Hybrid Acoustic-Lexical Deep Learning Approach for Deception Detection

Gideon Mendels[1], *Sarah Ita Levitan*[1], *Kai-Zhan Lee*[1], *Julia Hirschberg*[1]

[1]Columbia University, USA

gm2597@columbia.edu, sarahita@cs.columbia.edu, kl2792@columbia.edu, julia@cs.columbia.edu

Abstract

Automatic deception detection is an important problem with far-reaching implications for many disciplines. We present a series of experiments aimed at automatically detecting deception from speech. We use the Columbia X-Cultural Deception (CXD) Corpus, a large-scale corpus of within-subject deceptive and non-deceptive speech, for training and evaluating our models. We compare the use of spectral, acoustic-prosodic, and lexical feature sets, using different machine learning models. Finally, we design a single hybrid deep model with both acoustic and lexical features trained jointly that achieves state-of-the-art results on the CXD corpus.

Index Terms: deception, deep learning, computational paralinguistics

1. Introduction

Automatic deception detection is an important problem with implications for law enforcement, military, and intelligence agencies. Researchers in a variety of disciplines, including psychology, computer science, linguistics, and criminology, have worked to develop automated deception detection technologies. Despite many attempts spanning several modalities, there have been few objective successes, highlighting the inherent difficulty of this problem. In recent years the NLP and speech communities have been increasingly interested in deception detection. Language cues are inexpensive and easy to collect, and research examining text-based and speech-based cues to deception has been promising.

Previous work has examined text-based cues to deception in various domains, including criminal testimonies [1, 2, 3], hotel reviews [4], and opinions about controversial topics such as abortion [5]. There has been limited work examining acoustic-prosodic cues to deception, due to the lack of large, cleanly recorded corpora for deception. Hirschberg et al. [6] created the first large scale corpus of deceptive speech, comprising about 7 hours of subject speech, and found that acoustic-prosodic features are promising indicators of deception. In our ongoing work on deception detection, we have created a much larger corpus, the Columbia X-Cultural Deception (CXD) Corpus [7], comprising over 120 hours of subject speech. We previously developed a system using simple acoustic-prosodic features trained on a subset of the corpus and obtained about 3% absolute increase in accuracy above baseline [7].

In this work, we systematically evaluate several approaches to deception detection. We compare classification results for different feature sets and machine learning approaches. While prior work has focused on developing good features and used standard statistical machine learning for classification, in this work we explore deep learning approaches for deception detection for the first time. Deep learning has led to many advances in speech processing, and has not yet been applied to deception detection, possibly because of the lack of large training corpora. Leveraging the large CXD corpus, we compare the performance of several neural network architectures and features for this task. We present a novel hybrid approach that combines acoustic and lexical feature streams in a jointly trained deep neural network. This hybrid approach achieves state-of-the-art results for the CXD corpus (15% absolute improvement of F1-score over a random baseline, and about 3% absolute improvement over the LR trigram model), and our experiments varying the size of the train set suggest that additional data can further improve performance.

The remainder of the paper is organized as follows. Section 2 describes the corpus used, and Section 3 details the different feature sets. In Section 4, we report on the results of our deception classification experiments, including baseline statistical machine learning approaches as well as several deep learning systems. We conclude in Section 5 with a discussion and ideas for future work.

2. CXD Corpus

The CXD corpus is a collection of within-subject deceptive and non-deceptive speech from native speakers of Standard American English and Mandarin Chinese, all speaking in English. The corpus contains 170 conversations between 340 subjects. The corpus was collected using a fake resume paradigm, where subjects alternated between interviewing their partner and being interviewed, using a set of 24 biographical questions. Subjects were provided with financial incentive to lie effectively and judge deceptive statements correctly. While answering the questions, the interviewee pressed the T or F key on a keyboard, labeling each utterance spoken as true or false.

All of the audio recordings were transcribed using Amazon Mechanical Turk[1], and the transcripts were force-aligned with the audio. The speech was then automatically segmented into inter-pausal units (IPUs; separated by at least 50 ms) using Praat [8], and subsequently hand-corrected. The subject key-presses were aligned with the speech as well.

