30th May 2022

LEARNING STRIDES IN CONVOLUTIONAL NEURAL NETWORKS

Authors: Rachid Riad, Olivier Teboul, David Grangier & Neil Zeghidour

LIGHTWEIGHT CONVOLUTIONAL NEURAL NETWORKS BY HYPERCOMPLEX PARAMETERIZATION

Authors: Eleonora Grassucci, Aston Zhang, Danilo Comminiello

SAPIENZA
UNIVERSITÀ DI ROMA

Professor: Aurelio Uncini
Tutor: Danilo Comminiello

Students: Cocci Federica - 1802435
Santilli Sofia - 1813509

Problem statement

Convolutional layers

extraction from the input data of a feature map

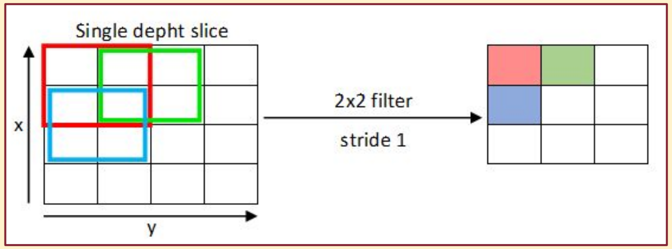
problem:

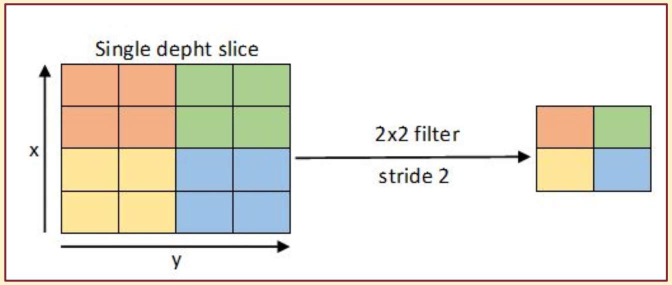
Location sensitive

solution:

Downsampling

using Strides





Pooling layer

Fixed Spectral pooling

Having the input x and the strides S

$$x \in R^{H \times W} \qquad S = (S_h, S_w) \in [1, H) \times [1, W)$$

- the Discrete Fourier Transform of x is computed

$$y = F(x) \in C^{H \times W}$$

- the bounding box crops the input in the frequency domain

$$\bar{y} \in C^{\lfloor \frac{H}{S_h} \rfloor \times \lfloor \frac{W}{S_w} \rfloor}$$

- the output is brought back to the spatial domain, through the inverse DFT

$$\bar{x} = F^{-1}(\bar{y}) \in R^{\lfloor \frac{H}{S_h} \rfloor \times \lfloor \frac{W}{S_w} \rfloor}$$

Fixed Spectral pooling

- Truncation in the frequency domain.

- Flexibility:

non-integer strides → more fine-grained downsizing

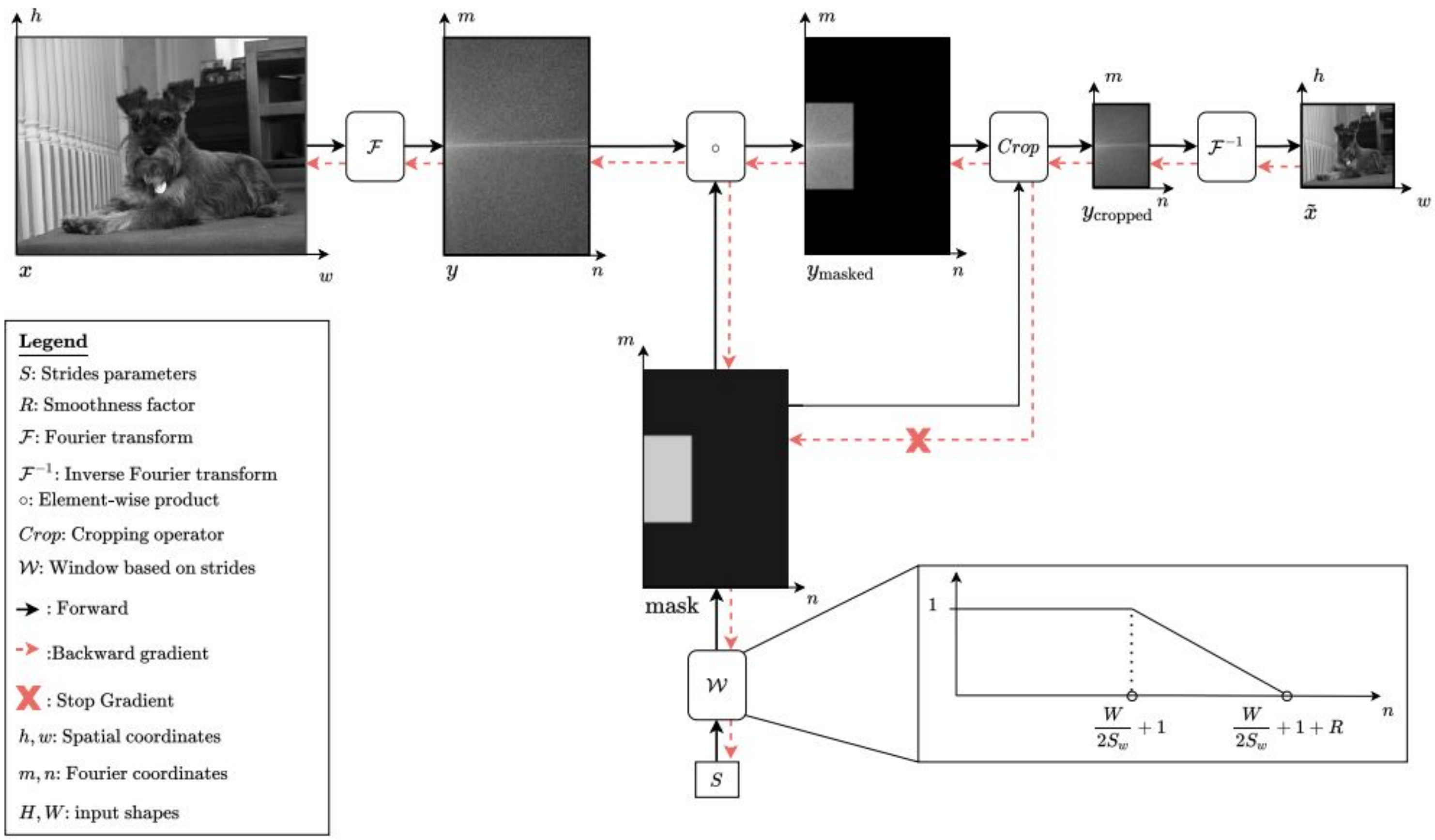
- Preservation of more information: it is a type of denoising.

