Incorporating Task-Related Information in Dimensionality Reduction of Neural Population Using Autoencoders

Qi Lian[1,2], Yunzhu Liu[3,4], Yu Zhao[1,2], and Yu Qi[2,5,6](✉)

[1] Qiushi Academy for Advanced Studies, Zhejiang University, Hangzhou, China
[2] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
qiyu@zju.edu.cn
[3] Information Technology Research Center, China Electronics Standardization Institute, Beijing, China
[4] China National Information Technology Standardization Technical Committee Secretariat, Beijing, China
[5] Frontiers Science Center for Brain and Brain-Machine Integration, Zhejiang University, Hangzhou, China
[6] Zhejiang Lab, Hangzhou, China

Abstract. Dimensionality reduction plays an important role in neural signal analysis. Most dimensionality reduction methods can effectively describe the majority of the variance of the data, such as principal component analysis (PCA) and locally linear embedding (LLE). However, they may not be able to capture useful information given a specific task, since these approaches are unsupervised. This study proposes an autoencoder-based approach that incorporates task-related information as strong guidance to the dimensionality reduction process, such that the low dimensional representations can better reflect information directly related to the task. Experimental results show that the proposed method is capable of finding task-related features of the neural population effectively.

Keywords: Neural population activity · Supervised dimensionality reduction · Long short-term memory network · Autoencoder

1 Introduction

In recent years, neural activities recorded from the primate cortex by implanted arrays of microelectrodes have gradually become a common tool for neural mechanism analysis [18,39]. Based on the extracted neural signals, several brain-machine interface (BMI) applications have been successfully applied. For example, algorithms that convert neural activity of a human with tetraplegia into the desired prosthetic actuator movements [14,15]. However, it remains a question about what insights can we gain from the recordings of a population of neurons

[32,33]. It is reported that population analyses are necessary for situations in which the neural mechanisms involve coordination of responses across neurons, where some mechanisms exist only at the level of the population and not at the level of single neurons [8]. Many studies of neural systems are shifting from single-neuron to population-level analyses.

The dimensionality reduction methods are traditionally defined as methods that map the high-dimensional data to low-dimensional data, which discover and extract features of interest into the shared latent variables [41]. Nowadays, dimensionality reduction plays an important role in the shifting process of neural signal analysis [8,10,31]. On the one hand, the recorded neural signal of a channel corresponds to an underlying neuron ensemble, the response of a particular neuron may obscure the information of other neurons within the ensemble. On the other hand, activities of nearby neurons tend to be dependent on each other, and they may be recorded by nearby channels [29]. Therefore, fewer channels are needed for the explanation of the recorded neural signals, and it is a common practice to select channels before subsequent analysis. Rather than inspecting each neuron separately, dimensionality reduction methods can analyze neural population recordings as a whole [8].

Several classical dimensionality reduction methods including linear and non-linear dimensionality reduction methods have been adopted to analyze neural signals. Principle component analysis (PCA) [19] is a linear dimensionality reduction method that projects the high-dimensional neural data into a new coordinate system, where the input data can be expressed with fewer variables and most of the variance of the data set can be captured. Non-linear dimensionality reduction methods have also been applied, such as the locally linear embedding method (LLE) [36] and Isomap [40]. LLE exploits local symmetries of linear reconstructions of the original dataset, it learns manifolds close to the dataset and project input data onto them. Isomap first determines the adjacency of the points on the manifold, and then the geodesic distances between all pairs of points are calculated on the manifold. Finally, the multidimensional scaling method is applied to obtain the embedding of data. The dimensionality reduction methods were employed using the population response signals alone in most existing studies [1,7,9,38]. In a real-world scenario, each data point in the high-dimensional firing rate space has a corresponding label comprised of one or more dependent variables, such as the subject's behavior, the subject's mental state, and so on. Neglecting the task-related information may cause the dimensionality reduction methods to fail to capture representative information of a specific task [24,30]. However, classical dimensionality reduction methods are unsupervised methods without effective ways to incorporate supervised task-related information.

Recent advances in deep artificial neural networks provide new techniques for nonlinear dimensionality reduction. The nonlinearity in neural networks enables non-linear multivariate data compression and visualization [5,13]. The autoencoder (AE) is firstly introduced in the 1980s, which plays an important role in unsupervised learning [37]. It is a simple yet effective unsupervised method to compress information of the input data. By reconstructing outputs from inputs using the criterion of the minimum possible amount of Euclidean distance, it learns a transformation that transforms inputs into a latent representation space [5]. Improvements of autoencoder including the denoising autoencoder (DAE) [42] and the variational autoencoder (VAE) [21] enhance the ability to learn effective representations from data. DAE aims to reconstruct clean data from noisy inputs. It can learn representations that are robust to the noise by adding Gaussian noises to samples or masking variables of samples randomly. The stacked denoising autoencoder explores a greedy strategy for building deep neural networks consist of several layers of denoising autoencoder [43]. The stacked layers are trained sequentially and a fine-tuning process is adopted to calibrate the whole neural network. VAE is proposed to learn better feature representation which can generate samples from the decoder. Instead of learning the encodings directly, it uses a variational Bayesian approach to optimize an approximation to the intractable posterior, which produces more stable and robust results. The strong feature extraction ability of the AEs can be employed for the dimensionality reduction of the neural population signals.

