DEEP NETWORKS WITH ADAPTIVE NYSTRÖM APPROXIMATION (IJCNN 2019)

Luc Giffon, Stéphane Ayache, Thierry Artières, Hachem Kadri

Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France

QARMA

éQuipe d'AppRentissage de MArseille / Marseille Machine Learning Team

Aix*Marseille université

Initiative d'excellence

INTRODUCTION

Aix*Marseille université
Initiative d'excellence

Take Home Message

We use an adaptive variant of the Nyström method for kernel approximation as a drop-in replacement for dense layers in Convolutional Neural Networks (CNN)

Usual CNN architecture with dense layer(s) :

Our proposed architecture with Nyström layer(s) :

Outline

- **Non-linear mappings :**
  - *Dense (Fully-connected) neural network layers*
  - *Kernel methods*

- **Adaptive Nyström Layer**
  - *Standard formulation*
  - *Multiple Kernel Learning (MKL) formulation*

- ***Experiments***
  - *Standard setting*
  - *Small sample set*
  - *Multiple Kernel Learning*

NON-LINEAR MAPPINGS

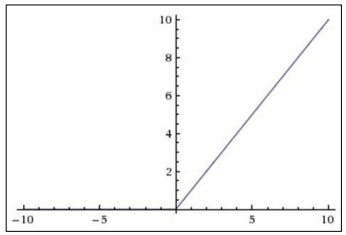
Dense (Fully-connected) layers

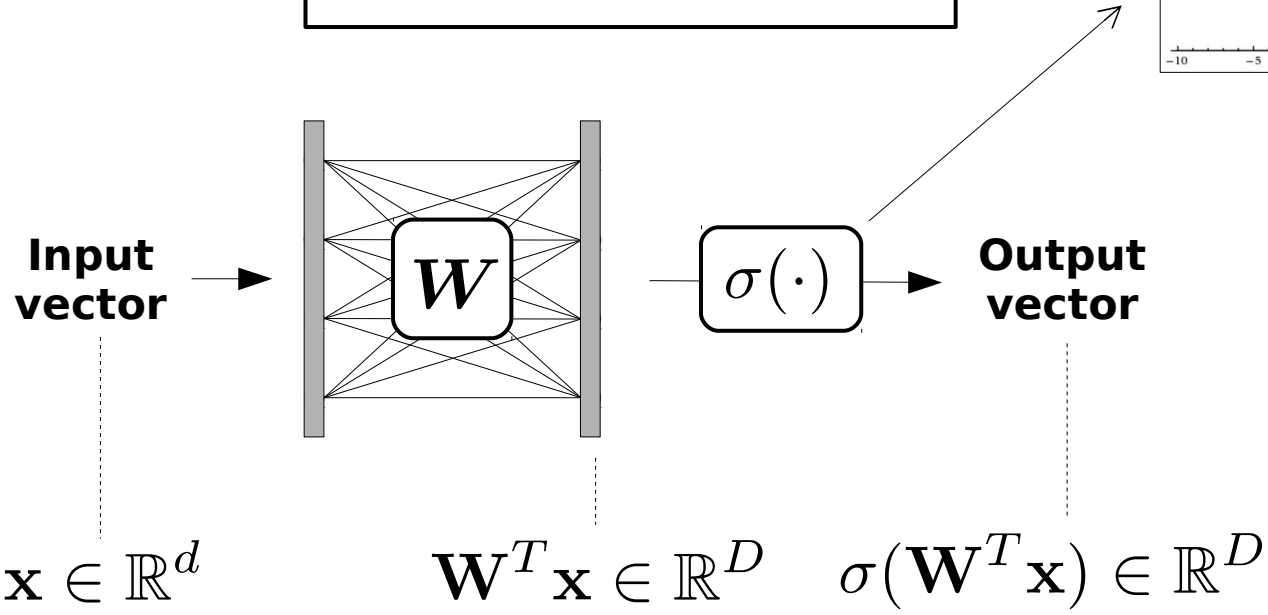
Each dense layer of a neural network learns a non-linear mapping of its input

Dense layer:

$$\phi_{dense}(\mathbf{x}) = \sigma(\mathbf{W}^T\mathbf{x})$$

Example : ReLU





Kernel methods

$$\forall i \ \mathbf{x}_i \in \mathbf{X} \qquad k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$$

Some kernel methods can give the feature map approximation $\tilde{\phi}$ for a kernel.

$$k(\mathbf{x}, \mathbf{z}) \approx \langle \tilde{\phi}(\mathbf{x}) \cdot \tilde{\phi}(\mathbf{z}) \rangle$$

(to keep the notations light, we drop the ~ on $\tilde{\phi}$ in the rest of the presentation)

