Simplifying ConvNets for Fast Learning

Franck Mamalet[1] and Christophe Garcia[2]

[1] Orange Labs, 4 rue du Clos Courtel, 35512 Cesson-Sévigné, France,
`franck.mamalet@orange.com`
[2] LIRIS, CNRS, Insa de Lyon, 17 avenue Jean Capelle, 69621 Villeurbanne, France,
`christophe.garcia@liris.cnrs.fr`

Abstract. In this paper, we propose different strategies for simplifying filters, used as feature extractors, to be learnt in convolutional neural networks (*ConvNets*) in order to modify the hypothesis space, and to speed-up learning and processing times. We study two kinds of filters that are known to be computationally efficient in feed-forward processing: fused convolution/sub-sampling filters, and separable filters. We compare the complexity of the back-propagation algorithm on ConvNets based on these different kinds of filters. We show that using these filters allows to reach the same level of recognition performance as classical *ConvNets* for handwritten digit recognition, up to five times faster.

1 Introduction

Convolutional Neural Networks (*ConvNets*), proposed by LeCun *et al.* [1], have shown great performances in various computer vision applications, such as handwritten character recognition [1, 2], facial analysis [3–5], videoOCR [6, 7], or vision-based navigation [8]. *ConvNets* consist of a pipeline of convolution and pooling operations followed by a multi-layer perceptron. They tightly couples local feature extraction, global model construction and classification in a single architecture where all parameters are learnt conjointly using back-propagation.

Constructing efficient *ConvNets* to solve a given problem requires exploring several network architectures by choosing the number of layers, the number of features per layer, convolution and sub-sampling sizes, and connections between layers, which impact directly training time.

Several approaches have been proposed to improve learning speed and generalization by reducing or modifying the hypothesis space, *i.e.* the network architecture. Pruning neural networks has been broadly studied for MLPs, and a survey is given in [9]. Concerning *ConvNets*, Jarrett *et al.* [10] have proposed to add sophisticated non-linearity layers such as rectified sigmoid or local contrast normalization to improve network convergence. Mrazova *et al.* [11] proposed to replace convolution layers by *Radial Basic Functions -RBF-* ones leading to faster learning when associated with a "winner-takes-all" strategy.

Optimization methods for efficient implementation on processors or hardware systems have also been proposed to accelerate learning. Some studies were carried out on fractionnal transformation of back-propagation [12], others on

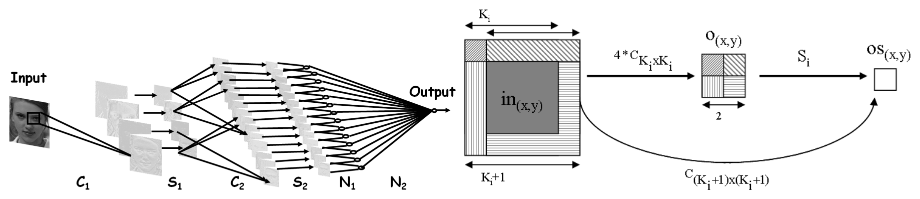

Fig. 1. (a) a typical ConvNet architecture with two feature extraction stages; (b) Fusion of convolution and sub-sampling layers.

parallelization schemes (comparisons are given in [13]). Recent works have been targeting graphic processing units -GPU- to speed-up back-propagation [2, 14].

In this paper, we will focus on *ConvNet* hypothesis space modifications, using simplified convolutional filters to accelerate epoch processing time. In section 2, we describe the reference *ConvNet* architecture, detail the proposed equivalent convolutional filters, and compare their back-propagation complexity. Section 3 presents in-depth experiments on handwritten digit recognition using different kinds of convolutional filters, and compares both generalization performances and training time. Finally, conclusions and perspectives are drawn in section 4.

2 Simplifying convolutional filters

In this section, we first describe the classical *ConvNet LeNet-5* [1], proposed by LeCun *et al.*. Then, we propose several equivalent networks architectures using simplified convolutional filters, and compare the complexity of the back-propagation algorithm on these layers.

