DEEPSTRÖM NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work has focused on combining kernel methods and deep learning. With this in mind, we introduce *Deepström* networks – a new architecture of neural networks which we use to replace top dense layers of standard convolutional architectures with an approximation of a kernel function by relying on the Nyström approximation. Our approach is easy highly flexible. It is compatible with any kernel function and it allows exploiting multiple kernels. We show that Deepström networks reach state-of-the-art performance on standard datasets like SVHN and CIFAR100. One benefit of the method lies in its limited number of learnable parameters which make it particularly suited for small training set sizes, e.g. from 5 to 20 samples per class. Finally we illustrate two ways of using multiple kernels, including a multiple Deepström setting, that exploits a kernel on each feature map output by the convolutional part of the model.

1 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; 14; 7; 6; 20; 23). Other studies focused on various ways to plug kernels into deep networks (13; 24; 5; 12; 25). This paper follows this latter line of research, it focuses on convolutional networks. Specifically, we propose *Deepström* networks which are built by replacing dense layers of a convolutional neural network by an adaptive approximation of a kernel function. Our work is inspired from *Deep Fried Convnets* (24) which brings together convolutional neural networks and kernels via *Fastfood* (9), a kernel approximation technique based on random feature maps. We revisit this concept in the context of Nyström kernel approximation (22). 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 a specific kernel function and not limited only to RBF kernels, as in *Fastfood* approximation. This is particularly 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 to using multiple kernels while the second is a multiple *Deepström* variant that exploits a Nyström kernel approximation for each of the feature map output by the convolutional part of the neural network.

Furthermore the specific nature of our architecture makes it use only a limited number of parameters, which favours learning with small training sets as we demonstrate on targeted experiments.

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 ful training sets) while requiring a reduced number of parameters compared to full deep networks and of the same order of magnitude as *Deep Fried* Convnets. 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) (4), and enabling taking into account the rich information in multiple feature maps of convolution networks through multiple *Deepström* layers.

The rest of the paper is organized as follows. We provide background on kernel approximation via the Nyström and the random Fourier features methods and describe the *Deep Fried Convnet* architecture in Section 2. The detailed configuration of the proposed *Deepström* network is described in Section 3. We also show in Section 3 how *Deepström* networks can be used with multiple kernels. Section 4 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 state of the art baselines on classification tasks with respect to accuracy and to complexity issues, in particular in the small training set size setting.

2 Background on Kernel Approximation and Deep Fried Convnets

Kernel approximation methods have been proposed to make kernel methods scalable. Two popular methods are Nyström approximation (22) and random features approximation (16). 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.

Nyström approximation (22) 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, $\left\{x_i \in \mathbb{R}^d, i = 1,..,n\right\}$, $\boldsymbol{K}$ 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 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 $\mathbf{x}_i \in \mathbf{L}$.

Random features approximation (16) 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 instances of this method are *Random Kitchen Sinks* (RKS) and *Fastfood* (17; 9). RKS approximates a *Radial Basis Function* (RBF) kernel using a random feature map defined as

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

where $\mathbf{z} \in \mathbb{R}^p$, $\mathbf{Q} \in \mathbb{R}^{q \times p}$ 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 σ is the hyper-parameter of the kernel. Note that σ is related to the parameters of the Gaussian distribution that generate the random features.

The *Fastfood* method (9) 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 $q = p$, by a product of diagonal and hadamard matrices according to

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

where $\mathbf{S}$,$\mathbf{G}$ and $\mathbf{B}$ are diagonal matrices of size $p \times p$, $\mathbf{\Pi} \in \{0,1\}^{p\times p}$ is a random permutation matrix, $\mathbf{H}$ is a Hadamard matrix which does not requite to be stored, and σ is an hyperparameter.

Matrix $\mathbf{V}$ may be used in place of $\mathbf{Q}$ in Eq. 2 to define the *Fastfood* nonlinear representation map

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

Note that this definition requires p 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 $q > p$ one may compute multiple $\mathbf{V}$ and concatenate the corresponding ϕ_{ff}.

