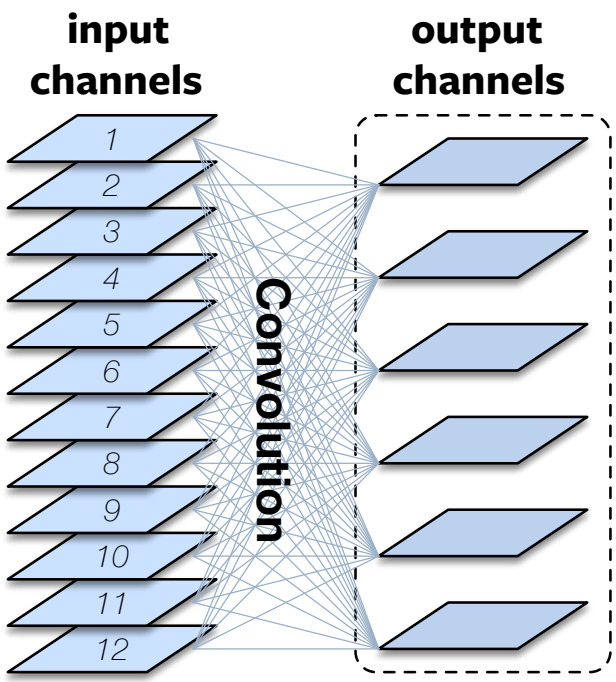
Advanced Topics in Machine Learning: Convolutional Networks (Part 2)

Laurens van der Maaten and Anton Bakhtin

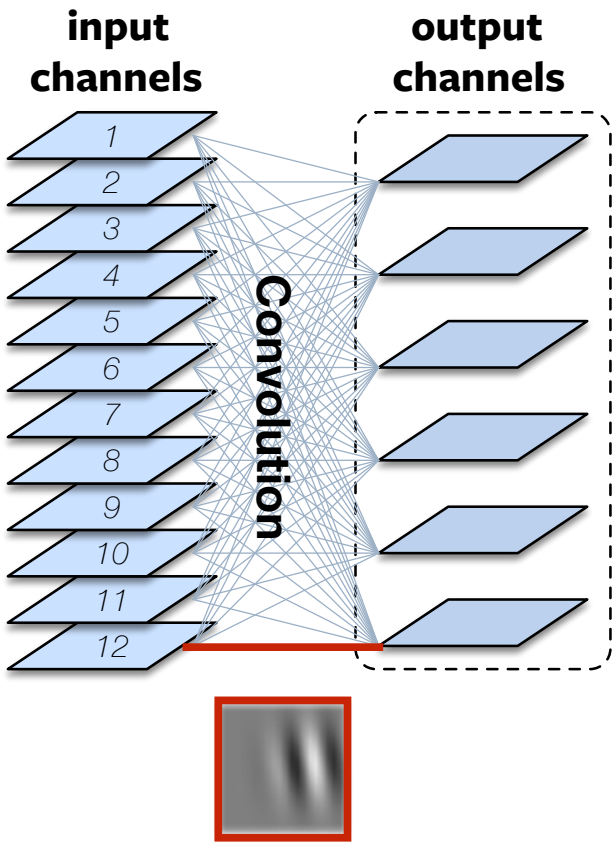
Recap on Convolutions

- Convolutional layers have five main hyperparameters:
  - Number of input channels
  - Number of output channels
  - Kernel size
  - Kernel stride
  - Input padding



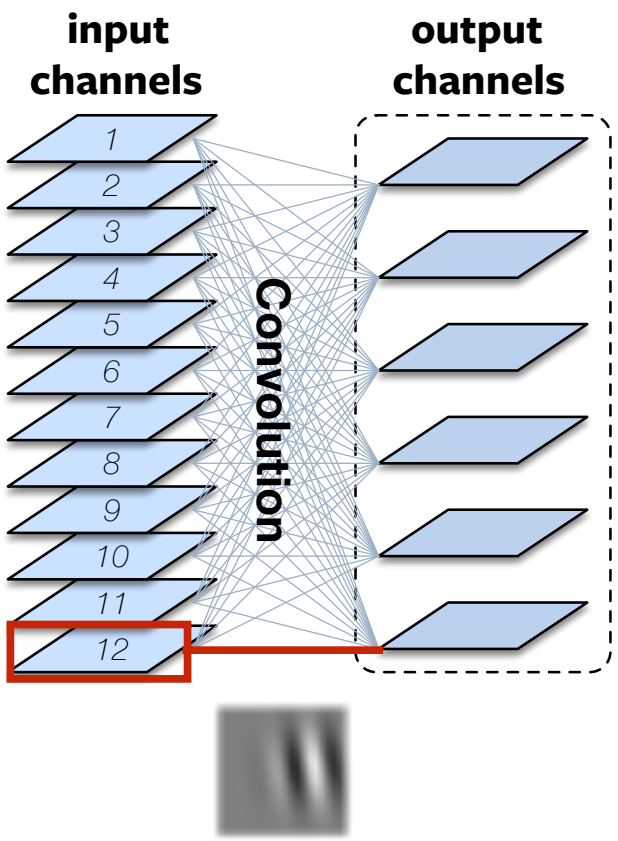
Recap on Convolutions

- Convolutional layers have five main hyperparameters:
  - Number of input channels
  - Number of output channels
  - Kernel size
  - Kernel stride
  - Input padding



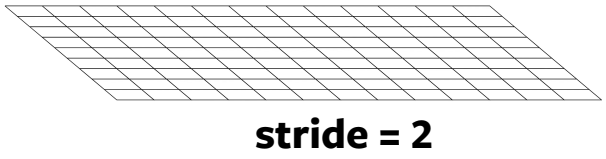
Recap on Convolutions

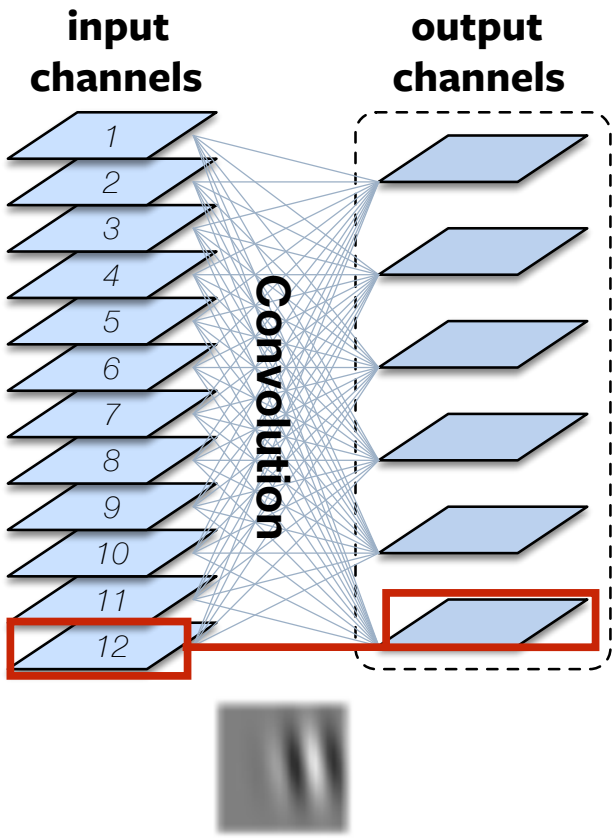
- Convolutional layers have five main hyperparameters:
  - Number of input channels
  - Number of output channels
  - Kernel size
  - Kernel stride
  - Input padding

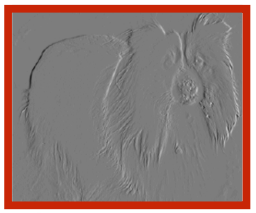


Recap on Convolutions

- Convolutional layers have five main hyperparameters:
  - Number of input channels
  - Number of output channels
  - Kernel size
  - Kernel stride
  - Input padding

