Machine Learning and
Computer Vision Group

IST AUSTRIA
Institute of Science and Technology

Deep Learning with Tensorflow

http://cvml.ist.ac.at/courses/DLWT_W17/

AlexNet

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton,
"Imagenet classification with deep convolutional neural networks",
Advances in neural information processing systems, 2012

Djordje Slijepcevic

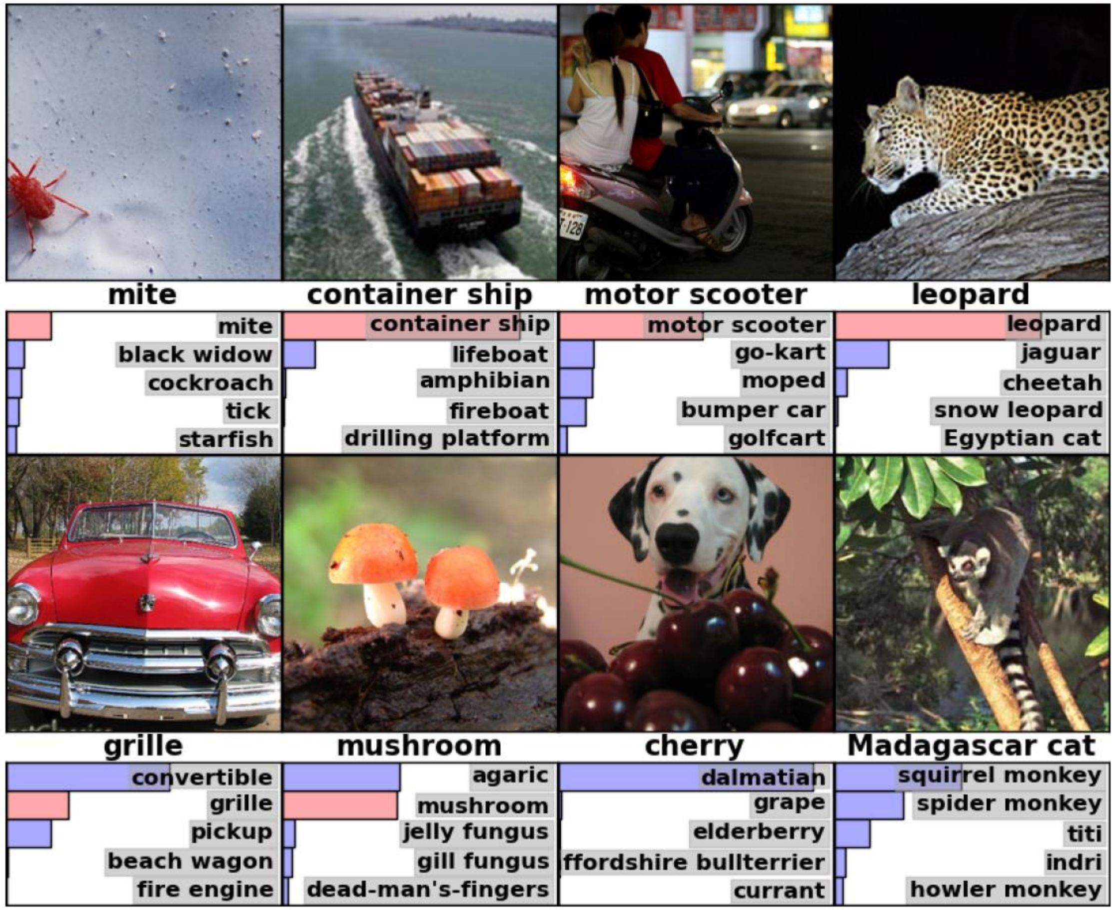
Introduction

- Convolutional Neural Network (CNN)
- Winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012
  - first successful CNN application for such a big dataset
  - top-5 test error rate of 15.3% (+10.9% compared to 2^{nd})
- Relatively simple layout (compared to modern architectures)
  - 5 conv. layers
  - 3 fully connected layers
  - max-pooling layers
  - dropout layers

