KERNEL METHODS MATCH DEEP NEURAL NETWORKS ON TIMIT

Po-Sen Huang†, Haim Avron‡, Tara N. Sainath‡, Vikas Sindhwani‡, Bhuvana Ramabhadran‡

†Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA
‡IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

huang146@illinois.edu, {haimav, tsainath, vsindhw, bhuvana}@ibm.com

ABSTRACT

Despite their theoretical appeal and grounding in tractable convex optimization techniques, kernel methods are often not the first choice for large-scale speech applications due to their significant memory requirements and computational expense. In recent years, randomized approximate feature maps have emerged as an elegant mechanism to scale-up kernel methods. Still, in practice, a large number of random features is required to obtain acceptable accuracy in predictive tasks. In this paper, we develop two algorithmic schemes to address this computational bottleneck in the context of kernel ridge regression. The first scheme is a specialized distributed block coordinate descent procedure that avoids the explicit materialization of the feature space data matrix, while the second scheme gains efficiency by combining multiple weak random feature models in an ensemble learning framework. We demonstrate that these schemes enable kernel methods to match the performance of state of the art Deep Neural Networks on TIMIT for speech recognition and classification tasks. In particular, we obtain the best classification error rates reported on TIMIT using kernel methods.

Index Terms— large-scale kernel machines, random features, distributed computing, deep learning, speech recognition

1. INTRODUCTION

The recent dramatic success of Deep Neural Networks (DNNs) in speech recognition [1] highlights the statistical benefits of marrying highly non-linear and near-nonparametric models with large datasets, with efficient optimization algorithms running in distributed computing environments. Deep learning models project input data through several layers of nonlinearity and learn different levels of abstraction. The composition of multiple layers of nonlinear functions can approximate a rich set of naturally occurring input-output dependencies. At the same time, the combinatorial difficulty of performing exhaustive model selection in the discrete space of DNN architectures and the potential to get trapped in local minima are well recognized as valid concerns. In this paper, we revisit kernel methods, considered "shallow" in the DNN sense, for large-scale nonparameteric learning. We ask whether, despite their shallow architecture and convex formulations, recent advances in scaling-up kernel methods via randomized algorithms allow them to match DNN performance.

Let $X \subseteq \mathbb{R}^d$ be in input domain of acoustic features. With a kernel function $k : X \times X \to \mathbb{R}$, there is an associated Reproducing Kernel Hilbert Space of functions H_k, with inner product $\langle \cdot, \cdot \rangle_{H_k}$ and norm $\| \cdot \|_{H_k}$, in which model estimation can be performed by minimizing a regularized objective function,

$$f^\star = \text{argmin}_{f \in H_k} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{H_k}^2$$

where $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denotes a training set, λ is a regularization parameter and $V(\cdot, \cdot)$ denotes a loss function. In this paper, we work with the squared loss $V(y, t) = (y - t)^2$ function so that the model estimation above is for kernel ridge regression, although all our techniques generalize to other loss functions. We also mainly work with the Gaussian kernel, though the randomization techniques considered in this paper apply to a broader family of kernels. According to the classical Representer Theorem [2], the minimizer for the above problem has the form,

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \; . \tag{1}$$

Plugging this formula back into the objective function with squared loss yields a dense linear system for the coefficients α:

$$(K + \lambda I)\alpha = \mathbf{y} \tag{2}$$

where $\mathbf{y}$ is the vector of labels and $\mathbf{K}$ is the $n \times n$ Gram matrix given by $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. It is the $O(n^2)$ storage requirements of the Gram matrix, the $O(n^3)$ computational expense of solving a large dense linear system during training, and the need to evaluate in $O(nd)$ time the sum in (1) during testing (assuming that evaluation of the kernel takes $O(d)$ time) that make kernel methods in this form rather unappealing for large datasets.

