DEEP COMPLEMENTARY BOTTLENECK FEATURES FOR VISUAL SPEECH RECOGNITION

Stavros Petridis

Imperial College London
Dept. of Computing, London, UK
stavros.petridis04@imperial.ac.uk

Maja Pantic

Imperial College London / Univ. of Twente
Dept. of Computing, UK / EEMCS, Netherlands
m.pantic@imperial.ac.uk

ABSTRACT

Deep bottleneck features (DBNFs) have been used successfully in the past for acoustic speech recognition from audio. However, research on extracting DBNFs for visual speech recognition is very limited. In this work, we present an approach to extract deep bottleneck visual features based on deep autoencoders. To the best of our knowledge, this is the first work that extracts DBNFs for visual speech recognition directly from pixels. We first train a deep autoencoder with a bottleneck layer in order to reduce the dimensionality of the image. Then the autoencoder's decoding layers are replaced by classification layers which make the bottleneck features more discriminative. Discrete Cosine Transform (DCT) features are also appended in the bottleneck layer during training in order to make the bottleneck features complementary to DCT features. Long-Short Term Memory (LSTM) networks are used to model the temporal dynamics and the performance is evaluated on the OuluVS and AVLetters databases. The extracted complementary DBNF in combination with DCT features achieve the best performance resulting in an absolute improvement of up to 5% over the DCT baseline.

Index Terms— Deep Bottleneck Features, Visual Speech Recognition, Deep Autoencoders, Long-Short Term Recurrent Neural Networks

1. INTRODUCTION

Deep bottleneck features (DBNFs) have been used successfully for acoustic speech recognition. The standard approach for extracting DBNFs involves the use of a deep network with a bottleneck layer in the middle [1]. The output of the bottleneck layer is treated as the bottleneck features which are then concatenated with other features, like Mel-Frequency Cepstral Coefficients, and fed to a temporal model, usually a Hidden Markov Model (HMM).

Different variants have also been proposed. Sainath et al. [2] trained a deep neural network with a constant number of hidden units per hidden layer and then used the output of the final layer as input to a deep autoencoder with a bottleneck layer in the middle. In this way, the features are first transformed to a discriminative representation before their dimensionality is reduced. Another approach has been proposed by Gehring et al. [3] who first train a stacked denoising autoencoder and then add a bottleneck layer and classification layers. The final network is fine-tuned to predict target phoneme states. Finally, Noda et al. [4] used a deep denoising autoencoder for producing denoised audio features without explicitly using the bottleneck features.

Recently few works on audiovisual or visual-only speech recognition have also been presented. Noda et al. [4] used a convolutional neural network to predict the phoneme that corresponds to an input image. Huang and Kingsbury [5] used a deep belief network in order to predict the posterior probabilities of HMMs states. However, to the best of our knowledge there are only three works which extract deep bottleneck visual features. Ngiam et al. [6] applied principal component analysis (PCA) to the mouth Region of Interest (ROI) and trained a deep autoencoder to extract bottleneck features. The features from the entire utterance were fed to a support vector machine ignoring the temporal dynamics of the speech. A similar approach was proposed by Ninomiya et al. [7] who also applied PCA to the mouth ROIs and used a deep autoencoder to extract bottleneck features but an HMM was used in order to take into account the temporal dynamics. Sui et al. [8] extracted local binary patterns from the mouth ROI and used a deep autoencoder to reduce their dimensionality. Then, the bottleneck features were concatenated with Discrete Cosine Transform (DCT) features and fed to an HMM.

In this work, we propose a framework based on a deep autoencoder to extract DBNFs for visual speech recognition directly from pixels. We also include DCT features during training in the bottleneck layer in order to reduce the redundant information of DBNFs and make them complementary to DCT features. We also use a Long-Short Term Memory Recurrent Neural Network (LSTM-RNN) in order to model the temporal dynamics. To the best of our knowledge, this is the first work which extracts DBNFs directly from pixels which are also complementary to DCT features.

First, an autoencoder is trained in order to compress the high dimensional image of the mouth ROI to a low dimen-

sional representation using a bottleneck layer in the middle. Then the decoding layers of the autoencoder are removed and replaced with classification layers and a softmax output layer in order to make the low dimensional representation more discriminative for visual speech recognition. At this stage, the DCT features can be appended to the bottleneck layer with the aim of making the DBNFs not only discriminative but also complementary to the DCT features. Finally, the complementary deep bottleneck features (DBNF-C) are concatenated with the DCT features and are fed to an LSTM-RNN which models the temporal dynamics of the utterance.