In this work, we use a subset of the CXD corpus. Because there is some ambiguity as to how to assign veracity labels to discrete segments of speech (some segments do not align with any key-presses, others align with both T and F key-presses), we use a strict rule to assign veracity labels. We retrieve all key-presses that lie between the current IPU start time and the following IPU start time. If all of these have the same label (all T or all F), we assign that label to the current IPU. Otherwise, if the key-presses for an IPU are conflicting, or if there are no available key-presses, we consider the IPU label to be ambiguous and eliminate it from our experimental dataset. This resulted in 49,106 IPUs, comprising 56.6% of all interviewee IPUs in the corpus at the time of these experiments. We utilize random under-sampling to balance the dataset, reducing it to 37,870 IPUs. We perform a 76/4/20 percent split into train,

[1] https://www.mturk.com/mturk/

Feature	Min	Max	Median	Mean	Stdv
Duration (s)	.013	13.25	1.06	1.38	1.06
Num words	1	47	3.0	4.46	4.24

Table 1: *IPU summary statistics*

validation, and test sets, resulting in 30,296, 1514, and 7574 samples respectively. Summary statistics for the IPU segments in this subset of the corpus can be found in Table 1.

3. Features

3.1. Acoustic-prosodic

We use the Interspeech 2013 (IS13) ComParE Challenge baseline feature set, which contains 6373 features from the computation of various functionals over low-level descriptor (LLD) contours extracted from openSMILE [9]. The LLD features include pitch (fundamental frequency), intensity (energy), spectral, cepstral (MFCC), duration, voice quality (jitter, shimmer, and harmonics-to-noise ratio), spectral harmonicity, and psychoacoustic spectral sharpness. This standard feature set has been successfully used for many computational paralinguistic tasks, including emotion recognition, native language detection, sincerity, and deception detection.

In addition to IS13, we use a smaller standard feature set designed for emotion recognition: the Interspeech 2009 (IS09) emotion challenge feature set. It contains 384 features extracted using openSMILE. Because emotion features have been shown to predict deception [10], we hypothesize that the IS09 emotion feature set will be useful for deception detection.

We also extract Mel-Frequency Cepstral Coefficients (MFCCs) using the `python_speech_features` [11] library, generating 13 cepstral coefficients per window of 256 frames, using a step of 100 frames.

3.2. Lexical

Ngrams Previous work has shown that a bag-of-words representation is useful for domain specific deception detection [4]. To capture differences in word usage between deceptive and truthful speech, we extract unigrams, bigrams, and trigrams from the corpus transcriptions. Table 2 lists the most useful ngrams for classifying deceptive and non-deceptive speech in our corpus.

Embeddings We also obtain distributed representations of words using GloVe [12] pre-trained word vectors. GloVe is an unsupervised learning algorithm that uses a log-bilinear regression model based on global word co-occurrence counts in a training corpus. We use a model trained on 2 billion tweets to produce 200-dimensional word vectors. Unlike ngram features, word embeddings have been shown to capture semantic relationships between words and are therefore very useful features for downstream NLP tasks.

4. Deception Classification Experiments

4.1. Baseline models

As a first step to building a classifier we trained two baseline models: (1) a Logistic Regression (LR) classifier trained using ngrams features, and (2) a Random Forest (RF) classifier trained using acoustic-prosodic features. For the LR model, we explored various techniques such as varying the number of ngrams, stopword removal, TF-IDF, and binary values. The best model uses simple trigram features with no additional manipulation, yielding an F1-score of 61.19%. Table 2 displays the top 10 ngram features for both the deceptive and non-deceptive classes. Although some of tokens are more corpus specific ("police", "columbia") we see that the majority are function words and fillers, which are corpus independent and should generalize to other domains.