With the introduction of the task-related information, the objective of dimensionality reduction for the neural population can now be defined as to project the data while differences in the dependent variables are preserved as many as possible. In the extreme, we can seek to ‘demix’ the effects of the different dependent variables, such that each latent variable captures the characteristic of a single dependent variable [8]. The AEs are powerful non-linear unsupervised models that can learn effective low-dimensional representation for neural population signals. They are also flexible models that can easily incorporate supervised task-related information into the learning process. Further, given that the neural population activities are time-series data that are recorded sequentially. We can learn even better low-dimensional representation by treating it as another type of task-related information, which is incorporated through the architecture design of our model. Specifically, the long short-term memory (LSTM) model [16] which is a type of recurrent neural network (RNN) [27] is adopted to incorporate the information.

In this paper, we investigate supervised dimensionality reduction techniques for the neural population. The learned low-dimensional representation can better capture features of interest directly related to the task. The contributions of this paper are two-fold. Firstly, we propose a supervised dimensionality reduction architecture which is suitable for different kinds of autoencoders. The architecture incorporates task-related information into the learning process of low-dimensional representation through an artificial neural network module, which is termed as ‘regressor’. The autoencoder takes multi-channel neural recordings from the primary motor cortex as input and reconstructs them. In the meantime, the regressor predicts the task-related information from the learned low-dimensional latent representations. Secondly, we propose a supervised architecture that considers the time-series nature of neural population activities. A sequential encoder and a sequential decoder based on LSTM are employed to transform the input data into the latent space and reconstruct the input data from the latent space, respectively. The task-related information is also employed through a regressor in this architecture. Experiments are carried out with different kinds of autoencoders under different settings. The results show that our proposed method learns a more effective task-related low-dimensional representation of the neural population.

2 Method

In this section, we first introduce the dataset we used in this paper. Then we give the background knowledge of various autoencoders and the LSTM. Finally, we introduce our proposed supervised autoencoder-based dimensionality reduction method for the neural population.

2.1 Dataset

A dataset that contains multi-channel spike firing signals with synchronous kinematic information is adopted to evaluate the performances of the supervised and unsupervised dimensionality reduction methods [44]. The dataset is recorded from a male macaque monkey that performs a radial-4 center-out task in a 2-D horizontal plane. For each trial, a target ball appears on the screen in front of the monkey, and the monkey is requested to move a cursor to the target with the joystick. Once the monkey hits the target ball within 2 s and holds for 300 ms, rewards will be given. The neural signal is recorded by a 96-microelectrode Utah array which is implanted in the monkey's arm area of the primary motor cortex contralateral to the arm used in the experiments. A total of 96 channels of neural signals are recorded with Cerebus multichannel system at a sample rate of 30 kHz. The raw signals are filtered by a high-pass Butterworth filter and the detected spikes are sorted with Offline Sorter software to produce binned spike rates. The trajectory of the joystick is recorded synchronously with neural signals by a micro-controller system at a sample rate of 1 kHz. We downsample the trajectory to correspond to the bins of spike rates. A channel selection method and a data selection method are further employed such that 8 subsets of spike data are obtained. The details of the dataset are shown in Table 1.

2.2 Prerequisites

Autoencoder and Its Variations. Consider a data set of samples $\{\mathbf{x}_n\}$ where $n = 1, \cdots, N$, and $\mathbf{x}_n$ is a Euclidean variable with dimensionality D. A fully connected layer of the neural network can be defined as

$$y = \phi(Wx + b), \tag{1}$$

where W and b denote trainable weights and bias, and ϕ denotes a non-linearity function. A basic autoencoder consists of an encoder and a decoder. The encoder

Table 1. The details of the dataset.

Subset	#Neuron	#Trials	#Up	#Down	#Left	#Right
1	61	74	17	19	17	21
2	63	81	19	21	18	23
3	61	70	17	14	19	20
4	56	64	20	11	19	14
5	49	83	25	19	22	17
6	56	75	21	14	21	19
7	61	85	17	23	20	25
8	61	79	14	23	17	25

The #Neuron denotes the number of neurons and the #Trials denotes the number of total trials. The #Up, #Down, #Left, #Right denote the number of trials with up, down, left, and right directions, respectively.

