MLSLP-2012

Learning Deep Architectures Using Kernel Modules

Li Deng

Microsoft Research, Redmond

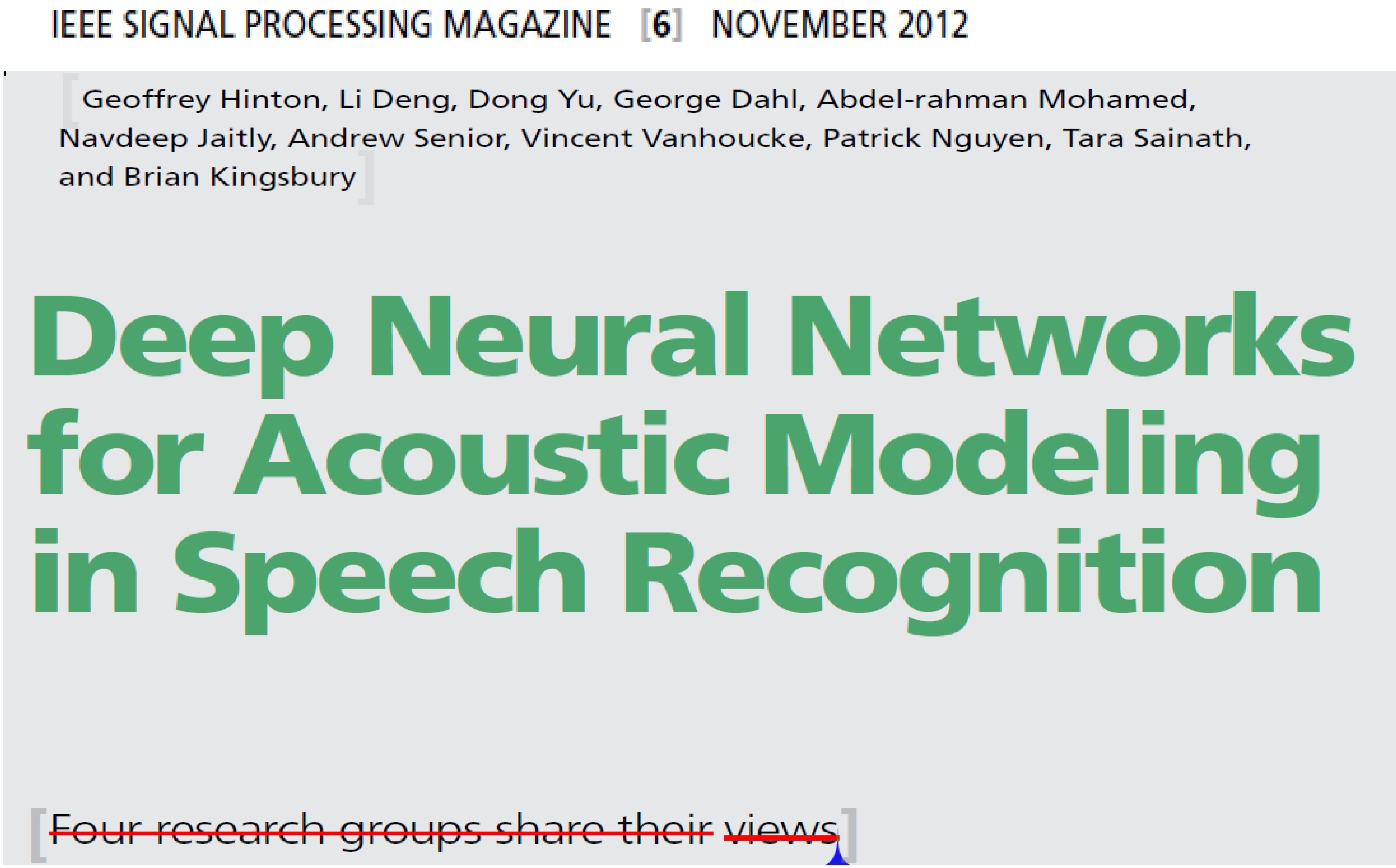
(thanks collaborations/discussions with many people)

Introduction

- Deep neural net ("modern" multilayer perceptron)
- Hard to parallelize in learning →
- Deep Convex Net (Deep Stacking Net)
- Limited hidden-layer size and part of parameters not convex in learning →
- (Tensor DSN/DCN) and **Kernel DCN**
- **K-DCN: combines elegance of kernel methods and high performance of deep learning**
- **Linearity of pattern functions (kernel) and nonlinearity in deep nets**

