CONVOLUTIONAL NEURAL NETWORKS

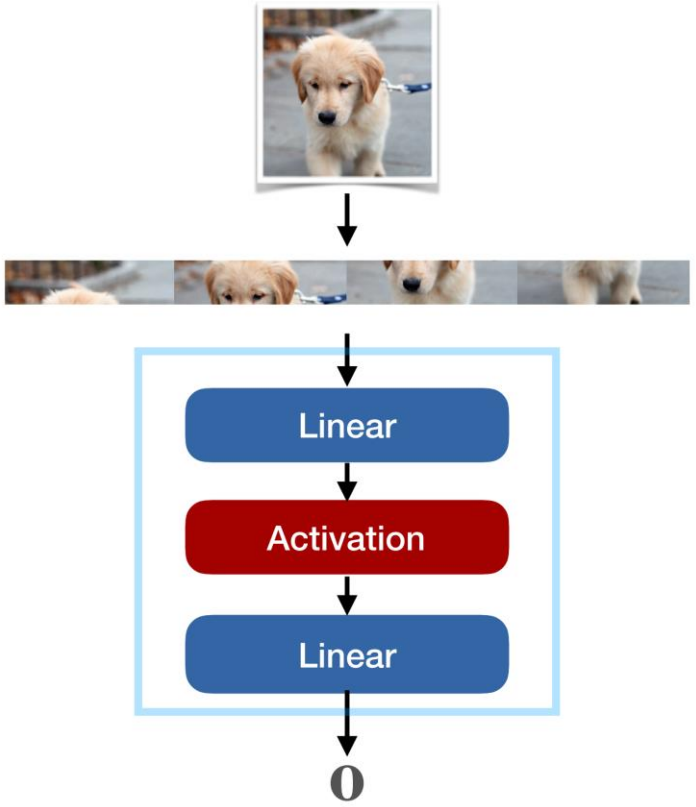
OSCAR LLORENTE GONZALEZ

OLLORENTE@COMILLAS.EDU

INDEX

1 - INTRODUCTION

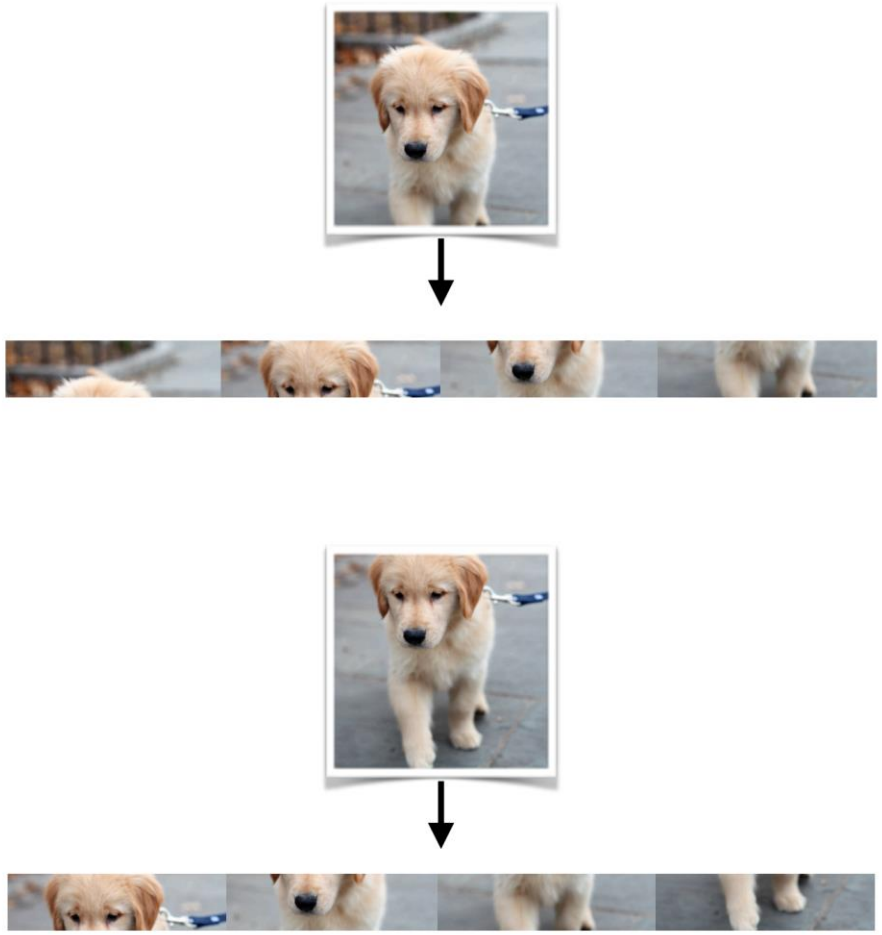
PROBLEMS WITH FEEDFORWARD NETS



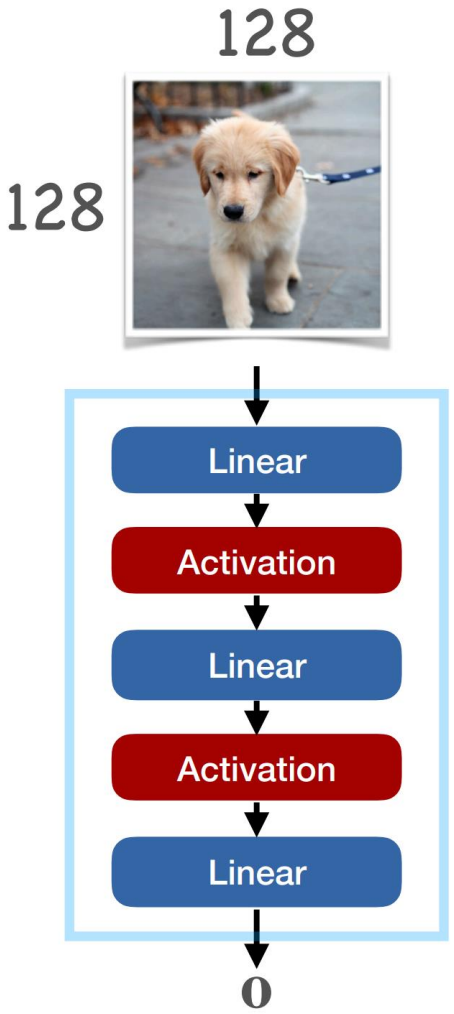
PROBLEMS WITH FEEDFORWARD NETS

Visual patterns are shift, rotation and scale invariant

Fully connected networks are not

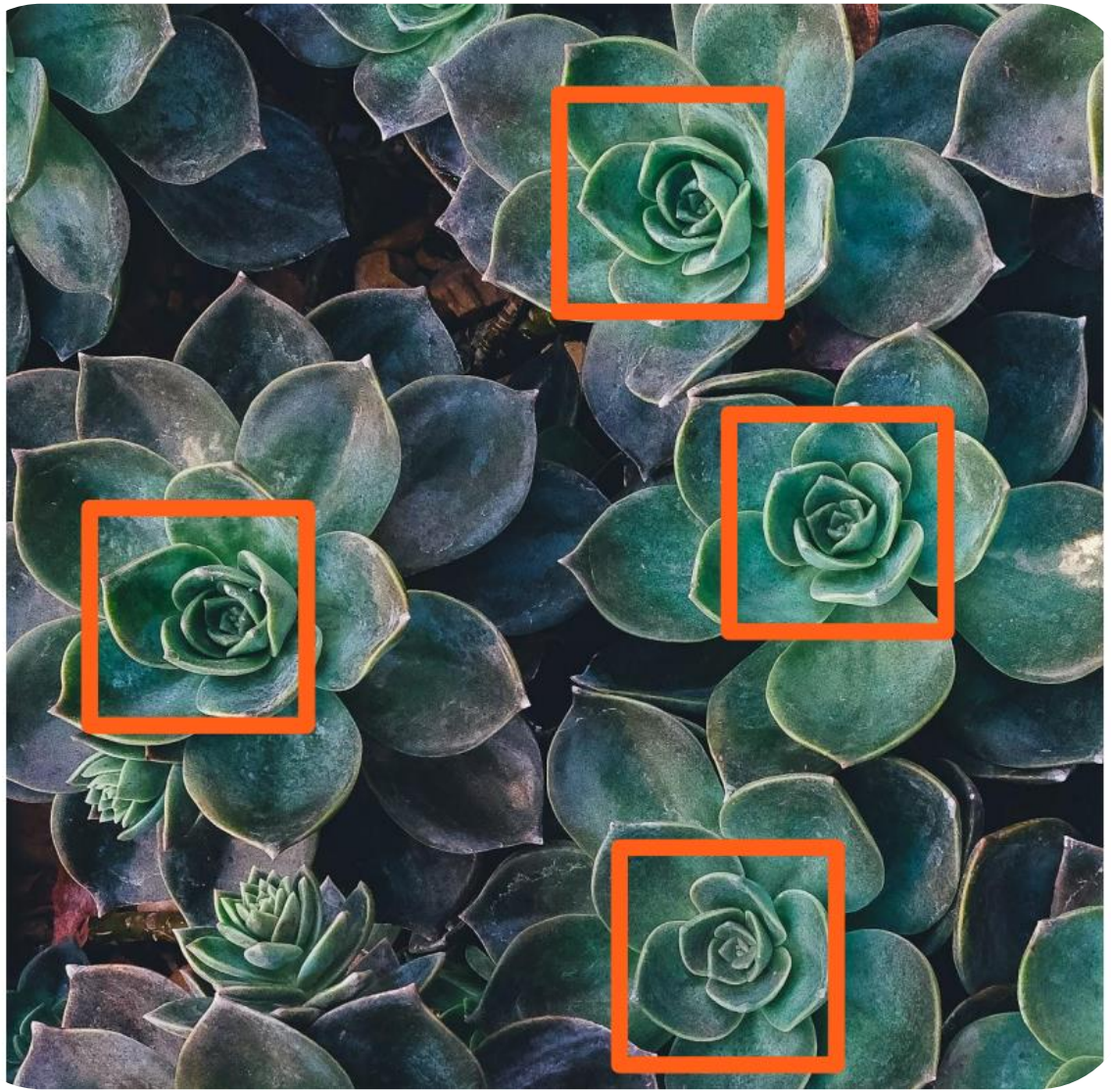


PROBLEMS WITH FEEDFORWARD NETS

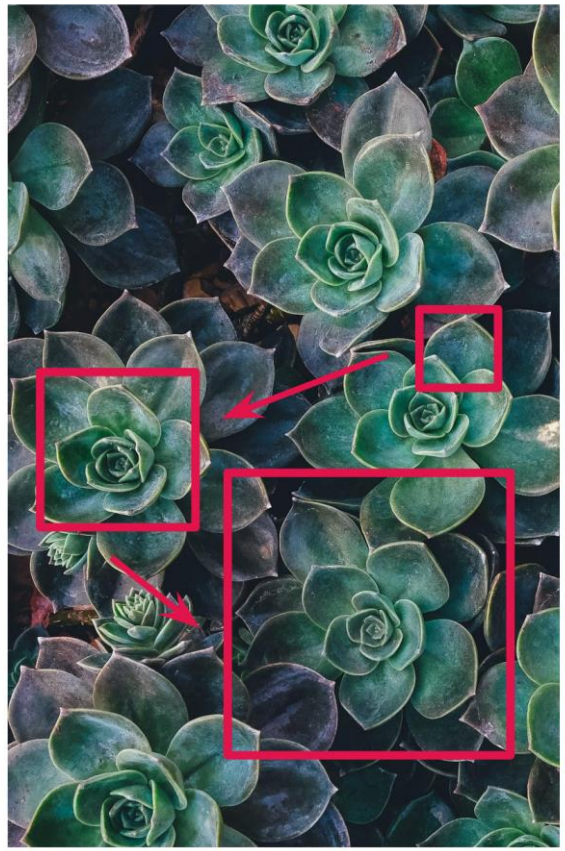


TAKING ADVANTAGE OF TOPOLOGICAL STRUCTURE

Weight sharing: use the same network parameters to detect local patterns at many locations in the image



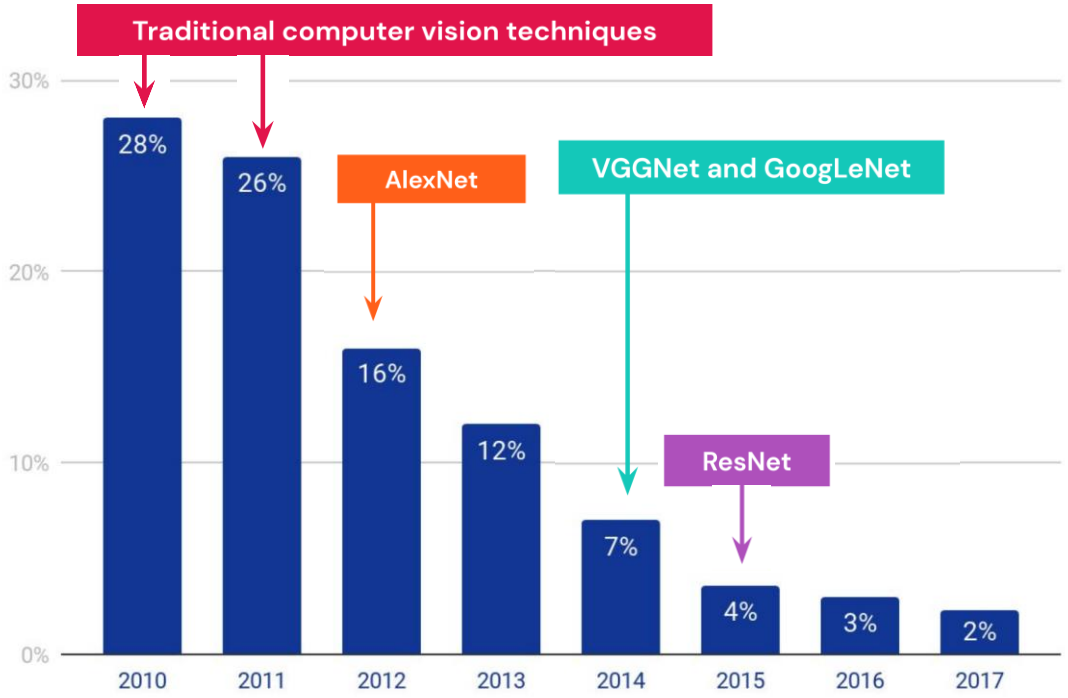
TAKING ADVANTAGE OF TOPOLOGICAL STRUCTURE



Hierarchy: local low-level features are composed into larger, more abstract features