is comprised of several fully connected layers and the layers are usually stacked one by one with reducing dimensionality. We can denote the encoded latent feature as $\mathbf{z}$, which is a Euclidean variable with dimensionality M. Then the encoder $E(\mathbf{x})$ can be defined as

$$E(\mathbf{x}) = \mathbf{z} = \phi_L(W^L\phi_{L-1}(W^{L-1}\phi_{L-1}(\cdots W^1\phi_1(\mathbf{x}) + b^1 \cdots) + b^{L-1}) + b^L), \quad (2)$$

where L denotes the number of stacked fully connected layers. Similarly, the decoder $D(\mathbf{z})$ can be defined as

$$D(\mathbf{z}) = \tilde{\mathbf{x}} = \phi_L(W^L\phi_{L-1}(W^{L-1}\phi_{L-1}(\cdots W^1\phi_1(\mathbf{z}) + b^1 \cdots) + b^{L-1}) + b^L), \quad (3)$$

where $\tilde{\mathbf{x}}$ denotes the reconstruction of $\mathbf{x}$. The loss function of the autoencoder is usually defined as the mean squared error between the input $\mathbf{x}$ and the reconstruction $\tilde{\mathbf{x}}$, which can be defined as

$$\mathcal{L}_{reconstruction} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_n - \tilde{\mathbf{x}}_n)^2, \quad (4)$$

where $\mathbf{x}_n$ and $\tilde{\mathbf{x}}_n$ denote the n^{th} sample and its reconstruction, respectively. In [5], the stacked fully connected layers of the encoder and decoder are trained layer-wise using a greedy strategy. However, as the proposed of more advanced techniques such as the Relu non-linearity function [28], the second-order optimizer Adam [20], and the batch normalization layer [17], the layer-wise training strategy is no longer needed. In this paper, we directly optimize the entire neural network for all autoencoder-based models.

The denoising autoencoder is proposed to make the learned representations robust to partial corruption of the input pattern [43]. It first corrupts the initial input $\mathbf{x}$ to get a partially destroyed version $\hat{\mathbf{x}}$ through a stochastic mapping.

The stochastic mapping process is usually defined as a randomly masking process, where a fixed number of features are chosen at random and their values are forced to 0. Another common corruption choice is to add Gaussian noise to each feature separately. In this paper, the stochastic mapping process that randomly masks features is selected as the default corruption choice.

The variational autoencoder introduces a stochastic variational inference that can deal with intractable posterior distributions [21]. Let us define the probabilistic encoder as $q_{\varphi}(\mathbf{z}|\mathbf{x})$ and the posterior of the generative model as $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})$. The prior over the latent variables can be defined to be a centered isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})$. We can then define $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ to be a multivariate Gaussian whose distribution parameters are estimated from $\mathbf{z}$ with an artificial neural network with multiple fully connected layers. Assume that the true posterior follows to an approximate Gaussian with diagonal covariance, which is defined as

$$log q_{\varphi}(\mathbf{z}|\mathbf{x}^i) = log\mathcal{N}(\mathbf{z};\boldsymbol{\mu}^i,\boldsymbol{\sigma}^{2(i)}\mathbf{I}), \tag{5}$$

where the mean and standard deviation are outputs of the encoding artificial neural network. Using the reparameterization trick, the estimator for the model and data point $\mathbf{x}^i$ is defined as

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\varphi};\mathbf{x}^i) \simeq \frac{1}{2}\sum_{j=1}^{J}(1+log((\sigma_j^{(i)})^2)-(\mu_j^{(i)})^2-(\sigma_j^{(i)})^2)+\frac{1}{L}\sum_{l=1}^{L} log p_{\boldsymbol{\theta}}(\mathbf{x}^i|\mathbf{z}^{i,l}), \tag{6}$$

where $\mathbf{z}^{i,l} = \boldsymbol{\mu}^i + \boldsymbol{\sigma}^i \odot \epsilon^l$ and $\epsilon^l \sim \mathcal{N}(0,\mathbf{I})$, and $\odot$ denotes element-wise product. The entire network can then be optimized with a standard back-propagation method [23].

Long Short-Term Memory. The LSTM is an improvement of vanilla RNN that aims to mitigate the gradient vanishing problem [6]. The input sequence is denoted as $\mathbf{x} = (x_1, \cdots, x_T)$, the hidden vector sequence is denoted as $\mathbf{h} = (h_1, \cdots, h_T)$, and the output vector sequence is denoted as $\mathbf{y} = (y_1, \cdots, y_T)$. The update rule of the hidden vector sequence of the vanilla RNN can be defined as

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{7}$$

where tanh denotes the hyperbolic tangent function, W_{xh} and W_{hh} are learnable weights and b_h is learnable bias. The output at timestamp t can be defined as

$$y_t = W_{hy}h_t + b_y, \tag{8}$$

where W_{hy} is the learnable weights and b_y is the learnable bias.

The LSTM architecture used in this paper is defined as

$$\begin{aligned}
i_t &= \tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_i),\\
j_t &= \mathrm{sigm}(W_{xj}x_t + W_{hj}h_{t-1} + b_j),\\
f_t &= \mathrm{sigm}(W_{xf}x_t + W_{hf}h_{t-1} + b_f),\\
o_t &= \tanh(W_{xo}x_t + W_{ho}h_{t-1} + b_o),\\
c_t &= c_{t-1} \odot f_t + i_t \odot j_t,\\
h_t &= \tanh(c_t) \odot o_t,
\end{aligned} \tag{9}$$

where sigm denotes the sigmoid function, the W_* variables are learnable weights and the b_* variables are learnable biases.