Microsoft Research

Deep Neural Networks

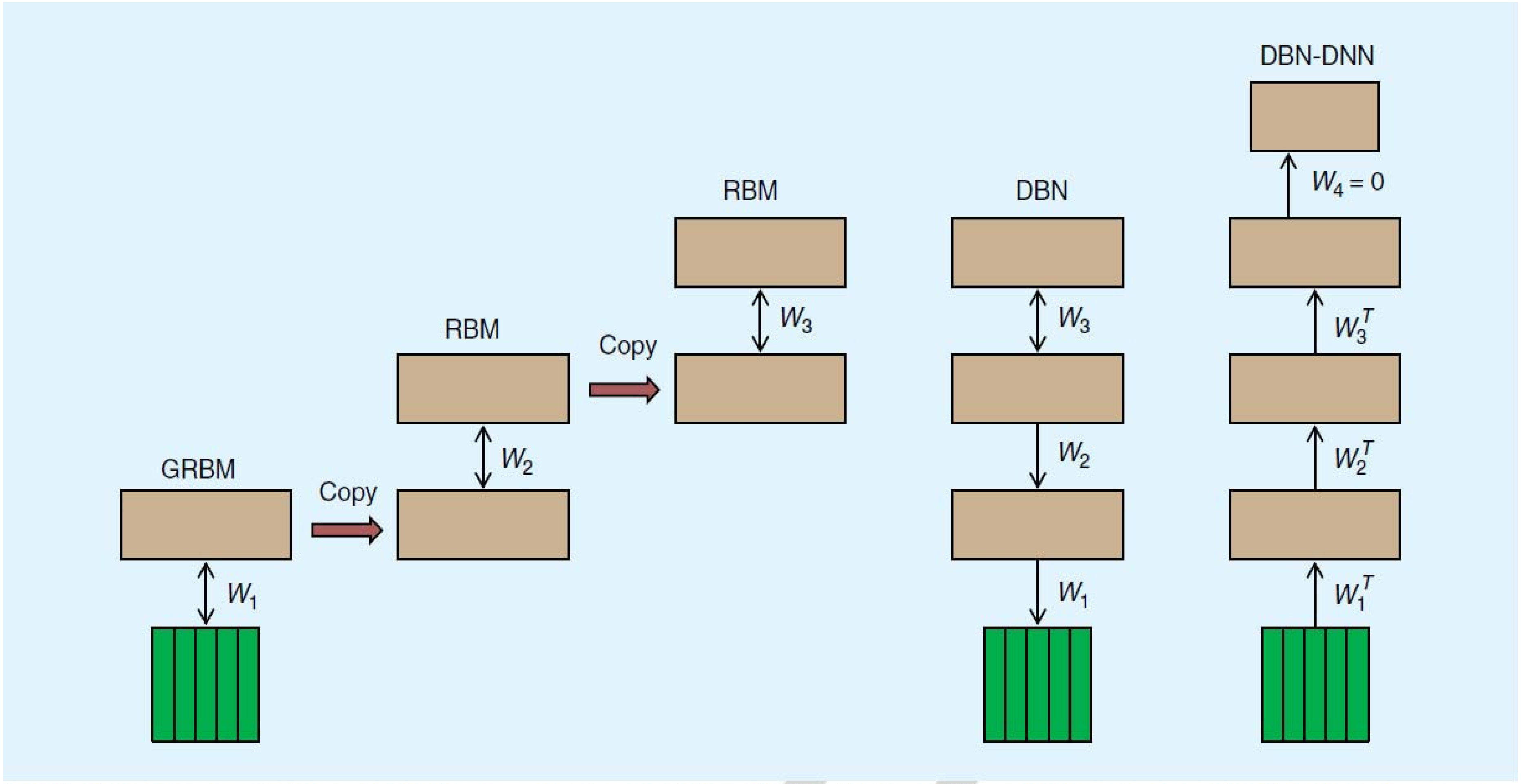


IEEE SIGNAL PROCESSING MAGAZINE [6] NOVEMBER 2012

[Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury]

Deep Neural Networks for Acoustic Modeling in Speech Recognition

[~~Four research groups share their views~~]



[FIG1] The sequence of operations used to create a DBN with three hidden layers and to convert it to a pretrained DBN-DNN. First, a GRBM is trained to model a window of frames of real-valued acoustic coefficients. Then the states of the binary hidden units of the GRBM are used as data for training an RBM. This is repeated to create as many hidden layers as desired. Then the stack of RBMs is converted to a single generative model, a DBN, by replacing the undirected connections of the lower level RBMs by top-down, directed connections. Finally, a pretrained DBN-DNN is created by adding a "softmax" output layer that contains one unit for each possible state of each HMM. The DBN-DNN is then ~~DT~~ to predict the HMM state corresponding to the central frame of the input window in a forced alignment.