Deceptive		**Non-deceptive**	
weight	feature	weight	feature
-1.2922	yeah did	1.5245	columbia
-1.2706	they didn	1.3183	facebook
-1.2591	uh just	1.2948	uh um
-1.2144	broke	1.2695	social
-1.2034	uh at	1.1988	sophmore
-1.1790	but he	1.1838	there was
-1.1569	oh no no	1.1746	have like
-1.1400	fell	1.1596	were just
-1.3180	police	1.3082	correct

Table 2: *Top N-gram features*

rank	feature	avg weight
1	energy	5.2237
2-13	MFCC	1.6373
14	voiceProb	1.2417
15-20	MFCC	1.2065

Table 3: *Top openSMILE IS09 features*

For the RF baseline model, we explore different values for the number of estimators in the forest, and compare the two openSMILE feature sets, as well as a feature selection method based on the ANOVA F-value between label and features. Our final RF model uses the openSMILE IS09 emotion feature set, 1000 estimators, and no feature selection, and it yields an F1 score of 59.54%. Table 3 displays the top IS09 feature groups for the RF model, obtained using Gini importance. This technique calculates feature importance as the sum over the number of splits across all trees that include a given feature, proportional to the number of samples that it splits. We observe that features representing functionals over MFCCs appear in 18 of the top 20 feature sets, suggesting usefulness in detecting deception.

Achieving 11.19% above a random baseline (50% since the dataset is balanced deceptive/non-deceptive) using only lexical features, and 9.54% above the random baseline using only acoustic-prosodic features, motivated us to design a model that combines both acoustic and lexical features. We trained a model on the entire feature set (openSMILE and ngrams) with RF but did not see any improvement compared to the original RF model.

Model	**Features**	**Prec.**	**Recall**	**F1**
LR	Trigrams	58.67	**63.95**	61.19
RF	OpenSMILE09	72.67	50.44	59.54
RF	OpenSMILE09, Trigrams	**76.11**	46.99	58.10
DNN	OpenSMILE13	63.65	58.03	60.71
DNN	OpenSMILE09	65.87	59.84	62.71
BLSTM	MFCC	54.19	55.10	54.64
BLSTM	WE	60.46	60.45	60.46
Hybrid	OpenSMILE09, WE	67.32	60.80	**63.90**

Table 4: *Classification experiments results (WE=Word Embeddings)*

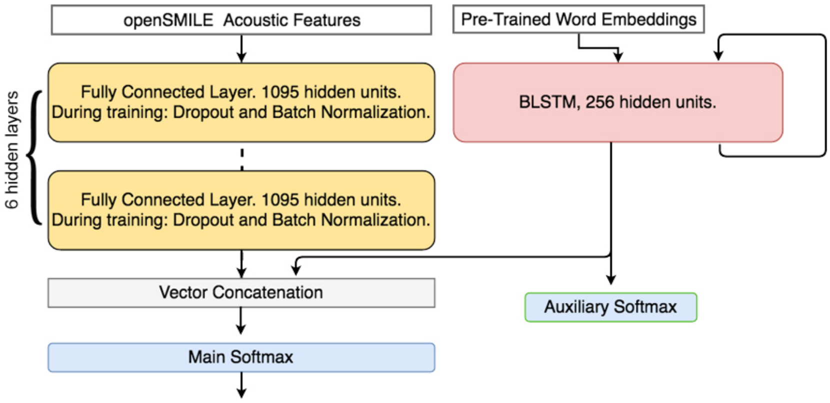
openSMILE Acoustic Features
Pre-Trained Word Embeddings
6 hidden layers
Fully Connected Layer. 1095 hidden units. During training: Dropout and Batch Normalization.
Fully Connected Layer. 1095 hidden units. During training: Dropout and Batch Normalization.
BLSTM, 256 hidden units.
Vector Concatenation
Auxiliary Softmax
Main Softmax

Figure 1: *Hybrid Acoustical Lexical Model Architecture, 4.2.5*

4.2. Deep learning models

In this section, we explore deep learning applications to deception detection. We first describe the Bayesian hyper-parameter optimization used to select the optimal hyper-parameters for our models. We then present the design and classification results of 4 deep learning models: lexical BLSTM, MFCC BLSTM, DNN-openSMILE, and a hybrid approach which achieves the best performance. The results of all these experiments, as well as the 2 baseline models, are presented in Table 4.

