HAL
open science

# Deep Networks with Adaptive Nyström Approximation

Luc Giffon, Stéphane Ayache, Thierry Artières, Hachem Kadri

## To cite this version:

Luc Giffon, Stéphane Ayache, Thierry Artières, Hachem Kadri. Deep Networks with Adaptive Nyström Approximation. IJCNN 2019 - International Joint Conference on Neural Networks, Jul 2019, Budapest, Hungary. hal-02091661v2

**HAL Id: hal-02091661**

**https://hal.science/hal-02091661v2**

Submitted on 27 Nov 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Deep Networks with Adaptive Nyström Approximation

Luc Giffon, Stéphane Ayache, Thierry Artières and Hachem Kadri
*Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France*
firstname.name@lis-lab.fr

***Abstract*—Recent work has focused on combining kernel methods and deep learning to exploit the best of the two approaches. Here, we introduce a new architecture of neural networks in which we replace the top dense layers of standard convolutional architectures with an approximation of a kernel function by relying on the Nyström approximation. Our approach is easy and highly flexible. It is compatible with any kernel function and it allows exploiting multiple kernels. We show that our architecture has the same performance than standard architecture on datasets like SVHN and CIFAR100. One benefit of the method lies in its limited number of learnable parameters which makes it particularly suited for small training set sizes, e.g. from 5 to 20 samples per class.**

## I. Introduction

Kernel machines and deep learning have mostly been investigated separately. Both have strengths and weaknesses and appear as complementary family of methods with respect to the settings where they are most relevant. Deep learning methods may learn from scratch relevant features from data and may work with huge quantities of data. Yet they actually require large amount of data to fully exploit their potential and may not perform well with limited training datasets. Moreover deep networks are complex and difficult to design and require lots of computing and memory resources both for training and for inference. Kernel machines are powerful tools for learning nonlinear relations in data and are well suited for problems with limited training sets. Their power comes from their ability to extend linear methods to nonlinear ones with theoretical guarantees. However, they do not scale well to the size of the training datasets and do not learn features from the data. They usually require a prior choice of a relevant kernel amongst the well known ones, or even require defining an appropriate kernel for the data at hand.

Although most research in the field of deep learning seems to have evolved as a "parallel learning strategy" to the field of kernel methods, there are a number of studies at the interface of the two domains which investigated how some concepts can be transferred from one field to another. Mainly, there are two types of approaches that have been investigated to mix deep learning and kernels. Few works explored the design of deep kernels that would allow working with a hierarchy of representations as the one that has been popularized with deep learning [2], [7], [8], [17], [25], [29]. Other studies focused on various ways to plug kernels into deep networks [6], [15], [16], [30], [31]. This paper follows this latter line of research. Specifically, we propose a new kind of architecture which is built by replacing the top dense layers of a convolutional neural network by an adaptive approximation of a kernel function. A similar approach proposed in the literature is *Deep Fried Convnets* [30] which brings together convolutional neural networks and kernels via *Fastfood* [11], a kernel approximation technique based on random feature maps. We revisit this concept in the context of Nyström kernel approximation [28]. One key advantage of our method is its flexibility that enables the use of any kernel function. Indeed, since the Nyström approximation uses an explicit feature map from the data kernel matrix, it is not restricted to any specific family of kernel function. This is also useful when one wants to use or learn multiple different kernels instead of a single kernel function, as we demonstrate here. In particular we investigate two different ways of using multiple kernels, one is a straightforward extension of the single kernel version while the second is a variant that exploits a Nyström kernel approximation for each of the feature map output of the convolution

Our experiments on four datasets (MNIST, SVHN, CIFAR10 and CIFAR100) highlight three important features of our method. First our approach compares well to standard approaches in standard settings (using full training sets) while requiring a reduced number of parameters compared to full deep networks. This specific feature of our proposal makes it suitable for dealing with limited training set sizes as we show by considering experiments with tens or even fewer training samples per class. Finally the method may exploit multiple kernels, providing a new tool with which to approach the problem of Multiple Kernel Learning (MKL) [5], and enabling taking into account the rich information in multiple feature maps of convolution networks through multiple Nyström layers.

It should be noted that some recent works has focused on using the Nyström approximation in conjonction with neural networks. In [15], Nyström method was used to enable tractable computation in convolutional kernel networks by projecting data features in a space of low dimension. In [4], it was used to transform input data before feeding them to a neural network in order to achieve better interpretability. Finally, [24] used an ensemble of Nyström approximation and then applied a neural network to optimize a kernel machine. To our knowledge, the Nyström approximation itself has not been studied as a drop-in replacement for standard fully-connected layers in deep networks neither it has been used to adaptively learn a metric on the feature space induced by the convolution filters, as presented in our work.

The rest of the paper is organized as follows. We provide background on kernel approximation via the Nyström and the random Fourier features methods in Section II. The detailed configuration of the proposed *Adaptive Nyström Newtorks* is described in Section III. We also show in Section III how adaptive Nyström networks can be used with multiple kernels. Section IV reports experimental results on MNIST, SVHN, CIFAR10 and CIFAR100 datasets to first provide a deeper understanding of the behaviour of our method with respect to the choice of the kernels and the combination of these, and second to compare it to the usual fully-connected layers on classification tasks with respect to accuracy and to complexity issues, in particular in the small training set size setting.

