a large proportion of the network without losing any accuracy. In Cai *et al.* (2014), the authors applied a similar technique to a speech recognition task, arguing that the use of the SVD directly on the randomly initialised weight matrices is not beneficial, such that they apply the pruning technique to a model that trained for a couple of iterations. More recently, Singh and Plumbley (2022) applied a different pruning technique (not based on the SVD) to a convolutional neural network (CNN) for acoustic scene classification. The authors remove convolutional filters by assuming that similar filters produce similar responses and are, therefore, redundant for the overall training.

The technique presented previously by Paul and Nelson (2021) is based on decomposing the weight matrices into their component SVD matrices. This differs from the methods used previously in Cai *et al.* (2014) and Xue *et al.* (2013) in that it enables the user to reduce training parameters in an iterative way during training and uses all three component matrices of the SVD during training with the backpropagation. It has been shown in Paul and Nelson (2021) that

by using the SVD approach, significant training time can be saved (up to 2/3) with little or no loss in generalization accuracy. This is achieved by removing training parameters between the input and hidden layer iteratively without changing the shape of the network. In this paper, the authors present an extension of the SVD technique to decrease the size of the hidden layer. The SVD approach applied to network models can be traced back to an early paper by [Psichogios and Ungar \(1994\)](#), where the authors use the SVD to reduce overfitting and improve the generalization error. These authors removed the redundant singular values and corresponding hidden layer nodes after the training was completed, which is in contrast to the current approach.

In the work presented here, the focus is on the pruning capabilities of the application of the SVD by iteratively removing neurons in the hidden layer during training such that the network size can be gradually reduced over time. The technique is applied here to multilayer perceptrons (MLPs) with only one hidden layer, but the method can be easily extended to MLPs with multiple layers. Compared to other previously proposed pruning techniques that use the SVD, the work presented here does not require a full training of the model before discarding training weights (e.g., [Psichogios and Ungar, 1994](#); [Yang et al., 2020](#)), it is discarding neurons in the hidden layer progressively, not just one time at the beginning or at some point during the training (e.g., [Cai et al., 2014](#); [Xue et al., 2013](#)) and, overall, it is able to adapt to the training data and task to be solved as will be discussed later. In addition, the authors believe that use of the SVD approximation on the matrix between the input and hidden layer could potentially offer useful information about what audio content from the training samples (for example, frequency content) is found by the network model to be more important during the training. This way, one could potentially better understand the patterns found by the MLP in the input data while discarding redundant information.

2. Reduction of network dimensions using the SVD

Beginning with a simple MLP network with one hidden layer, the forward and backward propagations can be written using vector and matrix notation as discussed by [Paul and Nelson \(2021\)](#). The matrices $\mathbf{W}^{(2)}$, $\mathbf{W}^{(1)}$ denote the matrices of weights relating the input to the hidden layer and the hidden layer to the output layer, respectively, the vectors $\mathbf{b}^{(2)}$, $\mathbf{b}^{(1)}$ are the vectors of bias weights at each layer and $\mathbf{a}^{(2)}$, $\mathbf{z}^{(2)}$, respectively, denote the inputs and outputs of the hidden layer. Note that the layers are counted from the output layer backward and, thus, $\mathbf{a}^{(1)}$ denotes the input into the output layer, and $\mathbf{z}^{(1)} = \hat{\mathbf{y}}$ is the output of the network. The network is trained by using the method of steepest descent (or one of its variants) such that the weight matrices are updated at every iteration. The general equation for the steepest descent is given by $\mathbf{W}(\tau + 1) = \mathbf{W}(\tau) - \eta \partial L / \partial \mathbf{W}(\tau)$, where L denotes the loss function to be minimised, τ denotes the index defining the update of the matrices during backpropagation, and \mathbf{W} is replaced by either $\mathbf{W}^{(1)}$ or $\mathbf{W}^{(2)}$ when training the above model. The equations for the relevant matrix derivatives are given in the previous paper ([Paul and Nelson, 2021](#)).

