Neural Networks and Deep Learning

CNNs

Adam Bloniarz

Department of Computer Science
adam.bloniarz@colorado.edu

February 6, 2020

University of Colorado **Boulder**

Discrete Convolution

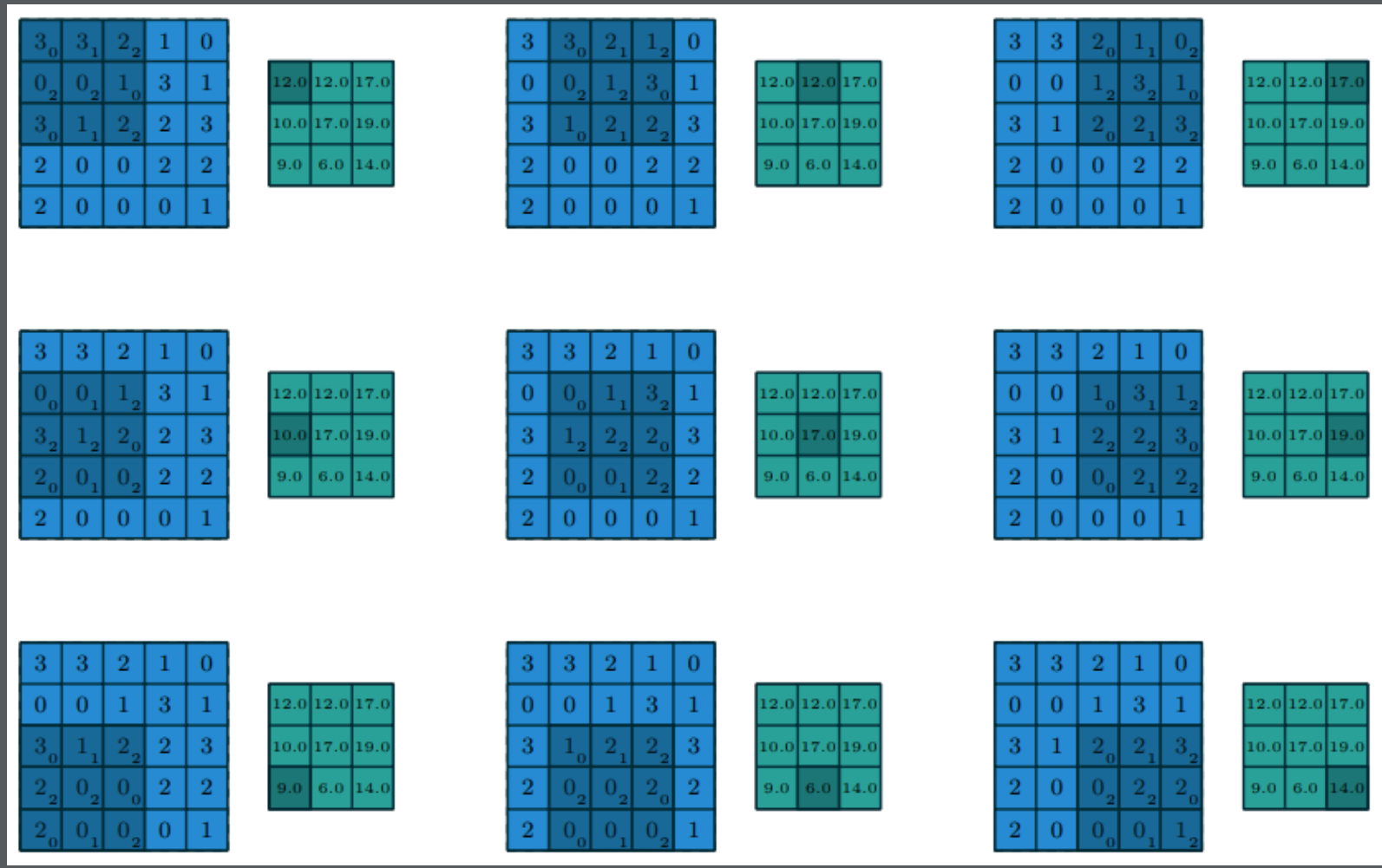


Image source: A guide to convolution arithmetic for deep learning.
blue: input feature map, *shaded blue*: kernel, *green*: output feature map.

Operator notation

Continuous signal (infinite resolution): Let $I(x, y)$ be an image, and let $K(x, y)$ be a convolutional filter.

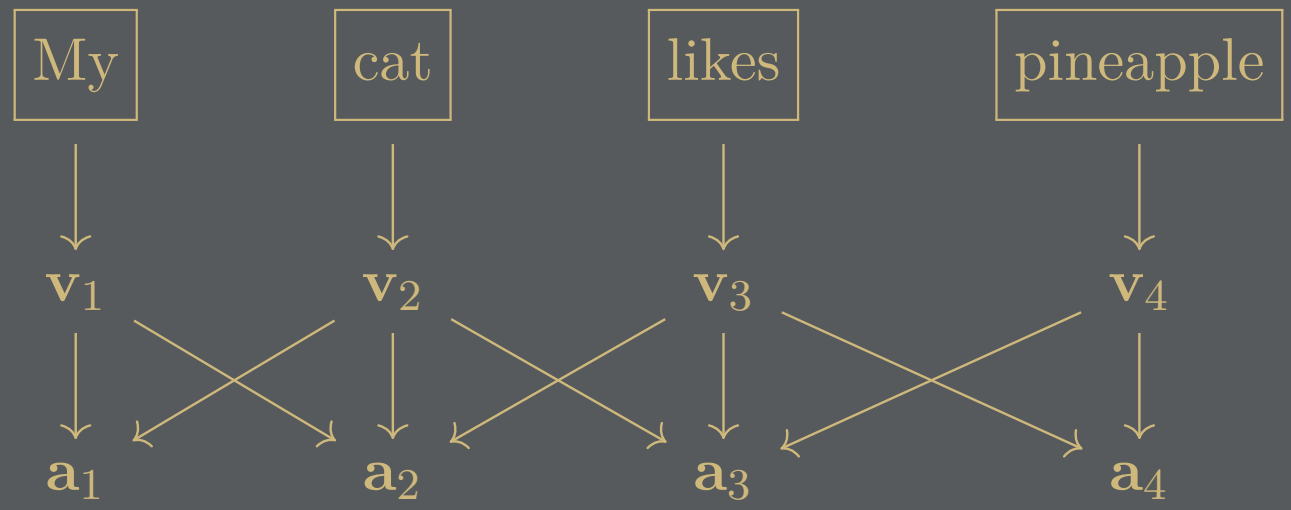
$$S(x, y) = (I * K)(x, y) = \int \int I(x - u, y - v)K(u, v)\, dudv$$

Discrete signal (finite resolution): Let $I(i, j)$ be an image, and let $K(m, n)$ be a convolutional filter.