To handle the limitations in large-scale tasks, several approaches have been proposed. One of the popular directions is to approximate the kernel matrix by linearization, such as the Nyström method [3, 4] and random Fourier features based methods [5, 6]. Although the Nyström method has shown better performance theoretically [7], it involves more expensive computations and larger memory requirements for large-scale tasks. Hence, our focus in this paper is on improving the efficiency of random Fourier features based methods.

A kernel function $k(\mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ can be associated with a high dimensional inner product feature space H such that $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_H$. One such feature space is H_k itself, while another can be obtained through the Mercer's Theorem. For the Gaussian kernel, the dimensionality of these feature spaces is infinite and hence it is preferred to perform computations implicitly using the kernel and its Gram matrix. However, this is the source of increased computational complexity with respect to the number of datapoints.

Instead of using the implicit feature mapping in the kernel trick, Rahimi and Recht proposed a random feature method for approximating kernel evaluation [5]. The idea is to explicitly map the data

to a Euclidean inner product space using a randomized feature map $\mathbf{z} : \mathbb{R}^d \rightarrow \mathbb{R}^D$ such that the kernel evaluation can be approximated by the Euclidean inner product between the transformed pair:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \approx \mathbf{z}(\mathbf{x})^{\mathrm{T}} \mathbf{z}(\mathbf{y}) \quad (3)$$

This feature map is justified via Bochner's theorem which associates each continuous shift-invariant kernel on $\mathbb{R}^d$ with a unique probability density, from which a Monte-Carlo sampling procedure defines a random feature map. In this construction, the Gaussian kernel can be expressed as the expectation of $\mathbf{z}(\mathbf{x})^{\mathrm{T}}\mathbf{z}(\mathbf{y})$, where $\mathbf{z}(\mathbf{x}) = \cos(\omega^{\mathrm{T}}\mathbf{x} + b)$, ω is drawn from a Gaussian distribution, b is drawn from a uniform distribution on $[0, \pi)$, and the cosine function is applied entry-wise. To obtain a lower-variance estimate, we can concatenate D randomly chosen z_ω into a column vector $\mathbf{z}$ and normalize each component by $1/\sqrt{D}$. 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 of the kernel function $k(\mathbf{x}, \mathbf{y})$. With this approximation, instead of solving (2), one can instead solve a $D \times D$ standard regularized linear regression problem,

$$(Z^T Z + \lambda I)\mathbf{w} = Z^T \mathbf{y}$$

where Z is a feature space data matrix whose rows are $\mathbf{z}(\mathbf{x}_i), i = 1 \ldots n$. Predictions can be made with $f(\mathbf{x}) = \mathbf{z}(\mathbf{x})^{\mathrm{T}}\mathbf{w}$. The storage cost of materializing Z is $O(nD)$; while the training and testing complexity is $O(nD^2)$ and $O(dD)$ respectively. These are much more appealing due to linear scaling with respect to number of training points, provided D is small enough. More recent techniques [6] offer similarly performing random Fourier feature maps, with faster mapping, which translates into reducing the time needed for generating Z and faster predictions.

In practice, when the number of training examples increase and the dimension of original features is large, the minimum random feature space dimensionality $D^\star$ needed for competitive performance tends to be very large, and that poses a scalability problem. As an example, the TIMIT dataset needs $D = 200,000$ to get performance on par with state-of-the-art methods. In such settings, materializing the corresponding feature space matrix Z requires several terabytes of storage, leading to a large, dense least-squares regression problem, which likely needs an out-of-core solver to solve exactly.