2.3 Supervised Autoencoders-Based Dimensionality Reduction for Neural Population

The architecture of our proposed supervised autoencoders for neural signal dimensionality reduction is shown in Fig. 1. Binned and smoothed neural firings are served as raw inputs. The supervised autoencoder module is divided into three parts including the encoder, the latent representation, and the decoder. The encoder first transforms the raw inputs into their latent representations through the encoder. Two separate forks stem from the latent representation. The first one is the unsupervised decoder which reconstructs the inputs from the latent representations. The second one is a supervised regressor which incorporates the task-specific information (kinematic information). The supervised regressor is implemented as an artificial neural network that takes the latent representation as input and predicts corresponding task-related information. The artificial neural network can be built by stacking several fully connected layers. The distance between the predicted movements and the kinematic information is measured by the mean squared error function.

The architecture of our proposed supervised autoencoder based on LSTM that considers the time sequence characteristic of the neural population is shown in Fig. 2. In Fig. 1, the encoder and the decoder are built as artificial neural networks that consist of fully connected layers. Now the encoder and the decoder are built as multi-layer LSTM networks. At each timestamp, the LSTM encoder takes current spikes and the previous hidden state as input and generates current hidden state and output. The output is considered as the latent representation, and two forks stem from the latent representation including the unsupervised LSTM decoder and the supervised regressor. The unsupervised LSTM decoder takes the latent representation as input and reconstructs the input spikes. The supervised regressor is the same as the one shown in Fig. 1, which takes the latent representation as input and predicts task-related information. Note that, we reconstruct the input spikes and predict task-related information at each timestamp.

The loss of our proposed model consists of two parts including the unsupervised reconstruction loss and the supervised regression loss. The unsupervised

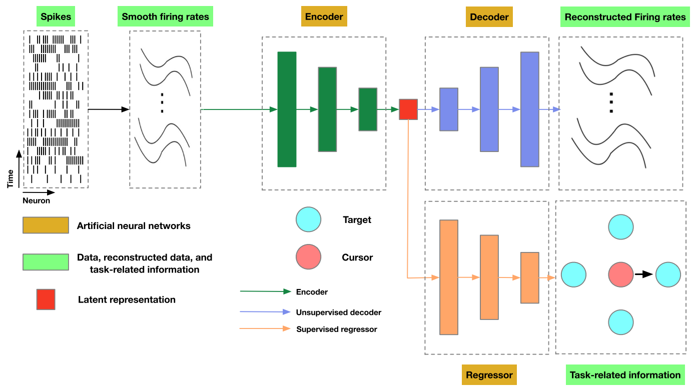

Fig. 1. The architecture of our supervised Autoencoders for dimensionality reduction. Binned and smoothed neural firings are served as raw inputs. The supervised autoencoder module can be divided into three parts including the encoder, the latent representations, and the decoder. The supervised encoder first transforms the raw inputs into their latent representations. Then two separate forks stem from the latent representation including the unsupervised decoder and the supervised regressor.

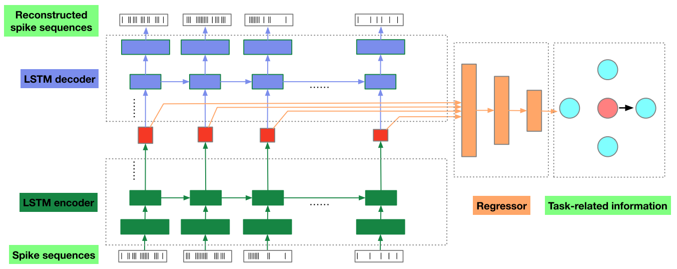

Fig. 2. The architecture of our proposed supervised autoencoder based on LSTM. This model considers the time sequence characteristic of the neural population. At each timestamp, the LSTM encoder takes current spikes and the previous hidden state as input and generates current hidden state and output. The LSTM decoder reconstructs the input spikes and the regressor predicts the task-relation information.

reconstruction loss computes the mean square error between the input spikes and the reconstructed spikes, which is denoted as $\mathcal{L}_{reconstruction}$. The supervised regression loss computes the mean square error between the predicted task-related information and the ground truth recorded simultaneously with the spikes, which can be denoted as $\mathcal{L}_{regression}$. We have also added an L2-regularization to the network to prevent overfitting, and its loss can be denoted as $\mathcal{L}_{regularization}$. Thus, the overall loss of our model can be defined as

$$\mathcal{L} = \mathcal{L}_{reconstruction} + \lambda_1 * \mathcal{L}_{regression} + \lambda_2 * \mathcal{L}_{regularization}, \tag{10}$$

where λ_1 and λ_2 are coefficients that trade off different losses. The entire network can be optimized using the standard back-propagation method.

3 Experimental Results

In this section, we first introduce the default settings we used for autoencoder-based models. Then we introduce the criteria we employed for performance evaluation. After that, we compare our proposed method with other unsupervised methods. Finally, we evaluate our proposed method under different settings including different types of autoencoders, different kinds of incorporated task-related information, and different levels of added noises to inputs.