Microsoft Research

[TABLE 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

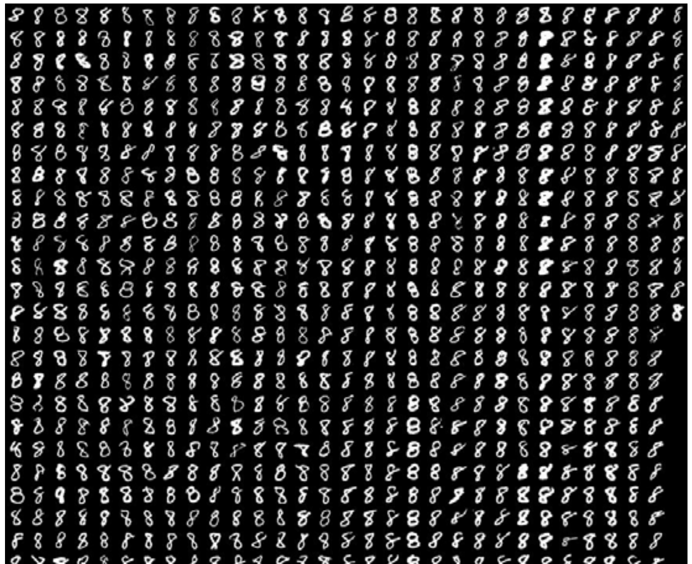
CURRENTLY, THE BIGGEST DISADVANTAGE OF DNNs COMPARED WITH GMMs IS THAT IT IS MUCH HARDER TO MAKE GOOD USE OF LARGE CLUSTER MACHINES TO TRAIN THEM ON MASSIVE DATA SETS.

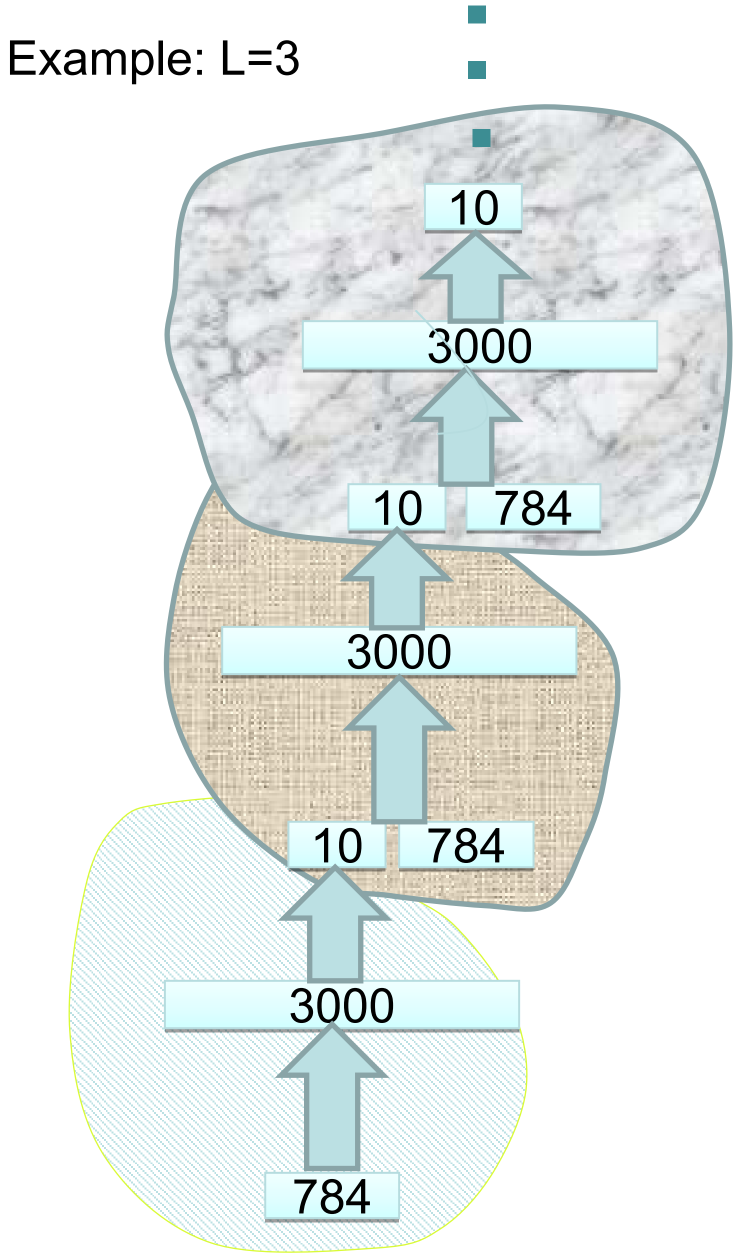
Deep Stacking Network (DSN)

