Dilated convolution with learnable spacings

Ismail Khalfaoui-Hassani[1], Thomas Pellegrini[1,2] and Timothée Masquelier[3]

[1]Artificial and Natural Intelligence Toulouse Institute (ANITI)
[2]IRIT, CNRS, Toulouse, France.
[3]CerCo UMR 5549, CNRS. Université de Toulouse, France.

Who are we?

Ismail Khalfaoui Hassani
Phd Student
ANITI
ismail.khalfaoui-hassani@univ-tlse3.fr

Thomas Pellegrini
Researcher
IRIT
thomas.pellegrini@irit.fr

Timothée Masquelier
Research Director
CNRS-CerCo
timothee.masquelier@cnrs.fr

DCLS in a nutshell

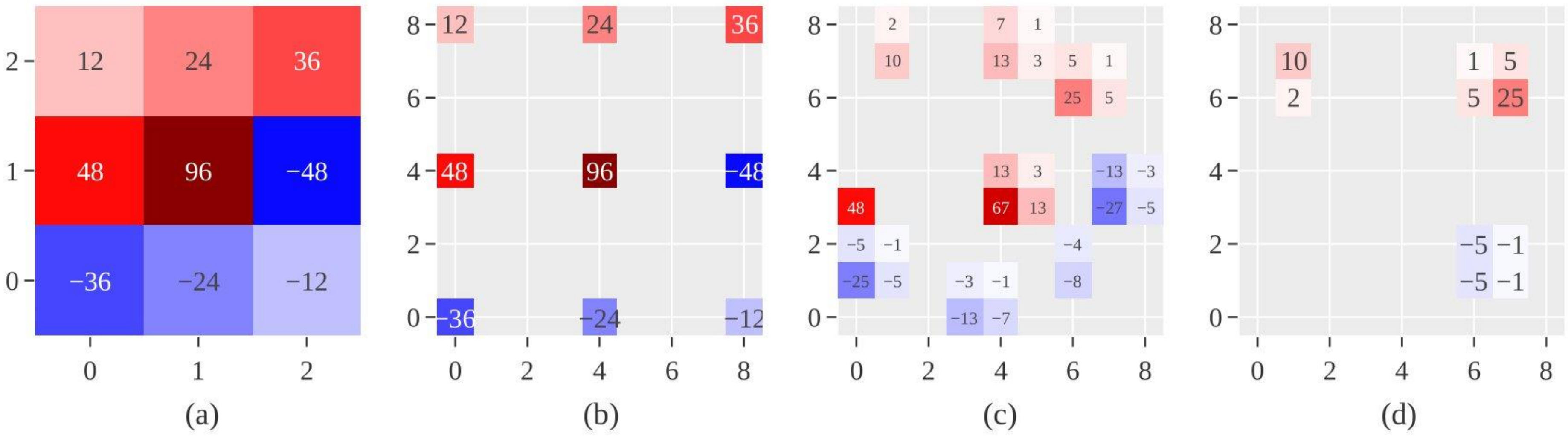


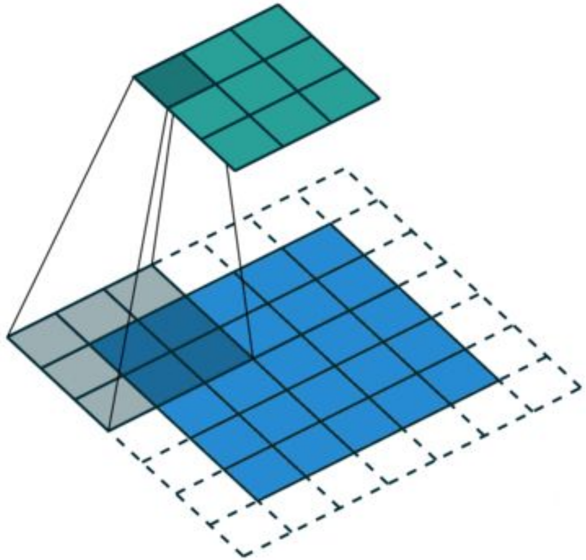
Figure 1.

(a): a standard 3 x 3 kernel.
(b): a dilated 3 x 3 kernel with dilation rate 4.
(c): a 2D-DCLS kernel with 9 kernel elements and a dilated kernel size of 9. Each weight is spread over up to four adjacent pixels.
(d): a 2D-DCLS kernel with 3 kernel elements and still a dilated kernel size of 9.

