

Deep Convolutional Neural Networks

Ray Jones & Verena Kaynig, IACS

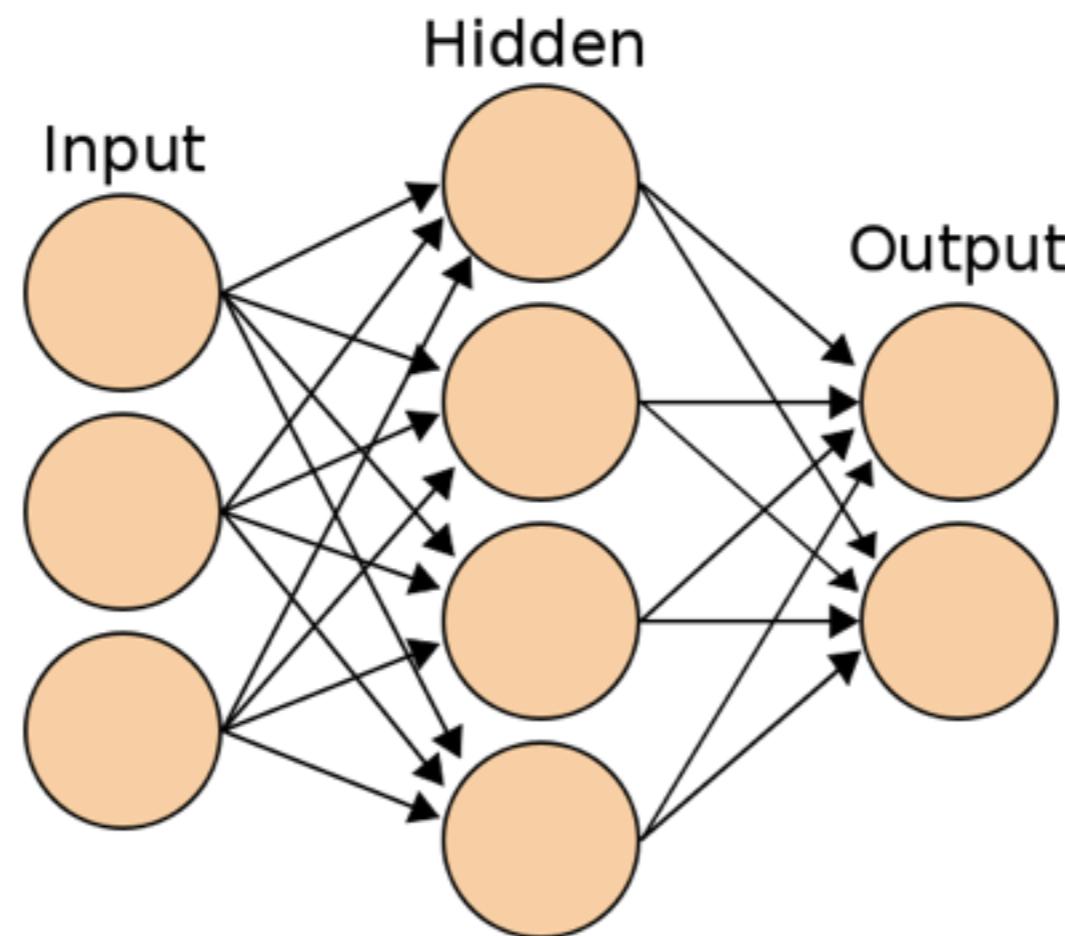
https://github.com/vkaynig/IACS_ComputeFest_DeepLearning

<https://goo.gl/hfsl0u>

Typical Feedforward NN

Series of matrix multiplies followed by nonlinear transforms:

$$\mathbf{x}_{i+1} = \sigma_i(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)$$



Typical Feedforward NN

Series of matrix multiplies followed by nonlinear transforms:

$$\mathbf{x}_{i+1} = \sigma_i(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)$$

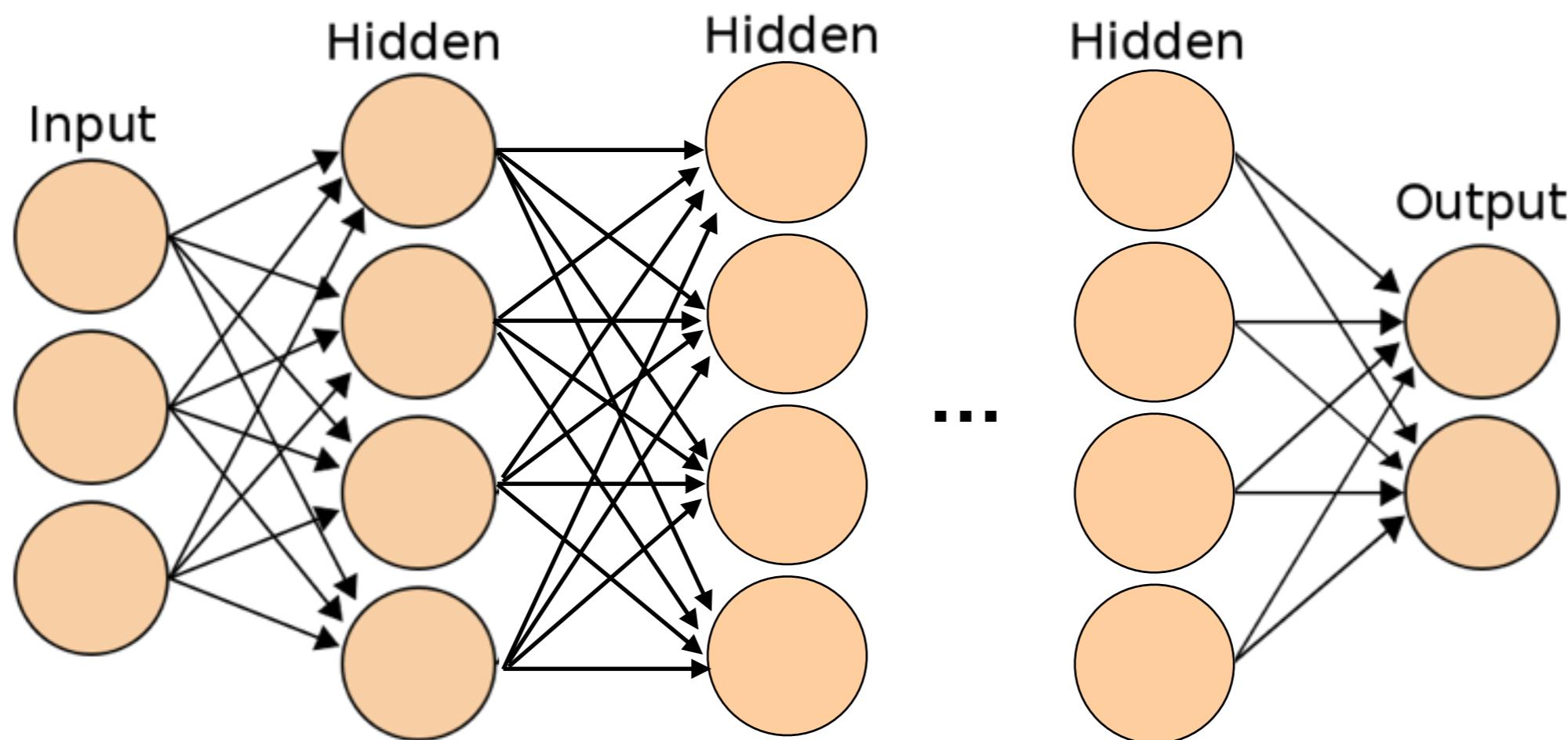
\mathbf{x}_i = input data at layer i

σ = nonlinearity at layer i

\mathbf{W}_i = weight matrix at layer i

\mathbf{b}_i = bias at layer i

Deep NN

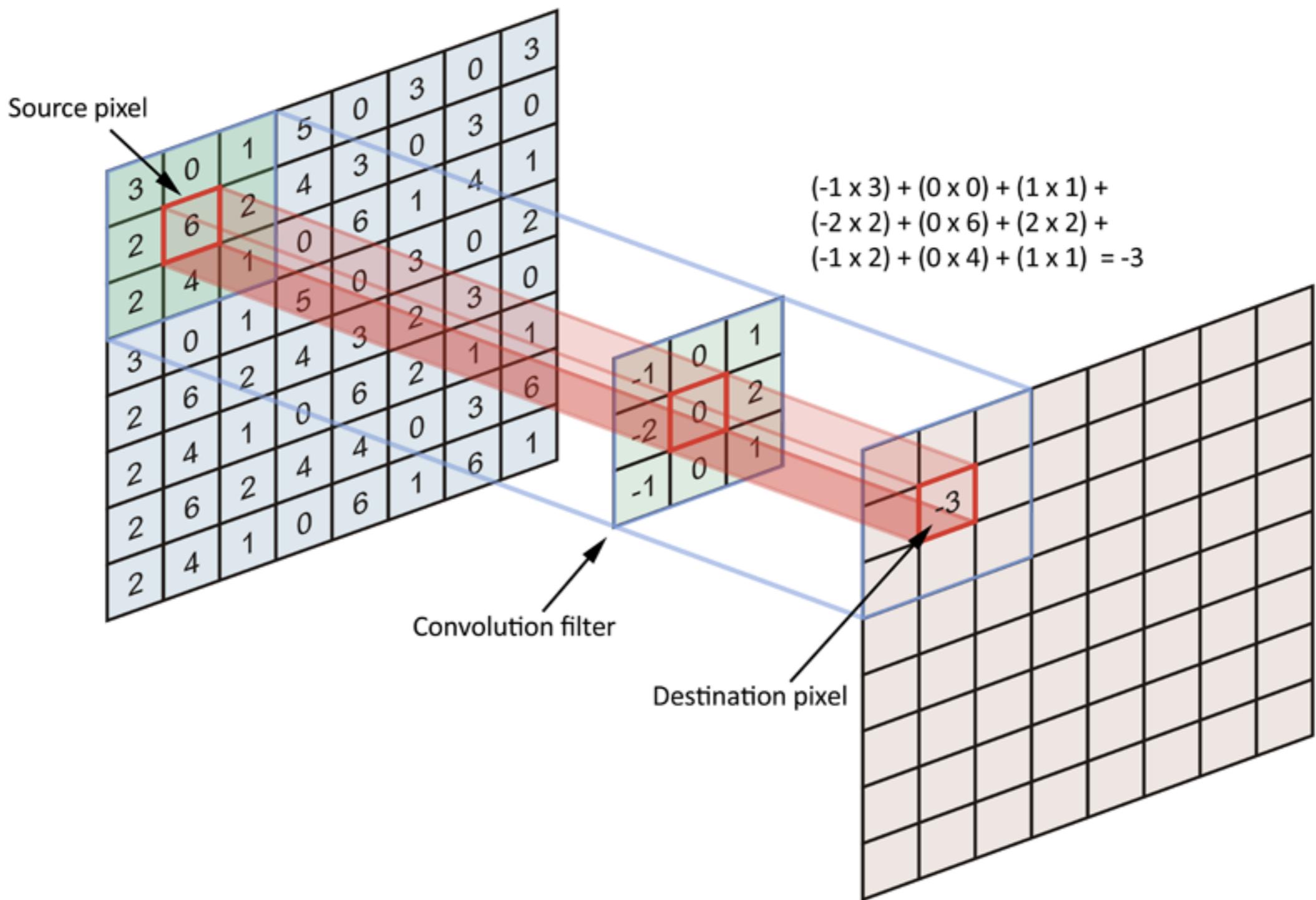


$$\mathbf{x}_{i+1} = \sigma_i(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)$$

Deep vs. Convolutional

- Deep:
 - More powerful than shallow networks.
 - Increased parameters = harder to train & evaluate.
- Convolutional:
 - Uses sparse & structured **W** between layers.
 - reduced training and evaluation cost.
- Origins in Computer Vision
 - Large input space (images)
 - Structured input, which we can take advantage of.