- **“Stacked generalization” in machine learning:**
  - **Use a high-level model to combine low-level models**
  - **Aim to achieve greater predictive accuracy**
- **This principle has been reduced to practice:**
  - **Learning parameters in DSN/DCN** (Deng & Yu, Interspeech-2011; Deng, Yu, Platt, ICASSP-2012)
  - **Parallelizable, scalable learning** (Deng, Hutchinson, Yu, Interspeech-2012)

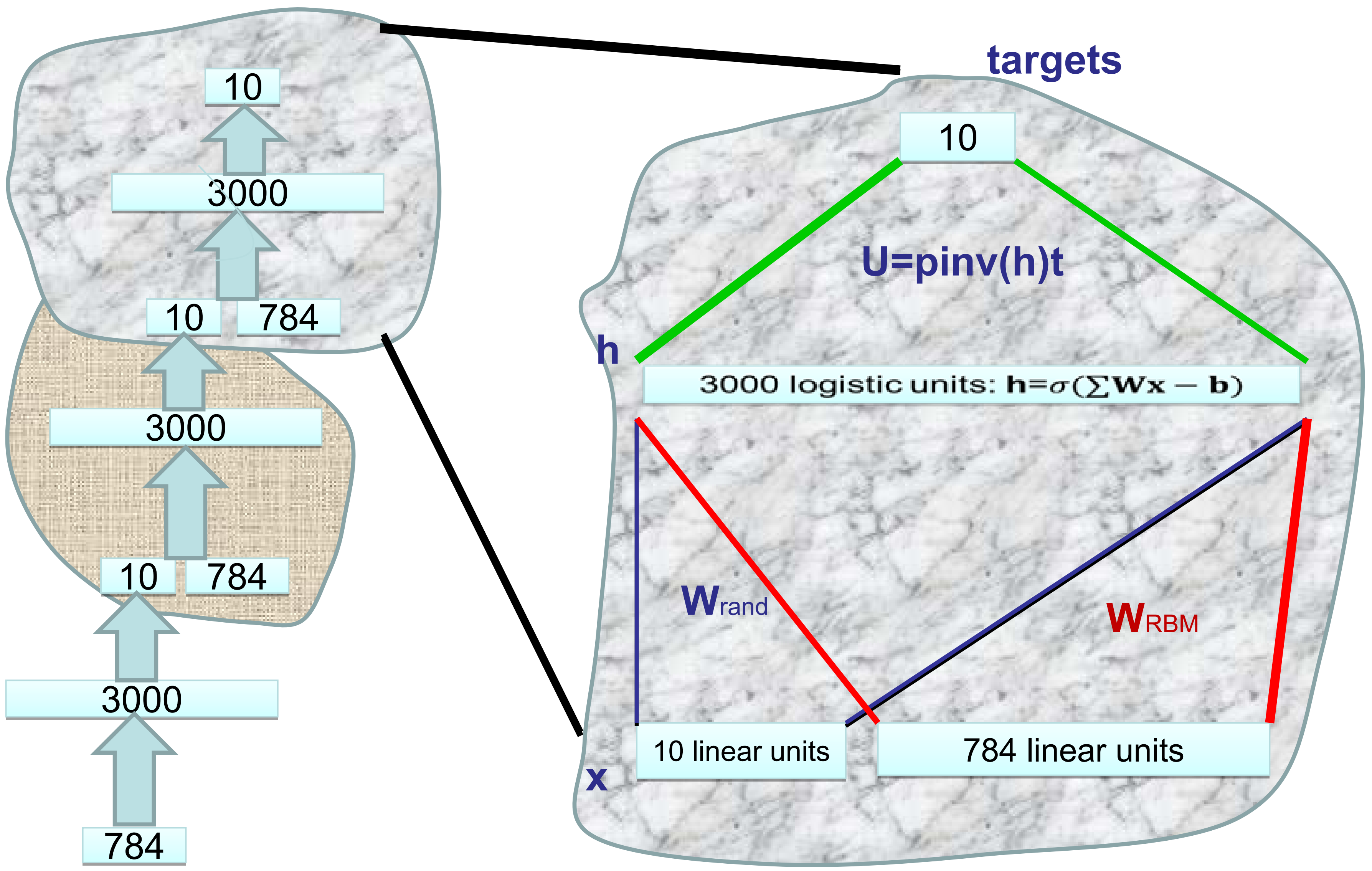
DSN/DCN Architecture

- Many modules
- Still easily trainable
- Alternating linear & nonlinear sub-layers
- Actual architecture for digit image recognition (10 classes)
- MNIST: 0.83% error rate (LeCun's MNIST site)