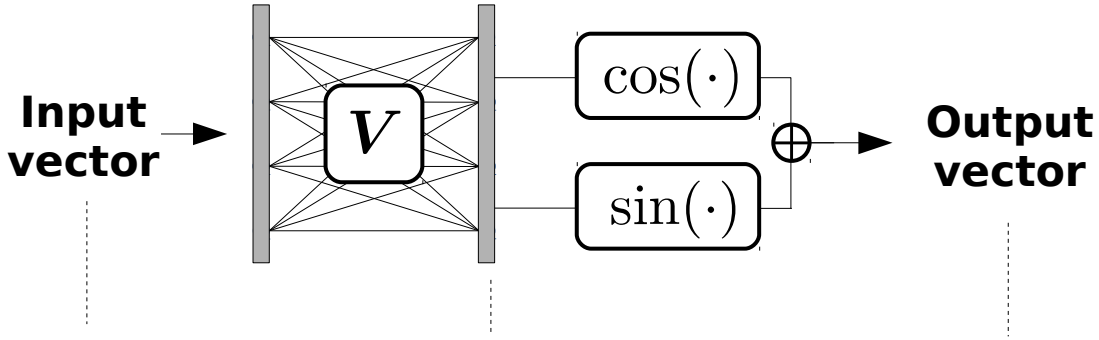
Kernel methods : Random features

Fast-Food (FF) :

FF is a fast approximation of the Gaussian kernel. It is a variant of the RKS method, a general approximation method for RBF kernels.

$$\phi_{FF}(\mathbf{x}) = [\cos(\mathbf{Vx}), \sin(\mathbf{Vx})] \qquad \mathbf{V} = \frac{1}{\sigma\sqrt{d}}\mathbf{SHG\Pi HB}$$

- **S**, **G** and **B** diagonal random
- $\mathbf{\Pi}$ random permutation
- **H** Hadamard matrix



(This mapping can be done multiple times In parallel then concatenated)

Kernel methods : Nyström method

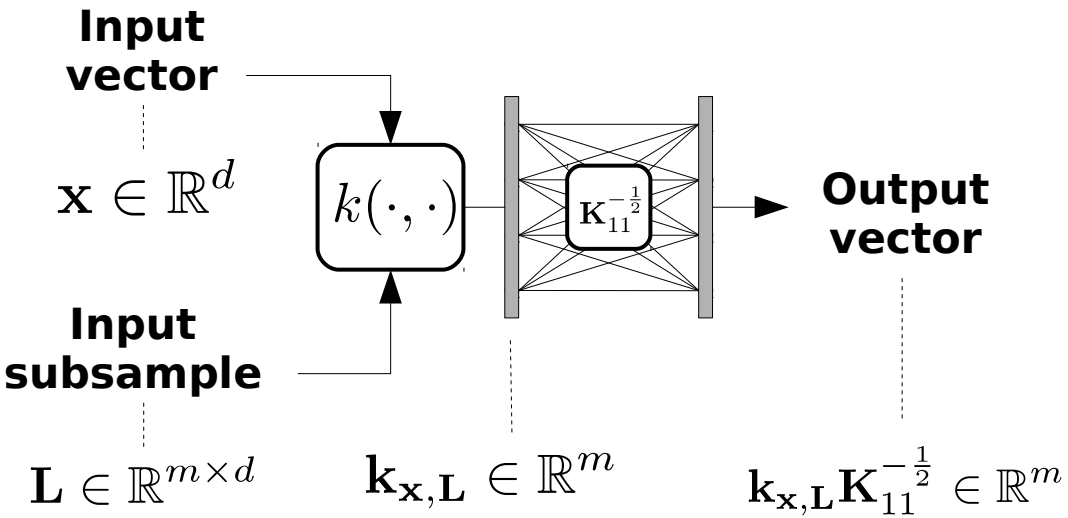
Nyström method for kernel approximation :

The Nyström method gives a low rank approximation of a Kernel matrix. For this approximation, we can extract the feature map approximation of the kernel.

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{21}^T \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \qquad \forall i,j \in 1...m \; \mathbf{K}_{11\,i,j} = k(\mathbf{L}_i, \mathbf{L}_j);\; \mathbf{L} \subset \mathbf{X};$$

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

$$\mathbf{k}_{\mathbf{x},\mathbf{L}} = [k(\mathbf{x}, \mathbf{L}_i,),\ ...,\ k(\mathbf{x}, \mathbf{L}_m)]$$