The approach taken in the previous paper to speed up the learning rate of the network was to apply the SVD to the matrix of weights $\mathbf{W}^{(2)}$ and, therefore, the forward propagation between these two layers becomes

$$\mathbf{a}^{(2)} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)\mathbf{x} + \mathbf{b}^{(2)}, \tag{1}$$

where \mathbf{U} is the matrix of left singular vectors, $\mathbf{\Sigma}$ is the diagonal matrix of singular values, and \mathbf{V} is the matrix of right singular vectors. The new MLP-SVD architecture is depicted in [Paul and Nelson \(2021\)](#). As described there, one can derive a backpropagation algorithm to update these three newly created matrices during training. The speed with which the network can be trained is improved by discarding a number of the smallest singular values at a series of points during the training process. The dimensions of all three component matrices are reduced at each of these “discarding points” while giving enough time for the new network architecture to adapt. Thus, by removing small singular values, one can also remove columns of \mathbf{U} and rows of \mathbf{V}^T at each discarding point. The network, hence, ends up training far fewer matrix parameters than in the original MLP model and can achieve the same accuracy as the original MLP model. The removal of the singular values is determined by their magnitude relative to the largest singular value based on a threshold. It has also been found that the points during training at which singular values are discarded is best accomplished by spacing the discarding points logarithmically with more frequent discarding points toward the start of training with reducing frequency as training progresses. The number of iterations that determine the discarding points can be computed from

$$d(n) = d(n - 1) \cdot 10^{(b-a)/(N-1)}, \tag{2}$$

where N is the total number of discarding points, n is the iteration index, and a and b define, respectively, the lower and higher bounds of the sequence of discarding points. If the lower bound is defined to be, for example, three, the value of a in Eq. (2) would be $a = \log_{10}(3)$. The value of the index, τ , at which discarding takes place during training is given by the nearest integer value of $d(n)$.

The aim of the new work described here is to use the same approach to singular value discarding as described above but also to reduce the number of neurons in the hidden layer at each discarding point. To do this, a couple of additional steps have to be introduced during the forward propagation. First, assume that before any discarding of singular values is undertaken at the n th discarding point, the SVD of weight matrix, $\mathbf{W}_{n-1}^{(2)}$, shows that Eq. (1) can be written as

$$\mathbf{a}_{n-1}^{(2)} = (\mathbf{U}_{n-1}\boldsymbol{\Sigma}_{n-1}\mathbf{V}_{n-1}^T)\mathbf{x} + \mathbf{b}_{n-1}^{(2)}. \quad (3)$$

Now, note that after the small singular values have been removed, the matrices in the SVD are replaced by \mathbf{U}_n , $\boldsymbol{\Sigma}_n$, and \mathbf{V}_n . Note that, in particular, a corresponding number of columns of \mathbf{U}_n can be removed, thus, reducing the dimension of this matrix. The first step is then to pre-multiply Eq. (3) by \mathbf{U}_n^T , which gives

$$\mathbf{U}_n^T\mathbf{a}_{n-1}^{(2)} = (\mathbf{U}_n^T\mathbf{U}_n\boldsymbol{\Sigma}_n\mathbf{V}_n^T)\mathbf{x} + \mathbf{U}_n^T\mathbf{b}_{n-1}^{(2)}. \quad (4)$$

The second step is to multiply the remaining two SVD matrices to form a new weight matrix such that $\mathbf{W}_n^{(2)} = \boldsymbol{\Sigma}_n\mathbf{V}_n^T$, where the dimensions of $\mathbf{W}_n^{(2)}$ depend on the number of singular values discarded. Following this, as $\mathbf{U}_n^T\mathbf{U}_n = \mathbf{I}$, the identity matrix, the new forward propagation of the MLP-SVD model is given by

$$\mathbf{a}_n^{(2)} = \mathbf{W}_n^{(2)}\mathbf{x} + \mathbf{b}_n^{(2)}, \quad (5)$$

$$\mathbf{z}_n^{(2)} = h(\mathbf{a}_n^{(2)}), \quad (6)$$

$$\mathbf{a}^{(1)} = \mathbf{W}_n^{(1)}\mathbf{z}_n^{(2)} + \mathbf{b}^{(1)}, \quad (7)$$

