A Voice Conversion Mapping Function based on a Stacked Joint-Autoencoder

Seyed Hamidreza Mohammadi, Alexander Kain

Center for Spoken Language Understanding, Oregon Health & Science University
Portland, OR, USA
mohammah@ohsu.edu, kaina@ohsu.edu

Abstract
In this study, we propose a novel method for training a regression function and apply it to a voice conversion task. The regression function is constructed using a Stacked Joint-Autoencoder (SJAE). Previously, we have used a more primitive version of this architecture for pre-training a Deep Neural Network (DNN). Using objective evaluation criteria, we show that the lower levels of the SJAE perform best with a low degree of jointness, and higher levels with a higher degree of jointness. We demonstrate that our proposed approach generates features that do not suffer from the averaging effect inherent in back-propagation training. We also carried out subjective listening experiments to evaluate speech quality and speaker similarity. Our results show that the SJAE approach has both higher quality and similarity than a SJAE+DNN approach, where the SJAE is used for pre-training a DNN, and the fine-tuned DNN is then used for mapping. We also present the system description and results of our submission to Voice Conversion Challenge 2016. Index Terms: voice conversion, deep neural network, autoencoder, joint-autoencoder

1. Introduction
A Voice Conversion (VC) system converts speech produced by a source speaker to sound similar to that of a target speaker. Various approaches have been proposed; most commonly, a generative approach analyzes speech frame-by-frame, then maps extracted source speaker features towards target speaker features, with a subsequent synthesis procedure. The mapping is achieved using a non-linear regression function, which must be trained on aligned source and target features from existing parallel or artificially parallelized [e.g. 1] speech.

Recently, various Artificial Neural Network (ANN) architectures have been proposed for the task of feature mapping in the context of VC: Deep Neural Networks (DNNs) [2] and pre-trained DNNs [3, 4, 5, 6, 7, 8]. These ANN variations have consistently achieved improvements in both quality and similarity over Gaussian Mixture Models (GMMs). However, the DNNs still exhibit the “averaging effect”, seen as decreased variance in the output features, since the objective during training is to minimize a mean-squared or cross-entropy error function on aligned features that are likely to exhibit the many-to-one mapping problem [9, 10].

In a previous study [7], we proposed a Stacked Joint-Autoencoder architecture to pre-train a DNN. This architecture has the property of joining two Autoencoders (AEs), each of which is being trained on the input and output variables, respectively, at the hidden layers. This forces the autoencoders to learn similar hidden layer representations for what amounts to different views of the same process: specifically, in the case of voice conversion, the source and target speaker features can be regarded as two different views of an identical phonetic context (since the data are aligned). We showed that, when using a large amount of unrelated speakers’ data during unsupervised training prior to joining the AEs, we needed fewer parallel utterances during supervised training to achieve similar VC performance as compared to supervised-only training of a DNN.

In this study, we propose using a more generalized SJAE architecture directly for feature mapping, as opposed to using it merely for pre-training a DNN. We show that the averaging effect is minimal and the variance of the generated features is close to that of the original target features. For comparison, we also construct DNNs from the proposed SJAE, as well as from a regular stacked AE (without any joining), with subsequent fine-tuning via back-propagation.

2. Network Architectures
For a classification task, a neural network can be pre-trained on the input data in order to capture the input data distributions more effectively. Typically, no special effort is made to also capture the data distribution of the labels, since their distributions are trivial, relieving the need for any pre-training to learn that distribution. However, in a VC task, or any regression tasks with high-dimensional outputs, capturing both the source and target speaker data distributions is useful, similar to the advantage of using a Joint-Density Gaussian Mixture Model [11] over other, non-joint GMM approaches.