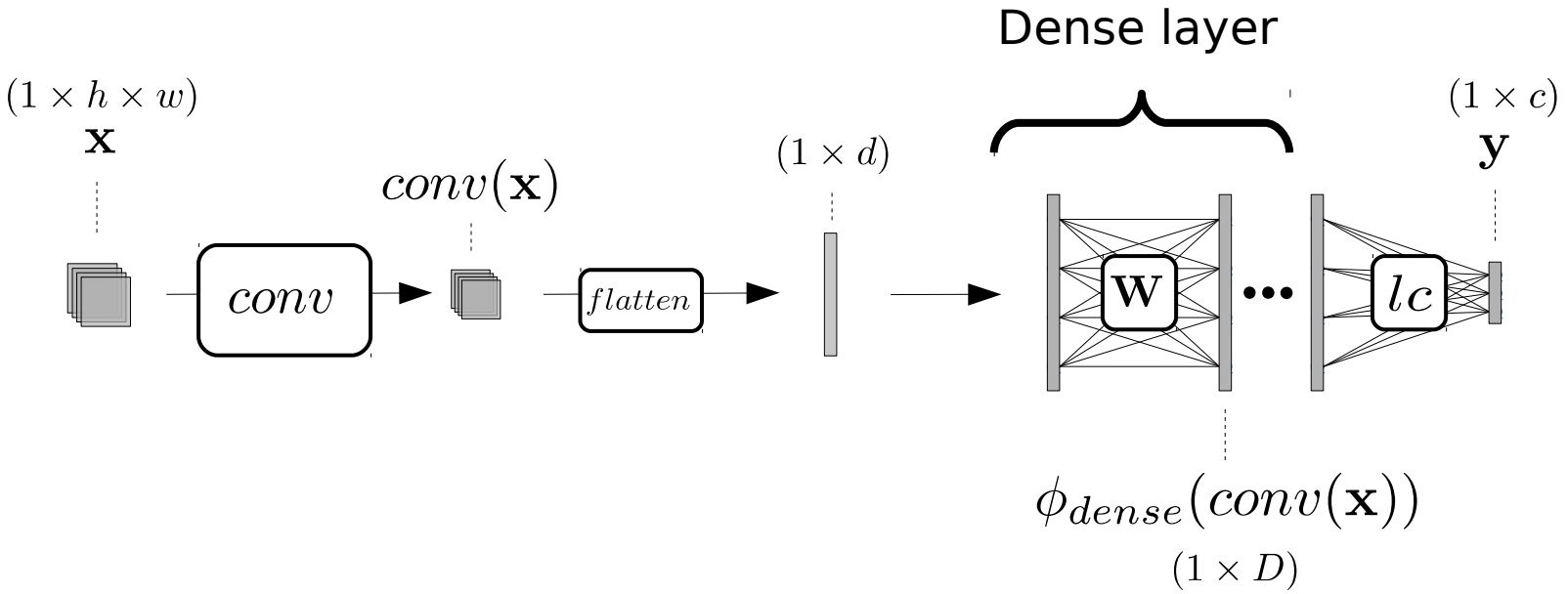


ADAPTIVE NYSTRÖM LAYER

Aix*Marseille université
Initiative d'excellence

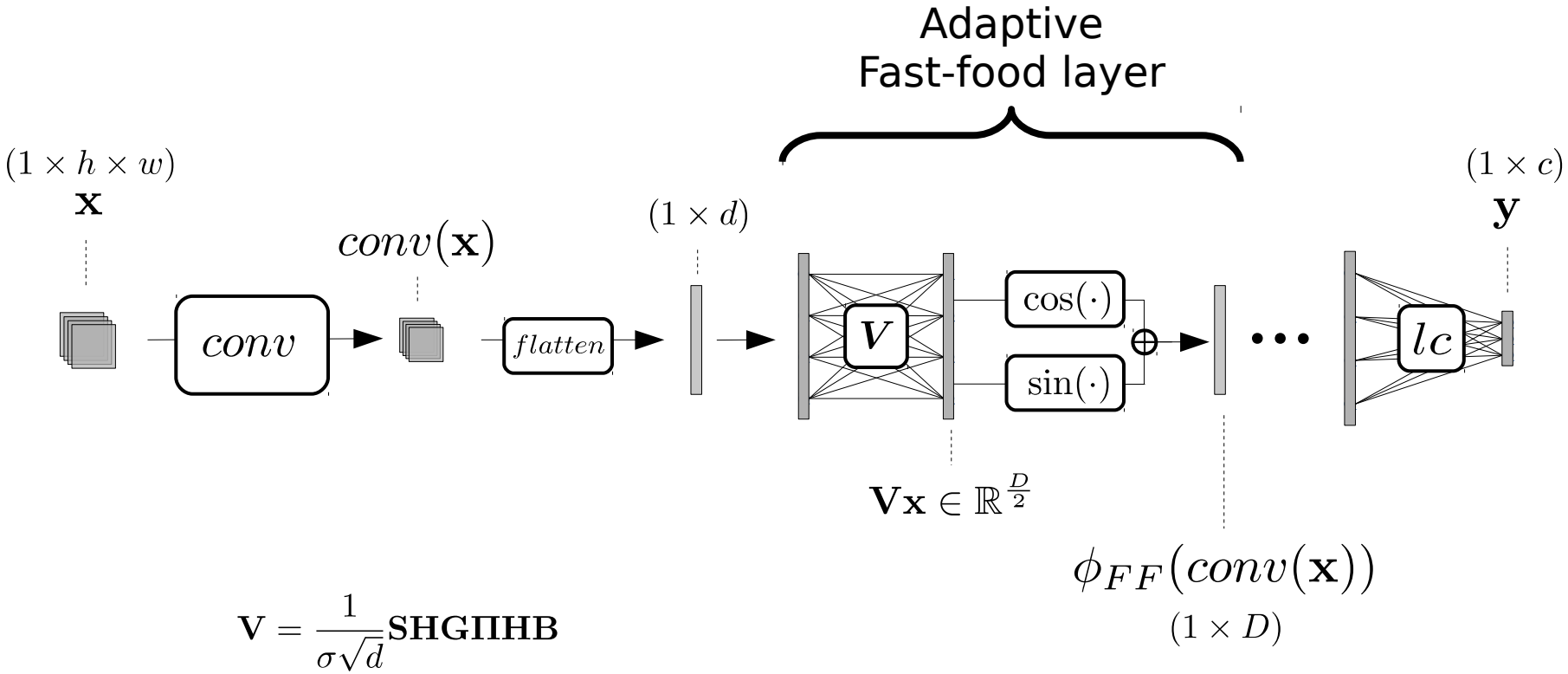
Standard formulation

CNN with Dense layer :