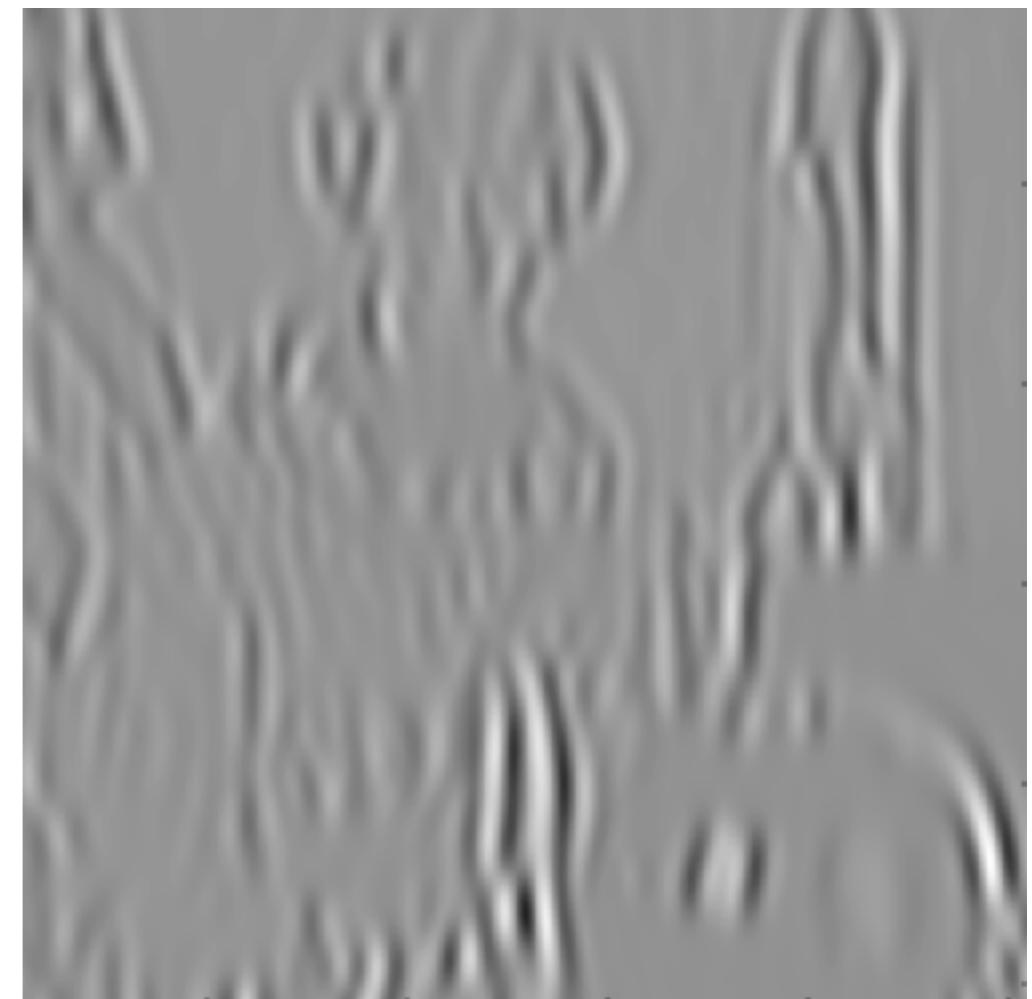
What is a Convolution



2D Convolution



$$\ast \quad \begin{matrix} & & \\ & \text{---} & \\ & & \end{matrix} \quad =$$



Vision works with High Dimensional Inputs

- A typical image will be several megapixels.
- $1K \times 1K$ image => 1 million inputs
 - untenable to train a fully connected network
- Fortunately, images have some useful properties.

Images are Structured, Local, and Hierarchical



Nearby pixels are more strongly related than distant ones.

Objects are built up out of smaller parts.

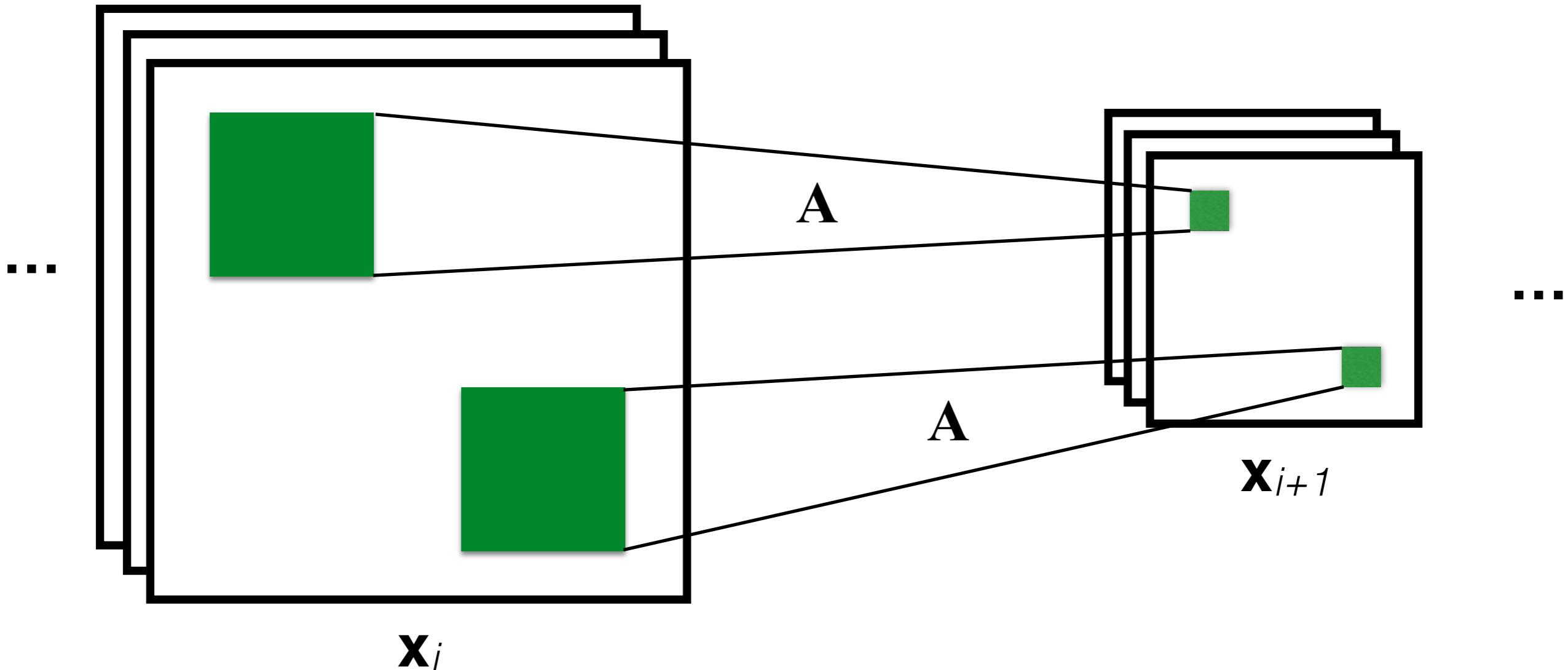
Images are Invariant



CNNs & W for Vision

- Local Connectivity (locality)
- Parameter Sharing (translation invariant)
- Low-level features are learned (structure)
- Pooling (hierarchical)

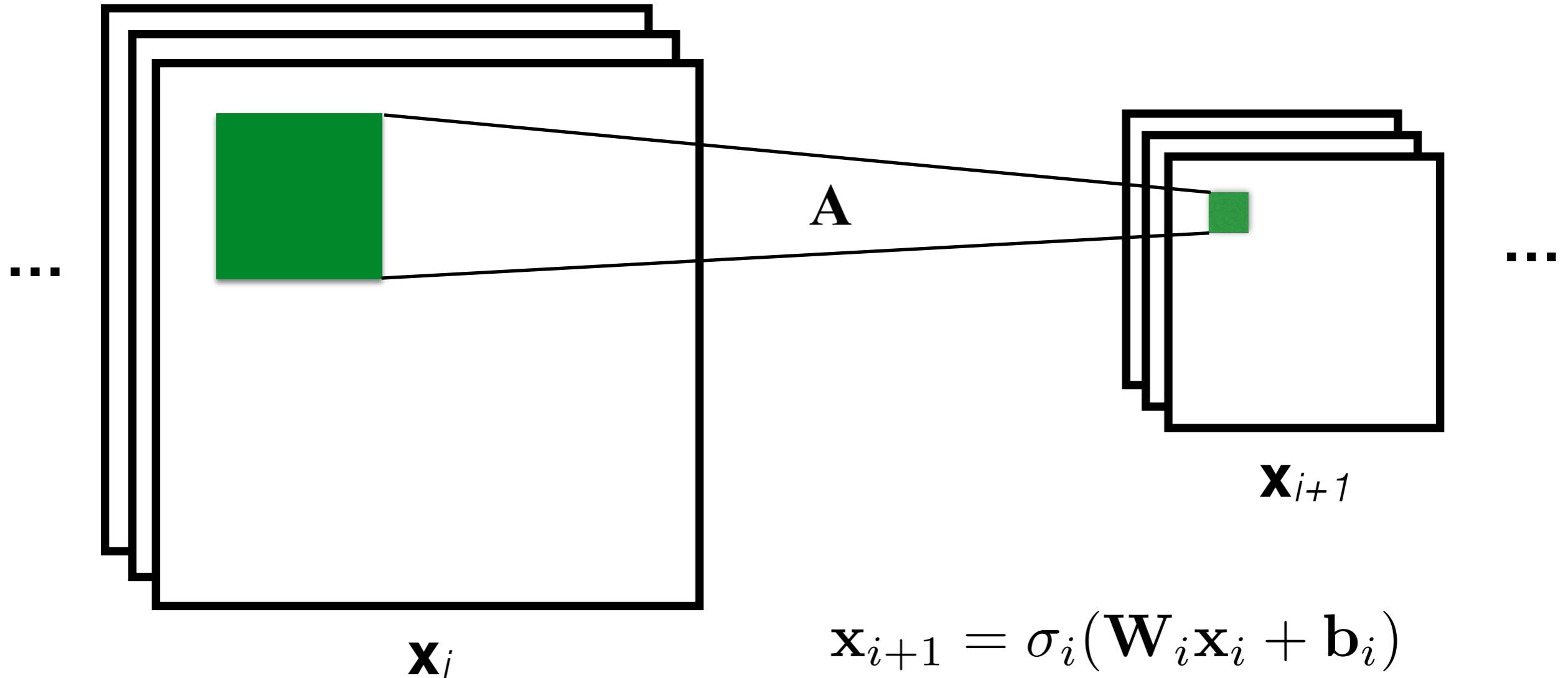
Convolutional NN



Convolution \leftrightarrow Same \mathbf{A} at different locations.

