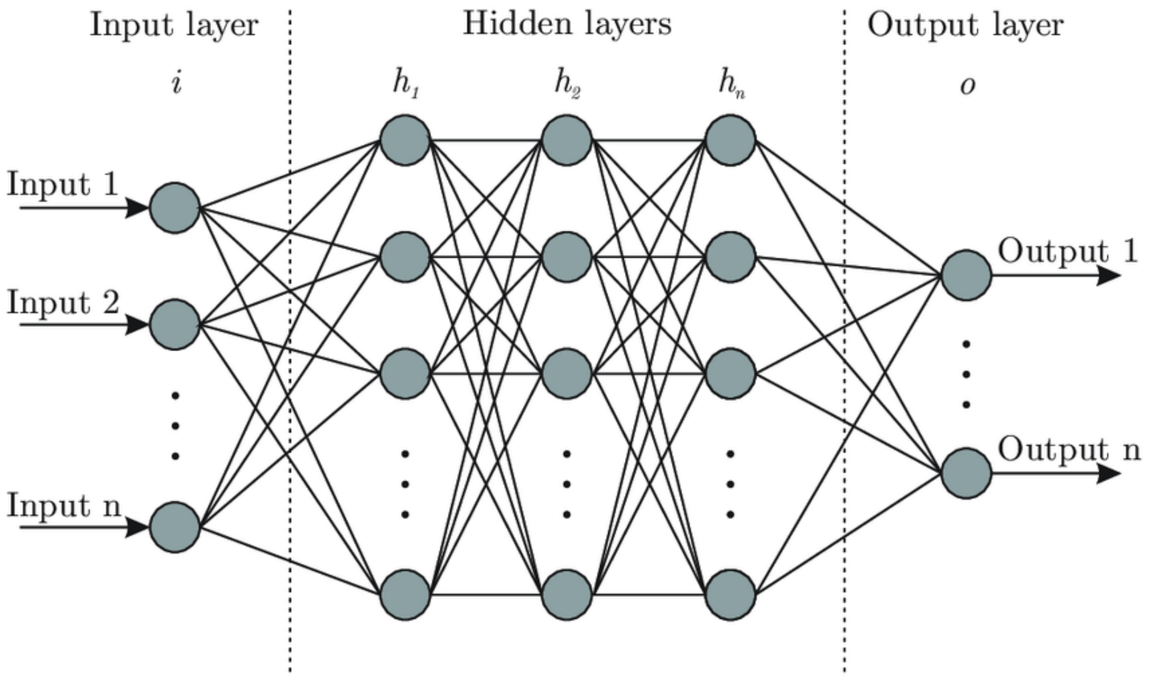
Convolutional Neural Networks

Instructor: Xiaoqian Wang
9/23/2024

Slides prepared based on the Lectures slides of Convolutional Networks from
https://www.deeplearningbook.org/lecture_slides.html

Review of MLP

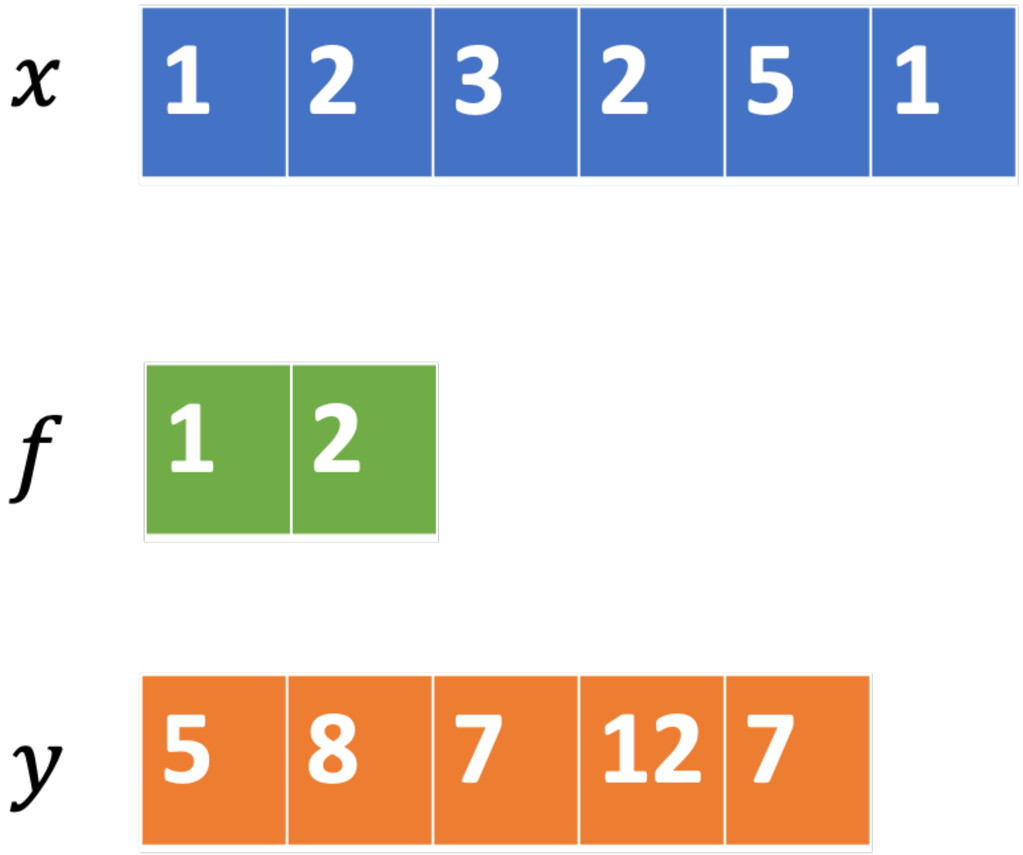
- Large number of parameters
- Disregard spatial information