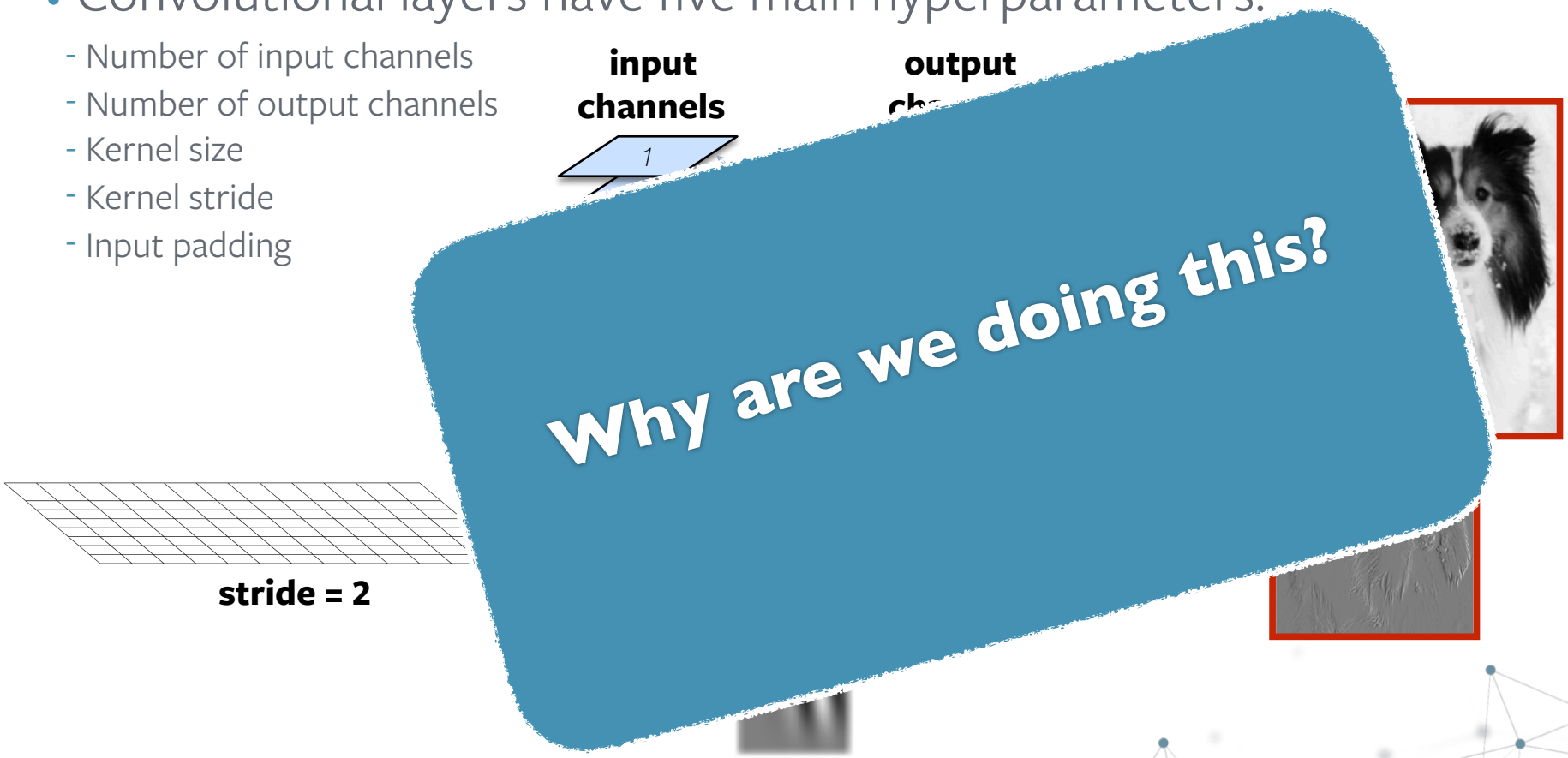






Recap on Convolutions

- Convolutional layers have five main hyperparameters:
  - Number of input channels
  - Number of output channels
  - Kernel size
  - Kernel stride
  - Input padding

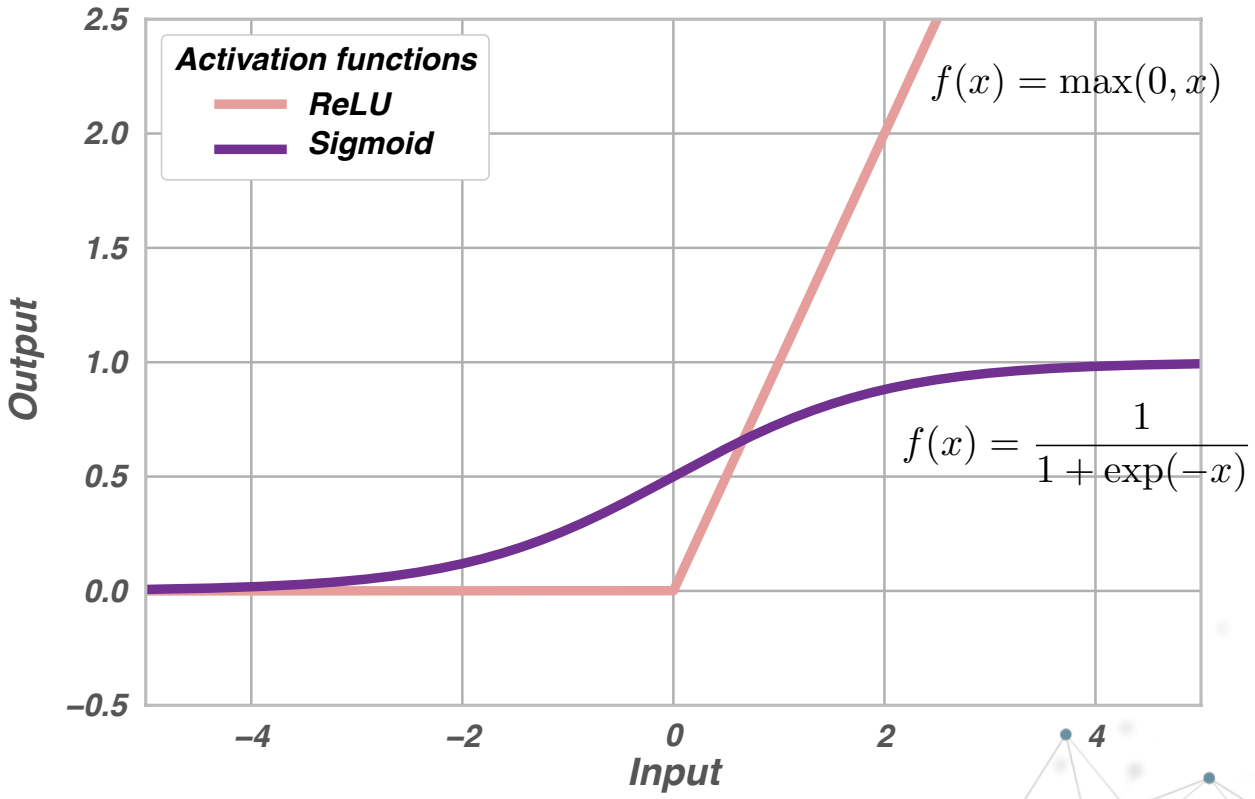


Recap on Convolutions

- Convolutional layers have five main hyperparameters:
  - Number of input channels
  - Number of output channels
  - Kernel size
  - Kernel stride
  - Input padding