- Strides still an hyperparameter, not learnable

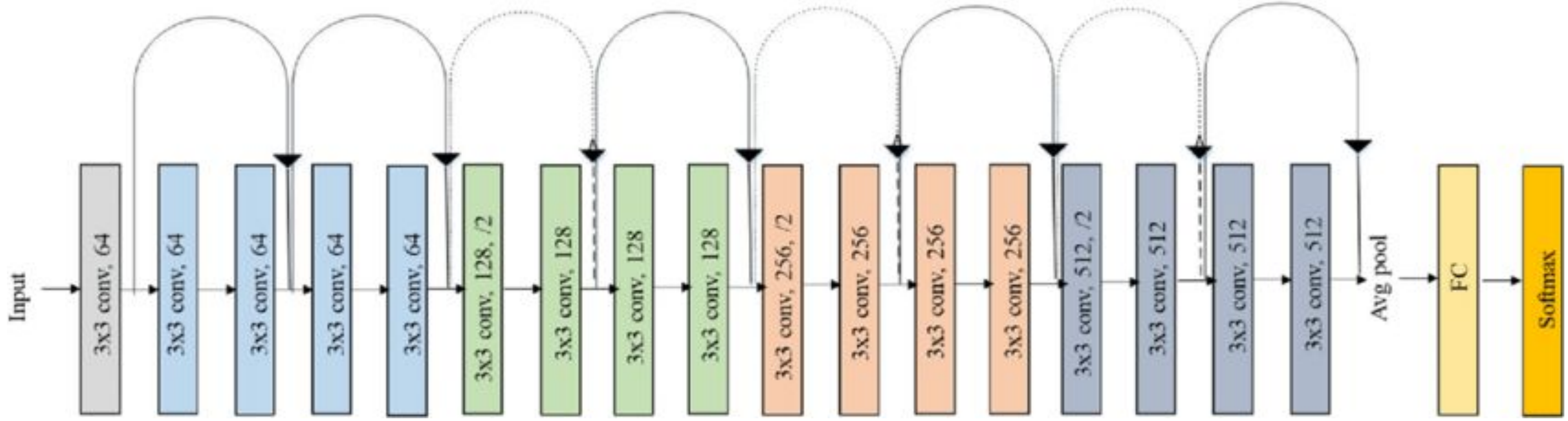
Diffstride (learnable strides)

$$mask_w = min[max[\frac{1}{R}(R + \frac{W}{2S_w} - n), 0], 1]\ , n \in [0, \frac{W}{2} + 1]$$

$$mask_h = min[max[\frac{1}{R}(R + \frac{H}{2S_h} - |\frac{H}{2} - m|), 0], 1]\ , m \in [0, H]$$