Convolutional Network

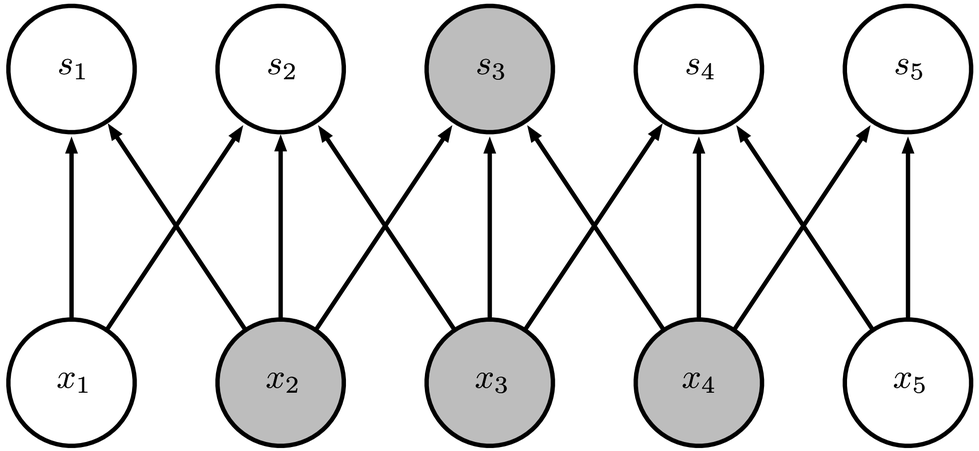
- Less parameters: scale up networks by replacing matrix multiplication via convolution
  - Sparse connections
  - Parameter sharing
- Translational Invariance
  - Convolution and Pooling
- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, …)
- Everything else stays the same
  - Maximum likelihood, Back-propagation, etc.

1D convolution



Sparse Connectivity

Sparse connections due to small convolution kernel



Dense connections

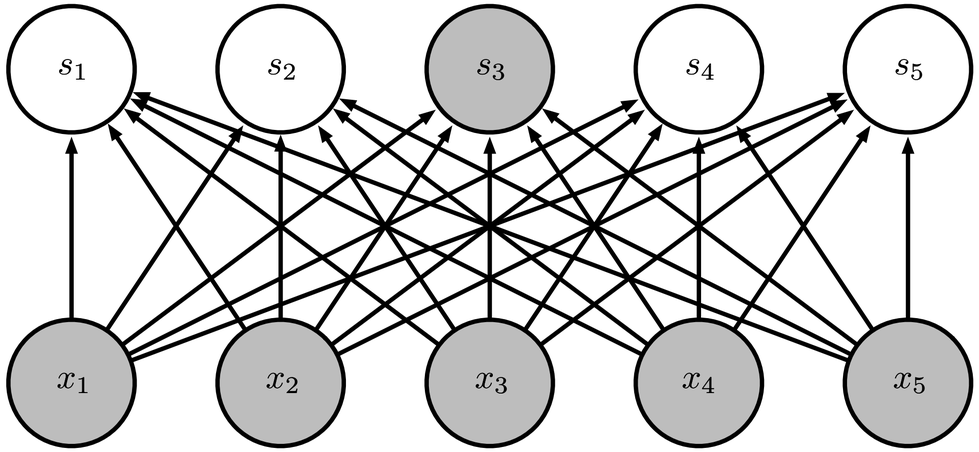


Figure 9.3

Growing Receptive Fields

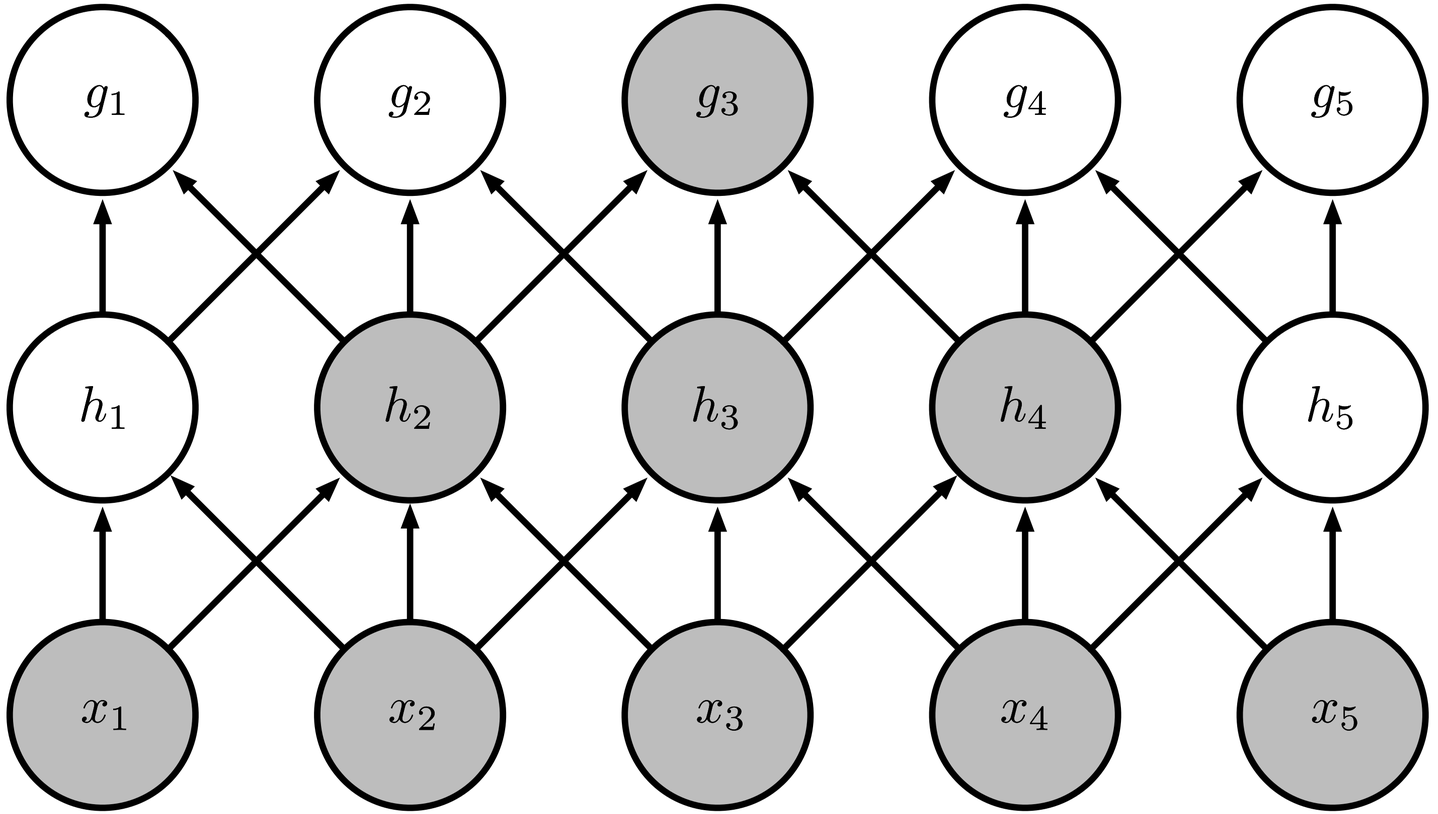
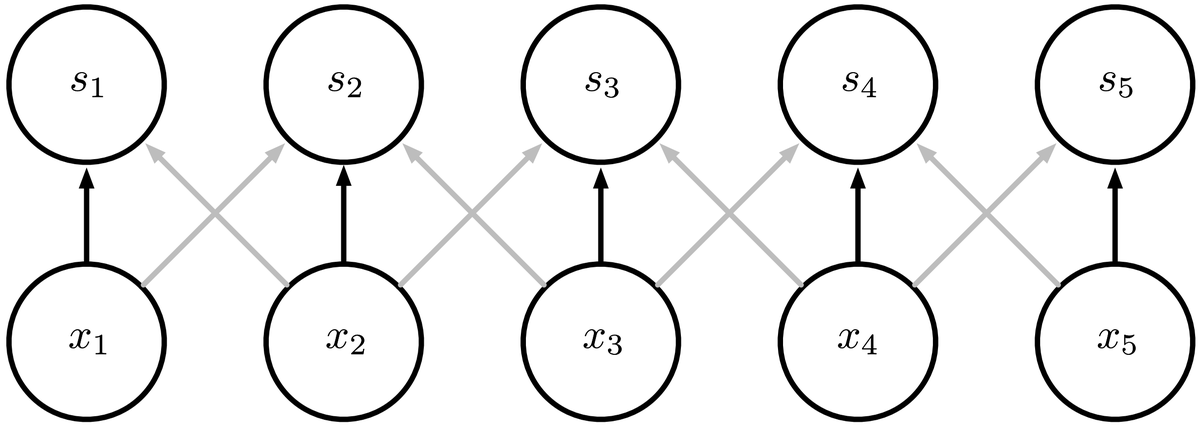