The original model of *ConvNet*, illustrated in *Figure* 1.(a), is based on convolutional filters layers interspersed with non-linear activation functions, followed by spatial feature pooling operations such as sub-sampling. Convolutional layers C_i contain a given number of planes. Each unit in a plane receives input from a small neighborhood (local receptive field) in the planes of the previous layer. Each plane can be considered as a feature map that has a fixed feature detector, that corresponds to a convolution with a trainable mask of size $K_i \times K_i$, applied over the planes in the previous layer. A trainable bias is added to the results of each convolutional mask, and a hyperbolic tangent function, used as an activation function, is applied. Multiple planes are used in each layer so that multiple features can be detected. Once a feature has been detected, its exact location is less important. Hence, each convolutional layer C_i is typically followed by a pooling layer S_i that computes the average (or maximum) values over a neighborhood in each feature map, multiplies it by a trainable coefficient, adds a trainable bias, and passes the result through an activation function.

Garcia *et al.* [3, 5, 7] have shown that, for different object recognition tasks, state-of-the-art solutions can be achieved without non-linear activation functions in convolutional layers. Thus, in the rest of this paper, we will only consider C_i

layers with identity activation function. We will also consider average pooling layers S_i performing a sub-sampling by two. For a C_i layer, its input map size $W_{in} \times H_{in}$, its output map size $W_i \times H_i$, and the following S_i sub-sampled output map size $SW_i \times SH_i$ are connected to the convolution kernel size K_i by: $(W_i, H_i) = (W_{in} - K_i + 1, H_{in} - K_i + 1)$ and $(SW_i, SH_i) = (W_i/2, H_i/2)$.

Since these layers rely on *local receptive fields*, the complexity of the back-propagation delta-rule algorithm for a given element is proportional to its output map size and the cardinal of its connections with the following layer, that is, proportional to $(W_i \times H_i)$ for C_i layers and $(SW_i \times SH_i \times K_{i+1}^2)$ for S_i layers.

Weight sharing in these layers implies a complexity of the weight update algorithm that is proportional to output map and kernel sizes: *i.e.* $(W_i \times H_i \times K_i^2)$ for C_i layers, and in $(SW_i \times SH_i)$ for S_i layers.

In the remainder of this section, we present our proposition to learn modified *ConvNets* where C_i and S_i layers are replaced by equivalent convolutional filters, and compare the back-propagation complexity of these layers.

2.1 Fused convolution and sub-sampling filters

It has been shown by Mamalet *et al.* [15, 16], that a convolutional layer C_i with linear activation followed by a sub-sampling layer S_i can be replaced in the feed-forward pass (when the learning phase is achieved) by an equivalent fused convolutional/sub-sampling layer CS_i which consists of single convolutions of $(K_i + 1) \times (K_i + 1)$ kernel size applied with horizontal and vertical input steps of two pixels, followed by a non-linear activation function (this two pixels step serves as sub-sampling, see *Figure* 1.(a)), leading to speed-up factors up to 2.5 [15, 16]. Kernel weights $\tilde{w}$ and bias $\tilde{b}$ are obtained respectively by linear combination of original weights w and bias b.

In this paper, we propose to learn directly these fused convolution/sub-sampling layers, *i.e.* convolution maps of kernel size $(K_i + 1) \times (K_i + 1)$ with an input step of two pixels. One can notice that the hypothesis space represented by fused convolution/sub-sampling layers CS_i is larger than the one represented by the pair (C_i, S_i).

The output map size of a layer CS_i is $SW_i \times SH_i$ and is connected to a CS_{i+1} convolution with a step of two pixels. The complexity of the back-propagation algorithm for such a CS_i layer is proportional to $(SW_i SH_i (K_{i+1} + 1)^2/4)$. The update weight algorithm complexity is is proportional to $(SW_i SH_i (K_i + 1)^2)$.

