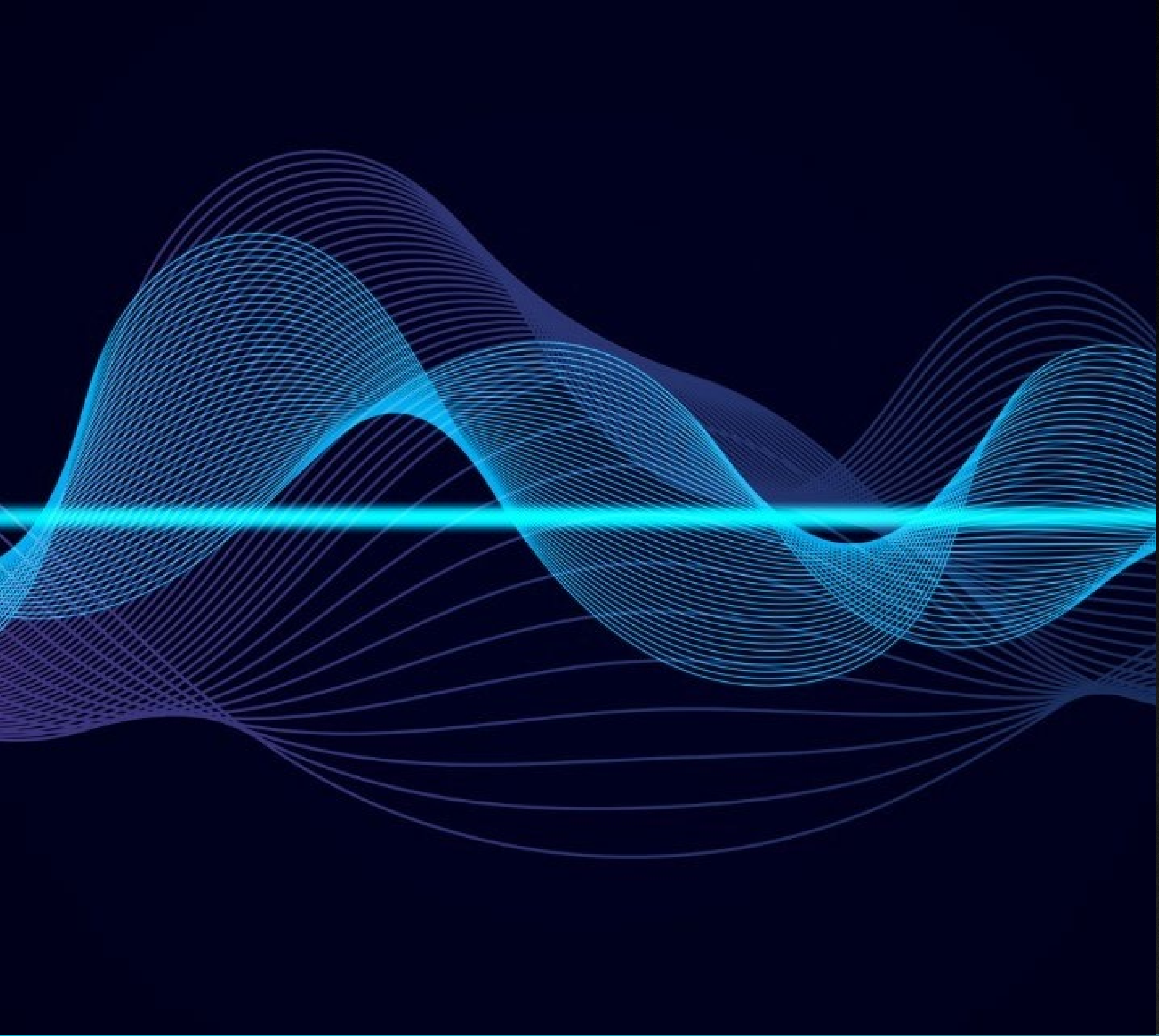


Computational Physics

PHYS 6260

Deep Learning: Convolutional NNs

Announcements:

- HW7: Due Wednesday 3/27
- Progress report: Due Monday 4/1

Lecture Outline

We will cover these topics

- Operations in a CNN
- Putting it together
- Example pioneering CNNs
- Feature learning / visualization

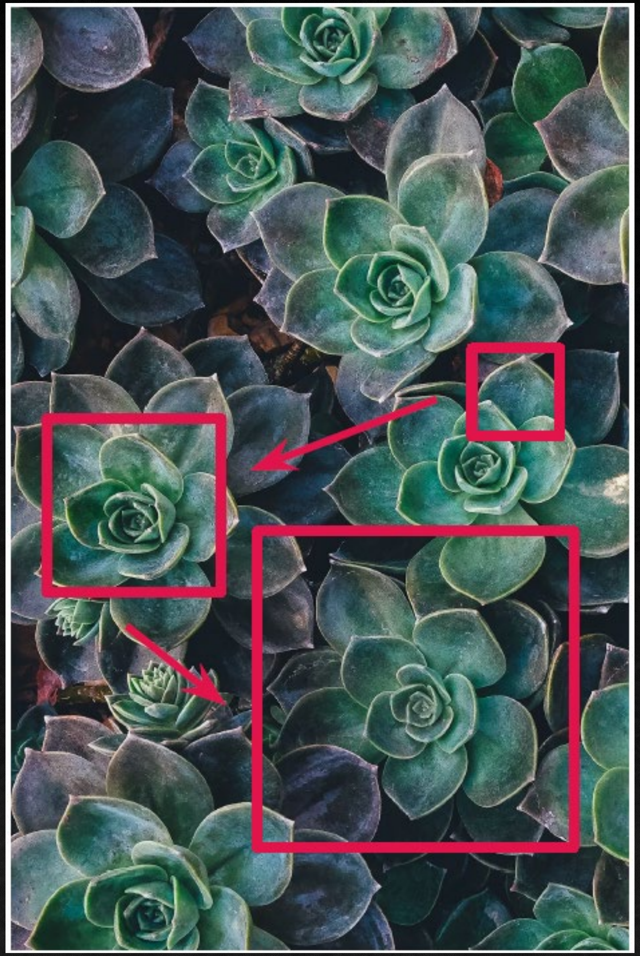
NNs for image classification: can we do better?

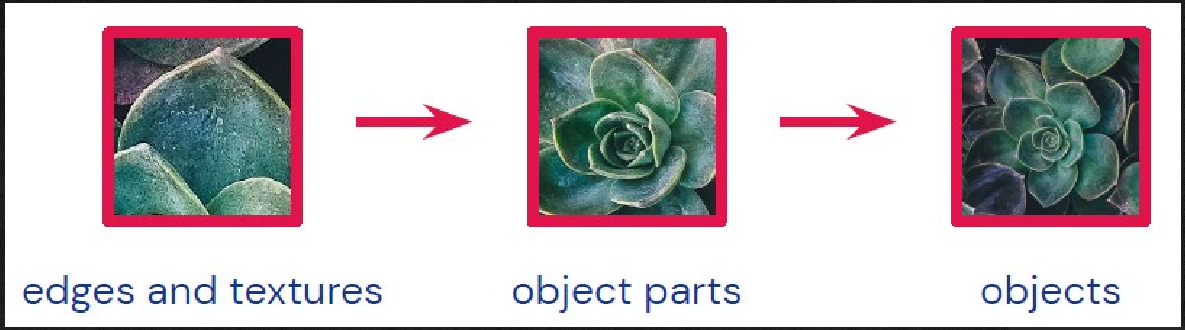
- In a “vanilla” neural net, the inputs are just a vector of pixels

- Can we incorporate the clustering patterns into the NN?
- Enter the convolutional NN (CNN aka convnets)

Convolutional NNs: topology

- Locality: nearby pixels are more strongly correlated
- Translation invariance: meaningful patterns can occur anywhere in the image
- Weight sharing: use the same network parameters to detect local patterns at many locations in the image
- Hierarchy: local low-level features are composed into larger, more abstract features



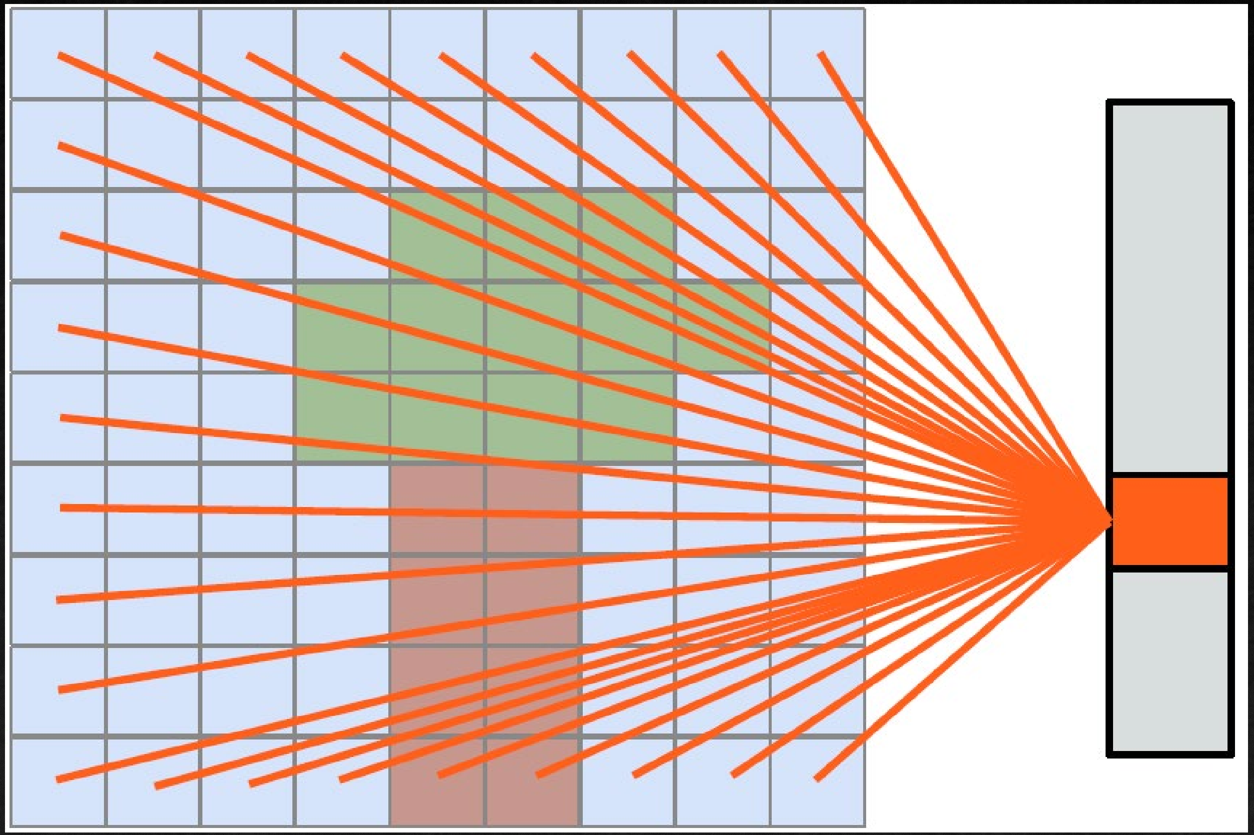


From fully connected to locally connected

- Using convolutions to locally connect pixels

- A fully connected unit is the weighted sum of all pixels (b is a constant)