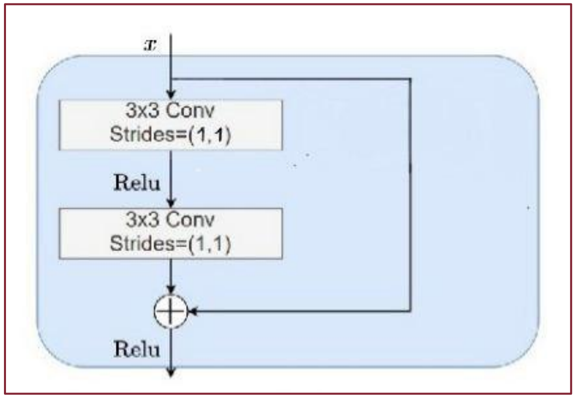


Network: ResNet-18

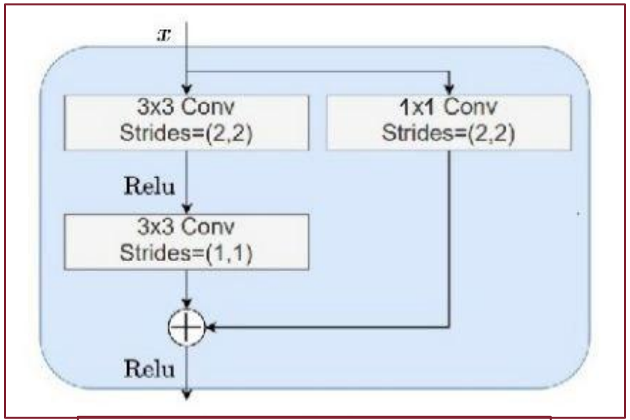


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				

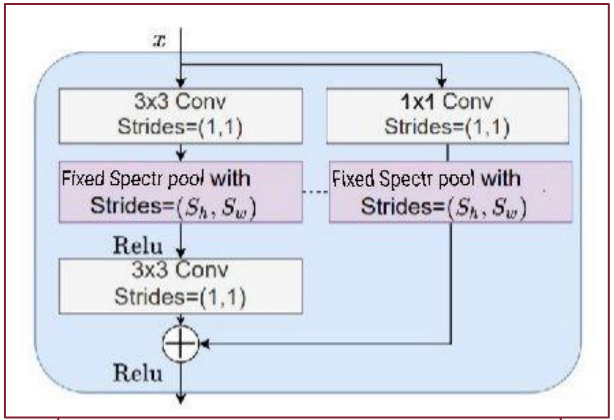
Identity and Residual blocks



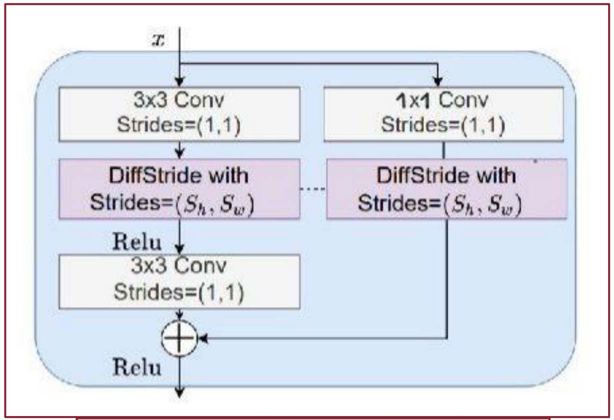
Identity block



Residual block with a strided convolution



Residual block with a Fixed Spectral pooling layer



Residual block with a Diffstride layer

Dataset: CIFAR-10

- 60.000 coloured images 32x32
- 10 classes (6.000 images per class): *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*

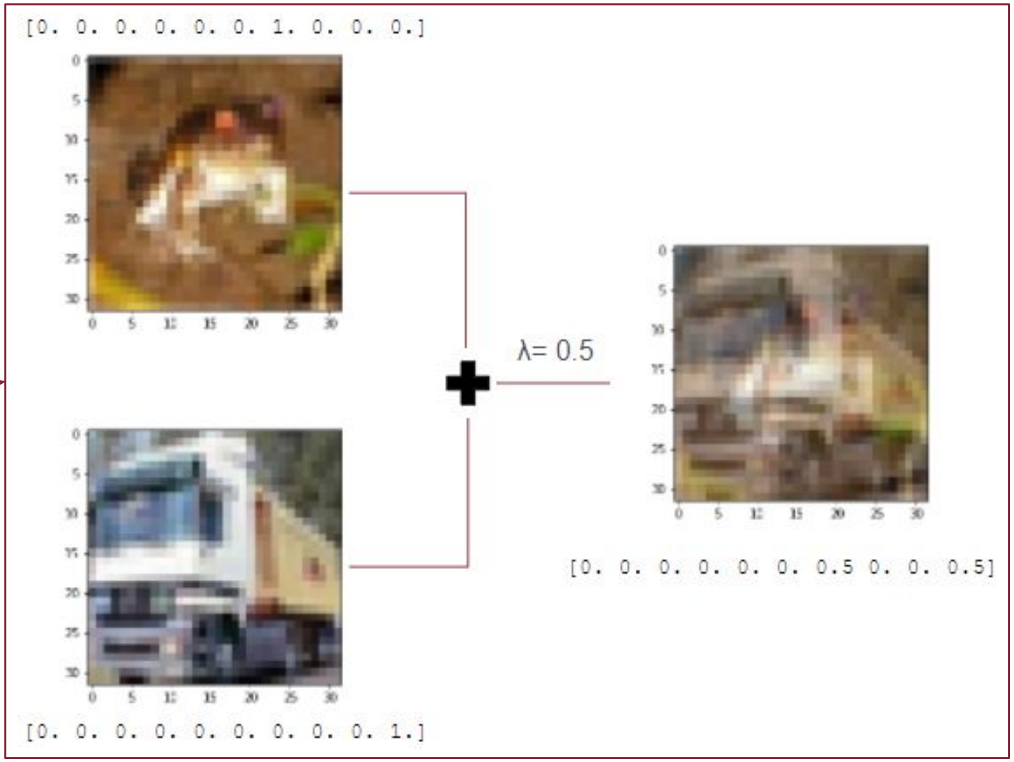
- 50.000 training images, 5.000 validation images, 5.000 test images