Figure 9.4

Parameter Sharing

Convolution shares the same parameters across all spatial locations



Traditional matrix multiplication does not share any parameters

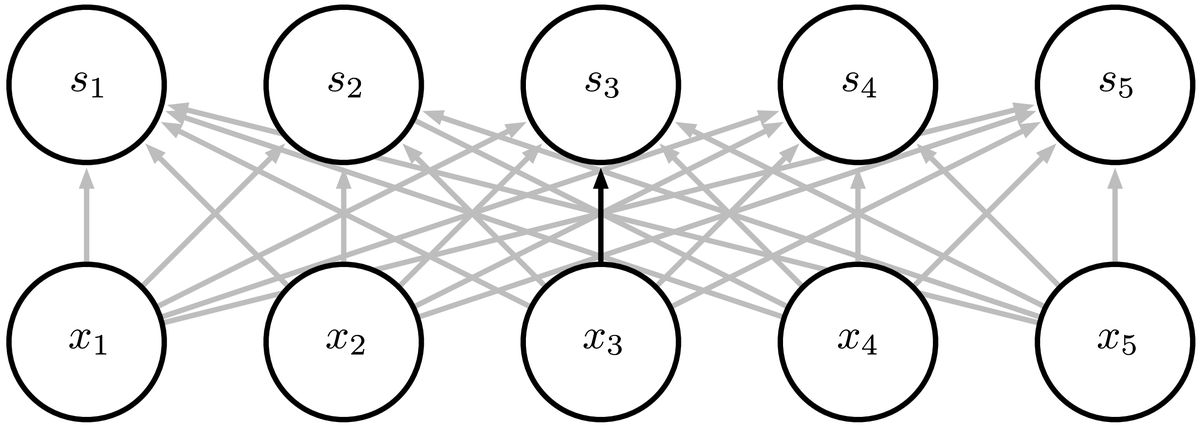


Figure 9.5

Efficiency of Convolution

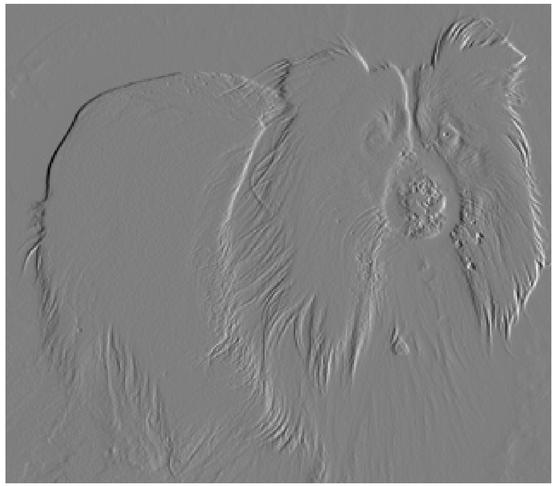
Input size: 320 by 280
Kernel size: 2 by 1
Output size: 319 by 280

	Convolution	Dense matrix	Sparse matrix
Stored floats	2	319*280*320*280 > 8e9	2*319*280 = 178,640
Float muls or adds	319*280*3 = 267,960	> 16e9	Same as convolution (267,960)