$$y = \sum_{i \in image} w_i x_i + b$$

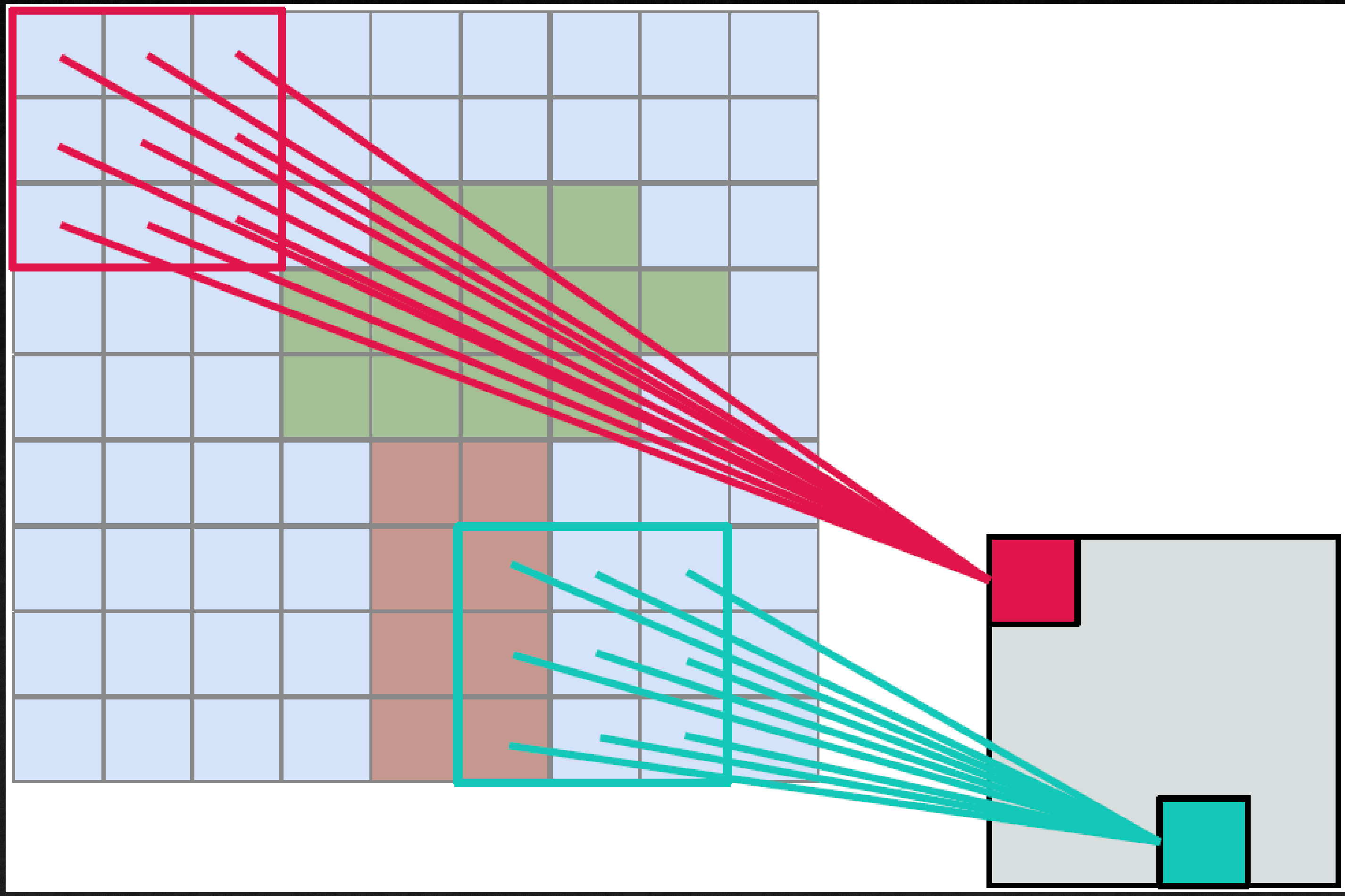


From fully connected to locally connected

- Locally connected units are computed through spatially varying filters
- For example, the image is filtered (unique for each placement) as