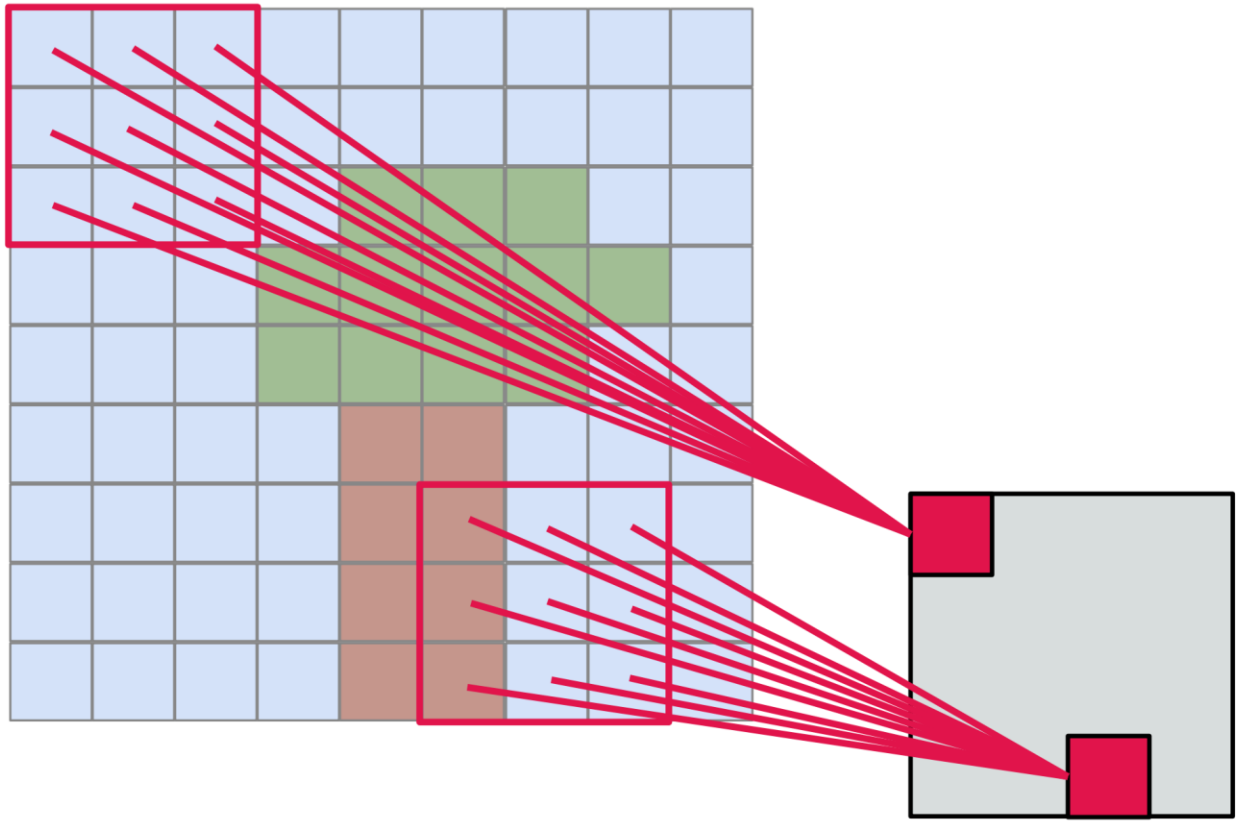
IMAGENET


IMAGENET


Traditional computer vision techniques
AlexNet
VGGNet and GoogLeNet
ResNet
30%
20%
10%
0%
28%
26%
16%
12%
7%
4%
3%
2%
2010
2011
2012
2013
2014
2015
2016
2017

2 - CONVOLUTIONAL LAYERS

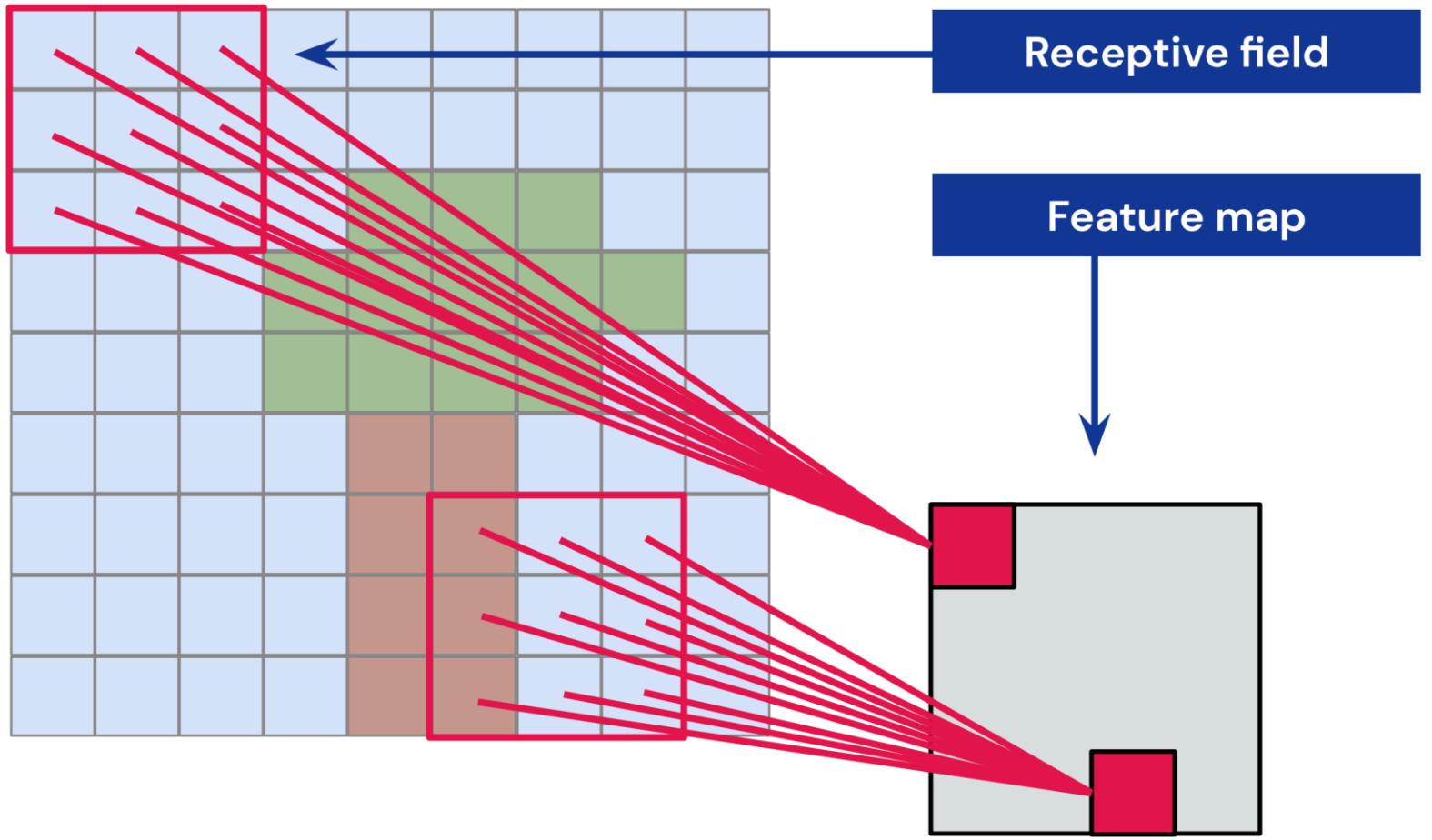
KERNEL



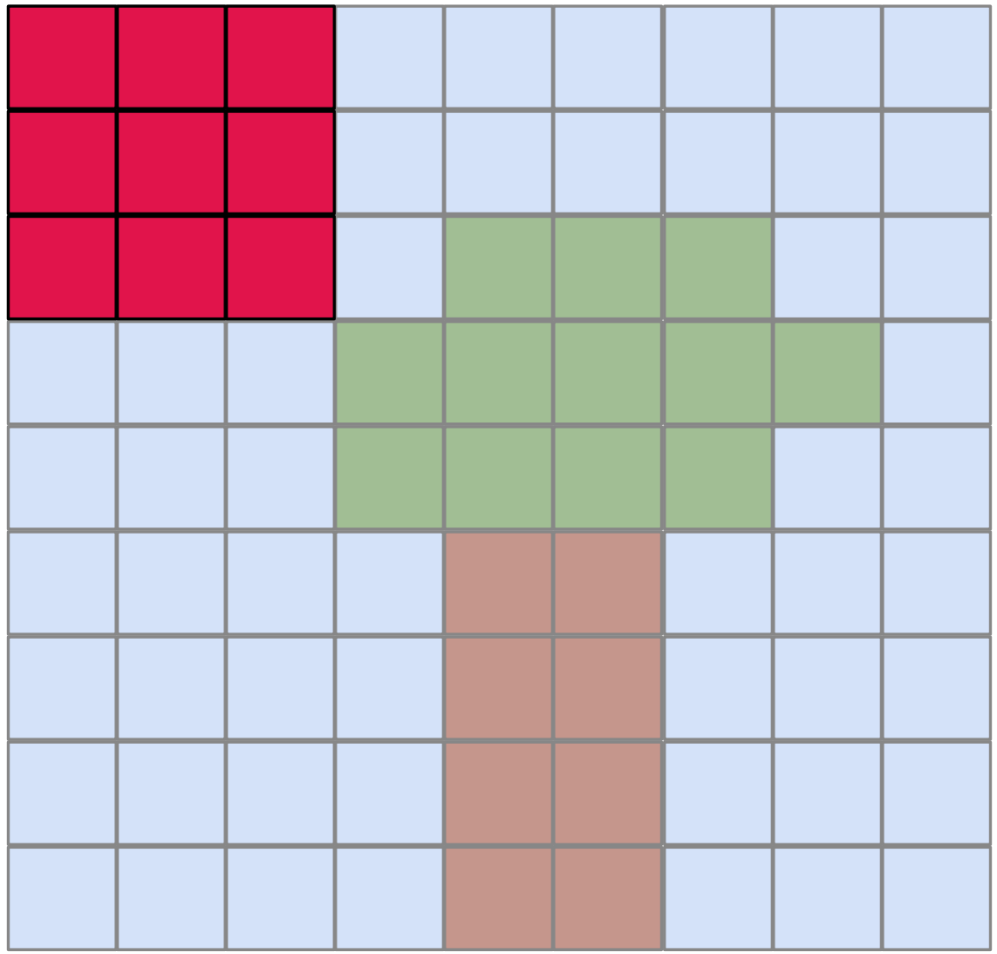
$$y = \mathbf{w} * \mathbf{x} + b$$

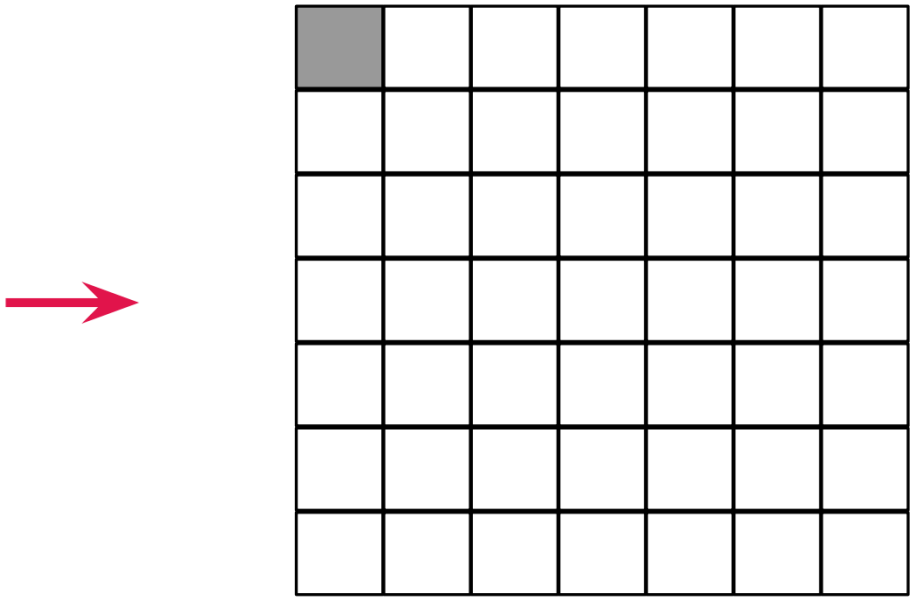
convolutional units
3×3 receptive field

FETURE MAP & RECEPTIVE FIELD



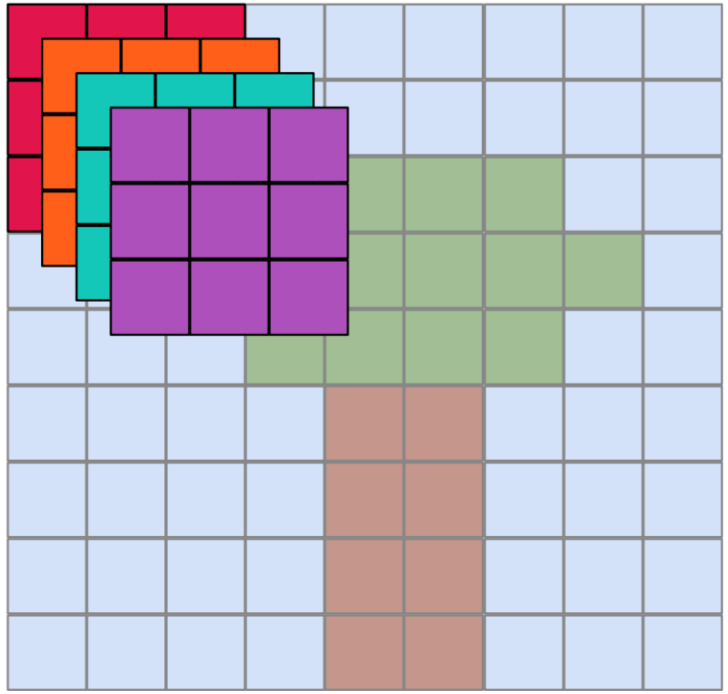
SLIDING ACROSS THE IMAGE

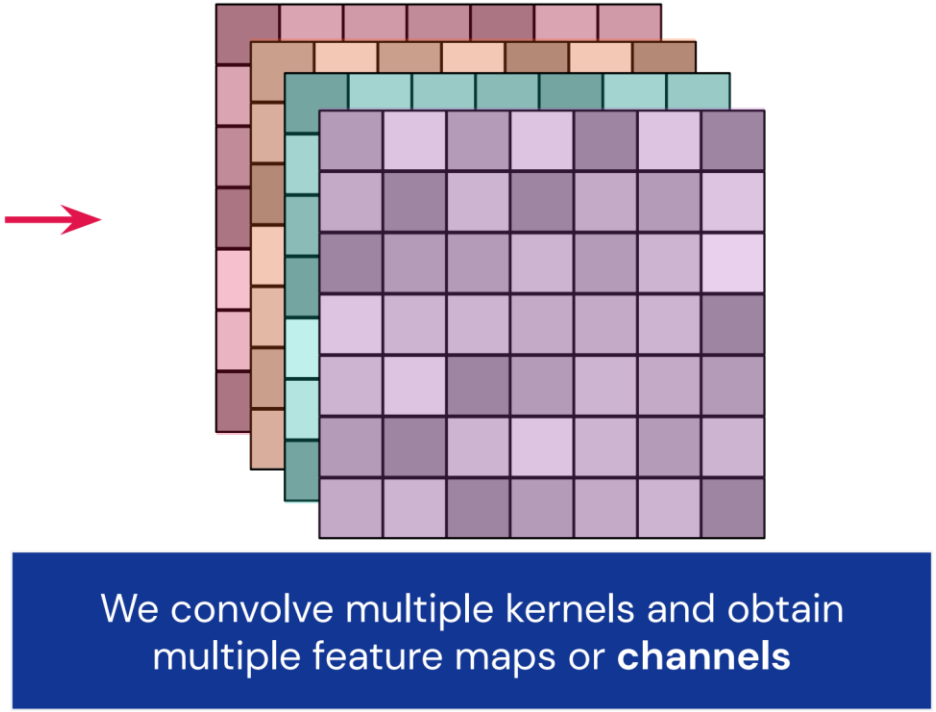




The **kernel** slides across the image and produces an output value at each position