We perform experiments on two different databases, OuluVS and AVLetters. The use of complementary DBNFs in combination with DCT features results in the best performance leading to an absolute improvement of up to 5% over the DCT baseline. We also demonstrate the benefit of making the DBNFs complementary to DCT features and show that the bottleneck size does not have a great impact on the performance of the system.

2. DATABASES

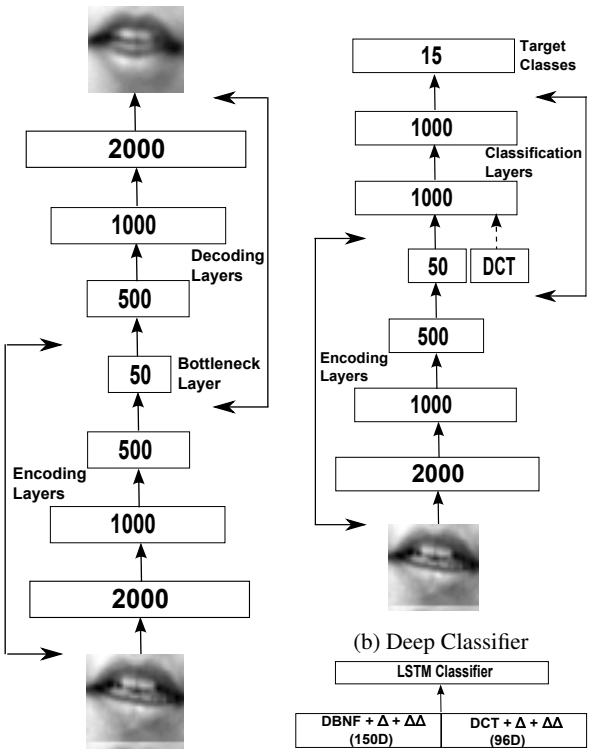
The databases used in this study are the AVLetters [9] and OuluVS [10]. The AVLetters contains 10 speakers saying 3 times the letters A to Z, so in total there are 30 utterances per letter. The mouth ROIs are provided and their size is 60 by 80.

The OuluVS contains 20 speakers saying 10 utterances, 5 times each, so in total there 100 examples per utterance. The utterances are the following: “Excuse me”, “Goodbye”, “Hello”, “How are you”, “Nice to meet you”, “See you”, “I am sorry”, “Thank you”, “Have a good time”, “You are welcome”. Sixty six points are tracked on the face using the tracker proposed in [11]. The mouth is located based on the mouth points and is rescaled to 32 by 32.

3. DEEP BOTTLENECK VISUAL FEATURES

The deep bottleneck visual feature extraction algorithm consists of 3 stages as shown in Fig. 1. The first stage involves the training of a deep autoencoder as shown in Fig. 1a. First, the encoding layers are trained in a greedy layer-wise manner using Restricted Boltzmann Machines (RBMs) [12]. The same architecture as in [13] is used, where 3 sigmoid hidden layers are used with sizes of 2000, 1000 and 500, respectively, followed by a linear bottleneck layer. Then, the decoding layers are initialised with the same weights as the encoding layers but in reverse order exactly in the same way as in [13]. The final step is the fine-tuning of the deep autoencoder for optimal reconstruction. The output of the bottleneck layer, which is a low dimensional representation of the input image, is treated as the DBNFs.

In the second stage, we remove the decoding layers and add two classification layers followed by an output softmax layer. Ideally, we would like the target classes to be the visemes shown in each image. However, there are no