1D convolution demo

Edge Detection by Convolution

Input



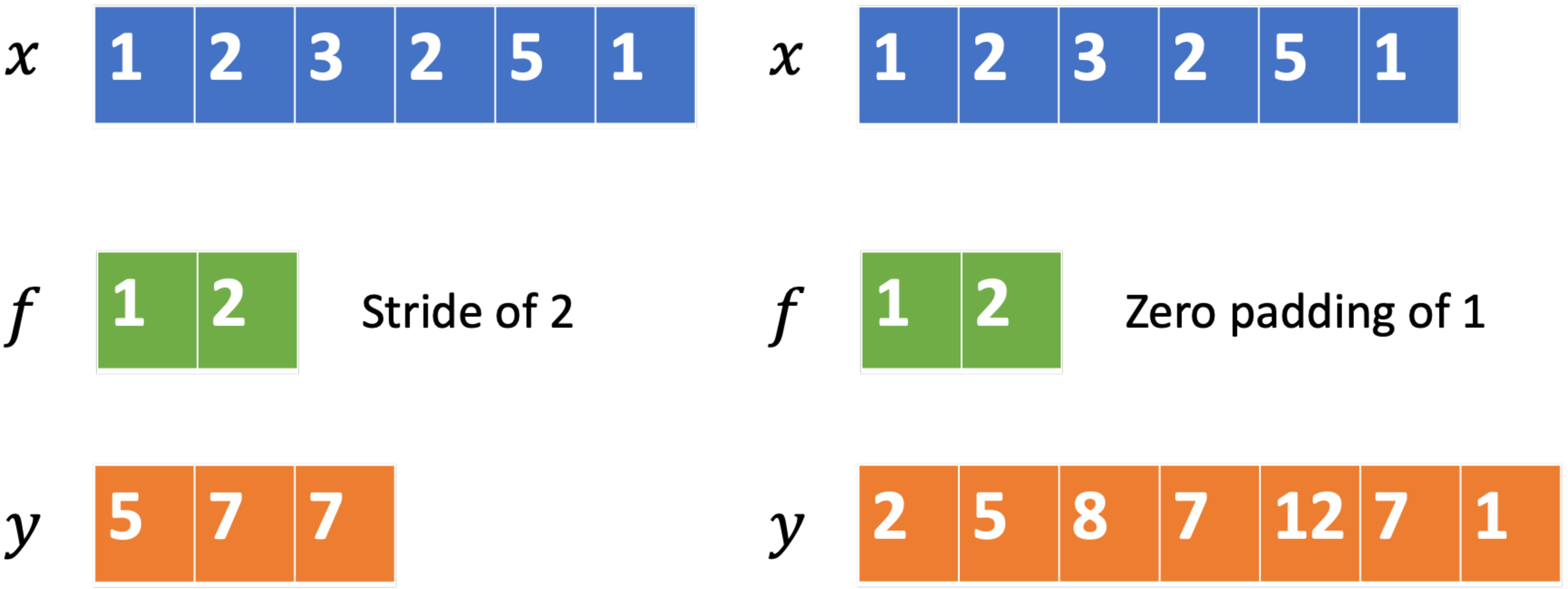
Output

1	-1

Kernel

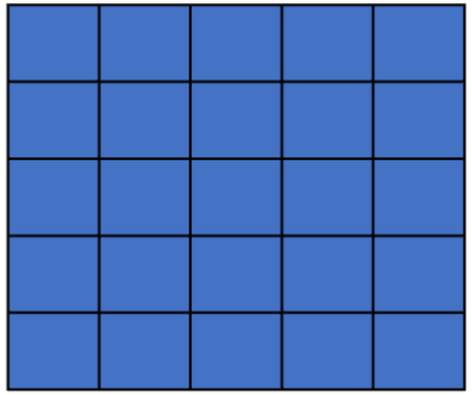
Figure 9.6

Padding or stride change output shape

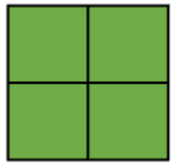


2D convolution are simple generalizations to matrices

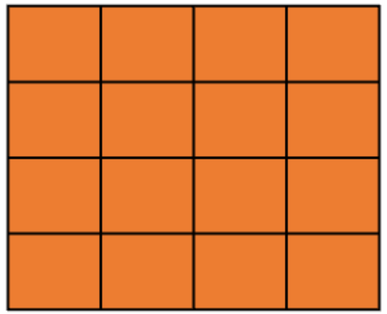
x



f

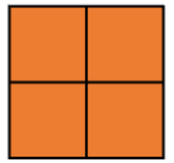


y



Stride of 2

y



demo of 2D convolution and higher dimension

Output shape in CNN

- Convolution input parameters
  - $ChannelIn = C_{in}$
  - $ChannelOut = C_{out}$ $(equivalent\ to\ \#\ filters)$
  - $KernelSize = [K_0, K_1]$
  - $Stride = [S_0, S_1]$
  - $Padding = [P_0, P_1]$
- $C_{out} = \#\ filters$

Output shape in CNN

- Convolution input parameters
  - $ChannelIn = C_{in}$
  - $ChannelOut = C_{out}$ $(equivalent\ to\ \#\ filters)$
  - $KernelSize = [K_0, K_1]$
  - $Stride = [S_0, S_1]$
  - $Padding = [P_0, P_1]$
- $C_{out} = \#\ filters$
- Output spatial dimensions
  - $H_{out} = \left\lfloor \frac{(H_{in} + 2\,P_0 - K_0)}{S_0} + 1 \right\rfloor$
  - $W_{out} = \left\lfloor \frac{(W_{in} + 2\,P_1 - K_1)}{S_1} + 1 \right\rfloor$
- Output batch dimension should match input