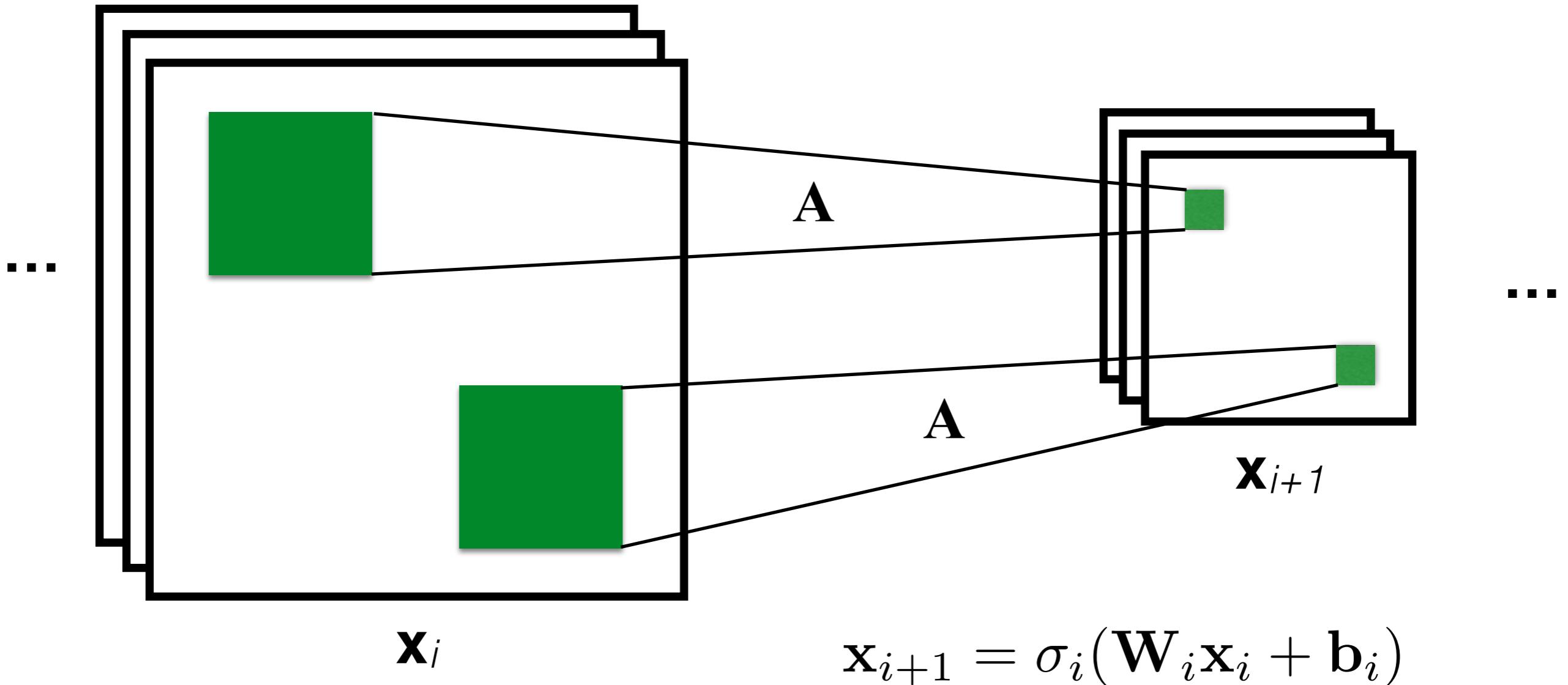
\mathbf{W} is made up of offset copies of \mathbf{A} (Toeplitz matrix)

Local Connectivity



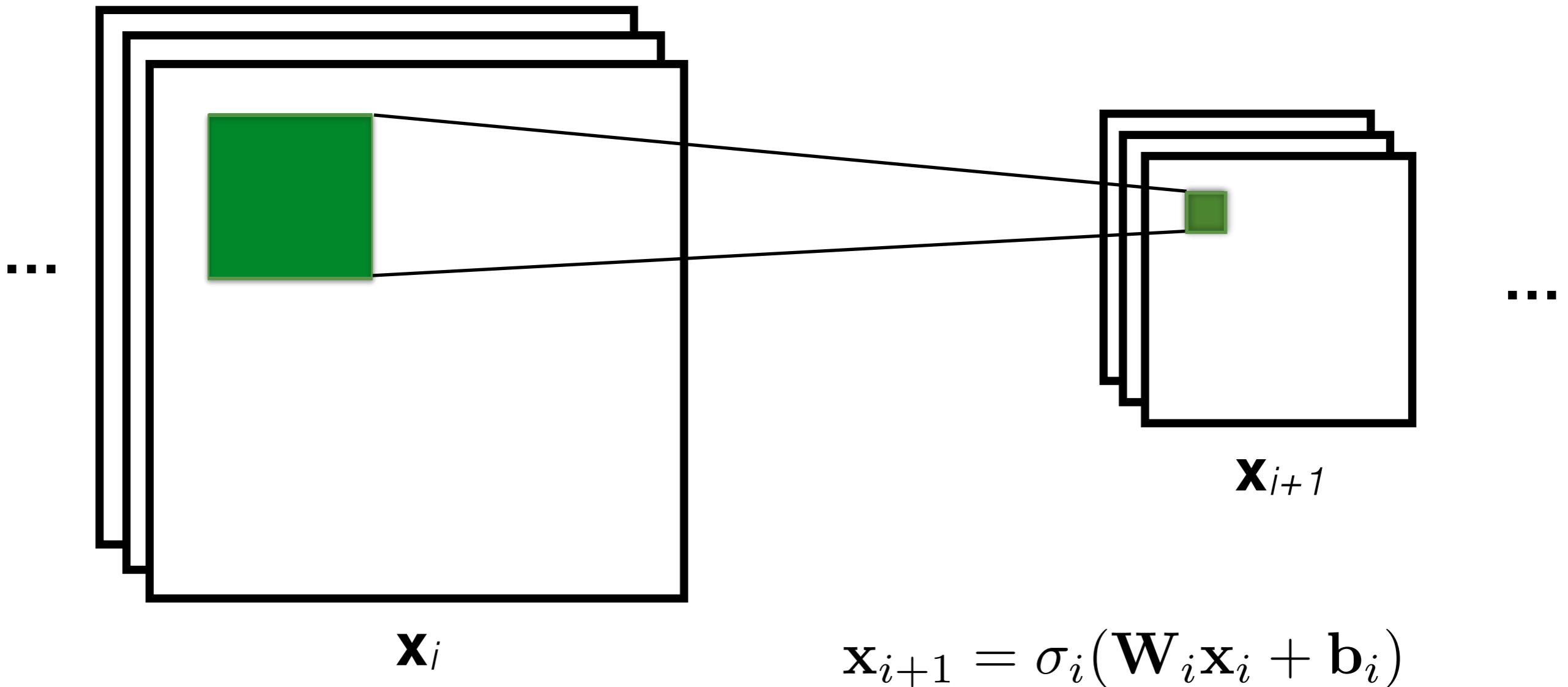
Each output has depends on only a small region of the input, so \mathbf{W} is sparse.

Parameter Sharing

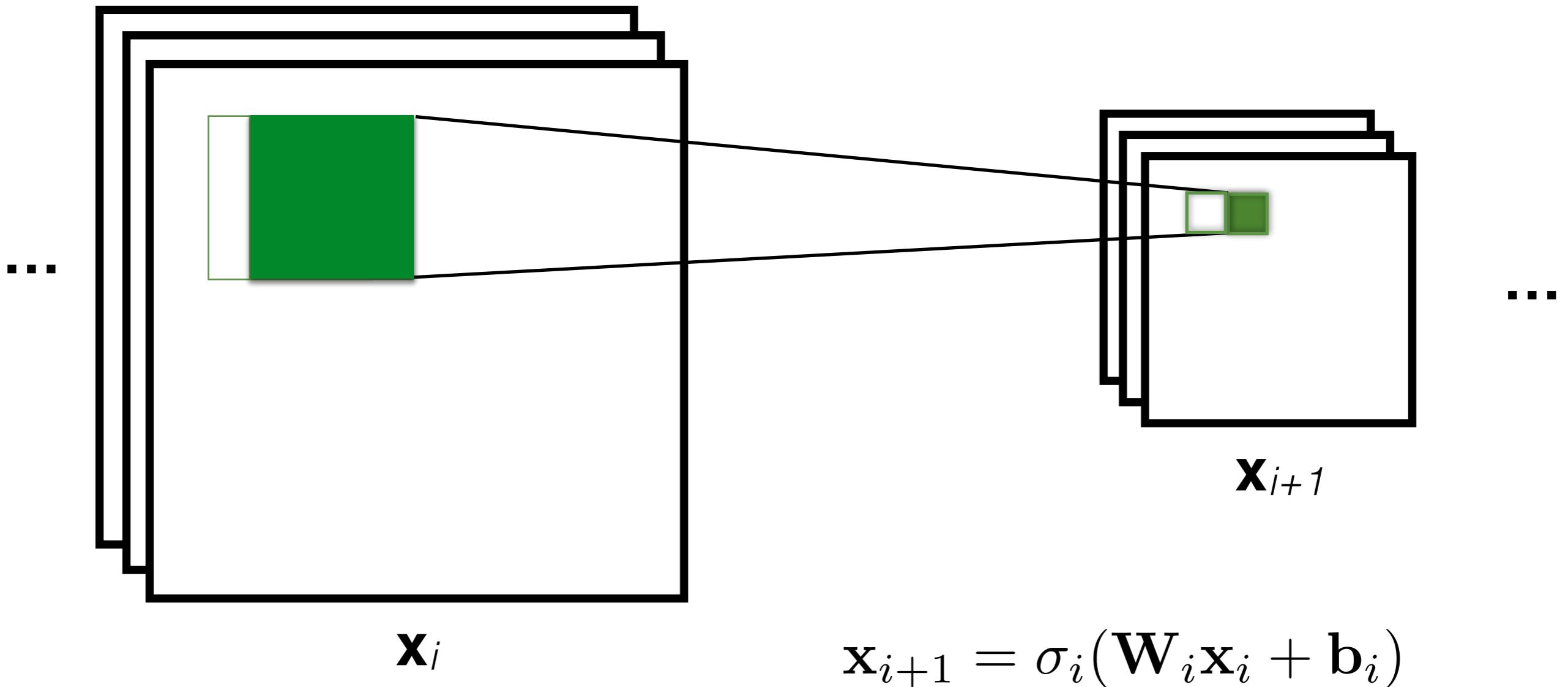


Each \mathbf{A} is the same, reducing free parameters in \mathbf{W} .