3.1 Settings

The kinematic information is considered as the task-related information by default, which is the position of the joystick. Firstly, the recorded neural signals and kinematic information are smoothed with a window size set to 5. Then we standardize and scale the smoothed spikes to the range [0, 1]. The parameters λ_1 and λ_2 are set to 1 and 1e$-$4, respectively. The encoder we used in this paper is an artificial neural network consists of two fully connected layers with 64 and 32 units. The decoder we used in this paper is an artificial neural network consists of two fully connected layers with 32 and 64 units. The same encoder and decoder settings are used for all autoencoder models. The regressor we used to incorporate the supervised information is an artificial neural network consists of one fully connected layer with 32 units and a linear layer. The autoencoder and the denoising autoencoder use the Relu nonlinearity function, and the variational autoencoder uses the tanh nonlinearity function. No nonlinearity functions are applied after the last layer of the encoder, decoder, and the regressor for all models. We run ten trials for all models, and the final performance is obtained by averaging over ten trials for each of them. For all models, the weights are initialized with the He initialization method [12]. For autoencoder models without LSTM, the batch size is set to 64, the learning rate is set to 1e$-$3, and we run 200 epochs for each trial. The Adam optimizer is adopted for optimization.

For the autoencoder model based on LSTM, we mean-center the recorded neural signals and the kinematic information. The batch size is set to the number of trials of the subset, which means we optimize the network using the whole data of a subset at each step. We train the whole network for 5000 steps. The LSTM encoder is a two-layer LSTM network with 64 and 32 units. The LSTM decoder is a two-layer LSTM network with 32 and 64 units. The regressor is an artificial neural network consists of one fully connected layer with 32 units and a linear layer. The learning rate is set to 5e$-$3, and we decay the learning rate with a ratio set to 0.95 for every 500 steps. The Rmsprop is adopted for optimization [4]. The layer normalization is applied in our LSTM encoder and LSTM decoder [3]. Hereafter, the supervised versions of AE, DAE, and VAE are denoted as SAE, SDAE, and SVAE. Without loss of generality and to avoid introducing assumptions upon the dataset, the supervised autoencoder based on LSTM uses vanilla AE as building blocks and we denote it as LSTM-SAE.

3.2 Criterion

Two criteria are employed for performance comparison. The first one is the intra-class distance, the inter-class distance, and their ratio. The intra-class distance is defined as

$$d(\Omega_i)^2 = \frac{1}{N_i N_i} \sum_{k=1}^{N_i} \sum_{l=1}^{N_i} ||\mathbf{x}_k^i - \mathbf{x}_l^i||_2^2, \tag{11}$$

where Ω_i denotes the i^{th} class, $\mathbf{x}_k^i$ denotes the k^{th} samples of the i^{th} class, and N_i denotes the number of samples of the i^{th} class. The inter-class distance is defined as

$$d(\Omega_i, \Omega_j) = \frac{1}{N_i N_j} \sum_{k=1}^{N_i} \sum_{l=1}^{N_j} ||\mathbf{x}_k^i - \mathbf{x}_l^j||_2^2, \tag{12}$$

and the ratio is defined as

$$\mathcal{R} = \frac{1}{2(C-1)} \frac{\sum d(\Omega_i, \Omega_j)}{\sum d(\Omega_i)^2}, \tag{13}$$

where C denotes the number of classes. The second criterion is the silhouette score [34], which is a measure of how similar an object is to its own cluster compared to other clusters. Its value ranges from −1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

Table 2. Performance comparison with existing methods.

Method	$d(\Omega_i)^2$	$d(\Omega_i, \Omega_j)$	$\mathcal{R}$	Silhouette score
PCA	4.1199	8.8835	0.3700	0.1362
LLE	2.5551	8.5650	0.7292	0.3231
Isomap	3.2198	11.0912	0.6513	0.3129
LDA	3.2950	11.3075	0.5748	0.3799
NCA	3.2897	10.5672	0.5380	0.3879
KDA	**1.4771**	9.3935	**1.0720**	0.5128
AE	3.6004 ± 0.1268	9.5917 ± 0.2535	0.4626 ± 0.0149	0.2323 ± 0.0122
DAE	3.5703 ± 0.0757	8.8452 ± 0.1283	0.4277 ± 0.0104	0.1964 ± 0.0142
VAE	3.9912 ± 0.0528	9.0624 ± 0.0989	0.3873 ± 0.0088	0.1492 ± 0.0089
SAE	2.5356 ± 0.1340	11.3700 ± 0.3324	0.7983 ± 0.0683	0.5197 ± 0.0403
SDAE	2.5473 ± 0.0872	9.9521 ± 0.3430	0.6767 ± 0.0286	0.4653 ± 0.0218
SVAE	2.7154 ± 0.0558	11.6781 ± 0.1991	0.7281 ± 0.0049	0.5486 ± 0.0047
LSTM-SAE	2.3423 ± 0.1148	**13.7175 ± 0.3547**	1.0279 ± 0.0313	**0.6458 ± 0.0115**

AE, DAE, and VAE denote autoencoder, denoising autoencoder, and variational autoencoder, respectively. SAE, SDAE, SVAE denote supervised autoencoder, supervised denoising autoencoder, and supervised variational autoencoder, respectively. LSTM-SAE denotes the supervised autoencoder based on LSTM.