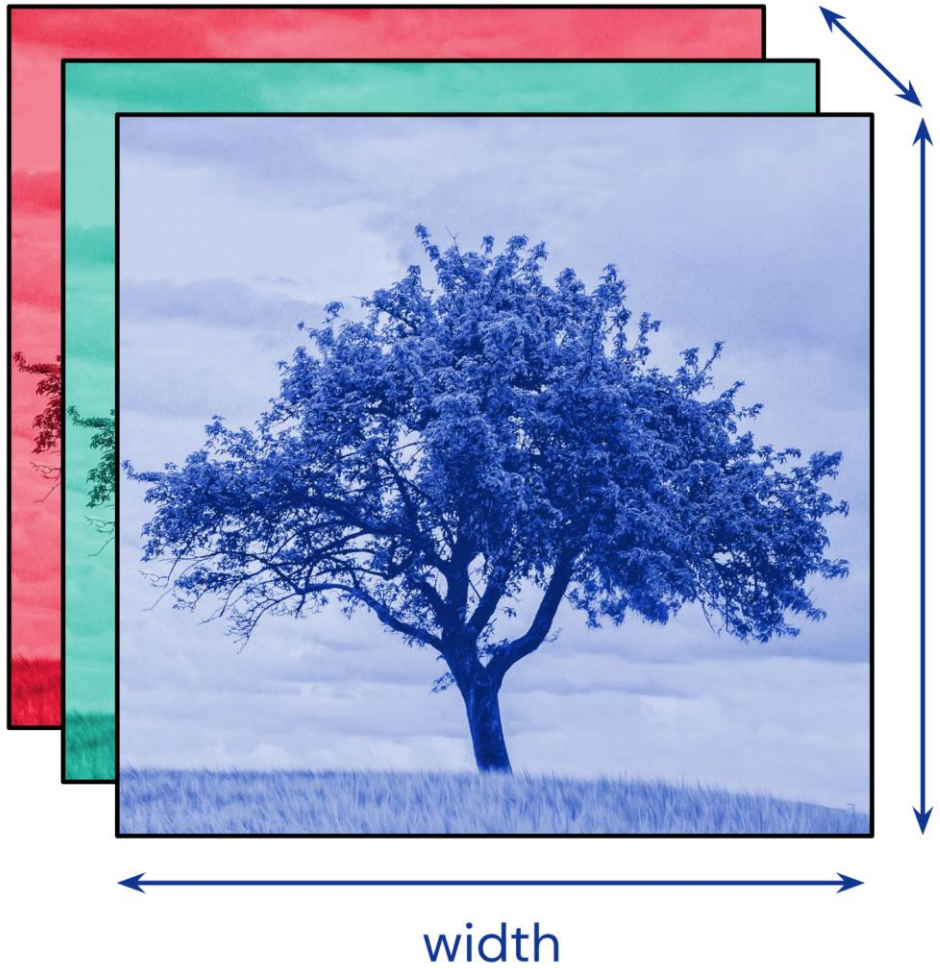
MULTIPLE CHANNELS





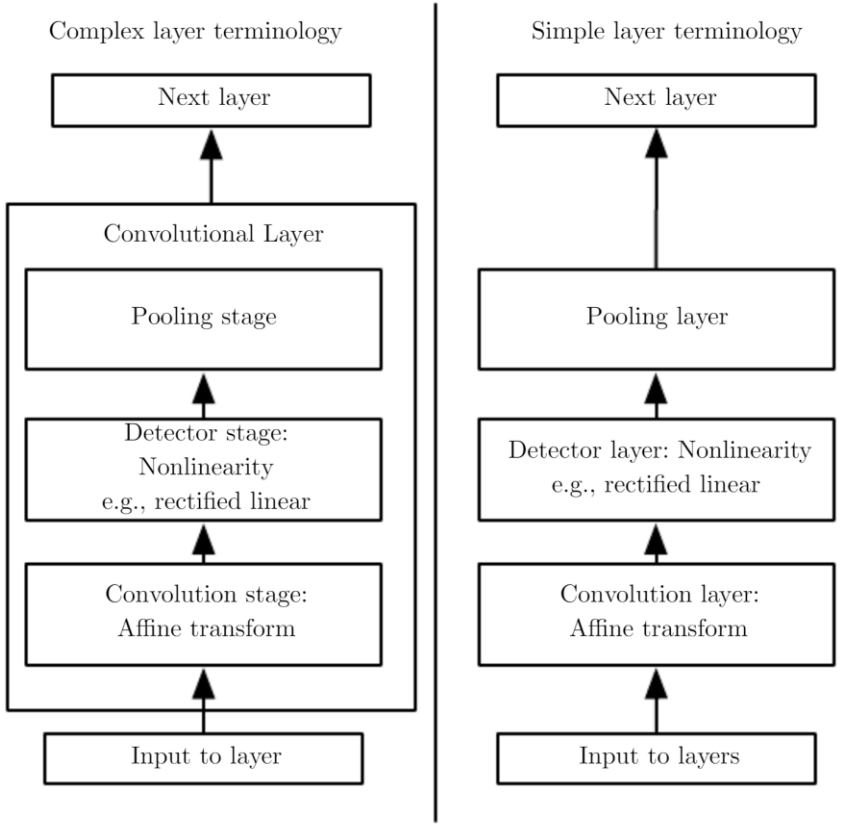
We convolve multiple kernels and obtain multiple feature maps or **channels**

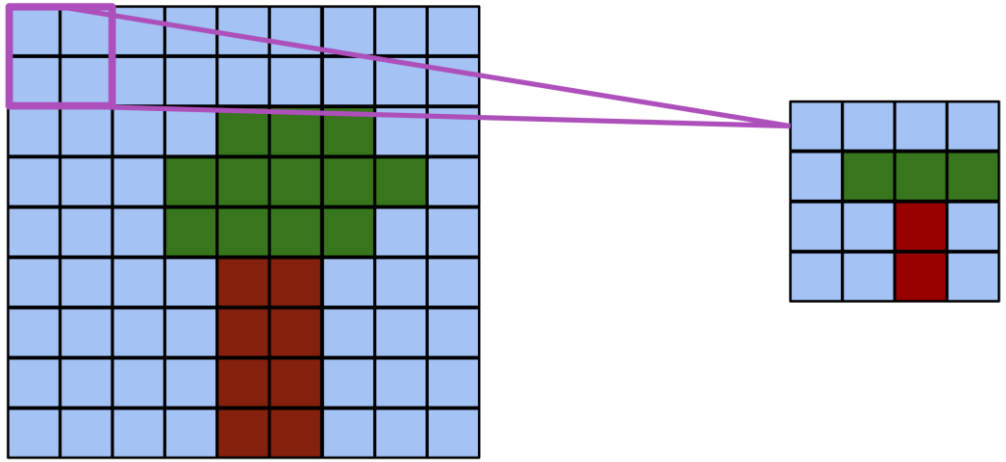
INPUTS AND OUTPUTS ARE TENSORS



3 - POOLING

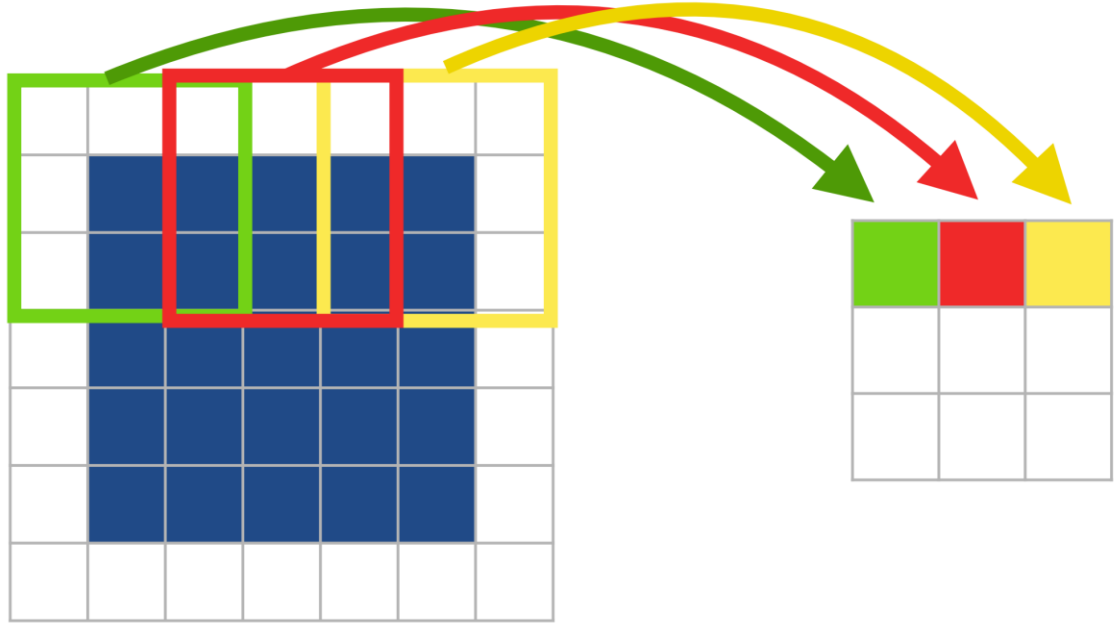
WHAT IS POOLING?



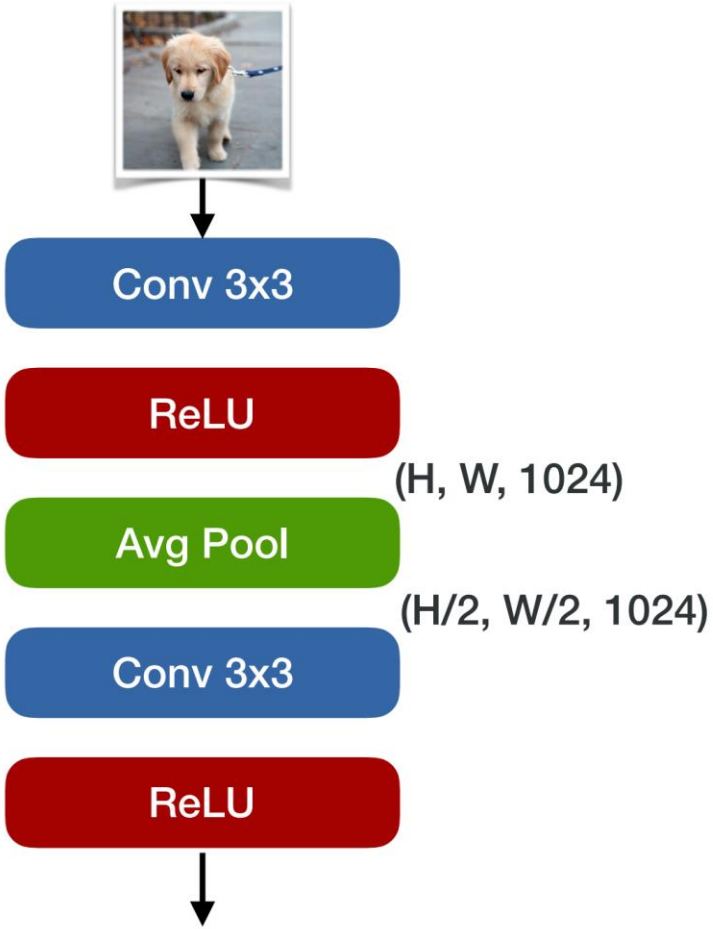


AVERAGE POOLING

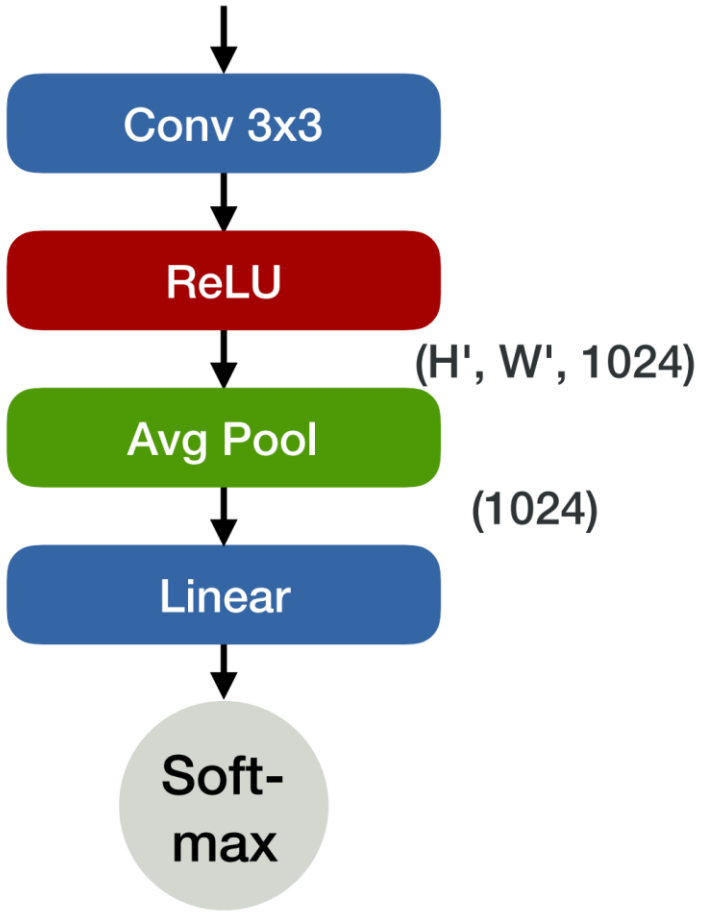
$$f_c(\mathbf{x}) = \text{mean}_{i,j}(\mathbf{x}_{i,j,c})$$