This paper is motivated by the question of how to make randomized feature map techniques for estimating nonlinear kernel models more scalable. Towards this objective, we propose two complimentary techniques:

- We describe a parallel least squares solver that is specialized for kernel ridge regression with random Fourier feature approximations. Our solver does not materialize Z explicitly, but rather computes blocks of the associated covariance matrix $Z^T Z$ in one pass, followed by on-the-fly calculations in a distributed block coordinate descent procedure. As a side benefit of using an iterative solver, we find that with early-stopping, an explicit regularizer is not needed. In practice, we therefore set $\lambda = 0$. This implies that our approach only requires a single kernel parameter (the bandwidth of the Gaussian kernel). The parallelization details of our least squares solver and its benchmarking in high-performance computing environments will be described in more detail in a separate paper. Here, we demonstrate its value for speech recognition problems.
- We develop an ensemble learning approach where multiple low-dimensional random Fourier models are combined together. While the initial models focus on the entire dataset, the subsequent models focus on high-error areas of the input domain. This enables more efficient data fitting.

Our results indicate that on TIMIT, the generalization performance of deep learning non-convex models can be matched by "shallow" kernel machines with convex optimization using the proposed schemes. Informally, when we say "match" we mean that the best generalization performance obtained on TIMIT with our scalable kernel ridge regression approaches is very similar or better than achieved with DNN (with MSE loss), with comparable number of parameters. We do not report a training time comparison due to differences in the underlying compute environment: our distributed solvers for kernel-methods were trained on an IBM BlueGene/Q Nodecard (32 nodes x 16 cores per node) while DNNs were trained on a GPU.

The remainder of this paper is organized as follows. Section 2 reviews some related work. Section 3 introduces the proposed algorithms: the ensemble kernel machines and the scalable solver. The experimental setups, along with results, are described in Section 4. Section 5 concludes the paper and discusses future work.

2. RELATED WORK

Cho and Saul [8] proposed an arc-cosine kernel function, which mimics the computation in a multilayer neural network. Cheng and Kingsbury further applied the arc-cosine kernel to TIMIT speech recognition task [9]. The sub-sampling strategy in computing the kernel functions with millions of training data, and the usage of MFCC-LDA features lead to a worse result compared with deep neural networks. Huang et al. proposed a kernel deep convex network [10, 11]. The architecture consists of kernel ridge regression with random Fourier approximation in each layer. The model is convex in each layer, but is not convex overall. Furthermore, the dimensionality of random Fourier approximation is relatively small such that the approximation error of the kernel machine is high.

3. PROPOSED METHODS

In this section, we consider the multivariate kernel ridge regression setting with a k-dimensional output space. We denote $Y = [\mathbf{y}_1 \ldots \mathbf{y}_k]$ where $Y_{ij} = 1$ if the i-th training data has label j and $Y_{is} = -1, s \neq j$ otherwise. Correspondingly, the matrix of coefficients is represented by $W = [\mathbf{w}_1 \ldots \mathbf{w}_k]$.

3.1. Ensemble of Kernel Machines

As mentioned in Section 1, when the number of training examples increases and the original feature dimension is high, the dimensionality $D^\star$ also needs to be high for competitive generalization performance approaching that of exact kernel methods. When the dimensionality is high, the memory usage and computation of the feature map is also costly.

We can tradeoff speed and accuracy with feature maps with dimensionality $D < D^\star$. One may view such a random features model as a weak learner, whose performance can be improved by generating an ensemble of multiple weak learners. Hence, we propose to incorporate the idea of ensemble methods [12] to aggregate the models from different weak learners trained with $D < D^\star$ random features, as shown in Algorithm 1. Initially all training samples are assigned with uniform weights. For each learner t, we first sample the training samples according to the weights. Given the sampled training data Z_t with labels Y_t, we solve the least-squares problem $||Z_t W - Y_t||_F^2$. Then, each training sample is reweighted based on its classification result. Difficult samples - with higher classification error - become more likely to be sampled for training the $(t+1)$-th