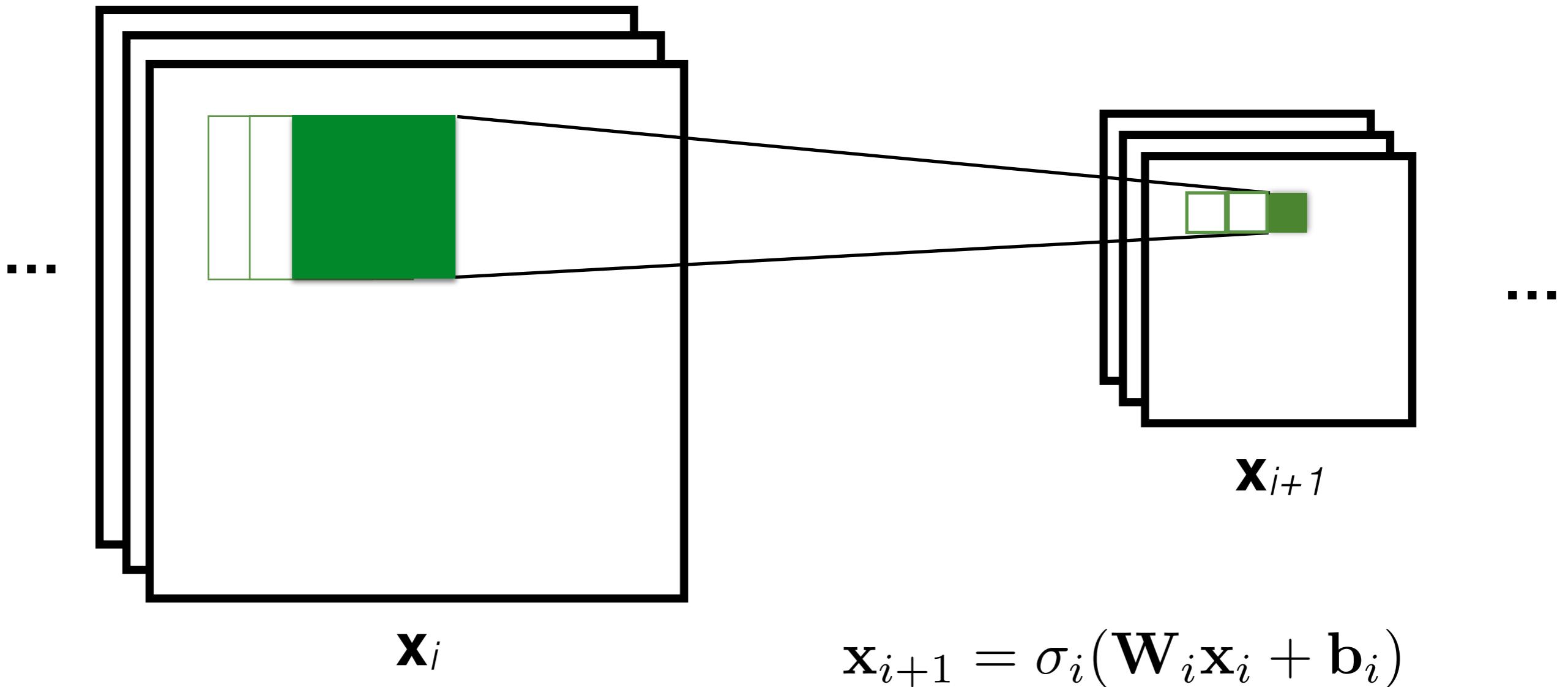
Sliding window



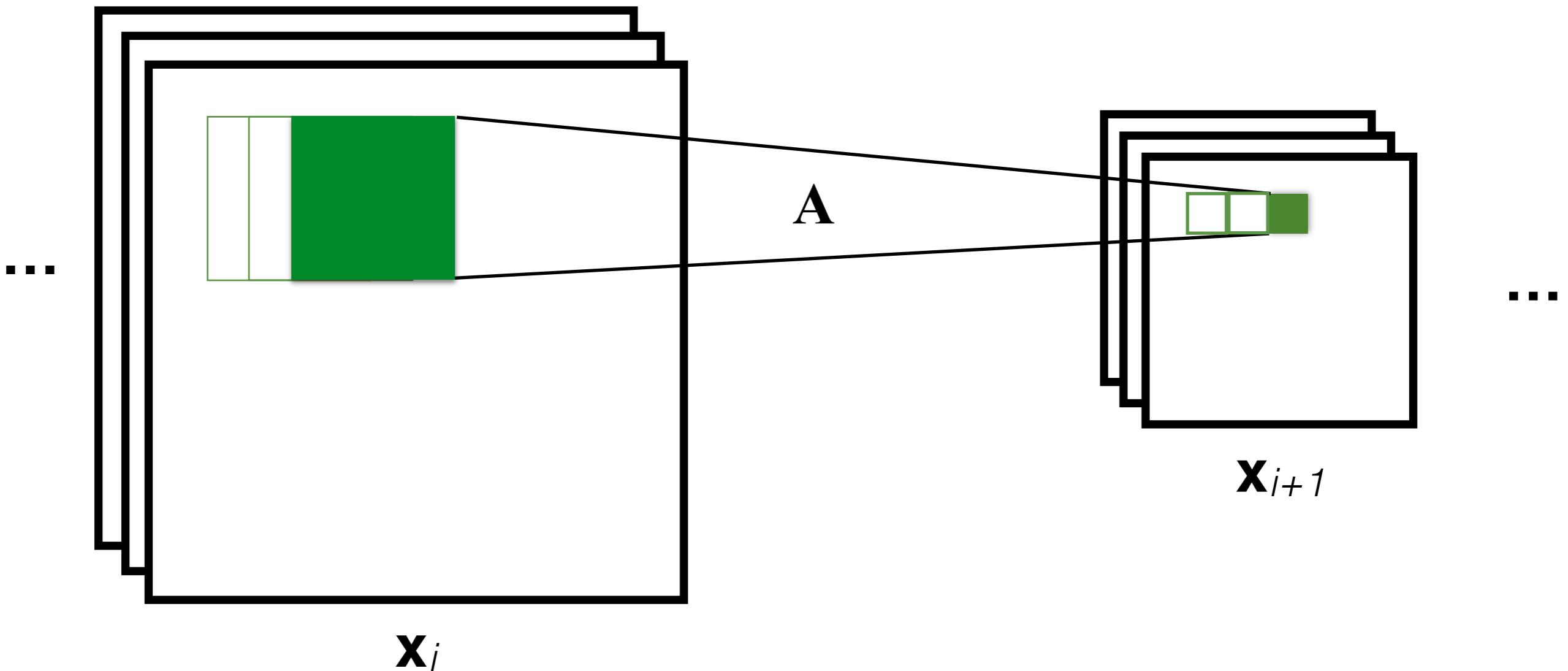
Sliding window



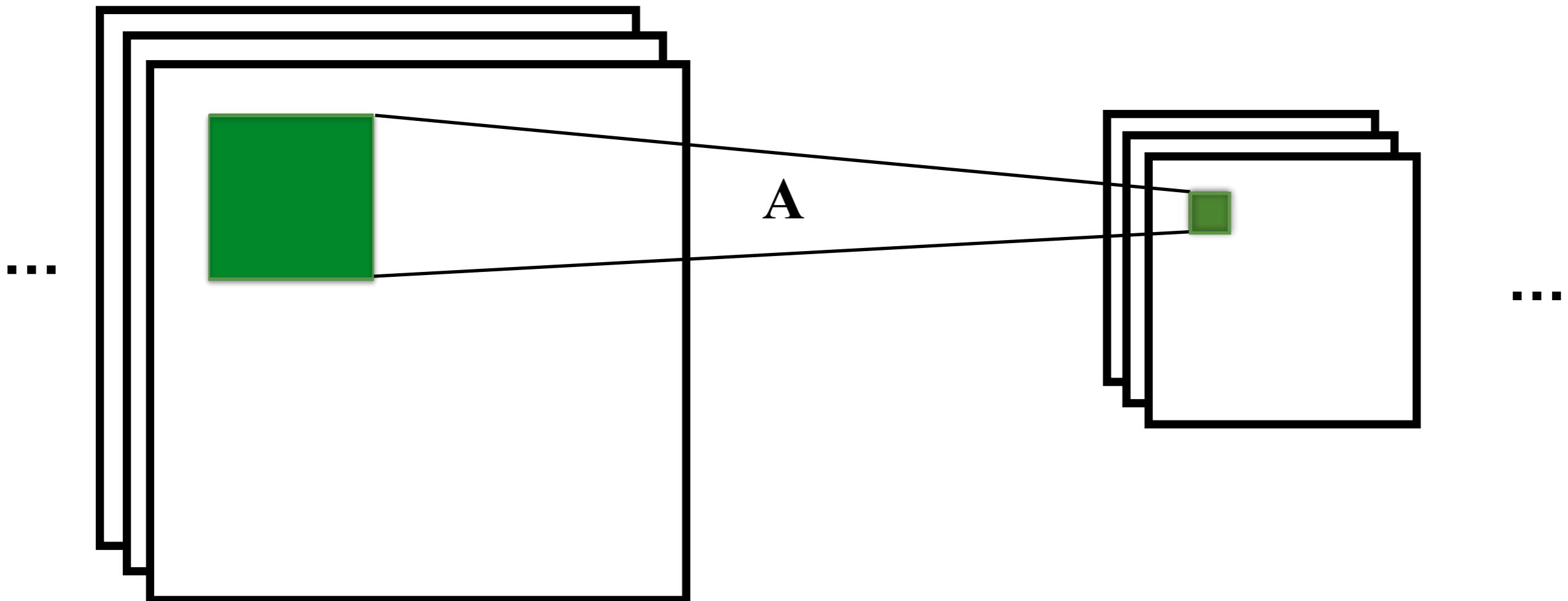
Sliding window



This is the same as the
Convolution of A with \mathbf{x}_i



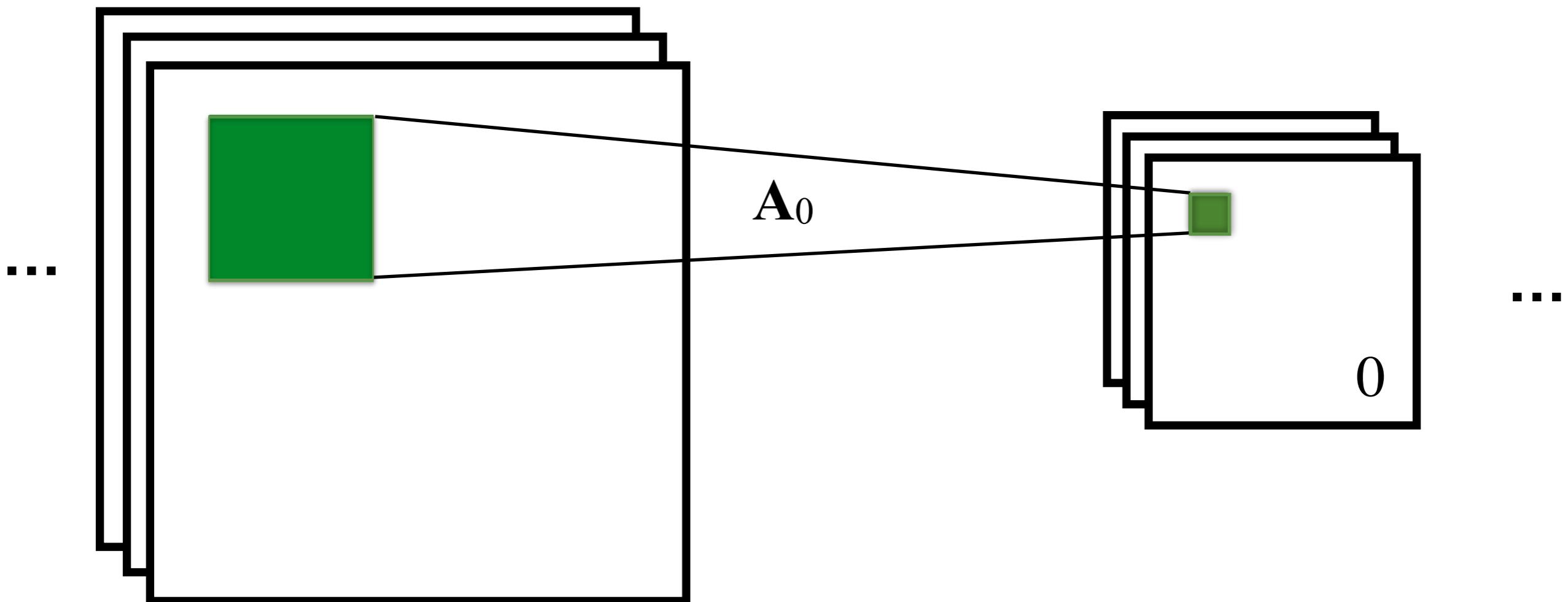
Input/output dimensions



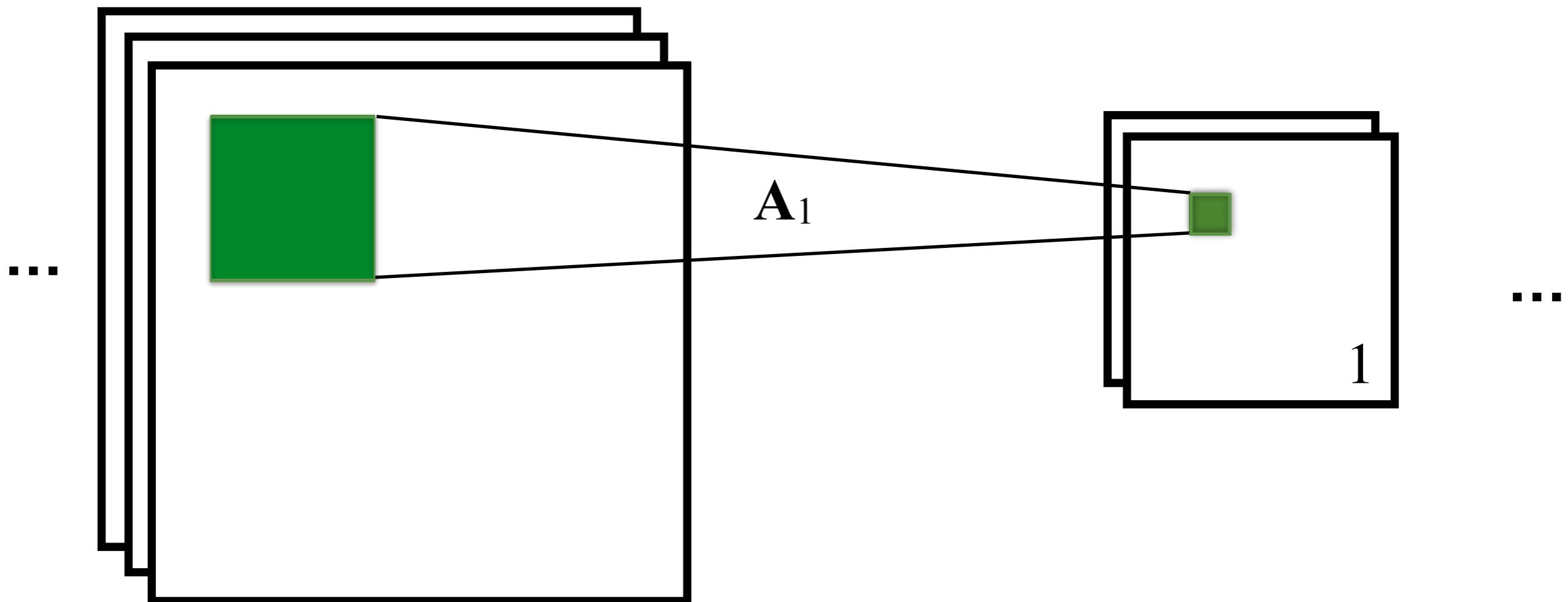
The input and output of each CNN layer may have several “channels”, aka “Feature Maps.”

