



Leveraging Second-Order Log-Linear model for improved deep learning based ASR performance

Ankit Raj, Shakti P. Rath, Jithendra Vepa

Samsung Research Institute India - Bangalore

{a.raj, shakti.rath, jithendra.v}@samsung.com

Abstract

Gaussian generative models have been shown to be equivalent to discriminative log-linear models under weak assumptions for acoustic modeling in speech recognition systems. In this paper, we note that the output layer of deep learning model consists of a first-order log-linear model, also known as logistic regression, which induces a set of homoscedastic distributions in the generative model space, resulting in linear decision boundaries. We leverage the above equivalence to make the deep learning models more expressive by replacing the first order log-linear model with a second-order model, which leads to heteroscedastic distributions, as a result, the linear decision boundaries are replaced with quadratic ones. We observe that the proposed architecture yields a significant improvement in speech recognition accuracy compared to the conventional model having a comparable number of parameters. Relative improvement of 8.37% and 3.92% in word error rate (WER) is obtained for shallow and deep feed-forward networks respectively. Moreover, with Long Short-Term Memory (LSTM) networks with projection matrix, we obtain significant relative improvement in WER over the standard architecture.

Index Terms: log-linear models, gaussian generative models, Long Short-Term Memory, acoustic modeling

1. Introduction

With the advancement of deep learning, Deep Neural Networks (DNNs) ([1] [2]) and Recurrent Neural Networks (RNNs) [3] have replaced Gaussian Mixture Models (GMMs) [4] due to their superior performance for automatic speech recognition (ASR) tasks. Recently, Long Short-Term Memory (LSTM)-RNNs based acoustic models [5] have been shown to outperform DNN and vanilla RNN models [6]. Since DNNs are low-footprint networks, they are preferred for on-device models and currently being used for many production systems.

Several studies have been done to design the deep learning models capable of encoding meaningful features which can potentially enhance the acoustic modeling. In [7], a network is proposed which takes raw multichannel waveforms as input directly and learns a bank of bandpass beamformers. [8] proposes to combine Convolutional Neural Networks (CNNs) [9] and LSTMs layers which transforms features into space making that output easier to classify. [10] presents time-delay neural networks (TDNNs) which are effective in modeling long-term temporal dependencies. The goal of this paper is to explore the impact and performance improvements on expanding feature space at penultimate layer because of second-order terms.

Recently Heigold et al. [11] showed that under rather weak assumptions, there is an equivalence between generative and discriminative model [12]. Specifically, a Gaussian model with a prior was shown to generate exactly the same posteriors as the ones generated by a log-linear model with suitable parameters.

In other words, given a Gaussian generative model, log-linear model parameters can be computed to yield the same posteriors and vice versa. As a special case, it was shown that a first-order log-linear model (only 0-th and 1-st order terms) induces a set of Gaussian models with homoscedastic distributions, i.e., all covariance matrices being equal to constant (any positive definite matrix) [13]. It is well known from the classification decision theory that under homoscedastic assumptions, the decision boundaries are hyper-planes, leading to linear classifiers [14].

In this paper, we note that neural network based acoustic models, whether DNN or LSTM, may be seen as layer by layer processing of acoustic features, followed by a first-order log-linear classifier (also known as logistic regression) at the output layer. Lower layers essentially discover low-level feature representation while the higher layers progressively learn more abstract features. Therefore, it may be inferred that the input and hidden layers learn to project the features into space where they become as linearly separable as possible, while the output layer (a linear classifier) operating to predict the classes using hyperplane classifiers [14]. We argue that the homoscedasticity (or equivalently the linear classification scenario) is rather a sub-optimal assumption. To this end, we propose to extend the log-linear model in DNN and LSTM to include a second-order term. Using the equivalence, it may be shown that, in this case, the log-linear model becomes equivalent to heteroscedastic generative models, i.e., all co-variances are different.

The main contribution of the paper is to show that under this proposed framework, DNN, as well as LSTM, outperforms the conventional model that is based on first-order log-linear model. Our results show that with the proposed method, we can obtain the performance that can only be achieved with the conventional system having significantly more parameters. Hence, our approach results in reducing footprint of the network without affecting accuracy, making it suitable for on-device systems. Earlier works on the similar lines are: introducing low-rank layer before output [15] and projection matrix after LSTM layer [5].

