Deep Maxout Networks applied to Noise-Robust Speech Recognition

F. de-la-Calle-Silos, A. Gallardo-Antolín, C. Peláez-Moreno

Department of Signal Theory and Communications.
Universidad Carlos III de Madrid.
Leganés (Madrid), Spain.

November, 20th, 2014

Outline

Introduction

1. Deep Neural Networks (DNN) have become very popular for acoustic modeling due to the improvements found over traditional Gaussian Mixture Models (GMM).
2. Not many works have addressed the robustness of these systems under noisy conditions.
3. New methods to improve the accuracy of DNNs by using techniques such as dropout and maxout.
4. Experiments on TIMIT (both clean and noisy conditions) show improvements.

Introduction

1. Deep Neural Networks (DNN) have become very popular for acoustic modeling due to the improvements found over traditional Gaussian Mixture Models (GMM).
2. Not many works have addressed the robustness of these systems under noisy conditions.
3. New methods to improve the accuracy of DNNs by using techniques such as dropout and maxout.
4. Experiments on TIMIT (both clean and noisy conditions) show improvements.

Introduction

1. Deep Neural Networks (DNN) have become very popular for acoustic modeling due to the improvements found over traditional Gaussian Mixture Models (GMM).
2. Not many works have addressed the robustness of these systems under noisy conditions.
3. New methods to improve the accuracy of DNNs by using techniques such as dropout and maxout.
4. Experiments on TIMIT (both clean and noisy conditions) show improvements.

Introduction

1. Deep Neural Networks (DNN) have become very popular for acoustic modeling due to the improvements found over traditional Gaussian Mixture Models (GMM).
2. Not many works have addressed the robustness of these systems under noisy conditions.
3. New methods to improve the accuracy of DNNs by using techniques such as dropout and maxout.
4. Experiments on TIMIT (both clean and noisy conditions) show improvements.

Why Deep Neural Networks?

1. Improved performance.
2. DNNs have a larger number of hidden layers leading to systems with many more parameters than the later:
   - ☺ Less influenced by training and testing mismatch.
   - ☹ Can easily suffer from overfitting: pretraining, dropout, maxout.
3. Hybrid ANN/HMM architectures [Bourlard and Morgan, 1994]:
   - Usually models senones (tied states) directly (although there might be thousands of senones).
   - Longer context windows, less restrictions on input features.

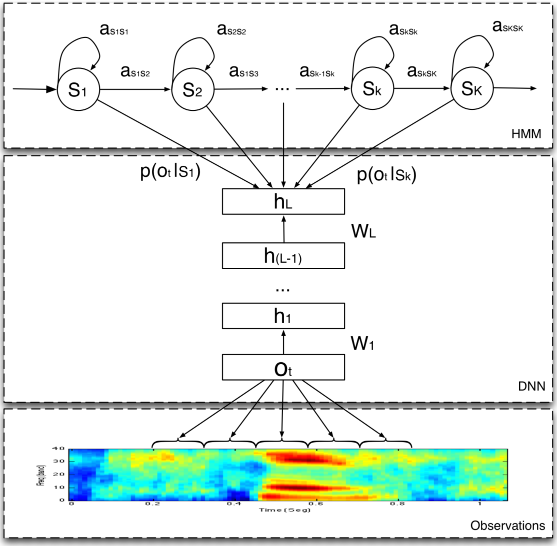
Why Deep Neural Networks?

1. Improved performance.
2. DNNs have a larger number of hidden layers leading to systems with many more parameters than the later:
   - ☺ Less influenced by training and testing mismatch.
   - ☹ Can easily suffer from overfitting: pretraining, dropout, maxout.
3. Hybrid ANN/HMM architectures [Bourlard and Morgan, 1994]:
   - Usually models senones (tied states) directly (although there might be thousands of senones).
   - Longer context windows, less restrictions on input features.

Why Deep Neural Networks?

1. Improved performance.
2. DNNs have a larger number of hidden layers leading to systems with many more parameters than the later:
   - ☺ Less influenced by training and testing mismatch.
   - ☹ Can easily suffer from overfitting: pretraining, dropout, maxout.
3. Hybrid ANN/HMM architectures [Bourlard and Morgan, 1994]:
   - Usually models senones (tied states) directly (although there might be thousands of senones).
   - Longer context windows, less restrictions on input features.

