# CONTRASTIVE AUTO-ENCODER FOR PHONEME RECOGNITION

*Xin Zheng*[⋆†], *Zhiyong Wu*[⋆†‡], *Helen Meng*[⋆‡] *and Lianhong Cai*[⋆†]

[⋆] Tsinghua-CUHK Joint Research Center for Media Sciences, Technologies and Systems
Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China
[†] Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[‡] Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, China
zhengx11@mails.tsinghua.edu.cn, zywu@sz.tsinghua.edu.cn,
hmmeng@se.cuhk.edu.hk, clh-dcs@tsinghua.edu.cn

**ABSTRACT**

Speech data typically contains task irrelevant information lying within features. Specifically, phonetic information, speaker characteristic information, emotional information and noise are always mixed together and tend to impair one another for certain task. We propose a new type of auto-encoder for feature learning called contrastive auto-encoder. Unlike other variants of auto-encoders, contrastive auto-encoder is able to leverage class labels in constructing its representation layer. We achieve this by modeling two auto-encoders together and making their differences contribute to the total loss function. The transformation built with contrastive auto-encoder can be seen as a task-specific and invariant feature learner. Our experiments on TIMIT clearly show the superiority of the feature extracted from contrastive auto-encoder over original acoustic feature, feature extracted from deep auto-encoder, and feature extracted from a model that contrastive auto-encoder originates from.

***Index Terms***— auto-encoder, contrastive auto-encoder, phoneme recognition, deep neural network (DNN), restricted Boltzmann machine (RBM)

## 1. INTRODUCTION

Speech is one of the most natural way of communication for human being and various kinds of techniques and applications have been developed. However, feature selection for speech signal has been an unsettled problem for decades. For many speech applications, typical features such as Mel-frequency cepstral coefficient (MFCC) and perceptual linear prediction (PLP) were used. One of the most obvious problems for these classical features is that they might contain lots of information that are irrelevant to specific tasks and might impair the performance of particular application.

Auto-encoder (AE) is a neural network which has three layers and its third layer (output) is trained to be as like first layer (input) as possible and thus has the same number of neurons. The output of second layer acts as a compact representation or "code" for the input data. The function of AE is much like principal component analysis (PCA) but AE works in a non-linear fashion. The idea of AE was extended to several other variants such as deep AE [1], sparse AE, denoising AE [2] and contractive AE [3]. All of these ideas have been formalized and successfully applied to various applications [4] and taken an important part of deep learning.

Recently, Chen and Salman [5][6][7] proposed an deep neural architecture (DNA) for modeling speakers in speaker recognition. The idea is to exploit the equivalents and distinctions of training data to build a transformation of features that is most invariant to what we are interested in. Specifically, in the task of speaker verification, original acoustic features may contain some information that is irrelevant to speaker characteristics such as phonetic information and background noise. If we are able to build certain transformation that makes the representation obtained contain information only related to speaker characteristics, a speaker recognition system built from such representation can be counted on to give better performance. So they build their DNA with two deep AEs but simultaneously optimizes both the performance of two AEs and the interaction between them. By optimizing the loss function given by Chen, the DNA is able to build certain transformation that we concern most from data (eg., speaker-specific characteristic) and has been proved to give better results on the task of speaker recognition than using Gaussian mixture model (GMM) or convolutional deep belief network.

We suspect that speech recognition may be facing a similar situation and a transformation that can help get rid of speaker characteristics or background noise may potentially be beneficial. Hence, we propose a new type of AE called contrastive auto-encoder (CsAE), which is an improvement of Chen's DNA and also consists of two AEs. The CsAE modifies the loss function of Chen's DNA in order to make the representation learned more suitable for classification. Also, our model can be seen as an AE that automatically learns invariant transformations.

We use the CsAE on TIMIT for the task of phoneme recognition. The original acoustic feature was transformed through a trained CsAE and recognition performance was compared with original acoustic feature, feature extracted through a deep AE with same architecture and feature extracted with Chen's DNA. The results clearly show the superiority of our model.