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$$

Aside: CNNs in NLP

Convolution can be used to represent a word in its context.



$$\mathbf{a}_i = \mathbf{F}_{-1}\mathbf{v}_{i-1} + \mathbf{F}_0\mathbf{v}_i + \mathbf{F}_1\mathbf{v}_{i+1}$$

To get longer-range dependencies, other mechanisms exist, e.g. RNNs, transformers.

[MNIST colab]

Convolutional layer does not provide translation invariance, but rather *equivariance*.

Convolution operator commutes with translation operator:

Let $g_{u,v}$ be a shift operator:

$$(g_{u,v} \circ I)(i,j) = I(i-u, j-v)$$

Then

$$(g_{u,v} \circ I) * K = g_{u,v} \circ (I * K)$$

Invariance can be provided by pooling operations.

Let $H(i, j)$ be a hidden layer representation of the image. Global pooling provides translation invariance:

$$\sum_{i,j} H(i, j)$$

However, it destroys all spatial information. CNNs use local pooling operations to increase invariance to local deformation, while retaining some spatial information.

Pooling

- Pooling operations use some function to summarize subregions, such as taking the average or the maximum value.
- Pooling works by sliding a window across the input and feeding the content of the window to a pooling function.
- In some sense, pooling works very much like a discrete convolution, but replaces the linear combination described by the kernel with some other function.
- Types -
  - Max pooling
  - Norm pooling (l_∞ norm is max-pooling)
  - Average pooling

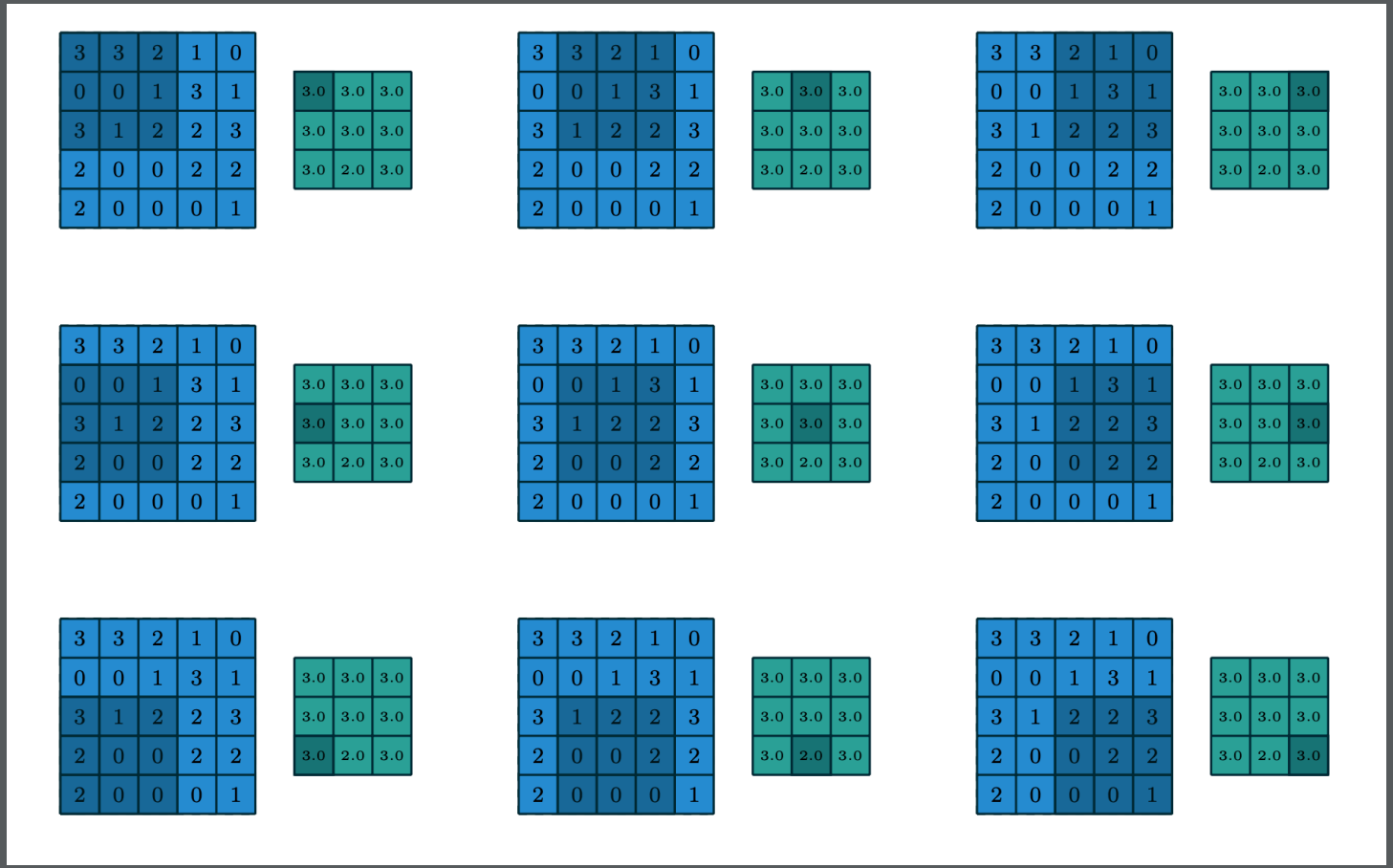
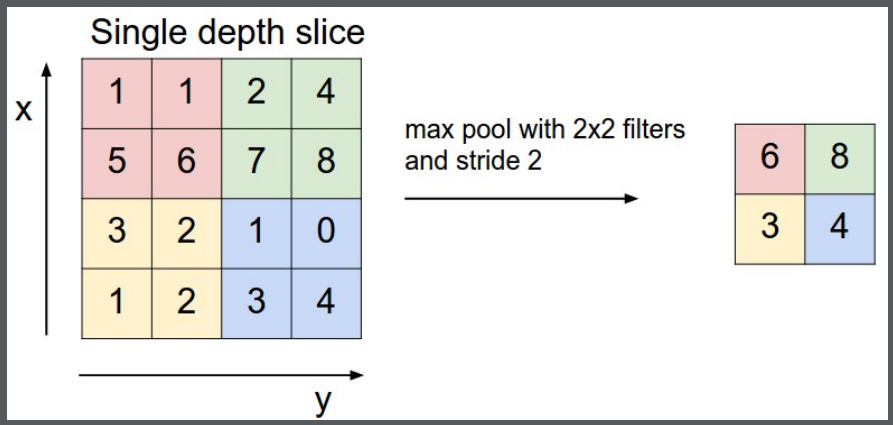


