RANDOM FEATURES FOR KERNEL DEEP CONVEX NETWORK

Po-Sen Huang[†], *Li Deng*[‡], *Mark Hasegawa-Johnson*[†], *Xiaodong He*[‡]

[†]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA
[‡]Microsoft Research, Redmond, WA, USA

{huang146, jhasegaw}@illinois.edu, {deng, xiaohe}@microsoft.com

ABSTRACT

The recently developed deep learning architecture, a kernel version of the deep convex network (K-DCN), is improved to address the scalability problem when the training and testing samples become very large. We have developed a solution based on the use of random Fourier features, which possess the strong theoretical property of approximating the Gaussian kernel while rendering efficient computation in both training and evaluation of the K-DCN with large training samples. We empirically demonstrate that just like the conventional K-DCN exploiting rigorous Gaussian kernels, the use of random Fourier features also enables successful stacking of kernel modules to form a deep architecture. Our evaluation experiments on phone recognition and speech understanding tasks both show the computational efficiency of the K-DCN which makes use of random features. With sufficient depth in the K-DCN, the phone recognition accuracy and slot-filling accuracy are shown to be comparable or slightly higher than the K-DCN with Gaussian kernels while significant computational saving has been achieved.

Index Terms— kernel regression, deep learning, spoken language understanding, random features

1. INTRODUCTION

Deep learning has achieved many state of the art results in speech and language processing in recent years [1, 2]. By exploiting deep architectures, deep learning techniques are able to learn different levels of abstraction and further discriminate among data. While deep learning techniques such as deep neural networks have shown remarkable results in recognition and classification tasks, training deep learning models has proved to be computationally difficult [1, 2].

Another architecture, Deep Convex Network (DCN), was proposed to address the scalability issue [3, 4]. Instead of using a large number of hidden units in DCN, Kernel Deep Convex Network (K-DCN) was further proposed to use a kernel trick so that the number of hidden units in each DCN layer is unbounded [5]. However, K-DCN with a Gaussian kernel suffers some limitations in memory and computation, when there are a large number of training and testing samples.

The authors thank Benjamin Recht, Sanjiv Kumar, and Abhishek Singh for helpful discussions.

In this paper, we propose a method for efficiently training and testing K-DCN using the Bochner Fourier-space sampling approximation of a Gaussian kernel in each of the K-DCN modules. By projecting original features to a higher dimensional space explicitly, we can achieve similar performance as K-DCN but with better computational efficiency in time and memory. We demonstrate the effectiveness of our approach on ATIS slot filling and TIMIT phone classification tasks.

The remainder of this paper is organized as follows: Section 2 introduces the kernel deep convex network (K-DCN). Section 3 discusses the limitation of the original K-DCN, and possible solutions in the literature which only addressed the computation and memory problem in training but not in evaluation. Section 4 presents random features and the application to solve the computation and memory problem in both training and evaluation for each of the K-DCN modules. The experimental setups, along with results, are described in Section 5. Section 6 concludes the paper and discusses future work.

2. KERNEL DEEP CONVEX NETWORK

Kernel Deep Convex Network (K-DCN) was proposed in [5], which is a kernel version of the deep convex network (DCN) [3, 4]. In the DCN or K-DCN framework, the outputs from all previous modules are concatenated with the original input data as a new input for the current module. Figure 2 shows an example of a three-layer K-DCN architecture.

The single module of a K-DCN is a kernel ridge regression, which can be expressed as:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) = \mathbf{k}(\mathbf{x})^{\mathrm{T}} \alpha \tag{1}$$

where a sample $\mathbf{x}$ is evaluated with respect to all the training samples $\{\mathbf{x}_i\}_{i=1}^{N}$, vector $\mathbf{k}(\mathbf{x})$ is with element $k_n(\mathbf{x}) = k(\mathbf{x_n}, \mathbf{x})$, and α is the regression coefficient. Using the training data, the ridge regression coefficient has a closed form solution of the following form:

$$\alpha = (\lambda \mathbf{I} + \mathbf{K})^{-1} \mathbf{Y} \tag{2}$$

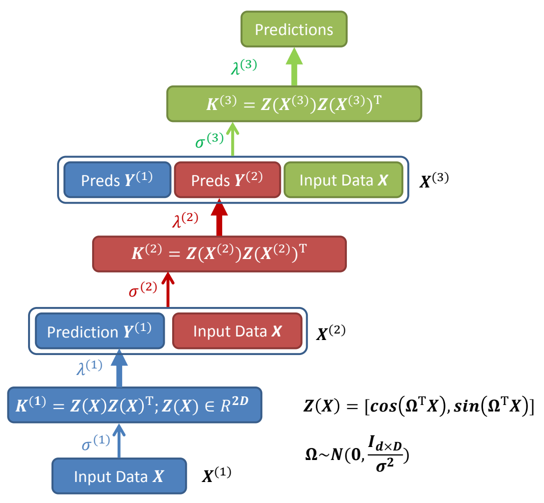
Predictions
$\lambda^{(3)}$
$K^{(3)} = Z(X^{(3)})Z(X^{(3)})^{\mathrm{T}}$
$\sigma^{(3)}$
Preds $Y^{(1)}$
Preds $Y^{(2)}$
Input Data X
$X^{(3)}$
$\lambda^{(2)}$
$K^{(2)} = Z(X^{(2)})Z(X^{(2)})^{\mathrm{T}}$
$\sigma^{(2)}$
Prediction $Y^{(1)}$
Input Data X
$X^{(2)}$
$\lambda^{(1)}$
$K^{(1)} = Z(X)Z(X)^{\mathrm{T}}; Z(X) \in R^{2D}$
$\sigma^{(1)}$
Input Data X
$X^{(1)}$
$Z(X) = [cos(\Omega^{\mathrm{T}}X), sin(\Omega^{\mathrm{T}}X)]$
$\Omega \sim N(0, \frac{I_{d\times D}}{\sigma^2})$

Fig. 1. Architecture of a three-layer K-DCN with random Fourier features. $\mathbf{\Omega}$ is a random matrix with values sampled from $\mathcal{N}(0, \mathbf{I}_{d\times D}/\sigma^2)$. $\mathbf{Z}(\mathbf{X}) = [\cos(\mathbf{\Omega}^{\mathrm{T}}\mathbf{X}), \sin(\mathbf{\Omega}^{\mathrm{T}}\mathbf{X})]$ is a random projection of input $\mathbf{X}$. Parameters σ and λ are the standard deviation for the Gaussian random variable and the regularization parameter for kernel ridge regression, respectively.

where λ is the regularization parameter, $\mathbf{K} \in \mathbb{R}^{N\times N}$ is a kernel matrix with elements $K_{mn} = k(\mathbf{x}_m, \mathbf{x}_n)$, $\{\mathbf{x}_i\}_{i=1}^N$ are from the training set, and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times M}$ are the label vectors for training, where M is the number of classes and $\mathbf{y}_i$, $1 \le i \le N$, is a vector of size M with the only nonzero entry one as the class label [6].

3. LARGE SCALE KERNEL MACHINES

3.1. Limitation in large scale cases

In the kernel regression framework, solving Eq. (2) suffers difficulty when there is a large number of training data. Suppose there are N training samples. It takes $O(N^2)$ storage for the kernel matrix and $O(N^3)$ time for computing the matrix inversion.

In the evaluation stage, as shown in Eq. (1), each sample is evaluated with respect to all the training samples. Suppose $\mathbf{x}$, $\{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^d$. To evaluate a sample, it takes $O(Nd)$ operations to compute $k(\mathbf{x}, \mathbf{x}_i)$ with each training vector $\mathbf{x}_i$, and the training vectors must be retained in memory. For a large training dataset, these testing costs are significant.

3.2. Related Work

To handle the limitation in large scale data cases, several approaches have been proposed. The approaches can be categorized into two categories: (i) selecting a subset of training data and (ii) approximating the kernel matrix.

(i) Burges and Schölkopf proposed a reduced set method by finding a smaller number training set in order to improve classification speed [7]. Baudat and Anouar further proposed a feature vector selection method by finding a subset $\mathcal{S}$ of the training data, forming a basis in the feature space, where the cardinality of the subset $|\mathcal{S}| \ll N$. Equation (1) can be approximated as:

$$f(\mathbf{x}) \approx \sum_{i\in\mathcal{S}} \beta_i \phi(\mathbf{x})\phi(\mathbf{x}_i) = \sum_{i\in\mathcal{S}} \beta_i k(\mathbf{x}, \mathbf{x}_i) \tag{3}$$