## 2. CONTRASTIVE AUTO-ENCODER

### 2.1. Model Description

A CsAE consists of two deep AEs. However, unlike AE and other variants of AEs, the training of CsAE must be accomplished in a pure supervised fashion. Each pair of inputs for a CsAE consists of two samples that belong to a same class and may or may not be the same sample. Each sub-autoencoder (sub-AE) has $K$ layers for encoding and $K$ layers for decoding and thus making $2K+1$ layers altogether. The outputs of the $K$th layer of both sub-AEs are contrasted, which means the difference of them contributes to the loss function. We would like such difference to be as small as possible, yet maintain the ability of both sub-AEs to reconstruct the original input signal.

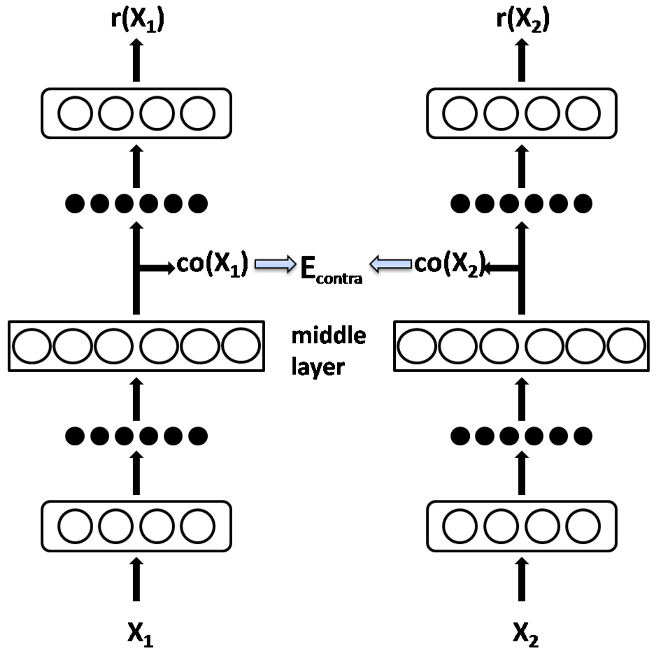

**Fig. 1**. An illustration of contrastive auto-encoder, where middle layer means the $K$th layer given each sub-AE has $2K+1$ layers. $r(X)$ and $co(x)$ are defined in section 2.2.

## 2.2. Loss Function

The loss function of a CsAE is a weighted sum:

$$\begin{aligned} E(X_1, X_2; W) = \alpha[E_1(X_1; W) + E_2(X_2; W)] \\ +(1-\alpha)E_{contra}(X_1, X_2; W) \end{aligned} \quad (1)$$

$X_1$ and $X_2$ in (1) represent two input feature vectors for the two sub-AEs of CsAE respectively but with one condition - they are different in general, but same in certain sense we concern. Concretely, for the task of phoneme recognition, $X_1$ and $X_2$ are different occurrences of a same phoneme. In section 3 we will elaborate how we arrange our training cases to make this requisition satisfied.

The first part of loss function $E$ is the sum of squared $l^2$-norm of reconstruction error of each sub-autoencoder. That is,

$$E_i(X_i; W) = \|r(X_i) - X_i\|_2^2, \quad i = 1, 2 \quad (2)$$

Here $r(X)$ represents the reconstruction of the original feature with an auto-encoder. The optimization of this part guarantees the "code" to be a reasonable representation of original feature. The second part of loss function $E$ is the contrastive part and optimization of this part leads to similar representation

$$E_{contra}(X_1, X_2; W) = \|co(X_1) - co(X_2)\|_2^2 \quad (3)$$

where $co(X)$ means comparative output of input signal $X$. It is the same with the output of the $K$th layer.

Intuitively, penalization of term (3) endows the traditional AE with an additional capability of making the transformed representation similar to each other within classes and thus hopefully making the classification later much easier. This term also differs from Chen's model and we will discuss this point in section 2.4.

## 2.3. Training Algorithm

We found in our experiments that directly optimizing the loss function (1) is difficult even with L-BFGS [8]. So in practice, similar to typical deep AE training strategy [1], the CsAEs are trained with a two-phase learning algorithm. Before elaborating this algorithm, we first make clear our notations used for the algorithm.