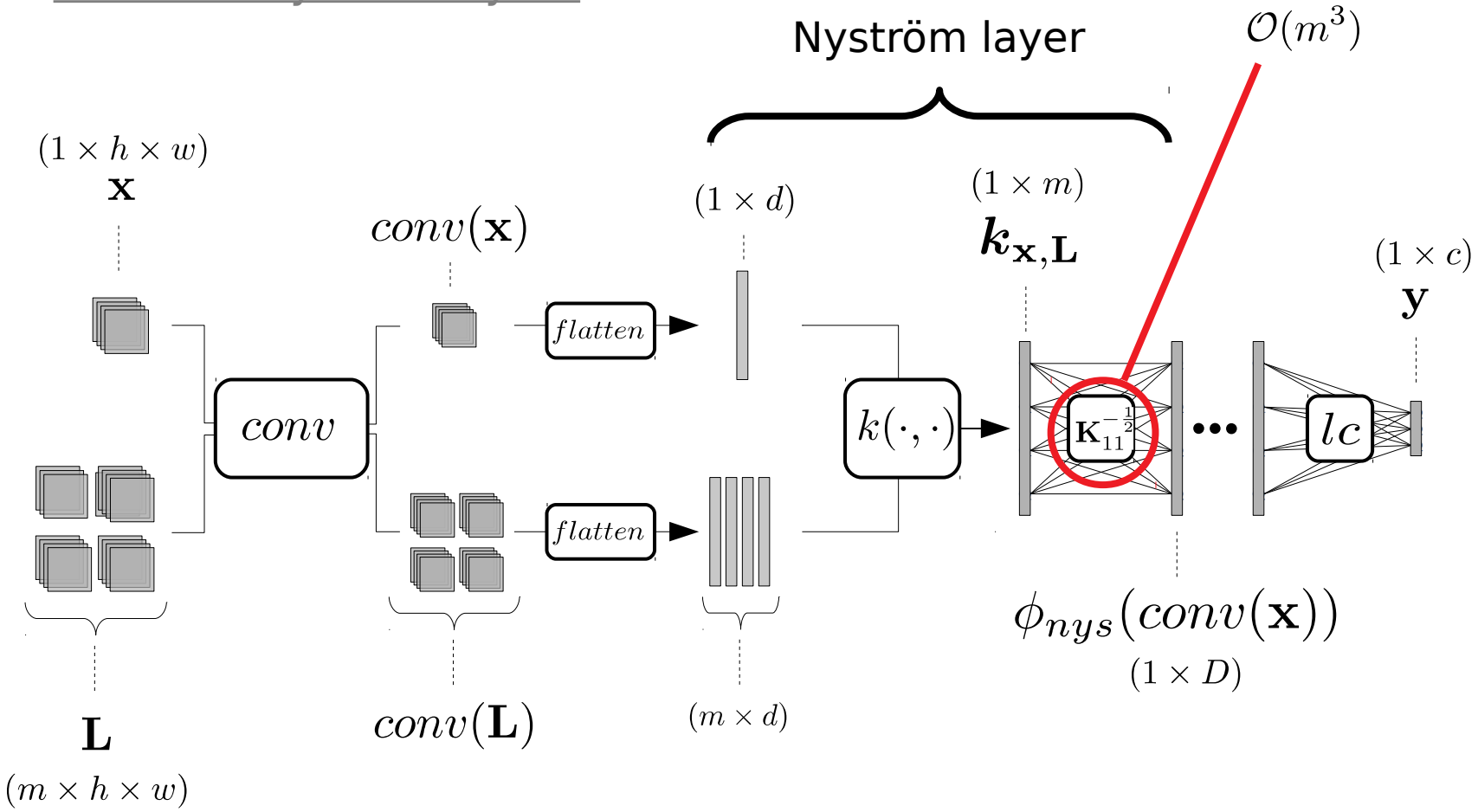
Standard formulation

CNN with Fast-food layer : *(Deep Fried Convnets – Yang et al. 2014)*



Standard formulation

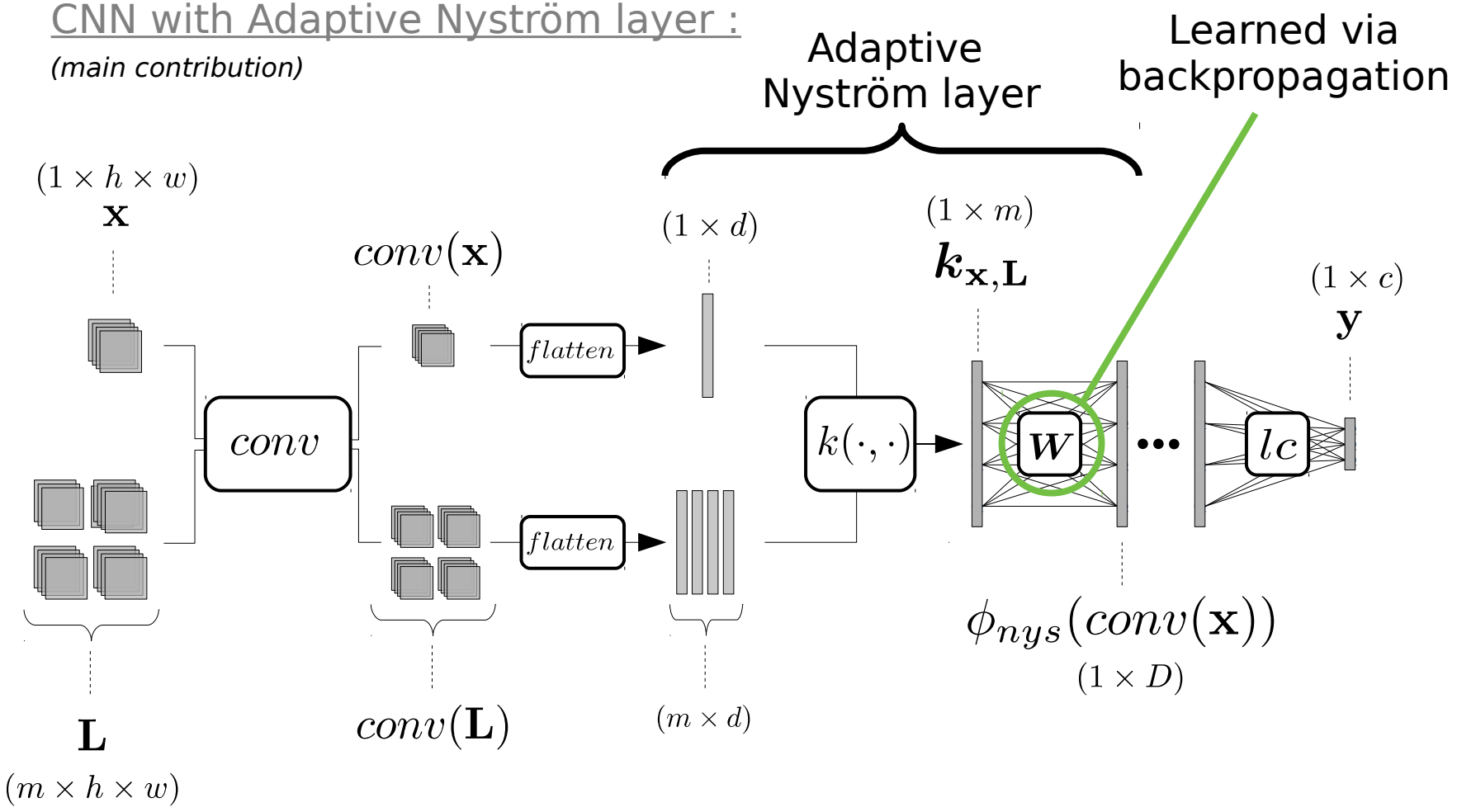
CNN with Nyström layer :



Standard formulation

CNN with Adaptive Nyström layer :

(main contribution)



Multiple Kernel Learning (MKL) formulation

In the Multiple Kernel setting, we can learn several adaptive Nyström layers in parallel then merge them by concatenation or weighted sum.

