Bridging deep and kernel methods

Lluís A. Belanche[1] and Marta R. Costa-jussà[2] *

1- Computer Science School - Dept of Computer Science
Universitat Politècnica de Catalunya
Jordi Girona, 1-3, 08034 Barcelona, Catalonia, Spain

2- TALP Research Center - Dept of Signal Theory and Communications
Universitat Politècnica de Catalunya
Jordi Girona, 1-3, 08034 Barcelona, Catalonia, Spain

Abstract. There has been some exciting major progress in recent years in data analysis methods, including a variety of deep learning architectures, as well as further advances in kernel-based learning methods, which have demonstrated predictive superiority. In this paper we provide a brief motivated survey of recent proposals to explicitly or implicitly combine kernel methods with the notion of deep learning networks.

1 Introduction

Multilayer artificial neural networks have experienced a rebirth in the data analysis field, displaying impressive empirical results, extending even to classical artificial intelligence domains, such as game playing, computer vision, multimodal recognition, natural language processing and speech processing. The versatility of such methods to perform non-linear feature extraction and deal with a diversity of complex data –such as video, audio and text– has lead many-layered ("deep") parametric models to get over well-established learning methods, like kernel machines or classical statistical techniques. Deep neural networks, for example, are very successful in machine learning applications although their number of parameters is typically orders of magnitude larger than the number of training examples. However, their training is a delicate and costly optimization problem that raises many practical challenges.

Learning several representational levels to optimize a final objective is the main trademark of *deep* learning, from which multiple tasks and fields have benefited lately. Key advances in deep neural networks in the last years include:

- Long-Short Term Memories (LSTM) [1] for their capacity to maintain contextual information in the data;
- Convolutional Neural Networks [2] for their capacity to identify regular patterns in the data; and

*This study has been funded by the Spanish Government project TIN2016-79576-R, the Spanish *Ministerio de Economía y Competitividad* and the European Regional Development Fund, through the postdoctoral senior grant *Ramón y Cajal* and the contract TEC2015-69266-P (MINECO/FEDER, EU).

- The Auto-Encoder structure [3] for its capacity to learn a compact represention of data.

Current architectures that are raising the interest of the community include phased LSTMs [4] –capable of training longer sequences than LSTMs in an efficient way– and adversarial networks [5], able to produce new data by training in competition generative and discriminative neural methods.

On the other hand, non-parametric kernel methods usually involve solving a tractable convex problem and are able to handle non-vectorial data directly, leading to a higher expressive power. Kernel methods involve the use of positive semi-definite matrices to represent complex relations by mapping points into high (possibly infinite) dimensional feature spaces, providing a solid framework in which to represent many types of data, as vectors in $\mathbb{R}^d$, strings, trees, graphs, and functional data, among others [6]. This expressive richness of kernel methods has a price: by resorting to implicit computations, the methods operate on the Gram matrix (the matrix of implicit inner products of the data), which may raise serious computational issues for modern datasets.

Support Vector Machines (SVMs) are not the only possible method making use of the *kernel trick* [6]. Over the years, these and other kernel-based methods have gained prominence, both for supervised tasks (such as classification and regression) and unsupervised tasks (such as novelty detection, clustering, and feature extraction). These include the Relevance Vector Machine [7], Spectral clustering [8], Kernel Linear Discriminant Analysis [9], Kernel Principal Components Analysis [10], Kernel Canonical Correlation Analysis [11] and Kernel Independent Component Analysis [12], among others. Their main drawback is arguably the computational complexity dependence on the number of data points, both in terms of time and space requirements.

A natural and emerging field of research is given by their hybridization, which can be done in many fruitful ways. Ideas from the deep learning field could be transferred to the kernel framework and *vice versa*. In recent years, researchers in these hitherto separate areas are developing "meeting points", a promising one being Gaussian Processes (roughly, the "Bayesian view" of kernel methods). As an example, there is a natural duality between inner products of features and kernels which can be used, for instance, to design new neural layers using kernel functions. This is because one central lesson taught by deep machines is that complex representations must be *learned*. In this sense, the Gram matrix of the data (being unsupervised) may not capture the required similarity relations needed to predict a yet unseen target variable. The incorporation of stacked parameterized learning mechanisms then opens the way to more expressive architectures.