***Deep Fried Convnets* (24)** Our attention in this work is especially focused on combining kernel approximation with deep learning architecture. *Deep Fried Convnets* is a deep learning architecture that replaces dense layers of a convolutional neural architecture by a *Fastfood* approximation of a kernel. This allows to take advantage of the low complexity cost in terms of computation and memory of *Fastfood* to reduce significantly the computation cost and the number of parameters of the fully-connected layers of the deep convolutional neural network. More formally, let $conv(\mathbf{x})$ be the representation of the data sample $\mathbf{x}$ learned by a convolutional neural network. It may include a number of convolution blocks, each including convolution and pooling layers, batch normalization and nonlinear activation. In Deep Fried Convnets, an input $\mathbf{x} \in \mathbb{R}^d$ is mapped to the representation spaces $conv(\mathbf{x}) \in \mathbb{R}^p$ and then the Fastfood feature map ϕ_{ff} is applied to the convolutional representation $conv(\mathbf{x})$ instead of the fully-connected layers. The feature representation of $\mathbf{x}$ with Deep Fried Convnets is then $(\phi_{ff} \circ conv)(\mathbf{x}) \in \mathbb{R}^q$. It is of note that this method is dedicated to RBF kernels. In (24), wo architectures have been proposed. The first one relies on the *Fastfood* kernel approximation method as described above. The second one is a variant of *Fastfood* called *Adaptive-Fastfood*. It involves learning the weights of matrices $\mathbf{S}$, $\mathbf{G}$ and $\mathbf{B}$ through gradient descent rather than setting them randomly, while matrices $\mathbf{\Pi}$ and $\mathbf{H}$ are kept unchanged.

In the next section we introduce *Deepström* Networks as an alternative to Deep Fried Convnets. They are based on Nyström approximation and are not limited to RBF kernels. They also allow the use of multiple different kernels and find an appropriate kernel function.

3 *Deepström* NETWORKS

In this section, we describe our new *Deepström* model which combines the desirable characteristics of Nyström approximation and convolutional neural networks. First, we start by revisiting the concept of Nyström kernel approximation from a feature map perspective.

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 data x_i (18). It is defined as

$$\phi_{emp}: \quad \mathbb{R}^d \to \mathbb{R}^n \tag{4}$$

$$x \mapsto \big(k(x,x_1),\ldots,k(x,x_n)\big). \tag{5}$$

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

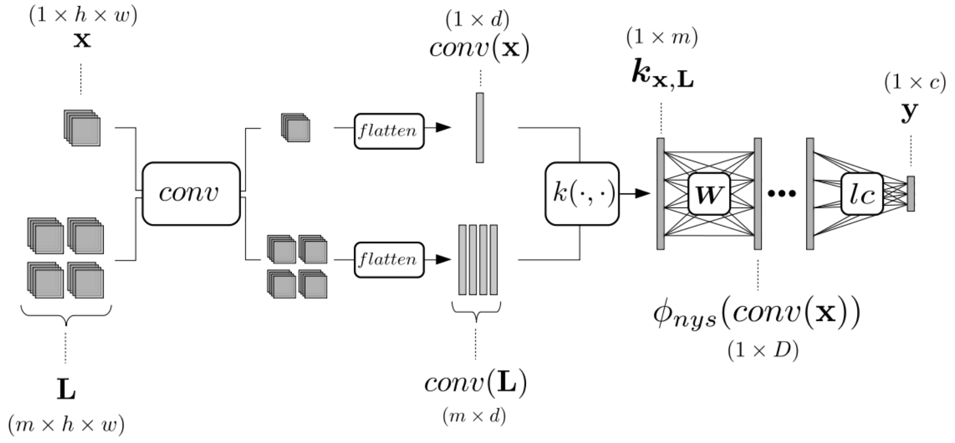

Figure 1: The *Deepström network* architecture involves a usual convolutional part, *conv* , including multiple convolutional blocks, and a *Deepström* layer which is then fed to (eventually multiple) standard dense layers up to the classification layer. The *Deepström* layer computes the kernel between the output of the $conv$ block for a given input and the corrsponding representations of the trains samples in the subsample L.

kernel matrix. This feature map is of dimension n and then is not interesting when the number of example is large. The feature map of the Nyström approximation is given by

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

where $\mathbf{k}_{\mathbf{x},\mathbf{L}} = [k(\mathbf{x},\mathbf{x}_1),\ldots,k(\mathbf{x},\mathbf{x}_m)]^T$ with $\mathbf{x}_i \in \mathbf{L}$. From an empirical kernel map point of view,$\phi_{nys}(\mathbf{x})$ can be seen as an "empirical kernel map" (18) 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 *Adapative Deepström Network*.