The remainder of this paper is organized as follows: Section 2 describes the equivalence between generative and log-linear models and how deep learning models are related to log-linear models. Section 3 describes how we leverage the equivalence by introducing second-order terms at penultimate layer to make the log-linear models even more expressive. Experimental setup and results for DNNs and LSTMs based acoustic models are discussed in Section 4, followed by conclusions in Section 5.

2. Equivalence of Log-linear and Gaussian models

In this section, we present a detailed account of log-linear, showing the equivalence with Gaussian model with a prior [11]. Unlike generative models, log-linear models are discriminative models that directly express the posterior-probabilities of the

Table 1: Transformation of Log-linear model into gaussian model parameters

1.	Σ_s	=	$-\frac{1}{2}(\lambda_{s2} + \Delta\lambda_2)^{-1}$
2.	μ_s	=	$\Sigma_s \lambda_{s1}$
3.	$p(s)$	=	$\exp(\lambda_{s0} + \frac{1}{2}(\mu_s^T \Sigma_s^{-1} \mu_s + \log 2\pi\Sigma_s + \Delta\lambda_0))$

classes. It was shown by Heigold et al. that these models can be equivalently represented by Gaussian model with priors, such that the posterior probabilities induced by both are exactly the same. In other words, given the parameters of the Gaussian model, the parameters of the log-linear model can be analytically computed, and vice versa. The log-linear models with generalized features can be expressed as follows:

$$p_\lambda(s|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp\left(\sum_i \lambda_{si} f_i(\mathbf{x})\right) \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^D$ denotes the input features, $f_i : \mathbb{R}^D \rightarrow \mathbb{R}$, $\mathbf{x} \mapsto f_i(\mathbf{x})$ is a generalized feature mapping, $\lambda_{si} \in \mathbb{R}$ are model parameters, $s \in \{1, \dots, S\}$ are the classes and $Z_\lambda(x)$ is a normalization constant that makes sure that the quantities are probabilities. By taking the generalized features as : zeroth - ($f_0(x) = 1$), first - ($f_{1d}(x) = x_d$), and second - ($f_{2dd'}(x) = x_d x_{d'}$) order features, the second order log-linear model takes the form

$$p_\lambda(s|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp(\lambda_{s0} + \lambda_{s1}^T \mathbf{x} + \mathbf{x}^T \lambda_{s2} \mathbf{x}) \quad (2)$$

where the log-linear model parameters are $\lambda = \{\{\lambda_{s0} \in \mathbb{R}\}, \{\lambda_{s1} \in \mathbb{R}^D\}, \{\lambda_{s2} \in \mathbb{R}^{D \times D}\}\}$. This is the form of the model we are interested in this paper. Now, let us consider a generative Gaussian model with prior – the joint distribution is given by

$$p_\theta(s, \mathbf{x}) = p(s)N(\mathbf{x}|\mu_s, \Sigma_s) \quad (3)$$

where the parameters are $\theta = \{\{\mu_s \in \mathbb{R}^D\}, \{\Sigma_s \in \mathbb{R}^{D \times D} : \Sigma_s \succ 0\}\}$. $p(s)$ is the class prior such that $\{p(s) \in \mathbb{R}^+ : \sum_s p(s) = 1\}$. By Bayes' rule, $p_\theta(s|\mathbf{x}) = p_\theta(s, \mathbf{x})/p_\theta(\mathbf{x})$. In [11], it was shown that the parameters of the Gaussian distribution can be transformed to the log-linear space such that the two models induce the same posterior probabilities. The transformation in the reverse direction, however, is not straight-forward as the generative model parameters are restricted to satisfy certain constraints, namely, the covariance matrix being positive definite and the priors must be positive and sum to unity. To this end, it was noted that if a constant term (class independent) is added to the log-linear parameters, the posterior probabilities remain unchanged. In particular, a transformation of the parameters, $\lambda_{si} \mapsto \lambda_{si} + \Delta_i$, does not alter the posterior in Eq. 2. By taking advantage of this ambiguity, the mapping in the reverse direction was worked out, which are shown in Table 1. The $\Delta\lambda_2$ is chosen such that the co-variance becomes positive definite and $\Delta\lambda_0$ is chosen to ensure that the priors terms remain probabilities.