demo of activation function and pooling

Max Pooling and Invariance to Translation

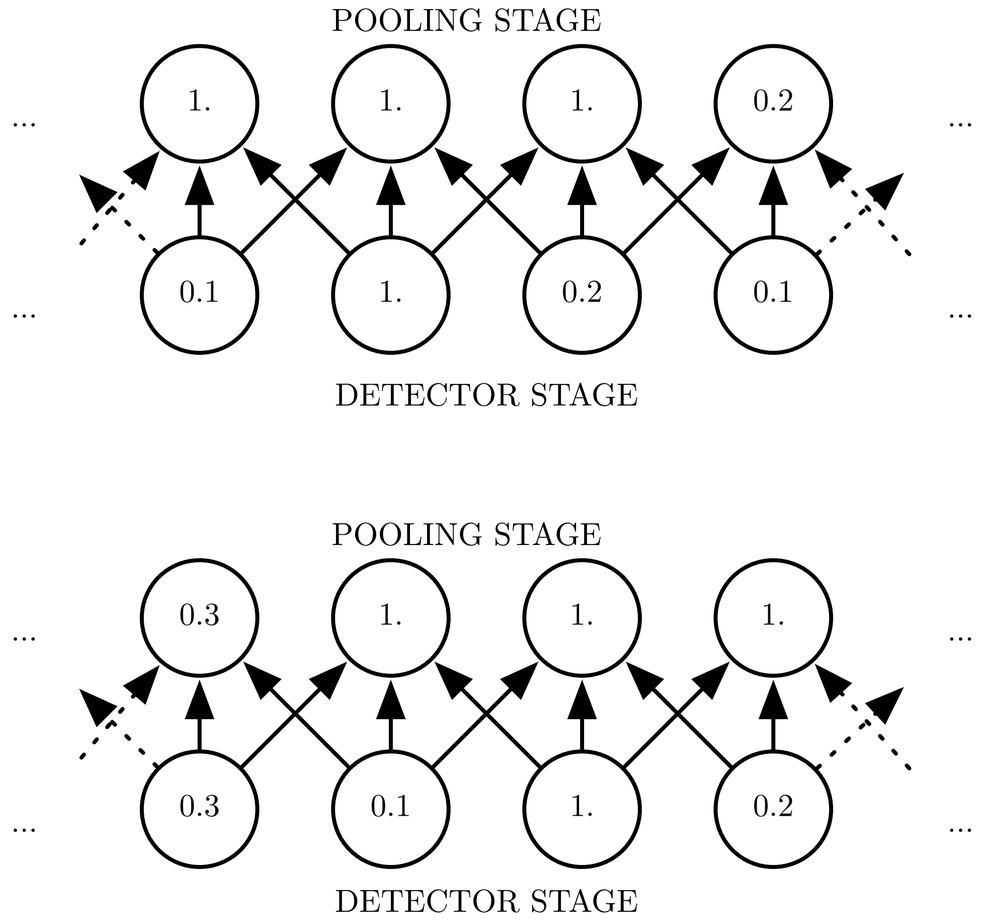


Figure 9.8

Pooling with Downsampling

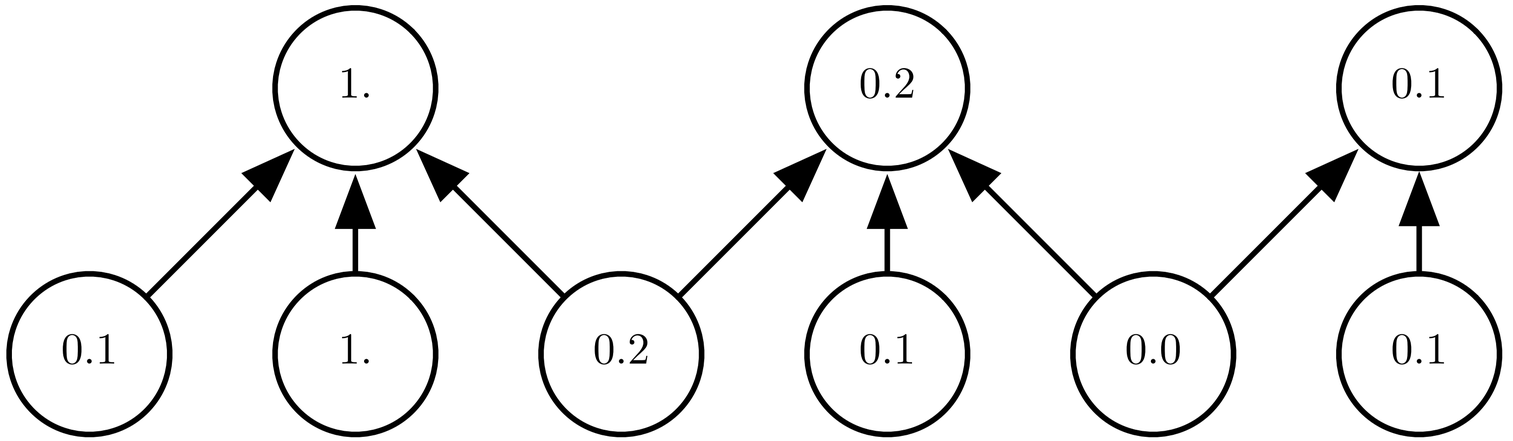


Figure 9.10

Example Classification Architectures

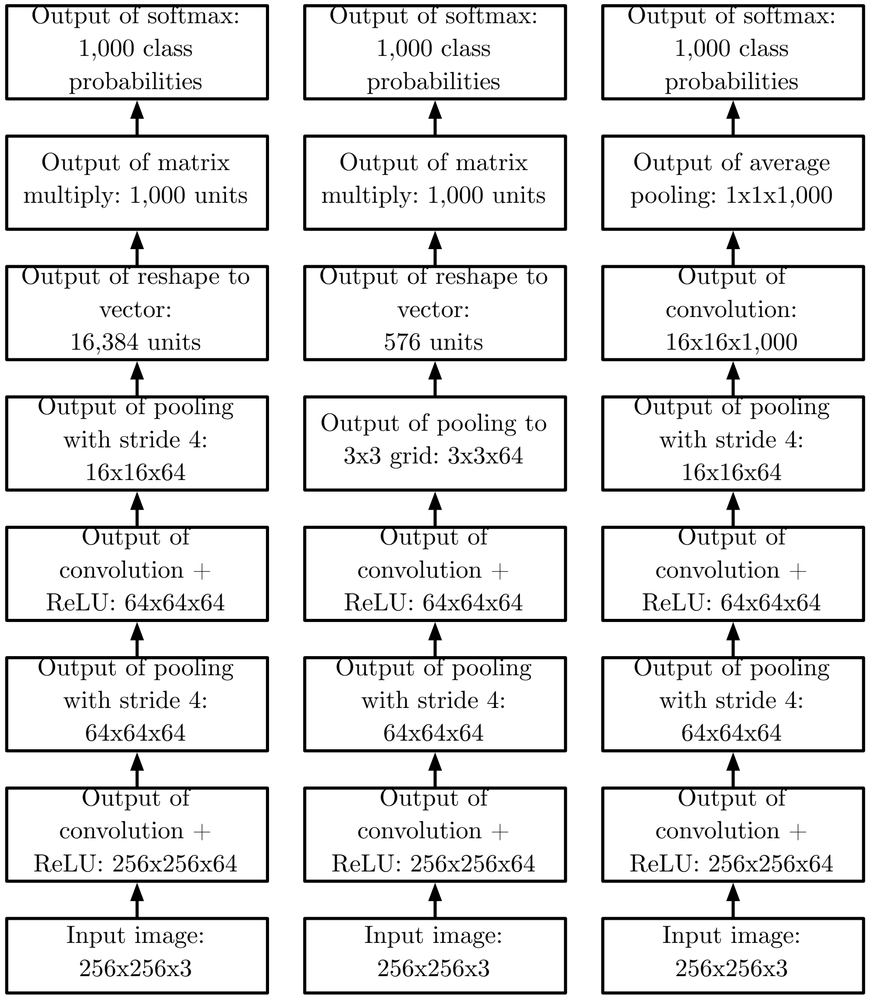


Figure 9.11

Number of parameters in CNN

- Convolution input parameters
  - $ChannelIn = C_{in}$
  - $ChannelOut = C_{out}$ $(equivalent\ to\ \#\ filters)$
  - $KernelSize = [K_0, K_1]$
  - $Stride = [S_0, S_1]$
  - $Padding = [P_0, P_1]$
- #parameters of convolutional layer = $(K_0 * K_1 * C_{in} + 1) * C_{out}$
- ReLU function doesn't have parameters
- #parameters of padding layer = 0

Convolutional Network Interpretation

- Visualization of features in different layers
  - DeCovNet (Zeiler and Fergus, 2014) https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
- Attribution
  - GradCAM (Selvaraju et al. 2017)
  - LIME (Ribeiro et al. 2016)

Training a neural network is not easy

- Training can be slow: Batch Normalization (Ioffe, Sergey, and Christian Szegedy., 2015)
  - The basic idea is to normalize **each feature** in an input batch of each layer during the **forward pass**