Concatenation :

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

Weighted sum :

$$\phi_{nys_{mkl}} = \sum_{i=1}^{l} \alpha_i \mathbf{W}_i \boldsymbol{k}^i_{\mathbf{x},\mathbf{L}}$$

Kernel function examples :

- **Chi2 kernel**

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\mathbf{x}_i + \mathbf{x}_j}$$

- **Linear kernel**

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

- **Gaussian kernels with different sigmas**

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma^2})$$

- **Kernel on single feature map**

$$k^z(\mathbf{x}_i, \mathbf{x}_j) = k(conv(\mathbf{x}_i)_z, conv(\mathbf{x}_j)_z)$$

EXPERIMENTS

Aix*Marseille université
Initiative d'excellence

Standard setting

- **Datasets :**

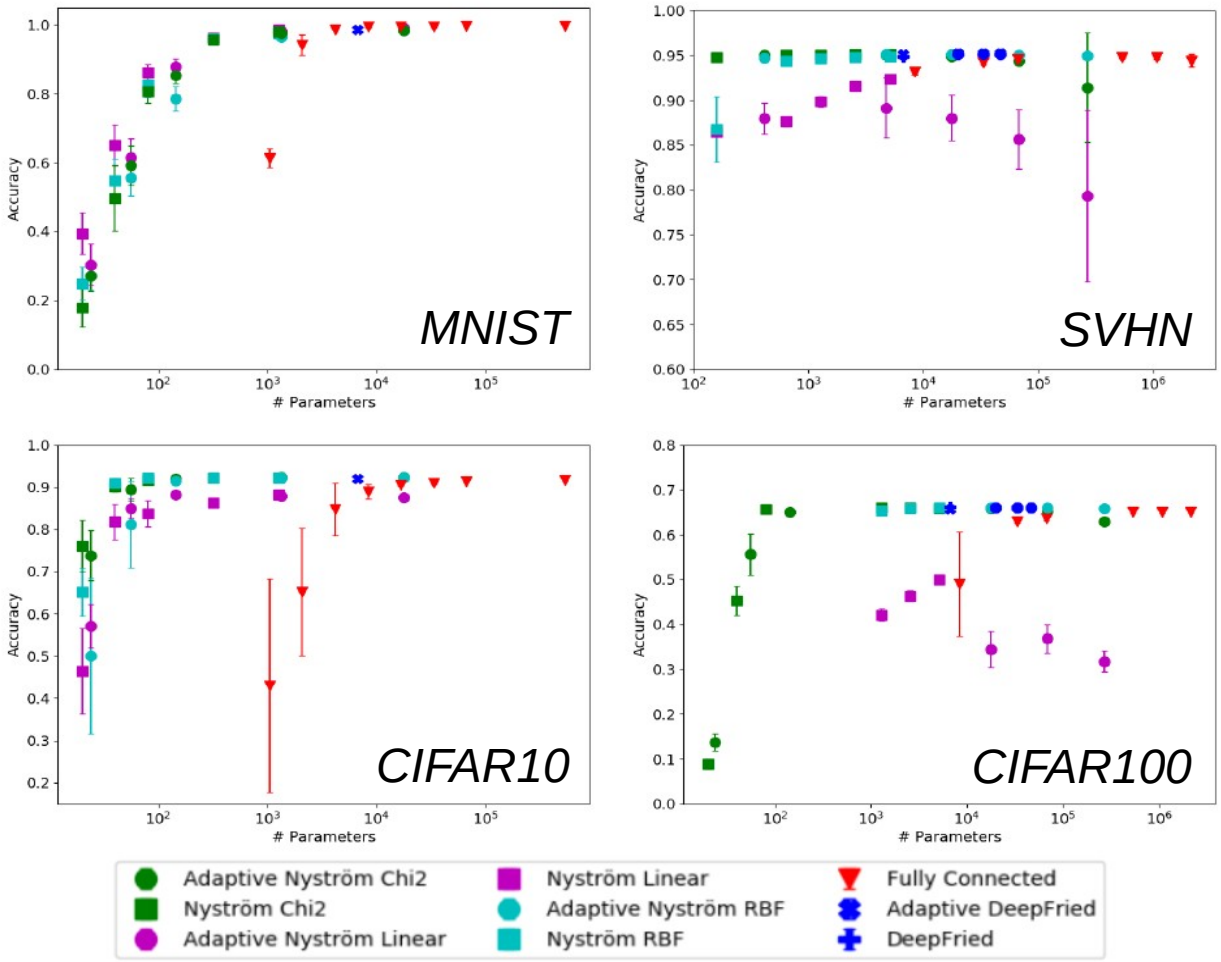
Dataset	Input shape	# classes	Training set size	Validation set size	Test set size
MNIST	$(28 \times 28 \times 1)$	10	40 000	10 000	10 000
SVHN	$(32 \times 32 \times 3)$	10	63 257	10 000	26 032
CIFAR10	$(32 \times 32 \times 3)$	10	50 000	10 000	10 000
CIFAR100	$(32 \times 32 \times 3)$	100	50 000	10 000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