3.3 Comparison with Existing Methods

Several classical unsupervised and supervised methods are employed for comparison with our proposed supervised autoencoder methods. The unsupervised methods include PCA [19], LLE [36], and Isomap [40]. The number of neighbors is setting to 5 for LLE and Isomap. The supervised methods include LDA [26], NCA [35], and KDA [11]. The employed KDA uses the 'RBF' kernel and the corresponding parameter gamma is setting to 5. Note that, the discrete direction information is adopted as the task-related information for the classical supervised methods. The targeted dimensionality reduction methods for neuronal population data including dPCA [22], TDR [25] mTDR [2] are not considered in this paper because of the limited number of experimental task variables of the adopted dataset. We have also included the unsupervised autoencoder and its variations for comparison. The corruption ratios of the DAE and SDAE are set to 0.1. The dimensionality of the latent representation is set to 2. The learned features are scaled to the range [0, 1] before we compute the distances, ratio, and silhouette of the trials.

The results are shown in Table 2. As we can see, our proposed LSTM-SAE obtains the best performance of the inter-class distance and the Silhouette score. The KDA obtains the best performance of the intra-class distance and the best ratio. Our LSTM-SAE obtains an intra-class distance of 2.3423 and an inter-class distance of 13.7175, which leads to a ratio of 1.0279 that is comparable to the best ratio of 1.0720 obtained by KDA. Our LSTM-SAE also obtains the best Silhouette score of 0.6458. The better intra-class distance and ratio obtained by KDA is mainly due to the fact that KDA only considers the direction information and neglects the trace information. The consequences are two-fold, on the one hand, KDA can maps samples into a more compact region of the low-dimensional space, which results in better intra-class distance and ratio. On the other hand, KDA may fail to separate points from different directions in the low-dimensional space, given limited direction information and powerful kernel. The statement is confirmed by the visualization we will discuss later. KDA obtains the best performance among the baseline methods and outperforms unsupervised autoencoders. The supervised autoencoders (SAE, SDAE, and SVAE) obtain comparable performances with KDA. The supervised autoencoders beat their corresponding unsupervised versions by big margins. The results show that the incorporation of supervised information is crucial for the learning of discriminative low-dimensional representations.

The visualizations of the learned representations of different methods are shown in Fig. 3. The eighth subset of the dataset is selected for visualization. We use different colors for different classes, which represent different directions. The red lines plot trials with direction 'up', the green lines plot trials with direction 'down', the blue lines plot trials with direction 'left', and the yellow lines plot trials with direction 'right'. The numbers of trials with different directions are shown in Table 1, each trial is visualized as a single line. As we can see in Fig. 3, compares with other existing methods, KDA obtains better latent representations with better cohesion within each class and separation between classes.

Autoencoders without supervised information including AE, DAE, and VAE fail to learn discriminative latent representations. However, autoencoders that take advantage of supervised information including SAE, SDAE, and SVAE learn better latent representations, as we can see from the improved performances in Table 2 and the discriminative latent representations in Fig. 3. As shown by our proposed LSTM-SAE, considering the time-series nature of the neural population and incorporating it into the architecture design can further improve the performance. As we have mentioned earlier, KDA maps samples into a more compact region with disordered lines of different directions, and some directions can be indistinguishable.

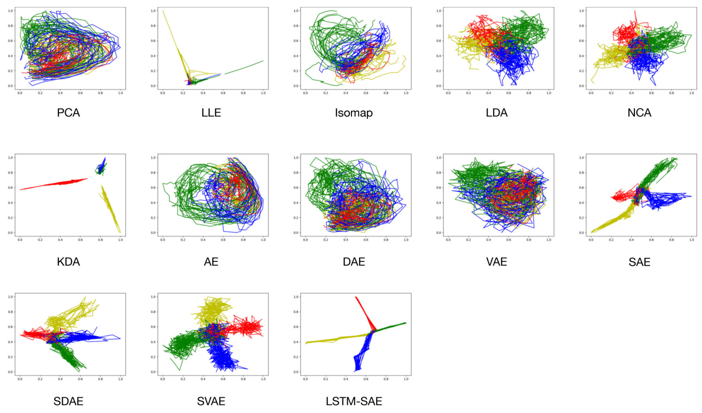

Fig. 3. The visualizations of the latent representations of different methods. We visualize classical unsupervised methods including PCA, LLE, and Isomap, and supervised methods including LDA, NCA, and KDA. Unsupervised autoencoders including AE, DAE, and VAE are also visualized. Our proposed methods including SAE, SDAE, SVAE, and LSTM-SAE are visualized in the second and the third row. The red lines plot trials with direction 'up', the green lines plot trials with direction 'down', the blue lines plot trials with direction 'left', and the yellow lines plot trials with direction 'right'. (Color figure online)

3.4 Model Evaluation Under Different Settings

In this section, we evaluate our proposed supervised autoencoder-based methods under different settings. Firstly, we evaluate our proposed methods with different types of autoencoders. Then we evaluate our proposed methods with different kinds of task-related information. After that, we evaluate the performances with different levels of noise adding to the inputs.