We observe that the second-order log-linear model defined by Eq. 2 induces heteroscedastic distributions in the generative model space, which leads to quadratic decision boundaries. When λ_{s2} is set to zero, Eq. 2 reduces to a first-order log-linear model given by

$$p_\lambda(s|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp(\lambda_{s0} + \lambda_{s1}^T \mathbf{x}). \quad (4)$$

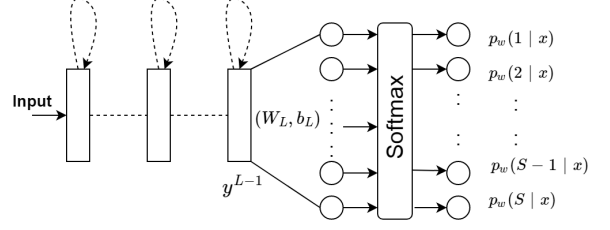


Figure 1: Typical neural network set-up

Using the equivalence again, without the loss of generality the covariance matrices may be set to $\Sigma_s = \mathbf{I}, \forall s$, which corresponds to homoscedastic distributions. In this setting, the generative model induces linear (hyperplane) decision boundaries.

2.1. Comparison of Neural Network and Log-linear models

In this section, we draw a comparison between neural-network based acoustic model and log-linear model. We note that the output layer of DNN (or LSTM) consists of a logistic regression model as shown in Fig. 1, defined as

$$O_s = p_w(s|\mathbf{x}) = \frac{\exp(\mathbf{w}_{Ls}^T \mathbf{y}^{L-1} + b_{Ls})}{\sum_{k=1}^S \exp(\mathbf{w}_{Lk}^T \mathbf{y}^{L-1} + b_{Lk})} \quad (5)$$

where \mathbf{w}_{Ls} is the weights pertaining to the s -th output unit and b_{Ls} is the bias. By comparison with Eq. 4, we note that the output layer corresponds to a first-order log-linear model, with the difference that the features are processed by a neural network up to the last hidden layer, i.e., $\mathbf{y}^{L-1} = \phi(\mathbf{x})$. This leads to the familiar conclusion that the neural networks process features layer by layer projecting them onto space at the output layer where they can be classified using a linear classifier, which is a first-order log-linear model (or logistic regression).

3. Proposed Method

Taking motivation from the equivalence discussed above, we propose to replace the first-order log-linear component in the output layer of the neural network with a second-order term. Now, the expression for (DNN or LSTM) prediction is given by

$$p_w(s|\mathbf{x}) = \frac{\exp(\varphi_s)}{\sum_{k=1}^S \exp(\varphi_k)}, \quad (6)$$

where

$$\varphi_s = \mathbf{y}^{L-1T} \mathbf{W}_{2s} \mathbf{y}^{L-1} + \mathbf{w}_{1s}^T \mathbf{y}^{L-1} + b_s. \quad (7)$$

\mathbf{W}_{2s} in Eq. 7 refers to the affine transform in the second-order term. The motivation behind the modification is to remove the constraint of homoscedasticity in the standard deep learning models. By forcing the second order parameters \mathbf{W}_{2s} to be (non-constant and) trainable, the covariance matrices in the generative space would naturally become different for each state, giving rise to heteroscedastic distributions. As a consequence, the linear decision boundaries are replaced with quadratic ones, which are more optimal.

However, the downside of having a second-order term is that this introduces a considerably large number of parameters into the network. Specifically, for a network with S number of output units and k number of neurons at last hidden layer,

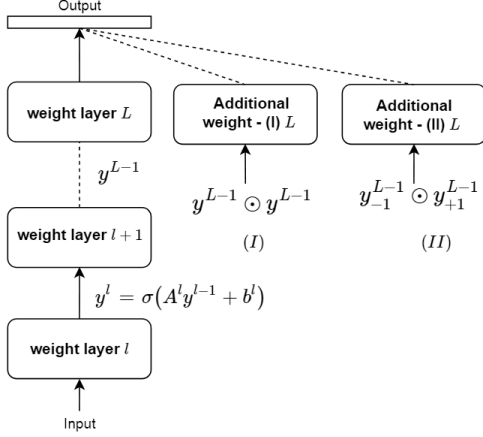


Figure 2: Proposed model architecture

$k^2 S$ additional parameters are introduced (k^2 for every state s), which is hard to train. In order to deal with the issue, our approach is to impose restrictions on the structure of \mathbf{W}_{2s} . In particular, we use diagonal or bi-diagonal matrices. We refer to this approach as Second-Order Log-Linear Model with Diagonal (SeLoM-D) or Bi-diagonal (SeLoM-B) \mathbf{W}_{2s} matrices.