Although computation and storage are lower with a smaller set of training samples, this approach requires $O(NL^2)$ computation cost to search a subset of size L sampled among N vectors. Moreover, selecting a subset of training data loses information by throwing away training samples, and is inefficient in finding a good subset among a large scale dataset [8].
(ii) To reduce the storage and computation cost on a kernel matrix, the kernel matrix can be approximated by throwing away individual entries [9], by throwing away entire columns [10, 11], or by a linear kernel [12]. Nyström methods sample a subset columns of the original kernel matrix [10, 11, 13]. With Nyström Woodbury Approximation, Equation (2) can be computed efficiently, as shown in Eq. (4).

$$\begin{aligned}
&(\lambda\mathbf{I} + \mathbf{K})^{-1}\mathbf{Y} \\
&\approx (\lambda\mathbf{I} + \mathbf{C}\mathbf{W}_{\mathbf{k}}^{+}\mathbf{C}^{\mathrm{T}})^{-1}\mathbf{Y} \\
&= \frac{1}{\lambda}\Big(\mathbf{Y} - \mathbf{C}[\lambda\mathbf{I}_{\mathbf{k}} + \mathbf{W}_{\mathbf{k}}^{+}\mathbf{C}^{\mathrm{T}}\mathbf{C}]^{-1}\mathbf{W}_{\mathbf{k}}^{+}\mathbf{C}^{\mathrm{T}}\mathbf{Y}\Big)
\end{aligned} \tag{4}$$

where $\mathbf{C} \in \mathbb{R}^{m\times N}$ is formed by selecting m columns of $\mathbf{K}$, $\mathbf{W}$ is the intersection of these m columns with the corresponding m rows of $\mathbf{K}$, and $\mathbf{W_k}^{+}$ is the pseudo inverse of the k-rank approximation of W. However, in the evaluation stage, the limitation mentioned in Section 3.1 still exists. A linear random projection method was proposed in [9] to speed up kernel matrix evaluation, but it still takes $O(N \log d)$ to evaluate a sample. The random features method approach maps input data to a random feature space and approximate a kernel function by a linear kernel [12]. The detailed theorem and application to K-DCN will be presented in the next section.

4. RANDOM FEATURES

The kernel trick of a kernel function $k(\mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ is to compute an inner product between implicitly mapped features in a high dimensional space, i.e., $k(\mathbf{x}, \mathbf{y}) = \langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle$. Instead of using the implicit feature mapping in the kernel trick, Rahimi and Recht proposed a random feature method for approximating kernel evaluation [12]. The idea is to explicitly map the data to a Euclidean inner product space using a randomized feature map $\mathbf{z} : \mathbb{R}^d \to \mathbb{R}^D$ such that the kernel evaluation can be approximated by the inner product between the transformed pair:

$$k(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y})\rangle \approx \mathbf{z}(\mathbf{x})^{\mathrm{T}}\mathbf{z}(\mathbf{y}) \tag{5}$$

Theorem 1. (Bochner's Theorem [14]) *A kernel $k(\mathbf{x},\mathbf{y}) = k(\mathbf{x}-\mathbf{y})$ on $\mathbb{R}^d$ is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure.*

Bochner's theorem guarantees that the Fourier transform of a shift-invariant and properly scaled kernel is a probability density. Defining $z_\omega(\mathbf{x}) = e^{j\omega^{\mathrm{T}}\mathbf{x}}$, we get

$$k(\mathbf{x}-\mathbf{y}) = \int_{\mathbb{R}^d} p(\omega)e^{j\omega^{\mathrm{T}}(\mathbf{x}-\mathbf{y})}d\omega = \mathbb{E}_\omega[z_\omega(\mathbf{x})z_\omega(\mathbf{y})^*] \tag{6}$$

where $z_\omega(\mathbf{x})z_\omega(\mathbf{y})^*$ is an unbiased estimate of $k(\mathbf{x},\mathbf{y})$ when ω is sampled from $p(\omega)$. For example, $p(\omega) = (2\pi)^{-\frac{D}{2}}e^{-\frac{||\omega||_2^2}{2}}$ is the Fourier transform of a Gaussian kernel $k(\Delta) = e^{-\frac{||\Delta||_2^2}{2}}$.

To reduce the variance of the estimate, we can concatenate D randomly chosen z_ω into a column vector $\mathbf{z}$ and normalize each component by $\sqrt{D}$. Therefore, the inner product $\mathbf{z}(\mathbf{x})^{\mathrm{T}}\mathbf{z}(\mathbf{y}) = \frac{1}{D}\sum_{j=1}^{D} z_{\omega_j}(\mathbf{x})z_{\omega_j}(\mathbf{y})^*$ is a lower variance approximation to the kernel function $k(\mathbf{x},\mathbf{y})$. The approximation converges to the kernel function exponentially fast in D [12]. The quality of the approximation is determined by the mapping dimension D [15].