Each output has its own **A**.

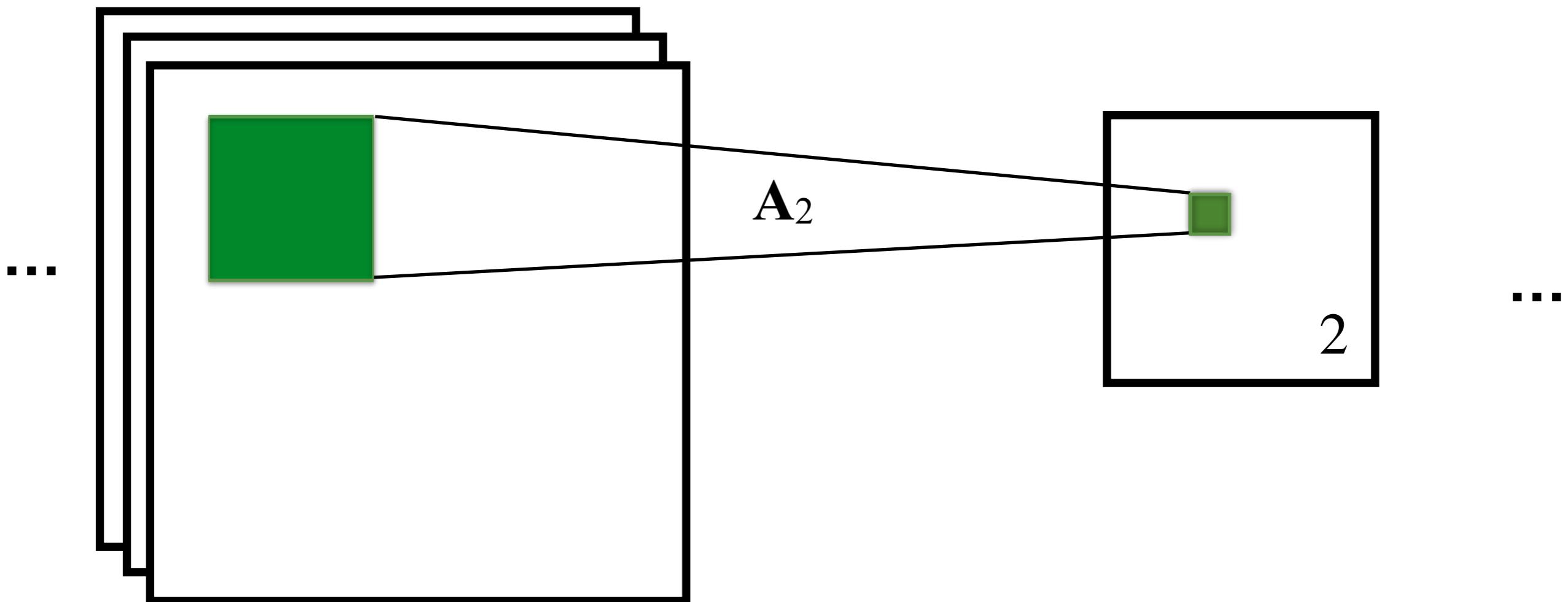
Input/output dimensions



Input/output dimensions

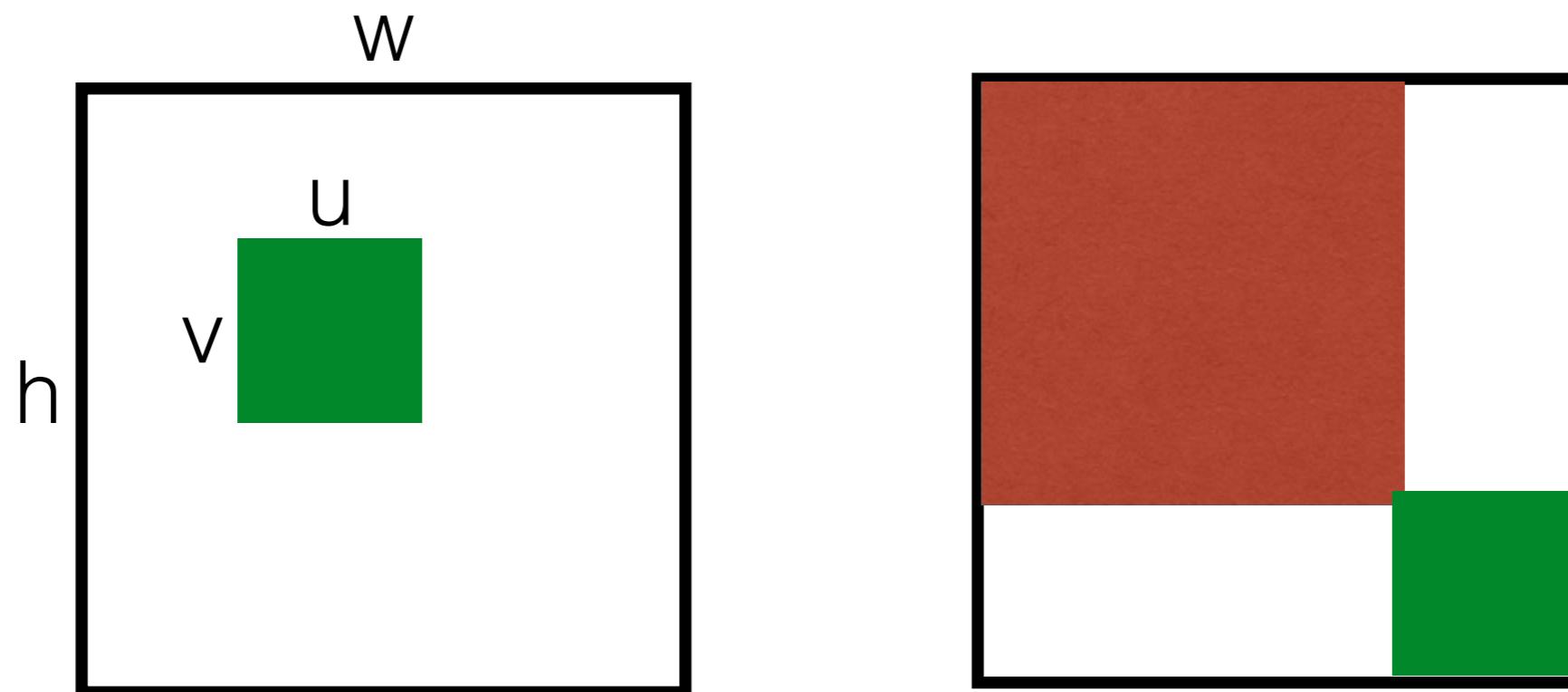


Input/output dimensions



But!

- Each **A** uses the entire depth of its input.
- If the input is $D \times w \times h$ and there are N filters of size $D \times u \times v$, the output will be of size:
$$N \times (w - u + 1) \times (h - v + 1)$$
- Sometimes easier just to "pad" with zeros to keep same size.

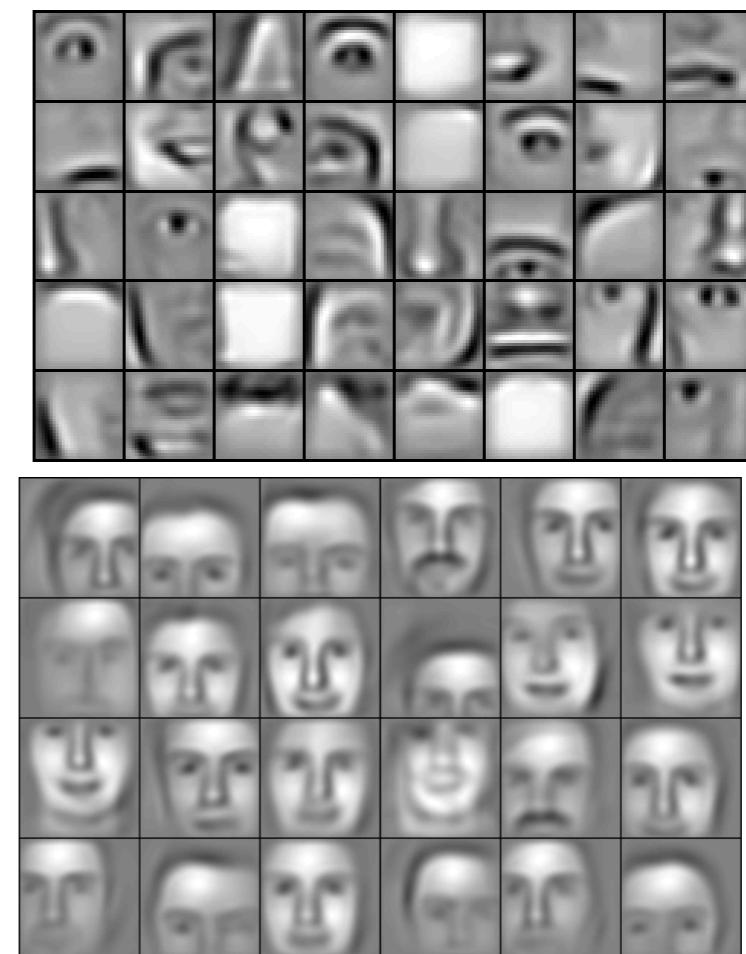
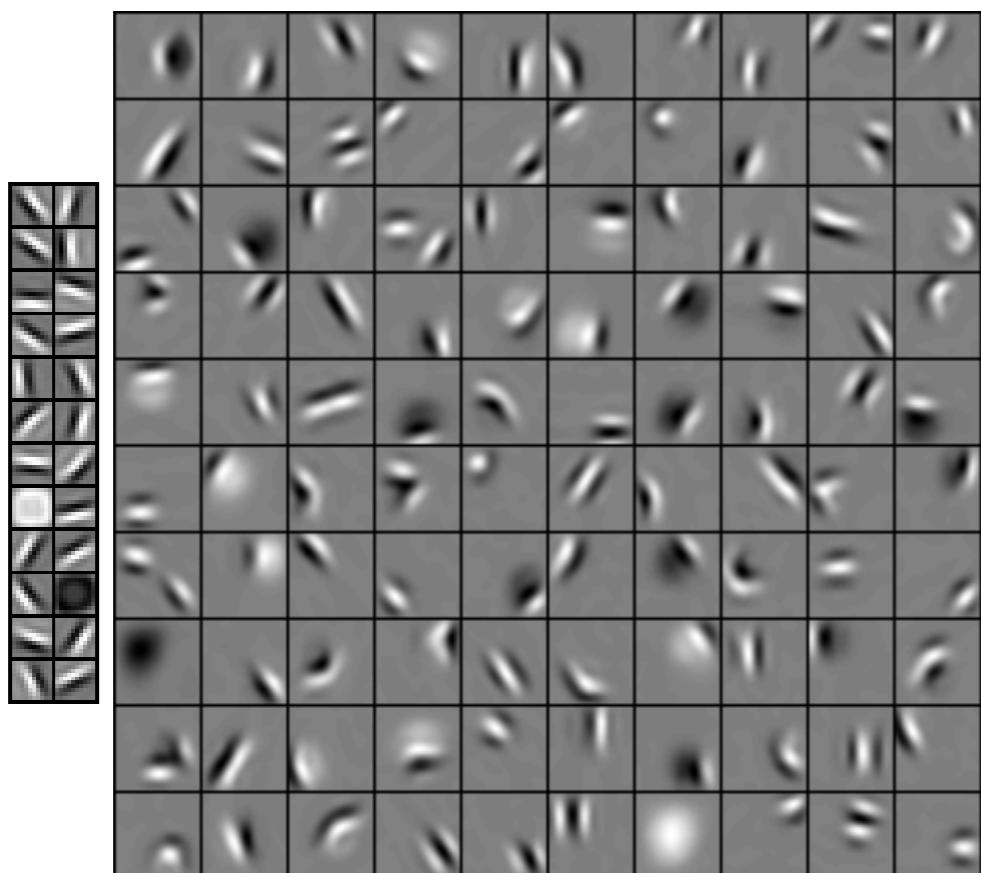


Benefits of using Convolutions

- Fewer parameters, less computation.
- Fast implementations.
- Locality preserving across layers.

