# BiDirectional Neural Network For Facial Emotion Classification in SFEW Database[2]

Jiaxu LIU

Australian National University, Canberra ACT 0200, AU
u6920122@anu.edu.au

**Abstract.** The SFEW database contains unconstrained facial expressions close to real world. In former research, current machine learning techniques are not robust enough for this uncontrolled environment[2]. In accordance with the effectiveness and limitation of BP network in classification problem, we conduct an experiment of applying BiDirectional Neural Network on classification by training the network simultaneously in positive and negative propagate direction. In section 2 the method is explained, and in section 3 the classification result is presented, compared, and analysed. Further more, the experiment is concluded and some future works are presented in this paper.

## 1 Introduction

For classification problems, we can apply both traditional machine learning classifiers and deep learning methods including various neural network structures. Generally, these methods perform well on lab-controlled data which means they tend to have high accuracy in prediction. However, expression analysis in (close to) real world situations is a non-trivial task and requires more sophisticated methods at all stages of the approach[2].

Nowadays, deep learning techniques have made significant improvement in classification comparing to traditional classification approaches like SVM or RandomForest. For most cases, we design the model without putting in prior knowledge about the solution toward problems. The SFEW Database is consisting of close to real world facial conditions which made the problem more complex, In our point of view, to simplify the real world problem, we need to reuse all prior knowledge of image constitution, at least in particular, reusing all available input information[4].

To use available input information reversely, one way is to implement two separate networks and merge the output[5], which has been demonstrated to be practicable, while the drawback is that we cannot find optimal merging algorithm since network trained by same dataset should not be consider as independent[5]. Thus in this paper, the bidirectional version of Neural Network is implemented

and experimented on SFEW dataset. Also, the paper will give insight whether bidirectional model could overcome the limitations of unidirectional network in image classification.

## 2 Method

In this section, we will start from introducing the topology of BPNN and how the modified version of BPNN – BDNN is constructed, we will explain the method of training, and how this technique would perform on one-to-one relation dataset classification. Through all the topology graphs, we use one hidden layer for better understanding and less mathematical representation complexity. Notice that it is very easy to extend the one hidden case into multiple layers.

### 2.1 BP Neural Networks

A unidirectional BP neural network is consist of three parts, namely input layer, hidden layer and output layer[6] [fig.1]. When we perform forward propagation, we first input a set of X and compute the neuron values of hidden layers:

$$S_j = \theta_1(\sum W_{ij}^1 X_i) \tag{1}$$

Where S is the activated value of hidden layer, j is the row number of neurons and $\theta$ is the nonlinear activation function adding non-linear property to the layer. Then we compute the values of output layer:

$$\hat{Y}_j = \theta_2(\sum W_{ij}^2 S_i) \tag{2}$$

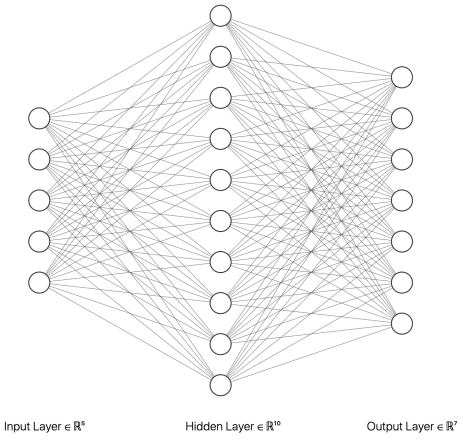



**Fig. 1.** Topology of BP Neural Networks with one hidden layer

### 2.2 Bidirectional Neural Networks

The BDNN could be either implemented independently, or merging two mirror BPNN, the point is that, the weight for each epoch in training the first network(naming 'model1') is reused in training the second reversed network(naming 'model2'), while transposing the weights, and vice versa in training model1 [fig.2].

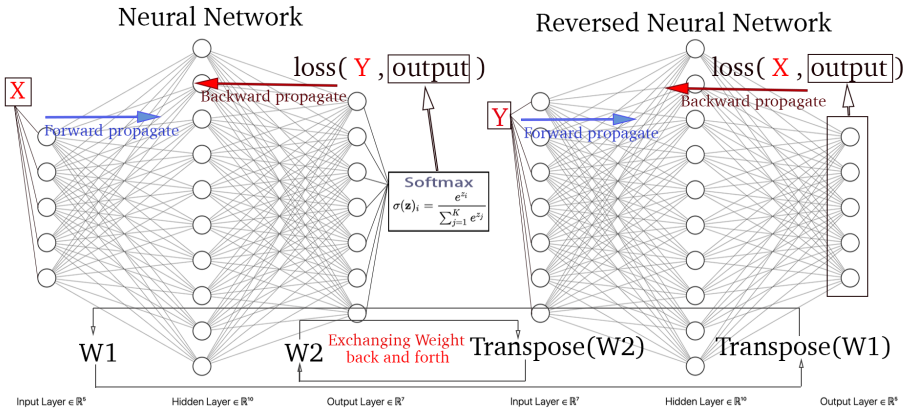



**Fig. 2.** Topology of the bidirectional Neural Networks with one hidden layer

More specifically, assume $X$ as the input data ,$Y$ as the output data we want to predict, and $\hat{Y}$ as the predicted data, same as formula(1,2), we can represent $\hat{Y}$ as