Image source: A guide to convolution arithmetic for deep learning.

In general, pooling may have a stride parameter, which leads to downsampling of the image.



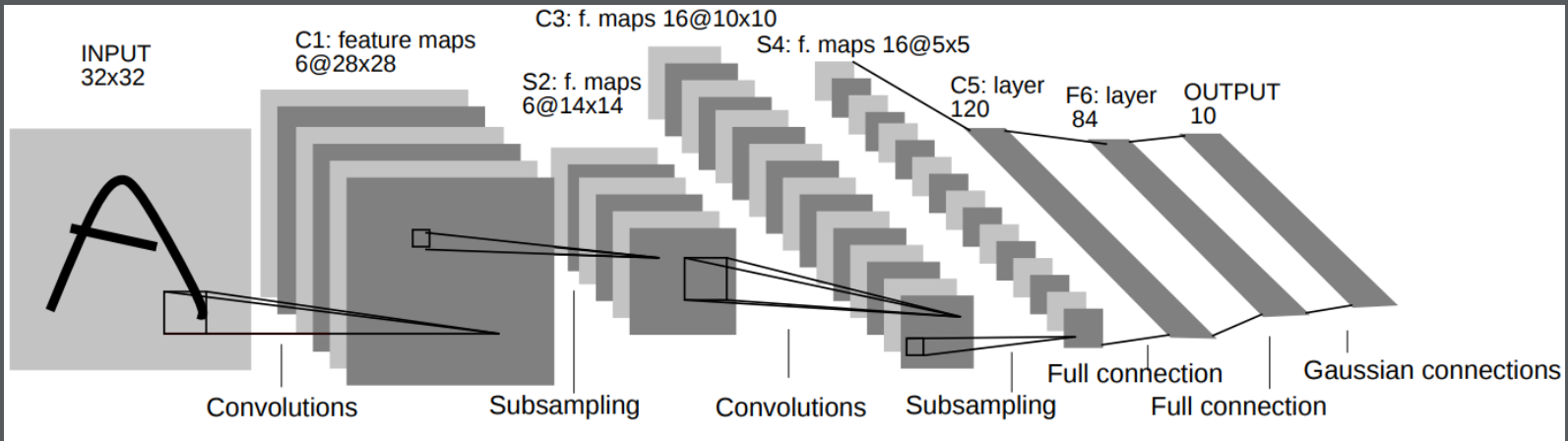
Source: http://cs231n.github.io/convolutional-networks/

Convolutional neural networks CNNs stack convolution, nonlinearities, and pooling layers.

There are fully connected layers prior to the softmax and loss layers.

LeNet (1998)

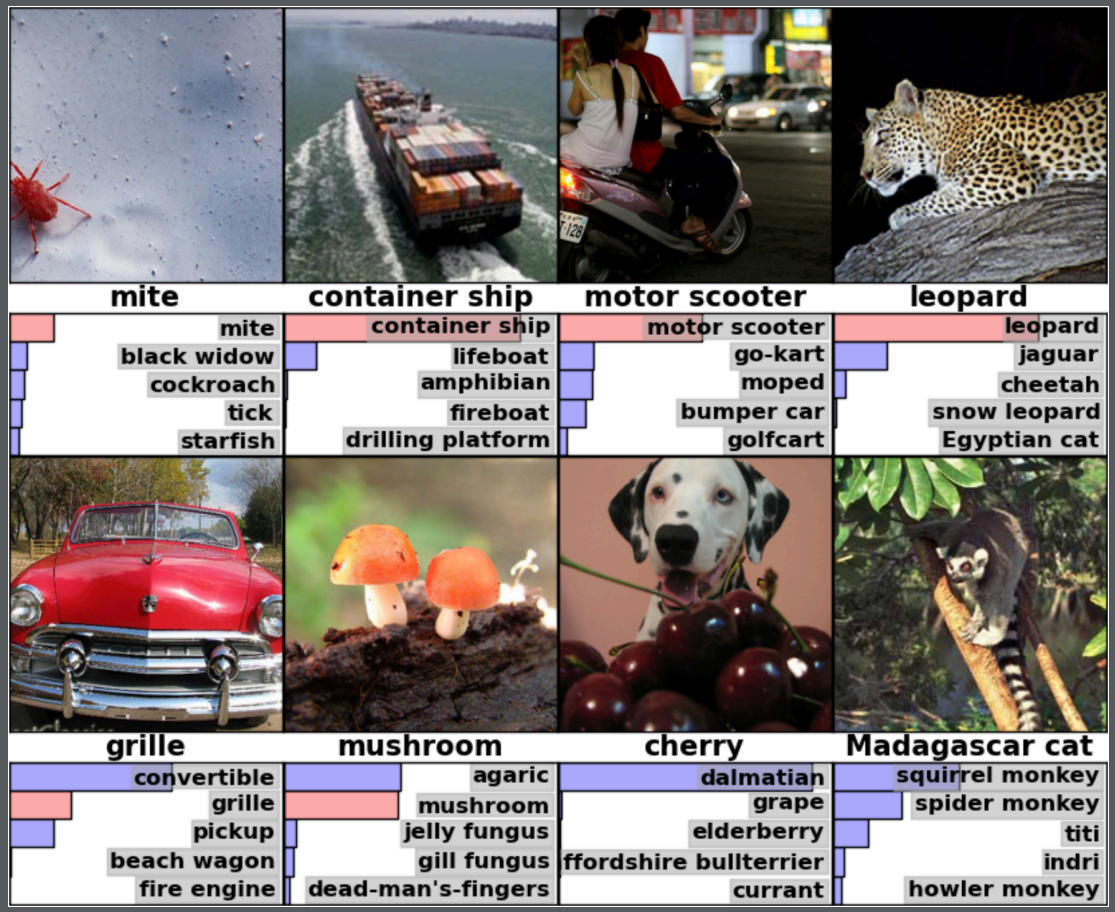
Gradient-Based Learning Applied to Document Recognition



