Emotion Recognition in Speech with Deep Learning Architectures

Mehmet Erdal, Markus Kächele(✉), and Friedhelm Schwenker

Institute of Neural Information Processing, Ulm University,
James-Franck-Ring, 89081 Ulm, Germany
markus.kaechele@uni-ulm.de

Abstract. Deep neural networks (DNNs) became very popular for learning abstract high-level representations from raw data. This lead to improvements in several classification tasks including emotion recognition in speech. Besides the use as feature learner a DNN can also be used as classifier. In any case it is a challenge to determine the number of hidden layers and neurons in each layer for such networks. In this work the architecture of a DNN is determined by a restricted grid-search with the aim to recognize emotion in human speech. Because speech signals are essentially time series the data will be transformed in an appropriate format to use it as input for deep feed forward neural networks without losing much time dependent information. Furthermore the Elman-Net will be examined. The results shows that by maintaining time dependent information in the data better classification accuracies can be achieved with deep architectures.

1 Introduction

Paralinguistic information like the intonation are important parts in a conversation. We can consider these kinds of information as the semantics of a spoken utterance. For example, the word "yes" is basically an expression of agreement, but with a contemptuous intonation it can mean exactly the opposite namely rejection and this can be an evidence that the speaker is angry. Hence it is possible to perceive the emotional state of the speaker with paralinguistic information conveyed in the speech signal. Because emotions could be crucial for the interpretation of a spoken utterance, efforts are made to give computers the ability to recognize emotion in speech to improve the human-computer interaction (cf. [15]). Nowadays this is a growing field of research which is known as *affective computing*. Therefore the aim of speech emotion recognition is to identify the high-level affective state of an utterance from the low-level features. The task here is to recognize specific pattern as sequences in the speech signal and to categorize them into several classes of emotions.

There are several machine learning models that can be used for classification. In machine learning theory a model is an algorithm which learns from data to tackle a specific task without having to have been explicitly programmed. The learning process is often called training. One of those models are artificial

DOI: 10.1007/978-3-319-46182-3_25

neural networks (ANN), which are slightly inspired by the functioning of the human brain. A deep neural network is an ANN with many layers of nonlinear processing units. The field of research that studies methods to train ANNs with deep architectures is called *deep learning*. Deep learning architectures (DLAs) have been shown to exceed preliminary state-of-the art results in several tasks including emotion recognition in speech [1–3].

2 Related Work

For a long time, DNNs were considered to be hard to train, because gradient-based learning algorithms with a random start initialization produced often poor solutions. Therefore an unsupervised greedy layer-wise learning algorithm for so called deep belief networks (DBNs) is proposed in [4]. A DBN is a composition of simple learning modules which have a layer of visible units to capture the input data and a layer of stochastic hidden units that learn to represent high-level abstractions of the input data. A so trained DBN serves as a start initialization by transforming it into a DNN. The DNN is then trained via backpropagation (bpp) to find a better solution than a randomly initialized network. The training algorithm for the DBN is therefore called pre-training and the phase with bpp is called fine-tuning.

In the work [5] a DBN is used to classify emotion in speech. Like in the present work the Berlin Database of Emotional Speech (EmoDB) with the Mel frequency cepstral coefficients (MFCCs) as speech features is used in the experiments. A multilayer perceptron (MLP) with one hidden layer served as model with a shallow network structure and as baseline. The best result of 60.32 % accuracy is obtained by the DBN in a speaker independent scheme. That was an improvement of 8.67 % over the baseline. In [6] a DBN is used to extract characteristic features for emotional expressions in the speech signal. An SVM is then used as classifier for the extracted features. In [1] a MLP with more than one hidden layer is used as DLA to recognize emotion in speech. Similar to the presented work the number of hidden layers and units are selected via cross validation. However the MLP is there used as feature-extractor and an extreme learning machine is then used to classify the data. In [17] a convolutional deep belief network (CDBN) was used for feature learning from audio data. The so generated features performed often better then MFCCs on several audio classification tasks. Using a more diverse set of features (i.e. including voice quality), the accuracy on the EmoDB dataset can be improved to over 88 % [23].

3 Deep Learning Architectures and Recurrent Neural Networks

In the present work DNNs are used as classifier to recognize emotion in speech. DNNs are not only standalone models, but also the basis of other deep learning models. For example, the deep RNN that is also used in this work, can be viewed as a special form of a DNN. In this section the basics of DNNs and RNNs are explained. Furthermore it will be shown how they can be used as classifiers.