Dataset

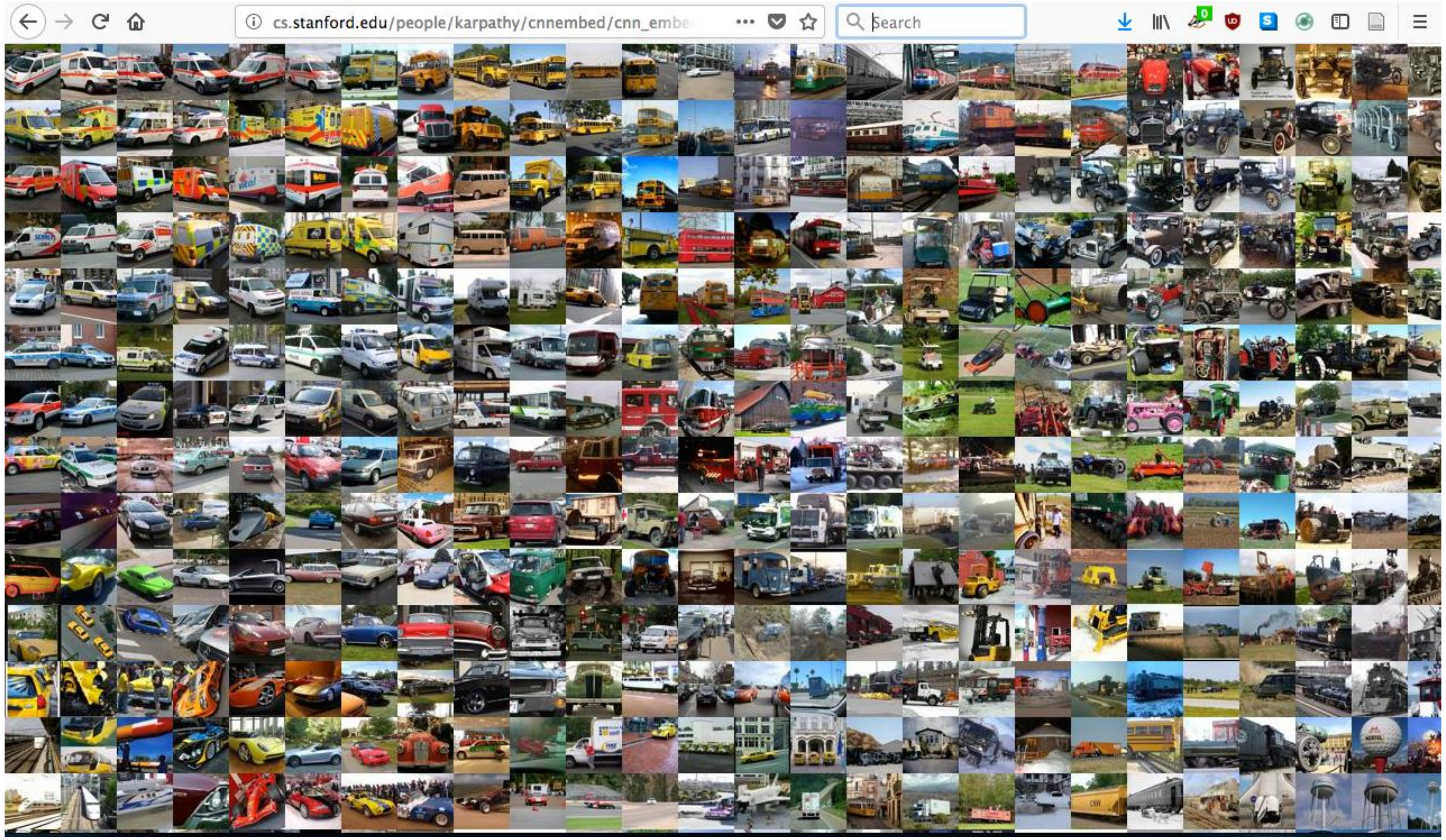
- ImageNet:
  - 15+ million labeled high-resolution images
  - 22000 categories
- ILSVRC uses a subset of ImageNet:
  - ~ 1000 images per category
  - 1000 categories
  - 1.2 million training images | 50000 validation images | 150000 testing images
- AlexNet:
  - images were down-sampled and cropped to 256×256 pixels
  - subtraction of the mean activity over the training set from each pixel