3.1. Diagonal matrices

Assuming that \mathbf{W}_{2s} is diagonal (SeLoM-D), the quadratic term in Eq. 7 will turn out to be:

$$\mathbf{y}^{L-1T} \mathbf{W}_{2s}^{diag} \mathbf{y}^{L-1} = \bar{\mathbf{w}}_{2s}^T (\mathbf{y}^{L-1} \odot \mathbf{y}^{L-1}) \quad (8)$$

where $\bar{\mathbf{w}}_{2s}$ is the vectorized representation of the diagonal elements of \mathbf{W}_{2s}^{diag} and \odot denotes element-wise multiplication. In this proposed architecture shown in Fig. 2, the additional component (I) represents the implementation of diagonal \mathbf{W}_{2s} matrix (from Eq. 8) which is a second-order log-linear model in \mathbf{y}^{L-1} space. The network training can be done in a straightforward manner by noting that

$$\begin{aligned} \boldsymbol{\varphi}_s &= \left[\mathbf{w}_{1s}^T; \bar{\mathbf{w}}_{2s}^T \right] \left[\mathbf{y}^{L-1T}; \left(\mathbf{y}^{L-1} \odot \mathbf{y}^{L-1} \right)^T \right]^T + b_s \\ &= \tilde{\mathbf{w}}_s^T \tilde{\mathbf{y}}^{L-1} + b_s \end{aligned} \quad (9)$$

where “;” denotes row-wise concatenation. The $\tilde{\mathbf{w}}_s$ can be trained using the standard back-propagation formula used for affine components, however with changes appropriate to the feature expansion shown in Eq. 9. We note that in this case, the number of parameters increases by kS .

3.2. Bi-diagonal matrices

In this set-up, the \mathbf{W}_{2s} matrix is assumed to be bidiagonal (SeLoM-B), i.e., principal diagonal and either the diagonal above or the diagonal below are non-zero. We note that without loss of generality, the matrix can be considered as bidiagonal instead of tridiagonal. For the sake of simplicity, we have taken the \mathbf{W}_{2s} matrix to be lower bidiagonal. The quadratic term in Eq. 7 becomes:

$$\begin{aligned} \mathbf{y}^{L-1T} \mathbf{W}_{2s} \mathbf{y}^{L-1} &= \bar{\mathbf{w}}_{2s}^T [\mathbf{y}^{L-1} \odot \mathbf{y}^{L-1}] \\ &\quad + \bar{\mathbf{w}}_{2s,(-1)}^T [\mathbf{y}_{-1}^{L-1} \odot \mathbf{y}_{+1}^{L-1}] \end{aligned}$$

where $\bar{\mathbf{w}}_{2s}$ and $\bar{\mathbf{w}}_{2s,(-1)}$ are the vectorized representations of diagonal and lower diagonal elements of \mathbf{W}_{2s} respectively, and

$$\mathbf{y}_{-1}^{L-1} = [y_1^{L-1}, y_2^{L-1}, \dots, y_{N-1}^{L-1}] \quad (10)$$

$$\mathbf{y}_{+1}^{L-1} = [y_2^{L-1}, y_3^{L-1}, \dots, y_N^{L-1}] \quad (11)$$

The reduced form (similar to first order term) can be modeled using two additional matrices (I & II) shown in Fig. 2. Besides the square terms, $(\mathbf{y}^{L-1} \odot \mathbf{y}^{L-1})$, as discussed in section 3.1, it introduces cross-terms with offset one $(\mathbf{y}_{-1}^{L-1} \odot \mathbf{y}_{+1}^{L-1})$.