LeNet-5 architecture

- An early ConvNet – note the two convolution layers and three fully-connected hidden layers.
- How many output filters at each convolutional layer?
- What is the effective receptive field of the fully-connected layers?

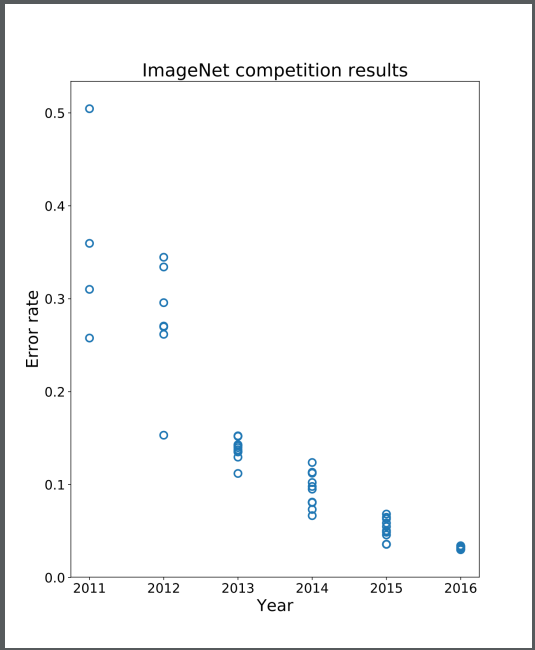
ILSVRC



Source: ImageNet Classification with Deep Convolutional Neural Networks

Top-5 task: Classifier provides 5 possible labels. Error if correct label is not in the top 5.

Starting in 2012, CNNs have been the top performers on the ILSVRC.



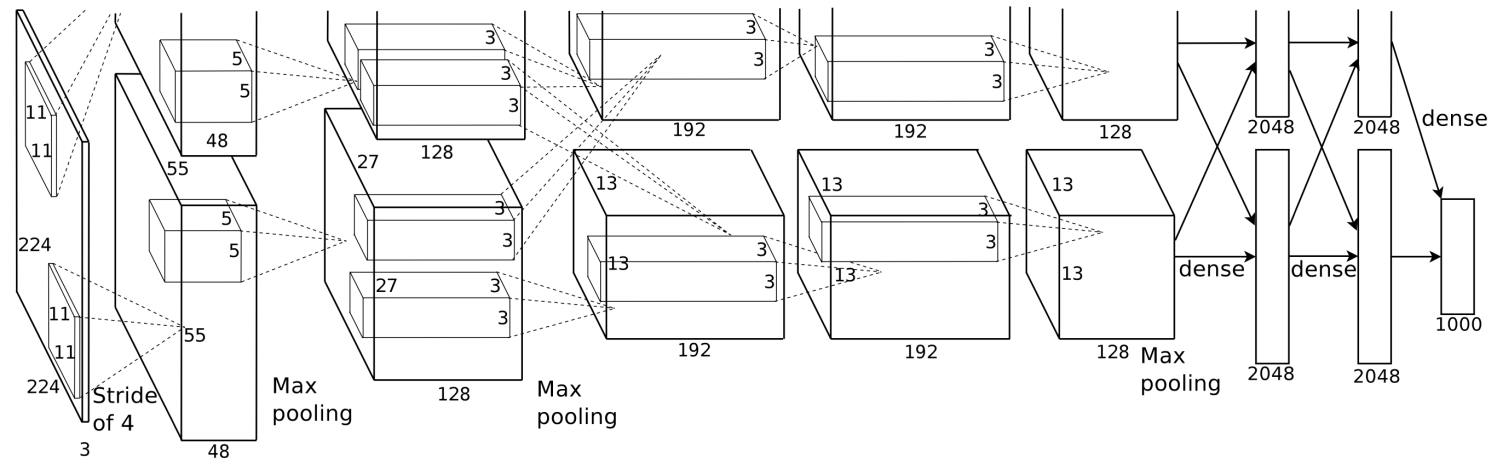
Error rate of of best model per team on top-5 classification task. Source: Wikipedia

2011: XRCE / Fisher vector
2012: AlexNet (8 layers)
2013: Zeiler / Fergus Net (8 layers)
2014: GoogLeNet (22 layers), VGG (19 layers)
2015: ResNet (152 layers)

Trends:

- Smaller convolutions (11x11 in AlexNet, 3x3 in VGG)
- More layers (8 in AlexNet, 152 in ResNet)
- 1x1 convolutions (projecting the filters to lower dimension)
- Fewer FC layers