$$\mathbf{z}^{(1)} = h(\mathbf{a}^{(1)}) = \hat{\mathbf{y}}, \quad (8)$$

where the new vector of inputs to the hidden layer is $\mathbf{a}_n^{(2)} = \mathbf{U}_n^T\mathbf{a}_{n-1}^{(2)}$, and the vector of hidden layer bias terms is given by $\mathbf{b}_n^{(2)} = \mathbf{U}_n^T\mathbf{b}_{n-1}^{(2)}$. It can be observed that the new dimension of $\mathbf{a}_n^{(2)}$ depends on how many singular values are removed from the original matrix of singular values, $\boldsymbol{\Sigma}_{n-1}$, because the same number of columns is removed from \mathbf{U}_{n-1} . Note also that the new bias term, $\mathbf{b}_n^{(2)}$, will also have the same dimensions as $\mathbf{a}_n^{(2)}$. Compared to the previous approach (Paul and Nelson, 2021), in this case, the hidden layer and its bias term are changed at every discarding point and their dimensions are reduced. Also, we only use $\mathbf{W}_n^{(2)}$ as a single weight matrix between the input and hidden layer during the forward propagation and, therefore, only a single set of gradients needs to be computed instead of three. If the dimensions of $\mathbf{z}_n^{(2)} = h(\mathbf{a}_n^{(2)})$ are reduced every time singular values are discarded, it follows that $\mathbf{W}_n^{(1)}$ also must adapt its size to the dimensions of $\mathbf{z}_n^{(2)}$. This is accomplished by removing as many columns of $\mathbf{W}_{n-1}^{(1)}$ as singular values are removed at each stage. Equations (5)–(8) describe a single iteration of the discarding process, but it is clear that as training continues, at the appropriately chosen discarding points, the process can be repeated and another SVD of the weight matrix can be used to discard more singular values of $\mathbf{W}_n^{(2)}$, leading to further reductions in the dimensions of the hidden layer.

3. Illustrative application of the method

The same problem of acoustic spectral classification that was used in the previous paper (Paul and Nelson, 2021) can be employed to illustrate the application of this method. The training data were synthesised from white noise signals that were passed through bandpass filters having different centre frequencies and bandwidths. Different datasets were generated by changing three main parameters. First, the bandwidth of the bandpass filter was changed between 10 and 100 Hz, then Δf was changed from 30 to 60 Hz, and the number of output classes was changed between three and nine classes. Using a sampling frequency of 16 kHz and a fast Fourier transform (FFT) length of 256 samples when transforming the signals into the frequency-domain, the spectral resolution available is around 62 Hz, which suggests that a difference between centre frequencies of $\Delta f = 30$ Hz might be more difficult to observe compared to $\Delta f = 60$ Hz. When each parameter was changed, all the others were kept the same such that there were a total of 12 comparisons between the 3 different network implementations. The training database was generated with 1000 signals from each class of bandpass filtered white noise transformed into the frequency-domain, and both training and validation datasets were shuffled before starting the training. All three networks had a single hidden layer, and the number of output nodes in the network corresponded to the number of classes of bandpass filtered white noise. The number of neurons in the hidden layer had different values to test the robustness of the time saving algorithms. The input layer contained 129 neurons corresponding to the magnitude of the FFT of the signals. The learning rate used for the following simulations was 0.001, and the networks were trained using the Adam optimizer, which is a variant of the method of steepest descent.

To solve such a classification task, the MLP-SVD network was trained to minimise the total cross-entropy. The cost function used in the output layer is the softmax function, which scales the input between zero and one and is usually used together with the cross-entropy loss function. Using the slightly new forward propagation equations, gradients of the error with respect to the two set of weights $\mathbf{W}_n^{(2)}$, $\mathbf{W}_n^{(1)}$ can be derived in the same way as described by Paul and Nelson (2021) for the regular MLP because the SVD matrices are only used for removing training parameters and building new weight matrices. It is important to note that all variables with a subscript, n , in the forward propagation will change dimensions at every discarding point. Following this, due to the discarding of singular values, the gradient equations, $\partial L/\partial \mathbf{W}_n^{(1)}$, $\partial L/\partial \mathbf{W}_n^{(2)}$, and $\partial L/\partial \mathbf{b}_n^{(2)}$, will have fewer parameters after every discarding point, n . The gradient equations used to update the weights and biases are derived in detail in the previous paper and will only be given here in their final form. The update equations for the first set of weights and biases are given by