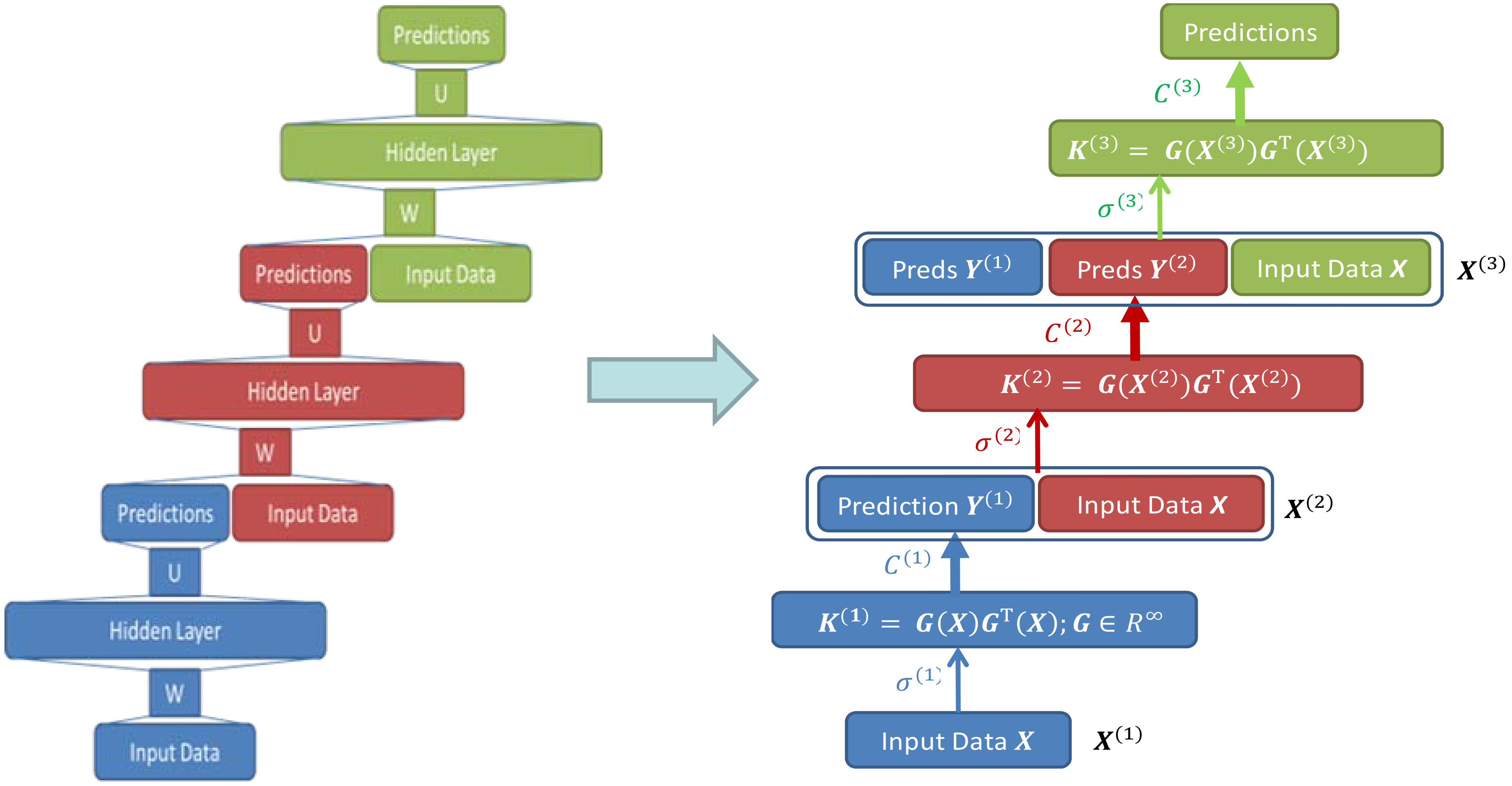




Anatomy of a Module in DCN



From DCN to Kernel-DCN



Kernel-DCN

- Replace each DCN module by kernel ridge regression
- $f(x) = \sum_u \alpha^u K(x_u, x) = K\alpha$, where $\alpha = (\lambda I + K)^{-1}T$
- $K = G([X|\, Y^{(l-1)}|\, Y^{(l-2)}|..Y^{(1)}])G^{\mathrm{T}}([X|\, Y^{(l-1)}|\, Y^{(l-2)}|..Y^{(1)}])$
- Symmetric kernel matrix in the kernel-feature space G typically in infinite dimension (e.g., Gaussian kernel)
- Kernel trick: no need to compute G directly
- Problem: expensive inverse when training size is large
- Solutions:
  - **Nystrom Woodbury approximation**
  - **Reduced rank kernel regression - Feature vector selection**