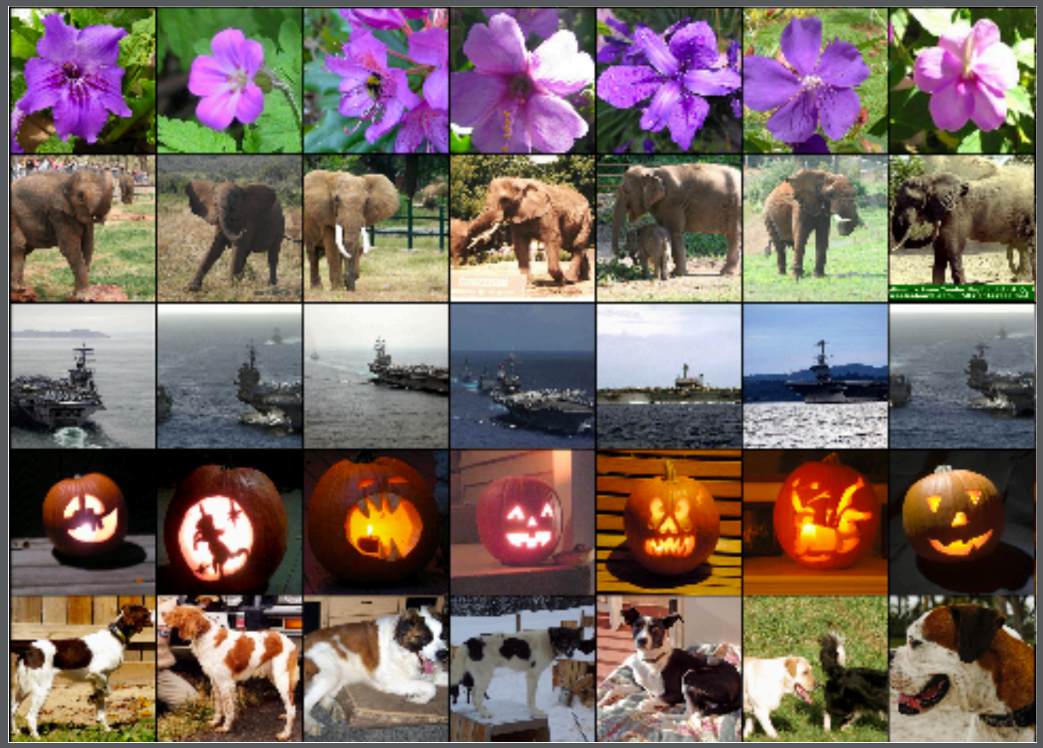
AlexNet



Source: ImageNet Classification with Deep Convolutional Neural Networks

- Won ILSVRC top-5 classification task with 15.3% error
- 5 convolutional layers, 3 FC layers
- ReLU, local response normalization (no longer common)
- Regularized with Dropout, weight decay
- Batch size 128, 90 epochs (1.2 million training images)
- 60 million parameters, 5-6 days on 2 GPUs

AlexNet



Source: ImageNet Classification with Deep Convolutional Neural Networks

Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

VGGNet (2014)

Very Deep Convolutional Networks for Large-Scale Image Recognition

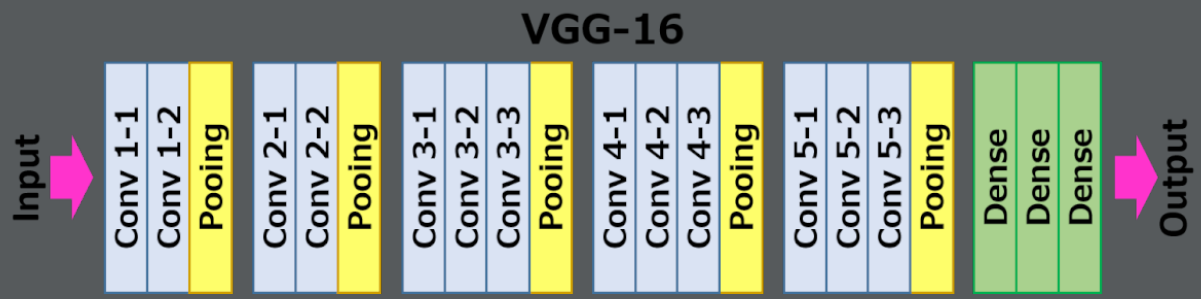
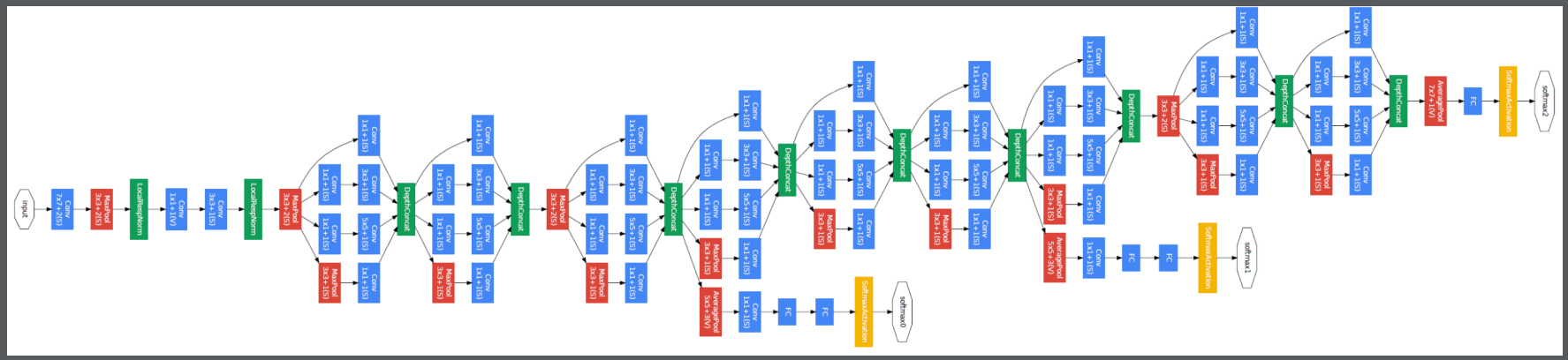


Image source: https://neurohive.io/en/popular-networks/vgg16/

- "a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters"
- Second places in the classification task in ILSVRC-2014.
- Nineteen convolutional layers, three fully-connected layers
- Filters are 3×3 throughout network