4.2.1. Bayesian hyper-parameter optimization

Previous work has indicated that Bayesian hyper-parameter optimization [13] is more time-efficient than random search [14] and far less computationally expensive than grid search [15]; therefore, in this experiment, we use the `spearmint` [16] library to perform such Bayesian optimization. Bayesian optimization operates on non-differentiable black-box functions or those that are computationally difficult to evaluate; in our case, this method was used to maximize the deep learning models' F1 score on the development set, based on various hyper parameters including learning rate, number of hidden layers, hidden unit count per layer, batch size, activation function, dropout rate, L2 regularization, batch normalization usage for each layer, and dropout usage for each layer.

There are two key parts to Bayesian optimization: the *prior distribution function* and the *acquisition function*. Upon each iteration of the optimization algorithm, the hyper-parameter vector $\boldsymbol{x}$ is evaluated and the prior distribution function, which is initially assumed, is updated based on the results from $\boldsymbol{x}$ to create a more accurate posterior distribution for use as the prior in the next time step. More importantly, the probabilistic acquisition function is maximized to determine which hyper-parameters vector to be tested next; it is a function $a(\boldsymbol{x}; \{\boldsymbol{x_n}, y_n\}; \theta)$ of the proposed next vector $\boldsymbol{x}$, the previous observation $\{\boldsymbol{x_n}, y_n\}$, and the parameter vector of the prior function θ.

4.2.2. Lexical BLSTM

Since ngram-based linear classifiers performed relatively well, we designed an additional lexical model based on the bidirectional long short-term memory (BLSTM) architecture. Recurrent models have been successful in related tasks of sentiment classification [17], speech recognition [18] and emotion detection [19]. The BLSTM model [20] is a modification of the original long short-term memory (LSTM) model [21] in that it analyzes input simultaneously in the forward and reverse time directions. Both models' effectiveness stem from the LSTM node's capacity to retain memory of its prior values with an internal state, bridging long temporal gaps. For every node at a given time-step t, with output gate y_{out}, input gate y_{in}, forget gate net, and differentiable activation functions g, h, output is defined as $y(t) = y_{out}(t)h(s(t))$ with internal state $s(t) = s(t-1) + y_{in}(t)g(net(t))$ [21]. We used pre-trained word embeddings described in 3.2 to compensate for the relatively small number of training examples. These GloVe embeddings are used to initialize the weights but we allow backpropagation to update embedding values during training. We use a single softmax layer that operates on the final output and state of the LSTM for prediction. Our final model uses a cell size of 256 and achieves an F1 score of 60% (Table 4).

4.2.3. MFCC BLSTM

Mel-Frequency Cepstral Coefficients (MFCC) are widely used for speech related tasks such as ASR, synthesis and speaker recognition. MFCC features tries to model human hearing by warping the frequencies output from the DFT onto a mel scale. While this was shown to improve phoneme recognition it is not clear if these features, in their raw form, are useful for deception detection. We train a BLSTM model as explained in 4.2.2 using the MFCC features described in section 3 with 512 hidden units. We use a sequence length of 750 windows which is the 90% percentile of our sequences length. We trim longer sequences and pad shorter ones. Each frame contains 13 MFCC values. Our MFCC model yields 54.64% F1 score after 300 epochs with a batch size of 128 (Table 4). The MFCC model performed poorly on the test set and under-fitted on the train set. We tried increasing the number of parameters (553, 473) but could not devise a model that fitted the train set successfully.