- **Convolutional Architectures :**
  - *LeNet for MNIST ;*
  - *VGG19 for SVHN, CIFAR10 and CIFAR100.*

- **Number of learnable parameters :**
  - *Range from 2 to 1024 hidden neurons in the dense hidden layer ;*
  - *Range from 2 to 128 subsamble size for the Nystrom layer ;*
  - *One stack of random features for the Fast-food layer.*

Standard setting

Our Nyström layer reaches dense layers accuracy with much less parameters. The adaptive variant performs better in most scenario.

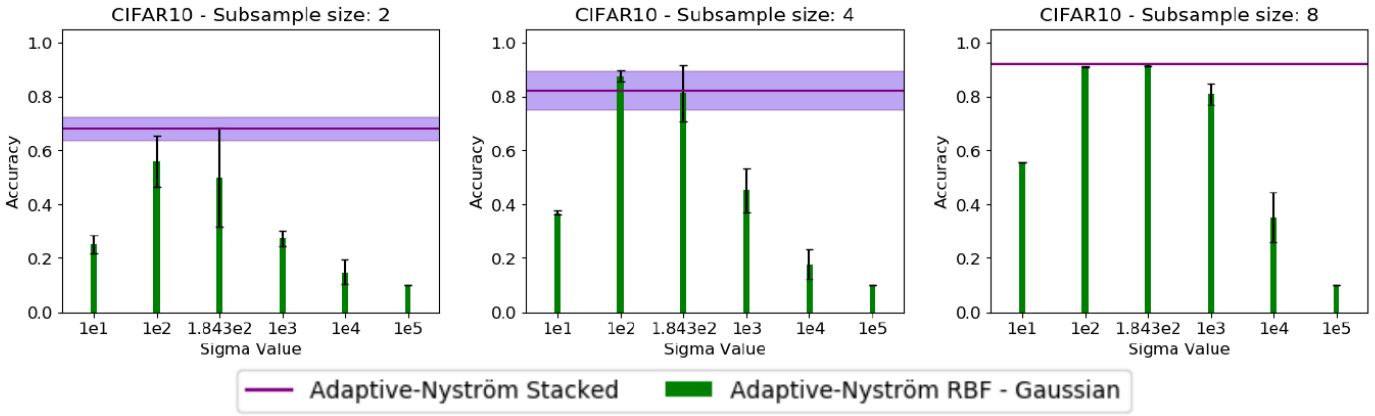


Small sample set

	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)

Multiple Kernel Learning

Multiple Gaussian kernel :

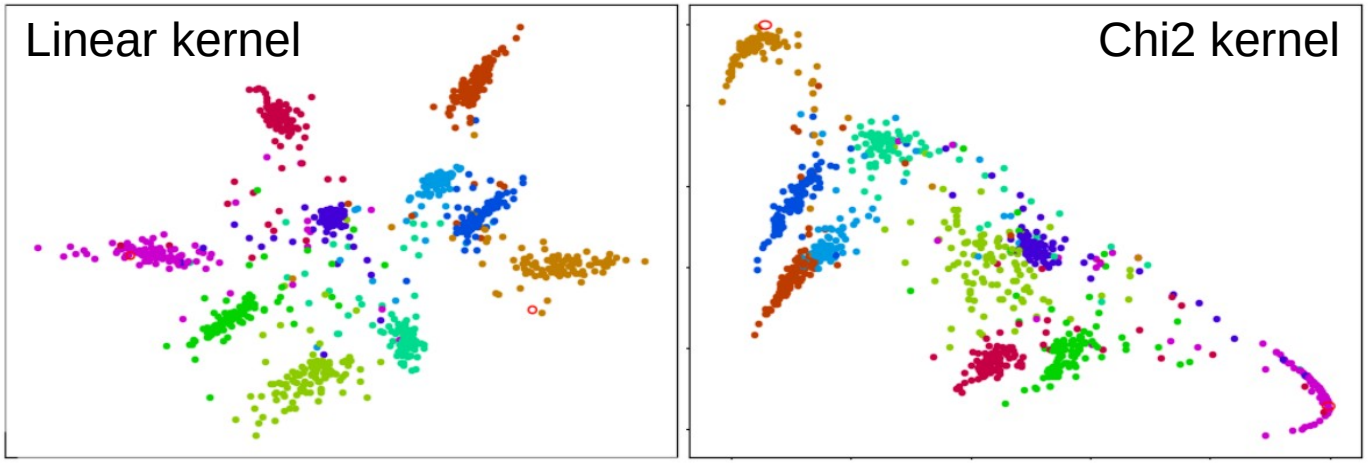


Multiple Kernel Learning

Kernel on single feature maps : CIFAR100

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

2D representations : CIFAR10



CONCLUSION

Aix*Marseille université
Initiative d'excellence

Conclusions

- **Learns fewer parameters than standard Dense layers while not reducing the performance ;**
- **Total flexibility in the choice of the kernel function in contrast with the Deep Fried Convnets ;**
- **Modular and able to deal with Multiple Kernel Learning paradigm ;**
- **Simple to implement with Keras or Tensorflow**