Images are Hierarchical

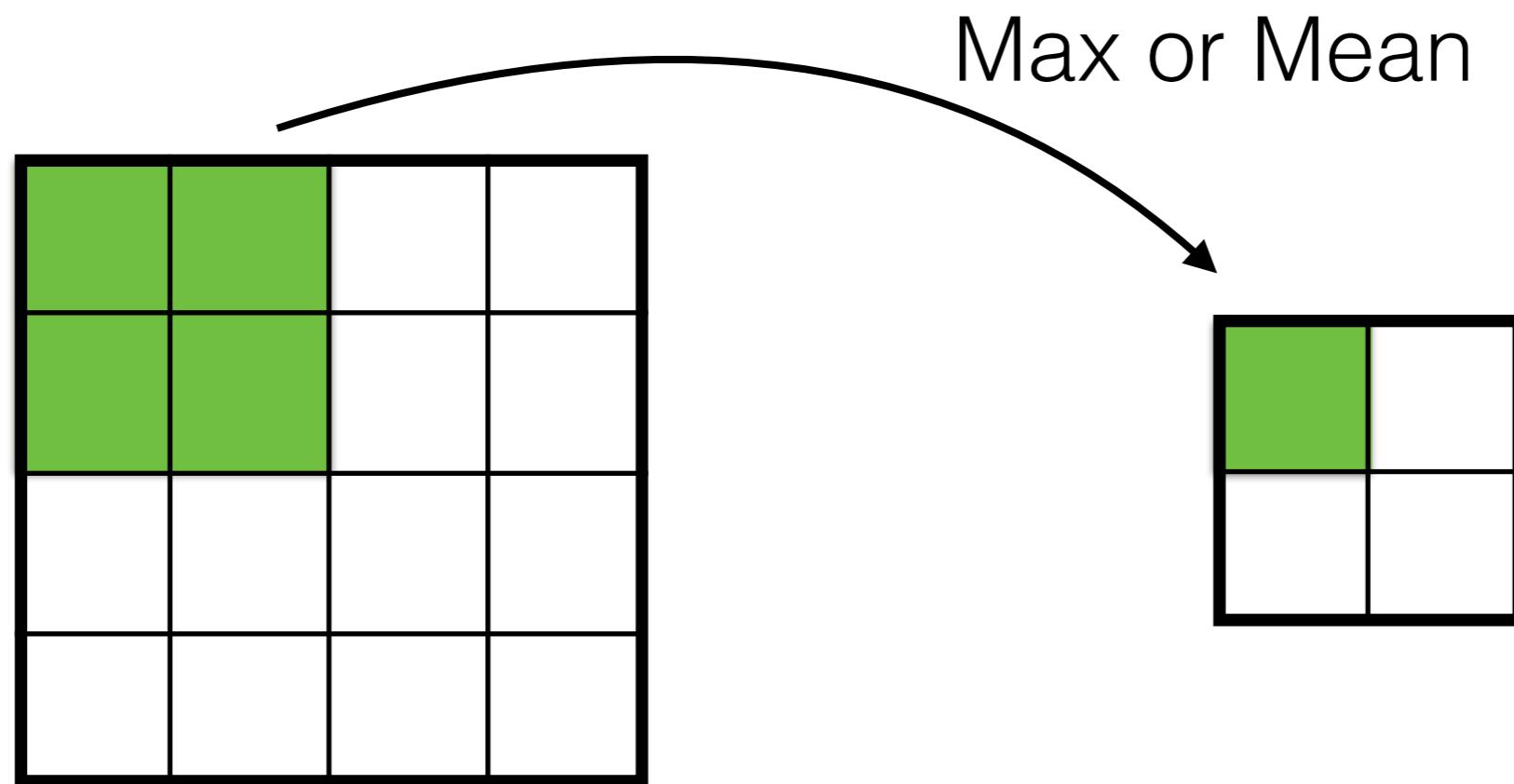
Big things are made up of smaller parts.



from Lee et al., *ICML*, 2009

Pooling

Combine small local regions into single output.
Adds hierarchy, translation-invariance, nonlinearity,
data reduction.



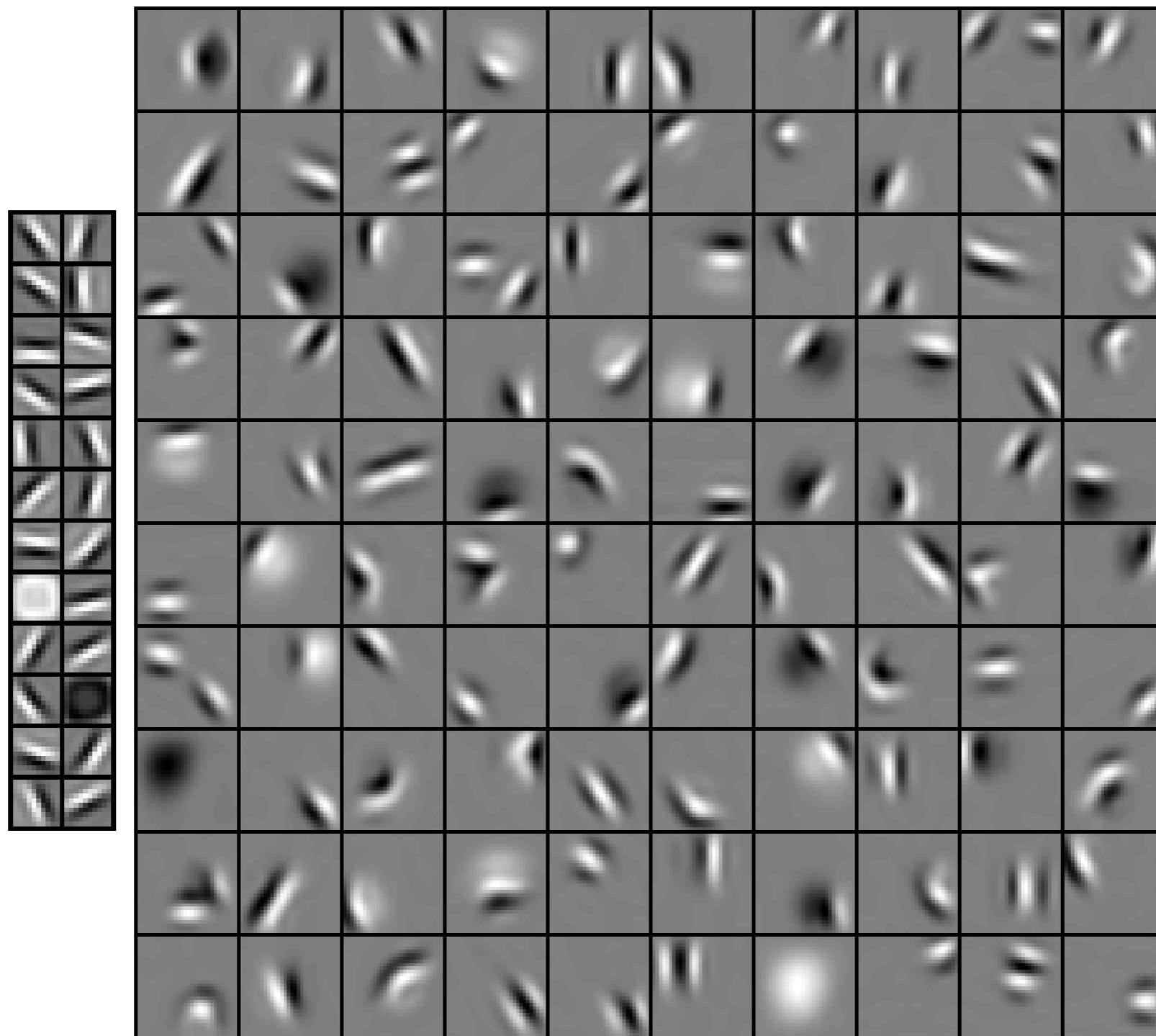
IPython Time

git clone [https://github.com/vkaynig/
IACS_ComputeFest_DeepLearning](https://github.com/vkaynig/IACS_ComputeFest_DeepLearning)

ConvolutionalNetworks.ipynb

<http://goo.gl/LCJGBn>

Learned features from Natural Images



Lee et al.,
ICML, 2009

Advanced Methods

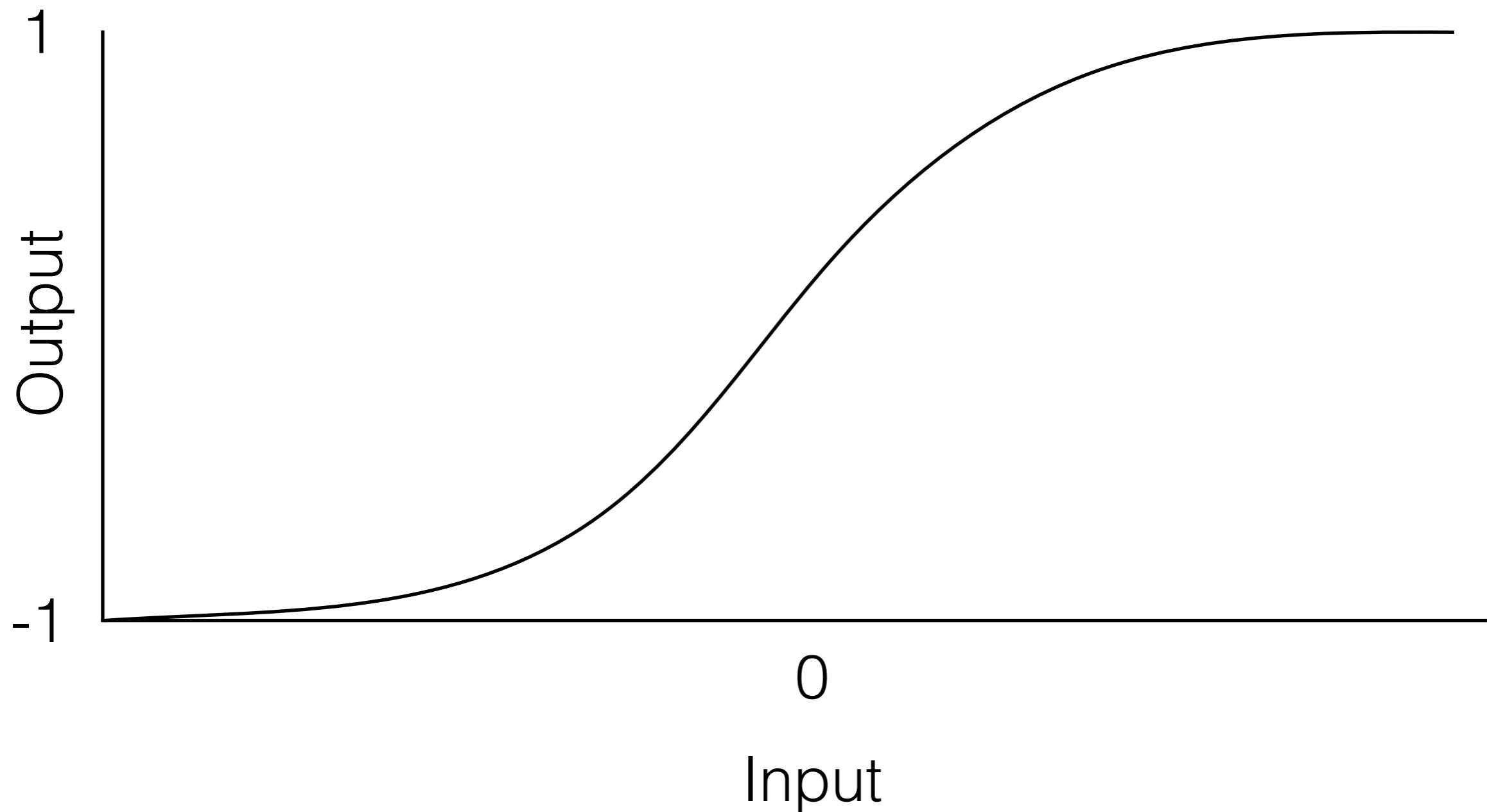
- Dropout & Batch Normalization
- Maxout
- Lasagne, Pylearn2, Caffe, Torch, TensorFlow
- “Intriguing properties of neural networks”
- "Residual Learning"

Dropout training

- Prevent outputs from becoming co-dependent.
- During training, randomly set some fraction of inputs at each layer to zero.
- After training, scale \mathbf{W} to compensate for dropout.
- Hidden: dropout fraction 0.5
Input: dropout fraction 0.2
- Theoretical links to model averaging.