Algorithm 1. MLP-SVD approach to reduce the number of neurons in the hidden layer.

```

1: Initialize MLP network data
2: Define: lowBound, highBound, nrPoints
3: discPoints ← logVec(lowBound, highBound, nrPoints) {Define discarding points vector}
4: for  $\tau \leftarrow 1$ , iterations do
5:   if  $\tau < \textit{lowBound}$  then
6:      $[\mathbf{W}_\tau^{(1)}, \mathbf{b}_\tau^{(1)}, \mathbf{W}_\tau^{(2)}, \mathbf{b}_\tau^{(2)}] = \textit{trainMLP}(\mathbf{W}_{\tau-1}^{(1)}, \mathbf{b}_{\tau-1}^{(1)}, \mathbf{W}_{\tau-1}^{(2)}, \mathbf{b}_{\tau-1}^{(2)})$  {Train regular MLP for first iterations}
7:   else
8:     if  $\tau = \textit{lowBound}$  OR  $\tau = \textit{any}(\textit{logVec})$  then
9:        $[\mathbf{U}_{n-1}, \mathbf{\Sigma}_{n-1}, \mathbf{V}_{n-1}^T] = \textit{svd}(\mathbf{W}_{n-1}^{(2)})$  {Matrix of weights  $\mathbf{W}_{n-1}^{(2)} = \mathbf{W}_{\tau-1}^{(2)}$ }
10:       $[\mathbf{U}_n, \mathbf{\Sigma}_n, \mathbf{V}_n^T] = \textit{removeVal}(\mathbf{U}_{n-1}, \mathbf{\Sigma}_{n-1}, \mathbf{V}_{n-1}^T)$  {Remove singular values}
11:       $\mathbf{W}_n^{(2)} = \mathbf{\Sigma}_n \mathbf{V}_n^T$ 
12:       $[\mathbf{W}_\tau^{(1)}, \mathbf{b}_\tau^{(1)}, \mathbf{W}_\tau^{(2)}, \mathbf{b}_\tau^{(2)}] = \textit{trainMLP}(\mathbf{W}_{\tau-1}^{(1)}, \mathbf{b}_{\tau-1}^{(1)}, \mathbf{U}_n, \mathbf{W}_n^{(2)}, \mathbf{b}_{\tau-1}^{(2)})$  {Train MLP with new  $\mathbf{W}_n^{(2)}$ }
13:    else
14:       $[\mathbf{W}_\tau^{(1)}, \mathbf{b}_\tau^{(1)}, \mathbf{W}_\tau^{(2)}, \mathbf{b}_\tau^{(2)}] = \textit{trainMLP}(\mathbf{W}_{\tau-1}^{(1)}, \mathbf{b}_{\tau-1}^{(1)}, \mathbf{W}_{\tau-1}^{(2)}, \mathbf{b}_{\tau-1}^{(2)})$  {If no discarding point, continue to train}
15:    end if
16:  end if
17: end for

```

$$\frac{\partial L}{\partial \mathbf{W}_n^{(1)}} = \mathbf{H}^{(1)T} \mathbf{d} \mathbf{z}_n^{(2)T}, \tag{9}$$

$$\frac{\partial L}{\partial \mathbf{b}^{(1)}} = \mathbf{H}^{(1)T} \mathbf{d}, \tag{10}$$

where $\mathbf{d}^T = [d_1, d_2, \dots, d_K]$ and $d_k = \partial L / \partial z_k^{(1)} = -y_k / z_k^{(1)}$. The matrix, $\mathbf{H}^{(1)T}$, corresponds to the derivative of the softmax function with respect to its input, $\partial \mathbf{z}^{(1)} / \partial \mathbf{a}^{(1)}$, and its computation is detailed by Paul and Nelson (2021). Following this, the gradient equations with respect to the second set of weights and biases are given by