## II. Background on Kernel Approximation

Kernel approximation methods have been proposed to make kernel methods scalable. Two popular methods are Nyström approximation [28] and random features approximation [20]. The former approximates the kernel matrix by an efficient low-rank decomposition, while the latter is based on mapping input features into a low-dimensional feature space where dot products between features approximate well the kernel function.

*a) Nyström approximation [28]:* It computes a low-rank approximation of the kernel matrix by randomly subsampling a subset of instances. Let consider a training set of $n$ training samples, $\{x_i \in \mathbb{R}^d, i = 1, .., n\}$, $\boldsymbol{K}$ be the kernel matrix defined as $\boldsymbol{K}(i,j) = k(\mathbf{x}_i, \mathbf{x}_j), \forall\ i,j \in [1, \ldots, n]$, and $\mathbf{L}$ be a subset of examples $\mathbf{L} = \{\mathbf{x}_i\}_{i=1}^m$ which is selected from the training set. Assuming the subset includes the first samples, or rearranging the training samples this way, $\boldsymbol{K}$ may be rewritten as:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{21}^T \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} \end{bmatrix},$$

where $\boldsymbol{K}_{11}$ is the Gram matrix on subset $\mathbf{L}$. Nyström approximation is obtained as follows

$$\boldsymbol{K} \simeq \tilde{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K}_{11} \\ \boldsymbol{K}_{21} \end{bmatrix} \boldsymbol{K}_{11}^{-1} \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{21}^T \end{bmatrix}.$$

From this approximation the Nyström nonlinear representation of a single example $\mathbf{x}$ is given by

$$\phi_{nys}(\mathbf{x}) = \mathbf{k}_{\mathbf{x},\mathbf{L}} \boldsymbol{K}_{11}^{-\frac{1}{2}}, \tag{1}$$

where $\mathbf{k}_{\mathbf{x},\mathbf{L}} = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_m)]^T$ with $\mathbf{x}_i \in \mathbf{L}$.

One key aspect of the Nyström methods is the sampling technique used to select informative columns from $\boldsymbol{K}$. It influences the subsequent approximation accuracy and thus the performance of the learning algorithm. See [10], [26] for more details and a comparison of various sampling methods for the Nyström approximation. It is of note that uniform sampling was adopted when the standard Nyström method was introduced [28]. It is a widely applied sampling method due to its low time consumption [26].

*b) Random features approximation [20]:* It computes a low-dimensional feature map $\tilde{\phi}$ of dimension $q$ such that $\langle \tilde{\phi}(\cdot), \tilde{\phi}(\cdot) \rangle = \tilde{k}(\cdot,\cdot) \simeq k(\cdot,\cdot)$. Two well-known instances of this method are *Random Kitchen Sinks* (RKS) [21] and *Fastfood* [11]. RKS approximates a *Radial Basis Function* (RBF) kernel using a random Fourier feature map defined as

$$\phi_{rks}(\mathbf{x}) = \frac{1}{\sqrt{p}} [\cos(\mathbf{Q}\mathbf{x}) \sin(\mathbf{Q}\mathbf{x})]^T, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{Q} \in \mathbb{R}^{q \times d}$ and $\mathbf{Q}_{i,j}$ are drawn randomly. If $\mathbf{Q}_{i,j}$ are drawn according to a Gaussian distribution then the method is shown to approximate the Gaussian kernel, i.e. $\langle \phi_{rks}(\mathbf{x}_1), \phi_{rks}(\mathbf{x}_2) \rangle \approx \exp(-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{\sigma})$ where $\sigma$ is the hyper-parameter of the kernel. Note that $\sigma$ is related to the parameters of the Gaussian distribution that generates the random features. Replacing the $\cos$ and $\sin$ activation functions by a Rectified Linear Unit (ReLU) allows to approximate the arc-cosine kernel instead of the RBF Gaussain kernel [2], [19].

The *Fastfood* method [11] is a variant of RKS with reduced computational cost for the Gaussian kernel. It is based on approximating the matrix $\mathbf{Q}$ in Eq. 2, when $q = d$, by a product of diagonal and hadamard matrices according to

$$\mathbf{V} = \frac{1}{\sigma\sqrt{d}} \mathbf{SHG\Pi HB},$$

where $\mathbf{S}$,$\mathbf{G}$ and $\mathbf{B}$ are diagonal matrices of size $d \times d$, $\mathbf{\Pi} \in \{0,1\}^{d \times d}$ is a random permutation matrix, $\mathbf{H}$ is a Hadamard matrix which does not requite to be stored, and $\sigma$ is an hyperparameter. Matrix $\mathbf{V}$ may be used in place of $\mathbf{Q}$ in Eq. 2 to define the *Fastfood* nonlinear representation map

$$\phi_{ff}(\mathbf{x}) = \frac{1}{\sqrt{p}} [\cos(\mathbf{V}\mathbf{x}) \sin(\mathbf{V}\mathbf{x})]^T. \tag{3}$$

Note that this definition requires $d$ to be a power of 2 to take advantage of the recursive structure of the Hadamard matrix. Note also that to reach a representation dimension $q > d$ one may compute multiple $\mathbf{V}$ and concatenate the corresponding $\phi_{ff}$.

In the next section we introduce deep adaptive Nyström networks. They combine efficiently kernel Nyström approximation with deep learning to learn nonlinear mappings between layers in a neural network architecture while requiring few parameters.