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Training a neural network is not easy

- Batch Normalization (Ioffe, Sergey, and Christian Szegedy., 2015)
  - In the testing phase (no mini batch), use running average of mean and variance, t is iteration index:

$$\mu_{run}^{t} = \lambda \mu_{run}^{t-1} + (1-\lambda)\mu_{B}^{t}$$
$$\sigma^{2}{}_{run}^{t} = \lambda \sigma^{2}{}_{run}^{t-1} + (1-\lambda)\sigma^{2}{}_{B}^{t}$$

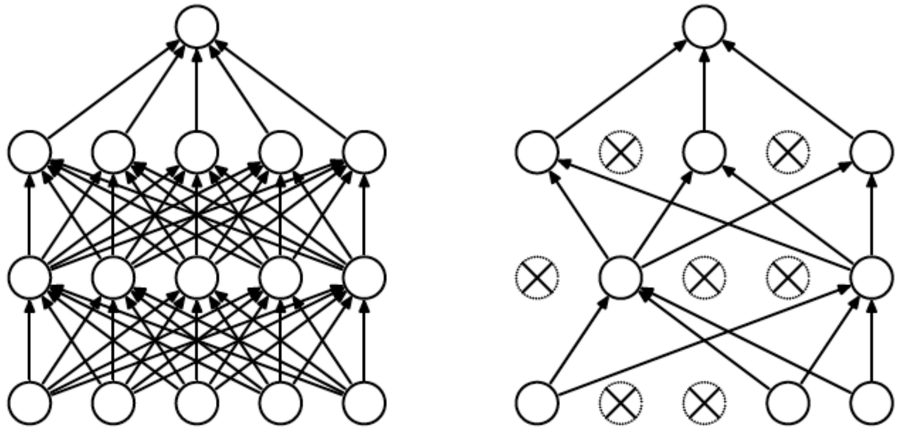
  - For CNNs, the channel dimension is treated as a “feature”

Training a neural network is not easy

- Batch Normalization can stabilize and accelerate training of deep models, and producing better models
- Not fully understood why it works
- Recommended reading:
  - Santurkar, Shibani, et al. "How does batch normalization help optimization?." Advances in neural information processing systems 31 (2018).

Training a neural network is not easy

- Overfitting
  - Dropout (Srivastava, Nitish, et al., 2014)
  - Weight decay
  - Early stopping



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Large, Shallow Models Overfit More