Task

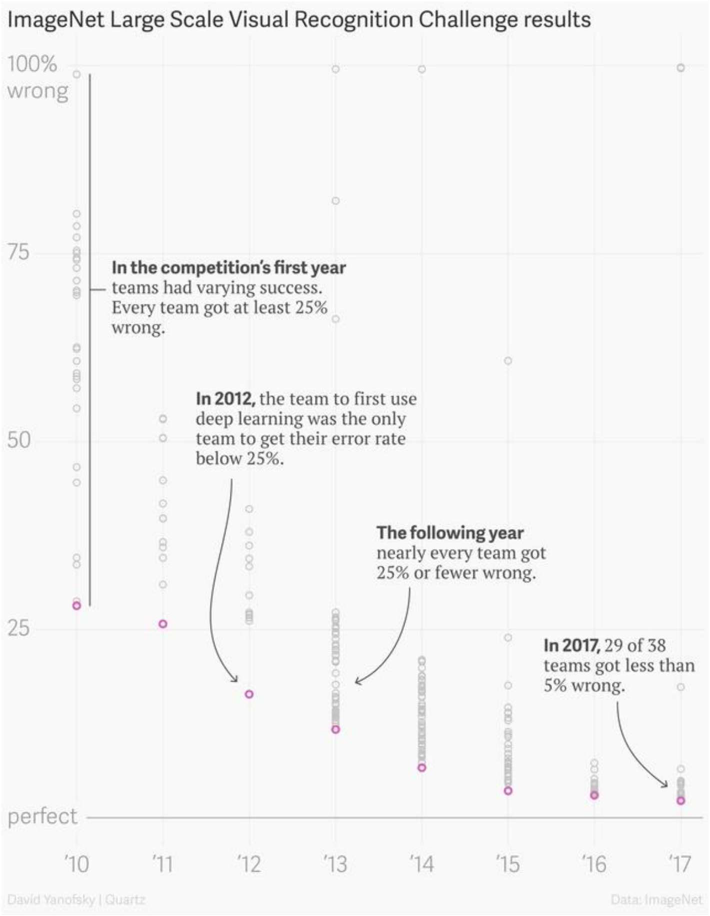


[A. Krizhevsky, I. Sutskever, G.E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012]

Dataset



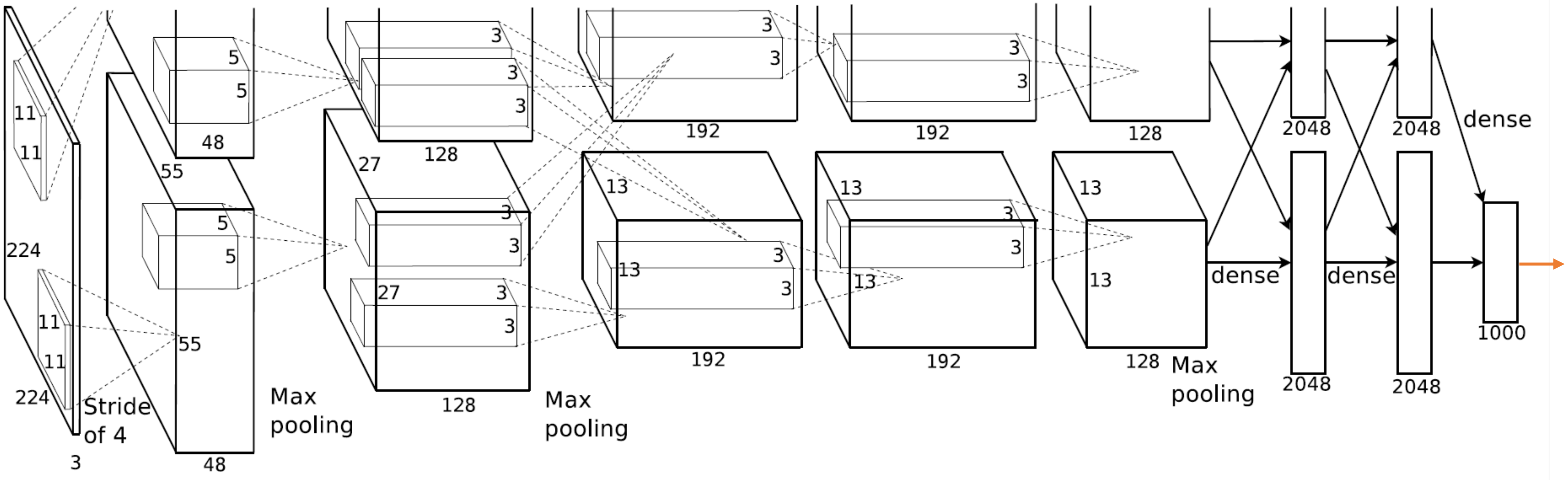
[http://cs.stanford.edu/people/karpathy/cnnembed/, 30.11.2017]



[https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/, 30.11.2017]