AVERAGE POOLING – REDUCE DIMENSIONALITY

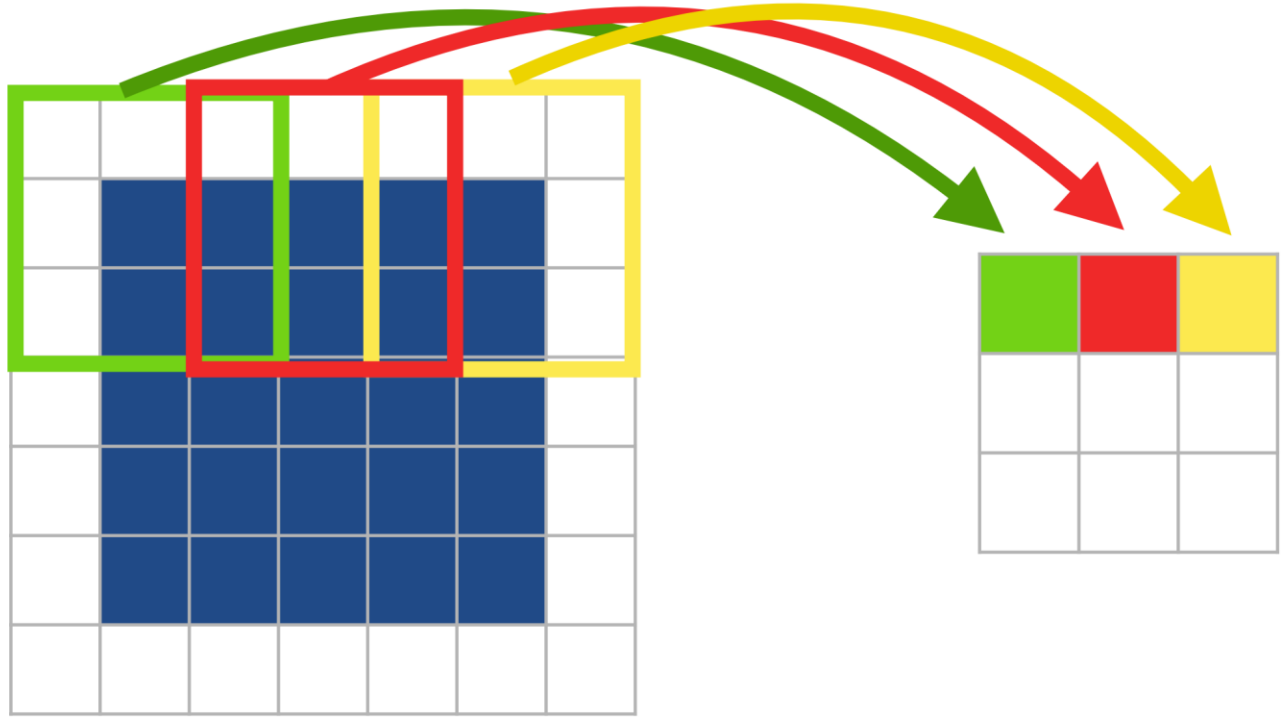


GLOBAL AVERAGE POOLING (GAP)

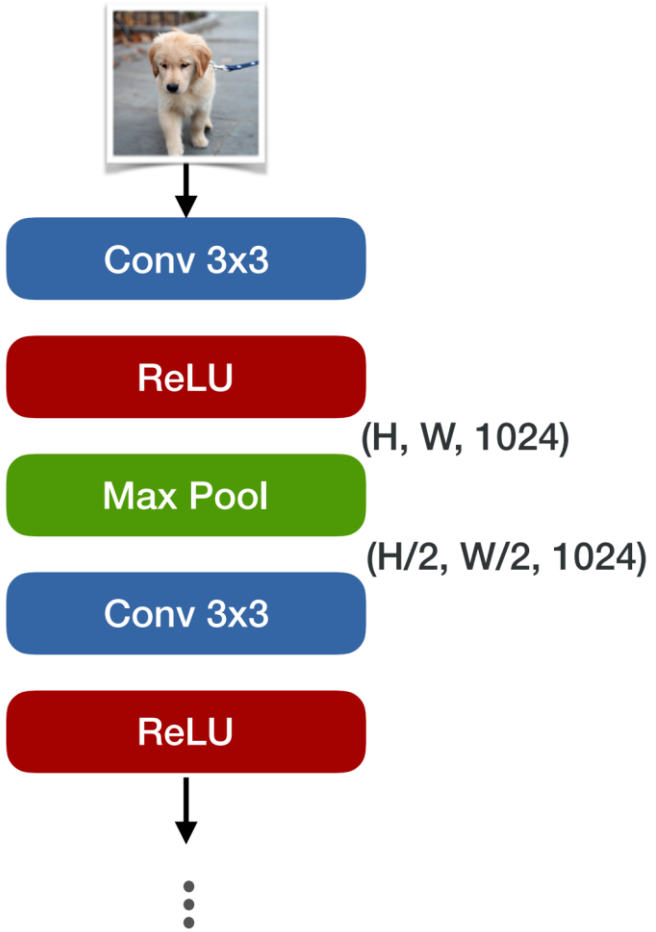


MAX POOLING

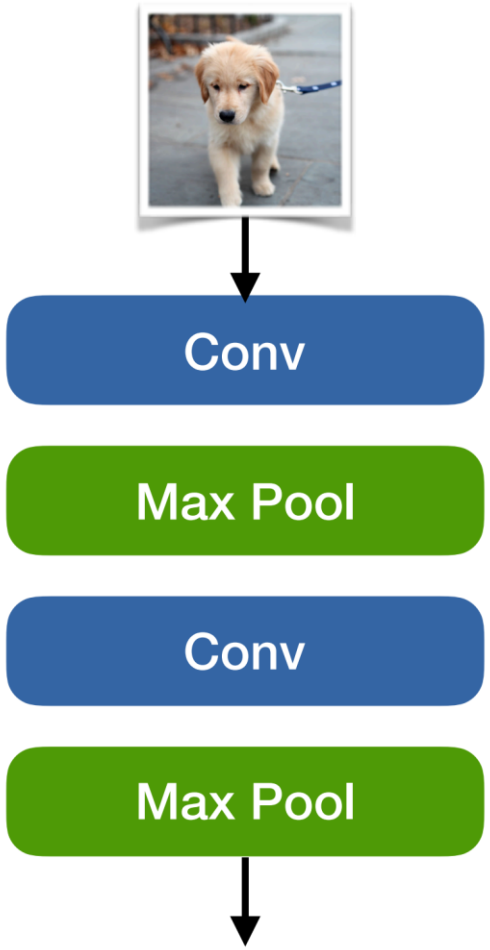
$$f_c(\mathbf{x}) = \max_{i,j}(\mathbf{x}_{i,j,c})$$