We first evaluate the performances of our proposed methods with different types of autoencoders. The results are shown in Table 2, and their corresponding visualizations are shown in Fig. 3. Compared with AE, SAE improves the ratio from 0.4626 to 0.7983 and the silhouette score from 0.2323 to 0.5197. Compared with DAE, SDAE improves the ratio from 0.4277 to 0.6767 and the silhouette score from 0.1964 to 0.4653. Compared with VAE, SVAE improves the ratio from 0.3873 to 0.7281 and the silhouette score from 0.1492 to 0.5486. As shown in Fig. 3, unsupervised autoencoders fail to learn discriminative latent representations of different directions. On the opposite, our proposed supervised autoencoders successfully learn discriminative latent representations for most of the trials. LSTM-SAE learns near-optimal latent representations, given that the start points of all trials should be the same and thus will overlap with each other. The results show that, compared with unsupervised autoencoders, our proposed supervised autoencoders can effectively improve the learned latent representations.

Table 3. The Silhouette scores of different supervised autoencoders with various combinations of task-related information.

Method	Information				
	P	V	A	PV	PVA
SAE	0.5197 ± 0.0403	0.3972 ± 0.0163	0.3090 ± 0.0233	0.5302 ± 0.0239	0.4675 ± 0.0195
SDAE	0.4653 ± 0.0218	0.3237 ± 0.0151	0.2576 ± 0.0165	0.4082 ± 0.0227	0.3695 ± 0.0194
SVAE	0.5486 ± 0.0047	0.4457 ± 0.0055	$\mathbf{0.3851 \pm 0.0098}$	0.6125 ± 0.0116	0.5795 ± 0.0089
LSTM-SAE	$\mathbf{0.6458 \pm 0.0115}$	$\mathbf{0.4703 \pm 0.0165}$	0.3361 ± 0.0127	$\mathbf{0.6701 \pm 0.0192}$	$\mathbf{0.6500 \pm 0.0059}$

P denotes the position information, V denotes the velocity information, and A denotes acceleration information. PV denotes the combination of the position and velocity information. PVA denotes the combination of the position, velocity, and acceleration information.

Next, we evaluate the performances of our proposed methods with different kinds of task-related information. Three kinds of task-related information are considered in this paper including the position, velocity, and acceleration. Five sets of experiments are carried out with different combinations of them. The Silhouette scores are shown in Table 3. As we can see, the most informative task-related information is the position, since all supervised models obtain their best performance given solely the position information. Comparison with position information available solely, the addition of velocity information on the basis of position information improves the performances of SAE, SVAE and LSTM-SAE, and hurts the performance of SDAE. Comparison with position information available solely, the addition of velocity and acceleration information hurts the performances of SAE and SDAE, but slightly improves the performance of SVAE and LSTM-SAE. As the results showed, LSTM-SAE obtains best performances in most cases, and SVAE utilizes the additional information most effectively.

Table 4. The Silhouette scores of different supervised autoencoders with various noise levels.

Method	Noise level				
	0.00	0.05	0.1	0.15	0.2
SAE	0.5197 ± 0.0403	0.4691 ± 0.0178	0.3939 ± 0.0230	0.3378 ± 0.0245	0.3039 ± 0.0156
SDAE	0.5176 ± 0.0234	0.5102 ± 0.0292	0.4663 ± 0.0187	0.4265 ± 0.0198	0.3933 ± 0.0240
SVAE	0.5486 ± 0.0047	0.4759 ± 0.0072	0.4032 ± 0.0080	0.3394 ± 0.0053	0.2836 ± 0.0052
LSTM-SAE	$\mathbf{0.6458 \pm 0.0115}$	$\mathbf{0.6358 \pm 0.0207}$	$\mathbf{0.6374 \pm 0.0206}$	$\mathbf{0.6158 \pm 0.0181}$	$\mathbf{0.5748 \pm 0.0135}$

Finally, we evaluate the performances of our proposed supervised autoencoders with different levels of added noises. The noises we added to the samples are identical to the corruption process we applied for the denoising autoencoder. Different corruption ratios are considered including 0.05, 0.1, 0.15 and 0.2. The noises are added in the testing stage after training completed. The performances are shown in Table 4. As we can see, as the level of noise increases, the performances of all models decrease. Compared with SAE and SVAE, SDAE is more robust to noise, which is a reasonable result because the training process of DAE has already considered robustness to noises. It is a surprise that our proposed LSTM-SAE also represents robustness to noises. We conjecture that the robustness may come from the time-series nature of the neural population, which implies that LSTM-SAE has successfully learned the dynamical time structure of the neural population.

4 Conclusions

In this paper, we address the problem of information loss using unsupervised dimensionality reduction methods on neural population signals. We design a supervised architecture base on autoencoder which incorporates task-related information as strong guidance to the dimensionality reduction process, thus the low dimensional representations can better capture information that is directly related to the task. We also consider the time-series nature of the neural population and incorporate it using an LSTM based autoencoder. Our experimental results show that the proposed architecture captures information related to the task effectively.

Acknowledgments. This work was partly supported by the grants from National Key R&D Program of China (2018YFA0701400), National Natural Science Foundation of China (No. 61673340), Zhejiang Provincial Natural Science Foundation of China (LZ17F030001), Fundamental Research Funds for the Central Universities (2020FZZX001-05), and the Zhejiang Lab (2019KE0AD01).