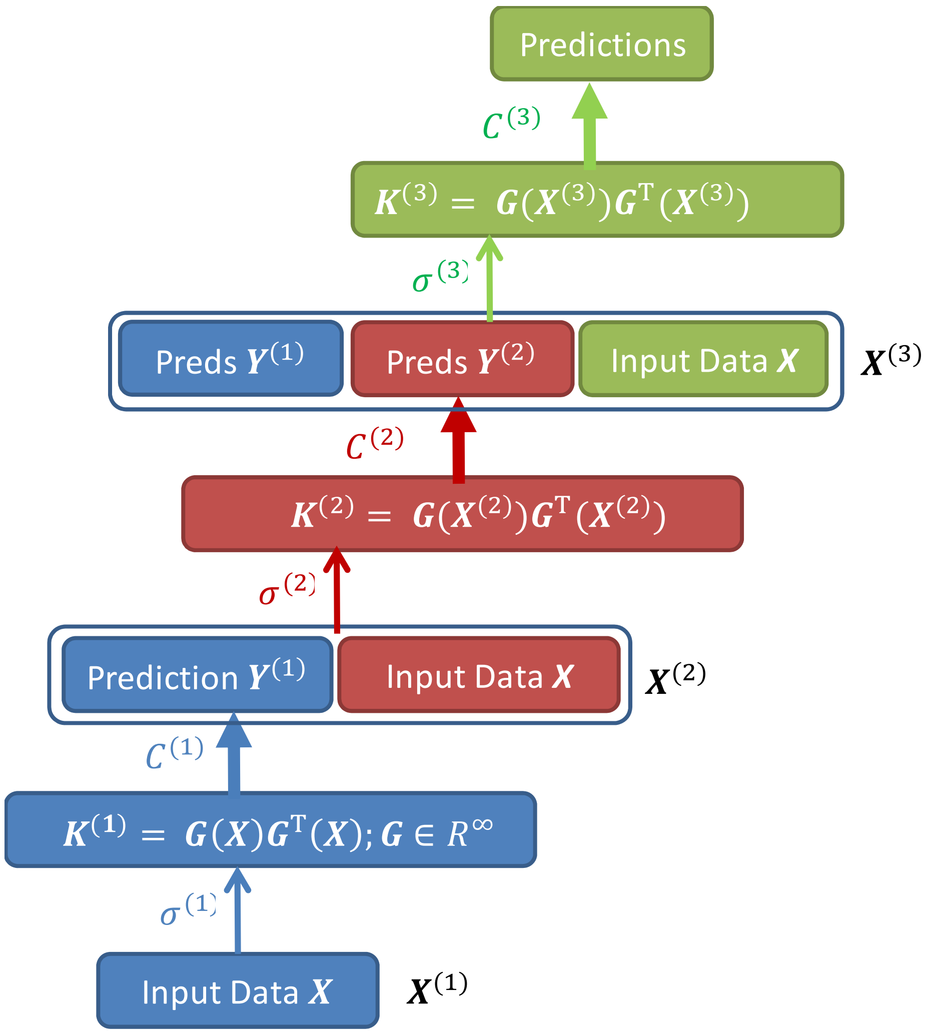
Nystrom Woodbury Approximation

- Kernel ridge regression
  - $f(x) = \sum_u \alpha^u K(x_u, x) = K\alpha$, where $\alpha = (\lambda I + K)^{-1} T$
  - K is large when number of training samples is large
  - Nystrom Woodbury Approximation
    - Sampling l columns from K->C: $K = \begin{array}{c} \begin{array}{ccc} C & & \\ \left[\boxed{\begin{matrix} W \\ K_{21} \end{matrix}} \right. & n & \left. \begin{matrix} K_{21}{}^T \\ K_{22} \end{matrix} \right] \\ l & & \end{array} \end{array}$
    - Approximation $\widetilde{K} = C W_k{}^+ C^T$
    - $(\lambda I + K)^{-1} = \frac{1}{\lambda}(I - C[\lambda I + W_k{}^+ C^T C]^{-1} W_k{}^+ C^T)$
    - SVD of W: O(l^3)
    - Computation of $\widetilde{K}$: O(nlk)
    - (with P-S. Huang on ASR confidence measure)

K-DSN Using Reduced Rank Kernel Regression

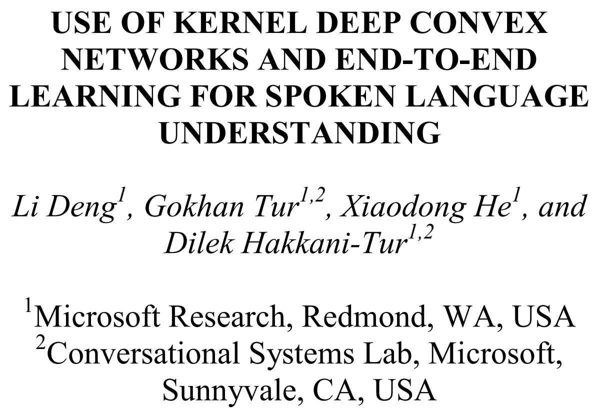
- Solution: **feature vector selection**
  - To identify a subset $\{x_i\}_{i \in S} \subset D$
    - $\phi(x) \approx \hat{\phi}_S(x) \approx \sum_{i \in S} \xi_i \phi(x_i), \forall x \in D$
  - Kernel can be written as a sparse kernel expansion involving only terms corresponding to the subset of the training data forming an approximate basis in the feature space.
    - $f(x) \approx \sum_{i \in S} \beta_i \phi(x_i) \phi(x) = \sum_{i \in S} \beta_i K(x_i, x)$
  - Feature vector selection algorithm from "**Feature vector selection and projection using kernels**" (G. Baudat and F. Anouar)

K-DCN: Layer-Wise Regularization



- Two hyper-parameters in each module
- Tuning them using cross validation data
- Relaxation at lower modules
- Special regularization procedures
- Lower-modules vs. higher modules

SLT-2012 paper:



USE OF KERNEL DEEP CONVEX NETWORKS AND END-TO-END LEARNING FOR SPOKEN LANGUAGE UNDERSTANDING

Li Deng[1], Gokhan Tur[1,2], Xiaodong He[1], and Dilek Hakkani-Tur[1,2]

[1]Microsoft Research, Redmond, WA, USA
[2]Conversational Systems Lab, Microsoft, Sunnyvale, CA, USA