Principle *Deepström* networks we propose are an alternative to *Deep Fried Convnets*. They are based on using the Nyström approximation rather than the *Fastfood* one to integrate any kernel function on top of convolutional layers of a deep net. Indeed, although Deep Fried Convnets yield state-of-the-art results with a significant gain with respect to memory resource and to inference complexity, it is restricted to the Gaussian kernel *Fastfood*, which may not be always the best choice in practice. In addition, our method can deal with multiple kernels. *Deepström* nets are Neural Networks that make use of a nonlinear representation function computed with the Nyström approximation (see Figure 1). Starting from *Deep Fried Convnets* we replace ϕ_{ff} with ϕ_{nys} so that a *Deepström* net implements a function $f(\mathbf{x}) = (lc \circ \phi_{nys} \circ conv)(\mathbf{x})$. 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 $\mathbf{k}_{\mathbf{x},\mathbf{L}}$, which is then linearly transformed by $\mathbf{W}$ before the linear classification layer (see Figure 1).

Two main structural differences between *Deep Fried Convnets* and our *Deepström* nets: (I) Nyström has the flexibility to use different kernel functions and to combine multiple kernels, and (II) in contrast to *Fastfood* the Nyström approximation is data dependent. 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 $\mathbf{K}_{11}$. In the case where the size of the subsample $\mathbf{L}$, m, is large, the computational complexity of the SVD is $O(m^3)$.This issue can be at least partially settled via *Adaptative-Deepström* network where, instead of setting the weights $\mathbf{W}$ as in Eq. 3, we learn these weights as parameters of the

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 1: Datasets statistics

network via stochastic gradient descent. In the case of multiple kernels, $k_1, \dots, k_l$, l *Deepström* layers can be computed in parallel then merged to encode the information provided by the different kernel representations. Learning the weights $\mathbf{W}_1, \dots, \mathbf{W}_l$ in this case is, in a way, related to multiple kernel learning. Alternatively one may exploit a *Deepström* layer on top of each of the output feature map by the convolutional part $conv$ and concatenate these as input to a classification layer, we call this a multiple *Deepström* architecture hereafter.

4 Experiments

We present a series of experimental results that explore the potential of *Deepström* 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 *Deepström* networks with various kernels and stress the very limited subsample size needed to reach state-of-the-art accuracy. Next we investigate the use of *Deepström* networks in a small training set setting, which shows that our approach may allow to learn new classes with only very few training samples, taking advantage of the reduced number of parameters learned by our model. Before describing all these results we detail the datasets used. Finally we investigate first the multiple kernel architecture and illustrate its interest when learning with RBF kernel to overcome the hyperparameter selection, and second we demonstrate the benefit of a multiple *Deepström* approach, combining kernels computed from individual feature maps.

4.1 Experimental settings

We conducted experiments on four well known image classification datasets: MNIST (11), SVHN (15), CIFAR10 and CIFAR100 (8), details on these datasets are provided in Table 1. We pretrained the convolutional layers using standard architectures on both datasets: Lenet (10) for MNIST and VGG19 (19) 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).

We compare three convolutional 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) *Deepstrom* 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. For the *Fastfood* approximation in *Deep Fried Convnets* we consider that ϕ_{ff} is gained with one stack of random features to form **V** in equation 3, except in the experiments of section 4.3 which yields a representation dimension up to 5 times larger. Regarding our approach ϕ_{nys}, we varied the subset size $\mathbf{L} \in \{2, 4, 8, 16, 32, 64, 128\}$, we tested with the linear, the RBF, and the Chi2 kernels, and we chose as output dimension the same size as 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 $1e^{-4}$. Dropout was used on representation layers with probability equal to 0.5. By default the RBF bandwidth was set to the inverse of the mean distance between the representations, after the convolutional part, of pairs of training samples. All experiments were performed with Keras (3) and Tensorflow (1).