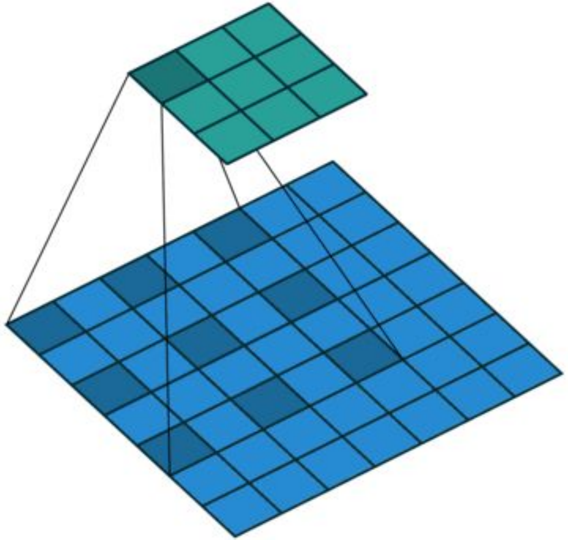
Motivation

- Recent papers indicate that to compete with transformers, CNNs need large kernels: e.g. 7x7 for ConvNeXt, 31x31 for RepLKNet, and even 51x51 for SLaK!
- However, the number of parameters explodes ! DCLS allows to increase the RF sizes without increasing the number of parameters !
- This is also true for the standard dilated convolution, but the regular grid is too rigid. DCLS is more flexible !

Convolutions in action



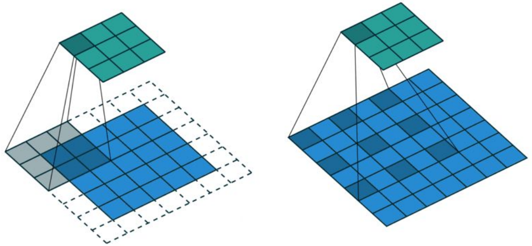
3x3 standard convolution

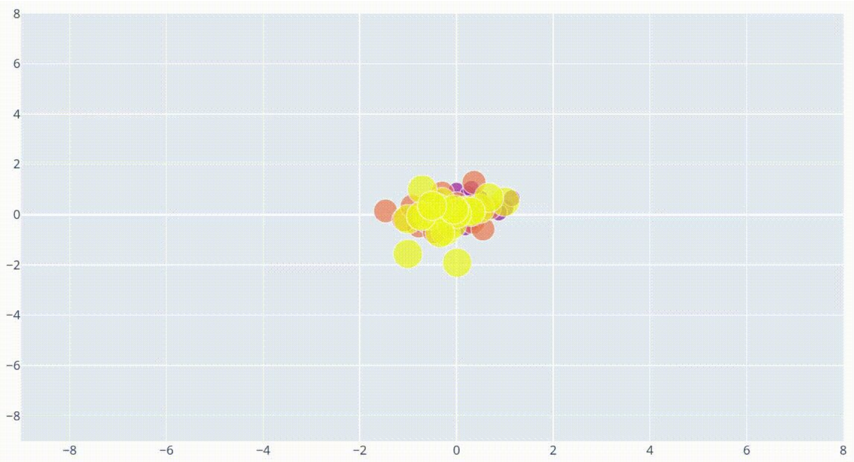


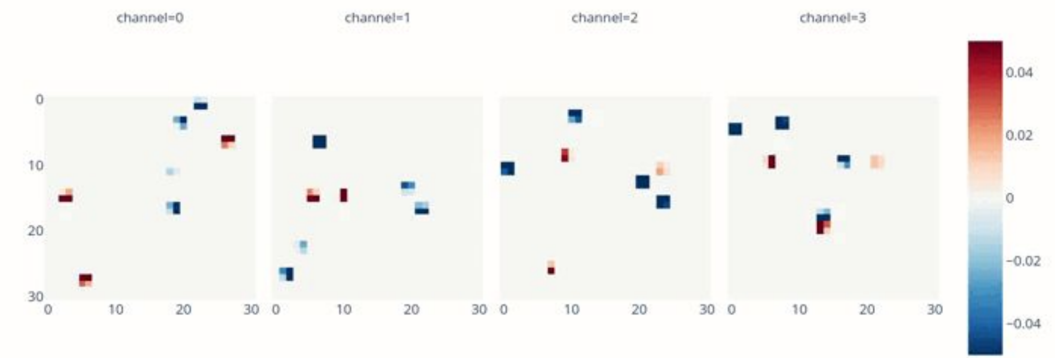
3x3 dilated convolution d1=d2=2

(animation by [Dumoulin and Visin, 2016])