To obtain real valued random features, we can replace $z_\omega(\mathbf{x})$ by the mapping $z_\omega(\mathbf{x}) = [\cos(\omega^{\mathrm{T}}\mathbf{x}), \sin(\omega^{\mathrm{T}}\mathbf{x})]$, which also satisfies the condition $\mathbb{E}_\omega[z_\omega(\mathbf{x})z_\omega(\mathbf{y})^*] = k(\mathbf{x},\mathbf{y})$. The vector $z(\mathbf{x}) = \frac{1}{\sqrt{D}}[\cos(\omega_1{}^{\mathrm{T}}\mathbf{x}),\ldots,\cos(\omega_D{}^{\mathrm{T}}\mathbf{x}),$ $\sin(\omega_1{}^{\mathrm{T}}\mathbf{x}),\ldots,\sin(\omega_D{}^{\mathrm{T}}\mathbf{x})]^{\mathrm{T}}$ is a $2D$-dimensional *random Fourier feature* of the input feature $\mathbf{x}$, where $\omega_1,\ldots,\omega_D \in \mathbb{R}^d$ are drawn from $p(\omega)$. For a Gaussian kernel, ω_i, $i = 1,\ldots,D$, are drawn from a Normal distribution $\mathcal{N}(0, \mathbf{I_d}/\sigma^2)$.

4.1. Solution

By using the random Fourier feature approximation, we can resolve the limitation mentioned in Section 3.1. In the training stage, first, define $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_N] \in \mathbb{R}^{d\times N}$ and $\mathbf{Z}(\mathbf{X}) = [\cos(\mathbf{\Omega}^{\mathrm{T}}\mathbf{X}), \sin(\mathbf{\Omega}^{\mathrm{T}}\mathbf{X})] \in \mathbb{R}^{2D\times N}$, where $\mathbf{\Omega}$ is a random matrix with values sampled from $\mathcal{N}(0, \mathbf{I}_{d\times D}/\sigma^2)$. The kernel matrix $\mathbf{K}$ can be approximated by $\mathbf{Z}(\mathbf{X})^{\mathrm{T}}\mathbf{Z}(\mathbf{X}) \in \mathbb{R}^{N\times N}$. We can write Eq. (1) in the matrix form as follows:

$$\mathbf{Y} = \mathbf{K}\alpha = \mathbf{Z}(\mathbf{X})^{\mathrm{T}}\mathbf{Z}(\mathbf{X})\alpha \tag{7}$$

Instead of computing and storing $\mathbf{Z}(\mathbf{X})^{\mathrm{T}}\mathbf{Z}(\mathbf{X})$ in the memory, we can use a trick to first compute the following equation:

$$\mathbf{Z}(\mathbf{X})\mathbf{Y} = \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^{\mathrm{T}}\mathbf{Z}(\mathbf{X})\alpha \tag{8}$$

Then, defining $\mathbf{W} = \mathbf{Z}(\mathbf{X})\alpha$, we can solve for $\mathbf{W}$ as follows:

$$\mathbf{W} = (\lambda\mathbf{I} + \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^{\mathrm{T}})^{-1}\mathbf{Z}(\mathbf{X})\mathbf{Y} \tag{9}$$

Table 1. Comparison between kernel ridge regression and random feature kernel regression using a Gaussian kernel

	Kernel Ridge Regression	Random Feature Kernel Regression
Kernel	$\mathbf{K}_{ij} = \exp(-\vert\vert\mathbf{x}_i - \mathbf{x}_j\vert\vert_2^2/2\sigma^2)$	$\mathbf{K}_{ij} = \mathbf{z}(\mathbf{x}_i)^{\mathrm{T}}\mathbf{z}(\mathbf{x}_j)$
Training Formula	$\alpha = (\lambda\mathbf{I} + \mathbf{K})^{-1}\mathbf{Y}$	$\mathbf{W} = (\lambda\mathbf{I} + \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^{\mathrm{T}})^{-1}\mathbf{Z}(\mathbf{X})\mathbf{Y}$
Evaluation Formula	$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x},\mathbf{x}_i)$	$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x})$