3.1 Deep Neural Networks

Deep learning is essentially a method to approximate a parametric function via neural networks with many hidden layers. For this purpose a neural network represents the function $f(\boldsymbol{x};\theta)$ where $\boldsymbol{x}$ is the input vector and θ a set of parameters. To be more precise f is a composition of functions. Therefore the smallest unit of a neural network is a so called neuron. It maps the weighted sum $\sum_{i=1}^{k} x_i w_i$ to an activation value via the function $f_{act}(\boldsymbol{x}^T\boldsymbol{w})$ where $\boldsymbol{x}$ is the vector of inputs for the neuron and $\boldsymbol{w}$ a vector of parameters denoted as weights. A layer of the network is a set of neurons that usually uses the same activation function. In this case a layer can be represented as function $f^{(i)}(\boldsymbol{x};W^{(i)}) = f_{act}({W^{(i)}}^T\boldsymbol{x})$ where $\boldsymbol{x}$ is the input vector of the layer, $W \in \mathbb{R}^{k \times l}$ the matrix that contains the weights of l neurons (each column of W represents the weights of a neuron) and i is the number of the layer. Putting all together such a neural network represents a composition of the layer functions with the parameters $\theta = \{W^{(1)}, W^{(2)}, \ldots, W^{(n)}\}$ (cf. [9]):

$$f(\boldsymbol{x};\theta) := f^{(n)}(\ldots f^{(2)}(f^{(1)}(f^{(0)}(\boldsymbol{x});W^{(1)});W^{(2)})\ldots;W^{(n)}) \tag{1}$$

As it can be seen, the output of a layer is the input of the next layer. This can be considered as forward propagation of the input through the network. Hence one speaks of a feed forward neural network. The first layer is called input layer and is only there to receive the input with the identity function $f^{(0)}(\boldsymbol{x}) = \boldsymbol{x}$. The last layer $f^{(n)}$ is called output layer. All other layers are called hidden layers. A feed forward neural network with more than one hidden layer is called deep neural network. A neural network can be modeled as acyclic directed graph as shown in Fig. 1.

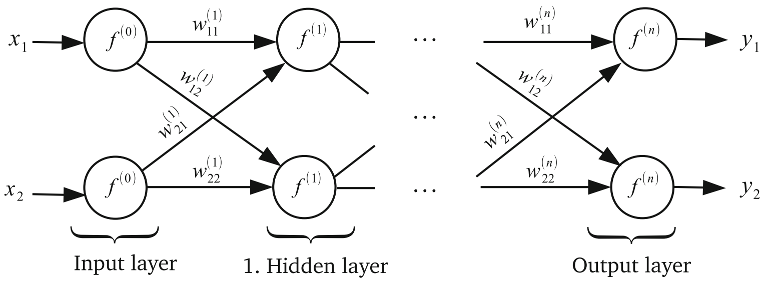


Fig. 1. DNN as directed acyclic graph. For simplification all layers have only two neurons.

It has been shown that it is very effective for DNNs to use the rectified linear function $relu(\boldsymbol{x}^T\boldsymbol{w}) = max(\boldsymbol{x}^T\boldsymbol{w}, 0)$ as activation for the hidden layers [7,8]. The activation function of the output layer depends on the task to tackle. To approximate a specific function f^* the backpropagation-algorithm is used to learn the parameters θ that result in the ideal case in the best approximation $\mathring{f}(\boldsymbol{x};\theta) \approx f^*(\boldsymbol{x})$. This algorithm is based on stochastic gradient descent and computes recursively the gradients beginning with the last layer back to the first hidden layer. The gradients are then used to update the parameters. For this purpose a training set of examples $X_{train} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i^m$ is iteratively processed by the bbp-algorithm. A detailed description of the bbp-Algorithm can be found in [22].

3.2 Dropout Regularization

DNNs have a high number of adjustable parameters. Such powerful models tend to memorize the training set which is called overfitting. This phenomena is also well known for other classification models and hence there are several regularization techniques to overcome this problem. For DNNs the so called dropout regularization has been proven to be an effective tool [18]. The key idea is to deactivate a certain percentage of neurons. This can be performed as follows for all layers $i = 2, 3 \ldots, l$:

$$\boldsymbol{x}^{(i+1)} = f^{(i)}({W^{(i)}}^T(\boldsymbol{x}^{(i)} * \boldsymbol{r}^{(i)})) \quad \text{with} \quad \boldsymbol{r}^{(i)} \sim Bernoullli(\rho) \tag{2}$$

$\boldsymbol{r}^{(i)}$ is a vector of Bernoulli distributed random variables which have the value 1 with probability ρ. Such a sampled vector is element-wise multiplied with the result of the previous layer $x^{(i)} = f^{(i-1)}(\ldots)$. Therefore a 0 in this vector deactivates one respective neuron. This constitutes noise in the training data that makes it more difficult to memorize the training set.