$$\hat{Y}_j = \theta_2(\sum W_{ij}^2 S_i) = \theta_2(\sum_j W_{ij}^2 \theta_1(\sum_k W_{ki}^1 X_k)) \tag{3}$$

$$loss = optimizer(Y, \hat{Y}) \tag{4}$$

After completed one forward&backward propagation in model1, for model2, we load Y as an input, note that for SFEW dataset, the label should be encoded and feed into 7 neurons, the strategy we use is quite simple: if label = i, we feed value "1" to the ith neuron, and the others are all fed with "0". Thus reversely, for $\hat{X}$

$$\hat{X}_j = \theta_2(\sum (W^1)_{ij}^T S_i) = \theta_2(\sum_j (W^1)_{ij}^T \theta_1(\sum_k (W^2)_{ki}^T Y_k)) \tag{5}$$

$$loss = optimizer(X, \hat{X}) \tag{6}$$

After the model2 is trained, we then shift the weights of model2, namely $W_2^T$ and $W_1^T$ to $W_1$ and $W_2$ in model1 respectively. Back and forth, we train the whole model within both input and output in the dataset.

Note that before we train the network, we need to preprocess the data to make the mapping $f(X) = Y$ and $f^{'}(Y) = X$ one-to-one, which means the mapping should be invertible and thus data should be preprocessed in specific way. The preprocessing will be explained in section 3.2.

### 2.3 BiDirectional LSTM(For comparison in experiment only)

LSTM is a modified version of RNN, while introducing 'cell' to handle long sequential memory. The idea of BiLSTM is to add a backward recurrent layer, and these backward layers are connected with each other going backward in time. The topology of model could be demonstrated as

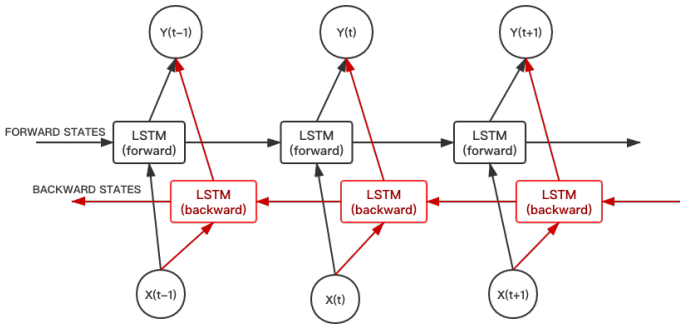



**Fig. 3.** General structure of the bidirectional LSTM shown unfolded in time for three time steps

Notice when we computing network activation of backward states, it is not back-propagation, its actually forward-propagation, which means part of propagation goes from left to right, and part goes from right to left. After computing the forward and backward activation, we could make the prediction at time step t. Also notice that the output is a list of data indicating the weight toward each label, we apply $softmax$ to get probability representation.

## 3 Experiment Results and Discussion

### 3.1 Dataset

SFEW has been developed by selecting frames from AFEW database where AFEW is consisting of facial expressions data extracted from movies.[10]. SFEW database varies widely in the perspective of expressions, head poses, age, focus,

image resolution, and illumination[2]. Totally, the database contains 700 extracted image, and the provided dataset contains 675 entries, with each of them made up of 10 features and 1 label. Five of the features are principal component extracted from LPQ descriptor[11], and the others are extracted from PHOG descriptor[12]. The label ranges from 1 to 7, indicating facial expression from angry to surprise. In the experiment, we train model which receive 10 features(or 5 features for each principal component) and predict the facial expression.

### 3.2 Data preprocess & Training strategy

As its given in the dataset the top 5 principal component, for each time step, we read in one entry of data into the neuron. The input is consist of 5(or 10 for all) extracted features, and the output is a list of probability normalised via $softmax$, indicating the probability toward each facial expression. Meanwhile, to fix the mean and variance of the input in the network layers and avoid the problem of gradient vanishing, we add a batch normalization(BN) to each layer to pull the distribution of the input value of each neuron of the neural network back to the normal distribution with a mean of 0 and a variance of 1.