DCLS in action

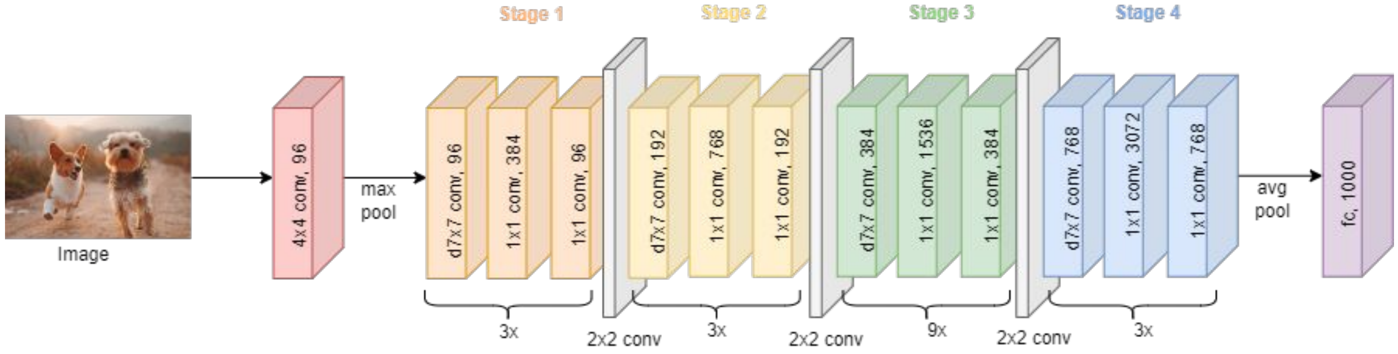






Experiment methodology

- Start from a ConvNext model with the baseline configuration.
- Rigorously fix the same configuration as the baseline (seed, cudnn benchmark, arguments, hyperparameters, effective batch size …).
- Replace all the depthwise separable convolutions by DCLS ones.



Results on image classification (ImageNet1k)

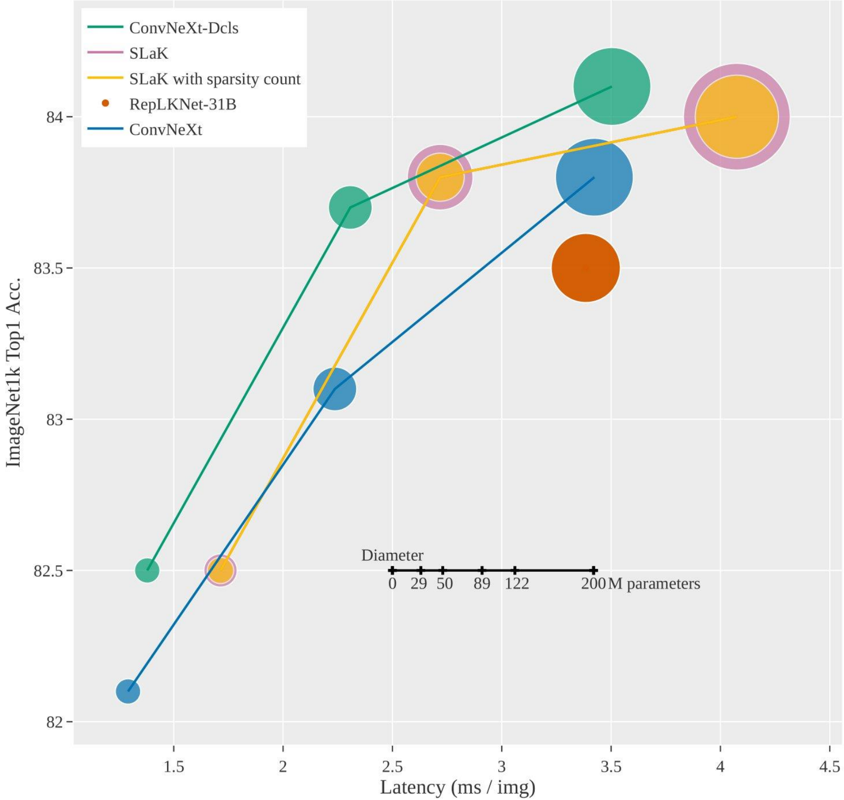


Figure 2. Classification accuracy on ImageNet-1K as a function of latency (i.e. inverse of the throughput). Dot diameter corresponds to the number of parameters.