2.2 Separable convolution filters

Another special case of convolutional filters are separable ones, *i.e.* convolutions that can be expressed as the outer product of two vectors: $C_i = Ch_i * Cv_i = Cv_i * Ch_i$, where Ch_i (resp. Cv_i) is a row (resp. column) vector of size K_i.

Figure 2.(a) shows a separable C_i feature map split into two successive 1D-convolutions. In feed-forward computation applied over a $W_{in} \times H_{in}$ input image, this transformation leads to a $K_i^2/(2K_i)$ speedup factor. If separable filters are

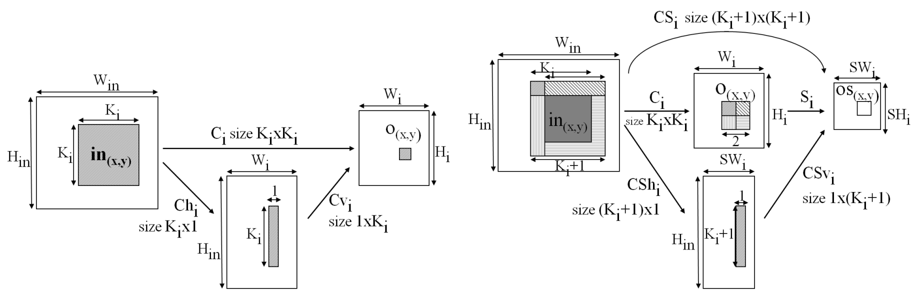

Fig. 2. (a) Separable convolution layers; (b) Fused separable convolution and sub-sampling layers.

broadly used in image processing, as far as we know, no study has been published on learning separable filters within *ConvNet* architectures.

We thus propose to restrict the hypothesis space using only separable convolutions in *ConvNets*, and directly learn two successive 1D-filters. If in the feed-forward application, horizontal and vertical filters are commutative, back-propagation in *ConvNets* may lead to different trained weights. Thus, we will evaluate either a horizontal convolution Ch_i whose output map size is $W_i \times H_{in}$, followed by a vertical one Cv_i (*Figure* 2.(a)), or a vertical convolution Cv_i whose output map size is $W_{in} \times H_i$, followed by a horizontal one Ch_i. No activation function is used in Ch_i and Cv_i layers.

We denote the first (resp. second) configuration $Ch_i * Cv_i$ (resp. $Cv_i * Ch_i$). The delta-rule complexity of the $Ch_i * Cv_i$ configuration is proportional to $(W_i H_{in} K_i + W_i H_i)$, since the Ch_i layer is connected to the Cv_i layer, which is itself connected to the S_i layer. The weight update algorithm is proportional to $(W_i(H_{in} + H_i)K_i)$. The complexity of the $Cv_i * Ch_i$ configuration is obtained by replacing H and W.

The hypothesis space represented by these separable convolutional filters is a more restricted set than the one of classical *ConvNets*.

2.3 Fused separable convolution and sub-sampling filters

Our third proposition is to combine the two previous kinds of filters to learn fused separable convolution and sub-sampling layers, which consist in either a horizontal convolution CSh_i with a horizontal step of two, whose output map size is $SW_i \times H_{in}$, followed by a vertical one CSv_i with a vertical step of two and an activation function, or a vertical convolution CSv_i with a vertical step of two, whose output map size is $W_{in} \times SH_i$, followed by a horizontal one CSh_i with a horizontal step of two and an activation function.

We denote the first configuration $CSh_i * CSv_i$ and the second $CSv_i * CSh_i$. The $CSh_i * CSv_i$ configuration is described in *Figure* 2.(b), underlining its equivalence with a traditional (C_i, S_i) couple or a CS_i layer.