Architecture

3 fully connected layers



1000-way
softmax

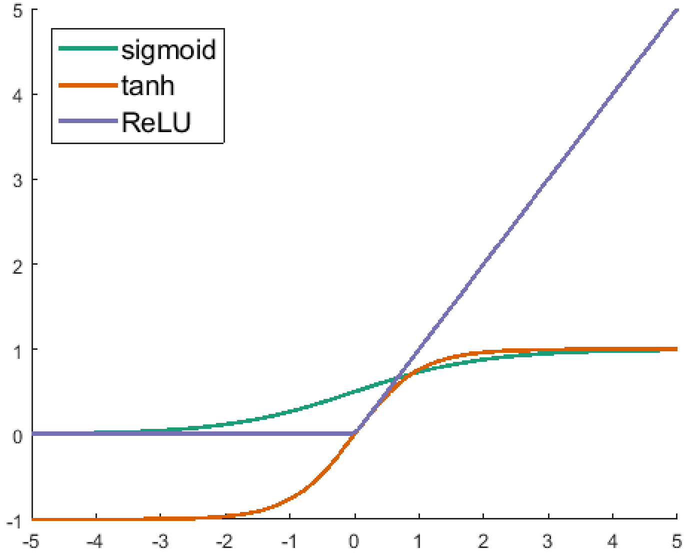
5 convolutional layers

[A. Krizhevsky, I. Sutskever, G.E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012]

Activation function

- Traditionally, saturating nonlinearities:
  - hyperbolic tangent function: $f(x) = \tanh(x) = 2 * \frac{1}{1+e^{-2x}} - 1$
  - sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$

  → slow to train



- Non-saturating nonlinearity:
  - Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$

  → quick to train

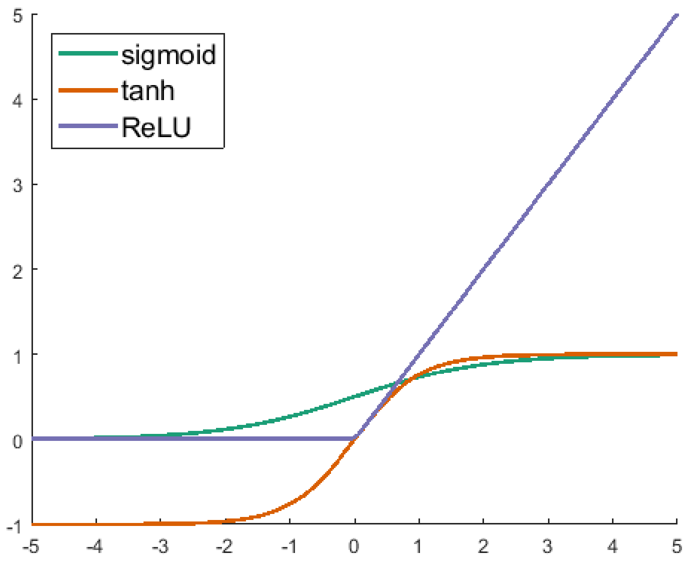
Activation function

- Traditionally, saturating nonlinearities:
  - Saturated neurons facilitate vanishing of gradients
  - exp function is a bit compute expensive