MAX POOLING – REDUCE DIMENSIONALITY

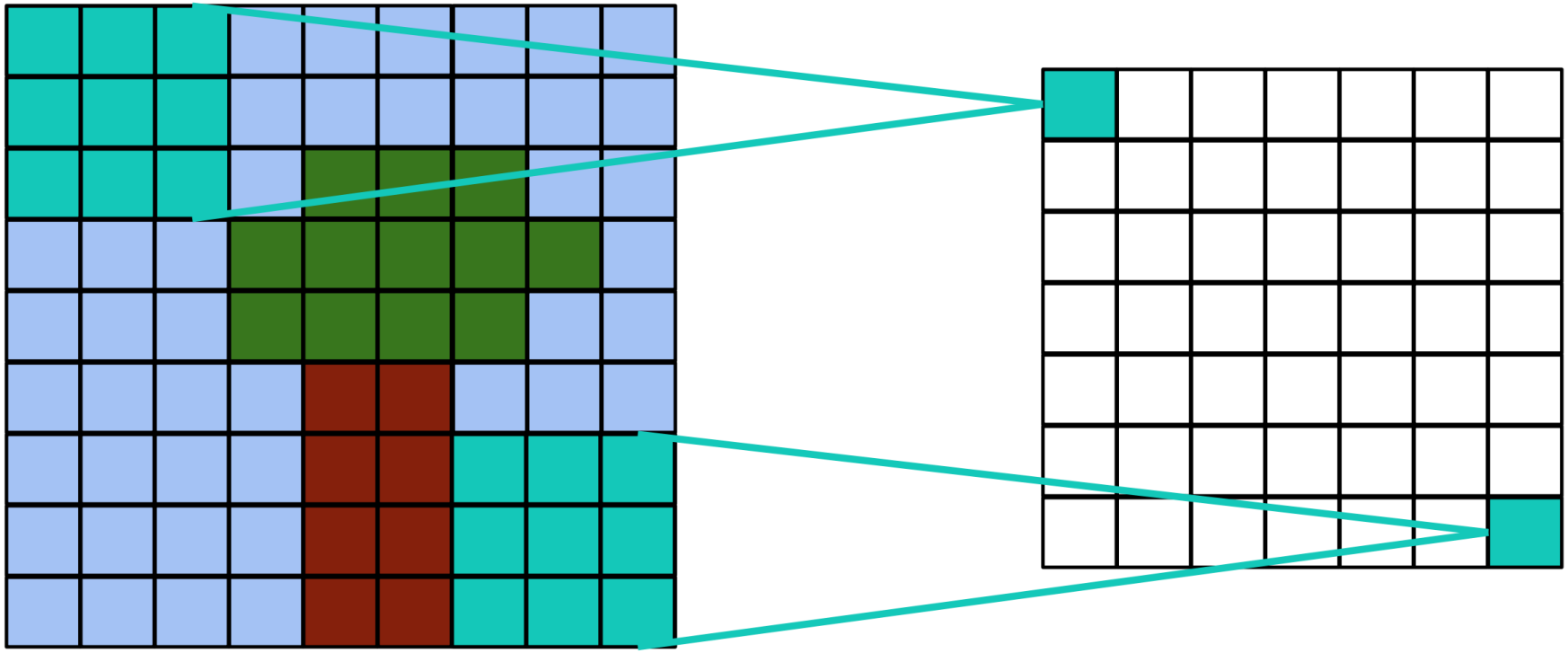


MAX POOLING – NONLINEARITY

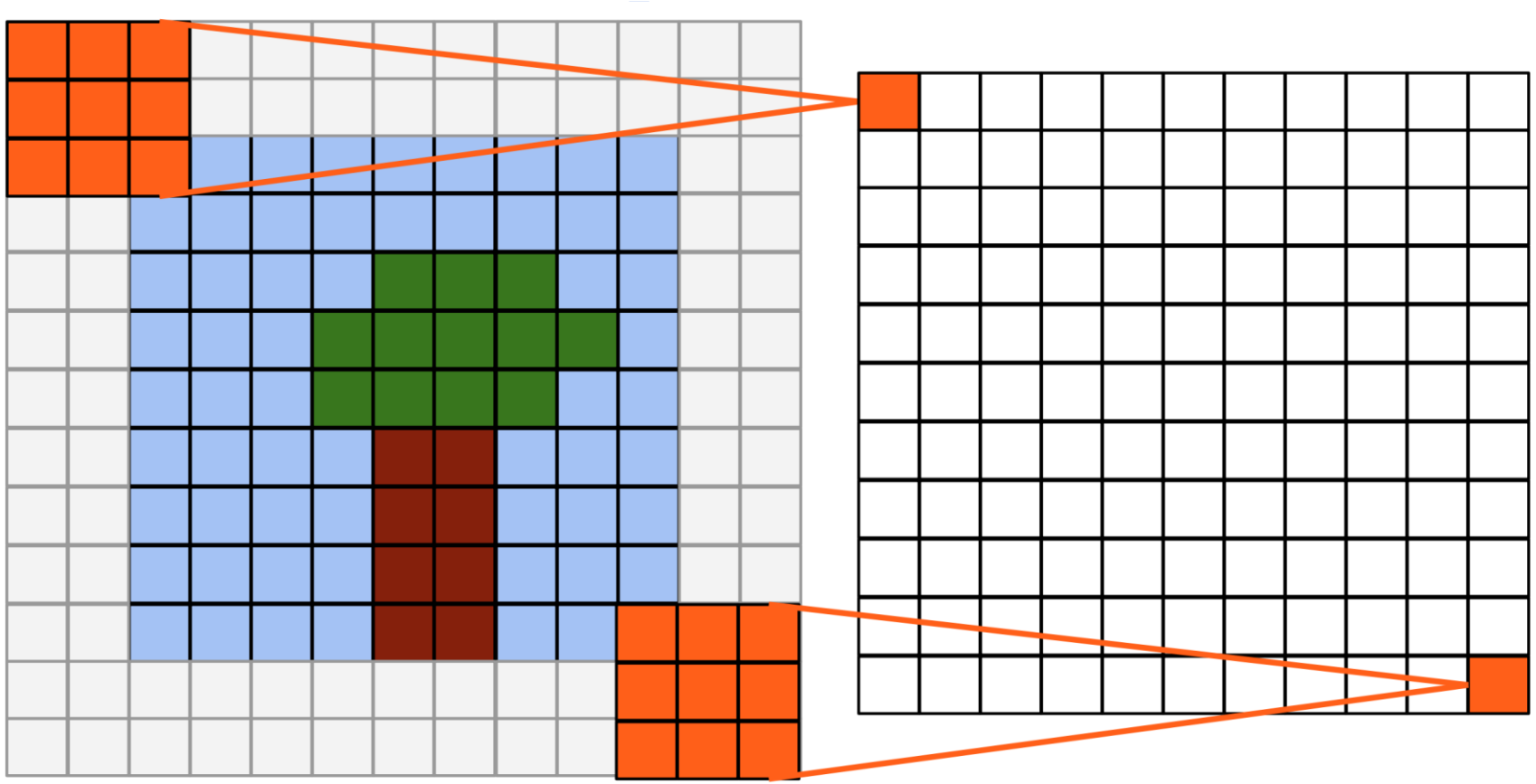


5 – PADDING, STRIDING AND GROUPS

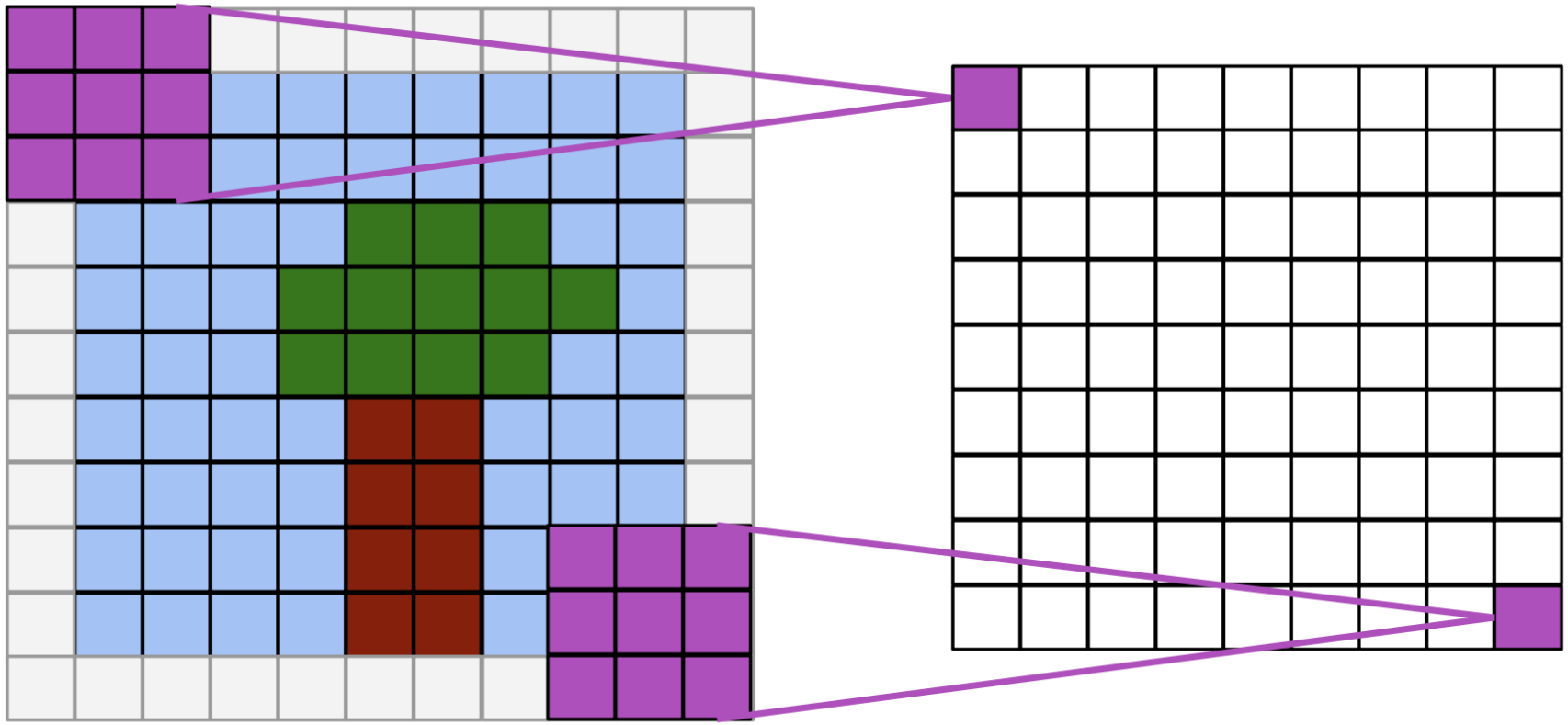
PADDING



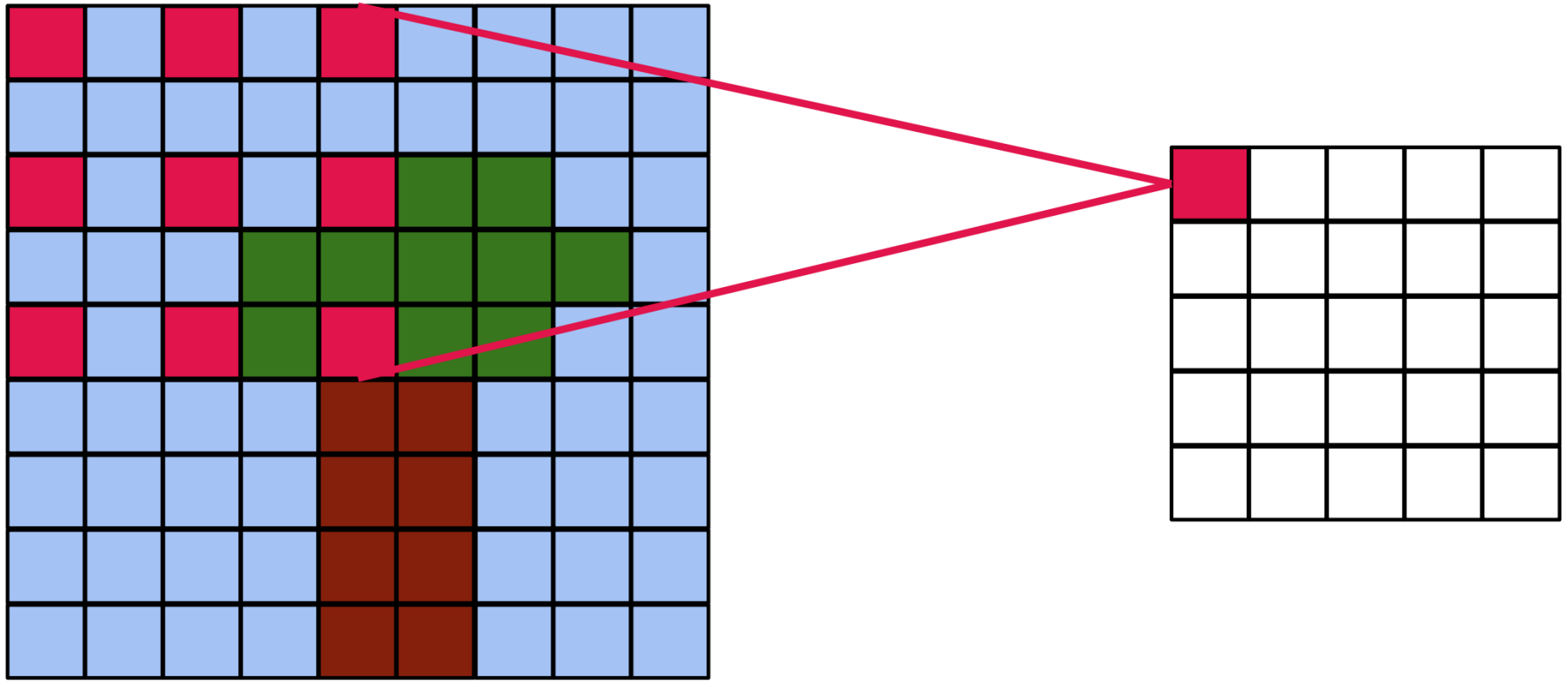
PADDING



PADDING

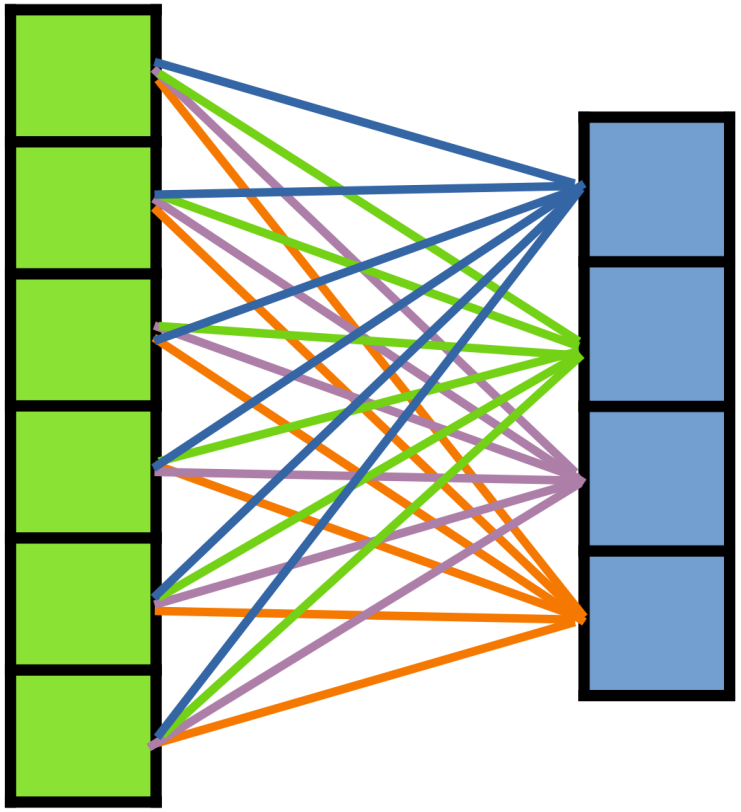


DILATION



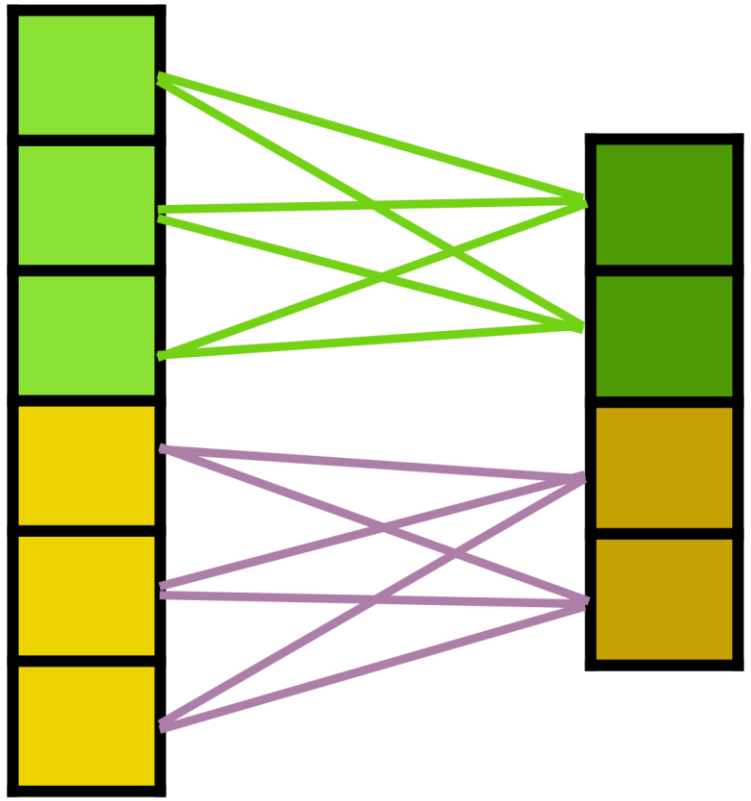
GROUPS

Every input channel is connected to every output channel



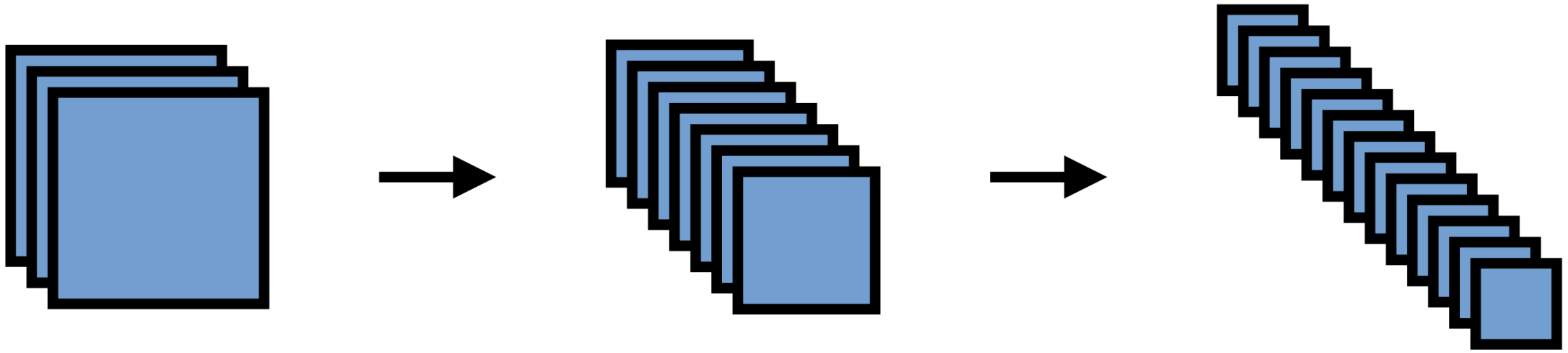
GROUPS

- Split channels into g groups
- Reduce parameters and computation by factor g