This brief survey is organized as follows. A compilation of recent and highly relevant work is provided in Section 2, followed by work specifically taylored to Natural Language Processing tasks (Section 3). Next, we describe some useful techniques that form the basis of many attempts at hybridization (Section 4). The paper ends with some insightful conclusions.

2 Recent work

There has been a growing interest in exploring the possibility of integrating deep learning networks and kernel machines. Here we only review a (hopefully representative) sample of recent published work.

- The work by I. Steinwart *et al.* [13] investigates iterated compositions of (weighted sums of) Gaussian kernels, while providing with an interpretation that shows some similarities with deep neural networks (DNNs). It is proven that these kernels are universal [14]. The approach compares favourably to standard SVMs, Random Forests, a Multiple Kernel Learning approach[1], as well as to some DNNs.

- The work by G. Pandey *et al.* [16] approaches single-layer *wide* (meaning infinitely large) learning based on arc-cosine kernels (see section 4). Exact and approximate (but faster) learning strategies are developed and shown to outperform both shallow and deep architectures of finite width, and with less number of parameters.

- Deep Gaussian Processes (DGPs) are probabilistic deep models obtained by stacking multiple layers, each one realized as a Gaussian Process (GP) [17]. The work by Cutujar *et al.* [18] adopts random Fourier features (see section 4) to DGPs, using stochastic variational inference for preserving the probabilistic representation of a regular GP. This work bridges the gap between DNNs and DGPs, exploiting the advantages of both. The random feature approximations to DGPs with arc-cosine kernels are approximated by DNNs with ReLU activation functions. The procedure is reported to lead to competitive performance (and faster inferences) compared to state-of-the-art approaches to DGP [19].

- The work by J. Mairal *et al.* [20] introduces a new type of kernel-based Convolutional Neural Network (CNN), which builds a multilayer image representation by stacking and composing kernels. The network invariance is encoded by a reproducing kernel, thereby obtaining simple networks, being easier to train and more robust against overfitting. The authors report comparable performance on several benchmarking image datasets, with simple architectures and no data augmentation.

- A.G. Wilson *et al.* [21] are able to construct kernels which encapsulate the expressive power of deep architectures, represented again as scalable probabilistic GPs, producing a probabilistic mapping with an infinite number of adaptive basis functions.

- T. Hazan *et al.* [22] show how deep infinite layers are naturally aligned with GPs and kernel methods in general, and design stochastic kernels that

[1] Multiple Kernel Learning (MKL) [15] aims at finding the best combination of kernels to solve a task.

rely on GPs to encode the information progressively built in layered deep infinite neural networks.

3 Deep learning applied to NLP

Deep learning has revolutionized several fields in the last years: in areas like speech recognition or image classification, DNNs now surpass the predictive performance previously achieved by non-parametric models or even human beings. Natural language processing (NLP) –understood as the field of computer science that aims to interpret and produce human language– perhaps exemplifies better than any other the qualitative gap that has been created. As presented in [23], basic NLP tasks such as part-of-speech tagging, chunking, named entity recognition and semantic role labeling can be approached effectively disregarding hand-made features by learning internal representations based on large amounts of (labeled or unlabeled) training data.

Advances in NLP aim nowadays to deploying crucial technologies –such as machine translation, speech recognition and speech synthesis– to reach real-world applications. These include spoken dialogue systems, speech-to-speech translation engines, chatbots, and the mining of social media for health and development [24]. As a paradigmatic example, modern *machine translation* –always a central topic in NLP– uses an autoencoder scheme with bidirectional recurrent neural networks as attention-based mechanism. This architecture permits to imagine a common representation of languages (i.e. an *interlingua*) that could be helpful in many other applications.[2] Google's Neural Machine Translation System constitutes a real working example of deep learning in action [26].

4 Some key techniques

A kernel function k implicitly defines a map $\phi : \mathcal{X} \to \mathcal{H}$ from an input space of objects $\mathcal{X}$ into some Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ (called the *feature space*). The "kernel trick" consists in performing the mapping and the inner product simultaneously by defining its associated kernel function:

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}, \ \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \tag{1}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes inner product in $\mathcal{H}$.