3.3 Deep Recurrent Neural Networks

A drawback of the DNN is that it cannot take into account the inputs from the past of a time series, because it has no memory. Hence it is principally not suitable to process temporal data like time series. A better model for this purpose is the Elman-Network, a recurrent neural network invented by Elman [10]. It has only one hidden layer and parallel to the input layer a so called context layer that is fully connected to the hidden layer which serves as additional input. At timestep t the context layer contains the output of the hidden layer at timestep $t-1$. The RNN represents following recursive function:

$$h(t) = f_h(W^T x(t) + C^T h(t-1)) \tag{3}$$
$$y(t) = f_o(V^T h(t)) \tag{4}$$

Here $x(t)$ is input vector, $h(t)$ the output vector of the hidden layer and $y(t)$ the output of the whole network at time t. W, C, V are the weight matrices for input to hidden layer, context to hidden layer and hidden to output layer. f_h and f_o are the activation function for the hidden layer and output layer. Figure 2 illustrates the recursive flow of $h(t)$ between the context and hidden layer.

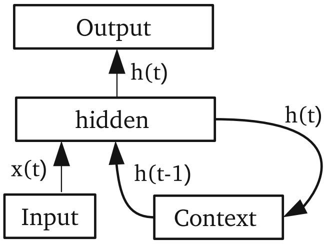


Fig. 2. Structure of a RNN.

After n time steps a DNN with n hidden layers is computed. In order to demonstrate this we concatenate W and C to one weight matrix U. To concatenate two vectors $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{h} \in \mathbb{R}^m$ we write $[\boldsymbol{x}; \boldsymbol{h}]$ with the vector $(x_1, \ldots, x_n, h_{n+1}, \ldots, h_{n+m})$ as result. For a easier understanding we consider only 3 time steps. Now we can write the hidden activation function as follows:

$$h(3) = f_h^{(3)}(U^T[x(3);\ \underbrace{f_h^{(2)}(U^T[x(2);\ f_h^{(1)}(U^T[x(1);\ f_h^{(0)}])])}_{\text{hidden layers of the previous 2 time steps}}]) \tag{5}$$

$f_h^{(t)}$ is here the hidden activation function at time step t and $f_h^{(0)}$ is an initial null vector. At time step $t = 3$ the additional input is $f_h^{(2)}$ which computes two hidden layers with the previous inputs $x(2)$ and $x(1)$. In this way the temporal data of the past is taken into account. With the hidden layer of the current time step and the output layer f_o a DNN with overall 3 hidden layers is computed. To train such a network a slightly modified bpp-algorithm is used called backpropagation through time [11].

3.4 Deep Neural Networks as Classifier

In general a classifier is a function $y = f^*(\boldsymbol{x})$ that maps an n-dimensional input $\boldsymbol{x}$ to a category y. To us a neural network for multinomial classification the softmax function is often used in the output layer. Each neuron represents one category $y \in \{c_1, \ldots, c_n\}$ and with the softmax function a probability for each category is predicted as follows:

$$P(y = c_i|\boldsymbol{x}) = \frac{e^{\boldsymbol{x}^T \boldsymbol{w}_i}}{\sum_{j=1}^{n} e^{\boldsymbol{x}^T \boldsymbol{w}_j}} \tag{6}$$

Consequently a neural network classifier $\boldsymbol{p} = f(\boldsymbol{x}; \theta)$ maps an input $\boldsymbol{x}$ to a vector of probabilities $\boldsymbol{p}$. The element $p_i \in \boldsymbol{p}$ with the highest probability stands for the predicted class k_i. To asses the performance of a classifier the classification accuracy with a test set $X_{test} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i^m$ with $X_{train} \cap X_{test} = \emptyset$ can be used as cost function:

$$accuracy := \frac{1}{m} \sum_{i=1}^{m} L(k_i, c_i) \quad \text{with} \quad L(k_i, c_i) = \begin{cases} 1 & \text{if } c_i = k_i \\ 0 & \text{else} \end{cases} \tag{7}$$

4 Feature Extraction and Preprocessing

The speech signal is provided as time series of amplitudes which is the result of an analog digital conversion. A time series is a sequence of scalars or vectors which depends on time. In case of the raw digital speech signal $\{x[t]\}$ the elements are scalars:

$$\{x[t]\} = \{x(t_0), x(t_1), \dots, x(t_{i-1}), x(t_i), x(t_{i+1}), \dots\} \tag{8}$$