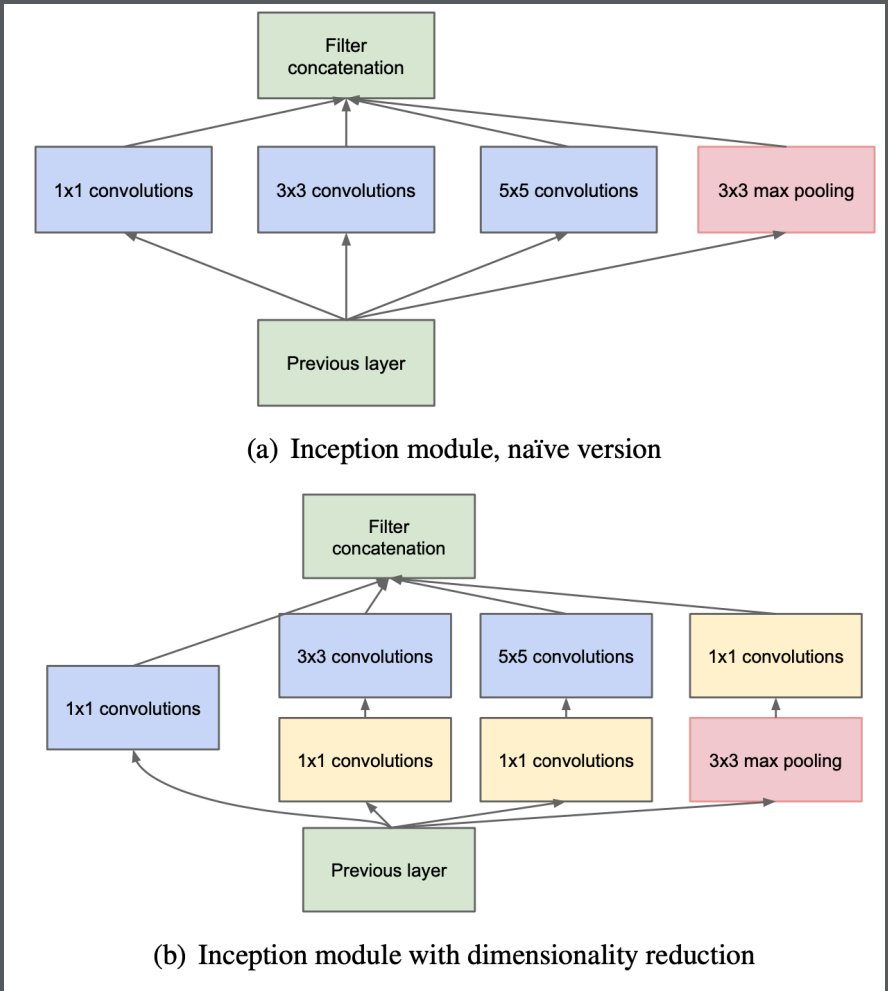
GoogLeNet 2014



Source: Going deeper with convolutions

- Very careful engineering of convolutional architecture, inspired by theoretical work of Arora et al.
- Achieves 12× parameter reduction compared to AlexNet, 1st place on top-5 classification task in 2014 (6.67% error rate)

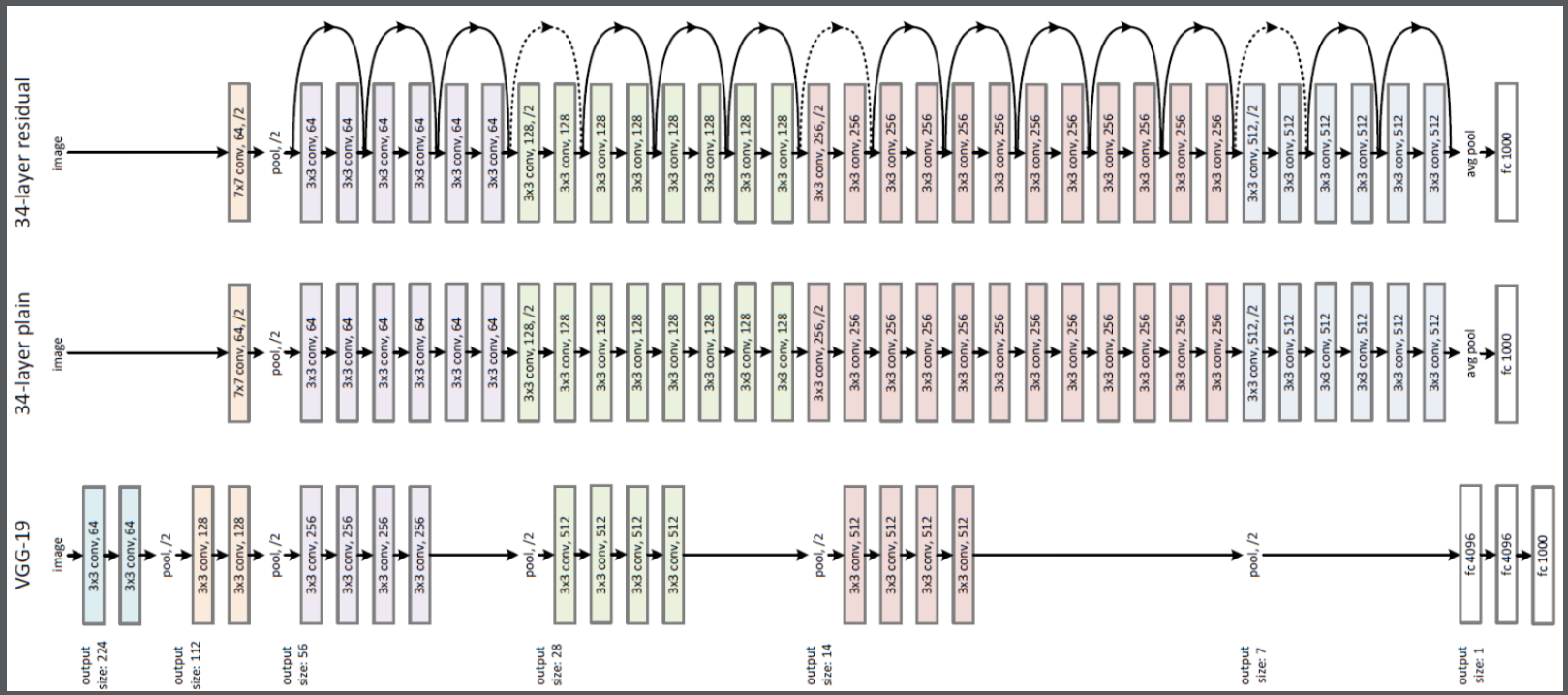
GoogLeNet 2014



Source: Going deeper with convolutions

- Concatenates filters of varying width
- Uses 1×1 convolution (i.e. dimension reduction) prior to convolution and after pooling.