Neural network kernels. Kernels for building wide large-margin classifiers have been introduced in [27]. These kernels typically rely on an integral representation, and provide interesting connections with neural networks, which makes them good candidates to benefit from deep neural architectures. As an example, the arc-cosine kernel [27], as many kernels do, computes the similarity of two data vectors, $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$. It requires a "degree" parameter n, the definition of the n-th order kernel being:

[2] The interlingua allows to translate among low-resourced language pairs [25].

$$k_n(\mathbf{x}, \mathbf{x}') = 2 \int d\mathbf{w} \frac{\exp(-\frac{||\mathbf{w}||^2}{2})}{(2\pi)^{d/2}} \Theta(\mathbf{w}^\top \mathbf{x}) \Theta(\mathbf{w}^\top \mathbf{x}') (\mathbf{w}^\top \mathbf{x})^n (\mathbf{w}^\top \mathbf{x}')^n \quad (2)$$

where $\Theta(z) := \frac{1}{2}(1 + sgn(z))$. The solution of the previous integral results in:

$$k_n(\mathbf{x}, \mathbf{x}') = \frac{1}{\pi} ||\mathbf{x}||^n ||\mathbf{x}'||^n J_n(\theta) \quad (3)$$

which shows a rather trivial dependence on the lenghts of $\mathbf{x}, \mathbf{x}'$, but a more complex relation via the angle (θ) between them:

$$\theta = \cos^{-1}\left(\frac{\mathbf{x}^\top \mathbf{x}'}{||\mathbf{x}|| \cdot ||\mathbf{x}'||}\right) \quad (4)$$

The function J_n is expressed as follows:

$$J_n(\theta) = (-1)^n (\sin\theta)^{2n+1} \left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^n \left(\frac{\pi - \theta}{\sin\theta}\right) \quad (5)$$

Now, a single-layer neural network with M hidden units maps input vectors $\mathbf{x}$ with functions of the kind $g(W\mathbf{x})$, being W the corresponding weight matrix. We consider the family of one-sided polynomial activation functions:

$$\Gamma_n(z) := \Theta(z) z^n$$

These functions vanish up to z and become a monomial afterwards (for example, one gets the step Heaviside function for $n = 0$). For a fixed n, the output of the network is then $y(\mathbf{x}) = \sum_{i=1}^{M} \Gamma_n(\mathbf{w}_i{}^\top \mathbf{x}) = \sum_{i=1}^{M} \Theta(\mathbf{w}_i{}^\top \mathbf{x})(\mathbf{w}_i{}^\top \mathbf{x})^n$ where $\mathbf{w}_i$ is the i-th row of the weight matrix. Then one can define an inner product f between a pair of input vectors as:

$$f(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{M} \Theta(\mathbf{w}_i{}^\top \mathbf{x}) \Theta(\mathbf{w}_i{}^\top \mathbf{x}') (\mathbf{w}_i{}^\top \mathbf{x})^n (\mathbf{w}_i{}^\top \mathbf{x}')^n$$

The relation with the arc-cosine kernel follows from assuming that the weights follow a standardized d-normal distribution $\mathbf{w}_i \sim \mathcal{N}(0, I_d)$ and taking limits on M, thereby simulating an infinite single-layer network; multiplying by a factor of $2/M$ and extending the discrete expected value to the continuous one:

$$\begin{aligned}
\lim_{M\to\infty}\frac{2}{M}f(\mathbf{x},\mathbf{x}') &= \lim_{M\to\infty}\frac{2}{M}\sum_{i=1}^{M}\Theta({\mathbf{w}_i}^\top\mathbf{x})\Theta({\mathbf{w}_i}^\top\mathbf{x}')({\mathbf{w}_i}^\top\mathbf{x})^n({\mathbf{w}_i}^\top\mathbf{x}')^n \\
&= 2\mathbb{E}_{\mathbf{w}\sim\mathcal{N}(0,I_d)}\left[\Theta({\mathbf{w}_i}^\top\mathbf{x})\Theta({\mathbf{w}_i}^\top\mathbf{x}')({\mathbf{w}_i}^\top\mathbf{x})^n({\mathbf{w}_i}^\top\mathbf{x}')^n\right] \\
&= 2\int d\mathbf{w}\frac{e^{-\frac{||\mathbf{w}||^2}{2}}}{(2\pi)^{d/2}}\Theta(\mathbf{w}^\top\mathbf{x})\Theta(\mathbf{w}^\top\mathbf{x}')(\mathbf{w}^\top\mathbf{x})^n(\mathbf{w}^\top\mathbf{x}')^n \\
&= k_n(\mathbf{x},\mathbf{x}')
\end{aligned}$$