Also, its mentioned in paper[1], the mapping of data from input to output should be invertible. To augment the invertibility of dataset, we add the mean of input(values of 10 features) to the output as a supervisor of learning(same as the 'extra node' in paper[1]), such statistically, we can regard the combination of label&mean as unique key. i.e. for data row i:

$$ExtraNode_i = \frac{\sum_{j=1}^{5} LPQ_{ij} + \sum_{j=1}^{5} PHOG_{ij}}{10} \tag{7}$$

### 3.3 Comparison of different models

In this section, we will compare the performance of General BPNN, BDNN, and BRNN. The experiments demonstrate the curve of training loss/accuracy of these models for each epoch on classifying seven expression classes, also, the performance of LPQ and PHOG descriptor will be compared.

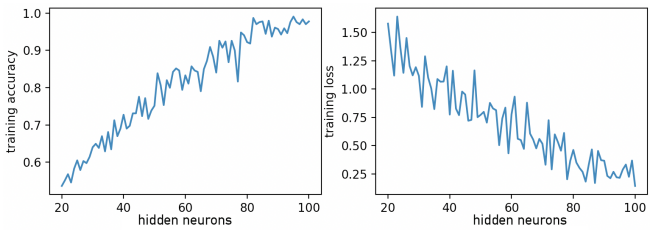



**Fig. 4.** BDNN training accuracy(left)&loss(right) within 20 to 100 hidden neurons

We used a five-fold cross validation script, which creates five subsets of the dataset for evaluation[fig.7(a),(b)]. We do experiment with all model's learning rate=0.02, and we use the batch gradient descent technique with batch size=64. Specifically for BPNN and BDNN, to prevent overfitting, we introduce a dropout layer on output except last layer, with probability equal to 0.1. Nevertheless , the network contains one hidden layer, as its presented in fig.4, we tested the bdnn model on different hidden unit size and find that with 100 epochs, the network start to converge at approximately 90%+ with 70 hidden layer size, thus in the experiment, we fix hidden neuron amount to 70.

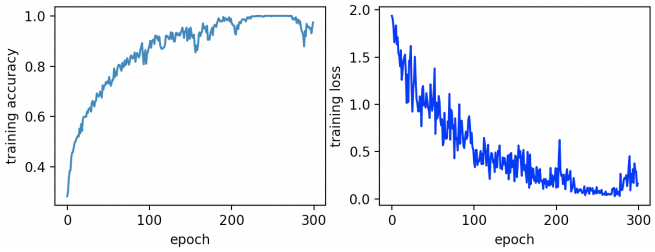



**Fig. 5.** BP network training accuracy(left)&loss(right) variation over 1000 epochs

Comparing BP[fig.5] to BDNN[fig.6], the BDNN converge with less epochs than BPNN and it takes BDNN 200 epochs to reach approximately 100% train accuracy, it somehow shows BDNN could have a better understanding on the dataset. Still, the faster convergence of BDNN could be regard as training the model twice(positively and reversely going) per epoch. Also, the testing accuracy has a little bit improvement, which is from 20.6% (for BPNN) to 26.80% (for BDNN) on average.

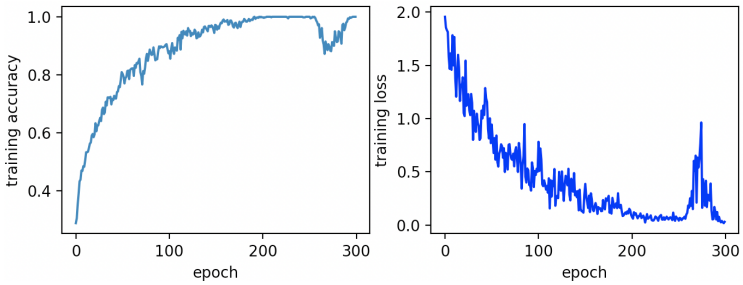
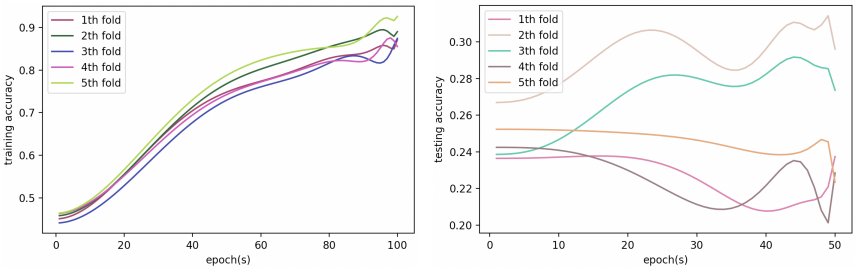



**Fig. 6.** BDNN training accuracy(left)&loss(right) variation over 1000 epochs

However, the testing we conducted differs drastically toward the result provided in paper[2], for a combination of LPQ and PHOG features, we apply

non-linear SVM[7] and random forest[8] classifier onto the data, and gets only 21% and 19% average accuracy respectively. For NN models, the BDNN model performs the best, which could sometimes reach 30% testing accuracy while most of time around 26%, the other two models performs relatively worse, with around 20% and 24% respectively for BPNN and BiLSTM [fig.8]. Still, none of the result meet the ideal(43.71% for LQ and 46.28% for PHOG [2]), which indicates that current NN models are still not robust enough for uncontrolled environment experiments[2].