The extra parameters, i.e., additional weight matrix (I) corresponding to diagonal \mathbf{W}_{2s} and additional weight matrices (I and II) for bidiagonal \mathbf{W}_{2s} are learned along with other parameters of the network while training jointly.

4. Experiments

Experiments are conducted using 500 hours of US English multi-condition data for acoustic model training. The alignments have been generated using a GMM-HMM model with context-dependent states (or senones) trained using boosted MMI method [16] [17]. Features used are 13-dimensional Mel Frequency Cepstral Coefficients (MFCC) extracted from every speech frames. All the models are initialized using layer-wise discriminative pre-training, followed by [18] cross-entropy (CE) fine-tuning. A WFST-based decoder has been used. The size of the n-gram language model is 20-million and a lexicon containing 1 million words has been used. The test was conducted on 4 hours of eval data set for computation of word error rate (WER). The Kaldi speech recognition toolkit [19] was used for the experiments. All experiments are conducted using either DNN and LSTM based acoustic model to evaluate the performance of our proposed method. The performance metrics are WER and number of parameters in the network. To verify that the observed performance improvement is not just because of having more parameters introduced by the second order terms, we evaluate the proposed method with the conventional model with an approximately equal number of parameters.

4.1. DNN models

4.1.1. Model Architecture

In the DNN set-up, we use a splicing of ± 5 at input layer to form 143 dimension input per frame. We have used the method of using low-rank layer before the output layer which significantly reduces the number of parameters as proposed by [15]. Two different architectures of the feed-forward models have been used as our baselines over which we have done experiments with our proposed method.

- DNN with 2 feed-forward hidden layers (1024 units) and 1 low rank layer (128 units)
- DNN with 5 feed-forward hidden layers (1024 units) and 1 low rank layer (128 units)

4.1.2. Results and Discussion

We performed experiments with 3 variants of both the baselines using the proposed method.

1. Introducing diagonal \mathbf{W}_{2s} term (SeLoM-D) at low-rank layer output (Section 3.1)
2. Making \mathbf{W}_{2s} term bidiagonal (SeLoM-B) at low-rank layer as described in Section 3.2

Table 2: Results on **3-layers** feed-forward models

Model Architecture	No. of Parameters	WER (%)
DNN - 3 layers	1.86M	15.09
+ SeLoM-D	2.39M	13.11
+ SeLoM-B	2.91M	12.91
DNN - 4 layers	2.91M	14.09

Table 3: Results on **6-layers** feed-forward models

Model Architecture	No. of Parameters	WER (%)
DNN - 6 layers	5.01M	13.15
+ SeLoM-D	5.54M	12.35
+ SeLoM-B	6.06M	12.01
DNN - 7 layers	6.06M	12.5

- Conventional feed-forward model with 1 hidden layer more than the baseline set-up which makes the number of parameters same as case 2.

Tables 2 and 3 show the results of our experiments using shallow and deep feed-forward networks respectively. Results show that using proposed method, these networks outperform their respective baselines significantly. However, relative improvement for shallow networks is more than that for deep networks. For shallow networks, a relative improvement of 8.37% in WER is obtained over a standard system having same number of parameters whereas it is 3.92% for the deep ones. This can be explained by fact that shallow networks learn comparatively low-level feature representation and hence has more scope of learning abstraction which is useful for better classification. On the other hand, deep networks learn high-level feature representation on their own which are already discriminative. Hence, introducing non-linearity doesn't help them as much as compared to the shallow networks. However, based on results of both the networks, it can be argued that the deep learning models become more expressive after the second-order term is introduced using our proposed method which cannot otherwise be obtained just by adding an extra layer in the conventional model.

4.2. LSTM models

Long Short-Term Memory (LSTM) networks are known to capture long-term temporal relations, hence, they are widely used for acoustic modeling. Standard LSTM cells architecture introduces a large number of parameters. Several variants like simplified LSTM [20], Gated Recurrent Units (GRUs) [21], are commonly used to reduce the number of parameters.

4.2.1. Model Architecture and Training

In our experiments, we have used LSTM with projection matrix (LSTMP) [5] that has an additional projection layer which maps hidden state to lower dimension, reducing the number of parameters significantly. Splicing (± 2) is used at input layer to form 65 dimension input per frame. 20 frames in the sequence are used for truncated backpropagation through time (BPTT) [22], 40 frames are used as left context to the sequence for prediction of the first label when passed through the model while training. The output is delayed by 5 time-steps for the LSTMP to see future frames while predicting for the current frame [5].

