Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition

IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 20, NO. 1, JANUARY 2012

George E. Dahl, Dong Yu, Senior Member, IEEE, Li Deng, Fellow, IEEE, and Alex Acero, Fellow, IEEE

Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition (CD-DNN-HMMs)

1. Architecture
2. Training Procedure
3. Experimental Results

Architecture

of CD-DNN-HMMs

What makes CD-DNN-HMMs special ?

- Deeper, more expressive NN architectures:
  ⟶ Unsupervised DBN pre-training algorithm.
- Use of posterior probabilities of *senones* as the output.

What is a senone ?

In a nutshell: a tied triphone state.

Transitions between words can bear more information the words themselves

→ *Triphones* or even *Quinphones*

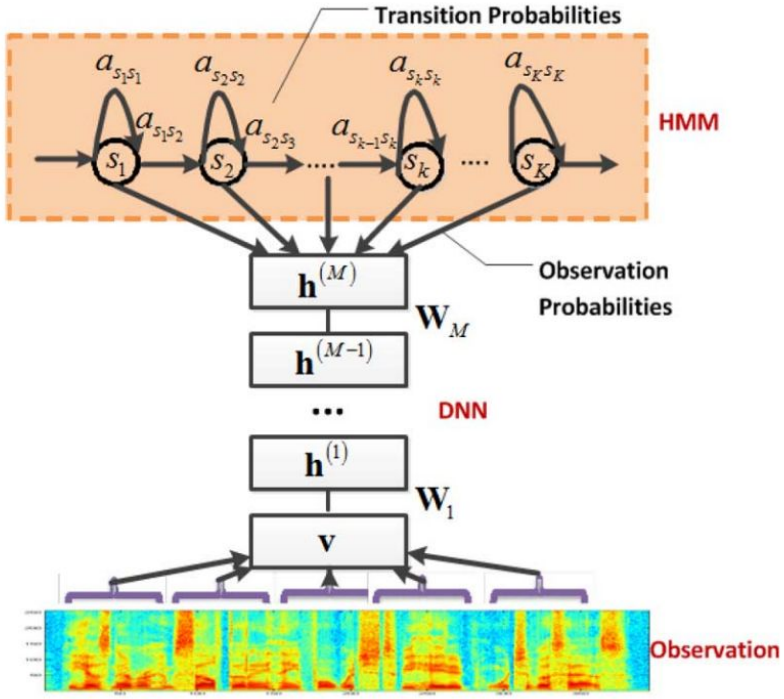
Match same range in waveform as just phones:

→ *Senone*

Prior Probability: Bayes Theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

CD-DNN-HMM Illustration



Decoded Word Sequence

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, p(w \mid \mathbf{x}) = \underset{w}{\operatorname{argmax}}\, p(\mathbf{x} \mid w) p(w) / p(\mathbf{x})$$

Training Procedure

For CD-DNN-HMMs

Algorithmic Main Steps

1. Convert CD-GMM-HMM to CD-DNN-HMM
2. Convert triphone states
3. Pre-Training & state-level alignment
4. Fine-tuning and re-estimating transition probabilities
5. End or iterate

Step 1: Convert CD-GMM-HMM to CD-DNN-HMM

Train a best tied-state CD-GMM-HMM system where state tying is determined based on the data-driven decision tree. Denote the CD-GMM-HMM: *gmm-hmm*.

Parse gmm-hmm and:

1. Give each senone name an ordered *senoneid* starting from 0 (training label for DNN fine-tuning).
2. Generate a mapping from each physical tri-phone state to the corresponding senoneid. Denote this mapping *state2id*.

Step 2: Convert triphone states

Convert gmm-hmm to the corresponding CD-DNN-HMM *dnn–hmm1* by borrowing the tri-phone and senone structure as well as the transition probabilities from *gmm-hmm*.

Step 3: Pre-Training & state-level alignment

Pre-train each layer in the DNN bottom-up layer by layer and call the result: *ptdnn*.

Use *gmm-hmm* to generate state-level alignment: *align-raw*. the training set.

Step 4: Fine-tuning and re-estimating transition probabilities

Convert *align-raw* to *align* where each physical tri-phone state is converted to *senoneid*, then fine-tune the DBN starting from *ptdnn*.

Denote the DBN: *dnn*.

Re-estimate the transition probabilities using dnn and *dnn-hmm1* to maximize the likelihood of observing the features. Denote the new CD-DNN-HMM: *dnn-hmm2*.

Step 5: End or iterate

Exit if no recognition accuracy improvement is observed in the development set; Otherwise use *dnn* and *dnn-hmm2* to generate a new state-level alignment *align-raw* on the training set and go to step 4.

Algorithmic Main Steps: Recap

1. Convert CD-GMM-HMM to CD-DNN-HMM
2. Convert triphone states
3. Pre-Training & state-level alignment
4. Fine-tuning and re-estimating transition probabilities
5. End or iterate

Experimental Results

On CD-GMM-HMMs/ CD-DNN-HMMs

Dataset Description

Bing Mobile voice search application (2008).

Examples of queries:

- “Mc-Donalds”, “Denny’s restaurant”, “oak ridge church.”