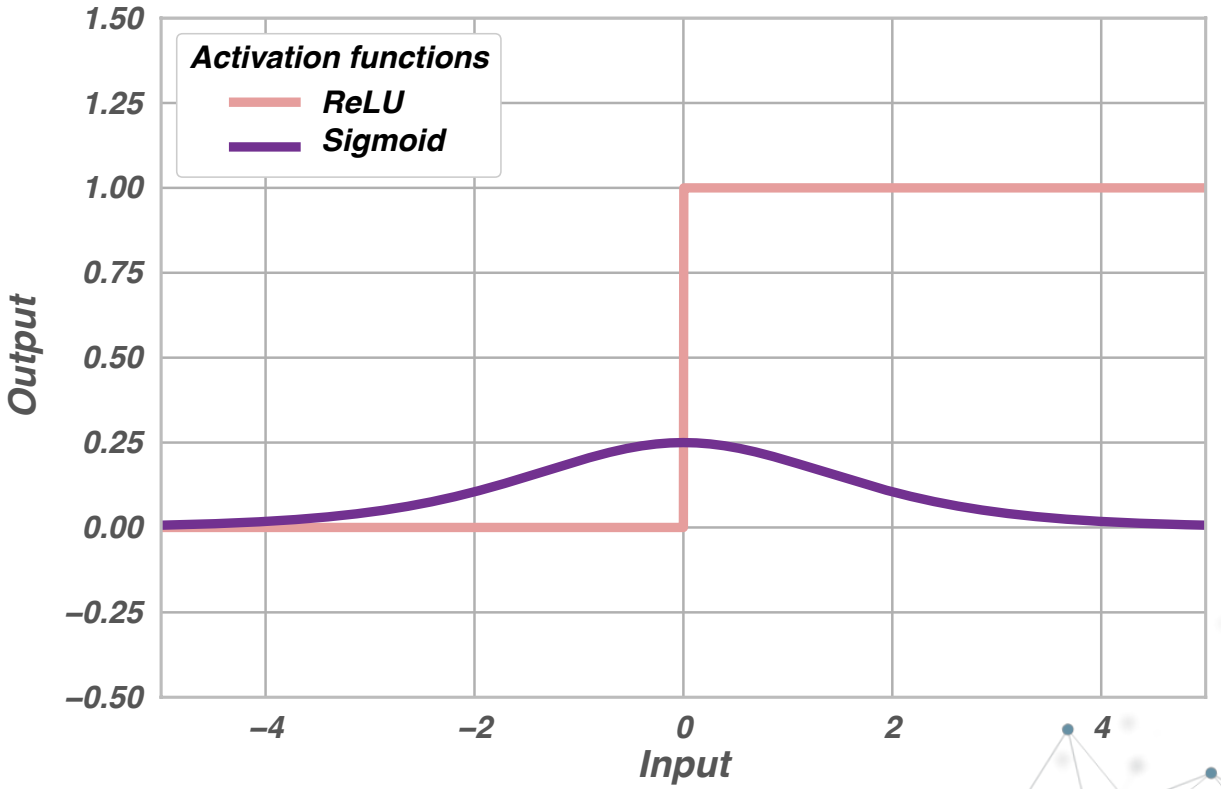
Rectified linear units

- **Rectified linear units** (ReLUs) have "better" gradients than sigmoids:



Rectified linear units

- **Rectified linear units** (ReLUs) have "better" gradients than sigmoids:

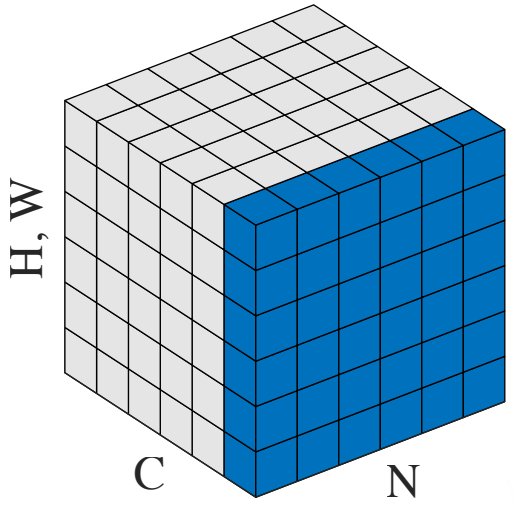


Batch normalization

- When using ReLU, gradients may easily blow up or go to zero
- **Batch normalization** make gradients better behaved:

Z-score input: $\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma}$

Affine transform: $\mathbf{y} = \gamma\hat{\mathbf{x}} + \beta$



* Note: Batch normalization works differently in training and test time.

Batch normalization

- Makes network learning more stable and produces better final models:

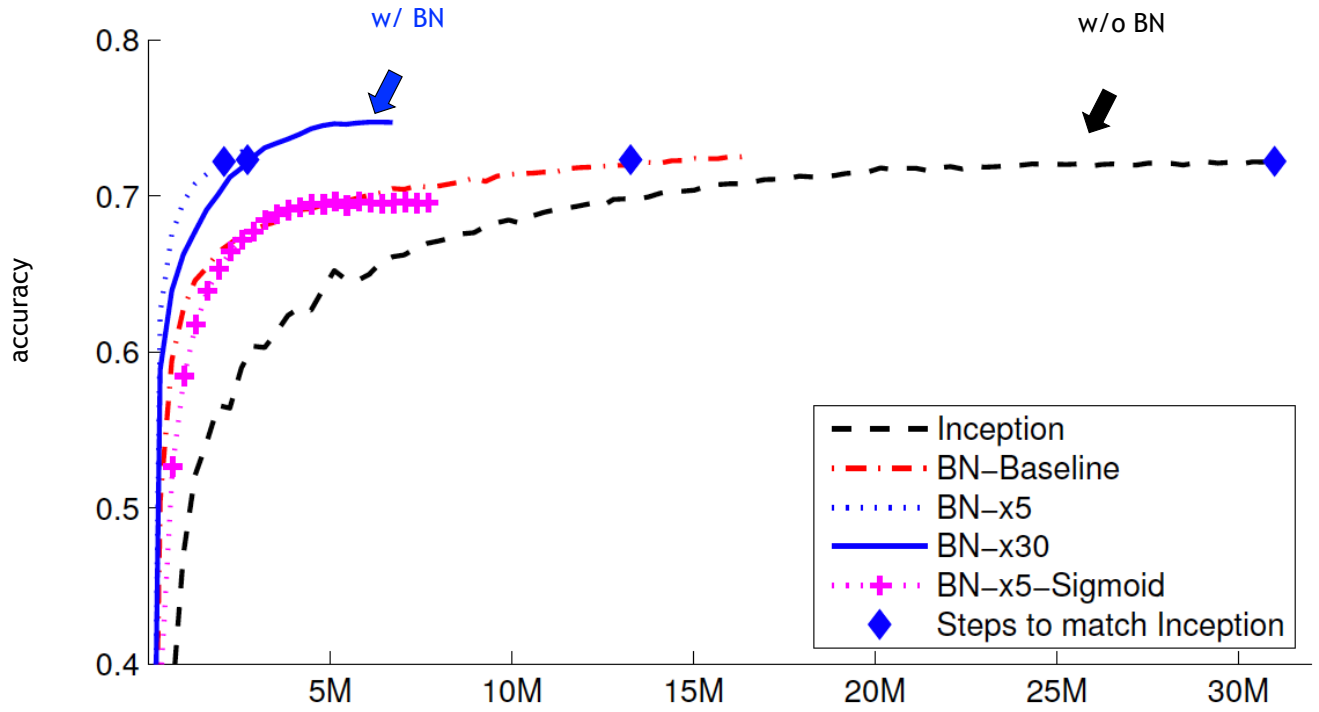
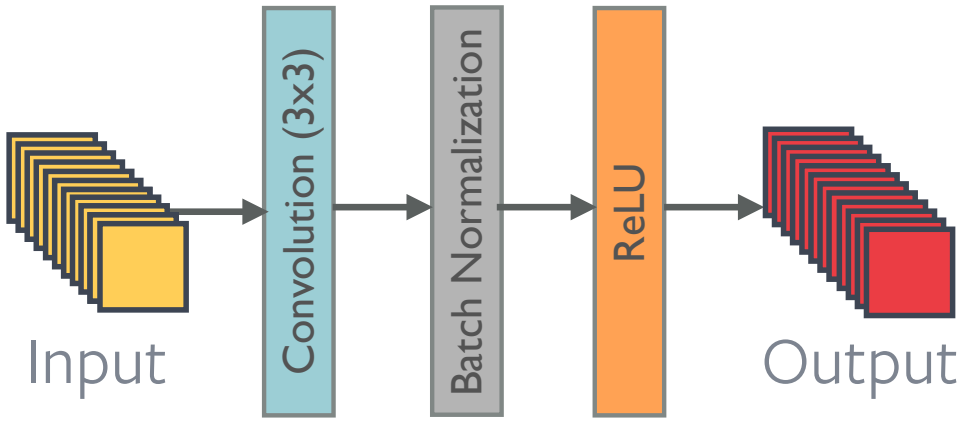


Figure credit: Ioffe & Szegedy

Building block

- Common building block comprises Conv2D + BatchNorm + ReLU:



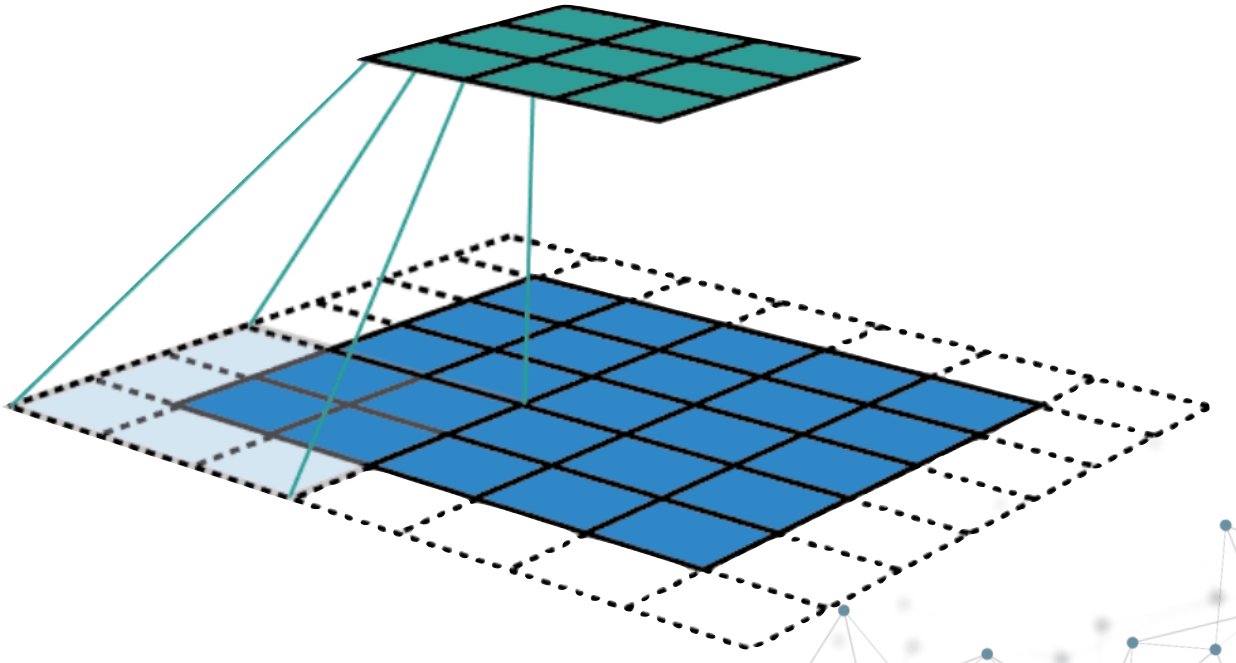
Filter sizes

- The **receptive field** of a feature is the part of the input that "influenced" the feature

Filter sizes

- The **receptive field** of a feature is the part of the input that "influenced" the feature:

Feature map

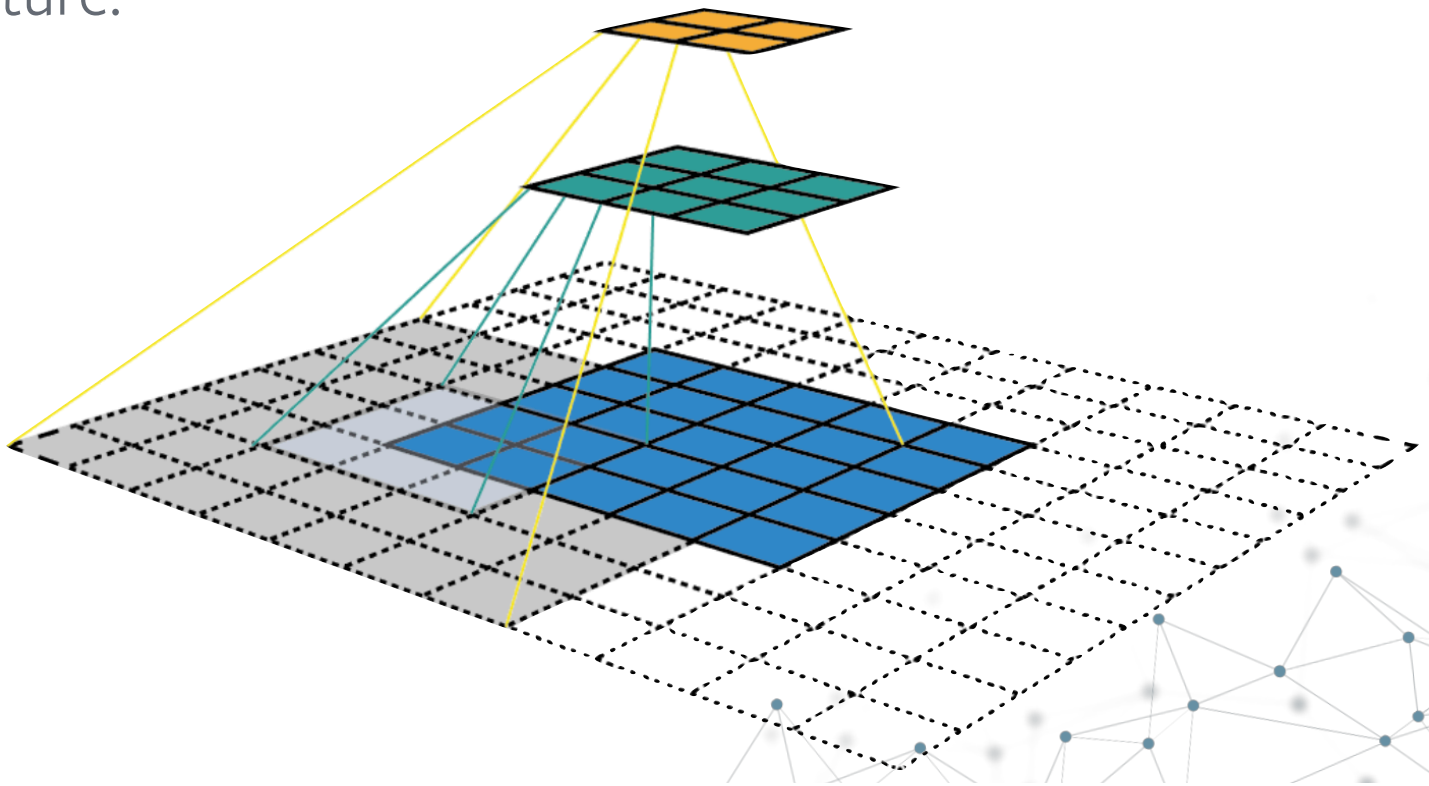


Input image

* Credits: Dang Ha The Hien

Filter sizes

- The **receptive field** of a feature is the part of the input that "influenced" the feature:



* Credits: Dang Ha The Hien

Filter sizes

- What is the receptive field of one **5x5 filter**?
  - And how much compute does it take?

Filter sizes

- What is the receptive field of one **5x5 filter**?
  - And how much compute does it take?

- What is the receptive field of **two 3x3 filters**?
  - How much compute does that take?

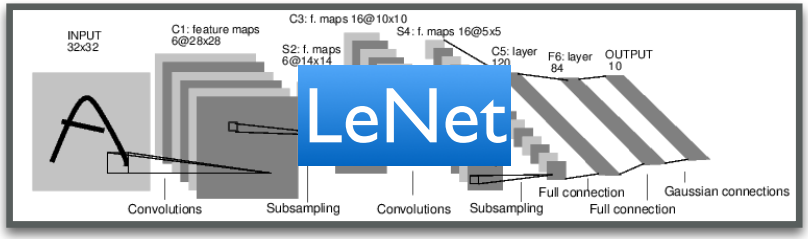
Filter sizes

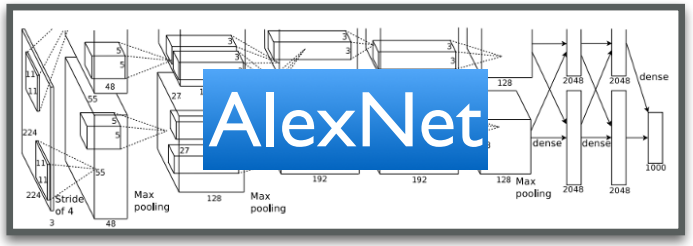
- What is the receptive field of one **5x5 filter**?
  - And how much compute does it take?

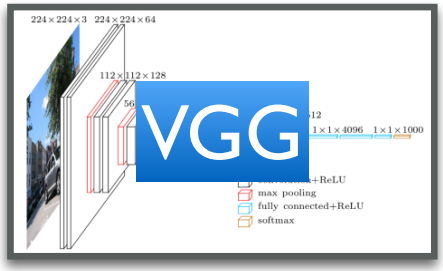
- What is the receptive field of **two 3x3 filters**?
  - How much compute does that take?