Table 4: Results on LSTM models

Model Architecture	No. of Parameters	WER (%)
LSTMP - 3 layers	3.43M	12.5
+ SeLoM-D	4.49M	12.05
+ SeLoM-B	5.55M	11.32
Small LSTMP + SeLoM-D	3.37M	12.21
LSTMP - 5 layers	5.27M	11.56

The baseline model consists of 3 uni-directional LSTMP hidden layers having 512 cells and projection matrix of dimension 512×256 , hence, effectively hidden state dimension is 256.

4.2.2. Results and Discussion

We have experimented with 3 variants of LSTMP (other than the baseline) using the proposed method.

- Introducing diagonal \mathbf{W}_{2s} term (SeLoM-D) at 3rd layer projection matrix output
- Making \mathbf{W}_{2s} term bidiagonal (SeLoM-B) at 3rd layer projection matrix output
- Reducing the 3rd layer projection matrix to 512×128 (small LSTMP) and introducing SeLoM-D \mathbf{W}_{2s} term

The first two experiments incorporate additional matrices corresponding to second-order terms because of which extra parameters are inserted. $\sim 30\%$ additional parameters have to be learned for the first variant whereas $\sim 60\%$ for the second one. In the third case, we reduce the last layer projection matrix dimension from 512×256 to 512×128 and introduce diagonal second order parameter, \mathbf{W}_{2s} . Projection matrices for first two layers are kept the same, i.e., 512×256 . This ensures that the model is almost of the same size (in fact, slightly lesser) as baseline model still it performs better (relative improvement of 2.32% in WER). Table 4 shows the results of our experiments. We note that the SeLoM model yields a consistent improvement over the baseline LSTMP model. In order to compare with LSTMP (3 layers) + SeLoM-B (analogous to what we did for DNNs), we have conducted experiment with LSTMP - 5 layers which has similar number of parameters. A relative improvement of 2.07% in WER is obtained with our proposed method over the standard 5 - layers LSTMP.

5. Conclusions

In this paper, we have leveraged the equivalence of generative gaussian and log-linear models for acoustic modeling. We have proposed a method which makes the distributions for deep learning models heteroscedastic which are otherwise homoscedastic in standard networks. Our experiments show that the relaxation of homoscedasticity by using second-order log-linear model (SeLoM) form makes the model more expressive resulting in significant improvement in performance. For shallow feed-forward networks, a relative improvement of 8.37% in WER is obtained over the standard model with a comparable number of parameters whereas that of 3.92% is obtained for the deep networks. For LSTM networks with projection matrix, a relative improvement of 2.07% in WER is obtained over the standard architecture. Results indicate that the expanded feature space obtained using the proposed method becomes more discriminative which cannot be merely obtained by adding an extra layer or incorporating more parameters.

6. References

- [1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.
- [2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [3] 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.
- [4] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and trends in signal processing*, vol. 1, no. 3, pp. 195–304, 2008.
- [5] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, 2014, pp. 338–342.
- [6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [7] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4624–4628.
- [8] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.
- [9] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.
- [10] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [11] G. Heigold, H. Ney, P. Lehnen, T. Gass, and R. Schluter, "Equivalence of generative and log-linear models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 5, pp. 1138–1148, 2011.
- [12] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002, pp. 841–848.
- [13] C. M. Bishop, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.
- [14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
- [15] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6655–6659.
- [16] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted mmi for model and feature-space discriminative training," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 4057–4060.
- [17] R. A. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 661–664.
- [18] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks." in *Interspeech*, 2013, pp. 2345–2349.
- [19] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.
- [20] Y. Lu and F. Salem, "Simplified gating in long short-term memory (lstm) recurrent neural networks," *arXiv preprint arXiv:1701.03441*, 2017.
- [21] Z. Tang, Y. Shi, D. Wang, Y. Feng, and S. Zhang, "Memory visualization for gated recurrent neural networks in speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2736–2740.
- [22] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural computation*, vol. 2, no. 4, pp. 490–501, 1990.