The raw speech signal contains to much unnecessary information that is an obstacle to recognize certain emotion pattern. Hence a crucial step towards emotion recognition in speech is to extract appropriate features from the speech signal. Furthermore feature extraction reduces the high dimensionality of the raw speech signal. In several studies it has been shown that the Mel-Frequency Cepstral Coefficients (MFCCs) are very useful features to recognize emotion in speech [12]. One reason for that could be that MFCC mimics some parts of the human speech production and speech perception. In the presented work the MFCC-Algorithm is treated as black box which delivers very useful features. The MFCC-algorithm divides the speech signal into overlapping frames with a window of sufficient size win_{mfcc} to compute the features. This window is shifted with a time step over the signal. The computed features for each frame are vectors. The result of the MFCC-algorithm is therefore a time series $\{\boldsymbol{v}[t]\}$ of vectors with time step δt:

$$\{\boldsymbol{v}[t]\} = \{\boldsymbol{v}(0), \boldsymbol{v}(\delta t), \boldsymbol{v}(2\delta t), \dots, \boldsymbol{v}(n\delta t)\} \tag{9}$$

Here n is the number of frames. The vector $\boldsymbol{v}(i) = (v_1, \dots, v_m)$ contains the m extracted features of the i-th frame. This time series can be represented as feature matrix $V \in \mathbb{R}^{n \times m}$ whereby the i-th row is the i-th element of the time series. A typical value for δt is 10 ms and since utterances have usually a duration of a few seconds, that results in feature matrices with several hundred rows. One simple approach to avoid this problem is to reduce the feature matrix to a vector $\overline{\boldsymbol{v}}$ by computing the mean of all rows of V (cf. [13]):

$$\overline{\boldsymbol{v}} = \frac{1}{n}\left(\textstyle\sum_{i=0}^{n} v_{i1}, \dots, \sum_{i=0}^{n} v_{im}\right) \tag{10}$$

But thereby the number of data points are the number of utterances and this is a sub-optimal solution, because one of the main problems of speech-based emotion recognition systems are the small number of available patterns [14]. Another problem is, that by building the mean of the time series much time dependent information are lost. Since emotions are generally expressed over time in an utterance, this could make it more difficult to recognize the emotion. Therefore we compute the mean over each $1 < l < n$ rows and get a new feature matrix $U \in \mathbb{R}^{\lfloor \frac{n}{l} \rfloor \times m}$ whose rows build a new time series $\{\boldsymbol{u}[t]\}$ with time step l. Then each element $\boldsymbol{u}(t_i)$ represents a section of $win_{\text{mfcc}} \cdot l$ ms of the origin speech signal $\{x[t]\}$.

5 Classification of Time Series

5.1 Windowing for Deep Feed Forward Neural Networks

Unlike RNNs feed forward neural networks are not constructed to process sequential data like time series. One method to take time distributed information into account is to window the time series before using it as input for the neural network. This is achieved by sliding a window win_{i} of size k over several steps along the times series $\{\boldsymbol{u}[t]\}$. In this way a section of k elements of $\{\boldsymbol{u}[t]\}$ are concatenated to a new vector $\boldsymbol{x}_{i-1} = [\boldsymbol{u}(t_{(i-1)s}); \ldots ; \boldsymbol{u}(t_{k-1})]$ whereby i is the number of the window and s the step size.

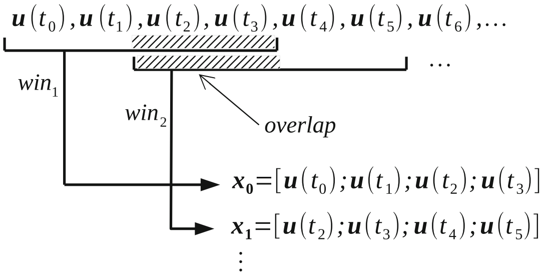

$\boldsymbol{u}(t_0), \boldsymbol{u}(t_1), \boldsymbol{u}(t_2), \boldsymbol{u}(t_3), \boldsymbol{u}(t_4), \boldsymbol{u}(t_5), \boldsymbol{u}(t_6), \ldots$
win_1
win_2
overlap
$\mathbf{x_0}=[\boldsymbol{u}(t_0);\boldsymbol{u}(t_1);\boldsymbol{u}(t_2);\boldsymbol{u}(t_3)]$
$\mathbf{x_1}=[\boldsymbol{u}(t_2);\boldsymbol{u}(t_3);\boldsymbol{u}(t_4);\boldsymbol{u}(t_5)]$

Fig. 3. Two steps of windowing the time series $\{\boldsymbol{u}[t]\}$ with a window size of $k = 4$ and step size of $s = 2$.

As it can be seen in Fig. 3 if $s < k$ an overlap of $k - s$ elements in the newly generated data $\boldsymbol{x}_i$ is created. By this overlap the neural network receives some information more than once.