## III. Deep Adaptive Nyström Networks

Deep adaptive Nyström network is a deep learning architecture that replaces dense layers of a convolutional neural architecture by a *Nytsröm* approximation of a kernel function. This allows to take advantage of the low complexity cost in terms of computation and memory of the Nytsröm method to reduce significantly the computation cost and the number of parameters of the fully-connected layers in deep convolutional neural networks. It is of note that although we introduce deep Nyström networks in the context of deep convolutional networks, they are general as they can also be applied to other deep neural network architectures, can work with any kernel

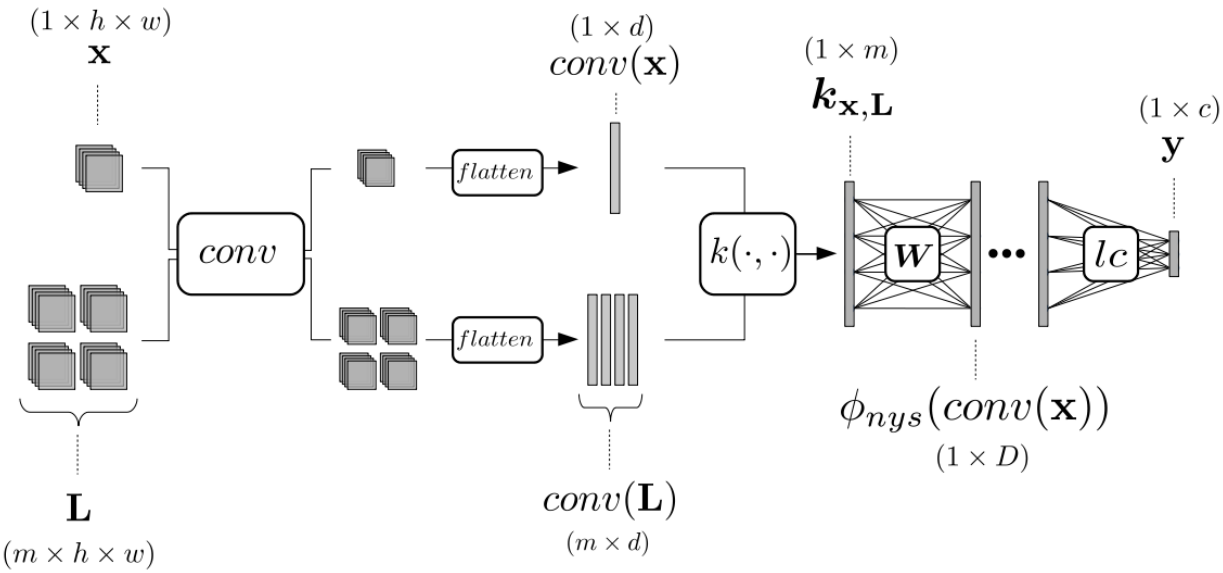


Fig. 1: The architecture we propose involves a usual convolutional part, *conv* , including multiple convolutional blocks, and a Nyström layer which is then fed to (eventually multiple) standard dense layers up to the classification layer. The Nyström layer computes the kernel between the output of the $conv$ block for a given input and the corrsponding representations of the trains samples in the subsample $L$ before applying a linear transformation $\mathbf{W}$ to the so obtained kernel vector $\mathbf{k}_{\mathbf{x},\mathbf{L}}$.

function or a combination of kernels, and do not require a vectorial representation of the features.

First, we start by revisiting the concept of Nyström kernel approximation from a feature map perspective.

*a) Nyström approximation from an empirical kernel map perspective:* The empirical kernel map is an explicit $n$-dimensional feature map that is obtained by applying the kernel function on the training input data [22]. It is defined as

$$\begin{aligned} \phi_{emp}: \quad & \mathbb{R}^d \to \mathbb{R}^n \\ & x \mapsto \big(k(x, x_1), \ldots, k(x, x_n)\big). \end{aligned} \tag{4}$$

An interesting feature of the empirical kernel map is that if we consider the inner product in $\mathbb{R}^n$ $\langle \cdot, \cdot \rangle_M = \langle \cdot, M \cdot \rangle$ with a positive semi-definite (psd) matrix $M$, we can recover the kernel matrix $K$ using the empirical kernel map by setting $M$ equals to the inverse of the kernel matrix. In other words, $K_{emp} := \big(\langle \phi_{emp}(x_i), \phi_{emp}(x_j) \rangle_{K^{-1}}\big)_{i,j=1}^n = K$. Since $K$ is a psd matrix, one can consider the feature $\phi'_{emp} : x \to K^{-1/2}\phi_{emp}(x)$ as an explicit feature map that allows to reconstruct the kernel matrix. This feature map is of dimension $n$ and then is not interesting when the number of example is large.

From an empirical kernel map point of view, $\phi_{nys}(\mathbf{x})$ (Equation 1) can be seen as an empirical kernel map [22] and $\boldsymbol{K}_{11}^{-\frac{1}{2}}$ as a metric in the empirical feature space. From this viewpoint, we think that it could be useful to learn a metric $\mathbf{W}$ in the empirical feature space instead of assuming it to be equal to $\boldsymbol{K}_{11}^{-\frac{1}{2}}$. In a sense, this should allow to learn a kernel by learning its Nyström feature representation. In the following, we call the setting where $\mathbf{W}$ is learned by the network as *deep adaptive Nyström*.

*b) Principle:* As illustrated in Figure 1, the model we propose is based on using the Nyström approximation to integrate any kernel function on top of convolutional layers of a deep net.