For each original input feature $X$ of a sub-AE, let $h_{kj}(X)$ denote the output of $j$th neuron in the $k$th layer for $k = 0, 1, 2, ..., 2K$. $\mathbf{h}_k(X) = (h_{kj}(X))_{j=1}^{|h_k(X)|}$ is a column vector formed with all the neurons in layer $k$. $|h_k(X)|$ is the number of neurons in layer $k$. With this notation, $\mathbf{h}_0(X) = X$, and $\mathbf{h}_{2K}(X) = r(X)$ is the reconstruction of feature $X$ through an AE and $co(X) = h_K(X) = (h_{Kj}(X))_{j=1}^{|h_K(X)|}$. Let $W_k$ denote the weight connections between layer $k-1$ and layer $k$, and $b_k$ denote the bias vector in layer $k$. So,

$$\mathbf{h}_k(X) = \sigma[\mathbf{u}_k(X)], \quad k = 1, 2, \ldots, 2K-1 \quad (4)$$

in which

$$\begin{aligned} \mathbf{u}_k(X) &= W_k'\mathbf{h}_{k-1}(X) + \mathbf{b}_k \quad (5) \\ \sigma(\mathbf{z}) &= \left(\frac{1}{1+e^{z_j}}\right)_{j=1}^{|\mathbf{z}|} \quad (6) \end{aligned}$$

We use notation $'$ to represent the transpose operation throughout this paper.

### *2.3.1. Pre-training*

Pre-training of a CsAE is essentially the training of two deep auto-encoders, and each consists of two steps: generative pre-training of a deep belief network (DBN) and discriminative fine-tuning as a deep neural network (DNN). The pre-training phase of the CsAE aims to minimizing the first part of loss function $E$.

A DBN is a probabilistic model which consists of many hidden layers. DBN can be trained with stacked restricted Boltzmann machines (RBM). One important property for DBN is that adding one more hidden layer guarantees to improve the variational bound of distribution of data it is modeling.

An RBM can be efficiently trained with contrastive divergence algorithm [9]. After training an RBM the inferred data of hidden layer can be used as newly observable data and another RBM can be trained with these data. This procedure can be repeated until the DBN was built.

Suppose we train a DBN with $K+1$ layers, and the $K$ weight matrices are denoted with $W_1, ..., W_K$. All the weight matrices in an AE can be initialized in the following way: for the bottom $K+1$ layers, $W_1, ..., W_K$ can be used; for the upper $K$ layers, we have $W_{K+i} = W'_{K-i+1}$ for $i = 1, ..., K$.

After initializing the AE with $2K+1$ layers, this DNN can be discriminatively trained with back-propagation algorithm to minimize the sum of square of $l^2$-norm of the difference between reconstruction vector and input vector.

### 2.3.2. *Contrastive Fine-tuning*

After the training of two sub-AEs, we use back-propagation algorithm with stochastic gradient descent (SGD) to minimize the loss function (1) all together. Suppose all the training cases are randomly grouped into batches with size $T$, and each case in a batch is a pair like $(X_1(t), X_2(t))$ indexed with $t$. Notice that $X_1(t)$ and $X_2(t)$ must have the same label (the label itself, however, is not used in the training of the CsAE). We derive the updating algorithm by analyzing the effect of one particular case.

Minimizing loss function $E$ can be obtained by minimizing $E_i$ and $E_{contra}$ separately. First, take the derivative of $E_i$ with respect to the output of last layer

$$\frac{\partial E_i(t)}{\partial \mathbf{u}_{2K}^{(i)}(t)} = -2(X_i(t) - r(X_i(t))) \tag{7}$$

For all the other layers, that is, $k = 2K-1, ..., 2, 1$

$$\frac{\partial E_i(t)}{\partial \mathbf{u}_k^{(i)}(t)} = \left( \frac{\partial E_i(t)}{\partial h_{kj}^{(i)}(t)} h_{kj}^{(i)}(t) \left(1 - h_{kj}^{(i)}(t)\right) \right)_{j=1}^{|h_k^i(t)|} \tag{8}$$

$$\frac{\partial E_i(t)}{\partial \mathbf{h}_k^{(i)}(t)} = [W_{k+1}^i]' \frac{\partial E_i(t)}{\partial \mathbf{u}_{k+1}^i(t)} \tag{9}$$

For the middle layer where $k = K$, the derivative can be derived with

$$\frac{\partial E_{contra}(t)}{\partial \mathbf{u}_k^{(i)}(t)} = 2 \cdot (-1)^{i-1} \left(co_j(X_1(t))(1 - co_j(X_2(t)))\right)_{j=1}^{|h_K^i(t)|} \tag{10}$$