ASR Hybrid Model



Deep Neural Networks

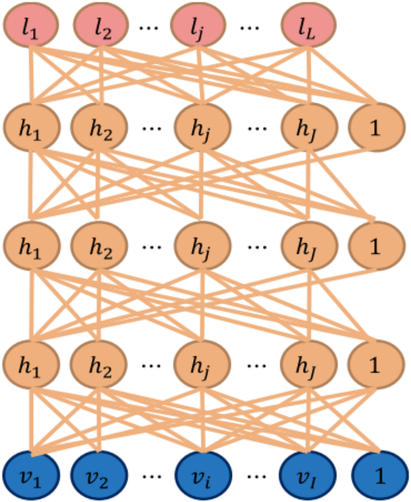


Figure from [Deng and Yu, 2014].

Outline

Avoiding overfitting

The overfitting problem

Reduce the chance that error back-propagation algorithm falls into a poor local minimum.

1. Addition of a pre-training stage.
2. Dropout: randomly omit hidden units in the training stage.
3. Deep Maxout Networks (DMNs): split hidden units at each layer into non-overlapping groups, each of them generating an activation using a max pooling operation.

Avoiding overfitting

The overfitting problem

Reduce the chance that error back-propagation algorithm falls into a poor local minimum.

1. Addition of a pre-training stage.
2. Dropout: randomly omit hidden units in the training stage.
3. Deep Maxout Networks (DMNs): split hidden units at each layer into non-overlapping groups, each of them generating an activation using a max pooling operation.

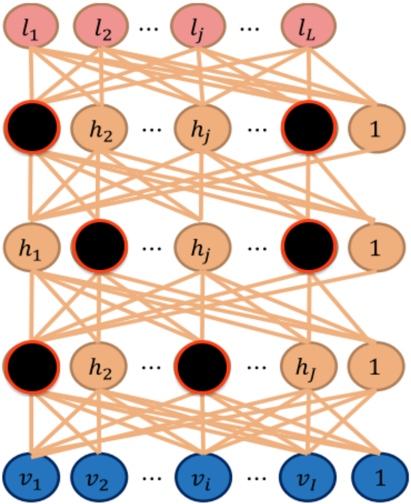
Avoiding overfitting

The overfitting problem

Reduce the chance that error back-propagation algorithm falls into a poor local minimum.

1. Addition of a pre-training stage.
2. Dropout: randomly omit hidden units in the training stage.
3. Deep Maxout Networks (DMNs): split hidden units at each layer into non-overlapping groups, each of them generating an activation using a max pooling operation.

Dropout



Modified from Figure from [Deng and Yu, 2014].

Dropout

Randomly omitting a certain percentage of the hidden units on each training iteration

$$\mathbf{h}^{(l+1)} = m^{(l)} \star \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)}\right), \quad 1 \leq l \leq L \qquad (1)$$

where $m^{(l)}$ is a binary vector of the same dimension of $\mathbf{h}^{(l)}$ whose elements are sampled from a Bernoulli distribution with probability p: Hidden Drop Factor (HDF) .

1. Only applied in the training stage whereas on testing all the hidden units become active.
2. Can be seen as an ensemble of DNNs.
3. Similar to bagging.

Dropout

Randomly omitting a certain percentage of the hidden units on each training iteration

$$\mathbf{h}^{(l+1)} = m^{(l)} \star \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)}\right), \quad 1 \leq l \leq L \qquad (1)$$

where $m^{(l)}$ is a binary vector of the same dimension of $\mathbf{h}^{(l)}$ whose elements are sampled from a Bernoulli distribution with probability p: Hidden Drop Factor (*HDF*) .

1. Only applied in the training stage whereas on testing all the hidden units become active.
2. Can be seen as an ensemble of DNNs.
3. Similar to bagging.