For a regression task, the mapping between source and target features can be learned using a neural network with random initialization of weights which are then fine-tuned via the back-propagation algorithm. One way to pre-train the network is via the following method: First, two separate AEs, one for the source and one for the target speaker’s features, are trained unsupervisedly. (If the size of the available source and target data is relatively small, one can also first train a single AE on unrelated speakers’ data as a starting point for source and target AE training.) Then, the neural network can be constructed using the encoding part of the source AE, an arbitrary mapping layer, and the decoding part of the target AE. Finally, the network is trained via back-propagation, supervisedly. The reason for the existence of the mapping layer is that the hidden layer values of the source and target autoencoders are likely to be uncorrelated, and thus the additional mapping is required to map these values [3, 4]. We will now introduce the concept of the Joint-Autoencoder (JAE), which is designed to eliminate the need for the additional mapping layer, by forcing the hidden representation to be similar for parallel source and target feature pairs.

2.1. Joint-Autoencoder (JAE)
Let $X_{N \times D_s} = [x_1, \ldots, x_N]^\top$, where $x = [x_1, \ldots, x_{D_s}]$, represent $N$ examples of $D_s$-dimensional source feature training
each trained on appropriate, non-parallel (i.e. “unlabeled” in native is to initialize the parameters from biases can be random, small values, but a likely better alteration of jointness. An example JAE architecture is depicted in $\alpha$ controls the tradeoff between the reconstruction and the de-
tween the hidden values of the two otherwise separate AEs, and
where $E_h = (1 - \alpha) \cdot \frac{r(x, \hat{x}) + r(y, \hat{y})}{d(h_x, h_y)}$ (2)
where $r$ is the reconstruction error, $d$ is the distortion error be-
tween the hidden values of the two otherwise separate AEs, and $\alpha$ controls the tradeoff between the reconstruction and the degree of 
jointness. An example JAE architecture is depicted in Fig. 1, top.

The JAE’s weights and biases are estimated using a stochastic gradient descent algorithm. The initial values of weights and biases can be random, small values, but a likely better alternative is to initialize the parameters from two individual AEs, each trained on appropriate, non-parallel (i.e. “unlabeled” in the context of regression) data. For example, if the VC task is

2.2. Stacked Joint-Autoencoder (SJAE)
Similar to AEs, which can be stacked to form Stacked Au-
Toencoders (SAEs), AEs can also be stacked together to form SJAEs. A first-level JAE is trained on source and target parame-
ters, which are then encoded (Fig. 1, top). Then a second-level JAE is trained on those source and target encodings (Fig. 1, middle). The process can be iterated until the desired depth is obtained. Each level can be assigned arbitrary $\alpha$ values; we describe the search for optimal values in Section 4.2.

2.3. SJAE-derived Mapping Network
The SJAE reduces the complexity of, or entirely eliminates the need for any additional mapping when used in the context of re-
gression, allowing the construction of a neural network consisting of the source-encoding part of the SJAE, followed directly by the target-decoding part:

$$F(x^{\text{test}}) = f_{\text{vis}}(V^\top h_{\text{hid}} + b_{\text{hid}}) + c_{\text{vis}}$$ (3)

where $x^{\text{test}}$ is an input feature vector (see Fig. 1, bottom). Arbi-
trarily deep structures can be created, depending on the number of levels in the SJAE.

2.4. SJAE-derived DNN
The SJAE-derived mapping network can also serve as an initial-
ization of a DNN, which is then fine-tuned. An advantage over other types of initialization strategies for DNNs in a regression framework is the greedy layer-by-layer training of the network layers, thus addressing the vanishing gradient problem. More-
over, this approach initializes all DNN layers independently of each other, helping the back-propagation start from a better initial state [5]. Finally, SJAE-based pre-training considers both input and output data distributions, which makes it well-suited for regression tasks.

3. VC Challenge 2016 Submission
For this challenge, we submitted a second type of VC system, one that can utilize the fact that a relatively large amount of training data were available. This second method was based on Frame Selection (FS), which works similar to unit-selection for Text-to-Speech systems, except the units in this case are frames, as proposed for text-independent VC [12, 13]. The goal is to find the best sequence of indices $S$ of the training target vectors, $Y$, such that $F_{\text{fs}}(x^{\text{test}}) = Y_{\text{S}}$, where $S$ is optimal with respect to spectral target and concatenation distortion measures. To en-
sure continuity, a maximum likelihood parameter generation al-
gorithm [14] is used in post-processing, using a fixed standard deviation computed from the training data as additional parameter. The features used in this system are single frames of 39th-
order mel-cepstra (MCEPs) and their deltas.