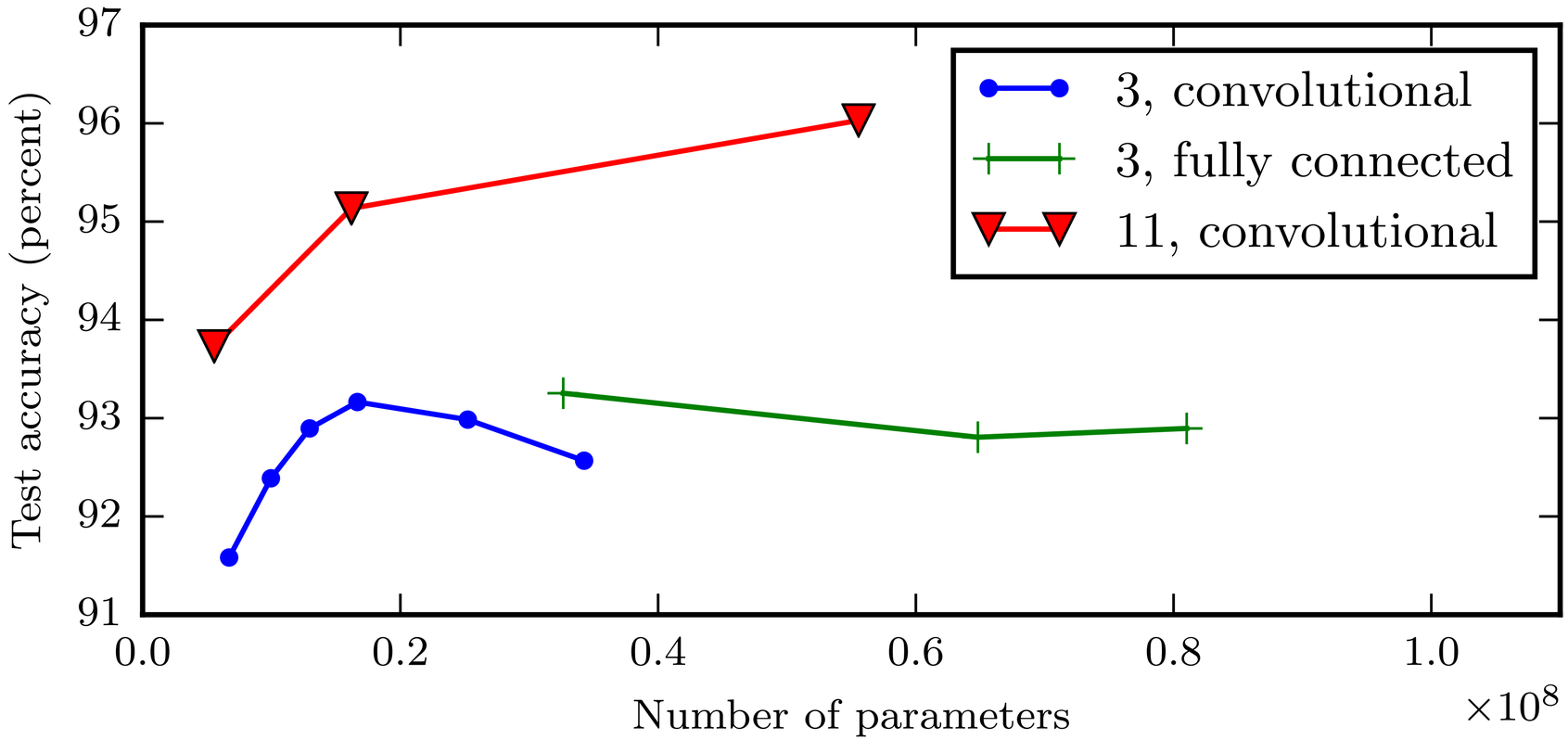


Figure 6.7

Slides from (Goodfellow 2017)

Training a neural network is not easy

- Residual network (ResNet) enables deeper networks because gradient information can flow between layers
  - Residual layers add back in the input $y = f(x)+x$
  - Notice that $f(x)$ models the difference between x and y (hence the name residual)
  - If the residuals = 0, then this is merely the identity function
  - Illustration image at link

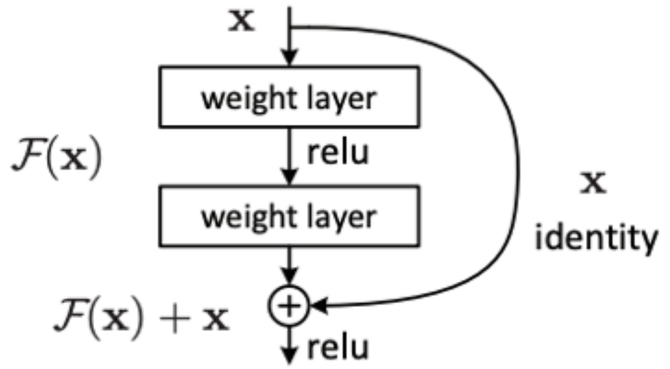


Figure 2. Residual learning: a building block.

Images from: He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Details of ResNet

- If the dimensionality is not the same, then use either fully connected layer or convolution layer to match

Images from: He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Details of ResNet

- If the dimensionality is not the same, then use either fully connected layer or convolution layer to match
  - In the 1D case, suppose $f(x): \mathbb{R}^d \rightarrow \mathbb{R}^m$, then we need to multiply x by linear operator to match the dimension