In this case, we compute and store the matrix $\mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^{\mathrm{T}} \in \mathbb{R}^{2D\times 2D}$, $\mathbf{Z}(\mathbf{X})\mathbf{Y} \in \mathbb{R}^{2D\times M}$ only.
In the evaluation stage, Eq. (1) can be written as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i k(\mathbf{x},\mathbf{x_i}) = \sum_{i=1}^{N}\alpha_i\mathbf{z}(\mathbf{x_i})^{\mathrm{T}}\mathbf{z}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x}) \tag{10}$$

where $\mathbf{w}^{\mathrm{T}} = \sum_{i=1}^{N}\alpha_i\mathbf{z}(\mathbf{x_i})^{\mathrm{T}}$ is solved in the training stage. The function $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x})$ requires only $O(D+d)$ operations and storage. Table 1 summarizes the comparison between kernel ridge regression and random feature kernel regression using a Gaussian kernel.

5. EXPERIMENTS

In this section, we examine the effectiveness of K-DCN with random Fourier features, denoted as *K-DCNRF*, on ATIS slot filling and TIMIT phone classification tasks.

5.1. ATIS Experiments

We conduct slot-filling experiments on the ATIS dataset following similar settings as described in [16]. In the task, each word will be tagged by a slot ID, and there is a total of 127 slot IDs. An example of a sentence and its slot ID sequence is provided below, following the in/out/begin (IOB) representation, e.g., Boston and New York are the departure and arrival cities and today is the date.

Table 2. An example of a sentence and its slot ID sequence.

sentence	show	flights	from	Boston	to	New	York	Today
Slot ID	O	O	O	B-dept	O	B-arr	I-arr	B-date

The training set consists of 4978 sentences and the test set consists of 893 sentences. In the experiment, we project each word to a 50-dimensional dense vector by looking-up a embedding mapping table, which is trained through unsupervised learning on Wikipedia text corpus [17]. Then we concatenate the embedding vectors in a context window to form a contextual vector as the input for K-DCN/K-DCNRF. As in classical classification tasks, the output of K-DCN/K-DCNRF is a 127-dimensional vector, each element corresponds to one slot ID. K-DCN/K-DCNRF is then trained on the training set. The results of K-DCN/K-DCNRF with various window sizes are reported in Table 3 in slot ID prediction error rate. In contrast, a logistic regression baseline that uses n-gram features (n=1~5) derived from a five-word window obtains a slot error rate of 4.38%. We also present a strong baseline using linear CRF. The result is obtained using only lexical features, with the default parameters of the CRF++ toolkit, following [18]. This gives a slot ID error rate of 3.42%. In Table 3, we show that, by projecting original features to a high dimensional space explicitly, K-DCNRF can achieve similar (or even slightly better) performance to K-DCN. Moreover, in the five-word window case, K-DCNRF outperforms two baseline results. Figure 2 shows the experimental results on the training and testing set at different layers.

Table 3. ATIS results with different window sizes

Window	Method	Layer (proj. dim)	Test Error (%)
1	K-DCNRF	36 (2000)	12.77
3	K-DCN	10	3.83
3	K-DCNRF	34 (16000)	**3.81**
5	logistic regression	-	4.38
5	CRF	-	3.42
5	K-DCNRF	76 (20000)	**2.78**
7	K-DCNRF	88 (30000)	2.74

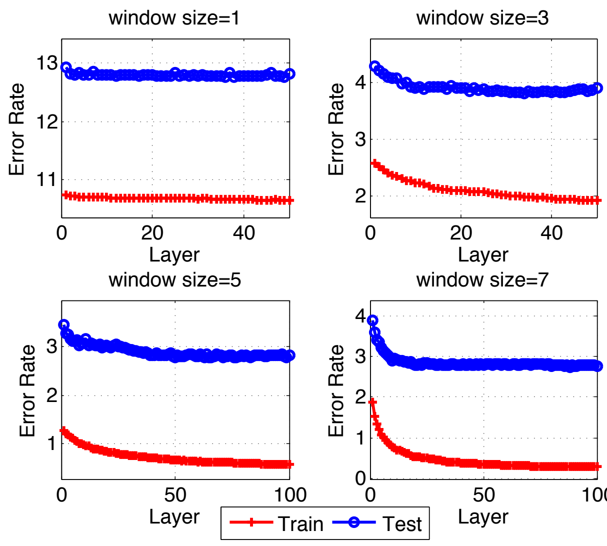

Fig. 2. Experimental results on the ATIS dataset, where the abscissa axis is the layer number and ordinate axis is the error rate.