$$\frac{\partial L}{\partial \mathbf{W}_n^{(2)}} = \mathbf{H}_n^{(2)T} \mathbf{W}_n^{(1)T} \mathbf{H}^{(1)T} \mathbf{d} \mathbf{x}^T, \tag{11}$$

$$\frac{\partial L}{\partial \mathbf{b}_n^{(2)}} = \mathbf{H}_n^{(2)T} \mathbf{W}_n^{(1)T} \mathbf{H}^{(1)T} \mathbf{d}, \tag{12}$$

where, similar to Eqs. (9) and (10), $\mathbf{H}_n^{(2)T}$ denotes the matrix of derivatives $\partial \mathbf{z}_n^{(2)} / \partial \mathbf{a}^{(2)}$, which usually turns out to be a diagonal matrix as described in Paul and Nelson (2021). To make the implementation steps clear, the pseudocode is shown below. The network starts training using the regular MLP until an iteration number denoted as *lower bound*, which, for this case, was equal to $\tau = 3$. Next, the SVD is applied on the weight matrix and the network starts to train using the new forward and backward propagations. The discarding points are calculated using a logarithmic approach as described above and in Paul and Nelson (2021) based on a lower bound [$a = \log_{10}(3)$], a higher bound corresponding to 1/3 of the total number of iterations [$b = \log_{10}(66)$], and the number of desired discarding points, which, for this case, was equal to $N = 3$. At each discarding point, the SVD is applied again on the weight matrix, and the network structure is changed depending on the number of discarded singular values.

4. Results

The algorithm presented above is compared in terms of time reduction to the technique presented in Paul and Nelson (2021) and the regular MLP using the same model problem of classifying closely related spectra. The time reduction will be expressed directly as time needed to finish training but also in terms of FLOPs (floating point operations) performed when the network model is implemented in its reduced form. It should be noted that the FLOP value is a rough estimate of the number of multiplications and additions needed to classify one test sample. The comparison between the three different networks will be made based on the validation accuracy at the end of training (how well the networks can generalize), but is also based on the training time being saved using the SVD methods. For simplicity, the technique presented in the previous paper will be denoted as MLP-SVD1 and the new technique as MLP-SVD2. The results shown below are averaged over ten trials.

Figures 1(a) and 1(b) show a comparison between the three techniques for $\Delta f = 30$ Hz and $\Delta f = 60$ Hz using three and nine output classes. It can be observed in both plots that all three network architectures have a similar performance in terms of accuracy. As expected, all networks perform worse if $\Delta f = 30$ Hz [Fig. 1(a)], and if nine output classes are used, the networks achieve a lower accuracy than if only three output classes are used. For both SVD

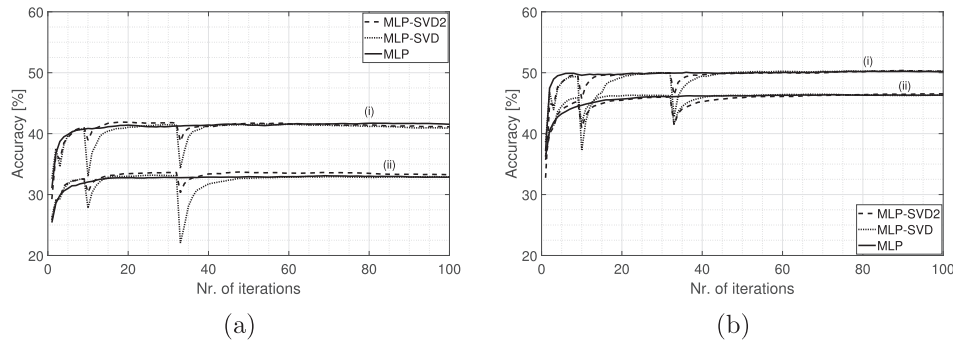


Fig. 1. Comparison of accuracy performances between the three techniques for (a) $\Delta f = 30$ Hz using (i) three output classes and (ii) nine output classes and (b) $\Delta f = 60$ Hz using (i) three output classes and (ii) nine output classes using a filter bandwidth of 100 Hz. All 3 networks had 20 neurons in the hidden layer.

approaches, the drop in accuracy can be clearly observed whenever singular values are discarded. A similar trend can be observed when increasing the hidden layer to 100 neurons. The performance of the networks remains the same, however, the training time increases and, therefore, the MLP-SVD2 approach saves more time. Table 1 shows a comparison between the different techniques for two hidden layer dimensions (20 and 100 neurons) using various dataset parameters.