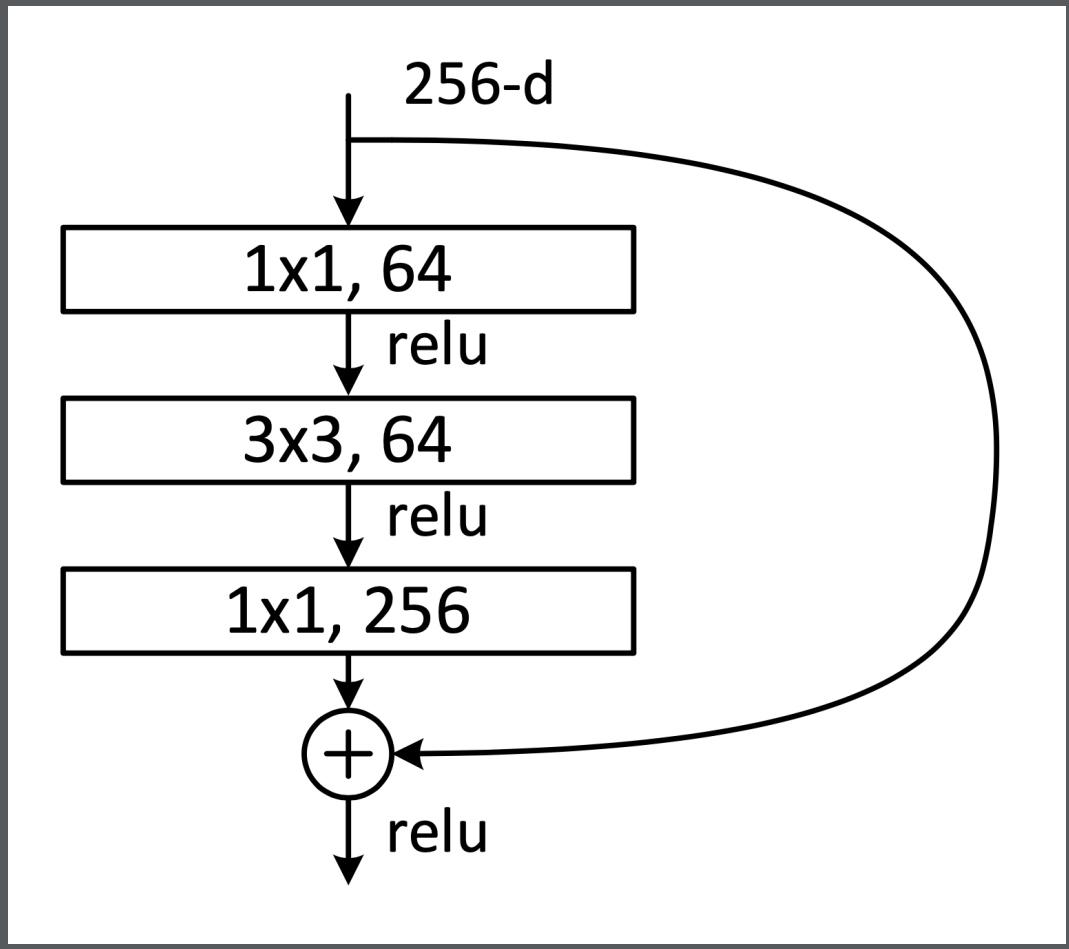
ResNet 2015



Source Deep residual learning for image recognition

- Authors observed that 'going deeper' eventually resulted in worse models - not just due to overfitting!
- Means that additional layers are not even able to the learn identity function.
- Employs 'skip connections'. Convolutional layers learn a 'residual' on previous feature maps.

ResNet (2015)

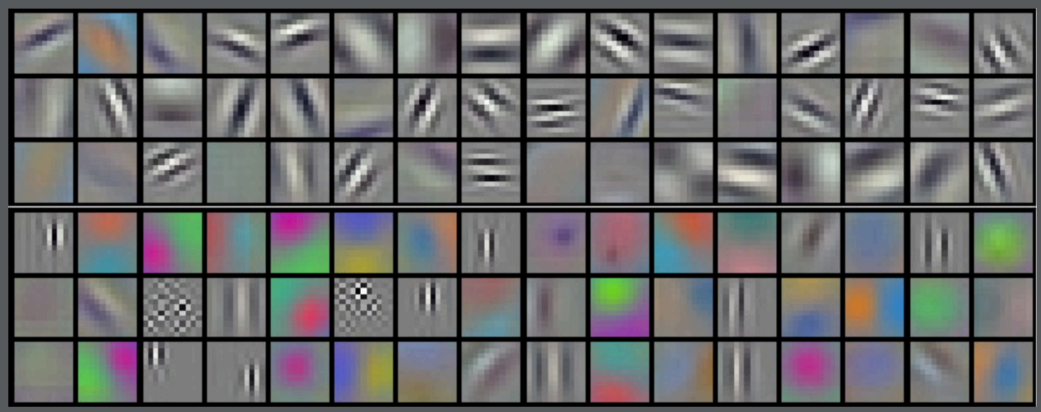


Residual learning: a building block

- 152-layer network achieved first place on the ILSVRC 2015 classification task (3.57% top-5 error)

Interpretation

First-layer filters can be direction visualized, because they occur in the image domain.