Generalizing the idea, many interesting kernels on $\mathbb{R}^d$ may be defined pointwise as an expected value:

$$k(\mathbf{x},\mathbf{x}') = \mathbb{E}_{\mathbf{w}}[\Gamma(\mathbf{w}^\top\mathbf{x})\Gamma(\mathbf{w}^\top\mathbf{x}')],$$

where $\Gamma:\mathbb{R}\to\mathbb{R}$ is a non-linear (activation) function. This encoding can be seen as a one-layer neural network with an infinite number of random weights.

Kernel compositions. Consider a positive-definite (p.d.) kernel $k_1:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ and its RKHS $\mathcal{H}_1$ with mapping $\phi_1:\mathcal{X}\to\mathcal{H}_1$. Consider also a p.d. kernel $k_2:\mathcal{H}_1\times\mathcal{H}_1\to\mathbb{R}$ and its RKHS $\mathcal{H}_2$ with mapping $\phi_2:\mathcal{H}_1\to\mathcal{H}_2$. Then $k_3:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ given by $k_3(\mathbf{x},\mathbf{x}') := k_2(\phi_1(\mathbf{x}),\phi_1(\mathbf{x}'))$ is p.d. and its RKHS mapping is $\phi_3 = \phi_2\circ\phi_1$.

As an example, consider the composition (say, k^n_{ACRBF}) of the arc-cosine (AC) and the standard RBF kernel:

$$k_{\text{RBF}}(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma\left\|\mathbf{x}-\mathbf{x}'\right\|^2\right), \gamma>0$$

giving

$$k^n_{\text{ACRBF}}(\mathbf{x},\mathbf{x}') = \frac{1}{\pi}||\phi(\mathbf{x})||^n||\phi(\mathbf{x}')||^n J_n\left[\cos^{-1}\left(\frac{\langle\phi(\mathbf{x}),\phi(\mathbf{x}')\rangle}{||\phi(\mathbf{x})||\cdot||\phi(\mathbf{x}')||}\right)\right] \tag{6}$$

where

$$\begin{aligned}
||\phi(\mathbf{x})||^n &= (||\phi(\mathbf{x})||^2)^{\frac{n}{2}} = (\langle\phi(\mathbf{x}),\phi(\mathbf{x})\rangle)^{\frac{n}{2}} = (k_{\text{RBF}}(\mathbf{x},\mathbf{x}))^{\frac{n}{2}} \\
\frac{\langle\phi(\mathbf{x}),\phi(\mathbf{x}')\rangle}{||\phi(\mathbf{x})||\cdot||\phi(\mathbf{x}')||} &= \frac{k_{\text{RBF}}(\mathbf{x},\mathbf{x}')}{\sqrt{k_{\text{RBF}}(\mathbf{x},\mathbf{x})}\sqrt{k_{\text{RBF}}(\mathbf{x}',\mathbf{x}')}}
\end{aligned}$$

After all these transformations, the resulting expression for $k^n_{\text{ACRBF}}(\mathbf{x},\mathbf{x}')$ is feasible to calculate as:

$$(k_{\text{RBF}}(\mathbf{x},\mathbf{x}))^{\frac{n}{2}}(k_{\text{RBF}}(\mathbf{x}',\mathbf{x}'))^{\frac{n}{2}}J_n\left[\cos^{-1}\left(\frac{k_{\text{RBF}}(\mathbf{x},\mathbf{x}')}{\sqrt{k_{\text{RBF}}(\mathbf{x},\mathbf{x})}\sqrt{k_{\text{RBF}}(\mathbf{x}',\mathbf{x}')}}\right)\right] \tag{7}$$

Since the RBF function is a normalized kernel, it fulfills $k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}) = 1$ and thus the formula becomes:

$$k^n_{\mathrm{ACRBF}}(\mathbf{x}, \mathbf{x}') = \frac{1}{\pi} J_n\Big(\cos^{-1}\big(\exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)\big)\Big) \tag{8}$$

where $\gamma > 0$ and $n \in \mathbb{N}$ are free parameters.