Algorithm 1: A Proposed Ensemble Algorithm

Input: training data: X and target vector y
Initialize distribution: $D_1(i) \leftarrow 1/n, i = 1, \ldots, n$
for each learner $t = 1, \ldots, T$ **do**
  Sample N samples from X according to distribution D_t, generating X_t.
  Generate a random Fourier feature map, mapping X_t to Z_t.
  Solve $\min_{W_t} \|Z_t W_t - Y\|_F^2$
  Prediction $\hat{y}_t \leftarrow Z_t W_t$.
  Compute the weighted training error: $e_t \leftarrow \sum_{i=1}^{N} D_t(i)(\hat{y}_{t_i} \neq y_i)$
  Define $\alpha_t = \frac{1}{2}(\frac{1-e_t}{e_t})$
  Update $D_{t+1}(i)$:

$$D_{t+1}(i) \leftarrow \frac{D_t(i)}{S_t} \times \begin{cases} e^{-\alpha_t} & \text{if } \hat{y}_{t_i} = y_i \\ e^{\alpha_t} & \text{if } \hat{y}_{t_i} \neq y_i \end{cases}$$

  S_t is a normalization factor.
end for
Output: for each $t = 1, \ldots, T$: W_t, α_t, feature map

learner. For samples which were classified incorrectly and are classified correctly in learner t, the weights will be reduced. The final decision is made based on the prediction from learners $t = 1, \ldots, T$, weighted by the classification error of each learner. This ensemble mechanism may be seen as a non-uniform sampling approach where the capacity of the overall model is distributed as per the hardness of classification.

3.2. Scalable Solver

We now discuss a more direct approach to handle very large random feature spaces, which in turn leads to solving very large least-squares problems of the form $\min_W \|ZW - Y\|_F^2$. Large number of random features poses a serious scalability problem, if the least squares solver is implemented naively. For example, consider the TIMIT dataset dataset with $100,000$ random features. Just storing the matrix Z in memory in double-precision requires over 1 terabyte of data. Once in memory, one has to solve the dense regression problem, which will take too long for a direct method. If terabyte order memory is not available, one traditional approach is to resort to slower out-of-core solution with costly disk I/O per iteration.

The key for a scalable solver is observing that the matrix Z is implicitly defined by the input matrix X. To take advantage of this we use a block coordinate descent algorithm. At each iteration block coordinate descent fixes the part of W that corresponds to coordinates (columns of Z) that are outside the current block, and optimizes the part of W that corresponds to coordinates inside the current block (i.e., a set of rows of W). This leads to solving the following least-squares problem in each iteration

$$\min_{\Delta W_b} \|Z_b \Delta W_b - R\|_2^F \,, \tag{4}$$

where Z_b is the part of Z that corresponds to the coordinates in block b (a set of columns of Z), ΔW_b is the update to W_b, the part of W that corresponds to the coordinates in block b (a set of rows of W), and $R = ZW - Y$ is the current residual vector. The update is then $W_b \leftarrow W_b + \Delta W_b$.

As we see each iteration requires only a part of Z, and there is no need to form the entire matrix. This is much more economical in memory use; our solver simply forms the relevant Z_b in each iteration, and solves (4).

Let s_b denote the size of block b. To avoid paying $O(ns_b^2)$ in each iteration, we form the Gram matrix $Z_b^T Z_b$ and factor it only during the first epoch. Subsequent epochs require only $O(ns_b)$ per iteration in addition to the time required to form Z_b.

We use parallel processing to allow us to work on large datasets. We designed our solver as a distributed-memory solver. We split the input matrix X and right-hand-size row-wise, with each processor receiving a subset of the rows of X and of the right-hand side. The solver also updates a copy of R, which is distributed row-wise with exactly the same split as X. All processors keep a synchronized copy of W. The row-wise split enables us to take advantage of the fact that mapping a row of X to a row of Z can be done independently of other rows of X, so the distribution of X implicitly defines a distribution of Z and Z_b. This enables the block coordinate descent to be performed with little synchronization and communication overhead.

See Algorithm 2 for a pseudo-code description. Detailed benchmarking and enhancements of this solver in high-performance computing environments will be reported in a separate paper.