$$y = \sum_{i \in 3\times3} w_i x_i + b$$

- Retains locality

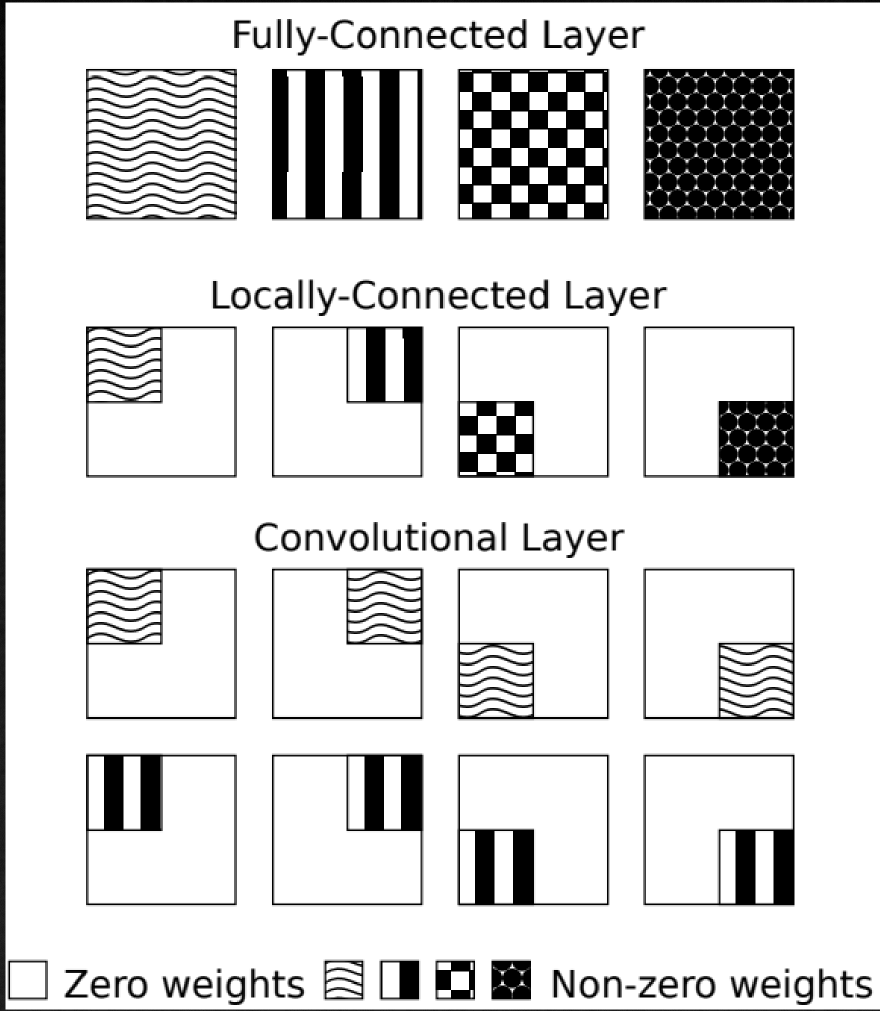


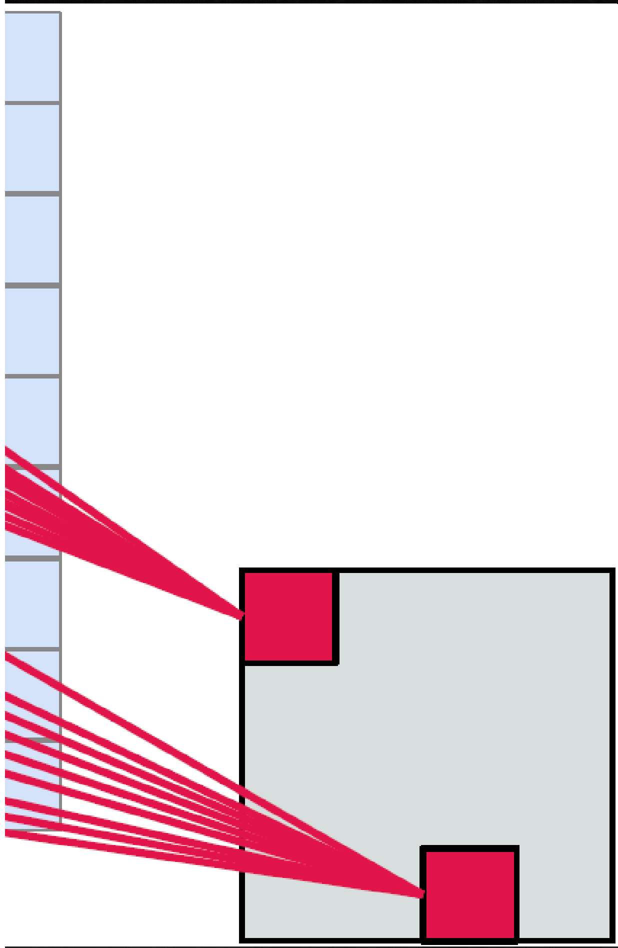
From locally connected to convolutional

- A convolution ($*$) uses the same filter for all locations

$$y = w * x + b$$

- Receptive field: inputs
- Feature map: output



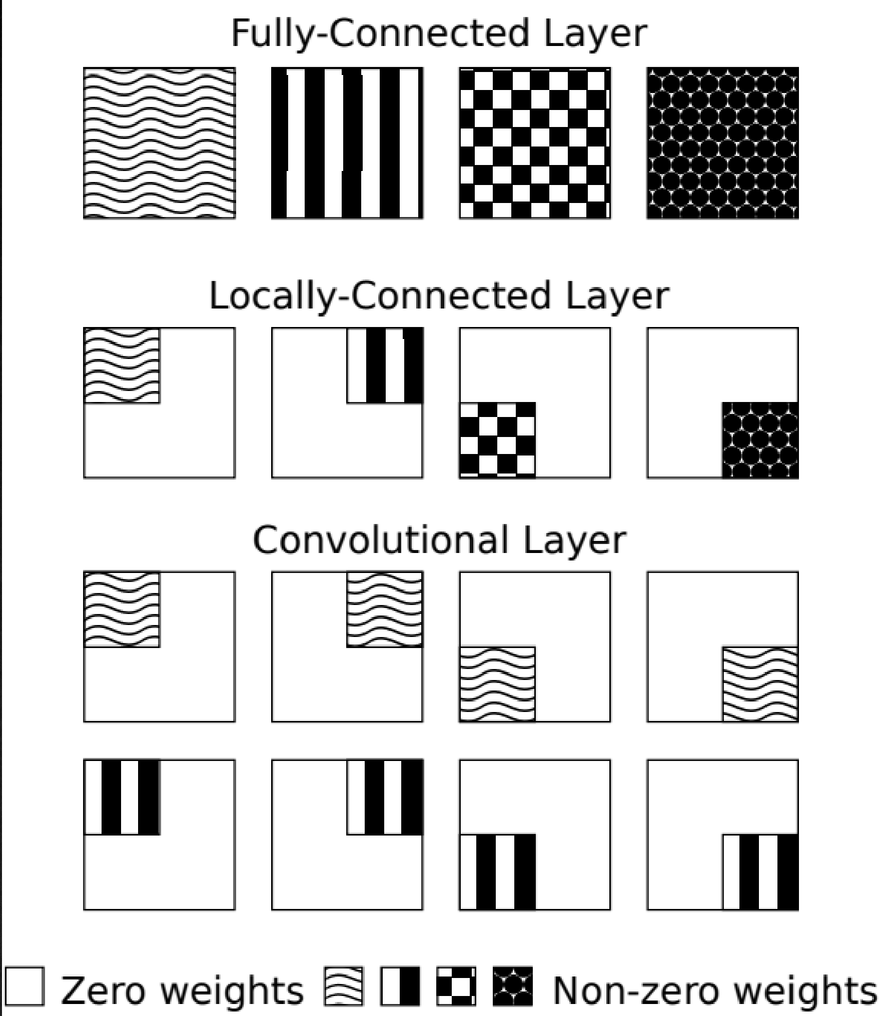


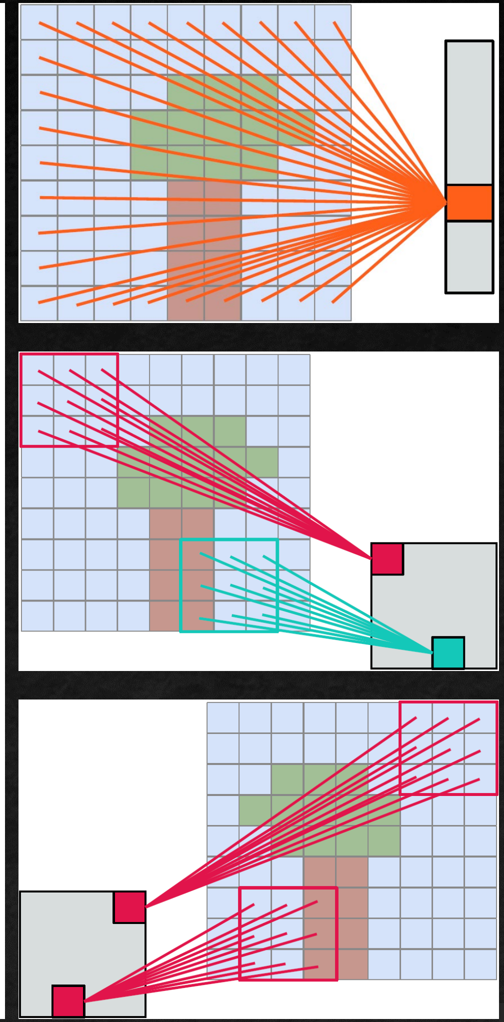
From locally connected to convolutional

- A convolution ($*$) uses the same filter for all locations

$$y = w * x + b$$

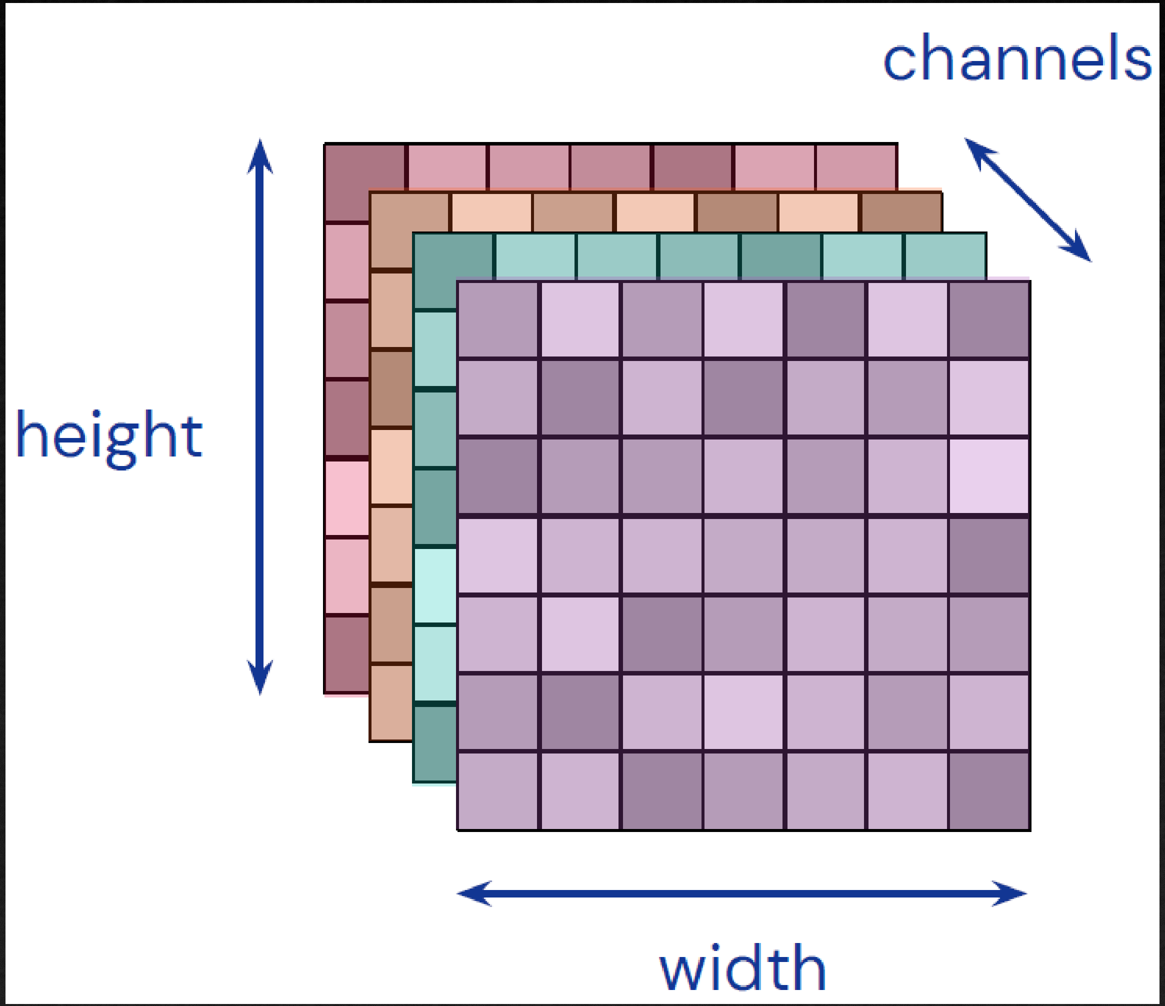
- Receptive field: inputs
- Feature map: output