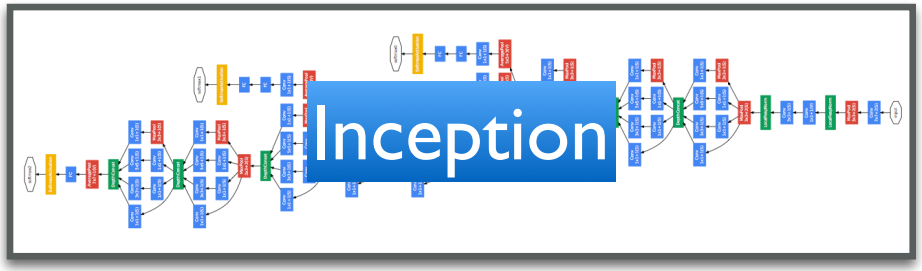
- Many current architecture use primarily **3x3 filters** for this reason

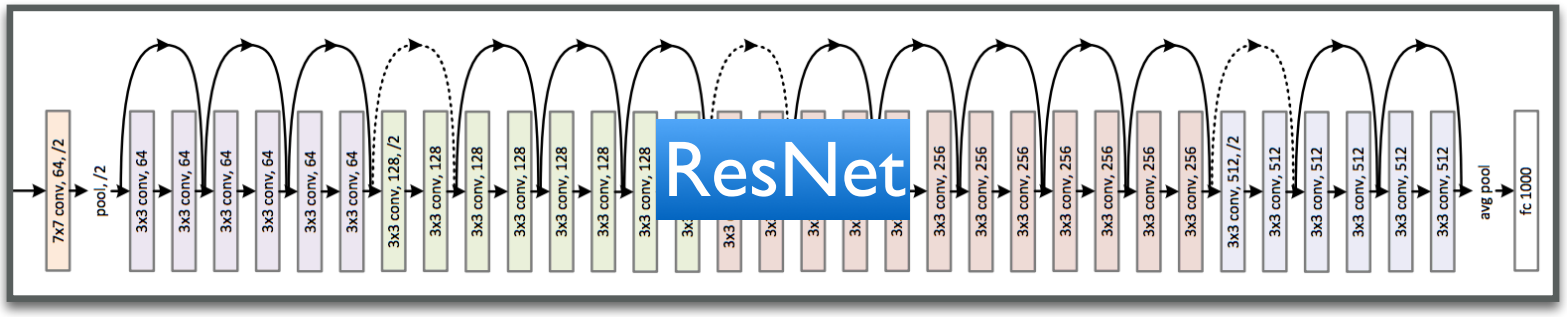
Convolutional networks







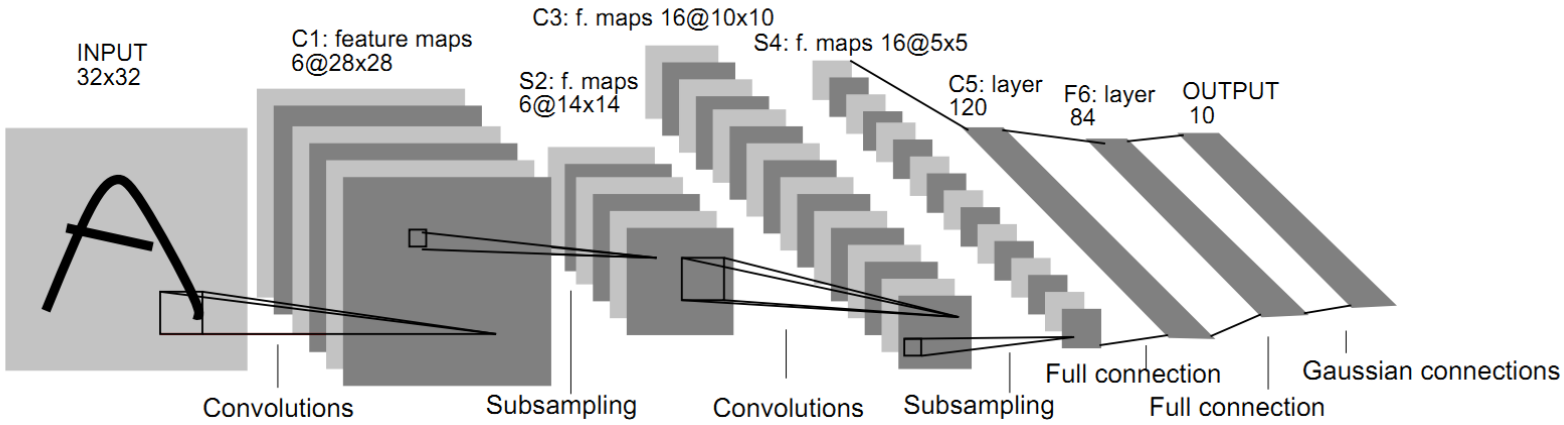




* Slide credits: Gao Huang

LeNet

- First model that really worked by using convolutions:

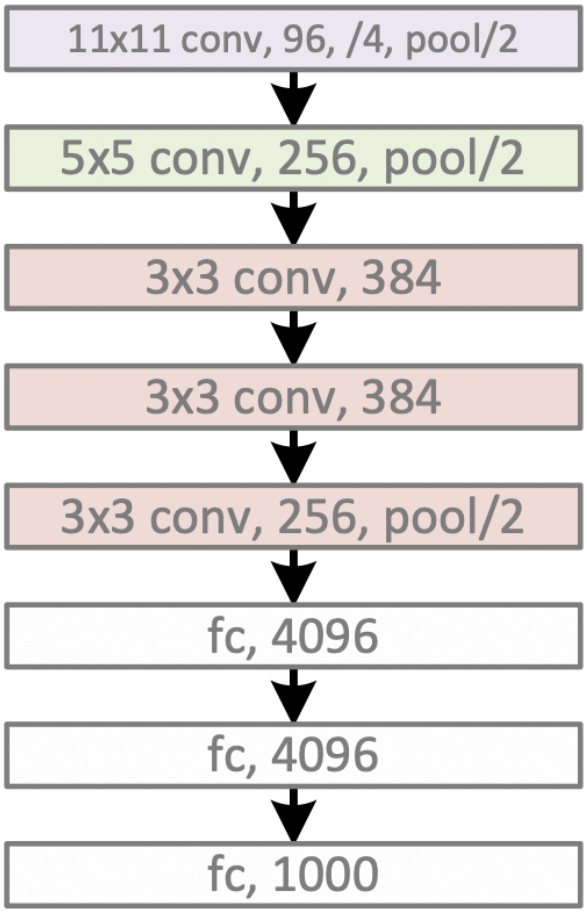


- Compared to today's networks, few convolutional layers and more fully-connected layers

AlexNet

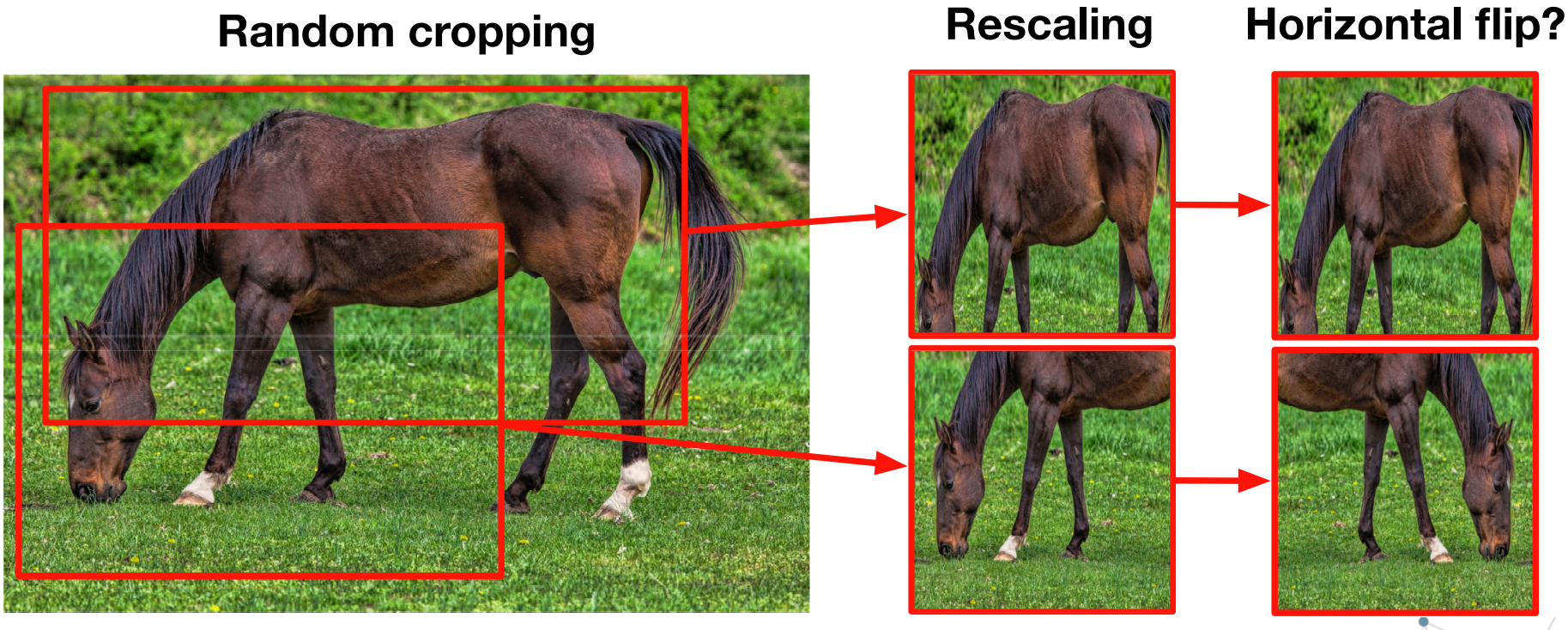
- LeNet-style model with three new components:
  - **Rectified linear units**
  - **Dropout** to reduce overfitting (currently less popular)
  - **Data augmentation** (still very important)