One main advantage of our method is its generic feature that enables the use of any kernel, or a combination of them. This is particularly useful if one wants to use multiple kernels or have no prior knowledge on which kernel to use for a particular task. Starting from an usual deep neural network, we replace the top fully-connected hidden layers with $\phi_{nys}$ (Equation 1). In order to compute the above Nyström representation of a sample $\mathbf{x}$ one must consider a subsample $\mathbf{L}$ of training instances. Since the kernel $k$ is computed on the representations given by convolutional layers, the samples in $\mathbf{L}$ must be represented in the same space, and hence must be processed by the convolutional layers as well. Once convolutional representations are calculated, the kernel function may be computed with an input sample and each instance in $\mathbf{L}$ in order to get the vector $\mathbf{k}_{\mathbf{x},\mathbf{L}}$, which is then linearly transformed by $\mathbf{W}$ before the linear classification layer (see Figure 1). However, one problem arises with the computation of $\mathbf{K}_{11}^{-\frac{1}{2}}$ which requires the computation of the Singular Value Decomposition (SVD) of $\mathbf{K}_{11}$ for each batch. In the case where the size of the subsample $\mathbf{L}$, $m$, is relatively large, the computational complexity of the SVD is $O(m^3)$ for a single batch whose size is likely to be of the same order of magnitude than $m$. This issue can be at least partially settled via *Adaptative-Nyström* network where, instead of setting the weights $\mathbf{W} = \boldsymbol{K}^{-\frac{1}{2}}$ as in Eq. 1, we learn these

weights as parameters of the network via gradient descent. In this Adaptive-Nyström scheme, the computational and storage complexity of our approach after the convolution are both $O(dm + m^2 + mc)$ against $O(dD + Dc)$ for standard fully-connected layers, with $d$ being the input dimension of the layer, $D$ the output dimension and $c$ the number of classes.

In a multiple kernels context, kernels $k^1, \ldots, k^l$ correspond to $l$ Nyström layers that can be computed in parallel then merged to encode the information provided by the different kernel representations. Learning the weights $\mathbf{W}_1, \ldots, \mathbf{W}_l$ in this case is, in a way, related to multiple kernel learning. This is a particularly interesting feature of our Nyström layer because it is sometimes difficult to know in advance which kernel will perform the best for a particular task, as demonstrated in Figure 2. One possible merging strategy for the different kernel representations is simply to concatenate them such as:

$$\phi_{nys_{mkl}} = \begin{bmatrix} \mathbf{W}_1 \boldsymbol{k}^1_{\mathbf{x},\mathbf{L}} \\ \vdots \\ \mathbf{W}_l \boldsymbol{k}^l_{\mathbf{x},\mathbf{L}} \end{bmatrix} \tag{5}$$

with $\boldsymbol{k}^i_{\mathbf{x},\mathbf{L}}$ being the kernel vector obtained using the $i^{\text{th}}$ kernel function $k^i$.

Comparing with *Deep Fried Convnets* that also consists in replacing the top dense layers of a convolutional neural network by using kernel approximation, we mention two main structural differences with our model: (I) Nyström approximation has the flexibility to use any kernel functions and to combine multiple kernels while the *Fastfood* approximation, used in the *Deep Fried Convnets*, is limited to shift-invariant kernels, and (II) in contrast to *Fastfood* the Nyström approximation is data dependent. The storage ($O(d+dc)$) complexity of the *Fastfood* approximation is lower than the Nyström approximation ($O(m^2 + mc)$) but we show in the experiments that this theoretical difference doesn't stand in practice, because the number of necessary examples is far lower than the dimension of the features in the output of the convolution blocks.

In the next section, we demonstrate the effectiveness of our method on some classical image classification datasets then we try to explore its possible extensions for few sample learning and multiple kernel learning, exploiting some well known advantages of kernel methods.

## IV. Experiments

We present a series of experimental results that explore the potential of deep adaptive Nyström networks with respect to various classification settings. First we consider a rather standard setting and compare our approach with standard models on image classification tasks. We explore in particular the behaviour of deep adaptive Nyström with various kernels and stress the very limited subsample size needed to achieve good accuracy performance. Next we investigate the use of Nyström layers in a small training set setting, which shows that our approach may allow to learn classes with only very few training samples, taking advantage of the reduced number of parameters learned by our model. Finally, we investigate the multiple kernel architecture and illustrate its interest when learning with RBF kernel to overcome the hyperparameter selection, and also we demonstrate the benefit of a multiple Nyström approach, combining kernels computed from individual feature maps.

Before describing all these results we detail the datasets used in our expriments.

### A. Experimental settings

We conducted experiments on four well known image classification datasets: MNIST [14], SVHN [18], CIFAR10 and CIFAR100 [9], details on these datasets are provided in Table I. We pretrained the convolutional layers using standard architectures on both datasets: Lenet [13] for MNIST and VGG19 [23] for SVHN, CIFAR10 and CIFAR100. We slightly modified the filters' sizes in Lenet network to ensure that the dimension of data after the convolution blocks is a power of 2 (needed for the *Deep Fried Convnets* architecture to which we compare). Those experiments focus on highlightling the potential of learning Adaptive Nyström and combination of them in a deep architecture. Learning jointly convolution layers and adaptive Nyström layer as an end-to-end strategy is also investigated and left for future work.

We compare three architectures in all conducted experiments. Pretrained convolutional parts are shared by the three architectures, which differ from the layers on top of it: (1) *Dense* architectures use dense hidden layers, i.e. these are classical convnets architectures ; (2) *Deep Fried* implements the *Fastfood* approximation (Equation 3) ; (3) *Nyström* stands for our proposal.