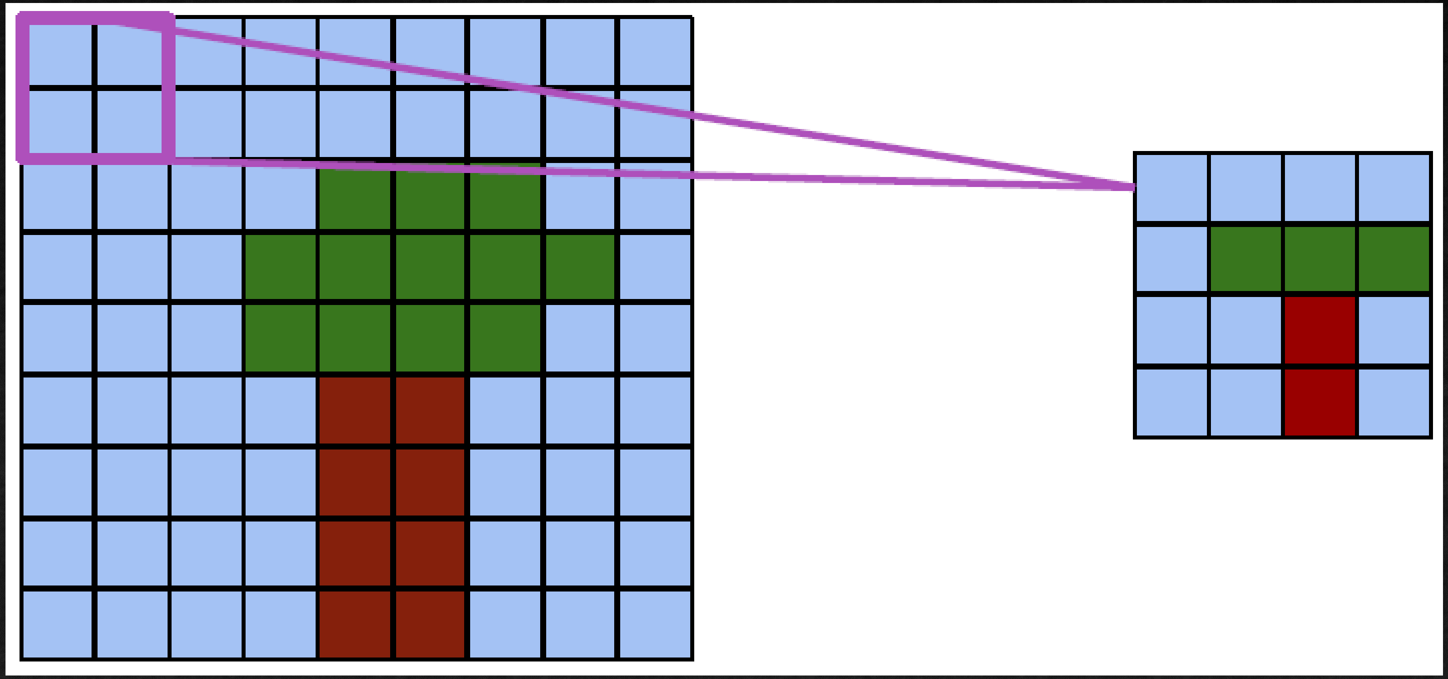
Inputs and outputs can be tensors

- Images: RGB
- Physics: different fields



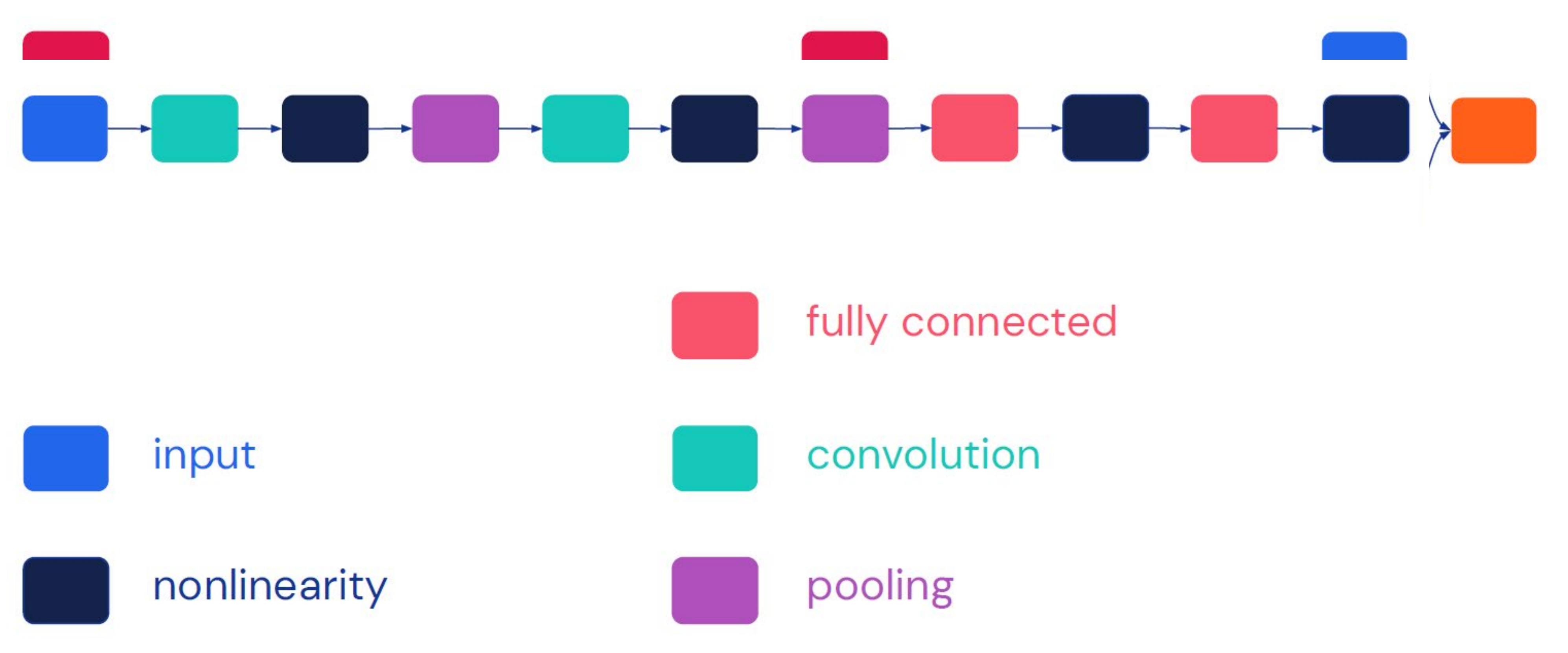
Pooling

- Compute mean or max over small windows to reduce resolution



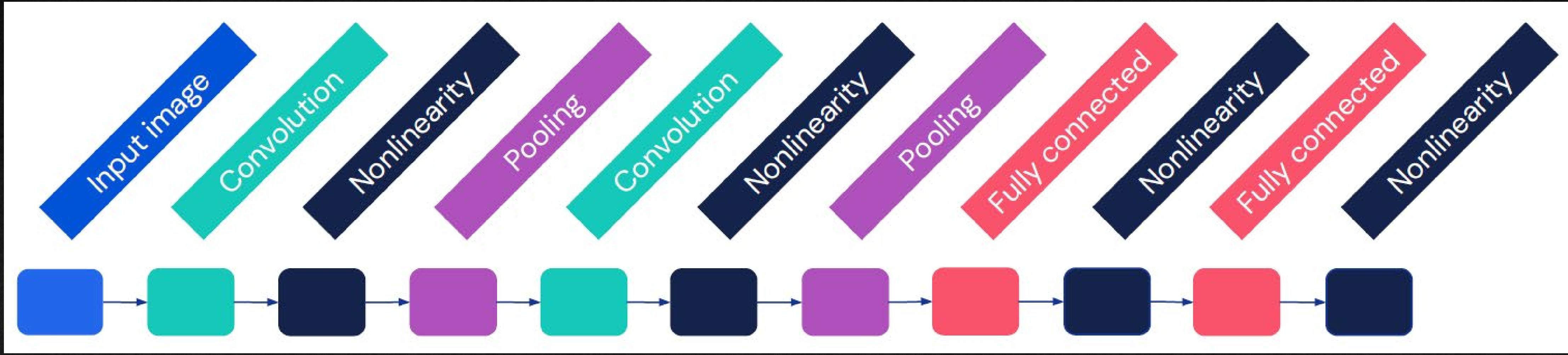
NNs as computational graphs

- In a deep NN, we have many connected layers, ultimately calculating a loss



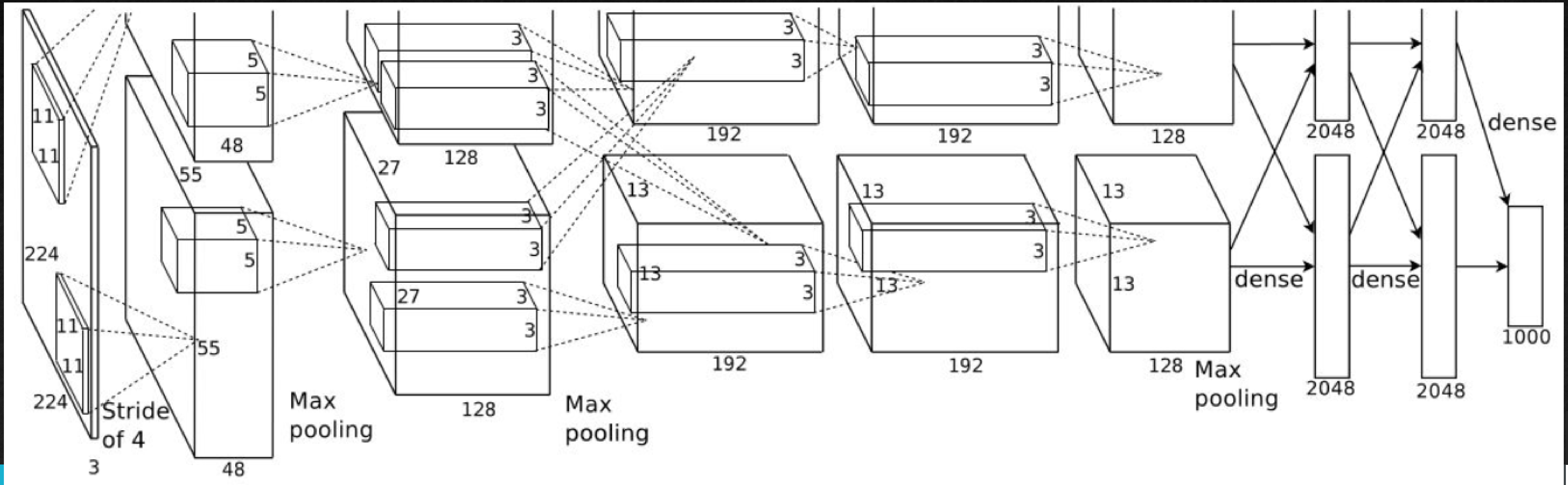
LeNet-5 (1998)

- One of the earliest CNN: Yann LeCun started development in 1989 on it
- LeCun et al. (1998): *Gradient-based learning applied to document recognition*