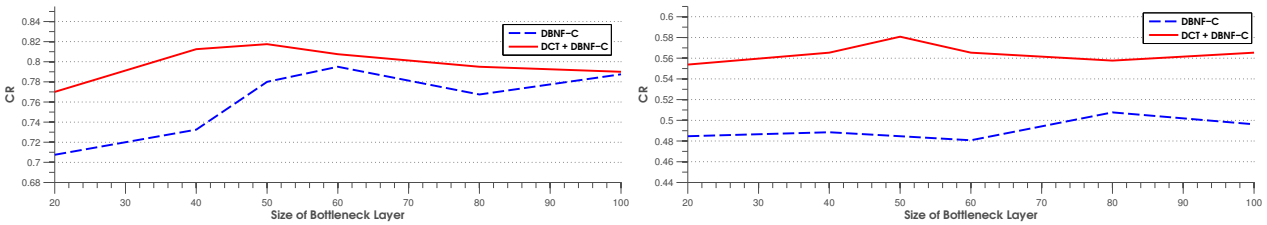


(a) Deep Autoencoder (b) Deep Classifier (c) LSTM Classifier

Fig. 1: System Overview. First, a deep autoencoder is trained (a) in order to compress the high dimensional image to a low dimensional representation which is the output of the bottleneck layer. Then, the decoding layers are replaced by classification layers (b) aiming to make the bottleneck features more discriminative. Optionally, the DCT features can be appended in the bottleneck layer in order to make the bottleneck features complementary to DCT features. Finally, both types of features are augmented with their first and second derivatives, concatenated and fed to an LSTM-RNN classifier (c) which models the temporal dynamics.

viseme labels in the datasets so we clustered the images using the k-means algorithm in order to create viseme-like clusters/targets. We set the number of clusters/targets to 15 since this is the usual number of visemes used in the literature [14]. The deep network is then fine-tuned with the aim of making the DBNFs more discriminative for visual speech recognition. During fine-tuning we can also append the DCT features to the bottleneck layer as shown in Fig. 1b. This will force the DBNFs not only to be discriminative but also complementary to DCT features.

Finally, in the third stage, we model the temporal dynamics of the features using an LSTM-RNN. The DBNFs and the DCT features are first augmented with their first and second derivatives and concatenated. Then, they are fed to an LSTM-RNN classifier which classifies the utterance.



(a) OuluVS (b) AVLetters

Fig. 2: Classification Rate (CR) as a function of the number of deep complementary bottleneck features (DBNF-C) and their combination with DCT (DCT + DBNF-C).

4. EXPERIMENTAL STUDIES

4.1. Experimental Setup

We first partition the data into training and test sets. The same protocol as the one used in [6], [9] is followed for the AVLetters datasets. The first two utterances of each subject are used for training and the last utterance is used for testing. This means that there are 520 training utterances and 260 test utterances. The total number of training and test frames is 12293 and 6269, respectively. Since the dimensionality of the mouth ROIs is too high (60 x 80), the images are first downscaled to 30 by 40 so the input dimensionality is 1200. A similar protocol is followed for the OuluVS database. The first 3 utterances of each subject are used for training and the other two are used for testing. There are 600 utterances for training and 400 utterances for testing. The total number of training and test frames is 16128 and 10946, respectively. The mouth ROI size is 32 by 32 so the input dimensionality is 1024.

The next step is the normalisation of data. As recommended in [12] the data should be z-normalised, i.e. the mean and standard deviation should be equal to 0 and 1 respectively, before training an RBM with linear input units. Hence, each image is z-normalised before training the autoencoder. Then, the extracted DBNFs and DCT features are also z-normalised before they are fed to the LSTM classifier.

Finally, the DCT features are extracted using a 2D DCT transform applied to the mouth ROI and the 32 lowest frequency coefficients are selected in a zig-zag left-to-right manner. The dimensionality of the DCT features after adding the first and second derivatives is 96.

4.2. Training

RBM Training: A Gaussian-Bernoulli RBM [12] is used for the first layer since the input (pixels) is real-valued, followed by two Bernoulli-Bernoulli RBMs and one Bernoulli-Gaussian RBM for the linear bottleneck layer. Each RBM is trained for 20 epochs with a mini-batch size of 100 and L2 regularisation coefficient of 0.0002 using contrastive divergence. The learning rate is fixed to 0.1 for the Bernoulli-Bernoulli RBMs and to 0.001 when one layer (input or bottleneck) is real-valued as suggested in [12].

Autoencoder Training: The autoencoder (Fig. 1a) is fine-tuned for 30 epochs using stochastic gradient descent and the mean squared error as a cost function. The mini-batch size is 100 and the learning rate is fixed to 0.001. Momentum is also used which increases linearly from 0.5 to 0.9 after 20 epochs. In order to reduce overfitting L2 regularisation is used together with the max-norm constraint [15]. The L2 regularisation coefficient is set to 0.005 and the maximun norm for the weights connected to any hidden neuron is set to 3 [15].