It can be observed that for all different datasets, the three networks perform very similarly, on average, in terms of accuracy. When it comes to training time, the newly proposed technique (MLP-SVD2) needs the least training time in all cases. Especially when the hidden layer has 100 neurons at the start of training, both MLP-SVD techniques save more than 50% of the total training time. The MLP-SVD2 approach trains in around 1/4 of the time needed by the MLP-SVD1 network. In terms of singular values discarded, it is interesting to observe that both SVD techniques discard a similar number of singular values during training, however, due to the fact that MLP-SVD2 changes the size of the hidden layer, more training time is reduced. In terms of FLOPs, it can be observed that both SVD techniques have a smaller number of operations than the original MLP, and the MLP-SVD2 technique ends up having the least number of FLOPs in most of the cases. On average, over all 10 trials, the MLP-SVD1 technique ends up having between 4 and 15 singular values at the end of training, whereas the MLP-SVD2 method has between 4 and 16 remaining singular values. Another interesting observation is that when the network has only 20 neurons in the hidden layer, both SVD approaches end up having more singular values at the end of training (12–16 singular values) compared to when the dimension of the hidden layer is increased to 100 neurons (4–8 singular values). Interestingly, this suggests that starting with a larger number of neurons and allowing the MLP-SVD2 algorithm to prune the network may result in superior network designs. This will be investigated further by the authors.

5. Discussion and limitations

The results presented above confirm the potential of the proposed pruning technique. By progressively discarding singular values logarithmically, the network model has enough time between the discarding points to adapt to the new reduced architecture. The main limitation of this technique is that the predefined singular value threshold determines the number of discarded parameters. A solution would be to introduce an adaptive threshold as proposed, for example, very recently in [Ke et al. \(2023\)](#). In addition, the SVD approach has been tested thus far on small MLP models with only one hidden layer. Further work will be investigated using multiple hidden layers, where the SVD technique could be applied to each

Table 1. Comparison of training times and performances between MLP, MLP-SVD1, and MLP-SVD2 using a network with 20 and 100 neurons in the hidden layer, $\Delta f = 60$ Hz, and a filter bandwidth of 10 Hz. Results are averaged over ten trials. The numbers in bold represent the best performance in terms of accuracy, training time, and FLOPs.

Hidden layer	Accuracy (%)				Training time (s)				FLOPs			
	20 neurons		100 neurons		20 neurons		100 neurons		20 neurons		100 neurons	
Output classes	3	9	3	9	3	9	3	9	3	9	3	9
MLP	80.33	72.64	80.20	71.62	11.79	39.51	65.37	165.66	5200	5400	27 000	28 000
MLP-SVD1	80.50	72.59	80.73	71.38	9.09	30.18	23.69	65.51	5000	4200	2600	5600
MLP-SVD2	80.47	73.06	79.27	71.81	7.63	23.02	5.34	16.85	4100	4300	1000	2200

matrix, relating two layers, or only on the same matrix, relating the input with the first hidden layer. When it comes to larger matrices, where the SVD is more time-consuming, the authors believe that the proposed SVD approach could be useful as it can be adapted to any model architecture, having the option to determine the number of discarding points required during the training. Fewer discarding points result in fewer SVD computations on the weight matrices but, potentially, also fewer discarded parameters. On the other hand, the choice of a higher threshold might result in reducing the hidden layer dimensions once only, thus, minimising the use of the SVD on a large matrix. Finally, it should also be emphasised that the same technique could be applied to any fully connected layer that comprises part of more advanced network models, many of which contain one or more such fully connected layers.