$$y = f(x) + Wx, \qquad \text{where } W \in \mathbb{R}^{m \times d}$$

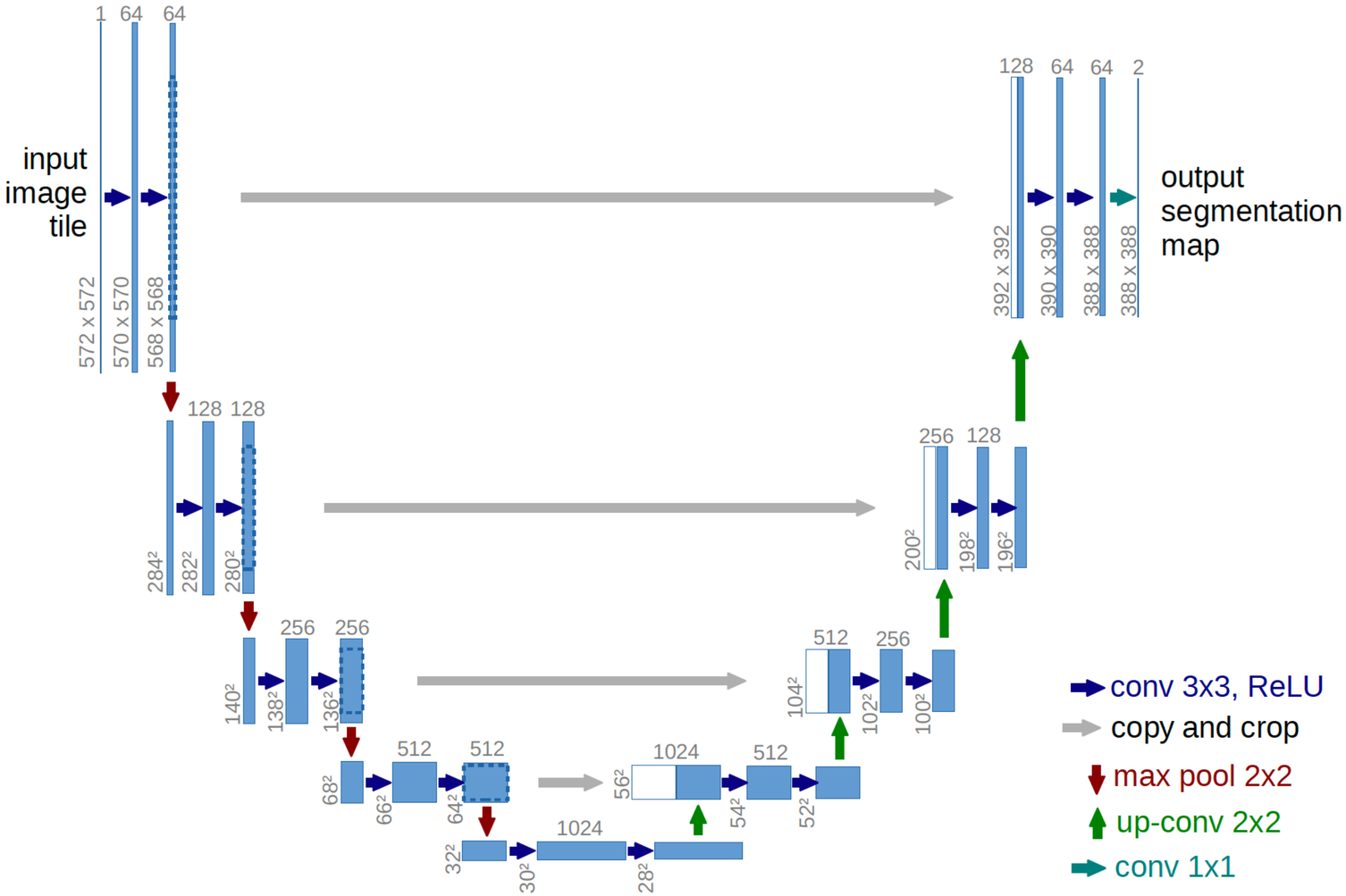
  - Similarly, for images, if $f(x): \mathbb{R}^{c \times h \times w} \rightarrow \mathbb{R}^{c' \times h' \times w'}$, we can apply a convolution layer to match the dimensions

$$y = f(x) + \text{conv}(x),$$
$$\text{where conv}(\cdot): \mathbb{R}^{c \times h \times w} \rightarrow \mathbb{R}^{c' \times h' \times w'}$$

Images from: He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Demo of dropout, batchnorm, resnet

U-Net



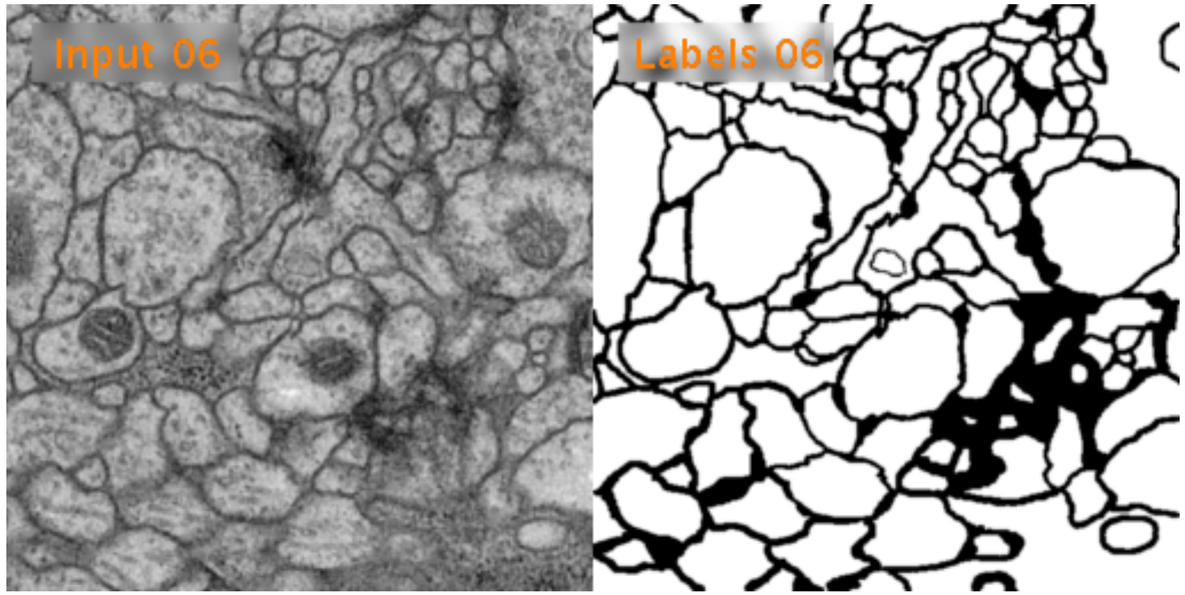
- Implementation Code: https://github.com/IntelAI/unet/blob/master/2D/model.py

U-Net

- Fully convolutional network
- A u-shaped architecture
  - a **contracting path** to capture context
  - a symmetric **expanding path** that enables precise localization
- Data augmentation

Biomedical Image Segmentation

- Assigning class label to each pixel



ISBI challenge on segmentation of neuronal structures in electron microscopy (EM) stacks https://imagej.net/events/isbi-2012-segmentation-challenge