Classifier Training: The classifier (Fig. 1b) is trained using exactly the same parameters as the autoencoder above. The only difference is that the cross-entropy error is used as the cost function. The weights in the new layers are initialised from a Gaussian distribution with $\mu = 0$ and $\sigma = 0.1$.

LSTM-RNN Training: A GPU implementation of LSTMs is used [16] with the default parameters, i.e., momentum is set to 0.9 and the input is corrupted with Gaussian zero-mean noise with standard deviation of 0.1. The learning rate is set to 10^{-4} and one hidden layer with 180 cells is used. LSTMs are trained per frame, therefore during evaluation we apply majority voting to assign a single label to the sequence.

4.3. Effect of Bottleneck Size

A crucial parameter in the design of the deep bottleneck feature extraction system is the size of the bottleneck layer. In order to investigate the effect of this parameter we evaluate the performance as the size varies. We should emphasise that the total number of bottleneck features is 3 times the bottleneck size since we add the first and second derivatives. For example, if the bottleneck size is 50 then the dimensionality of the features fed to the LSTM-RNN is 150.

Fig. 1 shows the classification rate as a function of the size of the bottleneck layer. In case of OuluVS, Fig. 2a, the performance remains relatively stable beyond 50 neurons when DBNF-C is considered achieving the maximum performance of 79.5% for 60 neurons. When DBNF-C features are combined with DCT features then the performance remains relatively stable beyond 40 neurons with a maximum of 81.8% when the size is 50.

Similar patterns are observed for the AVLetters database,

Table 1: Classification Rate on the OuluVS and AVLetters databases. The bottleneck size is set to 50. DBNF: Deep Bottleneck Features, DBNF-C: Deep Bottleneck Features which are complementary to DCT features.

	Dim	OuluVS	AVLetters
DBNF	150	75.8	44.6
DBNF-C	150	78.0	48.5
DCT	96	76.8	54.6
DCT + DBNF	246	79.8	55.4
DCT + DBNF-C	246	81.8	58.1

Fig. 2b. The main difference is that the performance is stable across all sizes. The performance of DBNF-C varies from 48.1% to 50.8% when 60 and 50 neurons are used, respectively, and the performance of DCT + DBNF-C varies between 55.4% and 58.1% for 20 and 50 neurons, respectively.

Overall, we observe that when the DBNF-C are combined with the DCT features the performance is more stable than when the DBNF-C features are used alone. The use of 40 to 60 bottleneck neurons results in very similar performance for both databases and further increasing the bottleneck size does not seem to improve the performance. This same conclusion was also reached in [7]. In both cases the maximum is achieved when the bottleneck size is 50, i.e., the dimensionality of DBNF-C is 150, and this is the size used in subsequent experiments.

4.4. Results

Results on both databases are shown in Table 1. The DCT features are used as a baseline since they are the most commonly used feature in visual speech recognition. As expected they perform well and they outperform the bottleneck features in most cases. The addition of the DBNFs to the DCT features leads to an absolute improvement of 3% and 0.8% for the OuluVS and AVLetters databases, respectively, over the DCT baseline. This confirms the hypothesis that the bottleneck features contain useful information for the task at hand. A further improvement is observed when the complementary DBNFs are combined with the DCT features. An absolute improvement of 5% and 3.5% for the OuluVS and AVLetters databases, respectively, is reported. This clearly shows that the deep bottleneck features benefit when they are trained together with the DCT features. The joint training reduces the redundant information in the deep bottleneck features and makes them more complementary to the DCT features.

The best performance obtained on the AVLetters (58.1%) is similar to the performance achieved by [10] but it is worse than the results reported by Ngiam et al. [6] and Pei et al. [17]. However, the comparison with these works is not fair. Ngiam et al. used two external databases with thousands of additional training examples in order to train the deep autoencoders. Pei et al. used both appearance and shape features and a different training and test protocol so the results cannot be directly compared. In addition, both approaches tackle the problem as a classification problem where features from the entire utterance are extracted and fed to a classifier assuming that the beginning and end of the utterance is known. On the other hand, we extract features directly from pixels on a per frame basis and we model the temporal dynamics without explicit knowledge of the beginning and end of the sequence.