Dropout

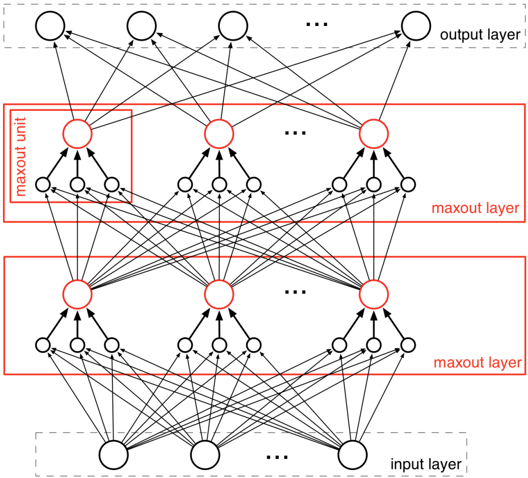
Randomly omitting a certain percentage of the hidden units on each training iteration

$$\mathbf{h}^{(l+1)} = m^{(l)} \star \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)}\right), \quad 1 \leq l \leq L \qquad (1)$$

where $m^{(l)}$ is a binary vector of the same dimension of $\mathbf{h}^{(l)}$ whose elements are sampled from a Bernoulli distribution with probability p: Hidden Drop Factor (*HDF*) .

1. Only applied in the training stage whereas on testing all the hidden units become active.
2. Can be seen as an ensemble of DNNs.
3. Similar to bagging.

Maxout



A Maxout Network of 2 hidden layers and a group size of $g = 3$.
The hidden nodes in red perform the max operation.

Maxout (DMN)

Each hidden unit takes the maximum value over the g units of a group

$$h_i^{(l+1)} = \max_{j \in 1,\ldots,g} z_{ij}^{(l+1)}, \quad 1 \leq l \leq L \tag{2}$$

where $z_{ij}^{(l+1)}$ is the lineal pre-activation values from the l layer:

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)} \tag{3}$$

1. DMNs reduce the number of parameters over DNNs: the weight matrix of each layer i is $1/g$ of its equivalent DNN.
2. Theoretical result: the capability of maxout units to approximate any convex function.

Maxout (DMN)

Each hidden unit takes the maximum value over the g units of a group

$$h_i^{(l+1)} = \max_{j \in 1,\ldots,g} z_{ij}^{(l+1)}, \quad 1 \leq l \leq L \tag{2}$$

where $z_{ij}^{(l+1)}$ is the lineal pre-activation values from the l layer:

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)} \tag{3}$$

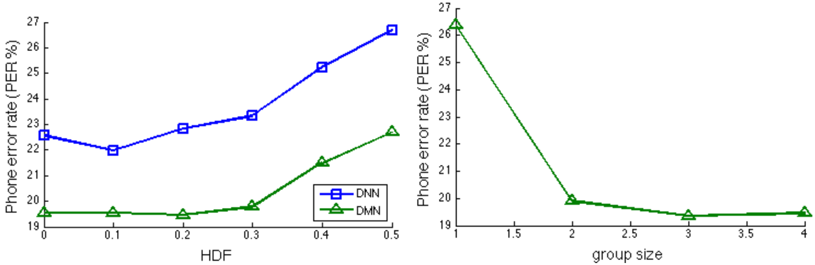
1. DMNs reduce the number of parameters over DNNs: the weight matrix of each layer i is $1/g$ of its equivalent DNN.
2. Theoretical result: the capability of maxout units to approximate any convex function.

Outline

Results: corpus

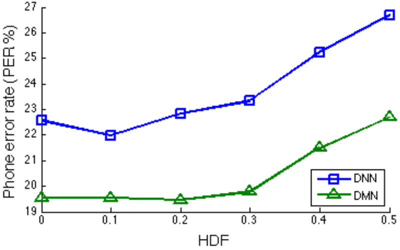
- Experiments were performed on the TIMIT corpus.
- 462 speakers training set, 50 speakers development set for tuning.
- Results are reported using the 24-speaker core test set.
- Added noise (white, street, music and speaker) using FANT.
- Kaldi toolkit: GMM-HMM.
- Kaldi + PDNN toolkit: DNN-HMM.

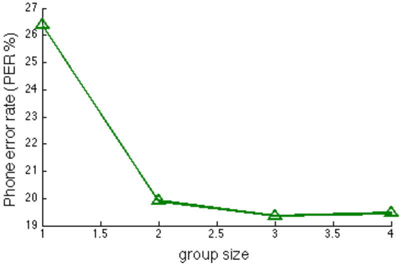
Results: tunning HDF and g



1. PER as a function of HDF and Group size g.
2. Both DNN and DMN have 5 layers.
3. Selected HDR=0.1 for DNN, HDR=0.2 and g=3 for DMN.