Classification

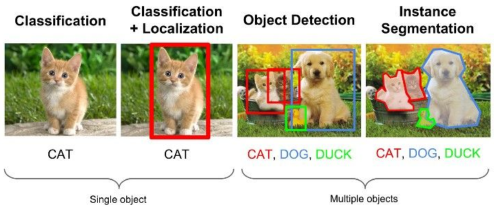
CAT

Results on image classification (ImageNet1k)

model	image size	# param.	FLOPs	throughput (image / s)	Top-1 acc.
Swin-T	224^2	28M	4.5G	757.9	81.3
ConvNeXt-T ●	224^2	29M	4.5G	**774.7**	82.1
ConvNeXt-T-dil2	224^2	29M	4.5G	**773.6**	80.8
ConvNeXt-T-ker17	224^2	30M	5G	560.0	82.0
SLaK-T ● ●	224^2	30M ● / 38M ●	9.4G	583.5	**82.5**
ConvNeXt-T-dcls ●	224^2	29M	5.0G	725.3	**82.5**
Swin-S	224^2	50M	8.7G	436.7	83.0
ConvNeXt-S ●	224^2	50M	8.7G	**447.1**	83.1
SLaK-S ● ●	224^2	55M ● / 75M ●	16.6G	367.9	**83.8**
ConvNeXt-S-dcls ●	224^2	50M	9.5G	433.4	**83.7**
Swin-B	224^2	88M	15.4G	286.6	83.5
ConvNeXt-B ●	224^2	89M	15.4G	**292.1**	83.8
RepLKNet-31B ●	224^2	79M	15.4G	**295.5**	83.5
SLaK-B ● ●	224^2	95M ● / 122M ●	25.9G	245.4	84.0
ConvNeXt-B-dcls ●	224^2	89M	16.5G	285.4	**84.1**

Table 2: **Classification accuracy on ImageNet-1K.** The inference throughput was calculated at inference using a single V100-32gb gpu and scaled to take into account all the optimizations used in Liu et al. (2022b). For the SLaK model, we report both the effective number of parameters returned by Pytorch ● and the one reported in Liu et al. (2022a) ●, that takes sparsity into account. The calculated FLOPs for this last model are different from the ones reported in Liu et al. (2022a), as we took the implicit gemm FLOPs into account.

Results on object detection (COCO)



backbone	FLOPs	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Cascade Mask-RCNN 3 × schedule							
ResNet-50	739G	46.3	64.3	50.5	40.1	61.7	43.4
X101-32	819G	48.1	66.5	52.4	41.6	63.9	45.2
X101-64	972G	48.3	66.4	52.3	41.7	64.0	45.1
Swin-T	745G	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-T ●	741G	50.4	69.1	54.8	43.7	66.5	47.3
ConvNeXt-dcls-T ●	751G	**51.2**	69.9	55.7	**44.5**	67.5	48.3
Swin-S	838G	51.9	70.7	56.3	45.0	68.2	48.8
ConvNeXt-S ●	827G	51.9	70.8	56.5	45.0	68.4	49.1
ConvNeXt-dcls-S ●	844G	**52.8**	71.6	57.6	**45.6**	69.0	49.3
Swin-B	982G	51.9	70.5	56.4	45.0	68.1	48.9
ConvNeXt-B ●	964G	52.7	71.3	57.2	45.6	68.9	49.5
ConvNeXt-dcls-B ●	987G	**53.0**	71.5	57.7	**46.0**	69.3	50.0

Table 4: **COCO object detection and segmentation results** using Cascade Mask-RCNN. Average Precision of the ResNet-50 and X101 models are from (Liu et al., 2021). FLOPs are calculated with image size (1280, 800).