References

1. Afshar, A., Santhanam, G., Byron, M.Y., Ryu, S.I., Sahani, M., Shenoy, K.V.: Single-trial neural correlates of arm movement preparation. Neuron **71**(3), 555–564 (2011)

2. Aoi, M., Pillow, J.W.: Model-based targeted dimensionality reduction for neuronal population data. In: Advances in Neural Information Processing Systems, pp. 6690–6699 (2018)
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
4. Bengio, Y., CA, M.: RMSProp and equilibrated adaptive learning rates for non-convex optimization. Corr abs/1502.04390 (2015)
5. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems, pp. 153–160 (2007)
6. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
7. Briggman, K.L., Abarbanel, H.D., Kristan, W.B.: Optical imaging of neuronal populations during decision-making. Science **307**(5711), 896–901 (2005)
8. Cunningham, J.P., Byron, M.Y.: Dimensionality reduction for large-scale neural recordings. Nature Neurosci. **17**(11), 1500–1509 (2014)
9. Durstewitz, D., Vittoz, N.M., Floresco, S.B., Seamans, J.K.: Abrupt transitions between prefrontal neural ensemble states accompany behavioral transitions during rule learning. Neuron **66**(3), 438–448 (2010)
10. Gibson, S., Judy, J.W., Markovic, D.: Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. IEEE Trans. Neural Syst. Rehabil. Eng. **18**(5), 469–478 (2010)
11. Hand, D.J.: Kernel Discriminant Analysis, p. 264. Wiley, New York (1982)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
14. Hochberg, L.R., et al.: Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. Nature **485**(7398), 372–375 (2012)
15. Hochberg, L.R., et al.: Neuronal ensemble control of prosthetic devices by a human with tetraplegia. Nature **442**(7099), 164–171 (2006)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
17. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
18. Jackson, A., Mavoori, J., Fetz, E.E.: Long-term motor cortex plasticity induced by an electronic neural implant. Nature **444**(7115), 56–60 (2006)
19. Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **374**(2065), 20150202 (2016)
20. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
22. Kobak, D., et al.: Demixed principal component analysis of neural population data. Elife **5**, e10989 (2016)
23. LeCun, Y., et al.: Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems, pp. 396–404 (1990)
24. Lian, Q., Qi, Y., Pan, G., Wang, Y.: Learning graph in graph convolutional neural networks for robust seizure prediction. J. Neural Eng. **17**, 035004 (2020)

25. Mante, V., Sussillo, D., Shenoy, K.V., Newsome, W.T.: Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature **503**(7474), 78–84 (2013)
26. McLachlan, G.J.: Discriminant Analysis and Statistical Pattern Recognition, vol. 544. Wiley, New York (2004)
27. Mikolov, T., Karafiát, M., Burget, L., Černocky, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association, pp. 1045–1048 (2010)
28. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML (2010)
29. Nordhausen, C.T., Maynard, E.M., Normann, R.A.: Single unit recording capabilities of a 100 microelectrode array. Brain Res. **726**(1–2), 129–140 (1996)
30. Pan, G., et al.: Rapid decoding of hand gestures in electrocorticography using recurrent neural networks. Front. Neurosci. **12**, 555 (2018)
31. Pang, R., Lansdell, B.J., Fairhall, A.L.: Dimensionality reduction in neuroscience. Current Biol. **26**(14), R656–R660 (2016)
32. Panzeri, S., Macke, J.H., Gross, J., Kayser, C.: Neural population coding: combining insights from microscopic and mass signals. Trends Cogn. Sci. **19**(3), 162–172 (2015)
33. Qi, Y., Liu, B., Wang, Y., Pan, G.: Dynamic ensemble modeling approach to nonstationary neural decoding in brain-computer interfaces. In: Advances in Neural Information Processing Systems, pp. 6089–6098 (2019)
34. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
35. Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood component analysis. Adv. Neural Inf. Process. Syst. (NIPS) **17**, 513–520 (2004)
36. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
37. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. California Univ., San Diego, La Jolla, Inst. for Cognitive Science, Technical report (1985)
38. Seidemann, E., Meilijson, I., Abeles, M., Bergman, H., Vaadia, E.: Simultaneously recorded single units in the frontal cortex go through sequences of discrete and stable states in monkeys performing a delayed localization task. J. Neurosci. **16**(2), 752–768 (1996)
39. Suner, S., Fellows, M.R., Vargas-Irwin, C., Nakata, G.K., Donoghue, J.P.: Reliability of signals from a chronically implanted, silicon-based electrode array in non-human primate primary motor cortex. IEEE Trans. Neural Syst. Rehabil. Eng. **13**(4), 524–541 (2005)
40. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
41. Van Der Maaten, L., Postma, E., Van den Herik, J.: Dimensionality reduction: a comparative. J. Mach. Learn. Res. **10**(66–71), 13 (2009)
42. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1096–1103 (2008)
43. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A., Bottou, L.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**(12), 3371–3408 (2010)
44. Zhou, L., et al.: Decoding motor cortical activities of monkey: a dataset. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 3865–3870. IEEE (2014)