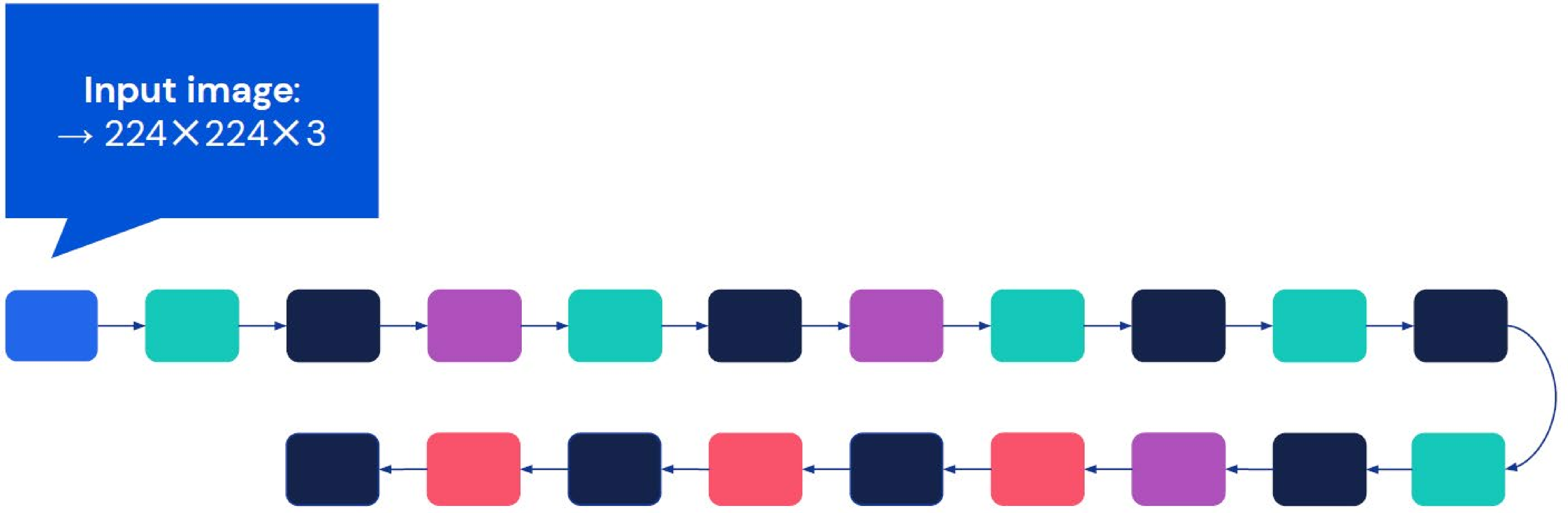


AlexNet (2012)

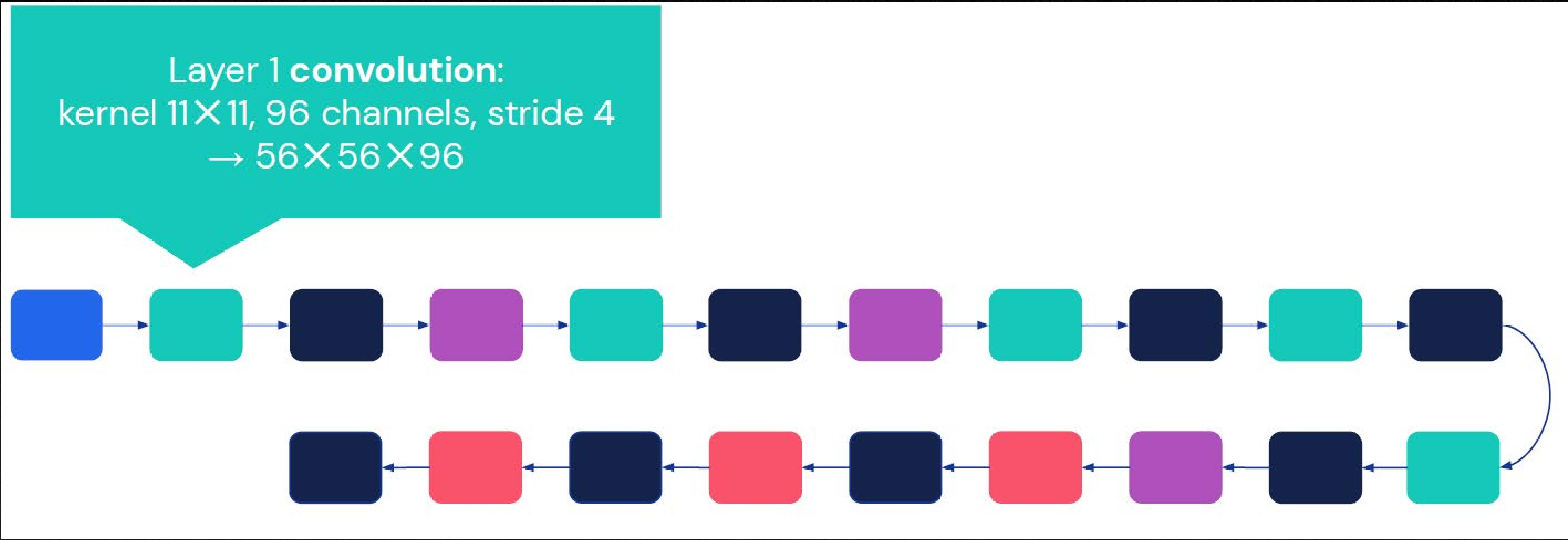
- Architecture: 8 layers, ReLU, dropout, weight decay
- Trained on 2 GPUs for 6 days on 1.2 million images
- Krizhevsky et al. (2012): *ImageNet classification with deep convolutional neural networks*



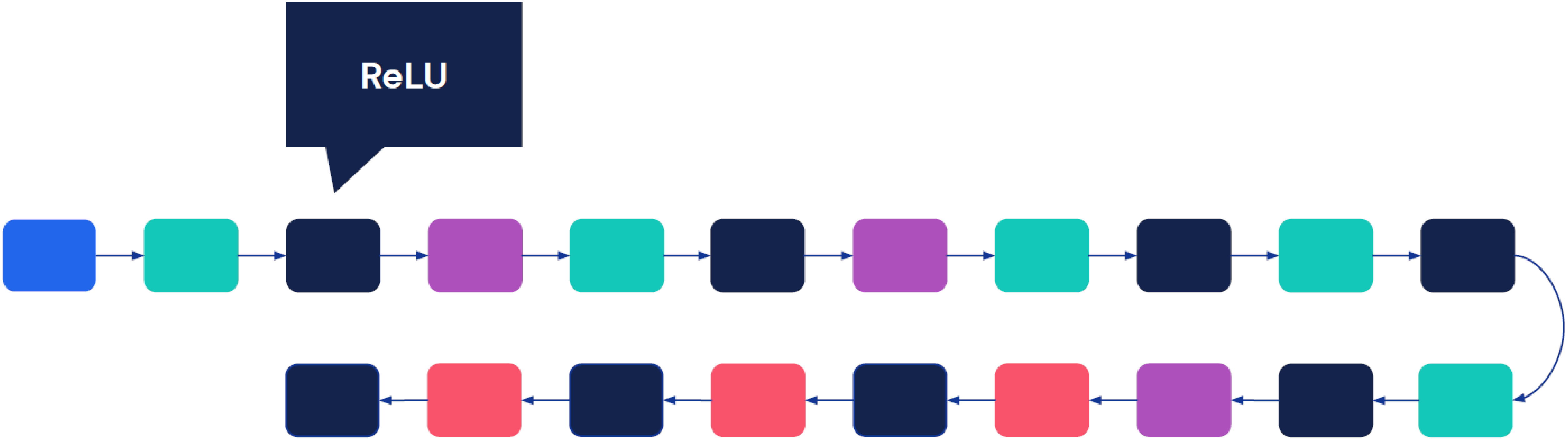
AlexNet (2012)



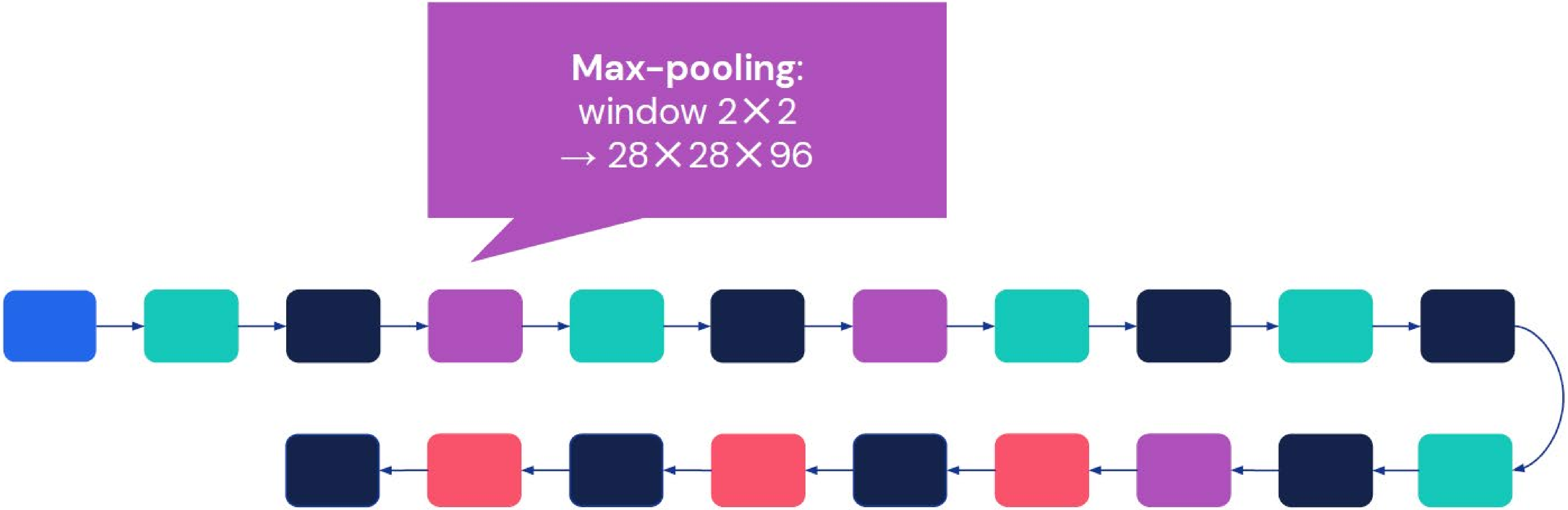
AlexNet (2012)