5.2 Voting Based Classification

To classify the utterance $\{\boldsymbol{u}[t]\}$ with the label y_j a majority voting scheme can bee used to predict a class c_l based on the associated windowed data set $\{(\boldsymbol{x}_1, y_j), \ldots (\boldsymbol{x}_m, y_j)\}$ as follows:

$$c_l = \arg\, \max_{l=1\ldots N} V_{c_l} \tag{11}$$

V_{c_l} are the votes for class c_l and determined for all available classes $l = 1 \ldots N$ with:

$$V_{c_l} = \sum_{i=0}^{m} G(\boldsymbol{x}_i, c_l) \quad \text{with} \quad G(x_i, c_l) = \begin{cases} 1 & \text{if } f(\boldsymbol{x}_i, \theta) \text{ predicted } c_l \\ 0 & \text{else} \end{cases} \tag{12}$$

6 Datasets

The used data is a significant component to asses an emotion recognition system. In this paper the widely used Berlin Database of Emotional Speech (EmoDB) serves as dataset [16]. The EmoDB consists of professional audio-recordings of seven acted emotions (anger, boredom, disgust, fear, happiness, sadness, neutral) spoken by 10 different actors in 10 different sentences. Overall 535 recorded and labeled utterances are provided. As mentioned above we used MFCCs as speech features. This was done by using window size of $win_{\mathrm{mfcc}} = 25\,\mathrm{ms}$ and a time step of $\delta t = 10\,\mathrm{ms}$ to extract the first twelve cepstral coefficients with the energy as 13-th feature per window. Furthermore the first and second order derivatives of the cepstral coefficients are computed for each feature vector $\boldsymbol{v}(t) = (v_1, \ldots, v_{13})$ for $i = 1, \ldots, 12$:

$$\Delta v_i = \frac{1}{2}(v_{i+1} - v_{i-1}) \tag{13}$$

$$\Delta\Delta v_i = \frac{1}{2}(\Delta v_{i+1} - \Delta v_{i-1}) \tag{14}$$

v_i is here the i-th element in the feature vector. As additionally statistical feature the standard deviation σ_j was computed for each column $j = 1, \ldots, 37$ of the feature matrix $M \in \mathbb{R}^{n \times 37}$ as follows:

$$\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_{ij} - \overline{v}_{m,j})^2} \tag{15}$$

Here $\overline{v}_{m,j}$ is the j-th element of the m-th mean vector over every l rows of the feature matrix with $m = 1, \ldots, \lfloor \frac{n}{l} \rfloor$. Hence for each utterance a feature matrix $U \in \mathbb{R}^{\lfloor \frac{n}{l} \rfloor \times 74}$ was computed whereby n is the number of feature vectors of the respective utterance. With this procedure we created two types of datasets D_c and D_p. D_c contains the means of each feature matrix and therefor consists of 535 feature vectors $\boldsymbol{x}_i \in \mathbb{R}^{74}$. For D_p we computed the means of each feature matrix partially over $l = 15$ rows as described in Sect. 4. After windowing the data with a window size of $k = 7$ and step size of $s = 1$ as described in Sect. 5 D_p contains 6915 features vectors $\boldsymbol{x}_i \in \mathbb{R}^{518}$.

6.1 Testing the Datasets with a Support Vector Machine

Because erroneous data can lead to wrong conclusion, we tested the generated data sets with a support vector machine with a radial basis kernel (SVM-RBF). Hence for a given training set $X \in \mathbb{R}^{n \times 74} \subset D$ the following classifier has to be learned:

$$f(\boldsymbol{x}) = \sum_{i}^{n} \alpha_i y_i exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}_i\|_2) + b \tag{16}$$

The classifier is learned by solving the following optimization problem:

$$\text{minimize: } \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{17}$$

$$\text{subject to: } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n \tag{18}$$

The tests were performed in a speaker independent scheme with leave-one-speaker-out cross-validation (LOSO) and a text independent scheme with leave-one-text-out (LOTO).

Table 1. Classification accuracies for SVM-RBF

Dataset	LOSO accuracy (avg)	LOTO accuracy (avg)
D_c	59.0 %	78.9 %
D_p	65.5 %	79.0 %

To optimize the parameters C and γ of the SVM a grid-search was performed with $C = \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, \ldots, 10^{9}\}$ and $\gamma = \{10^{-5}, 10^{-4}, \ldots, 10^{4}\}$. Table 1 shows that with the windowed dataset D_p of partially feature-means an improvement of 6.5 % classification accuracy could be achieved for the LOSO. In contrast to that, for LOTO the difference between D_c and D_p is negligibly small. Since the SVM can be considered as model with a flat architecture, these results suggest that more time dependent information can lead to better results only for a speaker independent scheme, even without a deep architecture.

7 Results

In this section we describe the results of our experiments. Like in the work of Albornoz et al. [5] we evaluated the models with two kinds of cross validation. Once with leave-one-speaker-out (LOSO) and once with leave-one-text-out (LOTO). LOSO is possibly the severest challenge for a speech based emotion recognition system, because the model has to abstract from the speaker dependent properties in the speech signal. To solve this very non-linear classification problem a very complex function has to be learned. And this is exactly where deep architectures could help [20]. For LOTO a much lesser complex relation between input data and desired output has to be learned. In this case deep learning methods could provide a less benefit [5].