Preprocessing

- normalization

```
mean=[0.4914, 0.4822, 0.4465]
std=[0.2470, 0.2435, 0.2616]
```

- random cropping
- random flipping left-to-right
- mix up



Experiments

Strides initialization for the 3 residual layers = *(2, 2, 2)*

	Strided convolution	Fixed Spectral pooling	Diffstride
mixup + batch 128	0,658	0,6698	0,817
mixup + batch 256	0,6766	0,6834	0,7854
no mixup + batch 128	0,7334	0,7575	0,8692
no mixup + batch 256	0,7464	0,7544	0,8172

We have used *150 epochs* for strided convolution and fixed spectral pooling

We have used *40 epochs* with *early stopping* for diffstride

Other experiments and Conclusions

Running these experiments *without mixup* and batch of *128*

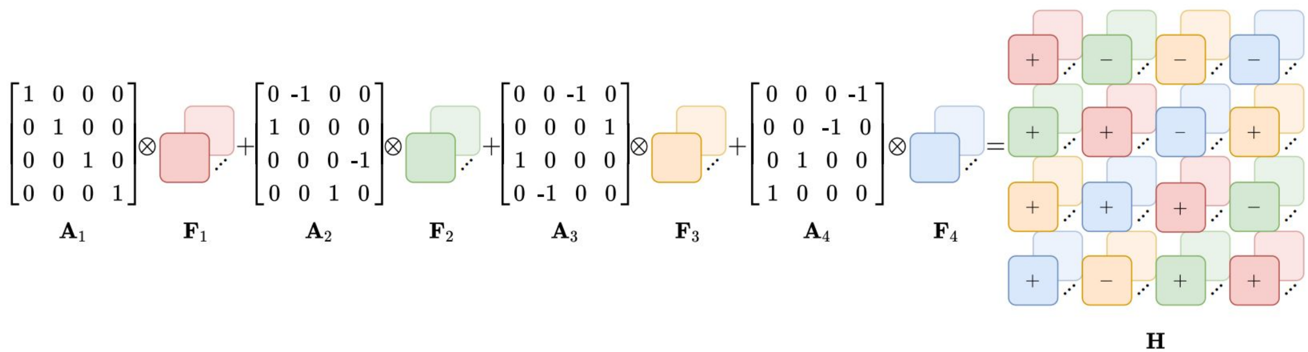
	Strided convolution	Fixed Spectral pooling	Diffstride
(2, 2, 3)	0,7368	0,7486	0,8672
(3, 1, 3)	0,7214	0,7378	0,8672
(3, 1, 2)	0,7458	0,7642	0,8712

- pro: Diffstride outperforms the standard downsampling layers
- contro: higher computational cost

Additional implementation with PHC layer

- PHC = parametrized hypercomplex convolution
- Reduce the overall number of parameters by a factor N
- From Pytorch to Tensorflow
- Generalizes the hypercomplex multiplication as sum of Kronecker products between two learnable matrices:

$$\boxed{H = \sum_{i=1}^{n} A_i \otimes F_i}$$



$$H \in \mathbb{R}^{s\times d\times k\times k}$$

s = input dimension

d = output dimension

k = filter size

$$\boxed{y = PHC(x) = H * x + b}$$

Experiments

- PHC layer used instead of 2D convolutions
- N = 3 → ⅓ parameters of the previous experiments
- Strides initialization for the 3 residual layers = *(2, 2, 2)*
- Dataset CIFAR 10

Strided convolution	Fixed Spectral pooling
0,4748	0,501

- Also with PHC layers, downsampling the image in the frequency domain brings improvements in the accuracy