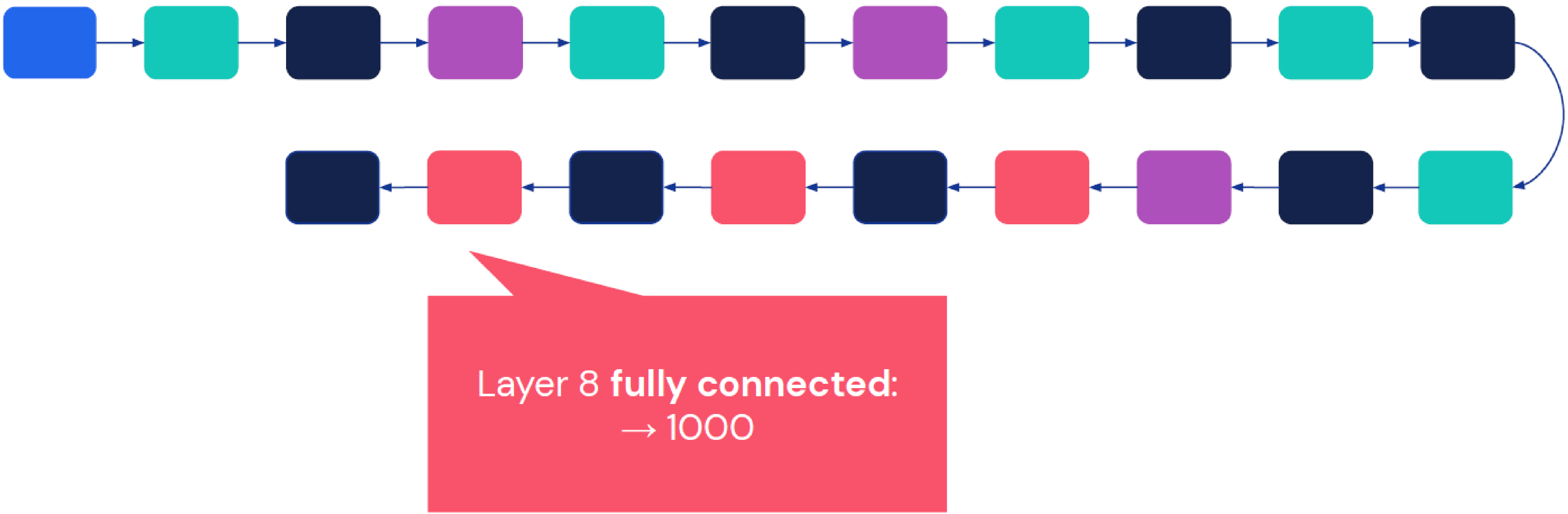
AlexNet (2012)



AlexNet (2012)

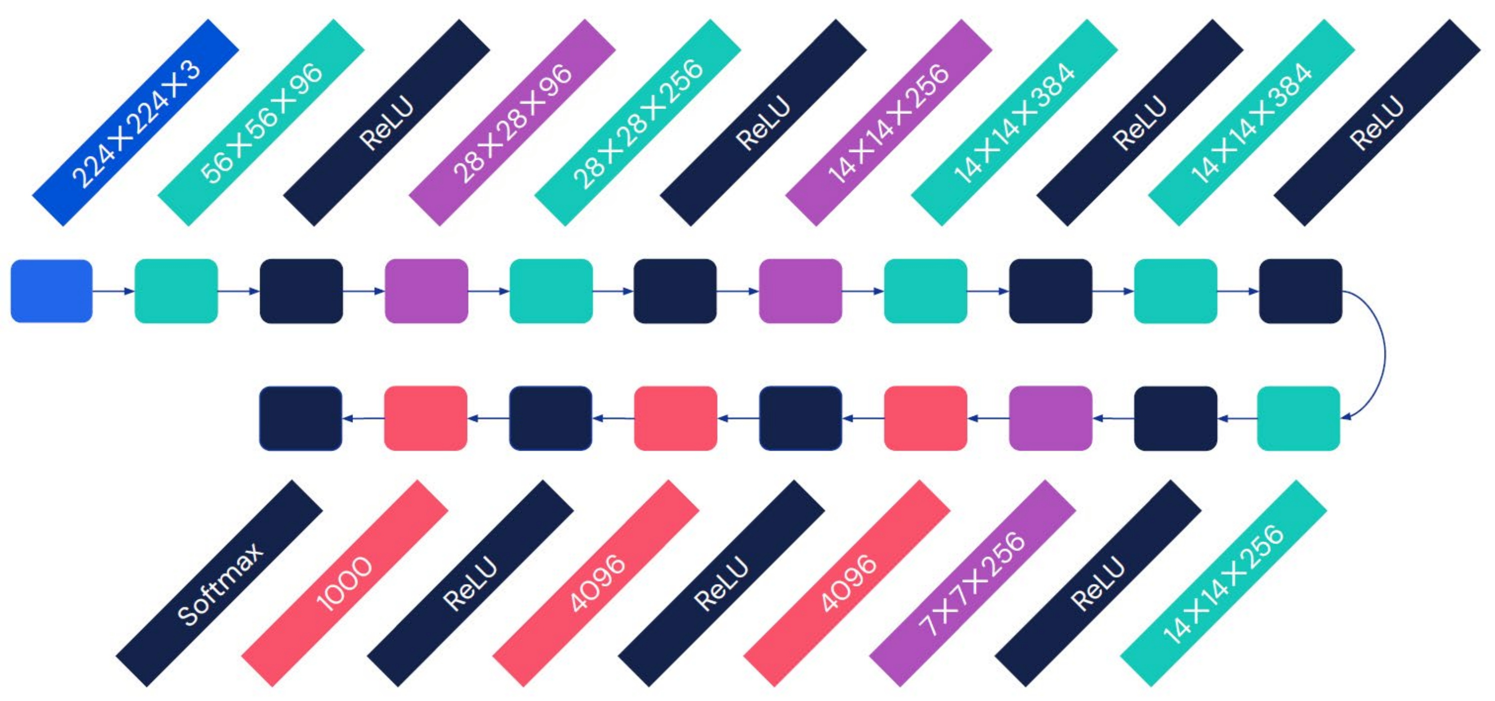


AlexNet (2012)

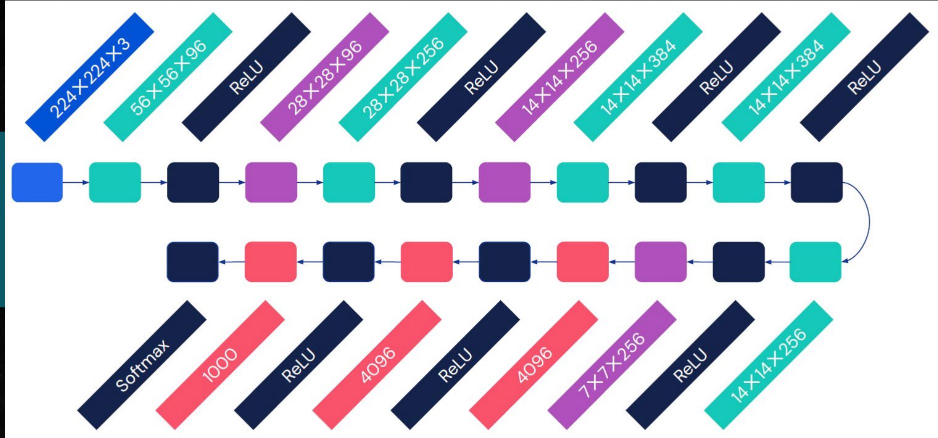


AlexNet (2012)

- Number of inputs = 150,528
- Number of neurons = 290,400 + 186,624 + 64,896 + 64,896 + 43,264 + 4,096 + 4,096 + 1,000 = 659,272



AlexNet (2012)



- Trained using stochastic gradient descent
- Batch size of 128 examples, a (small) weight decay w = 0.0005, and "momentum" v = 0.9
- The weights in each filter were updated as

$$v_{i+1} = 0.9 v_i - 0.0005\, \epsilon\, w_i - \epsilon \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} = w_i + v_{i+1}$$

- Here D_i is the batch, and ϵ is the learning rate (started at 0.01 and reduced three times during the training)
- Weights are initialized with a Gaussian random numbers ($\mu = 0.01$)

AlexNet (2012)

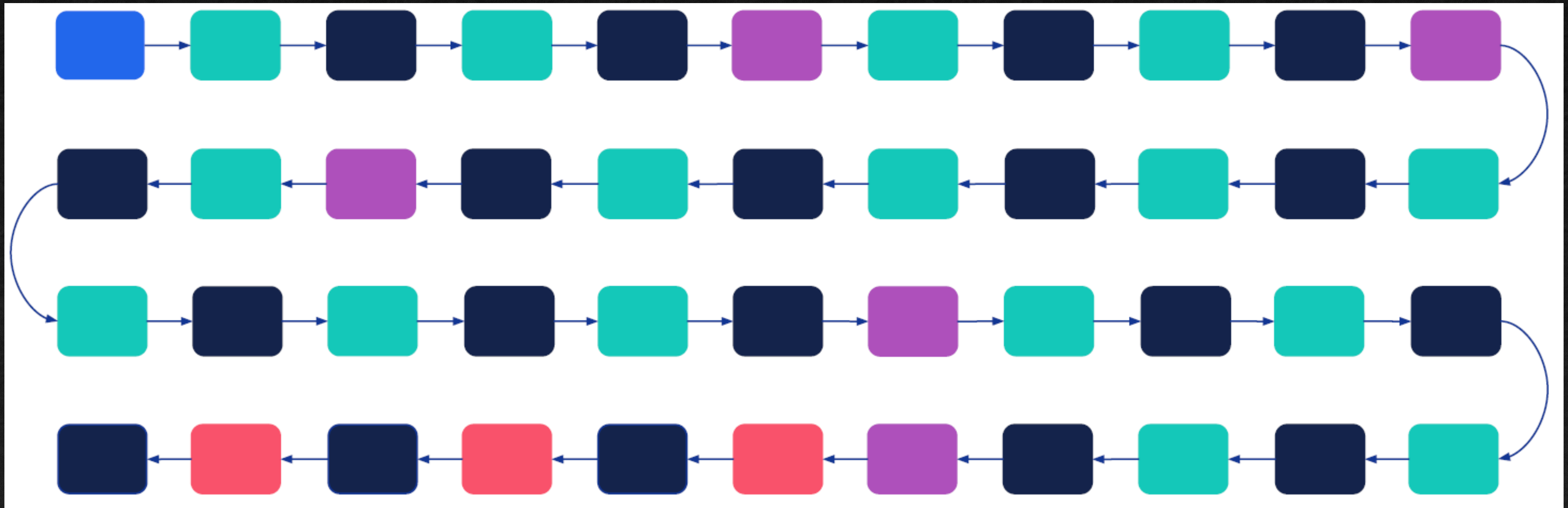
- 96 learned filters in the first convolutional layer