Kernel iterations. Exploiting further the natural duality between inner products of features and kernels, multilayer derived kernels can be obtained by iterated compositions of a kernel with itself. The process can be carried out a fixed number of times (which would correspond to the "depth") or trying to compute the limiting kernel function. In any case, the obtained kernel can be used in place of standard kernels (*e.g.* using it in a SVM).

Random Fourier features. Initially introduced for large-scale kernel machines, they provide randomly drawn features that constitute unbiased estimates for infinite kernel expansions. Random Fourier features are among the most popular and widely applied constructions because they provide an easily computable, low-dimensional feature representation for continuous scale-invariant kernels [28].

In essence, for scale-invariant kernels $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ on $\mathbb{R}^d$, Bochner's theorem [29] states that k is a p.d. function if and only if it is the Fourier transform of a non-negative measure[3] $\rho(\mathbf{w})$. Letting $\Gamma_{\mathbf{w}}(\mathbf{x}) := \exp(\mathrm{i}\, \mathbf{w}^\top \mathbf{x})$, then

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w}\sim\rho}[\Gamma_{\mathbf{w}}(\mathbf{x})\Gamma_{\mathbf{w}}(\mathbf{x}')^*] = \int_{\mathbb{R}^d} \rho(\mathbf{w}) \exp\big(\mathrm{i}\, \mathbf{w}^\top(\mathbf{x} - \mathbf{x}')\big)\, \mathrm{d}\mathbf{w} \tag{9}$$

where * stands for complex conjugation. Monte Carlo methods can be applied to estimate this expectation and get an approximation of the original kernel $k(\mathbf{x}, \mathbf{x}')$ as a (sampled) inner product of paired cosine and sine features:

$$k(\mathbf{x} - \mathbf{x}') \approx \frac{1}{M} \sum_{i=1}^{M} \Gamma_{\mathbf{w}_i}(\mathbf{x})\Gamma_{\mathbf{w}_i}(\mathbf{x}')^* \qquad \mathbf{w}_i \overset{\text{i.i.d.}}{\sim} \rho \tag{10}$$

Thanks to Euler's formula, eq. (9) can be written

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w}\sim\rho}\Big[\cos(\mathbf{w}^\top \mathbf{x}) \cos(\mathbf{w}^\top \mathbf{x}') + \mathrm{i}\, \sin(\mathbf{w}^\top \mathbf{x}) \sin(\mathbf{w}^\top \mathbf{x}')\Big] \tag{11}$$

If the kernel k is real-valued (as is most often the case), then the imaginary parts above can be split to write eqs. (9) and (10) as:

$$k(\mathbf{x} - \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}')\rangle \approx \hat{\phi}(\mathbf{x})^\top \hat{\phi}(\mathbf{x}') \tag{12}$$

[3] If $k(\mathbf{0}) = 1$, then $\rho(\mathbf{w})$ is a normalized probability density function.

with $\hat{\phi}(\mathbf{x}) = \frac{1}{\sqrt{M}}[\cos(\mathbf{w}_1^\top \mathbf{x}), \ldots, \cos(\mathbf{w}_M^\top \mathbf{x}), \sin(\mathbf{w}_1^\top \mathbf{x}), \ldots, \sin(\mathbf{w}_M^\top \mathbf{x})]^\top$. In kernel methods, the representer theorem allows to express the solution function as an expansion over the input data:

$$f(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \mathbf{x})$$

which can in turn be expressed as a linear expansion in the random features:

$$f(\mathbf{x}) \approx \sum_{n=1}^{N} \alpha_n \hat{\phi}(\mathbf{x}_n)^\top \hat{\phi}(\mathbf{x}) = \hat{\mathbf{f}}^\top \hat{\phi}(\mathbf{x}),$$

where $\hat{\mathbf{f}} = \sum_{n=1}^{N} \alpha_n \hat{\phi}(\mathbf{x}_n)$. This explicit kernel approximation via random features delves into profound relations between inner products of features (the standard neural network function) and kernels. The quality of this approximation has been recently studied [30].