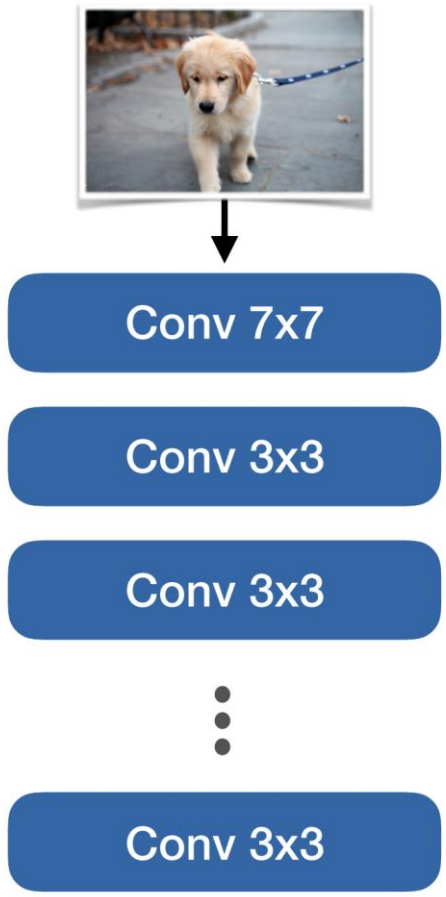


5 - DESIGN PRINCIPLES OF CONVOLUTIONAL NETWORKS

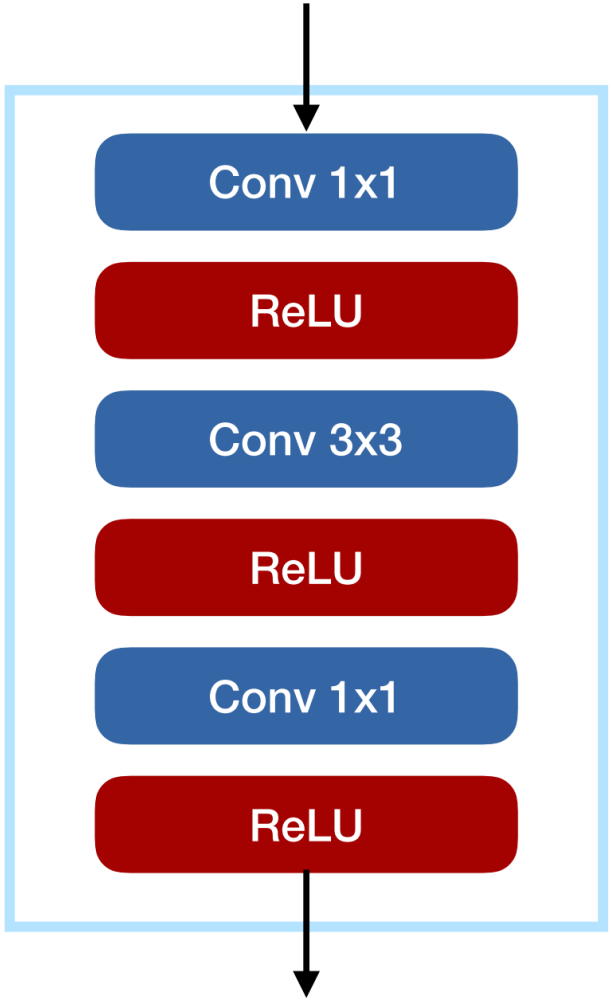
USE STRIDING, INCREASE CHANNELS



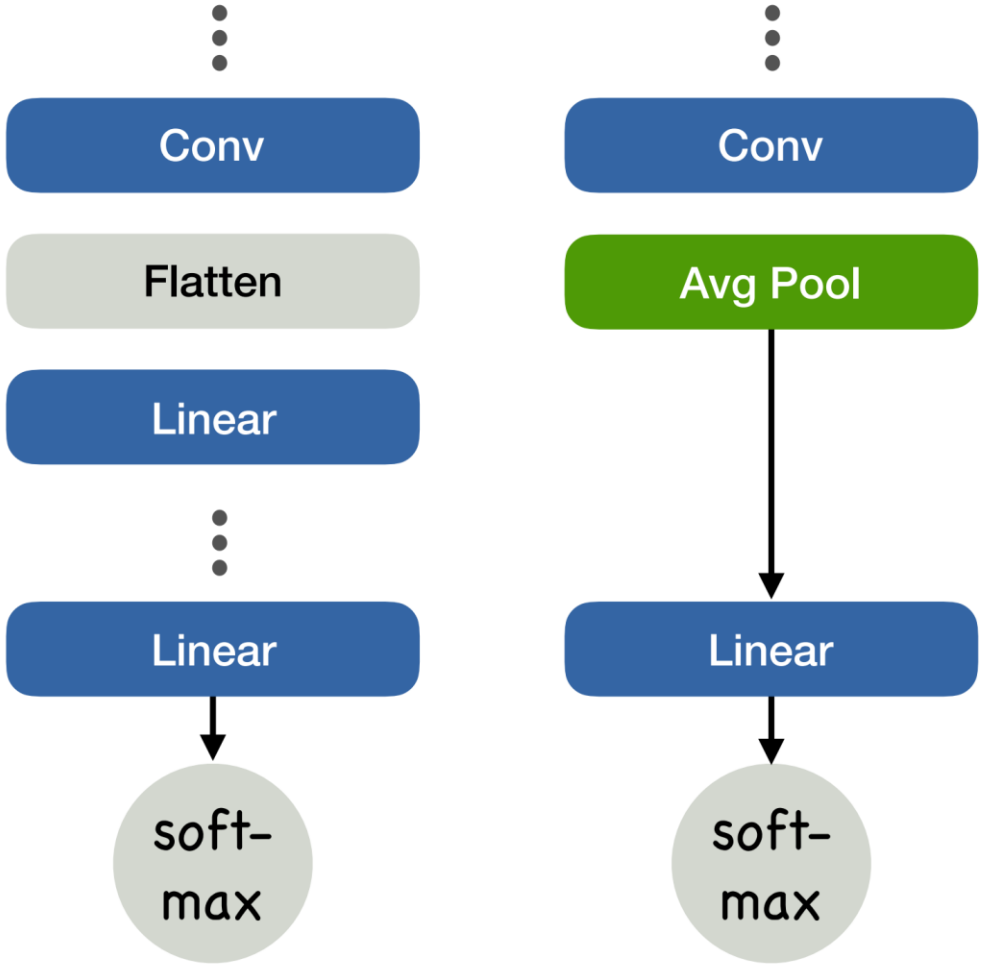
KEEP KERNELS SMALL



REPEAT PATTERNS

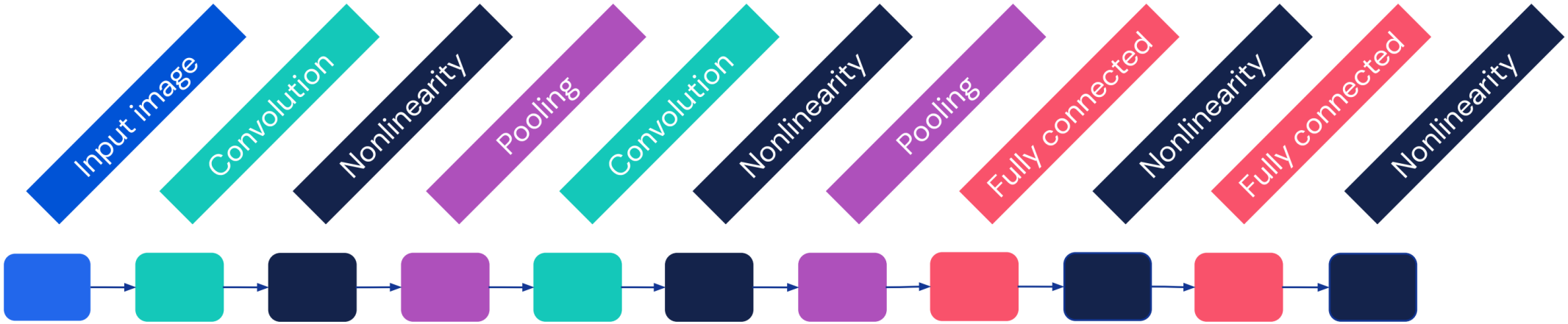


ALL CONVOLUTIONAL

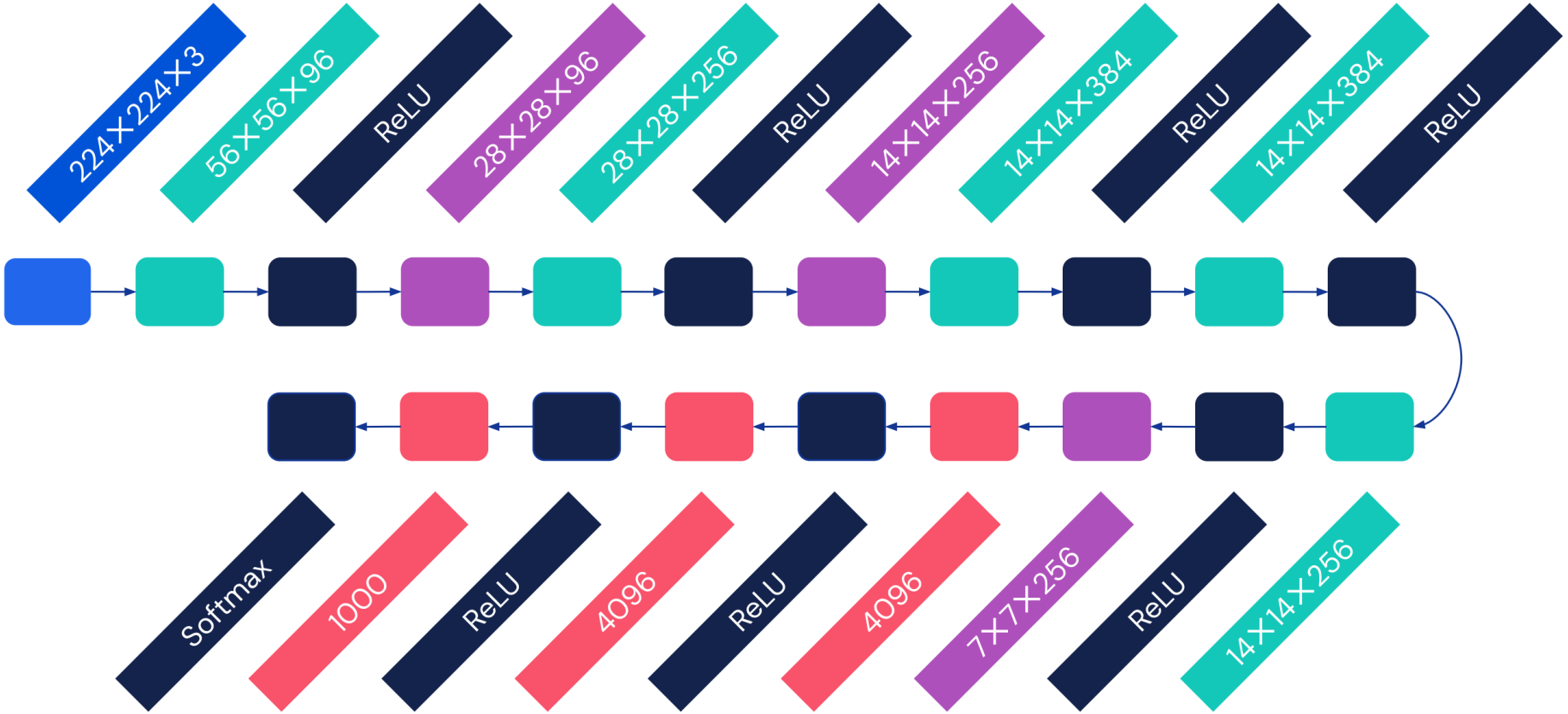


6 – MODEL EXAMPLES

LENET-5

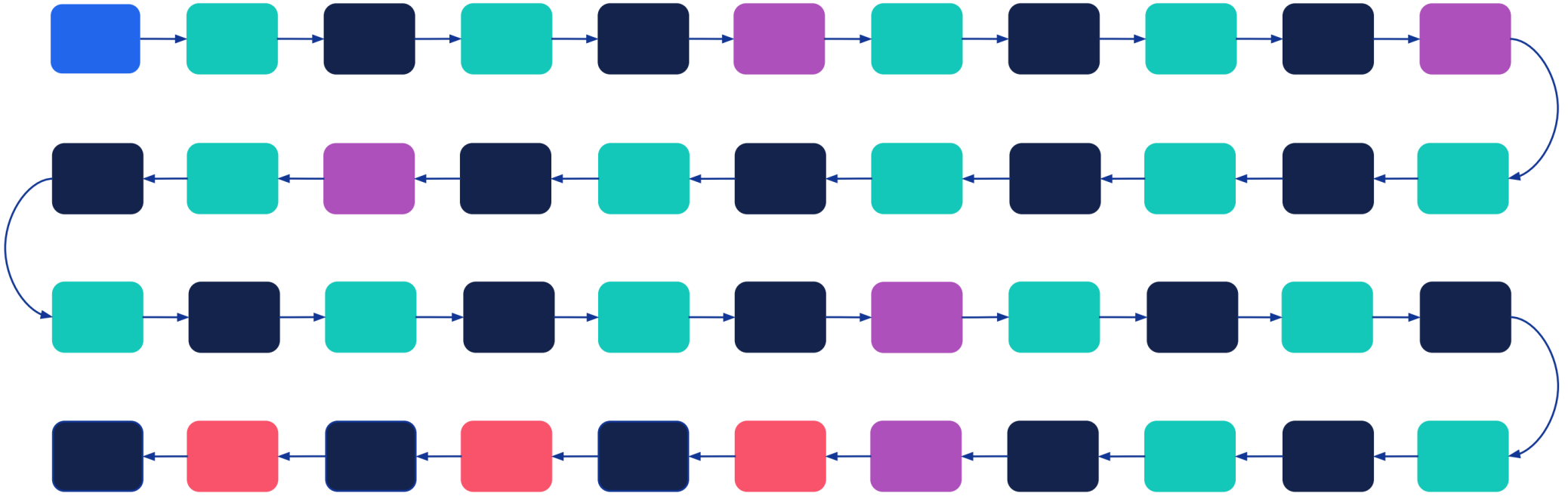

Input image
Convolution
Nonlinearity
Pooling
Convolution
Nonlinearity
Pooling
Fully connected
Nonlinearity
Fully connected
Nonlinearity