Table 1. Complexity comparison of back-propagation algorithm for different filters

	(W_i, H_i, K_i, K_{i+1})	$(40, 50, 7, 5)$	$(28, 28, 5, 5)$
C_i	$W_i H_i K_i^2$	98000	19600
S_i	$W_i H_i K_{i+1}^2/4$	12500	4900
CS_i	$(4(K_i+1)^2 + (K_{i+1}+1)^2)\frac{W_i H_i}{16}$	36500	8820
Speedup factor $(C_i, S_i)/CS_i$	$\frac{16K_i^2+4K_{i+1}^2}{4(K_i+1)^2+(K_{i+1}+1)^2}$	**3.0**	**2.8**
$Ch_i * Cv_i$	$K_i W_i (2H_{in} + H_i)$	45360	12880
Speedup factor $C_i/(Ch_i * Cv_i)$	$\frac{K_i H_i}{2H_{in}+H_i}$	**2.2**	**1.5**
$Cv_i * Ch_i$	$K_i H_i (2W_{in} + W_i)$	46200	12880
Speedup factor $C_i/(Cv_i * Ch_i)$	$\frac{K_i W_i}{2W_{in}+W_i}$	**2.1**	**1.5**
$CSh_i * CSv_i$	$(K_i+1)(3H_{in} + H_i)W_i/4$	17440	5208
Speedup factor $(C_i, S_i)/(CSh_i * CSv_i)$	$\frac{H_i(4K_i^2+K_{i+1}^2)}{(K_i+1)(3H_{in}+H_i)}$	**6.3**	**4.7**
$CSv_i * CSh_i$	$(K_i+1)(3W_{in} + W_i)H_i/4$	17800	5208
Speedup factor $(C_i, S_i)/(CSv_i * CSh_i)$	$\frac{W_i(4K_i^2+K_{i+1}^2)}{(K_i+1)(3W_{in}+W_i)}$	**6.2**	**4.7**

The $CSh_i * CSv_i$ delta-rule complexity is proportional to $(SW_i H_{in}(K_i + 1)/2 + SW_i SH_i(K_{i+1} + 1)/2)$ and its weight update complexity is proportional to $(SW_i H_{in}(K_i + 1) + SW_i SH_i(K_i + 1))$. The complexity of the $CSv_i * CSh_i$ configuration is obtained by replacing H and W.

The hypothesis space represented by these fused separable convolution and sub-sampling filters is larger than the one represented by separable convolutional ones (section 2.2), but smaller than the ones presented in section 2.1.

2.4 Comparison of the Back-propagation complexity for these filters

Table 1 gathers the complexity of the learning phase for each filter type described in this section. It also gives speedup factors compared to traditional C_i, S_i *ConvNet* layers, for some parameter values.

We can see in *Table* 1 that the back-propagation complexity of CS_i layers is up to three times lower than traditional *ConvNet* (C_i, S_i) layers. Separable convolution $Ch_i * Cv_i$ or $Cv_i * Ch_i$ learning is only two times faster, and fused separable convolution and sub-sampling $CSh_i * CSv_i$ or $CSv_i * CSh_i$ can lead to a speedup factor of up to six.

In the next section, we present two experiments showing that using such modified convolutional layers leads to comparable classification and recognition performances, and enable epoch processing acceleration closed to those given in *Table* 1.

3 Experiments

The main goal of these experiments is not to propose novel convolutional architectures for the following tasks, but to compare learning capabilities with modified filters. We thus use a reference *ConvNet* architecture similar to the well-known *LeNet-5* proposed by LeCun *et al.* [1] for handwritten digit recognition. From now on, we denote networks with the corresponding filter notation, *i.e.* a CS_i network stands for a *ConvNet* with CS_i layers.

3.1 Handwritten digit recognition

This experiment is based on the MNIST database introduced by LeCun *et al.* [1] which comprises 60,000 training and 10,000 test 28×28 images. State-of-the-art methods achieves a recognition rate of 99.65% [14] using a deep MLP trained on GPUs and elastic distortions on training images.