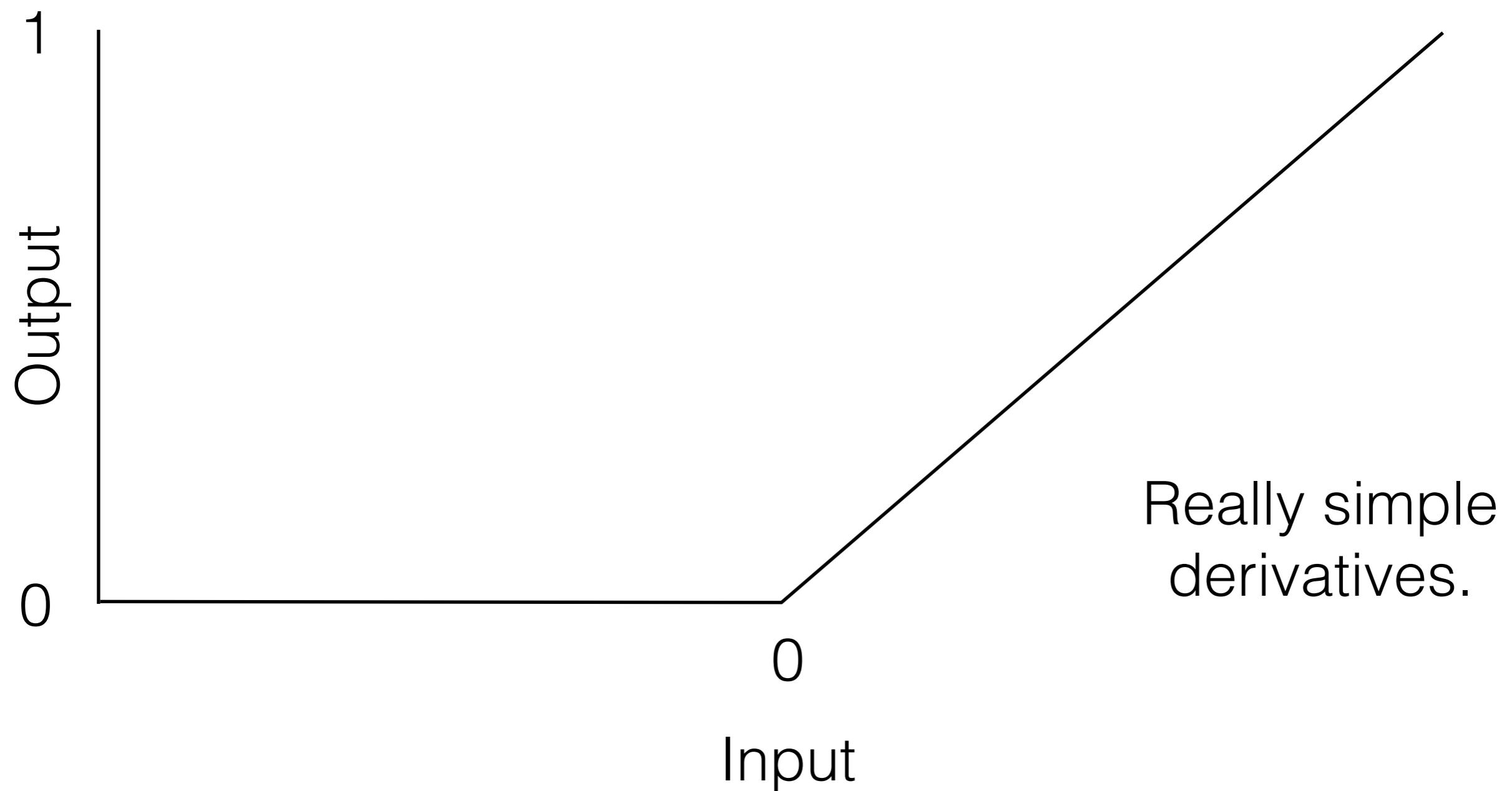
Nonlinear transformations

Sigmoid
 $\text{Tanh}(\text{input})$



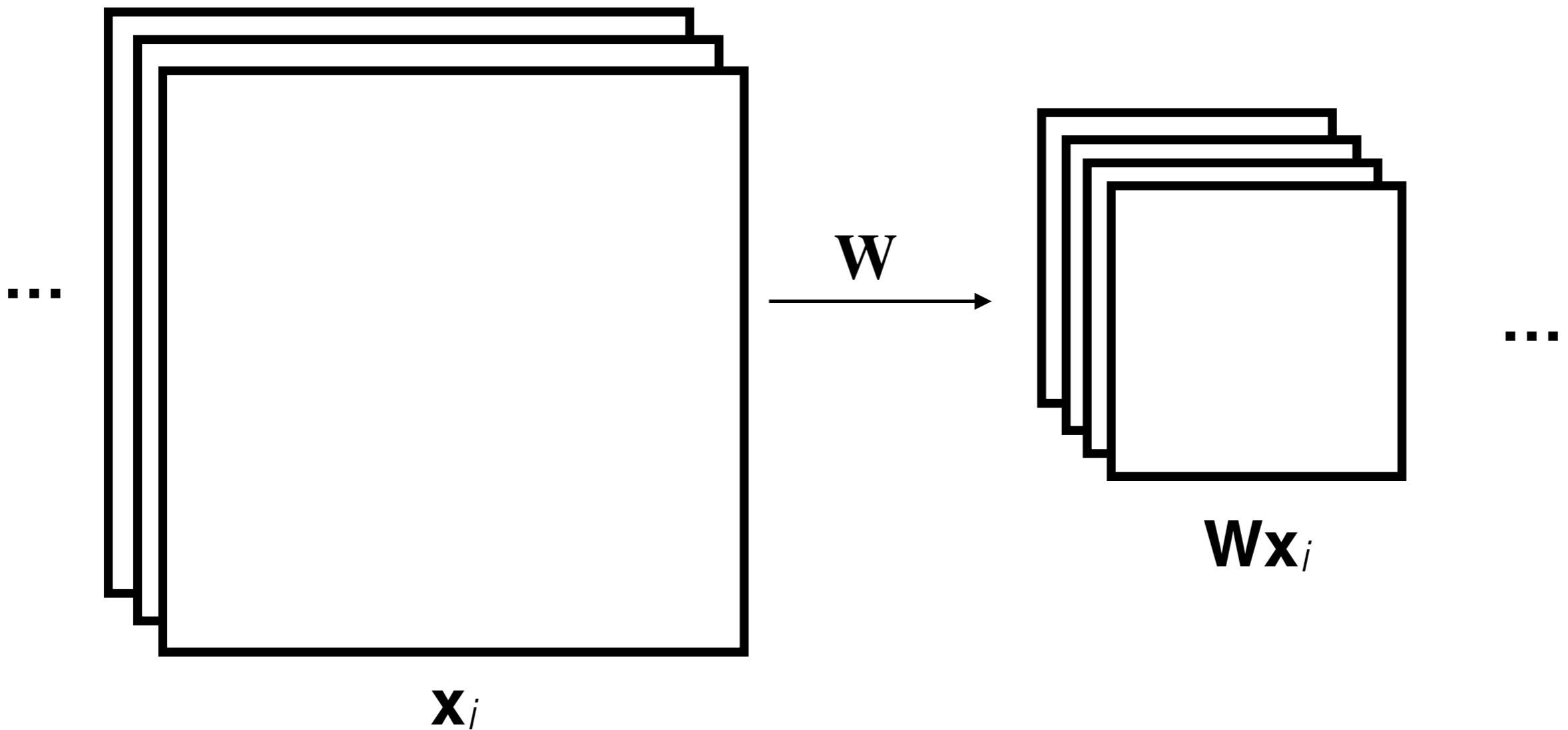
Nonlinear transformations

Rectified Linear Unit (ReLU)
 $\max(\text{input}, 0)$



Maxout

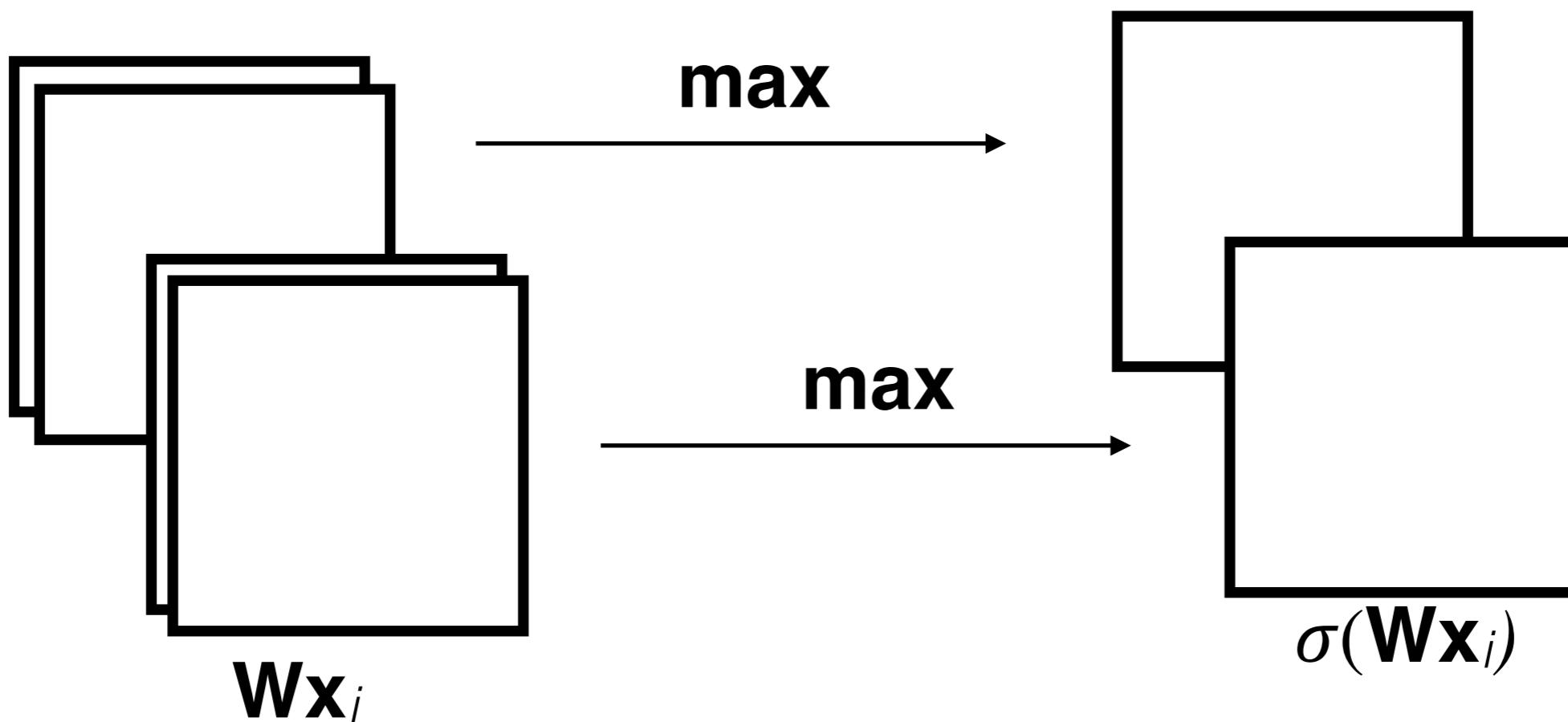
Like maxpooling, but using multiple feature maps.



$$\mathbf{x}_{i+1} = \sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b})$$