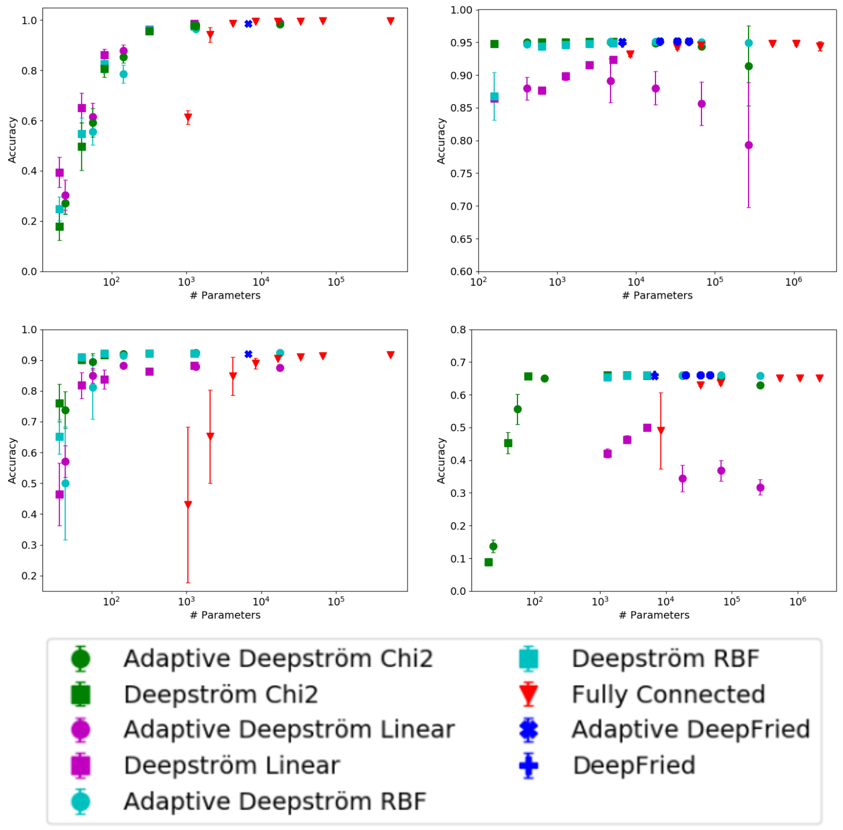

Figure 2: Accuracy of models as a function of the number of parameters (*conv* part not included) for various kernels on MNIST (top-left), SVHN (top-right), CIFAR10 (bottom-left) and CIFAR100 (bottom-right) datasets.

Note that the aim of all the experiments below is to investigate the potential of out architecture, not to reach or beat state of the art results on the datasets considered. We then compare results gained with our architecture and with state-of-the-art models, given a shared convolutional model. 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 a reasonable deep architecture, VGG19, for the three datasets CIFAR10, CIFAR100 and SVHN.

4.2 Exploring the potential of the method

We compare now *Deepström 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 parameters, and ignore parameters for convolutions layers to ease the readability. We repeated each experiments 10 times and plot average scores with standard deviations. *Deepström* models of increasing complexity (number of parameters) correspond to the use of 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 the Appendix that illustrates this.

The *Deepström network* is able to reach state-of-the-art performance using much fewer parameters than both classical networks and *Deep Fried Convnets*. 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 out method. The best kernel is clearly dependent on the dataset (linear on MNIST, Chi2 on SVHN and CIFAR100, RBF on CIFAR10). While *Random Features* in *DeepFried* are restricted to RBF kernels, 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 (21). We also notice the benefit of adaptive variants of *Deepström* model, suggesting that our model is able to learn and adapt useful Kernel function.

Finally, note that we obtained very similar results with neural architectures exploiting two hidden layers instead of one after the convolution module $conv$.

4.3 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.

Note that we do not exactly deal with a real small training set setting. 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 $conv$ that has been learned on the full CIFAR100 training set and investigate the performance of *Deepström* architectures 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.

Having at our disposal such a trained convolution model $conv$, 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 2 reports the comparison of network architectures on four datasets. We consider *Adaptive Deepström* 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 in *Deepström*. Reported results are averaged over 30 runs.

	MNIST		SVHN		CIFAR10		CIFAR100	
	5	20	5	20	5	20	5	20
D	**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)
ADF	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)
ADSL	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)
ADSR	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)
ADSC	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 2: Classification accuracy of Dense layers architectures (D); Adaptive DeepFried (ADF), Adaptive Deepström with linear (ADSL), RBF (ADSR), Chi2 (ADSC) kernels, on small training sets with 5 and 20 training samples per class. Variance of results, computed on 30 runs, are given in brackets.

One may see first that *Deepstrom* architectures 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, both ADSR and ADSC significantly outperfom Adaptive DeepFried for any dataset and perform on par or significantly better than Dense architectures on the hardest CIFAR100 dataset. Moreover one sees that no single kernel based *Deepstrom* architecture dominate on all settings, showing the potential interest of combining multiple kernels as following experiments will show.