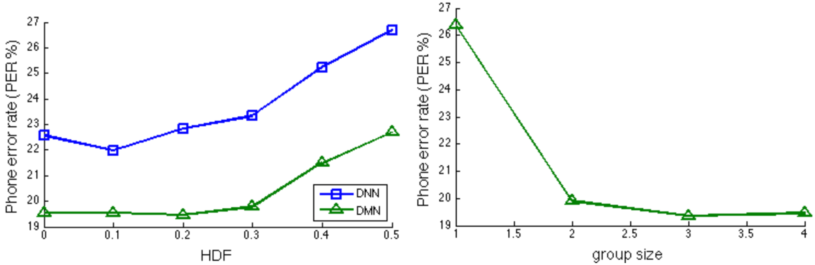
Results: tunning HDF and g





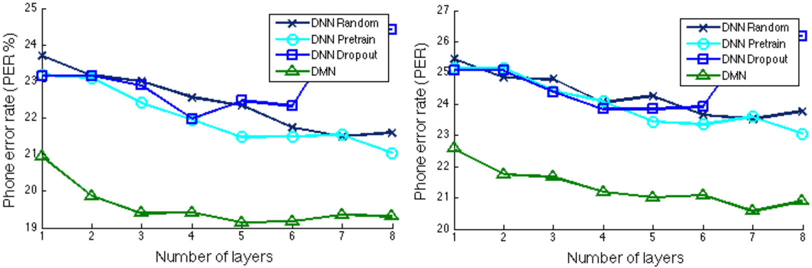
1. PER as a function of HDF and Group size g.
2. Both DNN and DMN have 5 layers.
3. Selected HDR=0.1 for DNN, HDR=0.2 and g=3 for DMN.

Results: tunning HDF and g



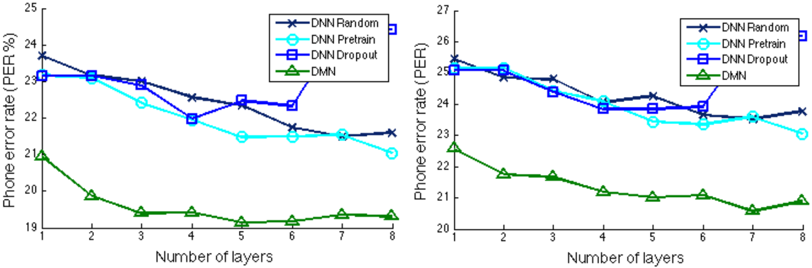
1. PER as a function of HDF and Group size g.
2. Both DNN and DMN have 5 layers.
3. Selected HDR=0.1 for DNN, HDR=0.2 and g=3 for DMN.

Results: tunning the number of layers



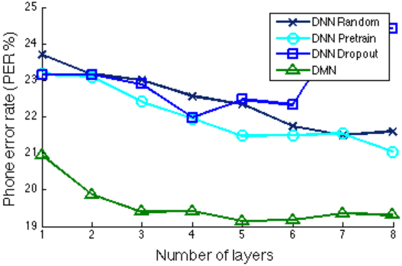
1. Number of hidden nodes: 1024.
2. 400 maxout units: 400x3 = 1200 hidden nodes.
3. DMM outperform the others.

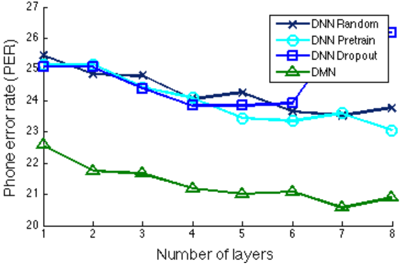
Results: tunning the number of layers



1. Number of hidden nodes: 1024.
2. 400 maxout units: $400x3 = 1200$ hidden nodes.
3. DMM outperform the others.

Results: tunning the number of layers





1. Number of hidden nodes: 1024.
2. 400 maxout units: $400x3 = 1200$ hidden nodes.
3. DMM outperform the others.

Results: baseline GMM-HMM comparisons

Method	Dev PER %	Eval PER %
Monophone	33.33	34.30
Triphone	28.64	30.42
Triphone + LDA + MLLT	26.44	27.62
Triphone + LDA + MLLT + SAT	23.56	25.79
DNN with random initialization (7 layers)	21.50	23.53
DNN with pretraining (8 layers)	21.05	23.05
DNN with dropout (4 layers)	21.98	23.84
DMN (5 layers)	19.15	21.01

1. Recognition results in terms of PER(%) for the TIMIT development and core test sets in clean conditions.
2. DNN-HMM outperform GMM-HMM.
3. DMN outperforms DNN.