- AlexNet played a key role in popularizing convolutional networks



Data augmentation

- Create many (hard) training examples from a single annotated image:

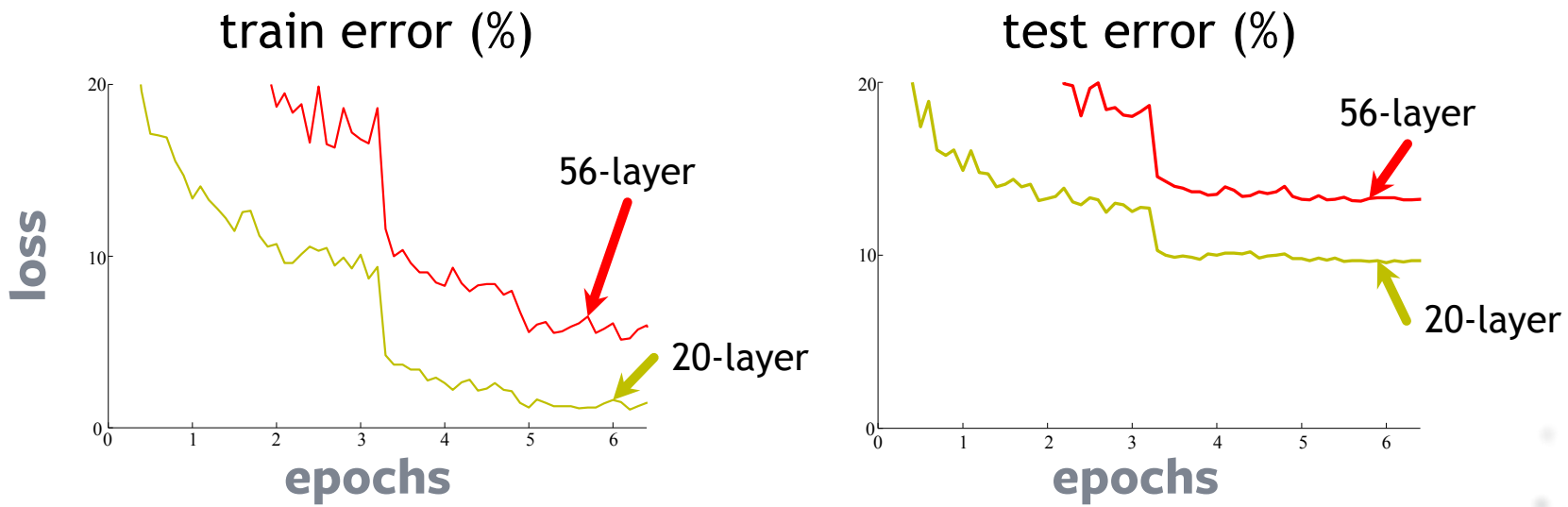


* Note: This is only used at training time! (Why?)


I WAS WINNING IMAGENET
UNTIL A DEEPER MODEL CAME ALONG
imgflip.com

Stacking layers

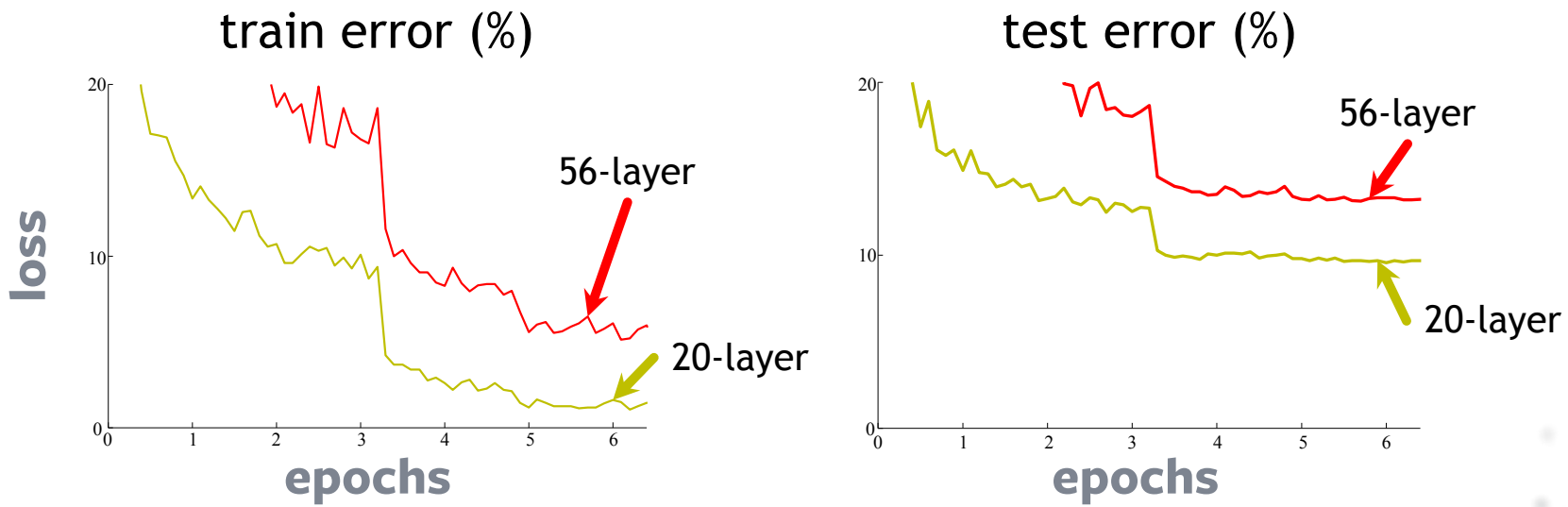
- Simple experiment with stacking many 3x3 convolutional blocks:



* Figure credit: Kaiming He

Stacking layers

- Simple experiment with stacking many 3x3 convolutional blocks:



- Does this suggest overfitting?