Deeper is better

- Each layer is a linear classifer by itself
- More layers – more non-linearities
- What limits the number of layers in CNNs?
  - Computational expense (runtime and memory)
  - Optimization difficulties
    - How to initialize?
    - Sophisticated optimizers
    - Normalization layers
    - Network design

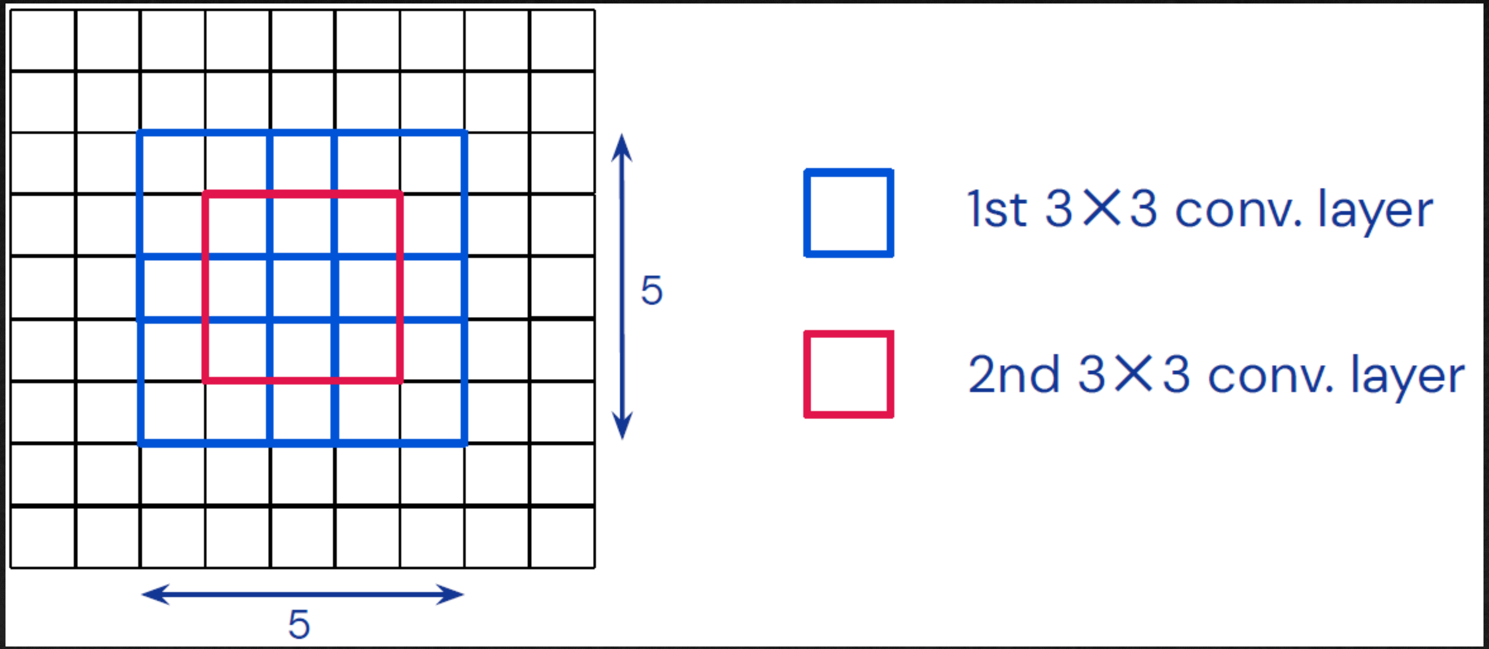
VGGNet (2014)

- Stacked many convolutional layers before pooling
- Used the “same” convolutions to avoid resolution reduction
- Simonyan & Zisserman (2015): *Very deep convolutional networks for large-scale image recognition*



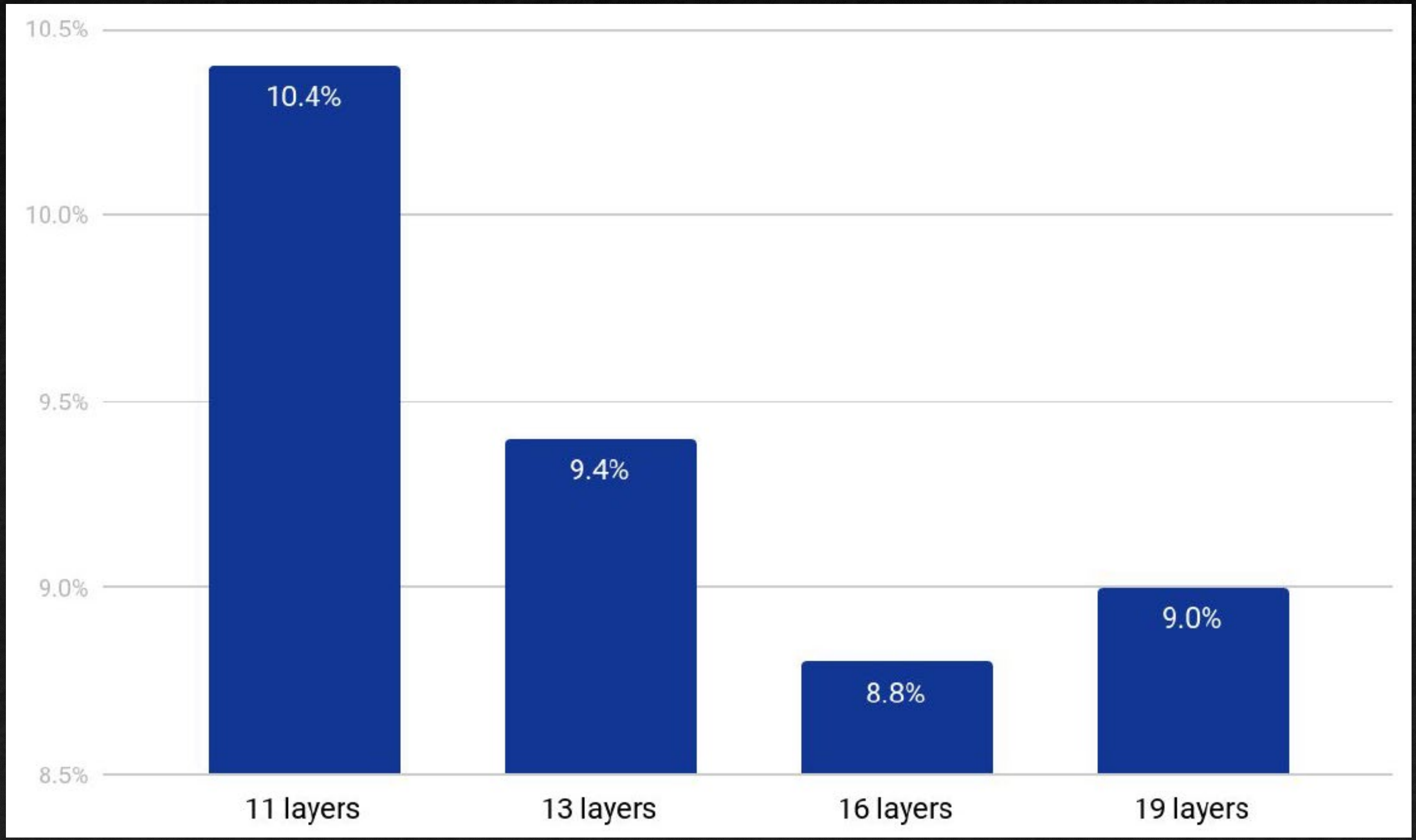
VGGNet (2014)

- Architecture: up to 19 layers, 3x3 kernels only
- Trained for 3 weeks on 4 GPUs
- Stacked 3x3 kernels



VGGNet (2014)