In this paper, we use a reference *ConvNet* architecture inspired by *LeNet-5* [1], and do not apply any distortion to the training images. As in [1], *ConvNet* inputs are padded to 32×32 images and normalized so that the background level corresponds to a value of -0.1 and the foreground corresponds to 1.175. For each network, we launch six training on 25 epochs and save the network after the last epoch (no overlearning is observed as in [1]). Then, generalization is estimated on the test set, and we retain the best one.

The 32×32 input image is connected to six C_1 5×5 kernel size convolution maps, followed by six S_1 sub-sampling maps. C_2 layers consists in fifteen 5×5 kernel size convolution maps which take input from one of the possible pairs of different feature maps of S_1. These maps are connected to fifteen S_2 sub-sampling layer maps. The N_1 layer has 135 neurons: each of the fifteen S_2 feature maps is connected to two neurons, and each of the remaining 105 neurons takes input from one of the possible pairs of different feature maps of S_2. N_2 is a fully connected 50 neurons layer. The ten N_3 fully connected output neurons use a softmax activation function. This network comprises 14,408 trainable weights.

We train networks using modified convolutional filters:

- Fused convolution and sub-sampling network where $C_i + S_i$ layers have been replaced 6×6 kernel size CS_i filters (*Figure* 1.(b)). This network has only five layers and 14,762 trainable weights,
- Separable convolution networks have nine layers, replacing each C_i layer by two $Ch_i * Cv_i$ or $Cv_i * Ch_i$ ones (*Figure* 2.(a)). They have 13,814 trainable weights,
- Fused separable convolution and sub-sampling networks comprise seven layers, each (C_i, S_i) couple is replaced by $CSh_i * CSv_i$ or $CSv_i * CSh_i$ ones (*Figure* 2.(b)). They have 13,829 trainable weights.

Figure 3 shows features maps obtained on an '8' handwritten digit input with the learnt networks CS_i, $CSh_i * CSv_i$ and $Ch_i * Cv_i$. *Table* 2 presents the results obtained on MNIST training and test databases with different kind of convolutional filters. The first line gives the reference performance of *LeNet-5* architecture with the same training database (no distortion). Our reference *ConvNet* architecture (C_i, S_i) has a performance of 1.28% error rate on MNIST test database. This small rate gap with *LeNet-5* results is mainly due to architecture differences (layer connections and output units).

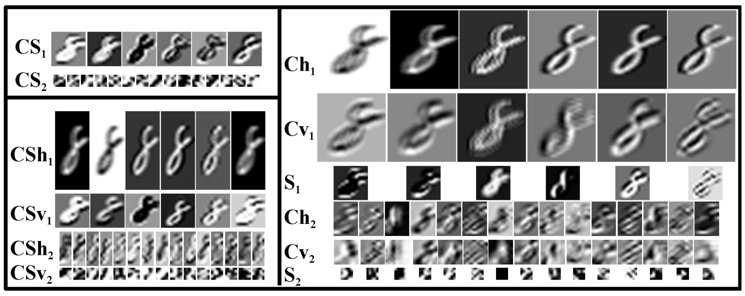

Fig. 3. Feature maps obtained with simplified convolutional filters (upper left: CS_i; bottom left: $CSh_i * CSv_i$; right: $Ch_i * Cv_i$).

Table 2. MNIST error rate (ER) for each kind of network

	Training ER (%)	Test ER (%)	Speedup Factor
LeNet-5 (no distortion) [1]	0.35	0.95	
(C_i, S_i)	0.46	**1.28**	1.0
CS_i	0.07	**1.32**	**2.6**
$Ch_i * Cv_i$	0.68	**1.52**	**1.6**
$Cv_i * Ch_i$	0.44	**1.45**	**1.6**
$CSh_i * CSv_i$	0.36	**1.49**	**3.3**
$CSv_i * CSh_i$	0.14	**1.61**	**2.9**

The CS_i network obtains the same generalization performances as traditional *ConvNet* and require 2.6 times less processing time per epoch, which is comparable to the estimation given in *Table* 1. Other configurations induce a loss of performance smaller than 0.4%, and enable speedup factor of 1.6 for separable filters, and up to 3.3 for fused separable ones. This latter is slightly lower than the estimation given in *Table* 1, due to the N_i back-propagation time which becomes predominant.