4. Experiments
4.1. Training
We used the VC challenge 2016 corpus as speech data, con-
sisting of 162 parallel sentences. We split the sentences into training (120), validation (22), and testing (20) sentences. The training sentences are further split into small, medium, and large training sets that have 5, 20, and 120 sentences, respec-
tively. For simplicity, we only considered four conversions: SF1$\rightarrow$TF1, SF2$\rightarrow$TM1, SM1$\rightarrow$TF2, and SM2$\rightarrow$TM2. As speech features, we used 39th-order MCEPs (excluding the 0th

Figure 1: Stacking JAEs and SJAE-derived mapping function

vectors. Using a parallelization method (e.g. time-alignment and subsequent interpolation), we can obtain the associated matrix $Y_{N \times D_y} = [y_1, \ldots, y_N]^\top$, where $y = [y_1, \ldots, y_D_y]$, rep-
resenting target feature training vectors. The source and target AEs can be represented as

$${h_x} = f_{\text{hid}}(Wx + b_{\text{hid}}), \quad {\hat{x}} = f_{\text{hid}}(W^\top h_x + b_{\text{hid}})$$

$${h_y} = f_{\text{hid}}(Vy + c_{\text{hid}}), \quad {\hat{y}} = f_{\text{hid}}(V^\top h_y + c_{\text{hid}})$$ (1)

where $W$ and $b$ are the weights and biases responsible for re-
constructing the source features, $V$ and $c$ are the weights and biases responsible for reconstructing the target features, and $h_x$ and $h_y$ are the source and target hidden layer values, respec-
tively. The transfer functions are represented by $f_{\text{hid}}$ and $f_{\text{hid}}$ for hidden and visible layers. The core idea of the JAE is to max-
imize the similarity of the encoding values between the source and target AEs, in addition to the goal of reconstruction. There-
fore, we modify the standard training cost function to include the error between the hidden layer encodings, i.e.

$$E_h = (1 - \alpha) \cdot \frac{r(x, \hat{x}) + r(y, \hat{y})}{d(h_x, h_y)}$$ (2)
coefficient / energy), extracted using the Ahocoder toolkit [15] with a 5 ms frame shift. Based on a study of phone recognition on the TIMIT database [16], we chose to model 15 frames (the current frame plus 7 preceding and following frames), for a total of 15×39=585 features per frame [7].

For training the SJAE, we use the TIMIT database by splitting it into source-specific and target-specific data that are most similar to the source and target speakers, respectively [7]. The SJAE is trained level-by-level, where the source AE is trained using non-parallel source-specific data, and the target AE is trained using non-parallel target-specific data. The JAE is then trained using the source and target supervised data. For the next level, the source-specific non-parallel data is encoded using the source AE (and similarly for the target-specific non-parallel data). The next level JAE is pre-trained (i.e. its source/target AE are trained separately) using the encoded non-parallel source- and target-specific data and then trained using the encoded source and target data. In this study, we use all of the TIMIT data as the non-parallel source-specific and target-specific data, leading to no distinction between source and target for the purposes of pre-training the SJAE. Finally, we derive the SJAE-Map and DNN from the trained SJAE as described in Section 2.3 and Section 2.4, respectively. For pre-training the DNN in the SAE+DNN case, we first train a single SAE on the TIMIT dataset and use the SAE to initialize the DNN weights and then fine-tune it [7].