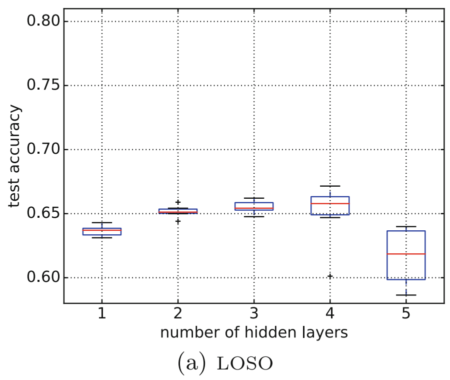
We investigated several neural network architectures which differ from the number of hidden layers and hidden units. In order to facilitate the examination all architectures shared the same setting of the hyper-parameters *learn rate* = 0.3, *learn rate decay* = 0.09, *momentum* = 0.9, *weight decay* = 0.007, *dropout* = 0.09, *mini batch size* = 64 and *epochs* = 55. To train the networks we used a variant of the backpropagation-algorithm based on Nesterovs Accelerated

Gradient Descent (cf. [19]). The weights for a given layer are initialized with random values $\in (-1, 1)$. We denote an architecture as $\mathrm{DNN}_{\mathrm{i}\times\mathrm{j}}$ whereby i is the number of hidden layers and j the number of hidden neurons. We first used a shallow neural network $\mathrm{DNN}_{1\times 80}$ with one hidden layer and 80 hidden units as baseline by using the smaller Dataset $D_c \in \mathbb{R}^{535\times 74}$. In the LOSO-scheme the $\mathrm{DNN}_{1\times 80}$ classified with 58.5 % accuracy (avg). With LOTO a substantially higher accuracy (avg) of 76.6 % is achieved.

7.1 Experiments with DNNs

In order to examine if an improvement can be obtained with more hidden layers in combination by using the larger dataset $D_p \in \mathbb{R}^{6195\times 518}$, we investigated the following set of architectures:

$$\{\mathrm{DNN}_{\mathrm{i}\times\mathrm{j}} \mid i \in \{1, 2, \ldots, 5\},\ j \in \{100, 200, \ldots, 600\}\}$$



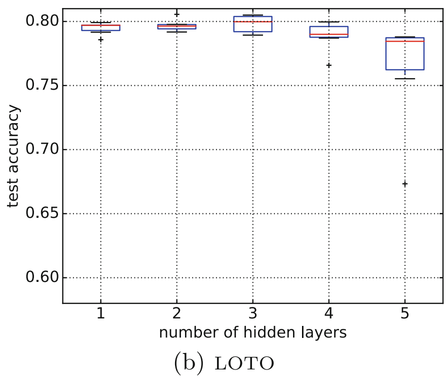
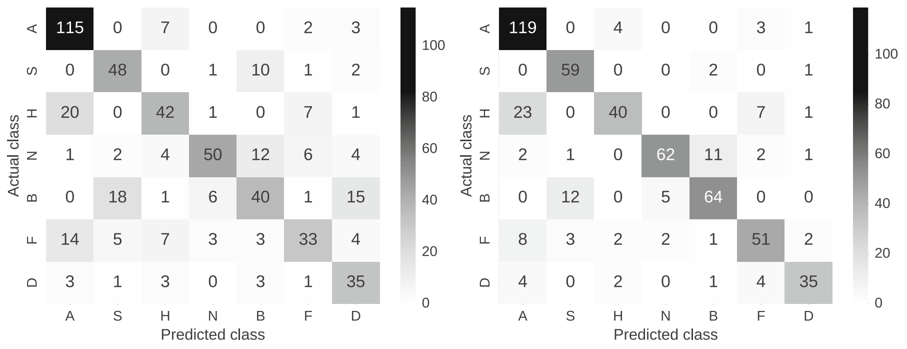


Fig. 4. Effect of adding more hidden layers to a DNN on classification accuracy with LOSO (a) and LOTO (b). Box plots shows the distribution of accuracies associated with 6 different numbers j of hidden neurons per layer $j \in \{100, 200, \ldots, 600\}$.

Figure 4(a) shows that for LOSO additional hidden layers improved the classification accuracy (avg) until the 4th hidden layer. However with the 5th hidden layer the accuracy gets worse. Maybe the DNNs are then to big and need more data to prevent overfitting. The best result of 67.2 % with was achieved by the network $\mathrm{DNN}_{4\times 500}$ with 4 hidden layers and 500 neurons per layer. The confusion matrix in Fig. 5(a) shows that the best classification performance is significantly obtained whit the emotion angry. One reason for that could be the far larger number of samples of this class. Another reason could be the high energy in the speech signal of angrily spoken utterances. As assumed above and also observed in [5] with LOTO additional hidden layers did not improve the accuracy significantly as Fig. 4(b) shows. The highest result of 80.6 % is here achieved by the network $\mathrm{DNN}_{2\times 600}$ with 2 hidden layers and 600 hidden neurons per layer.