4.2.4. DNN: openSMILE

Following our relative success with the openSMILE feature set and a Random Forest classifier we designed a Deep Neural Network model (MLP) based on the same feature set. Prior to training we normalize our features by removing the mean and scaling to unit variance. Centering and scaling are done independently on each feature. Our model consists of six fully connected layers, each with 1095 hidden units followed by a ReLU [22] activation. For prediction we use a softmax layer with two outputs that corresponds to the two classes in our task. We use categorical cross-entropy as our loss function. During training the output of each layer is normalized using Batch Normalization [23] and passed through a Dropout layer [24] with a 0.497 probability. Our model has many parameters and a high dropout rate reduces the risk of over-fitting. Additionally, we add L2 regularization on the weights with a value of 0.2. We train our model using stochastic gradient descent with a learning rate of 0.00134 that reduces by 50% for every 10 epochs with no improvement on training loss. The above hyper parameters were obtained using the Bayesian Optimization method described in section 4.2.1. Overall, we trained 528 different variations of this model. Our best model yields an F1 score of 62.71% on the IS09 openSMILE feature set and 60.71 % on the IS2013 feature set (Table 4.

4.2.5. Hybrid: LSTM + DNN

In our final experiment we combine our models from section 4.2.2 and section 4.2.4. One of the advantages of neural networks is the ability to tailor the architecture to the task and combine sequential and discrete features in a single model. Unlike most ensemble methods our hybrid model is trained jointly without explicit voting between the acoustic and lexical based areas. The two models are merged by taking the output of the last hidden layer in our DNN model training on IS09 features and concatenating it with the output of LSTM. Although the trigram model performed slightly better than the lexical BLSTM, its use is problematic in the hybrid model approach. The dimension of the trigram feature vector is 230,559 and concatenating it to the output of the acoustic model with dimension of 1095 would require the weights corresponding to the acoustic features to compensate for the dimension differences, which is sub-optimal for gradient propagation.

As with previous models we use the softmax function to normalize the last layer's output and generate class probabilities. We define this formally for clarity:

$$\Pr(Y = i \mid \boldsymbol{x}, W_{n-1}, b_{n-1}) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

where n is the number of hidden layers and $\boldsymbol{x}$ is $\boldsymbol{u}^\frown\boldsymbol{v}$, the concatenation of the DNN and LSTM outputs. This architecture failed to improve on the original DNN model yielding an F1 score of 62.70%. This led us to the hypothesis that during back propagation, the acoustic-based area of the network is being penalized more than the lexical area. To test our hypothesis, we attached an auxiliary softmax prediction layer to the LSTM output and used it to predict the test set. While the original lexical LSTM model achieved a score of 60% this area of the network only gave a score of 54%. This result confirmed our hypothesis that although the overall loss seemed to converge, the lexical area of the network was not optimized. Although it is possible to freeze the weights of the acoustic area and continue training the lexical area, that approach is not preferred due the manual intervention required. Instead, we computed the loss of the network twice: once for the main softmax and once for the auxiliary softmax. Using a parameter λ we compute a weighted sum of the two error matrices. This approach allows us to train the network without user intervention and achieve an F1 score of 63.90% (Table 4). We treated λ as a hyper-parameter and, using Bayesian optimization, found an optimal value of 0.67, doubling the significance of the loss computed from the auxiliary softmax relative to the main softmax. As shown in Table 4, this hybrid model achieves the highest F1-score of the 7 approaches described here. The architecture of this hybrid model is shown in Figure 1.

4.3. Impact of training size

We experiment with varying the size of the train set to evaluate the impact of amount of training data on classification performance. Fig. 2 shows the F1-scores of the RF openSMILE model, an LSTM-embeddings model, and the hybrid model, trained on increasing subsets of the training data from 10% to 100%. All models were evaluated on a fixed test set. We observe that both neural network models (LSTM and hybrid) are very sensitive to amount of training data, while the RF model does not benefit as much from the increase in training data; thus, this suggest that increasing the amount of training data improves the performance of the hybrid model more rapidly than that of ensemble classifiers or DNNs training on a single feature set.