The two architectures were trained using all possible jointness sequences initializing the model using the model on the left of the figure. The SAE was trained on TIMIT. The “+” sign represented. The best jointness sequence was {[0, 1, 0, 0], [1, 1, 0, 0], [0, 0, 1, 0], [0, 1, 1, 0], [0, 1, 0, 1]}. The AE was trained level-by-level, where the source AE is trained using non-parallel source-specific data, and the target AE is trained using non-parallel target-specific data. The JAE is then trained using the source and target supervised data. For the next level, the source-specific non-parallel data is encoded using the source AE (and similarly for the target-specific non-parallel data). The next level JAE is pre-trained (i.e. its source/target AE are trained separately) using the encoded non-parallel source- and target-specific data and then trained using the encoded source and target data. In this study, we use all of the TIMIT data as the non-parallel source-specific and target-specific data, leading to no distinction between source and target for the purposes of pre-training the SJAE. Finally, we derive the SJAE-Map and DNN from the trained SJAE as described in Section 2.3 and Section 2.4, respectively. For pre-training the DNN in the SAE+DNN case, we first train a single SAE on the TIMIT dataset and use the SAE to initialize the DNN weights and then fine-tune it [7].

For the first-level AE, f_{hid1} was a sigmoidal transfer function, and f_{enc} was linear. For all higher levels, we used sigmoid functions for both encoding and decoding. We let r and d equal the mean-squared error for the first-level AE, but for all higher levels we used the cross-entropy error for both. Additionally, we used a Gaussian corruption function for the first-level AE (also known as de-noising), whereas all higher levels used a binomial corruption function (also known as dropout).

4.2. Optimizing Jointness Factors
We conducted a grid search to find the optimal \( \alpha \) for each level of the SJAE, for the SM1→TF2 conversion, using the large training set. The possible range for \( \alpha \) was set between 0.0 and 0.5, with 0.1 increments, for all layers (i.e. there was no monotonicity requirement). We hypothesized that the lower levels (representing lower-level features such as frequencies) perform best with minimal joining, since the distributions are likely to be disparate at this level, whereas the higher levels benefit from a higher degree of jointness since they represent underlying abstract features (e.g. phonetic context). We evaluated two network architectures:

- **SJAE-Map**: A SJAE-derived mapping, where the SJAE had three levels, with hidden layer sizes of 500, 300, 200, from lowest-level to highest-level, respectively. The network was trained with a dropout rate of 0.2.

- **SJAE+DNN**: The previous network was used to initialize a DNN and fine-tuned for 1000 iterations. The two architectures were trained using all possible jointness factor combinations. The best-performing network in terms of mel-cepstral distortion (melCD) [14] on validation data was selected. The best jointness sequence was [0.1, 0.1, 0.5] for both SJAE-Map and SJAE+DNN, confirming our hypothesis.

4.3. Objective Evaluation
We compared the following five systems objectively: Shallow Neural Network (SNN), SAE+DNN, SJAE-Map, SJAE+DNN, and FS. The SAE was trained on TIMIT. The “+” sign represents initializing the model using the model on the left of the table.

<table>
<thead>
<tr>
<th></th>
<th>no map</th>
<th>small</th>
<th>medium</th>
<th>large</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>9.22</td>
<td>9.12</td>
<td>8.56</td>
<td>8.13</td>
</tr>
<tr>
<td>SNN</td>
<td>9.22</td>
<td>7.30</td>
<td>6.96</td>
<td>6.88</td>
</tr>
<tr>
<td>SJAE+DNN</td>
<td>9.22</td>
<td>7.13</td>
<td>6.77</td>
<td>6.65</td>
</tr>
<tr>
<td>SJAE-Map</td>
<td>9.22</td>
<td>7.95</td>
<td>7.63</td>
<td>7.42</td>
</tr>
<tr>
<td>SJAE+DNN</td>
<td>9.22</td>
<td>7.07</td>
<td>6.71</td>
<td>6.53</td>
</tr>
</tbody>
</table>

Table 1: Objective evaluation of proposed approaches

4.4. Visually inspecting output of the systems
In the first visualization, we set only one element in the innermost hidden layer of the SJAE layer to “1” and the others to “0”. In Fig. 2, the corresponding source (SM1, male) and target (TF2, female) speaker spectrum reconstructions from the one-hot hidden layer is depicted. We observe that the spectra are matching regarding spectral peak occurrence (with the female spectral peaks location represented as green higher in frequency compared to male spectral peaks represented as blue).