5 Conclusions and outlook

The mere existence and need for deep kernel methods could be arguable. Is cascading (large numbers of) non-linearities necessary to discover higher-order features? Is this the answer to beating the curse of dimensionality? There are in fact comparisons between deep learning models and shallow kernel machines where the latter are able to match the performance of state-of-the-art DNNs on TIMIT data for speech recognition and classification tasks [31].

The fact that kernel methods conform non-parametric models makes them different from the classical parametric flavor of neural networks. On the other hand, kernel methods can be linked to Gaussian Process models, and therefore they constitute a natural bridge to the vast field of Bayesian data analysis. The ideal method should be able to combine the structural properties of deep learning architectures with the non-parametric flexibility of kernel methods.

Computationally speaking, both kinds of methods are demanding and pose serious problems when applied to large datasets. Their hybridization could therefore bring benefits in both expressive power and scalability. In particular, we expect great benefits in fusing Convolutional Neural Networks with kernel methods, thereby combining the locality at several scales and compositional expressiveness of the former with the non-linearity and built-in regularization ability of the latter. An added advantage would be found in the opportunity to develop a better theoretical understanding or, at the very least, to gain analytical clarity.

We should not ignore that there are also some negative counterparts to using deep learning pervasively. A recent example would be found in the NLP area [32], the main argument being that natural language may not be continuous at the word level. For example, in the sentence *Peter likes apples* the proximity of *Peter* and *apples* is arbitrary rather than ontological, because *Peter* could like anything, a point quickly replied from the deep learning community [33].

Additional clear avenues for research are given by the use of supervision to better approximate the kernel for a specific task and to understand the properties of the feature spaces induced by these hybrid networks; the practical implementation of the known theoretical possibility of "emulation" of multilayer machines by shallow ones and vice versa; and the derivation of efficient layer-wise algorithms for training such networks.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism o f pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[3] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[4] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: accelerating recurrent network training for long or event-based sequences. *CoRR*, abs/1610.09513, 2016.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Bing Xu Mehdi Mirza, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[6] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[7] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[8] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856, 2001.

[9] Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10):2385–2404, 2000.

[10] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[11] Pei Ling Lai and Colin Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377, 2000.

[12] Francis R Bach and Michael I Jordan. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48, 2002.

[13] Ingo Steinwart, Philipp Thomann, and Nico Schmid. Learning with hierarchical gaussian kernels. *arXiv preprint arXiv:1612.00824*, 2016.

[14] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

[15] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

[16] Gaurav Pandey and Ambedkar Dukkipati. To go deep or wide in learning? In *AISTATS*, pages 724–732, 2014.

[17] Andreas C Damianou and Neil D Lawrence. Deep gaussian processes. In *AISTATS*, pages 207–215, 2013.

[18] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Practical learning of deep gaussian processes via random fourier features. *arXiv preprint arXiv:1610.04386*, 2016.

[19] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Accelerating deep gaussian processes inference with arc-cosine kernels. page ?????, 2016.

[20] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2014.

[21] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016.

[22] Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*, 2015.

[23] Ronan Collobert, Jason Weston, Léon Bottou, M̃ichael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[24] Julia Hirschberg and Christopher D. Manning. Advances in natural language processing. *Science*, 349:261–266, 2015.

[25] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*, 2016.

[26] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[27] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

[28] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, 2007.

[29] Walter Rudin. *Fourier analysis on groups*. John Wiley & Sons, 2011.

[30] Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random fourier features. In *Advances in Neural Information Processing Systems*, pages 1144–1152, 2015.

[31] Po-Sen Huang, Haim Avron, Tara N Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. Kernel methods match deep neural networks on timit. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 205–209. IEEE, 2014.

[32] Riza C. Berkan. Is google hyping it? why deep learning cannot be applied to natural languages easily. `https://www.linkedin.com/pulse/google-hyping-why-deep-learning-cannot-applied-easily-berkan-ph-d?trk=hp-feed-article-title-comment`, 2016.

[33] Riza C. Berkan. Neural networks are quite neat: a reply to riza berkan. `https://www.linkedin.com/pulse/neural-networks-quite-neat-reply-riza-berkan-yannick-versley`, 2017.