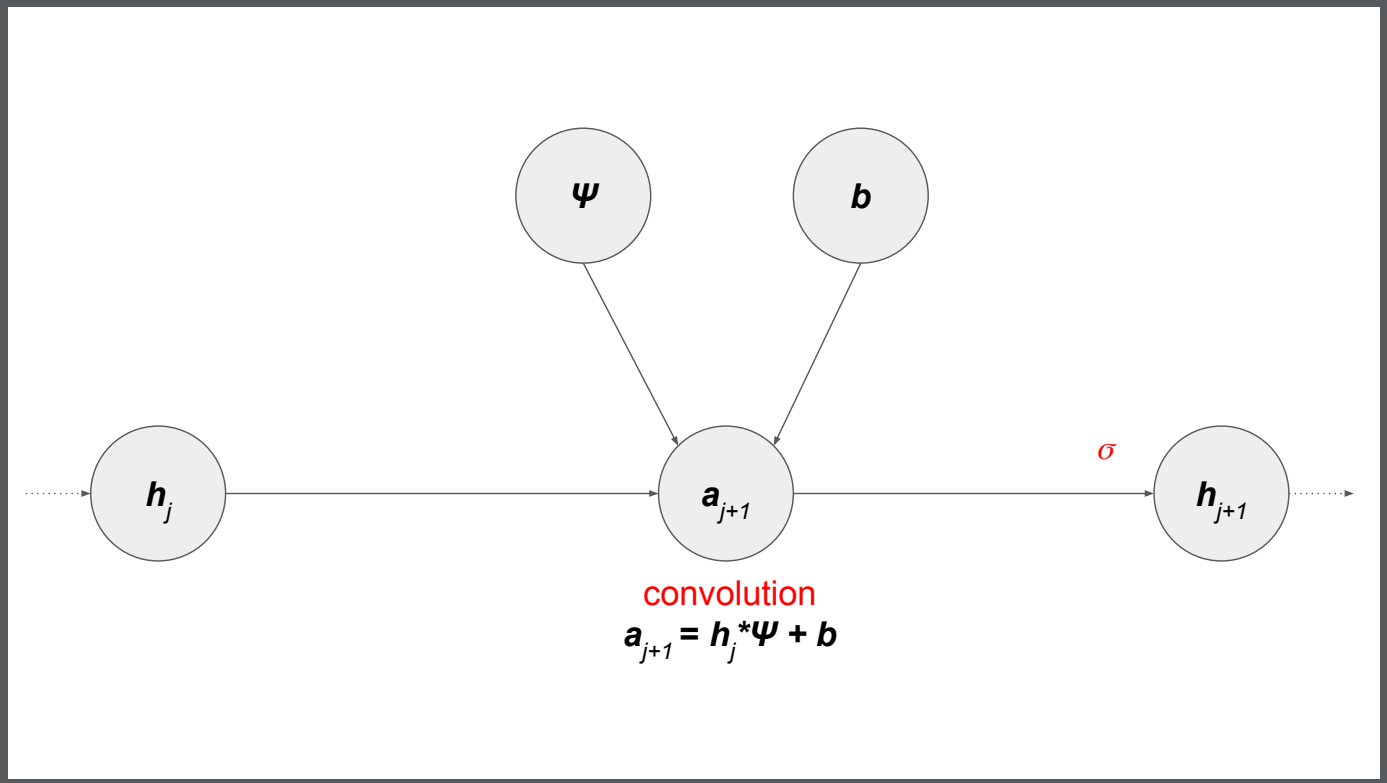
Source: ImageNet Classification with Deep Convolutional Neural Networks

Later filter banks live in the domain of the previous layer. Can't be directly visualized.

One approach: optimize the input image to increase the output of a particular neuron.

How neural networks build up their understanding of images

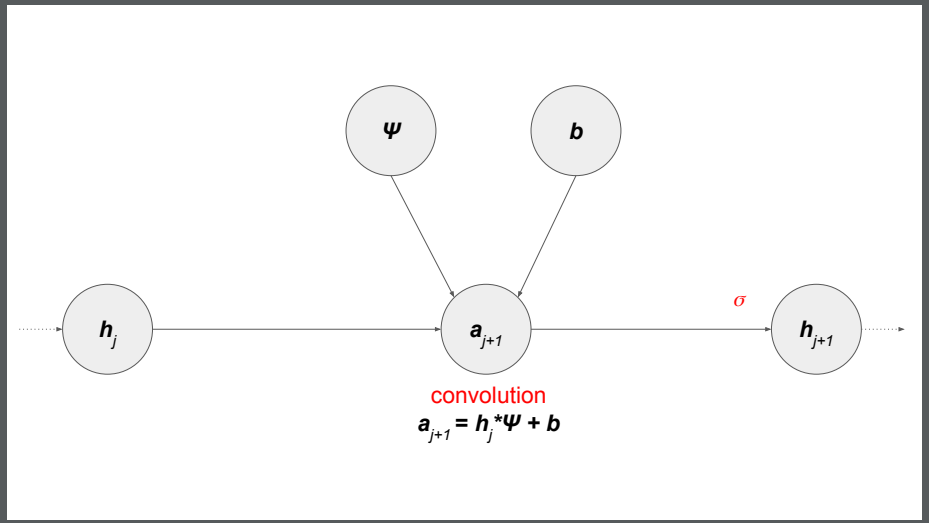
Backprop for CNNs



Consider a single-channel input and output:

$$\mathbf{h}_j, \mathbf{a}_{j+1}, \mathbf{h}_{j+1}, \mathbf{b} \in \mathbb{R}^{n \times n}$$

$\boldsymbol{\psi}$ is a $w \times w$ filter.



Input: $\nabla_{a_{j+1}} L$

$$\nabla_{\boldsymbol{\psi}} L = \mathbf{h}_j * \left(\nabla_{\mathbf{a}_{j+1}} L\right)$$
$$\nabla_{\mathbf{h}_j} L = \left(\nabla_{\mathbf{a}_{j+1}} L\right) * \boldsymbol{\psi}^{\tau}$$

Reverse operations are convolution too!

$\boldsymbol{\psi}^{\tau}$ is a 180° rotation of the filter $\boldsymbol{\psi}$ (also called the anti-diagonal transpose).

Note, this sweeps a lot of implementation detail under the hood (e.g. stride, padding, multiple channels).