- Error minimum at 16 layers

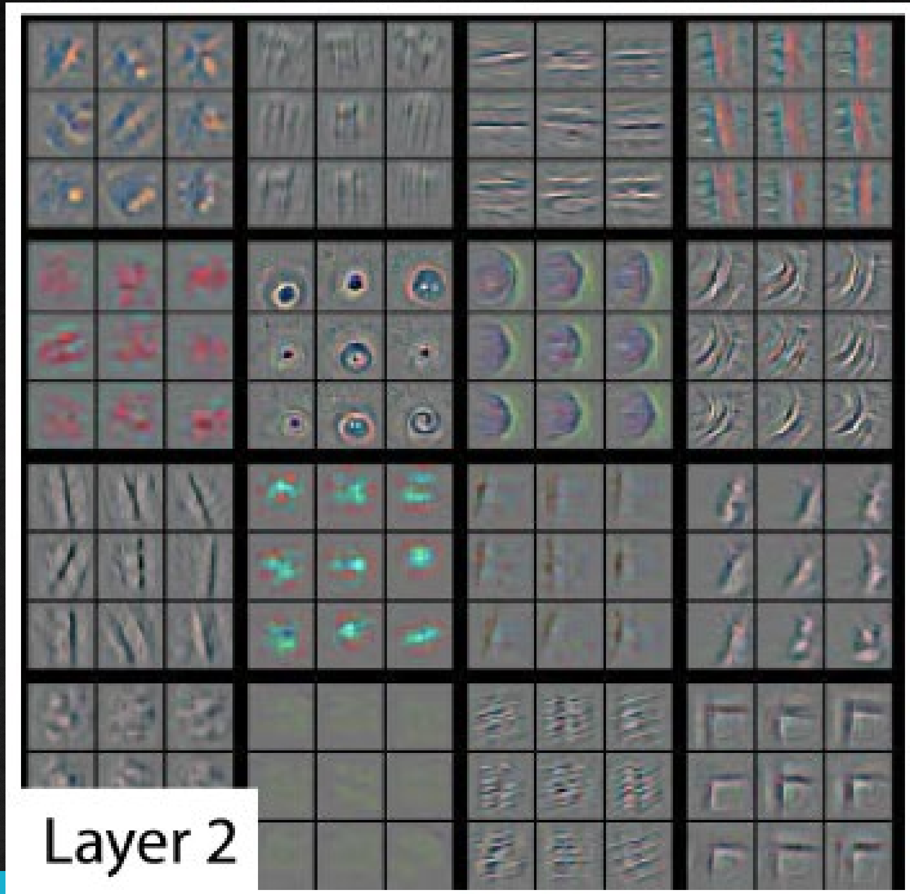


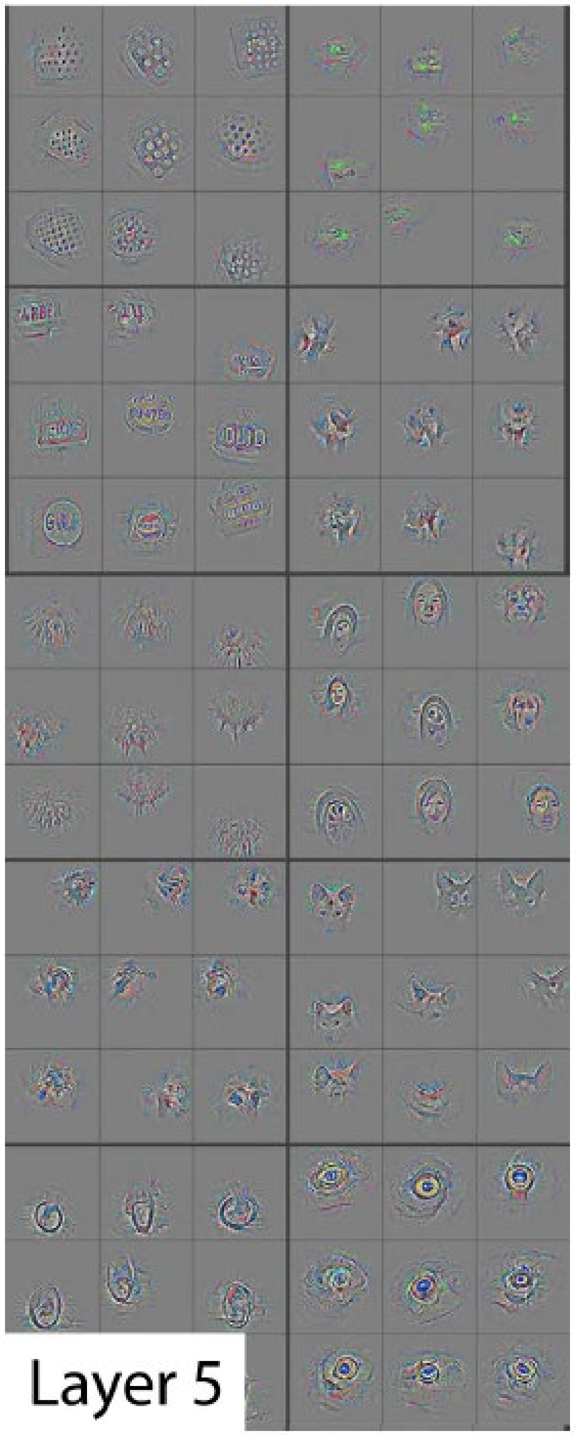
Data augmentation

- By design, CNNs are only robust against translation
- Data augmentation modifies the inputs through various transformations: rotation, scaling, shearing, warping, etc.
- Makes it robust against overfitting

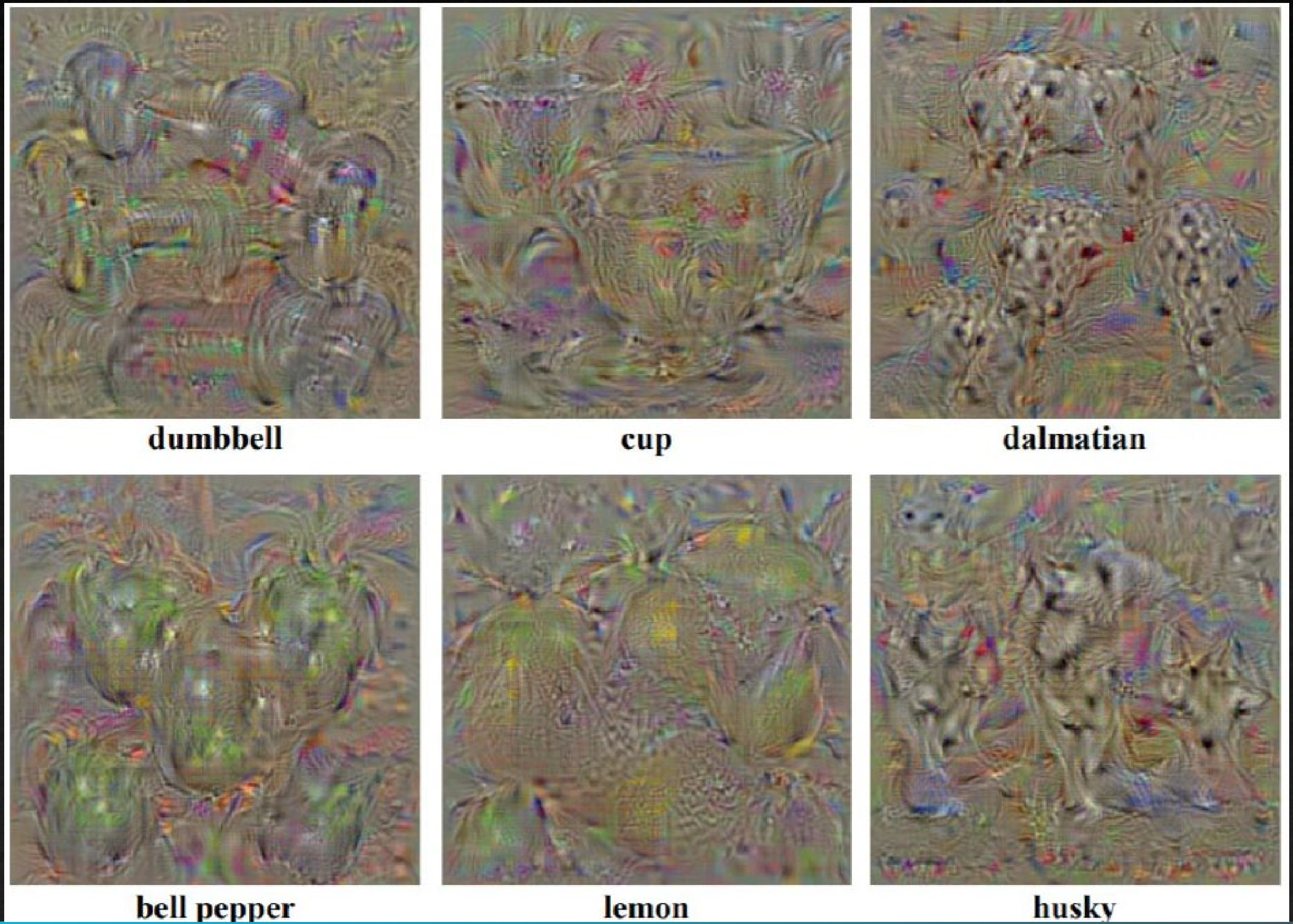
Visualizing what a CNN lear

- Zeiler & Fergus (2014): *Visualizing and understanding convolutional networks*

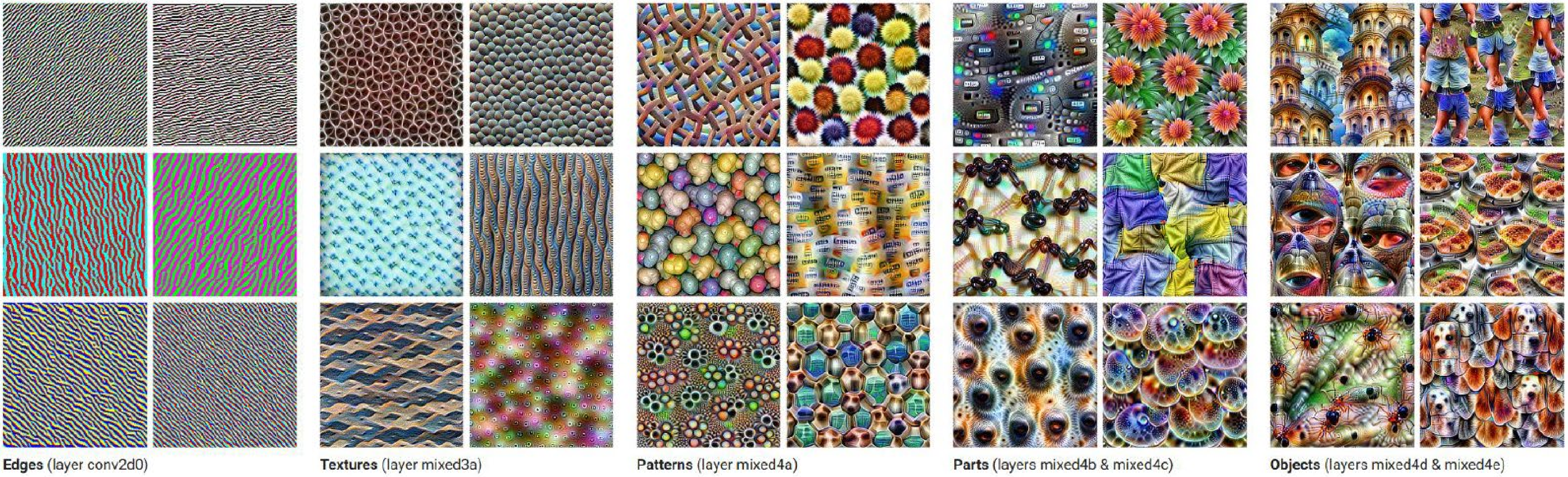




Visualizing what a CNN learns



Feature visualization: How CNNs build their understanding of images



Feature visualization allows us to see how GoogLeNet [1], trained on the ImageNet [2] dataset, builds up its understanding of images over many layers. Visualizations of all channels are available in the appendix.