* Figure credit: Kaiming He

Stacking layers

- Deeper models should not have higher training error

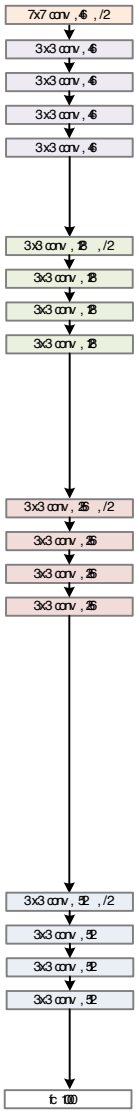
* Figure credit: Kaiming He

Stacking layers

- Deeper models should not have higher training error

- Solution by construction:
  - Train shallow model

"shallow" model

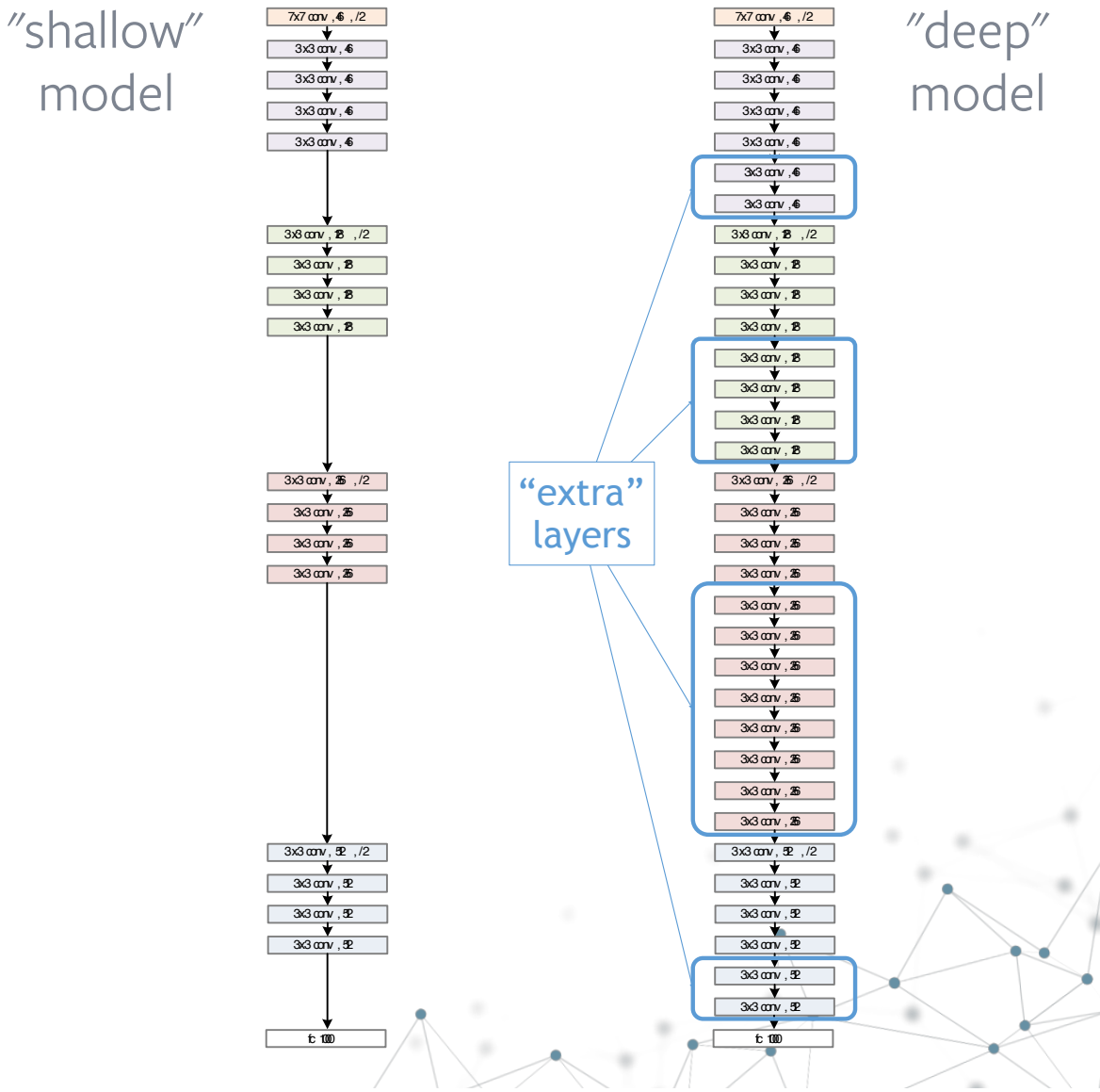


* Figure credit: Kaiming He

Stacking layers

- Deeper models should not have higher training error

- Solution by construction:
  - Train shallow model
  - Add additional layers set to identity
  - Train the deeper model further



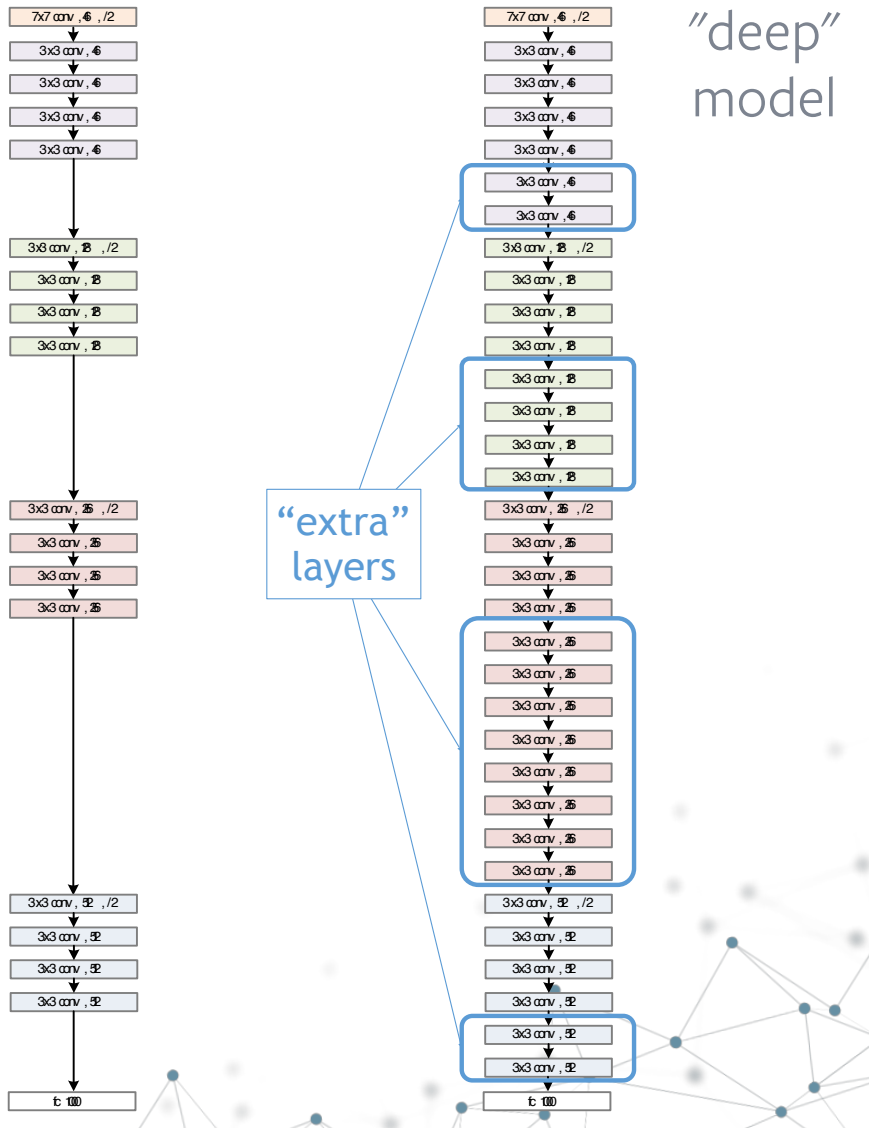
* Figure credit: Kaiming He

Stacking layers

- Deeper models should not have higher training error

- Solution by construction:
  - Train shallow model
  - Add additional layers set to identity
  - Train the deeper model further

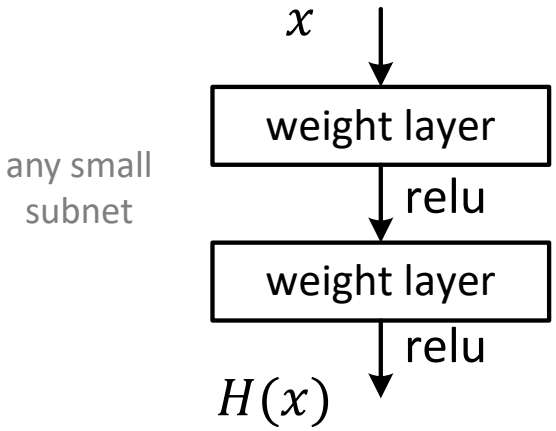
- Learning does not work well :(



* Figure credit: Kaiming He

Residual connections

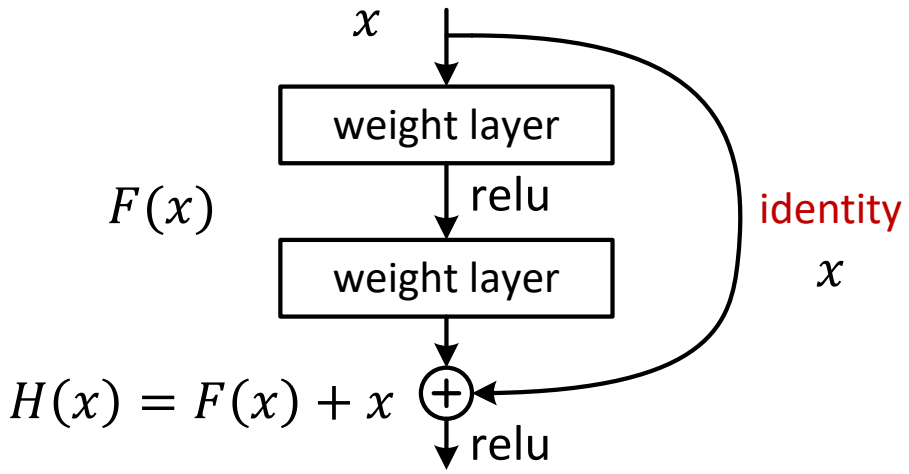
- Block architecture that achieves similar goal
- Take any "regular" network block...