Results: baseline GMM-HMM comparisons

Method	Dev PER %	Eval PER %
Monophone	33.33	34.30
Triphone	28.64	30.42
Triphone + LDA + MLLT	26.44	27.62
Triphone + LDA + MLLT + SAT	23.56	25.79
DNN with random initialization (7 layers)	21.50	23.53
DNN with pretraining (8 layers)	21.05	23.05
DNN with dropout (4 layers)	21.98	23.84
DMN (5 layers)	19.15	21.01

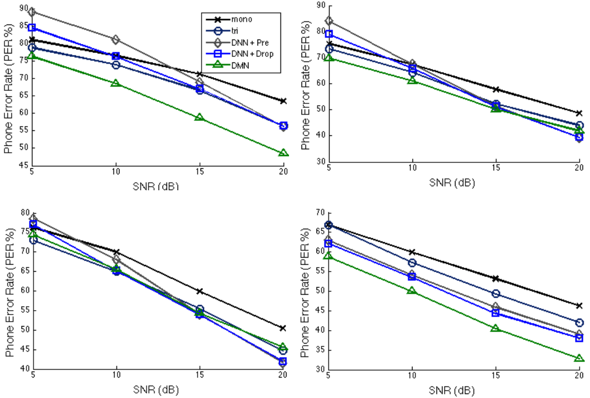
1. Recognition results in terms of PER(%) for the TIMIT development and core test sets in clean conditions.
2. DNN-HMM outperform GMM-HMM.
3. DMN outperforms DNN.

Results: baseline GMM-HMM comparisons

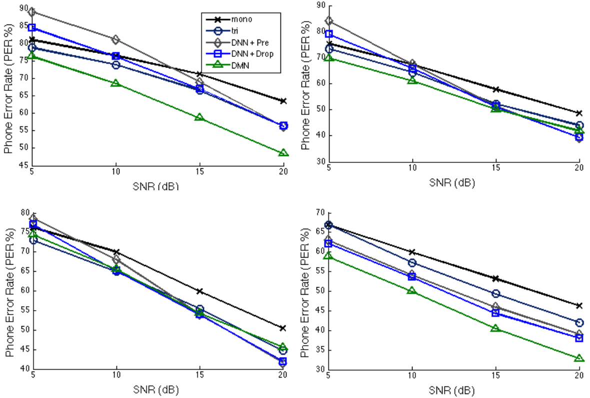
Method	Dev PER %	Eval PER %
Monophone	33.33	34.30
Triphone	28.64	30.42
Triphone + LDA + MLLT	26.44	27.62
Triphone + LDA + MLLT + SAT	23.56	25.79
DNN with random initialization (7 layers)	21.50	23.53
DNN with pretraining (8 layers)	21.05	23.05
DNN with dropout (4 layers)	21.98	23.84
DMN (5 layers)	19.15	21.01

1. Recognition results in terms of PER(%) for the TIMIT development and core test sets in clean conditions.
2. DNN-HMM outperform GMM-HMM.
3. DMN outperforms DNN.

Results: baseline GMM-HMM comparisons



1. PER [%] for test set in noisy conditions: white, street, music, speaker. Best configurations for each system.
2. DMN is always better at low SNR except for *music* where all the systems are similar.



1. PER [%] for test set in noisy conditions: white, street, music, speaker. Best configurations for each system.
2. DMN is always better at low SNR except for *music* where all the systems are similar.

Outline

Conclusions

- Deep Maxout Networks provide improved robustness due to their flexibility in the activation function.
- Future directions: larger databases, other DNN alternatives, analysis of errors.

Bourlard, H. and Morgan, N. (1994).

Connectionist Speech Recognition: A Hybrid Approach.

Kluwer international series in engineering and computer science: VLSI, computer architecture, and digital signal processing. Springer US.

Deng, L. and Yu, D. (2014).

Deep learning: Methods and applications.

Foundations and Trends® in Signal Processing, 7(3–4):197–387.