THANK YOU FOR YOUR ATTENTION

Aix*Marseille université
Initiative d'excellence

Kernel methods : Nyström method

Nyström method for kernel approximation :

The Nyström method gives a low rank approximation of a Kernel matrix. From this approximation, we can extract the feature map approximation of the kernel.

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{21}^T \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \qquad \forall i,j \in 1...m \ \mathbf{K}_{11\,i,j} = k(\mathbf{L}_i, \mathbf{L}_j);\ \mathbf{L} \subset \mathbf{X};$$

The Nyström method gives :

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

Then the Nyström feature map is :

$$\tilde{\mathbf{\Phi}} = \begin{bmatrix} \mathbf{K}_{11} \\ \mathbf{K}_{21} \end{bmatrix} \mathbf{K}_{11}^{-\frac{1}{2}} \quad \Rightarrow \quad \forall i\ \tilde{\mathbf{\Phi}}_i = \phi_{nys}(\mathbf{x}_i) = \mathbf{k}_{\mathbf{x}_i,\mathbf{L}} \mathbf{K}_{11}^{-\frac{1}{2}}$$

$$\mathbf{k}_{\mathbf{x},\mathbf{L}} = [k(\mathbf{x}, \mathbf{L}_i,),\ ...,\ k(\mathbf{x}, \mathbf{L}_m)]$$

Kernel methods : Random features

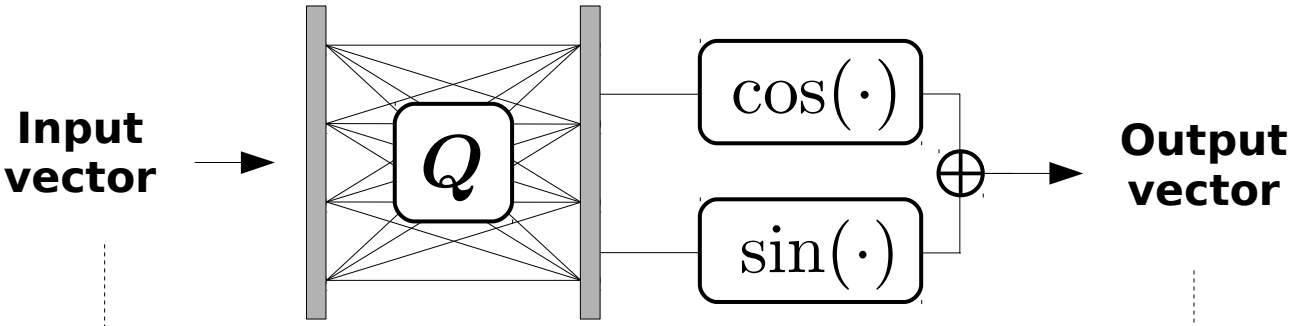
Random Kitchen Sinks (RKS) :

The RKS approximates a *Radial Basis Function* (RBF) kernel.

$$\phi_{RKS}(\mathbf{x}) = [\cos(\mathbf{Q}\mathbf{x}), \sin(\mathbf{Q}\mathbf{x})]$$

$$\mathbf{Q}_{i,j} \sim \mathcal{N}(\mu, \sigma)$$

(For the Gaussian kernel)



$\mathbf{x} \in \mathbb{R}^d$ $\quad \mathbf{Q}\mathbf{x} \in \mathbb{R}^{\frac{D}{2}}$ $\quad [\cos(\mathbf{Q}\mathbf{x}), \sin(\mathbf{Q}\mathbf{x})] \in \mathbb{R}^D$

Kernel methods

$$\forall i \ \mathbf{x}_i \in \mathbf{X} \qquad k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$$

Some kernel methods can give the feature map approximation $\tilde{\phi}$ for a kernel.

$$k(\mathbf{x}, \mathbf{z}) \approx \langle \tilde{\phi}(\mathbf{x}) \cdot \tilde{\phi}(\mathbf{z}) \rangle$$

$$\mathbf{K} \approx \tilde{\mathbf{K}} = \tilde{\mathbf{\Phi}} \tilde{\mathbf{\Phi}}^T \qquad \forall i \ \tilde{\mathbf{\Phi}}_i = \tilde{\phi}(\mathbf{x}_i)$$

(to keep the notations light, we drop the ~ on $\tilde{\phi}$ in the rest of the presentation)