For *Dense* architectures, we considered one hidden layer with *relu* activation function, and varied the output dimension as $\{2, 4, 8, 16, 32, 64, 128, 1024\}$ in order to highlight accuracies as a function of the number of parameters. Exploiting more hidden layers did not significantly increase performances, in our experiments. We hence let it to one in order to ease the readability of figures. For the *Fastfood* approximation in *Deep Fried Convnets* we consider that $\phi_{ff}$ is gained with one stack of random features to form $\mathbf{V}$ in equation 3, except in the experiments of section IV-C which yields a representation dimension up to 5 times larger. Regarding our approach and $\phi_{nys}$, we varied the subset size $\mathbf{L} \in \{2, 4, 8, 16, 32, 64, 128\}$, we compared linear, RBF and Chi2 kernels, and we chose as output dimension of $\mathbf{W}$ the same size as $\mathbf{K}_{11}^{-\frac{1}{2}}$, corresponding to the subset sample size. Finally we explored the adaptive as well as non-adaptive variants.

Models were learned to optimize the cross entropy criterion with Adam optimizer and a gradient step fixed to $1e^{-4}$. By default the RBF bandwidth was set to the inverse of the mean distance between convolutional representations of pairs of training samples. All experiments were performed with Keras [3] and Tensorflow [1].

The experiments below investigate the potential of our architecture. We do not aim to beat current state-of-the-art results on the datasets considered due to our limited resources with respect to the amount of necessary experiments.

| Dataset | Input shape | # classes | Training set size | Validation set size | Test set size |
|---|---|---|---|---|---|
| MNIST | $(28 \times 28 \times 1)$ | 10 | 40 000 | 10 000 | 10 000 |
| SVHN | $(32 \times 32 \times 3)$ | 10 | 63 257 | 10 000 | 26 032 |
| CIFAR10 | $(32 \times 32 \times 3)$ | 10 | 50 000 | 10 000 | 10 000 |
| CIFAR100 | $(32 \times 32 \times 3)$ | 100 | 50 000 | 10 000 | 10 000 |

TABLE I: Datasets statistics

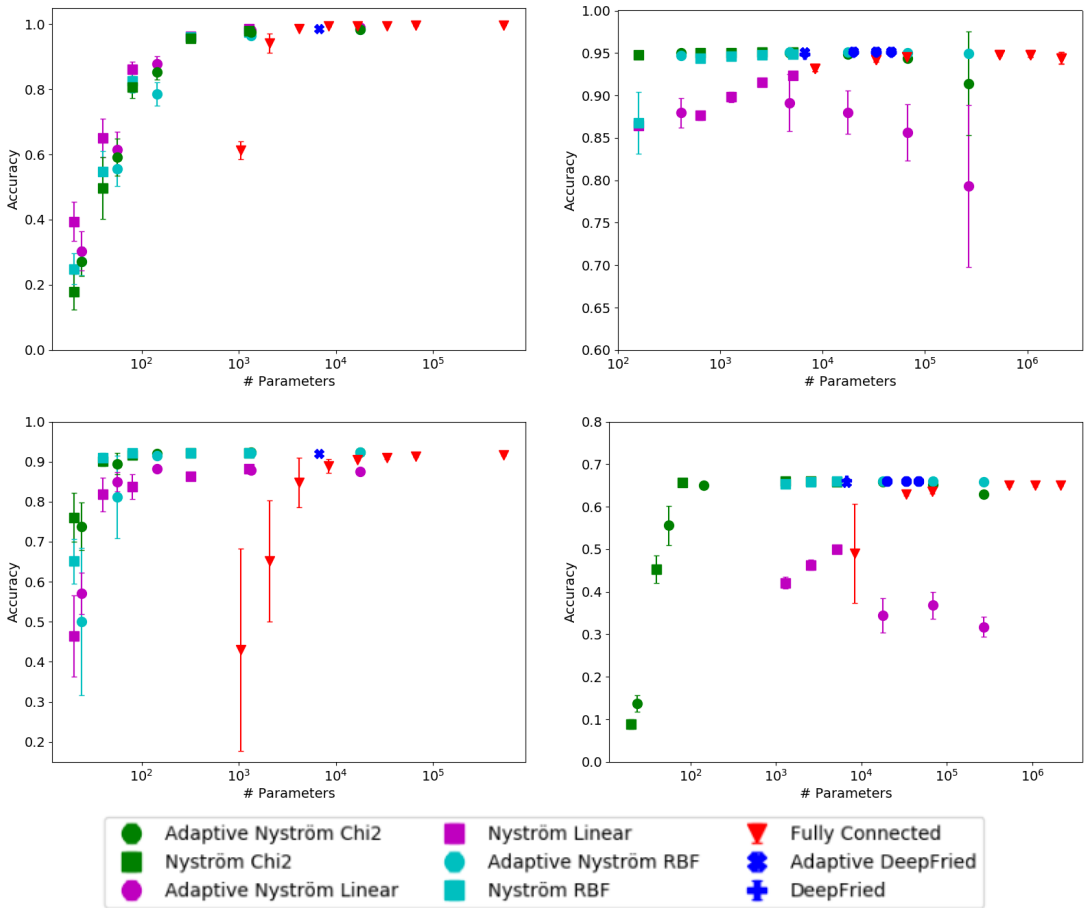


Fig. 2: Accuracy of models as a function of the number of parameters (*conv* part not included) for various kernels on MNIST (top-left), SVHN (top-right), CIFAR10 (bottom-left) and CIFAR100 (bottom-right) datasets. RBF is a shorthand for Gaussian RBF.