Due to lack of space we just report the most common mistakes for the best approach, i.e., DCT + DBNF-C. The best performing utterances for the OuluVS are the following: "I am sorry" and "Have a good time", whereas the most confused pair is "Thank you" and "Nice to meet you". For the AVLetters database, the most common confusions are between B and P, C and T, D and T, and U and Q. This is not surprising since both letters in each pair have the same visual representation. They consist of two phonemes where the first ones belong to the same viseme class [14] and the second one is the same. The letters which are classified correctly most of the time [1] are the following: F, O, R, V, Y. This is also not surprising since their visual representation is quite distinct from other other letters [14].

5. CONCLUSIONS

In this work, we present an approach to extract DBNFs directly from pixels. DBNFs are first created using a deep autoencoder which is then converted to a deep classifier in order to make the DBNFs more discriminative. We also append the DCT features in the bottleneck layer, which forces the DBNFs to become complementary to DCT features during training. The extracted complementary DBNFs are concatenated with the DCT features and fed to an LSTM classifier. Experimental results on two databases demonstrate that the extracted DBNFs when combined with the DCT features outperform the standard DCT baseline. In addition, the complementary DBNFs further improve the performance when combined with the DCT features, since they have been trained to contain less redundant and more complementary information to DCT features. Finally, we show that the performance does not depend much on the bottleneck size and a wide range of sizes leads to good performance.

6. ACKNOWLEDGEMENTS

This work has been funded by the European Community Horizon 2020 [H2020/2014-2020] under grant agreement no. 645094 (SEWA). The work of Stavros Petridis is also funded in part by the European Community 7th Framework Programme [FP7/2007-2013] under grant agreement no. 611153 (TERESA).

[1] They are classified with an accuracy of at least 70%.

7. REFERENCES

[1] Dong Yu and Michael L Seltzer, "Improved bottleneck features using pretrained deep neural networks.," in *INTERSPEECH*, 2011, vol. 237, p. 240.

[2] T.N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *ICASSP*, May 2012, pp. 4153–4156.

[3] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *ICASSP*, 2013, pp. 3377–3381.

[4] Kuniaki Noda, Yuki Yamaguchi, Kazuhiro Nakadai, Hiroshi G Okuno, and Tetsuya Ogata, "Audio-visual speech recognition using deep learning," *Applied Intelligence*, vol. 42, no. 4, pp. 722–737, 2015.

[5] Jing Huang and B. Kingsbury, "Audio-visual deep learning for noise robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 7596–7599.

[6] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng, "Multimodal deep learning," in *Proc. of the 28th international conference on machine learning*, 2011, pp. 689–696.

[7] Hiroshi Ninomiya, Norihide Kitaoka, Satoshi Tamura, Yurie Iribe, and Kazuya Takeda, "Integration of deep bottleneck features for audio-visual speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[8] Chao Sui, Roberto Togneri, and Mohammed Bennamoun, "Extracting deep bottleneck features for visual speech recognition," in *ICASSP*, 2015.

[9] Iain Matthews, Timothy F Cootes, J Andrew Bangham, Stephen Cox, and Richard Harvey, "Extraction of visual features for lipreading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 198–213, 2002.

[10] Guoying Zhao, Mark Barnard, and Matti Pietikainen, "Lipreading with local spatiotemporal descriptors," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1254–1265, 2009.

[11] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, "Incremental face alignment in the wild," in *In CVPR*, 2014, pp. 1859–1866.

[12] Geoffrey E Hinton, "A practical guide to training restricted boltzmann machines," in *Neural Networks: Tricks of the Trade*, pp. 599–619. Springer, 2012.

[13] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[14] Luca Cappelletta and Naomi Harte, "Phoneme-to-viseme mapping for visual speech recognition.," in *ICPRAM (2)*, 2012, pp. 322–329.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[16] Felix Weninger, Johannes Bergmann, and Björn Schuller, "Introducing currennt: The munich open-source cuda recurrent neural network toolkit," *Journal of Machine Learning Research*, vol. 16, pp. 547–551, 2015.

[17] Yuru Pei, Tae-Kyun Kim, and Hongbin Zha, "Unsupervised random forest manifold alignment for lipreading," in *ICCV*. IEEE, 2013, pp. 129–136.