ALEXNET

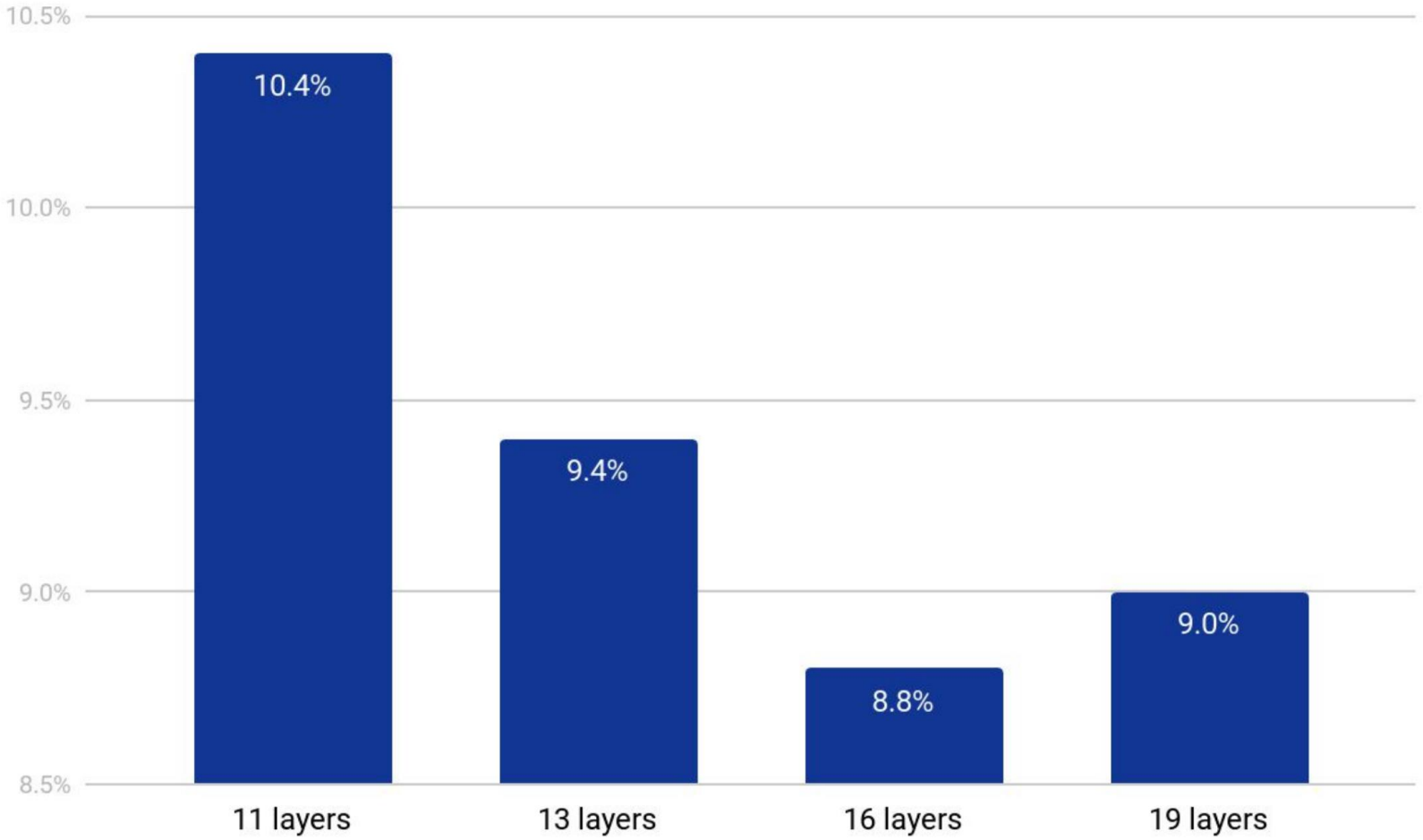

224X224X3
56X56X96
ReLU
28X28X96
28X28X256
ReLU
14X14X256
14X14X384
ReLU
14X14X384
ReLU
Softmax
1000
ReLU
4096
ReLU
4096
7X7X256
ReLU
14X14X256

7 - DEPTH

VGGNET (2014): BUILDING VERY DEEP CONVNETS



VGGNET (2014): ERROR PLATEAUS AFTER 16 LAYERS



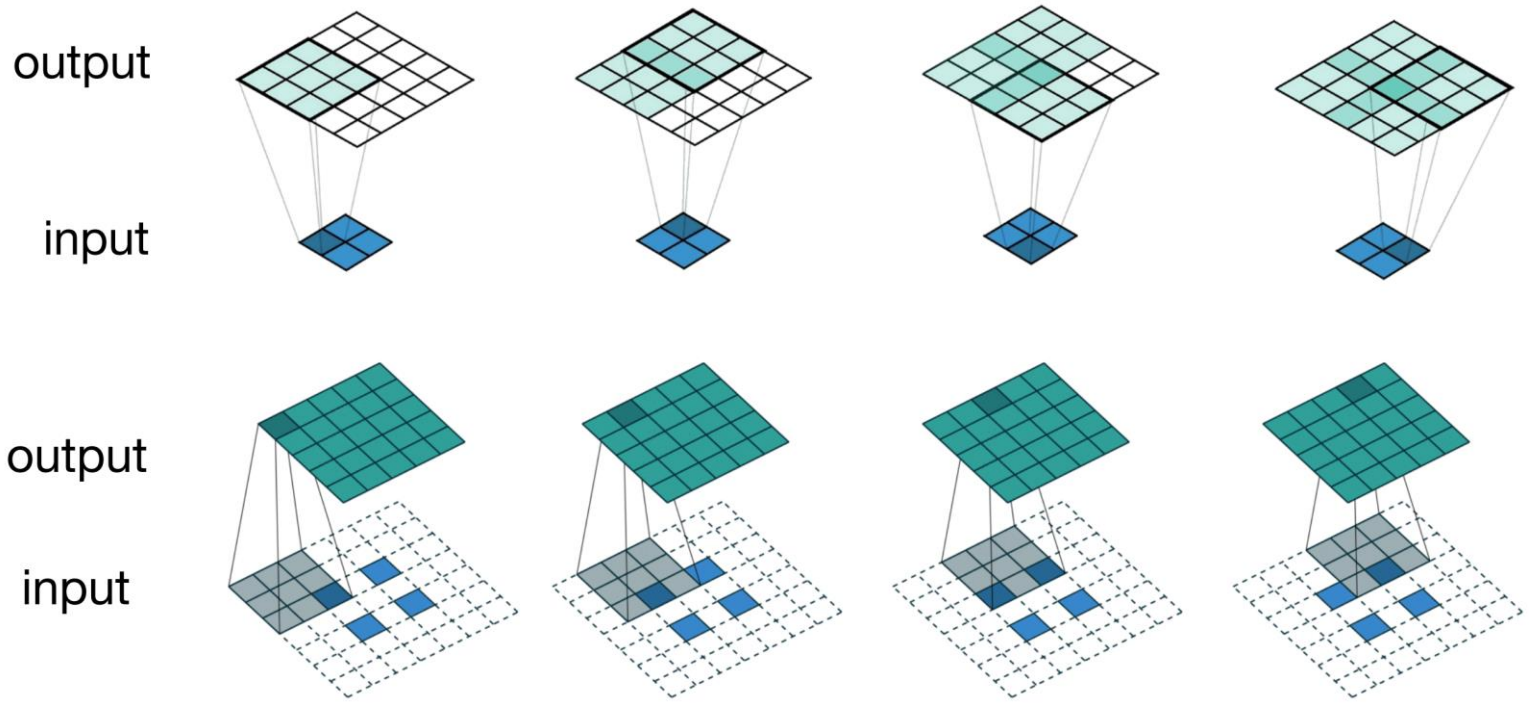
DATA AUGMENTATION

By design, convnets are only robust against **translation**

Data augmentation makes them robust against other transformations: rotation, scaling, shearing, warping, ...

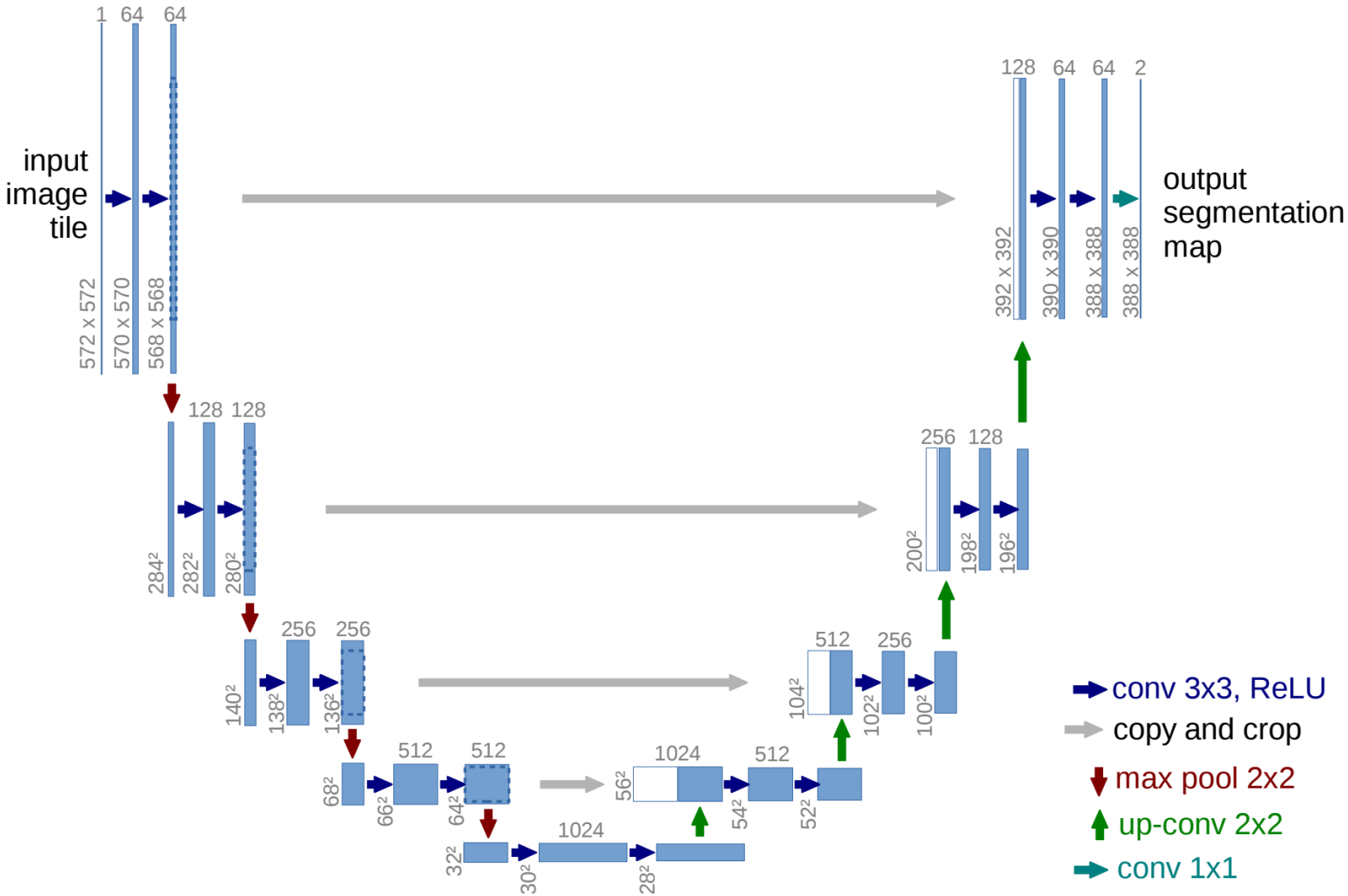
8 – UP-CONVOLUTIONS

UP-CONVOLUTION



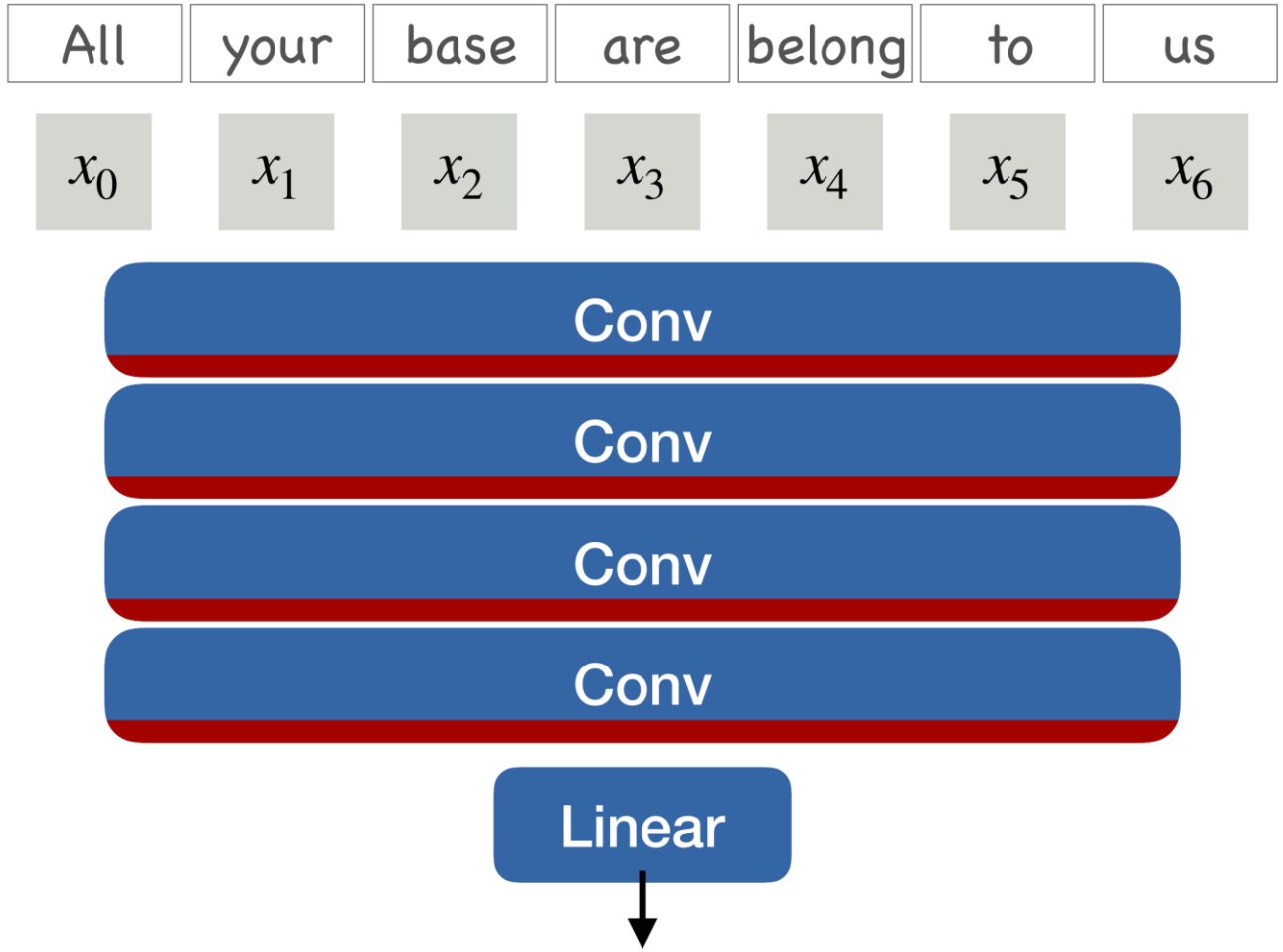

```
model: torch.nn.Module = torch.nn.ConvTranspose2d(16, 33, 3)
input: torch.Tensor = torch.randn(20, 16, 50, 100)
output: torch.Tensor = model(input)
```