Consequently, we did not use tricks such as data augmentation and extensive tuning and, in particular, we did not use the best known convolutional architecture for each of the dataset, we rather used reasonable deep architectures: VGG19 [23] for the three datasets CIFAR10, CIFAR100 and SVHN and Lenet [12] for the MNIST dataset. We then fix a shared convolutional model (VGG19) and compare results gained with classification layers formed by either Nyström approximation, Fastfood approximation, or fully connected layers. Finally, although some work has been done on more advanced sampling schemes for the Nyström approximation, we adopted a stratified uniform sampling of the examples from the labeled training set as it has been shown in [10] to be a good cost/performance ratio sampling strategy.

### B. *Exploring the potential of the method*

We compare now deep adaptive Nyström networks to two similar architectures, *Deep Fried Convnets* and classical convolutional networks (inspired from VGG19 and Lenet depending on the dataset). We vary the number of parameters of each architecture in order to highlight classification accuracy with respect to needed memory space.

Figure 2 shows the compared networks accuracy with respect to the number of estimated parameters (e.g. variables) in the last representation layer plus the linear classification layer. Note that we ignore parameters for convolutions layers to ease the readability of figures since they are all equals in the compared models. We repeated each experiments 10 times and plotted average scores with standard deviations. Nyström results of increasing complexity (number of parameters) correspond to

the use of a subsample of increasing size from 2 (leftmost point) to 128 (rightmost point). One may see that there is no need of a large subsample here. This may be explained since the convolutional part of the network has been learned to yield quite robust and stable representations of input images. We provide a figure in Section IV-E to illustrate this (see Figure 4).

Deep adaptive Nyström network is able to reach the same accuracy performance while using much fewer parameters than classical fully connected layers. Moreover, we also observe smaller variations that points out the robustness of our model. The flexibility in the choice of the kernel function is a clear advantage of our method, as illustrated, since the best kernel is clearly dependent on the dataset (linear on MNIST, Chi2 on SVHN and CIFAR100, RBF on CIFAR10). We show for instance a gain by using the Chi2 Kernel ($k(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{x}_1 - \mathbf{x}_2||^2/(\mathbf{x}_1 + \mathbf{x}_2)$) that had been used for image classification [27]. We also notice the benefit of adaptive variants of Nyström layer, suggesting that our model is able to learn a useful Kernel map, more expressive than one obtain from conventional Nyström approximation, and with fewer parameters than Fastfood approximation.

### C. Learning with small training sets

Here we explore the ability of our model to work with few training samples, from very few to tens of samples per class. It is an expected benefit of the method since the use of kernels could take advantage of small training samples.

These preliminary experiments aim to show how the final layers of a convolutional model may be learned from very few samples, given a frozen convolutional model. We actually performed the following experiments by exploiting a trained convolution model that has been learned on the full training set and investigate the performance of deep adaptive Nyström as a function of the training set used to learn the classification layers. One perspective of this work is to exploit such a strategy for domain adaptation settings where the convolutional model is trained on a training set within a different domain as the classes to be recognized.

Based on already trained convolution models, we leverage on the additional information that one may easily include in our models, which is brought by the subsample set. Notice that this subsample may include unlabeled samples since their labels are not used for optimizing the model. Table II reports the comparison of network architectures on four datasets. We consider *Adaptive Nyström* using Linear, RBF or Chi2 kernels and compare with *Dense* and *Adaptive Deepfried* for training set sizes of 5, 10 and 20 samples per class. We only consider here adaptive variants since they brought better results than their non adaptive counterparts. We obtain models with different complexities: by increasing the hidden layer size in standard convolutional models, or by stacking the number of matrices $V$ in *DeepFried* (up to 8 times, more was untractable on our machines), and by increasing the subset size for the Nyström approximation. Reported results are averaged over 30 runs.

One may see first that deep adaptive Nyström networks outperfom baselines on every setting except for 5 training samples per class on MNIST. The linear kernel performs well on MNIST but is significantly worse than baselines on harder datasets. At the opposite, deep adaptive Nyström with both RBF Gaussian kernel and Chi2 kernel significantly outperfom Adaptive DeepFried for all the datasets and perform equaly or significantly better than Dense architectures on the hardest CIFAR100 dataset. Intriguingly, the *Deep Fried Convnets* performs particularly bad on this setting. Moreover one sees that no single kernel based Nyström representation dominate on all settings, showing the potential interest of combining multiple kernels as following experiments will show.

### D. Multiple kernel learning

We report here results validating the multiple kernels strategies, that we described in section III and equation 5. We conducted two different experiments:

First, we considered a combination of RBF kernels with various bandwidths and for different subsample sizes.In this case, each kernel function of $\{k^i\}_{i=1}^l$ is the RBF function with a different sigma value. This multiple kernel strategy exploits kernels defined with various values of the hyperparameter and allows to automatically handle tuning of hyper-parameter which usually requires heavy cross validation. Figure 3 shows the accuracy on CIFAR10 dataset as a function of $\sigma$ value, where the performance of the multiple kernel Nyström is shown as a horizontal line. Plots report results for various subsample size equal to 2 (left), 4 (middle) and 8 (right), averaged over 10 runs. As may be seen, using our Multiple kernel network allows adapting the kernel combination optimally from the data without requiring any prior choice on the RBF bandwith.