(a) training accuracy (b) testing accuracy

**Fig. 7.** 5-fold cross validation training&testing accuracy of BDNN

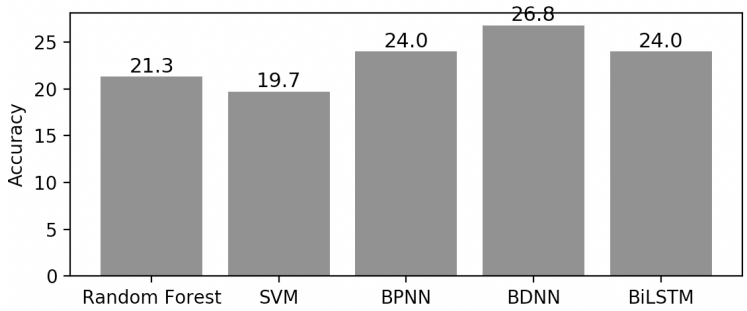



**Fig. 8.** 7 expression class accuracy comparison of SFEW based on LPQ+PHOG descriptors, and different classification methods

### 3.4 SPI Baseline

Based on the SPI protocol, we compute the baseline scores(Table.1). Same as the method in SFEW paper[2], we combined the features of two descriptors for better performance in training. Compared to result in paper[2] using SVM

classifier with RBF kernel, the BDNN method get a significant improvement on precision, recall and specificity.

| **Emotion** | **Angry** | **Disgust** | **Fear** | **Happy** | **Neutral** | **Sad** | **Surprise** |
|---|---|---|---|---|---|---|---|
| **Precision** | 0.27 | 0.083 | 0.24 | 0.35 | 0.21 | 0.24 | 0.39 |
| **Recall** | 0.28 | 0.076 | 0.24 | 0.35 | 0.25 | 0.26 | 0.30 |
| **Specificity** | 0.93 | 0.99 | 0.96 | 0.94 | 0.95 | 0.94 | 0.93 |

**Table 1.** Average expression classwise Precision, Recall and Specifity results

## 4 Conclusion and Future Work

We have shown the topology of BDNN and how can it be trained by a generalised error back-propagation algorithm to provide the capabilities of label classification. We demonstrated the effectiveness of our classifier by evaluating the model on SFEW database. The experiment have somewhat revealed the performance of BDNN compared to other models.

The BDNN model, consisting of two symmetrical BPNNs, learns features and inversely at the same time. In paper [2], principle features in the image are extracted by descriptor. In the future, instead of LPQ and HPOG, if the original image is provided for instance, we plan to combine BDNN with convolutional neural network, applying CNN Cascade method for face detection[9], use pooling technique instead of descriptors, see if it could outperform the general CNN models and contributes an increment to emotion classification accuracy in SFEW.

## References


1. Nejad, A. F., and T. D. Gedeon. "Bidirectional neural networks and class prototypes." Proceedings of ICNN'95-International Conference on Neural Networks. Vol. 3. IEEE, 1995.

2. Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011, November). Static facial expressions in tough conditions: Data, evaluation protocol and benchmark. In 1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011.

3. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

4. Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing 45.11 (1997): 2673-2681.

5. C. L. Giles, G. M. Kuhn, and R. J. Williams, "Dynamic recurrent neural networks: Theory and applications," IEEE Trans. Neural Networks, vol. 5, pp. 153–156, Apr. 1994.

6. Hecht-Nielsen, Robert. "Theory of the backpropagation neural network." Neural networks for perception. Academic Press, 1992. 65-93.

7. C.-C. Chang and C.-J. Lin. LIBSVM: a li- brary for support vector machines, 2001. http://www.csie.ntu.edu.tw/ cjlin/libsvm.

8. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

9. Li, Haoxiang, et al. "A convolutional neural network cascade for face detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

10. A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Acted Fa- cial Expressions in the Wild Database. In Technical Report, 2011

11. V. Ojansivu and J. Heikkil. Blur Insensitive Texture Classi- fication Using Local Phase Quantization. In Proceedings of the 3rd International Conference on Image and Signal Pro- cessing, ICISP'08, pages 236–243, 2008.

12. A. Bosch, A. Zisserman, and X. Munoz. Representing Shape with a Spatial Pyramid Kernel. In Proceedings of the ACM International Conference on Image and Video Retrieval, CIVR '07, pages 401–408, 2007.