U-NET


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

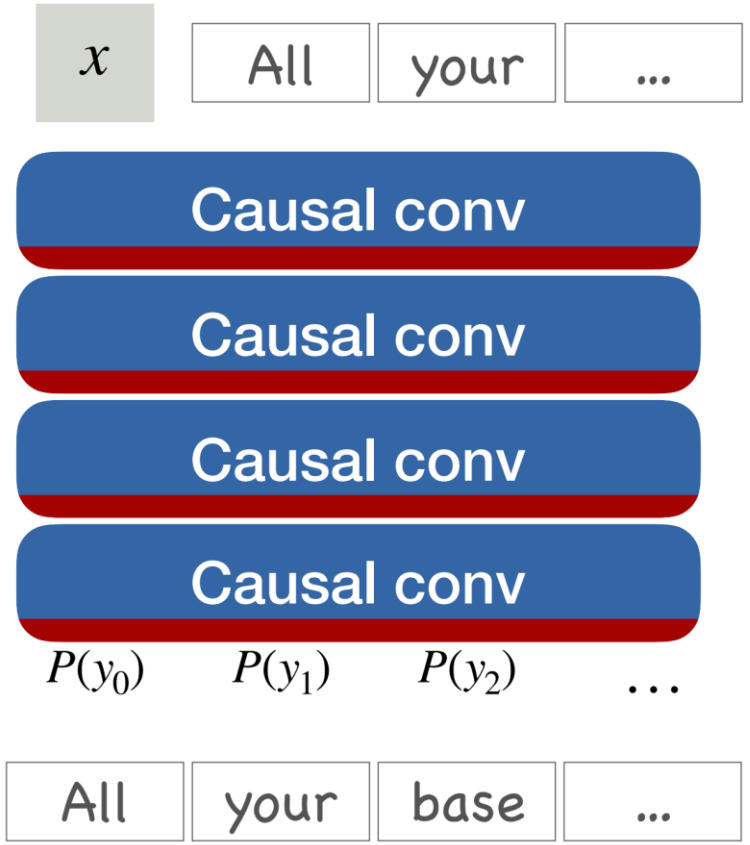
9 - OTHER DIMENSIONS

1D - TEMPORAL CONVOLUTIONS

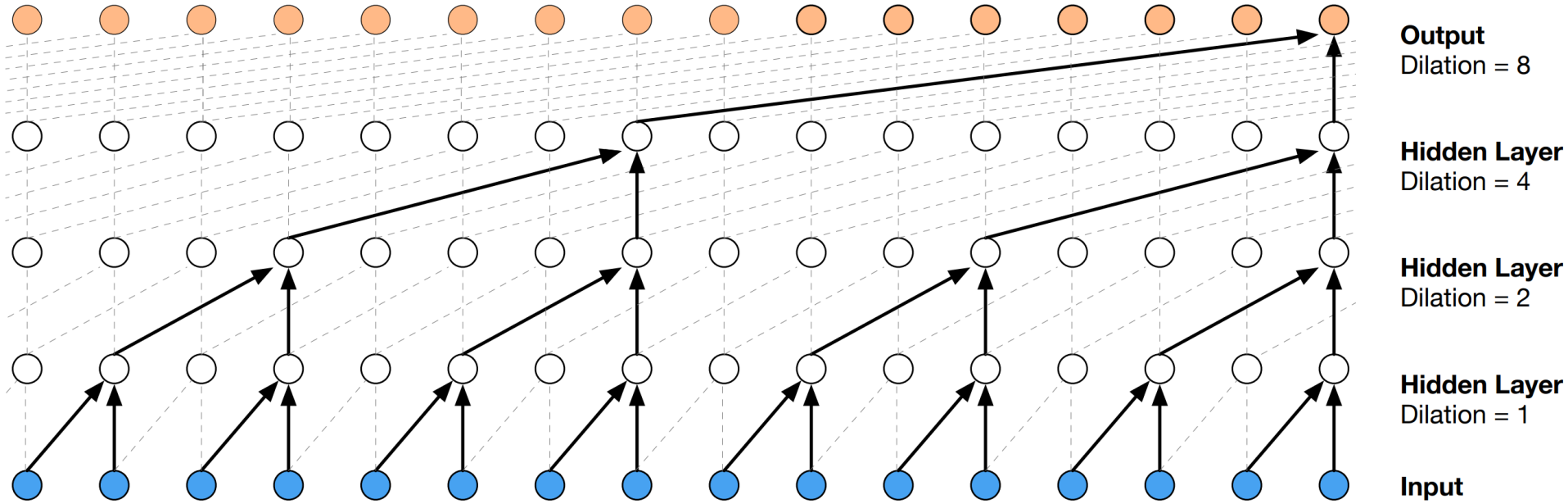


1D - CAUSAL CONVOLUTION

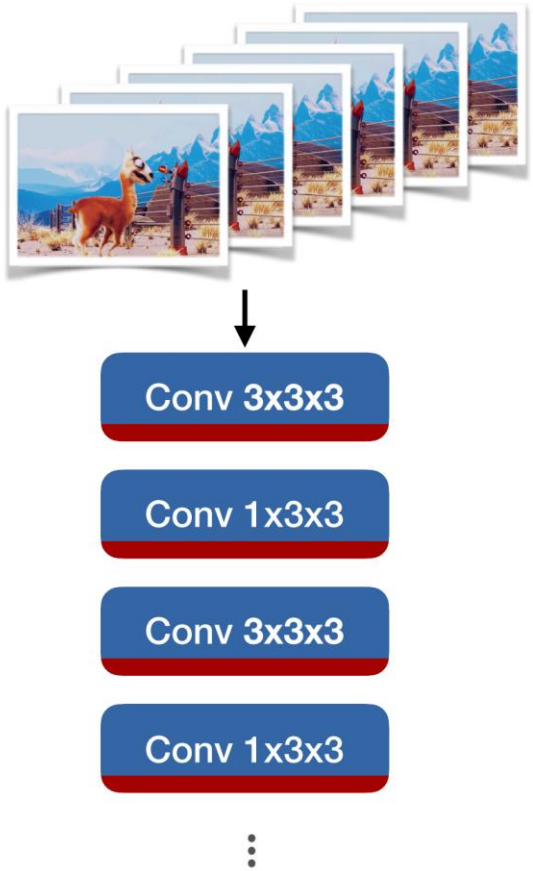
$$\mathbf{Z}_{t,b} = \mathbf{b}_c + \sum_{i=0}^{w-1} \sum_{j=0}^{C_1-1} \mathbf{X}_{t+i-w,b+j} \mathbf{w}_{i,j}$$



1D - WAVENET


Output
Dilation = 8
Hidden Layer
Dilation = 4
Hidden Layer
Dilation = 2
Hidden Layer
Dilation = 1
Input

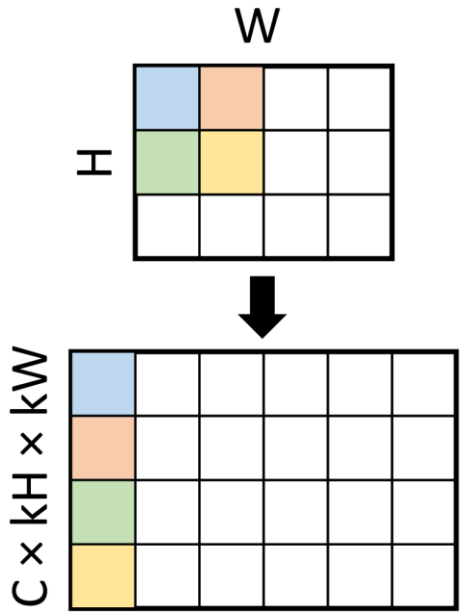
3D – TEMPORAL CONVOLUTION



9 - FOLD & UNFOLD

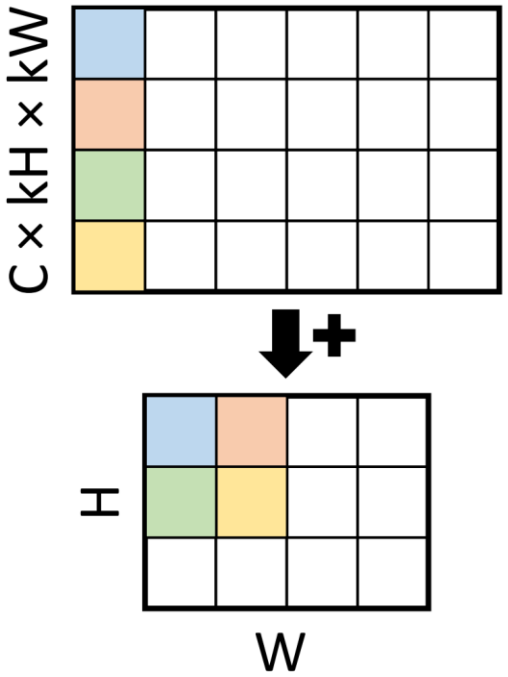
UNFOLD

- Receives a batch of images:
  - Image shape: N, C, H, W
- Extract blocks:
  - Block size: C × kH × kW
  - Num of block per image: H’ × W’
- Return a batch of blocks:
  - Output shape: N, C × kH × kW, H’ × W’



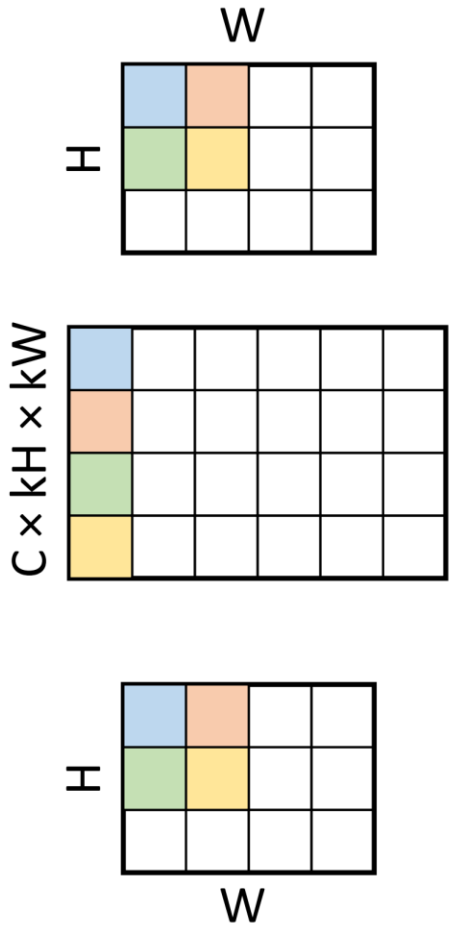
FOLD

- The opposite of `F.unfold`.
- Receives `blocks`.
- Receives `output_size`.
  - Creates an image `output` of that size.
  - Initialize `output` with zeros.
- Iterates over the blocks.
  - Adds each element to its place in `output`.



FOLD & UNFOLD

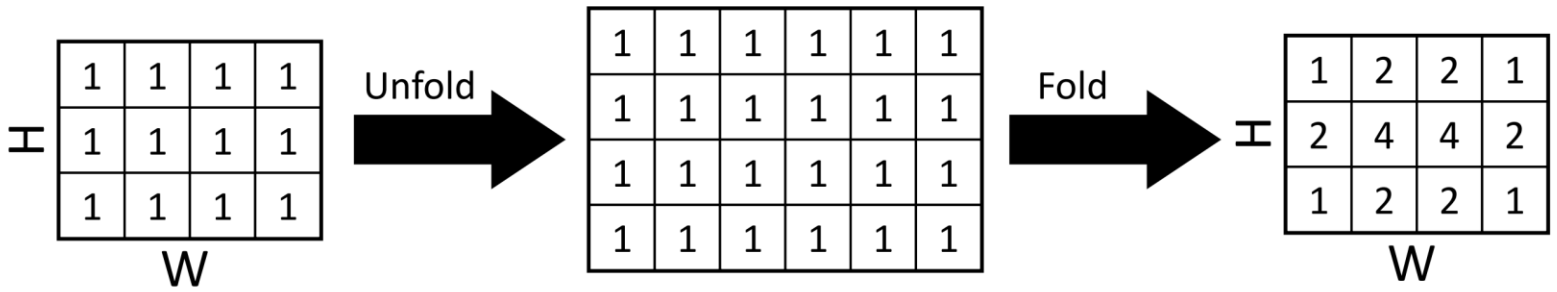
- Extract blocks from an image:
  - `F.unfold` (also called “im2col”).
- Apply an operation on each block:
  - Linear, max pooling, etc.
- Combine blocks into an image:
  - `F.fold` (also called “col2im”).

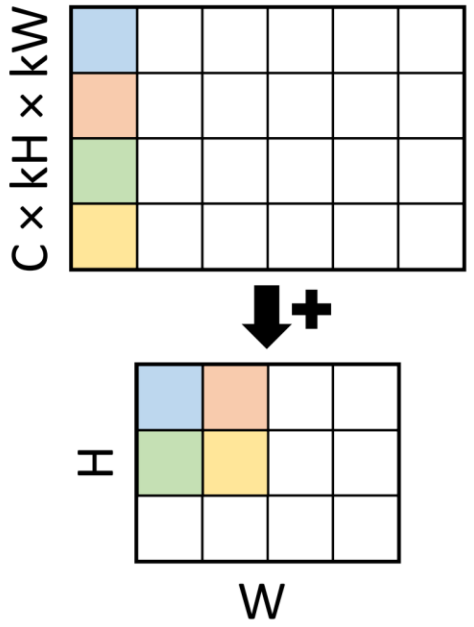


BLOCKS OVERLAPPING

```
blocks = torch.rand(size=(2, 1 * 2 * 2, 2 * 3))
output_size = (3, 4)

# fold back into an image
output = F.fold(blocks, output_size, kernel_size=2)
```





UNFOLD PARAMETERS

- **kernel_size** (*int or tuple*) – the size of the sliding blocks
- **dilation** (*int or tuple, optional*) – a parameter that controls the stride of elements within the neighborhood. Default: 1
- **padding** (*int or tuple, optional*) – implicit zero padding to be added on both sides of input. Default: 0
- **stride** (*int or tuple, optional*) – the stride of the sliding blocks in the input spatial dimensions. Default: 1

FOLD PARAMETERS

- **output_size** (*int or tuple*) – the shape of the spatial dimensions of the output (i.e., `output.sizes()[2:]`)
- **kernel_size** (*int or tuple*) – the size of the sliding blocks
- **dilation** (*int or tuple, optional*) – a parameter that controls the stride of elements within the neighborhood. Default: 1
- **padding** (*int or tuple, optional*) – implicit zero padding to be added on both sides of input. Default: 0
- **stride** (*int or tuple*) – the stride of the sliding blocks in the input spatial dimensions. Default: 1

REFERENCES

- https://www.philkr.net/cs342/material
- https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L3%20-%20UUCLxDeepMind%20DL2020.pdf
- https://arxiv.org/abs/1505.04597
- https://arxiv.org/abs/1609.03499v2
- https://dl4cv.github.io/DL4CV_Winter24/index.html