Algorithm 2: Scalable Distributed-Memory Solver

Input: training data: X and target Y
Distribute X and Y row-wise across machines;
Initialize: $W \leftarrow 0$, $R \leftarrow Y$.
Generate a random Fourier feature map (maps X to Z).
while weights have not converged **do**
  for each block b **do**
    $\triangleright$ Z_b denotes the columns of Z corresponding to block b. W_b denotes the rows of W corresponding to block b.
    Compute Z_b
    If (first epoch): compute and factor $Z_b^T Z_b$
    Compute $Z_b^T R$
    $\Delta W_b \leftarrow (Z_b^T Z_b)^{-1} Z_b^T R$
    $W_b \leftarrow W_b + \Delta W_b$
    $R \leftarrow R - Z_b \Delta W_b$
  end for
end while
Output: W, feature map

4. EXPERIMENTS

In this section, we examine the effectiveness of our approaches on the TIMIT corpus for phone classification and recognition tasks.

4.1. Corpus

Phone classification and recognition experiments were evaluated on the TIMIT corpus. The training set contains 3696 SI and SX sentences from 462 speakers. A separate development set of 50 speakers was used for hyperparameter tuning. Experiments are evaluated on the core test set, which consists of 192 utterances, with 8 utterances from each of the 24 speakers. The SA sentences are excluded from tests.

In all experiments, we use 40 dimensional feature space maximum likelihood linear regression (fMLLR) features [13] and then concatenate the neighboring 5 frames (11 frames in total) as the input feature for the experiments, which is reported as the state-of-the-art feature for deep neural network by Mohamed et al. [14]. In this paper, we focus on training kernel machines with the mean square error

(MSE) objective and compare the results with DNNs using MSE and cross entropy (CE) objectives. To have a fair comparison, we select the best architecture (number of hidden units and number of layers) for DNNs based on the development set. We report the number of hidden units and the number of hidden layers for the selected DNNs in Table 1, 2, and 3.

4.2. Classification task

We first look at the frame-level classification results on the TIMIT dataset. We use 147 (49 phonemes $\times$ 3 states) context independent states as targets. The training data consists of 2 million frames of fMLLR features, extracted using a 5 ms step size.

As mentioned in Section 1, the training and evaluation of an exact kernel ridge regression model is computationally constrained by the number of training samples. Hence, in order to evaluate performance with an exact kernel method, given computational limitations, we use a subset of 100,000 training samples (2 million frames). Table 1 shows the results. We can observe that the exact kernel achieves the best results by fitting the training data well. The error of random Fourier (RF) models decreases as the number of random features increases. By constructing an ensemble of several RF models, the kernel machines are able to get closer to the full kernel performance. The performance of kernel machines is significantly better than DNN with MSE objectives.

Table 1. Classification results (147 classes) on 100K samples, where D is the number of random features

Model (D - T learners)	Training Error (%)	Test Error (%)
Full RBF	0.07	38.61
MSE-RF (5K - 1)	33.32	44.08
MSE-RF (5K - 30)	3.69	39.59
MSE-RF (10K - 1)	23.70	42.08
MSE-RF (10K - 30)	0.20	39.33
Model (hidden units - layers)	Training Error (%)	Test Error (%)
MSE-DNN (1K - 1)	49.71	50.13
MSE-DNN (1K - 3)	36.79	40.12
MSE-DNN (1K - 5)	41.55	43.23

Table 2 shows the frame-level classification results on the full training set. Here, the exact kernel method becomes computationally infeasible. By increasing the number of random features of RF models using the scalable solver or combining several relatively low-dimensional RF models, the performance of MSE-RF is similar to the best results by MSE-DNN. The DNN with CE objective, minimizing the classification objectives directly, achieves the best results.