For $k = K-1, ..., 2, 1$

$$\frac{\partial E_{contra}(t)}{\partial \mathbf{u}_k^{(i)}(t)} = \left( \frac{\partial E_{contra}(t)}{\partial h_{kj}^{(i)}(t)} h_{kj}^{(i)}(t) \left(1 - h_{kj}^{(i)}(t)\right) \right)_{j=1}^{|h_k^i(t)|} \tag{11}$$

$$\frac{\partial E_{contra}(t)}{\partial \mathbf{h}_k^{(i)}(t)} = [W_{k+1}^i]' \frac{\partial E_{contra}(t)}{\partial \mathbf{u}_{k+1}^i(t)} \tag{12}$$

So, from stochastic gradient descent algorithm, for $k = K+1, ..., 2K$, the updating rule of each weight is

$$W_k^{(i)} \leftarrow W_k^{(i)} - \frac{\epsilon\alpha}{T} \sum_{t=1}^{T} \frac{\partial E_i(t)}{\partial \mathbf{u}_k^{(i)}(t)} [\mathbf{h}_{k-1}^{(i)}(t)]' \tag{13}$$

$$b_k^{(i)} \leftarrow b_k^{(i)} - \frac{\epsilon\alpha}{T} \sum_{t=1}^{T} \frac{\partial E_i(t)}{\partial \mathbf{u}_k^{(i)}(t)} \tag{14}$$

And for layer $k = 1, 2, ..., K$

$$\begin{aligned} W_k^{(i)} \leftarrow W_k^{(i)} - \frac{\epsilon}{T} \sum_{t=1}^{T} \Bigg( \alpha \frac{\partial E_i(t)}{\partial \mathbf{u}_k^{(i)}(t)} \\ + (1-\alpha) \frac{\partial E_{contra}(t)}{\partial \mathbf{u}_k^{(i)}(t)} \Bigg) [\mathbf{h}_{k-1}^{(i)}(t)]' \end{aligned} \tag{15}$$

$$b_k^{(i)} \leftarrow b_k^{(i)} - \frac{\epsilon}{T} \sum_{t=1}^{T} \left( \alpha \frac{\partial E_i(t)}{\partial \mathbf{u}_k^{(i)}(t)} + (1-\alpha) \frac{\partial E_{contra}(t)}{\partial \mathbf{u}_k^{(i)}(t)} \right) \tag{16}$$

$\epsilon$ represents learning rate in (13)(14)(15)(16).

## 2.4. Comparison to Chen's DNA

The essential difference between the CsAE and Chen's model [5][6][7] is that, in Chen's model, only part of neurons in the middle layers of the two sub-AEs contribute to the loss function. By his approach, the activity of both the neurons that contributes to the loss function and other neurons have explicit meanings. For example, half of neurons learn speaker-specific characteristics and the rest half contains phonetic and noise information. However, such model has the tendency to learn a trivial solution that makes all input data mapped to a uniform output, which can make the second part of the loss function minimized. To deal with this problem, Chen introduced his loss function as:

$$\begin{aligned} E(X_1, X_2; W) = \alpha[E_1(X_1; W) + E_2(X_2; W)] \\ +(1-\alpha) E_D(X_1, X_2; W) \end{aligned} \tag{17}$$

in which $E_1, E_2$ is the same as in (2), and $E_D$ is defined as:

$$\begin{aligned} E_D(X_1, X_2; W) = Y \|co'(X_1) - co'(X_2)\|_2^2 \\ +(1-Y) e^{-\|co'(X_1) - co'(X_2)\|_2^2} \end{aligned} \tag{18}$$

in which $co'(X)$ denotes the output of the part of neurons (half, for example) we would like to be similar between two sub-AEs. $Y = 1$ if $X_1$ and $X_2$ belongs to a same class and $Y = 0$ otherwise. Thus, the negative exponential penalization makes samples that belong to different class far from each other, but not so aggressive as to make different samples that are already far from each other even farther, hence avoids the trivial solution mentioned above.