Second, we investigated a variant of the Nyström architecture that we call *Multiple Nyström* approximations. Here we consider in parallel multiple Nyström approximations where kernels are dedicated to deal each with the output of a single feature map from the $conv$ layers. Let $conv(x)_i$ be the $i^{\text{th}}$ feature map of $x$; in *Multiple Nyström*, we use $k^i(conv(x), conv(y)) = k(conv(x)_i, conv(y)_i)$. Table III reports results on CIFAR100. We show the best performances obtained for each method by grid-searching on various hyper-parameters for each model, within a similar range of number of parameters. For *Dense* model, we considered one or two hidden layers of 16, 64, 128, 1024, 2048 or 4096 neurons. *Deepfried* is the adaptive variant where we varied the number of stacks in 1, 3, 5, 7. We use deep adaptive Nyström with subsamples of size 16, 64, 128, 256, 512. We observe that both Nyström layers outperform the considered baselines, demonstrating the interest in combining Nyström approximations.

### E. 2D representations

Figure 4 plots the 2-dimensional $\phi_{nys}$ representations of CIFAR10 test samples obtained with a subsample of size equal to 2 (while the number of classes is 10) and two different kernels. One may see here that the 10 classes are already significantly well separated in this low dimensional representation space, illustrating that a very small sized subsammple is already powerfull. Beside, we experienced that

| | MNIST | | SVHN | | CIFAR10 | | CIFAR100 | |
|---|---|---|---|---|---|---|---|---|
| | 5 | 20 | 5 | 20 | 5 | 20 | 5 | 20 |
| Dense | **49.7** (4) | 94.4 (0.5) | 65.6 (11.6) | 81.7 (3.9) | 39.1 (3.3) | 87.1 (3.7) | 19.2 (2.2) | 35.7 (2.7) |
| Adaptive-Deepfried | 12.4 (3.3) | 12.4 (1.4) | 16.7 (5) | 21.0 (6.4) | 28.3 (9.2) | 41.2 (3.6) | 3.9 (1.2) | 6.4 (0.8) |
| Adaptive-Nyström-L | 48.1 (5.5) | 95.0 (0.5) | 22.4 (6.9) | 29.6 (13.5) | 12.0 (5.6) | 27.8 (7.6) | 1.2 (0.6) | 1.9 (0.8) |
| Adaptive-Nyström-R | 41.2 (7.7) | **95.5** (0.3) | 42.1 (29.6) | 53.5 (33.6) | **70.8** (4.4) | **92.2** (0.1) | **24.7** (2.6) | **62.1** (1.2) |
| Adaptive-Nyström-C | 26.4 (7.7) | 92.3 (1.8) | **89.6** (3.1) | **93.3** (1.3) | 67.1 (4.7) | **92.2** (1) | 20.2 (2.2) | 55.4 (1.9) |

TABLE II: Classification accuracy of Dense layers architectures; Adaptive DeepFried, Deep Adaptive Nyström with linear (L), gaussian RBF (R), Chi2 (C) kernels, on small training sets with 5 and 20 training samples per class. Variance of results, computed on 30 runs, are given in brackets.

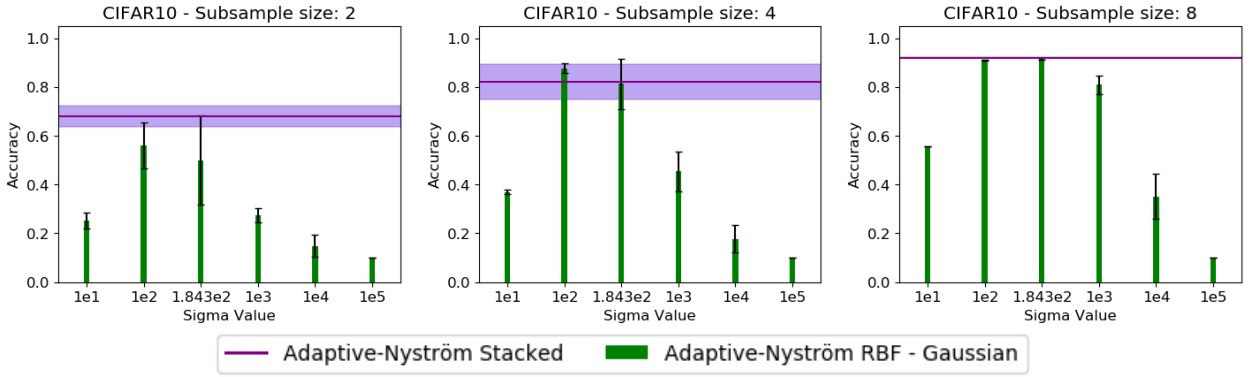


Fig. 3: Comparison of *Multiple Kernels* that combines RBF kernels with various values of the bandwidth hyper-parameter. Multiple kernels performance is shown as an horizontal line while single kernel using one specific value of the bandwidth hyper-parameter $\sigma$. Plots correspond to a subsample size equal to 2 (left), 4 (middle) and 8 (right).

| **Model** | **Accuracy (std)** | **Architecture** |
|---|---|---|
| Dense | 68.0 (0.7) | 1 hidden layer 1024 neurons |
| Adaptive-Deepfried | 67.6 (0.5) | 5 stacks |
| Adaptive-Nyström | **69.1** (0.2) | 256 subsamples + 512 Linear Kernels |
| Adaptive-Nyström | 67.6 (0.2) | 16 subsamples + 512 Chi2 Kernels |

TABLE III: *Multiple Nyström* experiments on CIFAR100 obtained on top of VGG19 convolutions.

applying Nyström on lower level features output by lower level convolution blocks may yield good performance as well while requiring larger subsamples.