Maxout

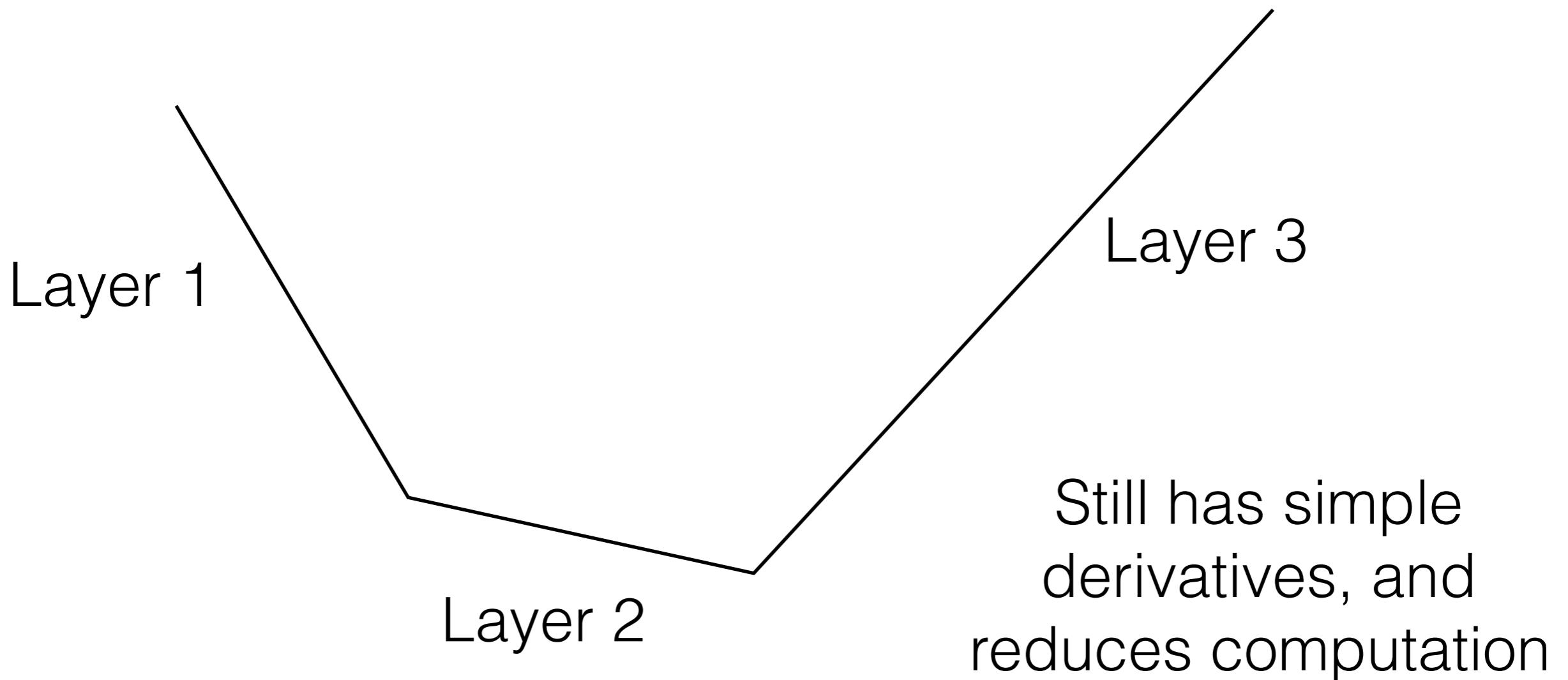
Like maxpooling, but using multiple feature maps.



$$\mathbf{x}_{i+1} = \sigma_i(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)$$

Maxout

Like ReLU, but with multiple linear functions.



Beyond Theano

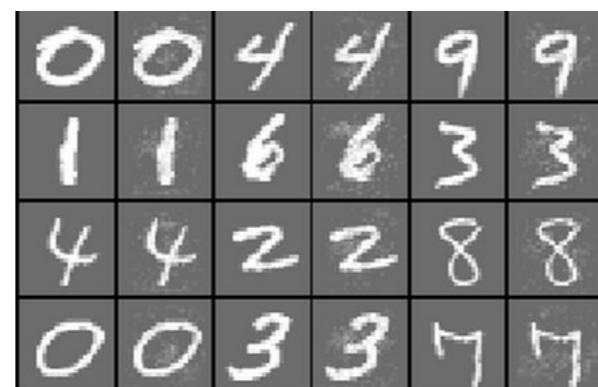
- Pylearn2 = Theano++, with YAML files for models.
- Lasagne = Theano++ with a simple wrapper
- Torch = Theano, but in Lua. torch-autograd is cool.
- Caffe = Image-centric, Convnets, not as flexible for Deep Learning.
- TensorFlow = Google's Theano (in Python wrapping C).
- Everyone uses CUDA and cuDNN.

“Intriguing Properties of Neural Networks”

Classification “manifold” is very nonlinear, in CNNs.
Small offsets can cause significant label changes.

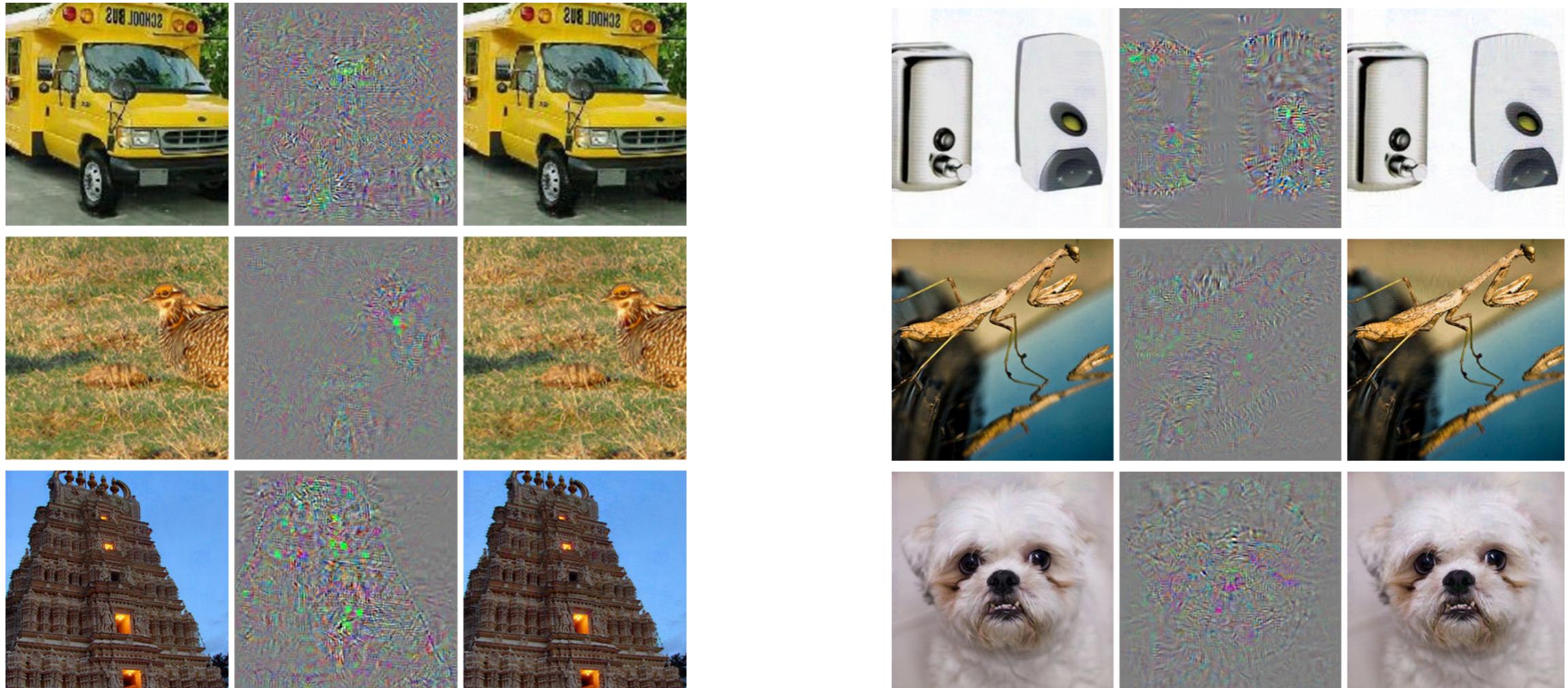


Even columns:
adversarial examples
(0% accuracy)



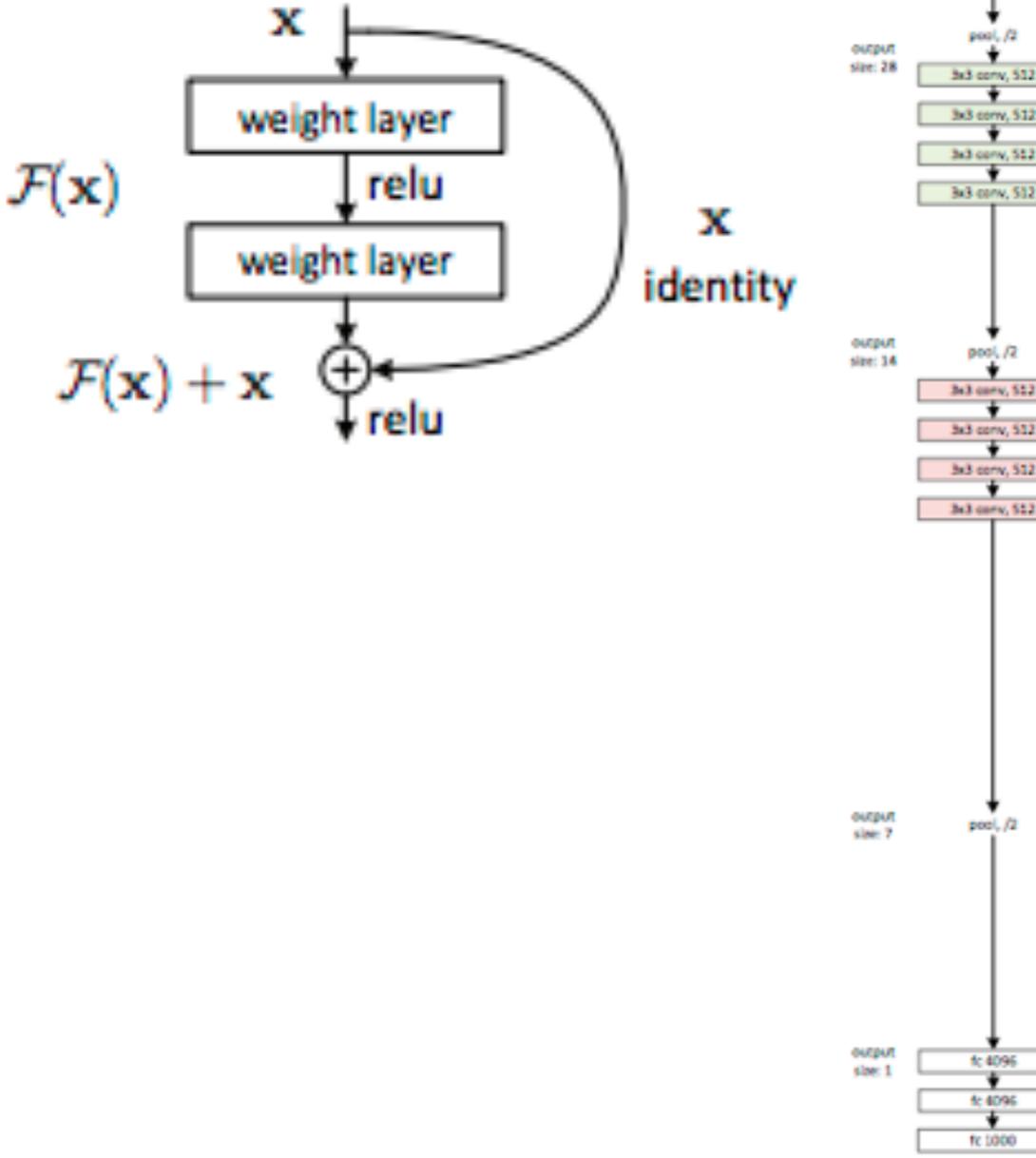
Even columns:
Gaussian noise
(50% accuracy)

“Intriguing Properties of Neural Networks”



Left: correctly identified
Center: added noise x10,
Right: “Ostrich”

Residual Learning



Winner, 2015

Imagine Large Scale Visual Recognition

MS "Common Objects in Context"

Thank You!