4.3. Recognition task

To convert the output of kernel machines for the hidden Markov model (HMM) Viterbi decoding, we follow the hybrid approach in [9]. We trained a weighted softmax function with the CE objective for phone posterior $p(q = i|x)$.

$$p(q = i \mid x) = \frac{\exp(\sum_j a_{ij} p_{ij} + b_i)}{\sum_k \exp(\sum_j a_{kj} p_{kj} + b_k)}, \quad (5)$$

where a_{ij} is a trainable weight for kernel output p_{ij} and b_i is a trainable bias for phone i. The weights are trained using stochastic gradient descent. By Bayes rule, $p(x|q = i) \propto p(q = i|x)/p(q = i)$, the posteriors can be transformed to HMM state emission probabilities. The priors $p(q = i)$ are estimated from the training data. Then,

Table 2. Classification results (147 classes) on the full training set, where D is the number of random features

Model (D - T learners)	Training Error (%)	Test Error (%)
MSE-RF (60K - 1)	27.16	35.95
MSE-RF (60K - 7)	14.56	**33.5**
Scal. MSE-RF (200K - 1)	18.43	34.08
Scal. MSE-RF (400K - 1)	11.95	**33.67**
Scal. MSE-RF (600K - 1)	9.15	33.69
Model (hidden units - layers)	Training Error (%)	Test Error (%)
MSE-DNN (4K - 2)	21.57	**33.53**
CE-DNN (1K - 3)	22.05	**32.99**

Table 3. Recognition results, where D is the number of random features

Model (hidden units - layers)	Test CI Error (%)	Test PER (%)
MSE-DNN (4K - 2)	33.53	22.3
MSE-DNN (2K - 2)	34.12	**22.2**
CE-DNN (1K - 3)	32.99	21.7
CE-DNN (4K - 3)	33.34	**20.5**
Model (D - T learners)	Test CI Error (%)	Test PER (%)
Scal. MSE-RF (400K - 1)	33.67	**21.3**

Viterbi decoding is used for phone recognition. Table 3 shows the recognition results on TIMIT. The MSE-RF achieves better performance compared to MSE-DNN and slightly worse performance than CE-DNN. Note that the DNNs chosen are the best performing models in terms of the number of hidden units and number of layers on the development set.

5. CONCLUSION

In this paper, we explore the comparison between deep learning models and shallow kernel machines. By using an ensemble approach and a complimentary scalable solver, we show that the model expressibility of kernel machines can match deep neural network models, for very similar generalization performance. Furthermore, kernel machines have the advantages of having a convex optimization based formulation and simpler model selection. To improve the phone classification and recognition performance, optimizing kernel machines with the CE objective can be further explored, as an alternative to MSE-based models explored in this paper. Another promising direction is the use of Quasi-Monte Carlo techniques to improve the efficiency of explicit feature map approximations along the lines proposed in [15].

6. ACKNOWLEDGMENT

Thanks to Ewout van den Berg for help with TIMIT experiments. H. Avron and V. Sindhwani acknowledge the support of the XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0323. This work was done while P.-S. Huang was a summer intern at IBM Research.

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. Kings-

bury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, Nov. 2012.

[2] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT press, 2001.

[3] 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.

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

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

[6] Q. Le, T. Sarlós, and A. Smola, "Fastfood – Approximating kernel expansions in loglinear time," in *International Conference on Machine Learning (ICML)*, 2013.

[7] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström method vs random fourier features: A theoretical and empirical comparison," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 485–493.

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

[9] C.-C. Cheng and B. Kingsbury, "Arccosine kernels: Acoustic modeling with infinite neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5200–5203.

[10] 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.

[11] P.-S. Huang, L. Deng, M. Hasegawa-Johnson, and X. He, "Random features for kernel deep convex network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[12] H. Xia and S. C.H. Hoi, "Mkboost: A framework of multiple kernel boosting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1574–1586, 2013.

[13] M.J.F. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer Speech & Language*, vol. 12, no. 2, pp. 75 – 98, 1998.

[14] A. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5060–5063.

[15] J. Yang, V. Sindhwani, H. Avron, and M. Mahoney, "Quasi-Monte Carlo feature maps for shift-invariant kernels," in *International Conference on Machine Learning (ICML)*, 2014.