4.4 Multiple kernel learning

We report here results gained using multiple kernels in two different ways.

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

Table 3: *Multiple Deepström* experiments on CIFAR100 obtained on top of VGG19 convolutions.

First we exploited the *Multiple Kernels* strategy that we described in section 3 for exploiting multiple kernels in the output of the convolutional blocks, $conv$. Figure 3 reports results gained when using a combination of RBF kernels with various bandwidths and for different subsample sizes. Our multiple kernel strategy, exploiting kernels defined with various values of the hyperparameter allows automatically handling this hyper-parameter which usually requires to be tuned either through cross validation or to be manually chosen. The plots show the accuracy on the CIFAR10 dataset as a function of σ value, where the performance of the multiple kernel *Deepström* 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 strategy allows adapting the kernel combination optimally from the data without requiring any prior choice on the RBF bandwith hyper-parameter.

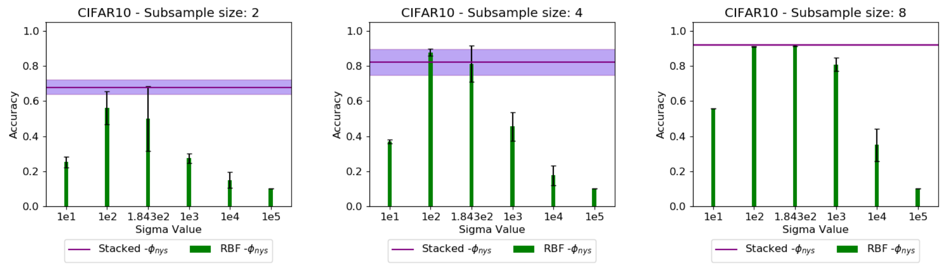

Figure 3: Comparison of *Multiple Kernels* that combines RBF kernels with various values of the bandwidth hyper-parameter. Multiple kernels performance is shown as an horizontal line while single kernel using one specific value of the bandwidth hyper-parameter σ. Plots correspond to a subsample size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

Second, we investigated another architecture that exploits *Multiple Deepström* approximations as presented in section 3. Here we use in parallel multiple Nyström approximations where kernels are dedicated to deal each with the output of a single feature map of the $conv$ part. Table 3 reports results on CIFAR100. We show the best performances obtained for each method by grid-searching on various hyper-parameters depending on the models, 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. *Deepström* is also the adaptive variant where the subsample size is in 16, 64, 128, 256, 512. We observe that both *Deepström* models outperform the considered baselines, demonstrating the interest in combining Deepström approximations.

5 Conclusion

We proposed *Deepström*, a new hybrid architecture that mixes deep networks and kernel methods. It is based on the Nyström approximation that allow considering any kind of kernel function in contrast to *Deep Fried Convnets*. Our proposal allows reaching state of the art results 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 *Deepström* architectures.

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

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

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

[7] 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.

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

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

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

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

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

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

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

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

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

[17] 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.

[18] 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.

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

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

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

[22] 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.

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

[24] 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.

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

APPENDIX

Figure 4 plots the 2-dimensional ϕ_{nys} representations of some 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 designing *Deepström Convnets* on lower level features output by lower level convolution blocks may yield state-of-the-art performance as well while requiring larger subsamples.

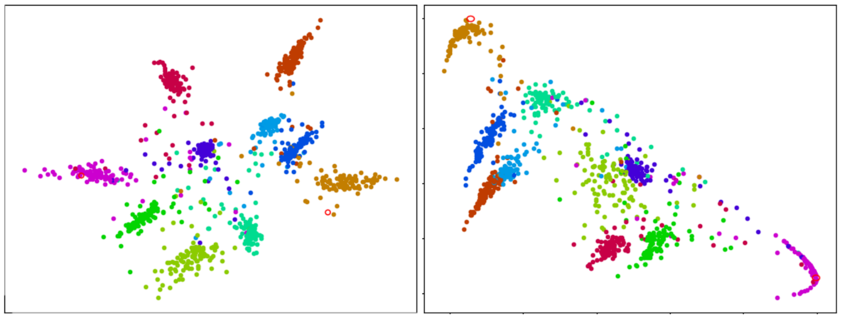

Figure 4: 2-dimensional ϕ_{nys} representation of 1000 randomly selected test set samples, obtained with a subsample set of size 2 and a linear kernel (left) or Chi2 kernel (right).