While such solution have been proved to be effective in modeling speakers in speaker recognition, Chen's DNA still offers room for improvement. With the loss function given by Chen, the model may try to learn another kind of trivial solution: all the data from the same class are mapped to a same output (minimizing the first term of 18) and meanwhile totally different from the output that mapped from other classes (minimizing the second term of 18). This seems to be what we would like to achieve, however, this can almost always be achieved, because the reconstruction of original data can always be aided with other neurons in the middle layer. Thus the output of neurons that we concern may lose the discriminative power for certain task, since the representation obtained can have nothing to do with original signal. In other words, this is a kind of overfitting. Instead of using this term to describe the discrepancy between how well the model fits training data and testing data, we are actually describing the discrepancy between those part of neurons (and weights that connect to these neurons) that are endowed with special meaning and those who do not, and such kind of overfitting can occur regardless of whether the model overfits the training data. Thus, the way the model is optimized implicitly signifies a degradation of performance if the feature extracted is used for classification.

Our model deals with both problems simultaneously. First, a trivial solution like all the inputs be mapped to a uniform output is impossible because such solution can never reconstruct the inputs well. So the negative exponential penalization for negative samples is not necessary in the CsAE. Second, for the same reason, a trivial solution that samples from each class are squeezed to a single point respectively is not possible because they cannot reconstruct the original data very well.

## 3. EXPERIMENTS

### 3.1. Experimental Setup

All experiments are conducted on TIMIT corpus. All SA records are removed from the data set. The data was divided into training, development and core test as in [10]. We used 40 coefficients Mel-scale log filter-bank (with energy) and velocity and acceleration (123 dimensions) as acoustic feature and 11 consecutive frames of acoustic feature are concatenated as input for baseline mono-phone hidden Markov model (HMM)- DNN phoneme recognition system, CsAE and Chen's DNA. The feature extracted by a single deep AE, CsAE and Chen's DNA are also used in a HMM-DNN phoneme recognition system for comparison. We used the feature from the middle layer right before the nonlinear activation was applied. All data used in neural networks (DNN, deep AE, CsAE and Chen's DNA) are normalized to zero mean and unit variance. The HMM was obtained by training a HMM-GMM model with maximum likelihood criterion.

During the decoding stage, we used a bigram phoneme language model trained from the training set and a same decoding parameters. For evaluation, 61 phones are mapped to 39 phonemes in the same way as in [11].

### 3.2. Configuration of neural networks

All DNNs in our experiments were pre-trained with stacked RBMs. We trained RBMs in the same way as in [12]. For all the DNNs in HMM-DNN phoneme recognition system, a same architecture of 6 hidden layers by 2000 units was applied. The DNNs were trained with cross-entropy criterion and SGD+momentum. All the other hyper-parameters were the same with [12].

All sub-AEs of Chen's DNA and the CsAE were also pre-trained with RBMs. For convenience of comparison, we used hidden layer sizes of [1500,1500,1500,3000,1500,1500,1500] for sub-AEs in Chen's model and [1500,1500,1500,1500,1500,1500,1500] for sub-AEs in CsAE. We used half of the neurons in the middle layer of Chen's model to optimize the loss function, which made the dimension of feature obtained from Chen's DNA and the CsAE both 1500.

When organizing the training data, since Chen's model have to use negative samples (samples that have different labels for two sub-AEs), we first randomized all the training samples within label, and then randomized half of training data irrespective their labels , which made almost half of training samples negative for Chen's model. For CsAE, we only had to randomize training samples within label.

### 3.3. Results

We trained a Chen's DNA, a deep AE and a CsAE with the Mel-scale log filter-bank (fbank) feature as input. We first trained two deep AEs with configuration described in section 3.2 (denoted as $AE_A$ and $AE_B$) and duplicated them as pre-training of Chen's DNA (denoted as $AE_C$) and CsAE (denoted as $AE_D$) respectively. Notice that training a deep auto-encoder has been recognized as a difficult problem and normally the models suffer from underfitting instead of overfitting [13]. For this reason, training deep AE has become a benchmarking problem [14][15][13]. To make such huge AEs give satisfactory performance, it took us a month to train each of them on a GPU. To make a fair comparison, the reconstruction error of $AE_A$ equaled 91% of reconstruction error of $AE_B$ when we stopped training them. After that, $AE_C$ and $AE_D$ were trained by following objective (17) and (1), and $AE_B$ was kept on training (denoted as $AE_E$) separately alongside with $AE_C$ and $AE_D$. Each of these three AEs was trained for 3000 epochs following their own objective with learning rate of 0.001 and $\alpha = 0.75$ in both (1) and (17).