Results on semantic Segmentation (Ade20k)

backbone	input crop.	mIoU (ss)	# param.	FLOPs
ConvNeXt-T ●	512^2	46.0	60M	939G
SLaK-T ●	512^2	**47.1**	65M	945G
ConvNeXt-T-dcls ●	512^2	**47.1**	60M	950G
ConvNeXt-S ●	512^2	**48.7**	82M	1027G
ConvNeXt-S-dcls ●	512^2	48.4	82M	1045G
ConvNeXt-B ●	512^2	49.1	122M	1170G
ConvNeXt-B-dcls ●	512^2	**49.3**	122M	1193G

Table 3: **ADE20K validation results** using UperNet (Xiao et al., 2018). We report mIoU results with single-scale testing. FLOPs are based on input sizes of (2048, 512).

Robustness evaluations

Model	FlOPs / Params	Clean	C(↓)	$\overline{\mathrm{C}}$(↓)	A	R	SK
ResNet-50	4.1/25.6	76.1	76.7	57.7	0.0	36.1	24.1
ConvNeXt-T	4.5/28.6	82.1	41.6	41.2	23.5	47.6	33.8
ConvNeXt-dcls-T	5.0/28.6	**82.5**	**41.5**	**39.7**	**23.9**	**47.8**	**34.7**
ConvNeXt-S	8.7/50.2	83.1	38.9	37.8	30.1	50.1	**37.1**
ConvNeXt-dcls-S	9.5/50.2	**83.7**	**37.8**	**35.2**	**33.7**	**50.4**	36.7
ConvNeXt-B	15.4/88.6	83.8	37.0	35.7	35.5	51.7	38.2
ConvNeXt-dcls-B	16.5/88.6	**84.1**	**36.3**	**34.3**	**36.8**	**52.6**	**38.4**

Table 5: **Robustness evaluation of ConvNeXt-dcls**. We reconducted this study for ConvNeXt. For ImageNet-C and ImageNet-Cbar, the error is reported rather than the accuracy. It was calculated for both datasets by taking the average error over 5 levels of noise severity and over all the noise categories available in the datasets.

Code

pip3 install dcls

- A cuda-PyTorch version + a PyTorch only version (cpu + gpu).
  - https://github.com/K-H-Ismail/Dilated-Convolution-with-Learnable-Spacings-PyTorch
- Requirements: PyTorch only.
- 1D, 2D and 3D versions are available !

```python
import torch
from DCLS.construct.modules import  Dcls2d

m = Dcls2d(96, 96, kernel_count=34, dilated_kernel_size=17, padding=8, groups=96)
input = torch.randn(128, 96, 56, 56)
output = m(input)
loss = output.sum()
loss.backward()
print(output, m.weight.grad, m.P.grad)
```

Learning techniques

- **Weight decay:** No weight decay on positions.
- **Positions initialization:** Initialising positions with normal law, std 0.5.
- **Positions Clamping:** Clamping the positions after each gradient step.
- **Dilated kernel size tuning:** Empirically tuned using histograms and agglutination. Same size for all the network. Fixed at 17x17.
- **Kernel count tuning:** Set to be at iso-parameter with the baseline. Same count for all the network.
- **Positions learning rate scaling:** Scaling the learning rate for positions (x5).
- **Synchronizing positions:** Sharing positions across layers.

What’s next ?

- Reconsider the choice of bilinear interpolation.
- Find interesting use cases for 1D (audio, temporal series …) and 3D convolution cases (video recognition, cloud points …).
- A dedicated architecture for DCLS via Neural Architecture Search
- Use DCLS to learn delays in spiking neural networks.

What’s your excuse for not trying it ?

- Easy to install and use.
- No additional param cost and a marginal throughput cost if used with depthwise separable convolution.
- Significantly increases performance.

THANKS!

Any questions?

Ismail Khalfaoui Hassani
Phd Student
ANITI
ismail.khalfaoui-hassani@univ-tlse3.fr

Thomas Pellegrini
Researcher
IRIT
thomas.pellegrini@irit.fr

Timothée Masquelier
Research Director
CNRS-CerCo
timothee.masquelier@cnrs.fr

https://github.com/K-H-Ismail/Dilated-Convolution-with-Learnable-Spacings-PyTorch

pip3 install dcls