(a) LOSO with $\text{DNN}_{4\times 500}$: 67.2 % (b) LOTO with $\text{DNN}_{2\times 600}$: 80.6 %

Fig. 5. Classification performance of the best DNNs of the individual classes illustrated as confusion matrix with A = Anger, B = Boredom, S = Sadness, H = Happiness, D = Disgust, F = Fear, N = Neutral.

The comparison of the two confusion matrices in Fig. 5 shows that the improvement with LOTO concerned mainly only the emotions sadness, neutral, boredom and fear. It could therefore be assumed that for the other emotions anger, disgust and happiness the speaker dependent properties have a reduced role for speech based emotion recognition.

7.2 Experiments with RNNs

The result of 67.2 % with the $\text{DNN}_{4\times 500}$ is only slightly better than the result of 65.5 % with the SVM-RBF. There is a suspicion that the windowed dataset D_p is the main reason for the good results in both cases. As described above we created D_p to maintain time dependent information. Hence we investigated now Elman-Networks which are constructed to process time series. The depth of a RNN depends on the length of the time series to process. To vary the number of hidden layers we created therefor several datasets by computing the means of each features matrix over several number l of rows as described in Sect. 4. Table 2 shows the smaller the averaged sections of the initial feature matrix, the longer the resulting time series and thus the more hidden layer in a respective unfolded RNN. The RNNs are trained via backpropagation through time (bptt) with the hyper-parameters *learn rate* = 0.004, *momentum* = 0.9 and *epochs* = 200. To asses the essential performance of the RNN for speech based emotion recognition no other optimization or regularization techniques are used. However to deal with the unbounded rectified linear function (cf. [21]) in the hidden layer we clipped the outputs $y = relu(x)$ of the hidden neurons as follows:

$$y = \begin{cases} y & \text{if} \quad y \le x_{\max} \\ x_{\max} & \text{else} \quad y > x_{\max} \end{cases} \tag{19}$$

In some way $x_{\max}$ can also be viewed as hyper-parameters and we chose the value $x_{\max} = 40$. We performed the tests again with both validation schemes LOSO and LOTO.

Table 2. Relationship between averaging and length of resulting time series

Avg. over l rows	$\approx$ Length of time series
30	10
60	5
90	4
120	3
150	2
180	2

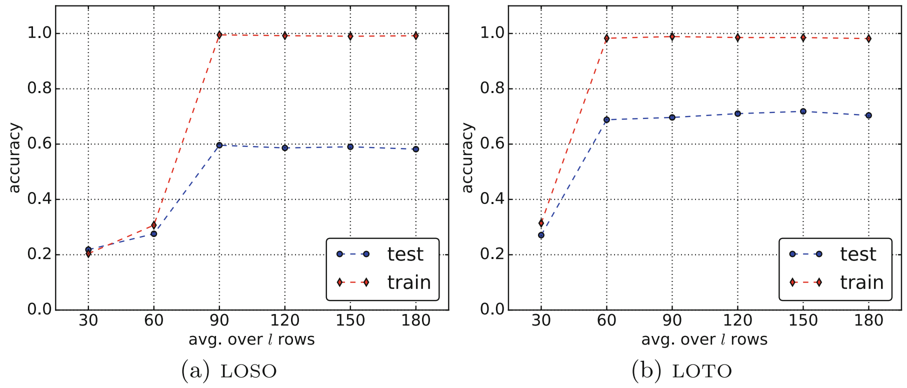


Fig. 6. Effect of using longer time series as input for a RNN on classification accuracy (test, train) with LOSO (a) and LOTO (b). The X axis shows the respective number l of averaged rows of the feature matrices.

Interestingly as with the DNN-experiments the highest classification accuracy (here: 59.5 %) with LOSO is again obtained by using a network with 4 unfolded hidden layers ($l = 90$) as it can be seen in Fig. 6(a). In view of the fact that only a basic version of the bptt-algorithm is used the result of 59.5 % accuracy with LOSO is highly promising. With LOTO the highest accuracy of 71.8 % is achieved by using the network with 2 hidden layers ($l = 150$). This suggests again that for LOTO deeper architectures may not improve the result significantly.