6. Conclusion

This work presented an extension of the approach described by Paul and Nelson (2021) to reduce the training time of a MLP with one hidden layer by discarding iteratively singular values during training. The novelty compared to the technique in Paul and Nelson (2021) is that, here, the dimensions of the hidden layer are reduced depending on the number of singular values discarded. Following this, the MLP network is able to reduce its dimensions until the training stops. The presented technique could be extended to more layers or other network architectures provided the SVD can be applied to the weight matrices, however, the time reduction will be task dependent, and the discarding parameters will have to be adjusted correspondingly.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC, UKRI) EP/R513325/1.

References

- Augasta, M., and Kathirvalavakumar, T. (2013). "Pruning algorithms of neural networks—A comparative study," *Open Comput. Sci.* 3(3), 105–115.
- Bianco, M. J., Gerstoft, P., Traer, J., Ozanich, E., Roch, M. A., Gannot, S., and Deledalle, C.-A. (2019). "Machine learning in acoustics: Theory and applications," *J. Acoust. Soc. Am.* 146(5), 3590–3628.
- Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Gutttag, J. (2020). "What is the state of neural network pruning?," in *Proceedings of Machine Learning Systems*, Austin, TX, Vol. 2, pp. 129–146.
- Cai, C., Ke, D., Xu, Y., and Su, K. (2014). "Fast learning of deep neural networks via singular value decomposition," in *PRICAI 2014: Trends in Artificial Intelligence, PRICAI 2014*, Lecture Notes in Computer Science, edited by D. N. Pham and S. B. Park (Springer, Berlin), Vol. 8862, pp. 820–826.
- Choudhary, T., Mishra, V., Goswami, A., and Sarangapani, J. (2020). "A comprehensive survey on model compression and acceleration," *Artif. Intell. Rev.* 53, 5113–5155.
- Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and De Freitas, N. (2013). "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, Lake Tahoe, NV [Neural Information Processing Systems (NeurIPS), La Jolla, CA], Vol. 26.
- Grumiaux, P.-A., Kitić, S., Girin, L., and Guérin, A. (2021). "A review of sound source localization with deep learning methods," arXiv e-prints arXiv-2109.
- Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). "Speeding up convolutional neural networks with low rank expansions," arXiv:1405.3866.
- Ke, Z. T., Ma, Y., and Lin, X. (2023). "Estimation of the number of spiked eigenvalues in a covariance matrix by bulk eigenvalue matching analysis," *J. Am. Stat. Assoc.* 118(541), 374–392.
- Paul, V. S., and Nelson, P. A. (2021). "Matrix analysis for fast learning of neural networks with application to the classification of acoustic spectra," *J. Acoust. Soc. Am.* 149(6), 4119–4133.
- Psychogios, D. C., and Ungar, L. H. (1994). "SVD-NET: An algorithm that automatically selects network structure," *IEEE Trans. Neural Netw.* 5(3), 513–515.
- Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.-Y., and Sainath, T. (2019). "Deep learning for audio signal processing," *IEEE J. Sel. Top. Signal Process.* 13(2), 206–219.
- Shmalo, Y., Jenkins, J., and Krupchyskiy, O. (2023). "Deep learning weight pruning with RMT-SVD: Increasing accuracy and reducing overfitting," arXiv:2303.08986.
- Singh, A., and Plumbley, M. D. (2022). "A passive similarity based CNN filter pruning for efficient acoustic scene classification," arXiv:2203.15751.
- Suzuki, T., Abe, H., Murata, T., Horiuchi, S., Ito, K., Wachi, T., Hirai, S., Yukishima, M., and Nishimura, T. (2018). "Spectral pruning: Compressing deep neural networks via spectral analysis and its generalization error," arXiv:1808.08558.
- Xue, J., Li, J., and Gong, Y. (2013). "Restructuring of deep neural network acoustic models with singular value decomposition," in *Interspeech*, Lyon, France [International Speech Communication Association (ISCA), Baixas, France], pp. 2365–2369.
- Yang, H., Tang, M., Wen, W., Yan, F., Hu, D., Li, A., Li, H., and Chen, Y. (2020). "Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Seattle, WA (IEEE Computer Society, Los Alamitos, CA), pp. 678–679.