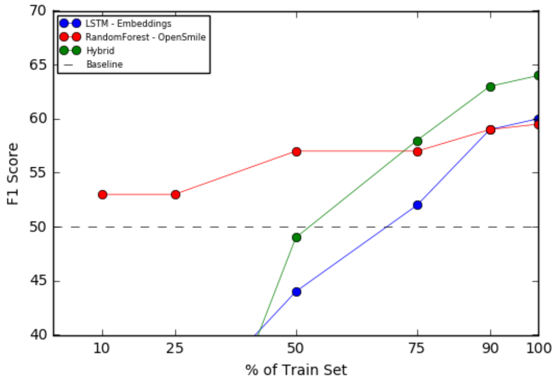

Figure 2: *Model performance w.r.t train set size*

5. Conclusions and Future Work

We have evaluated the performance of several machine learning approaches to the critical problem of deception detection. We developed a novel hybrid deep learning model that trains on both acoustic and lexical features, achieving state-of-the-art results on the CXD corpus. Although this hybrid model achieved the highest F1-score of 63.9%, the RF model trained on concatenated acoustic and lexical features achieved the highest precision of 76.11%. In this work we chose to optimize for F1-score, since this a balance of precision and recall. However, the choice of metric to optimize is largely dependent on the application. It is possible that a high precision model would be useful to practitioners in situations where classification precision is more important than recall.

We show that increasing the amount of training data leads to the most rapid increases in performance in the hybrid model. However, we used only 56.6% of the IPUs in the CXD corpus and are working to increase the amount of training data by improving alignment approaches between IPUs and veracity labels, so that we can test if a similar trajectory continues with additional data. In addition, we plan to evaluate these approaches on other deceptive speech corpora, in order to assess the robustness of the various features and models in other domains. This will also enable direct comparison of our methods with prior work on detection of deceptive speech.

6. References

[1] J. Bachenko, E. Fitzpatrick, and M. Schonwetter, "Verification and implementation of language-based deception indicators in civil and criminal narratives," in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2008, pp. 41–48.

[2] T. Fornaciari and M. Poesio, "Automatic deception detection in italian court cases," *Artificial intelligence and law*, vol. 21, no. 3, pp. 303–340, 2013.

[3] V. Pérez-Rosas, M. Abouelenien, R. Mihalcea, Y. Xiao, C. Linton, and M. Burzo, "Verbal and nonverbal clues for real-life deception detection." in *EMNLP*, 2015, pp. 2336–2346.

[4] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 309–319.

[5] M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards, "Lying words: Predicting deception from linguistic styles," *Personality and social psychology bulletin*, vol. 29, no. 5, pp. 665–675, 2003.

[6] J. Hirschberg, S. Benus, J. M. Brenier, F. Enos, S. Friedman, S. Gilman, C. Girand, M. Graciarena, A. Kathol, L. Michaelis *et al.*, "Distinguishing deceptive from non-deceptive speech." in *Interspeech*, 2005, pp. 1833–1836.

[7] S. I. Levitan, G. An, M. Wang, G. Mendels, J. Hirschberg, M. Levine, and A. Rosenberg, "Cross-cultural production and detection of deception from speech," in *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*. ACM, 2015, pp. 1–8.

[8] P. P. G. Boersma *et al.*, "Praat, a system for doing phonetics by computer," *Glot international*, vol. 5, 2002.

[9] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1459–1462.

[10] S. Amiriparian, J. Pohjalainen, E. Marchi, S. Pugachevskiy, and B. Schuller, "Is deception emotional? an emotion-driven predictive approach," *Interspeech 2016*, pp. 2011–2015, 2016.

[11] J. Lyons, "Python speech features," https://github.com/jameslyons/python_speech_features, 2017.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[14] B. E. Stuckman and E. E. Easom, "A comparison of bayesian/sampling global optimization techniques," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 5, pp. 1024–1032, 1992.

[15] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[16] H. I. P. S. Group, "Spearmint," https://github.com/HIPS/Spearmint, 2017.

[17] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification." in *EMNLP*, 2015, pp. 1422–1432.

[18] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[19] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5200–5204.

[20] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[24] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.