8 Conclusion

The performed experiments with the partially averaged and windowed time series suggest adding more hidden layers could improve the performance of feed forward neural networks for recognizing emotion in speech. The performance of the support vector machine also improved considerably when using the partially averaged feature matrices whereby the classification accuracy is only 1.7 % worse than with the DNN. Hence there is a suspicion that the method of preprocessing that we were using is the main reason for the improved results. To assess this more precisely more experiments have to be conducted.

Highly promising are the results with the RNNs, because only with a basic version of the backpropagation through time algorithm a accuracy of about 60 % is obtained. It would be interestingly to investigate if a significantly better performance can be obtained with several regularization and optimization techniques.

Acknowledgment. This paper is based on work done within the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (DFG). Markus Kächele is supported by a scholarship of the Landesgraduiertenförderung Baden-Württemberg at Ulm University. The work is furthermore supported by the *SenseEmotion* project funded by the German Ministry of Science, Research and Arts.

References

1. Kun, H., Dong, Y., Ivan, T.: Speech emotion recognition using deep neural network and extreme learning machine. In: 15th Annual Conference of the International Speech Communication Association, ISCA, Singapore, pp. 223–227 (2014)
2. Trigeorgis, G., Ringeval, F., Brueckner, R., Marchi, E., Nicoalou, A.M., Zafeiriou, S.: Adieu features? End-to-end speech emotion recognition using a deep convolutional recurrent network. In: 41st IEEE International Conference on Accoustics, Speech and Signal Processing, ICASSP, Shanghai, pp. 5200–5204 (2016)
3. Kim, Y., Lee, H., Provost, E.M.: Deep learning for robust feature generation in audiovisual emotion recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, Vancouver, pp. 3687–3691 (2013)
4. Hinton, G.E., Osinderos, S., The, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**, 1527–1554 (2006). MIT Press, Cambridge
5. Albornoz, E.M., Sánchez-Gutiérrez, M., Martinez-Licona, F., Rufiner, H.L., Goddard, J.: Spoken emotion recognition using deep learning. In: Bayro-Corrochano, E., Hancock, E. (eds.) CIARP 2014. LNCS, vol. 8827, pp. 104–111. Springer, Heidelberg (2014)
6. Huang, C., Gong, W., Fu, W.: A research of speech emotion recognition based on deep belief network and SVM. Math. Probl. Eng. **2014**, 1–7 (2014). Beijing, Article ID 749604
7. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the 14th International Conference of Artificial Intelligence and Statistics, JMLR Proceedings, Fort Lauderdale, pp. 315–323 (2011)

8. Maas, A., Hannun, A., Ng, A.: Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning for Audio Speech, and Language Processing, JMLR, Atlanta (2013)
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016, in preparation). http://www.deeplearningbook.org
10. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**, 179–211 (1990). Wiley
11. Werbos, P.: Backpropagation through time: what does it do and how to do it. Proc. IEEE **78**, 1550–1560 (1990)
12. Koolgaudi, S.G., Rao, K.S.: Emotion recognition from speech: a review. Int. J. Speech Technol. **15**, 99–117 (2012). Springer
13. Ma, Z., Fokoué, E.: A comparison of classifiers in performing speaker accent recognition using MFCCs. Open J. Stat. **4**, 258–266 (2014). Scientific Research Publishing Inc
14. Mohino-Herranz, I., Gil-Pita, R., Alonso-Diaz, S., Rosa-Zurera, M.: MFCC based enlargement of the training set for emotion recognition in speech. Signal Image Process. Int. J. **5** (2014)
15. Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., Taylor, J.: Emotion recognition in human-computer interaction. IEEE Signal Process. Mag. **18**, 32–80 (2001). IEEE
16. Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W., Weiss, B.: A database of German emotional speech. In: Proceedings of Interspeech, Lissabon, pp. 1517–1520 (2005)
17. Lee, H., Pham, P., Largman, Y., Ng, A.Y.: Unsupervised feature learning for audio classification using convolutional deep belief networks. In: Advances in Neural Information Processing Systems, vol. 22, pp. 1096–1104. Curran Associates Inc., Vancouver (2009)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105. Curran Associates Inc., Nevada (2012)
19. Sutskever, I., Martens, J., Dahl, G.E., Hinton, G.E.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning, Atlanta, pp. 1139–1147 (2013)
20. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. J. Mach. Learn. Res. **10**, 1532–4435 (2009). JMLR.org
21. Le, Q., Jaitly, N., Hinton, G.E.: A simple way to initialize recurrent networks of rectified linear units. CoRR (2015)
22. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backprop. In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, pp. 9–50. Springer, Heidelberg (1998)
23. Kächele, M., Zharkov, D., Meudt, S., Schwenker, F.: Prosodic, spectral and voice quality feature selection using a long-term stopping criterion for audio-based emotion recognition. In: Proceedings of the International Conference on Pattern Recognition (ICPR), pp. 803–808 (2014)