## V. Conclusion

We proposed deep Adaptive Nyström networks that can be used as a drop-in replacement for dense layers and hence form a new hybrid architecture that mixes deep networks and kernel methods. It is based on the Nyström approximation that allows considering any kind of kernel function in contrast to explicit feature map kernel approximations. Our proposal reaches the accuracy performance of standard fully-connected layers while significantly reducing the number of parameters on various datasets, enabling in particular, learning from few samples. Moreover the method allows to easily deal with multiple kernels and with multiple Nyström variations.

## Acknowledgement

This work was funded in part by the French national research agency (grant number ANR16-CE23-0006). Finally, we would like to thank Ama Marina Kreme for her help and discussions.

## References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[2] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

[3] François Chollet et al. Keras. https://keras.io, 2015.

[4] Danilo Croce, Daniele Rossini, and Roberto Basili. Explaining non-linear classifier decisions within kernel-based deep architectures. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 16–24, 2018.

[5] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of machine learning research*, 12(Jul):2211–2268, 2011.

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

[7] Uri Heinemann, Roi Livni, Elad Eban, Gal Elidan, and Amir Globerson. Improper deep kernels. In *Artificial Intelligence and Statistics*, pages 1159–1167, 2016.

[8] Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma. Local deep kernel learning for efficient non-linear svm prediction. In *International Conference on Machine Learning*, pages 486–494, 2013.

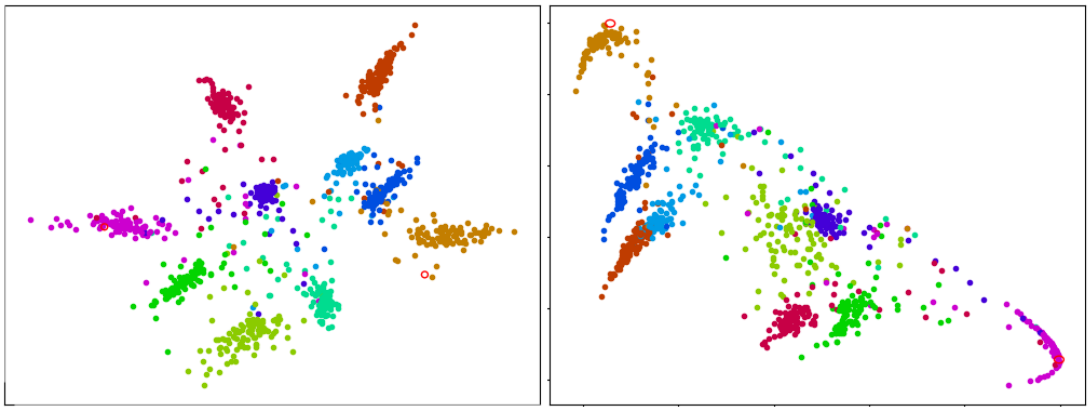

Fig. 4: 2-dimensional $\phi_{nys}$ representation of 1000 randomly selected test set samples from CIFAR-10 dataset, obtained with a subsample set of size 2 and a linear kernel (left) or Chi2 kernel (right). Each color represents a class.


[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[10] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

[11] Quoc V. Le, Tamas Sarlos, and Alex Smola. Fastfood-computing hilbert space expansions in loglinear time. In *International Conference on Machine Learning*, 2013.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[14] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits.

[15] Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in neural information processing systems*, pages 1399–1407, 2016.

[16] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635, 2014.

[17] Grégoire Montavon, Mikio L Braun, and Klaus-Robert M utller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12(Sep):2563–2581, 2011.

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[19] Gaurav Pandey and Ambedkar Dukkipati. Learning by stretching deep networks. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1719–1727, Bejing, China, 22–24 Jun 2014. PMLR.

[20] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In *In Neural Infomration Processing Systems*, 2007.

[21] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1313–1320. Curran Associates, Inc., 2009.

[22] Bernhard Scholkopf, Sebastian Mika, Chris JC Burges, Philipp Knirsch, K-R Muller, Gunnar Ratsch, and Alex J Smola. Input space versus feature space in kernel-based methods. *IEEE transactions on neural networks*, 10(5):1000–1017, 1999.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[24] Huan Song, Jayaraman J Thiagarajan, Prasanna Sattigeri, and Andreas Spanias. Optimizing kernel machines using deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

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

[26] Shiliang Sun, Jing Zhao, and Jiang Zhu. A review of nyström methods for large-scale machine learning. *Information Fusion*, 26:36–48, 2015.

[27] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):480–492, March 2012.

[28] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[29] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[30] Z. Yang, M. Moczulski, M. Denil, N. d. Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *2015 IEEE International Conference on Computer Vision (ICCV)*, volume 00, pages 1476–1483, Dec. 2015.

[31] Shuai Zhang, Jianxin Li, Pengtao Xie, Yingchun Zhang, Minglai Shao, Haoyi Zhou, and Mengyi Yan. Stacked kernel network. *arXiv preprint arXiv:1711.09219*, 2017.