5.2. TIMIT Experiments

We examine our proposed method on TIMIT phone classification tasks. We extract log-mel-spectrogram features with 24 filter banks and add delta and delta-delta information, with a window size of 25 ms and step size of 10 ms. To include contextual information, we concatenate a context of five frames on each side, with 11 frames in total. Each feature vector contains $39 \times 11 = 429$ elements. In the training set, we use all data from 462 speakers, which contains about 1.12 million samples. We use the standard TIMIT development set of 50 speakers, with 22,488 samples, for cross validation on parameters λ and σ. We test our method on the TIMIT core test set, consisting of 192 utterances with 57,920 samples. In the experiment, we use 183 senone labels as targets for K-DCNRF. The 183 target labels correspond to 61 phone units defined in TIMIT and can be further mapped to 39 phone classes. The frame-level state classification error is shown in Table 4. K-DCNRF outperforms its predecessor, DCN [3, 4], and shallow models [19]. Table 5 shows the detailed results of K-DCN with 44000 projection dimensions on the TIMIT core test set.

Table 4. Frame-level classification error rates of states

Method	Layer (hidden units/proj. dim)	Frame-level State Err.(%) (183 classes)
SVM	-	60.3
OMP [19]	-	48.9
DCN ([3])	6 (6000)	44.24
DCN ([3])	6 (7000)	44.04
DSN ([4])	8 (6000)	43.86
K-DCNRF	4 (44000)	**42.87**

Table 5. Detailed results of K-DCNRF (projection dimension=44000) for frame-level classification error on the TIMIT core test set.

Layer	Frame-level Phone Err.(%) (39 classes)	Frame-level Phone Err.(%) (61 classes)	Frame-level State Err.(%) (183 classes)
1	37.49	43.72	54.97
2	28.74	34.39	45.11
3	26.77	32.29	42.97
4	26.60	32.20	42.87
5	26.58	32.24	43.13

6. CONCLUSION

This paper described a method for efficiently training and testing kernel deep convex networks (K-DCN) using the Bochner Fourier-space sampling approximation of an RBF kernel. The computational savings afforded by this approach, enabled the application of K-DCN to solve classification tasks with on the order of a million or more training vectors. Training sets this large were impractical using the previous approaches. Evaluating on the task of frame-level state classification on the TIMIT data with about 1.12 million training samples, we used the proposed approach to reduce the error rate by 29% compared to an SVM. Since the Bochner approximation allows significant reduction in the computational complexity of a K-DCN, this approach will facilitate future work aiming to apply the K-DCN to large vocabulary speech recognition. In addition to using the approximated linear kernel, other kernel functions such as multilayer kernel machine [20] can be investigated in the K-DCN framework.

7. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, Nov. 2012.

[2] 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, Jan. 2012.

[3] L. Deng and D. Yu, "Deep convex network: A scalable architecture for speech pattern classification," in *Interspeech*, 2011.

[4] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Intenational Conference on Acoustics, Speech, and Signal Processing*, 2012.

[5] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *IEEE Workshop on Spoken Language Technology*, 2012.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., 2006.

[7] C. J.C. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector machines," in *Advances in Neural Information Processing Systems (NIPS)*. 1997, pp. 375–381, MIT Press.

[8] G. Baudat and F. Anouar, "Feature vector selection and projection using kernels," *Neurocomputing*, vol. 55, pp. 21–38, 2003.

[9] D. Achlioptas, F. McSherry, and B. Schlkopf, "Sampling techniques for kernel methods," in *Advances in Neural Infomration Processing Systems (NIPS)*, 2001, pp. 335–342.

[10] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, Dec. 2005.

[11] S. Kumar, M. Mohri, and A. Talwalkar, "Ensemble Nyström method," in *Advances in Neural Infomration Processing Systems (NIPS)*, 2009.

[12] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Infomration Processing Systems (NIPS)*, 2007.

[13] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[14] W. Rudin, *Fourier Analysis on Groups*, Wiley-Interscience, 1994.

[15] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," in *IEEE Workshop on Machine Learning for Signal Processing (MLSP)*, 2012.

[16] G. Tür, D. Hakkani-Tür, and L. Heck, "What's left to be understood in atis," in *Proc. of IEEE SLT Workshop*, 2010.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, 2011.

[18] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proc. of Interspeech*, 2007.

[19] O. Vinyals and L. Deng, "Are sparse representations rich enough for acoustic modeling?," in *Proceedings of INTERSPEECH*, 2012.

[20] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in Neural Infomration Processing Systems (NIPS)*, 2009.