Dataset Description (2)

INFORMATION ON THE BUSINESS SEARCH DATASET

	Hours	Number of Utterances
Training Set	24	32,057
Development Set	6.5	8,777
Test Set	9.5	12,758

CD-GMM-HMM Baseline Systems

- Maximum-likelihood (ML)
- Maximum mutual information (MMI)
- Minimum phone error (MPE) criteria

CD-GMM-HMM BASELINE RESULTS

Criterion	Dev Accuracy	Test Accuracy
ML	62.9%	60.4%
MMI	65.1%	62.8%
MPE	65.5%	63.8%

CD-DNN-HMM (Context dependency)

COMPARISON OF CONTEXT-INDEPENDENT MONOPHONE STATE LABELS AND CONTEXT-DEPENDENT TRIPHONE SENONE LABELS

# Hidden Layers	# Hidden Units	Label Type	Dev Accuracy
1	2K	Monophone States	59.3%
1	2K	Triphone Senones	68.1%
3	2K	Monophone States	64.2%
3	2K	Triphone Senones	69.6%

CD-DNN-HMM (Pre-training)

CONTEXT-DEPENDENT MODELS WITH AND WITHOUT PRE-TRAINING

Model Type	# Hidden Layers	# Hidden Units	Dev Accuracy
without pre-training	1	2K	68.0%
without pre-training	2	2K	68.2%
with pre-training	1	2K	68.1%
with pre-training	2	2K	69.5%

CD-DNN-HMM (Hidden Layers)

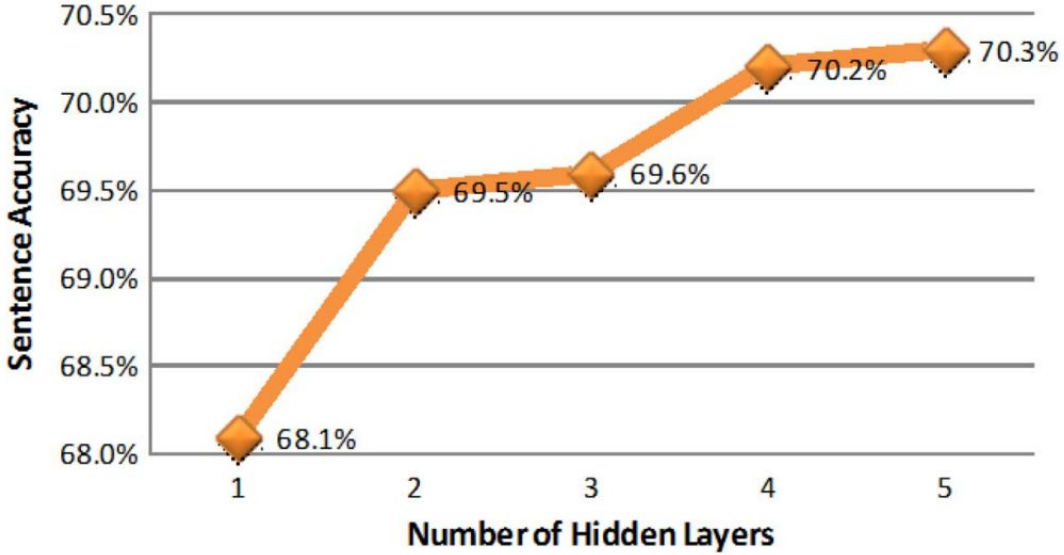


Fig. 2. Relationship between the recognition accuracy and the number of layers. Context-dependent models with 2 K hidden units per layer were used to obtain the results.

CD-DNN-HMM (Hidden Layers)

SUMMARY OF TRAINING TIME USING 24 HOURS OF TRAINING DATA AND 2 K HIDDEN UNITS PER LAYER

Type	# of Layers	Time Per Epoch	# of Epochs
Pre-train	1	0.2 h	50
Pre-train	2	0.5 h	20
Pre-train	3	0.6 h	20
Pre-train	4	0.7 h	20
Pre-train	5	0.8 h	20
Fine-tune	4	1.2 h	12
Fine-tune	5	1.4 h	12

5 layers:
Training → 62 hours
Fine-Tuning → 16.8 hours

CD-DNN-HMM (Decoding Time)

Dell Precision T3500 workstation:

- Quad core
- CPU clock speed: 2.66 GHz
- 8 MB of L3 CPU cache
- 12 GB of 1066 MHz DDR3 SDRAM

CD-DNN-HMM (Decoding Time) (2)

SUMMARY OF DECODING TIME

Processing Unit	# of Layers	DNN Time Per Frame	Search Time Per Frame	Real-time Factor
CPU	4	4.3 ms	1.5 ms	0.58
GPU	4	0.16 ms	1.5 ms	0.17
CPU	5	5.2 ms	1.5 ms	0.67
GPU	5	0.20 ms	1.5 ms	0.17

CD-DNN-HMM (Decoding Time) (2)

- Total time to train the system from scratch is about four days.
- Using a GPU speeds up training by about a factor of 30 faster than just using the CPU.
- Without using a GPU, it would take about three months to train the best system.

CD-DNN-HMM (Bottleneck)

Bottleneck: mini-batch stochastic gradient descend (SGD) algorithm used to train the DNNs:

- Inherently sequential
- Difficult to parallelize across machines.

So far SGD with a GPU is the best training strategy for CD-DNN-HMMs.

→ GPU can exploit the parallelism in the layered DNN structure.

Conclusion

Conclusion

- Training is considerably more expensive than for CD-GMM-HMM systems
- Decoding is still very efficient
- In theory, CD-DNN-HMM training is quite scalable
- In practice, it is quite challenging to train on tens of thousands of hours of data.

→ Finding new ways to parallelize training may require a better theoretical understanding of deep learning