Fig. 3 shows the original target, SJAE-Map, and SJAE+DNN generated spectra. The SJAE+DNN generated spectra appear more averaged (which subjectively results in more muffled speech), while SJAE-Map generated spectra have sharper formants, which is more similar to the “sharpness” of the original spectrum.

Finally, we look at the the standard deviation ratio of the generated features of SJAE-Map and SJAE+DNN to original target features. For each MCEP dimension, this is computed as the standard deviation of the generated features, divided by the standard deviation of the original features. As shown in Fig. 4, FS and SJAE-Map generated features have a ratio around 1.0, which shows that they have similar standard deviation as the original speech; however, SJAE+DNN has a lower ratio, especially for the higher MCEP coefficients. The ability of a mapper to retain the standard deviation of the generated features is typically correlated with higher speech quality [17].

4.5. Subjective Evaluation
To subjectively evaluate VC performance, we performed two perceptual tests: the first test measured speech quality and the second test measured speaker similarity. The listening experi-
ments were carried out using Amazon Mechanical Turk, with 100 participants who had approval ratings of at least 90% and were located in North America. We followed the VC challenge 2016 evaluation description. We compared the following systems: SNN, SAE+DNN, SJAE-Map, SJAE+DNN, and FS.

4.5.1. Speech Quality Test
We used the standard Mean Opinion Score (MOS) to evaluate the speech quality. The results are shown in Figure 5. For the large training set, SJAE-Map significantly ($p < 0.01$) outperformed all other systems. SJAE+DNN was also significantly ($p = 0.005$) better than SNN. FS had a bimodal score distribution due to varying degrees of concatenation errors, which resulted in the system not performing as well as other systems. For the small training set, SJAE-Map significantly ($p = 0.01$) outperformed SAE+DNN. Significance testing was performed using a $t$-test. The good performance of SJAE-Map is due to a marked decrease of the averaging effect, which resulted in notably less muffling in the generated speech.

4.5.2. Speaker Similarity Test
In this test, two stimuli, one original target and one converted target, were played for the listener. They were instructed to choose the similarity of the speakers of the stimuli by choosing from: definitely the same or the same (positive), different or definitely different (negative). The similarity score is the percentage of positive responses. For significance testing, we utilized the binomial experiment [18] with a sample size of 800. Significantly different system pairs are highlighted using a dashed line in Fig. 6. The results show that for the large training set, SJAE-Map and FS significantly outperformed the other systems. Also, both types of DNNs significantly outperformed the SNN. For the small training set, DNNs and SJAE-Map performed similarly, but all of them performed significantly better than SNN. FS did not perform well due to concatenation issues.

4.6. VC Challenge Evaluation
The submission to the VC challenge 2016 was a FS-based approach, as described in Section 3. In this experiment, 25 speaker conversion pairs were considered. For speech quality, an overall MOS score of 2.6 with a standard deviation of 1.05 was achieved for 1600 data points. For speaker similarity, an overall similarity score of 53.33% for 600 data points was achieved. The results are, to some extent, similar to what we achieved with AMT subjects, and four conversion pairs.

5. Conclusion
In this study, we proposed a novel method for training a regression function and applying it to a VC task. We constructed a SJAE, where instead of directly training from source to target features using back-propagation, we imposed a similarity constraint that was imposed on the hidden layers between the source and target AEs. The regression function, SJAE-Map was derived from this SJAE. We demonstrated that our proposed approach generated features that did not suffer from the averaging effect inherent in back-propagation training of DNNs. We also carried out subjective listening experiments to evaluate speech quality and speaker similarity. Our results showed that the SJAE-Map approach had both higher quality and similarity than a SAE+DNN approach, where the SJAE was used for pre-training a DNN, and the fine-tuned DNN is then used for mapping. This demonstrated that minimizing the mean squared error or cross-entropy cost function that is commonly applied on the predicted target features is prone to cause averaging effects; alternative cost functions or architectures (such as SJAE-Map proposed in this study) might help combat that effect in neural network-based VC.
6. References