4 Summary and future work

In this paper, we have introduced several modifications of the hypothesis space of Convolutional Neural Networks (*ConvNet*), replacing convolution and sub-sampling layers by equivalent fused convolution/sub-sampling filters, separable convolution filters or fused separable convolution/sub-sampling filters. We have proposed a complexity estimation of the back-propagation algorithm on these different kinds of filters which allows evaluating learning speedup-factor. We have presented experiments on the handwritten digit database MNIST using reference *ConvNets* which performed comparably to similar systems in the literature. We have trained the modified *ConvNets* using the simplified filters, and proven that classification and recognition performances are almost the same with a training time divided by up to five. To enhance convergence and generalization, the proposed convolutional filters could be interspersed with other non-linear units, such as rectification or local normalization [10], or also to form part of wider networks enabling to speed-up architecture and space exploration. Furthermore, we plan to combine these filters optimizations with parallel implementations on GPU which are known to be efficient in 1D and 2D convolution processing, and we believe it would allow processing of larger deep-learning networks.

References

1. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. of the IEEE*, Nov 1998.
2. K. Chellapilla, S. Puri, and P. Simard, “High Performance Convolutional Neural Networks for Document Processing,” in *Proc. of the Int. Workshop on Frontiers in Handwriting Recognition (IWFHR’06)*, 2006.
3. C. Garcia and M. Delakis, “Convolutional Face Finder: a neural architecture for fast and robust face detection,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Nov. 2004.
4. M. Osadchy, Y. LeCun, M. L. Miller, and P. Perona, “Synergistic face detection and pose estimation with energy-based model,” in *Proc. of Advances in Neural Information Processing Systems (NIPS’05)*, 2005.
5. C. Garcia and S. Duffner, “Facial image processing with convolutional neural networks,” in *Proc. Int. Workshop on Advances in Pattern Recognition*, 2007.
6. M. Delakis and C. Garcia, “Text detection with Convolutional Neural Networks,” in *Proc. of the Int. Conf. on Computer Vision Theory and Applications*, 2008.
7. Z. Saidane and C. Garcia, “Automatic scene text recognition using a convolutional neural network,” in *Proc. of Int. Workshop on Camera-Based Document Analysis and Recognition*, 2007.
8. R. Hadsell, P. Sermanet, M. Scoffier, A. Erkan, K. Kavackuoglu, U. Muller, and Y. LeCun, “Learning long-range vision for autonomous off-road driving,” *Journal of Field Robotics*, Feb. 2009.
9. R. Reed, “Pruning algorithms - a survey,” *IEEE Trans. on Neural Networks*, 1993.
10. K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?,” in *Proc. Int. Conf. on Computer Vision (ICCV’09)*, 2009.
11. I. Mrazova and M. Kukacka, “Hybrid convolutional neural networks,” in *Proc. of IEEE Int. Conf. on Industrial Informatics (INDIN’08)*, 2008.
12. J. Holt and T. Baker, “Back propagation simulations using limited precision calculations,” in *Proc. of Int. Joint Conf. on Neural Networks (IJCNN’91)*, 1991.
13. A. Petrowski, “Choosing among several parallel implementations of the backpropagation algorithm,” in *Proc. of IEEE Int. Conf. on Neural Networks*, 1994.
14. D. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Handwritten digit recognition with a committee of deep neural nets on GPUs,” *Computing Research Repository*, 2011.
15. F. Mamalet, S. Roux, and C. Garcia, “Real-time video convolutional face finder on embedded platforms,” *EURASIP Journal on Embedded Systems*, 2007.
16. F. Mamalet, S. Roux, and C. Garcia, “Embedded facial image processing with convolutional neural networks,” in *Proc. of Int. Symp. on Circuits and Systems (ISCAS’10)*, 2010.