Ideally, each sample should be contrasted with all the other samples in the same class in the transformed feature, and this can be approximated by randomizing all the training data within each class on-the-fly. Although we did not use this method, and a single randomization of whole training set clearly cannot satisfy such requirement, we perform a little test to make sure if this would be a problem. We tried to change the training set to another re-randomized training set in our process of training a contrastive auto-encoder. After such change, we observed no perceptible oscillation of both training error and validation error, which in a certain degree validates that the amount of data in TIMIT is sufficient for CsAE.

All results are reported in Table 1. From our experiments, features obtained from Chen's DNA was not able to outperform original fbank feature. We checked the outputs of neurons of the half of middle layer (after nonlinear activation) and found that most neurons are firmly on or off, and these outputs tend to be almost the same within each class, which demonstrate the validity of our reasoning in section 2.4. However, there's still a chance that Chen's model has the ability to disentangle the explanatory factors of original data. So we also tested the feature which comes from all the neurons in the middle layer, but the result is still not good enough.

**Table 1**. Phone error rate of different features

| feature | dev | core test |
|---|---|---|
| fbank | 21.97% | 23.26% |
| Chen's DNA | 22.80% | 24.77% |
| Chen's DNA with all neurons | 23.50% | 24.25% |
| deep AE | 22.39% | 22.91% |
| CsAE | 21.26% | 22.80% |
| fbank+CsAE | 21.00% | 22.20% |

The difference between fbank feature and feature extracted by an deep AE is not significant. But with the additional term that contrast middle layer outputs in loss function (1), the phone error rate (PER) of feature extracted from CsAE improved by more than 1% on development set compared with feature from deep AE. We also concatenate fbank feature with feature extracted by CsAE (fbank+CsAE) and obtained further improvements on both development set and core test set.

## 4. CONCLUSION

In this paper, we propose a new variant of auto-encoder called contrastive auto-encoder. We present the definition and learning algorithm of contrastive auto-encoder. It improves the model by Chen and Salman. We carefully analyze Chen's model and the difference between their model and ours is given. Our experiments on TIMIT verify our analysis. In the future, we would like to apply contrastive auto-encoder to other classification problems.

## 5. ACKNOWLEDGEMENTS

This work is supported by the National Basic Research Program of China (2012CB316401, 2013CB329304), the National Natural Science Foundation of China (61375027, 61370023), the Upgrading Plan Project of Shenzhen Key Laboratory (CXB201005250038A) and the Science and Technology R&D Funding of the Shenzhen Municipal.

## 6. REFERENCES

[1] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[2] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning (ICML'2008)*, 2008.

[3] X. Muller X. Glorot Y. Bengio S. Rifai, P. Vincent, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on Machine Learning (ICML'2011)*, 2011.

[4] X Glorot, A. Bordes, and Y., "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28 th International Conference on Machine Learning*, 2011.

[5] K. Chen and A. Salman, "Learning speaker-specific characteristics with a deep neural architecture," *IEEE Transactions on Neural Networks*, vol. 22(11), pp. 1744–1756, 2011.

[6] K. Chen and A. S, "Exploring speaker-specific characteristic-s with deep learning," in *Proceedings of International Joint Conference on Neural Networks*, 2011.

[7] A. Salman and K. Chen, "Extracting speaker-specific information with a regularized siamese deep network," in *Advances in Neural Information Processing Systems*, 2011.

[8] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A. Ng, and Q. V. Le, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 265–272.

[9] G. E. Hinton, "Training products of experts by minimizingc ontrastive divergence," *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[10] A. Halberstadt, *Heterogeneous measurements and multiple classifiers for speech recognition*, Ph.D. thesis, MIT, 1998.

[11] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.

[12] X. Zheng, Z. Wu, B. Shen, H. Meng, and L. Cai, "Investigation of tandem deep belief network approach for phoneme recognition," in *Proc. of ICASSP*, 2013.

[13] I. Sutskever, J. Martens, G. Dahl, and G. H, "On the importance of momentum and initialization in deep learning," in *Proceedings of the 30th International Conference on Machine Learning (ICML'2013)*, 2013.

[14] James Martens and Ilya Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1033–1040.

[15] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.