* Figure credit: Kaiming He

Residual connections

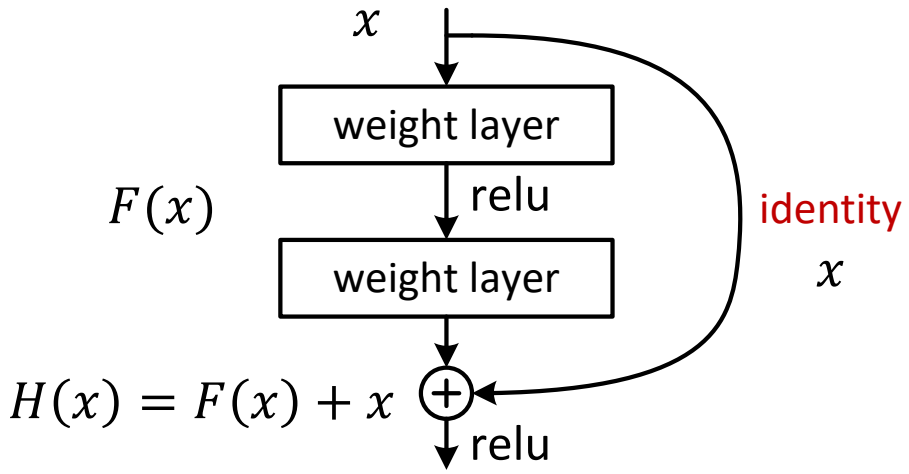
- Block architecture that achieves similar goal
- Take any "regular" network block... and make it **residual**:



* Figure credit: Kaiming He

Residual connections

- Block architecture that achieves similar goal
- Take any "regular" network block... and make it **residual**:

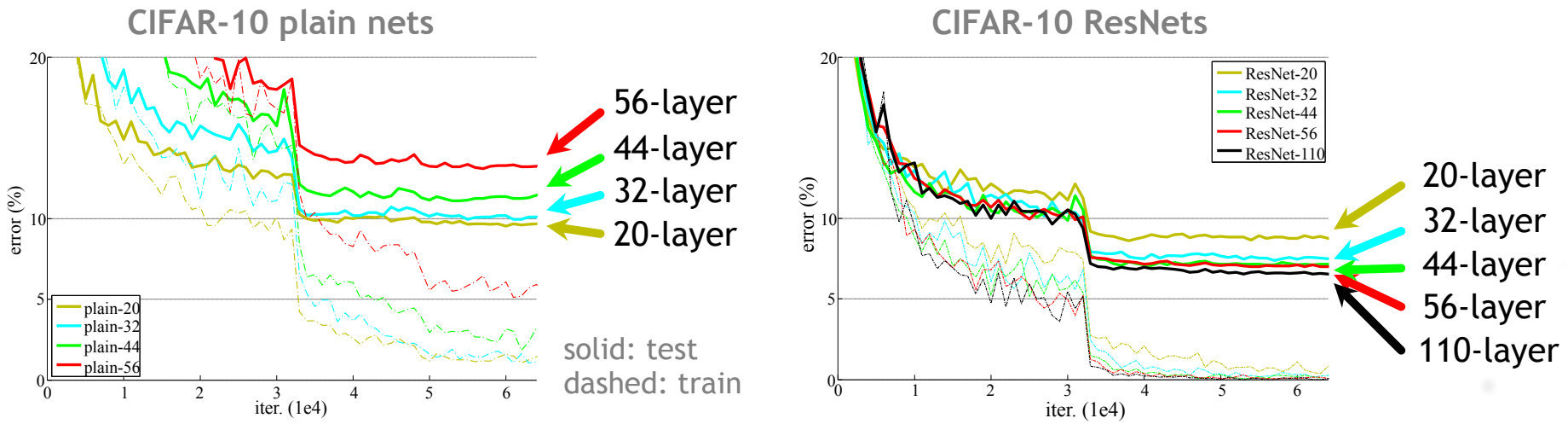


- Initializing weights to zero achieves the desired effect

* Figure credit: Kaiming He

Residual connections

- Deeper models can now be trained without problems:

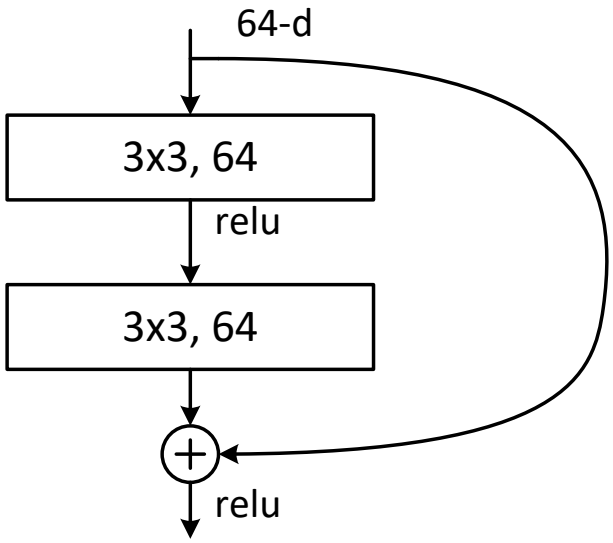


- Overfitting may still happen, but at least training error does not go up

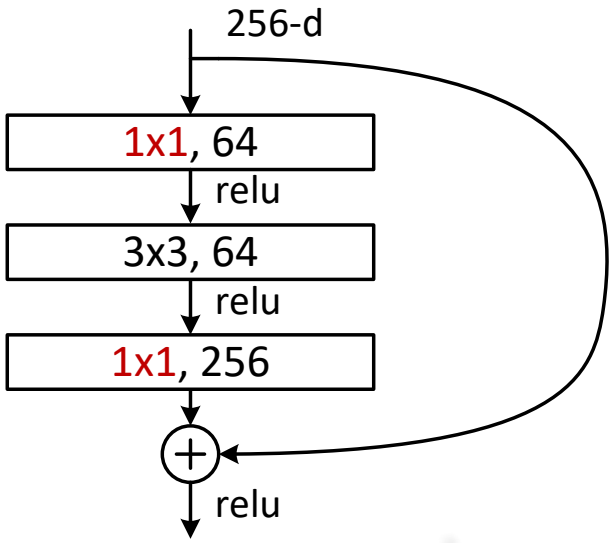
* Figure credit: Kaiming He

Residual networks

- In practice, residual networks (ResNet) use the following block design:



simple design

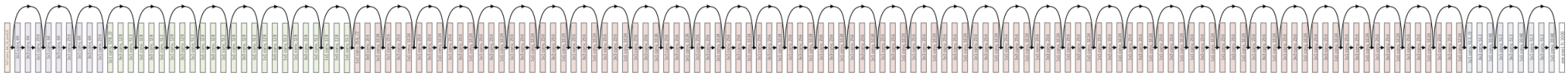


bottleneck design

* Figure credit: Kaiming He

Residual networks

- Full model has pooling layers for downsampling
- All pooling layers do max-pooling but the last one does average-pooling



- Every time the image size halves, the number of channels is doubled

Summary

- Convolutional networks vary in their architecture
- Pooling gradually eliminates spatial structure from the inputs
- Rectified linear units facilitate learning by having good gradients
- Batch normalization controls the scale of the gradients
- Residual helps loss propagation through network, allowing for successfully training deeper networks

Reading material

- S. Ioffe and C. Szegedy. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**. In *International Conference on Machine Learning (ICML)*, 2015.

- K. He, X. Zhang, S. Ren, and J. Sun. **Deep Residual Learning for Image Recognition**. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Questions?