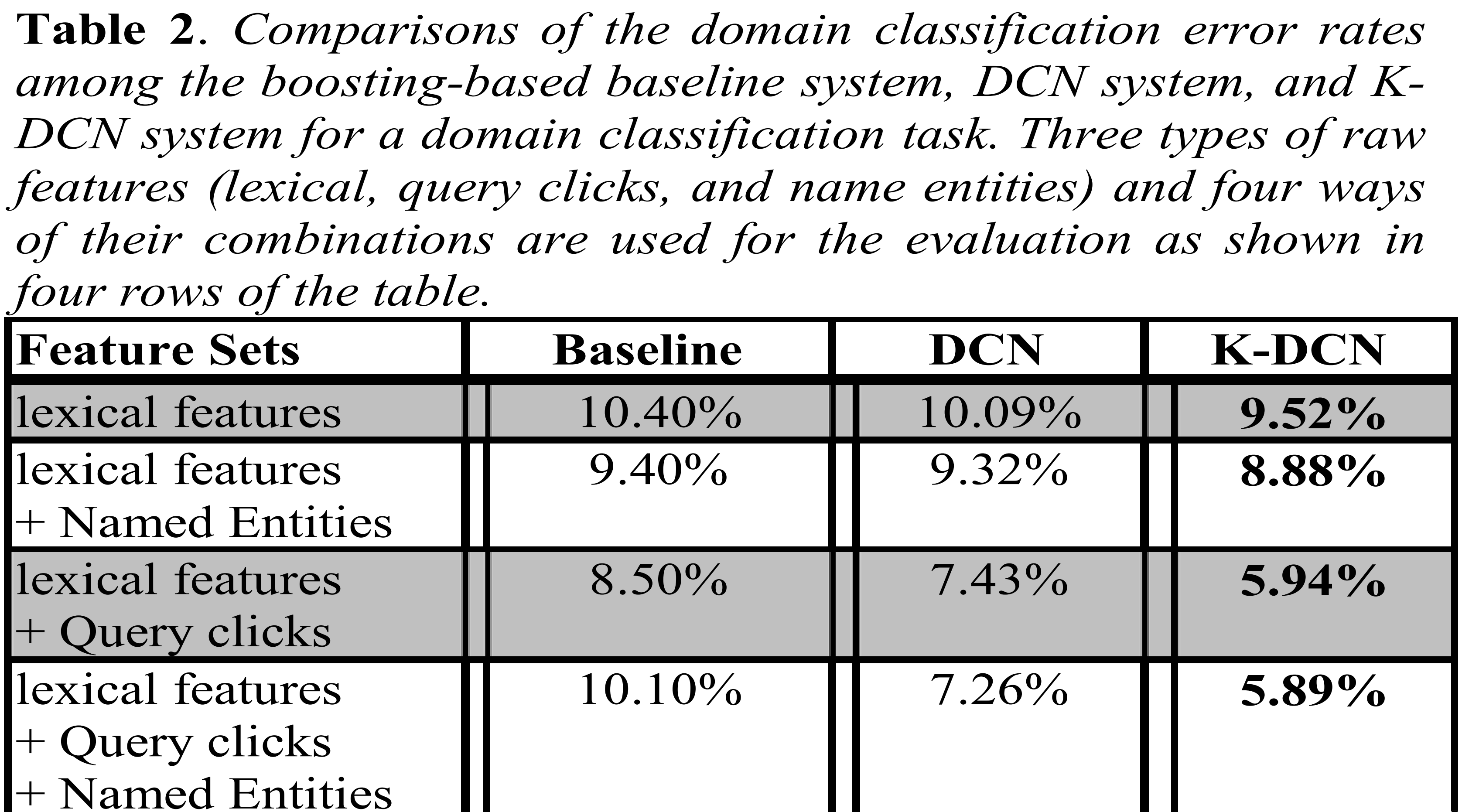


Table 2. *Comparisons of the domain classification error rates among the boosting-based baseline system, DCN system, and K-DCN system for a domain classification task. Three types of raw features (lexical, query clicks, and name entities) and four ways of their combinations are used for the evaluation as shown in four rows of the table.*

Feature Sets	**Baseline**	**DCN**	**K-DCN**
lexical features	10.40%	10.09%	**9.52%**
lexical features + Named Entities	9.40%	9.32%	**8.88%**
lexical features + Query clicks	8.50%	7.43%	**5.94%**
lexical features + Query clicks + Named Entities	10.10%	7.26%	**5.89%**

Table 3. *More detailed results of K-DCN in Table 2 with Lexical+QueryClick features. Domain classification error rates (percent) on Train set, Dev set, and Test set as a function of the depth of the K-DCN.*

Depth	**Train Err%**	**Dev Error%**	**Test Err%**
1	9.54	12.90	12.20
2	6.36	10.50	9.99
3	4.12	9.25	8.25
4	1.39	7.00	7.20
5	0.28	6.50	5.94
6	0.26	**6.45**	**5.94**
7	0.26	6.55	6.26
8	0.27	6.60	6.20
9	0.28	6.55	6.26
10	0.26	7.00	6.47
11	0.28	6.85	6.41