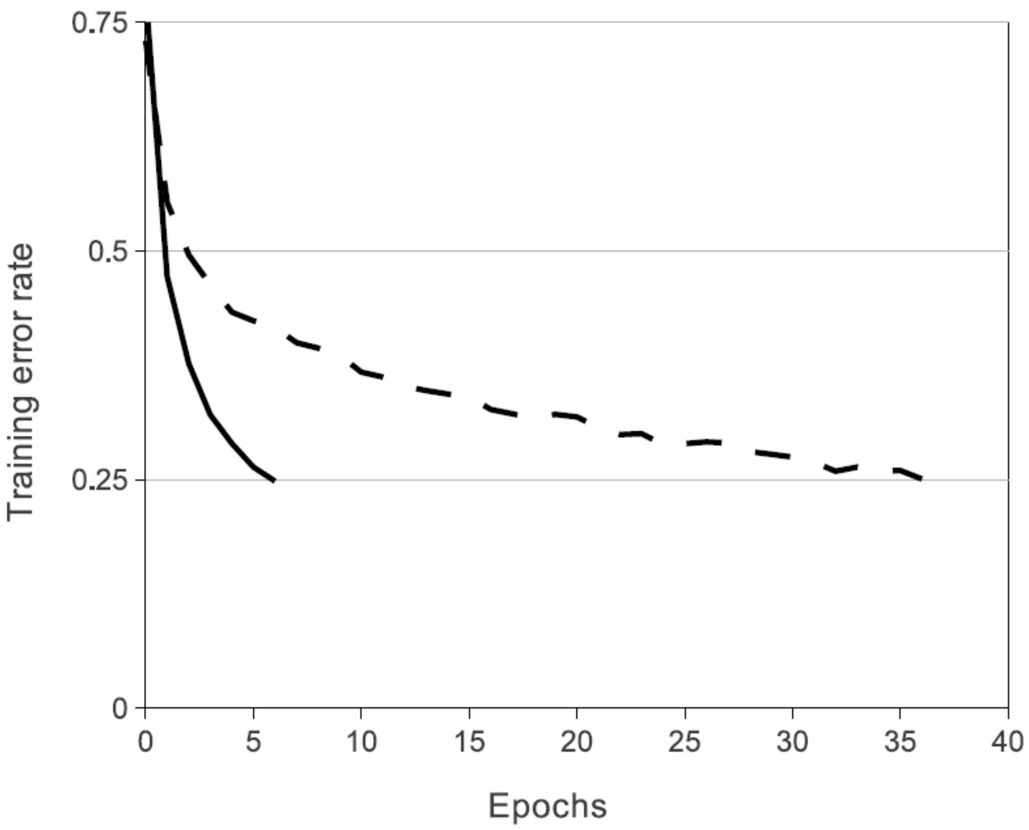
  → slow to train

- Non-saturating nonlinearity:
  - Does not saturate (in the + region)
  - Very computationally efficient

  → quick to train



Activation function



- Dataset: CIFAR-10
- Experiment:

  CNN (4 layers) + ReLUs (solid line)

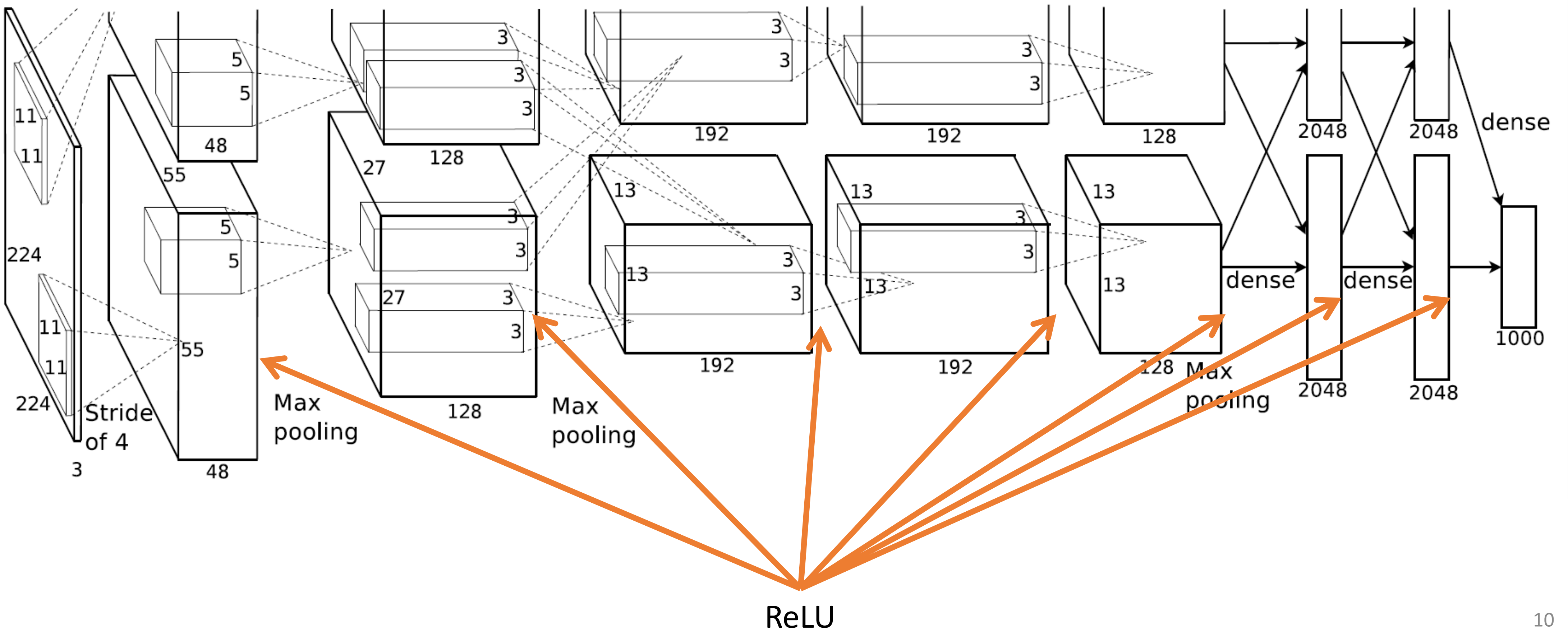
  vs.

  CNN (4 layers) + tanh (dashed line)

  → ReLUs six times faster

[A. Krizhevsky, I. Sutskever, G.E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012]

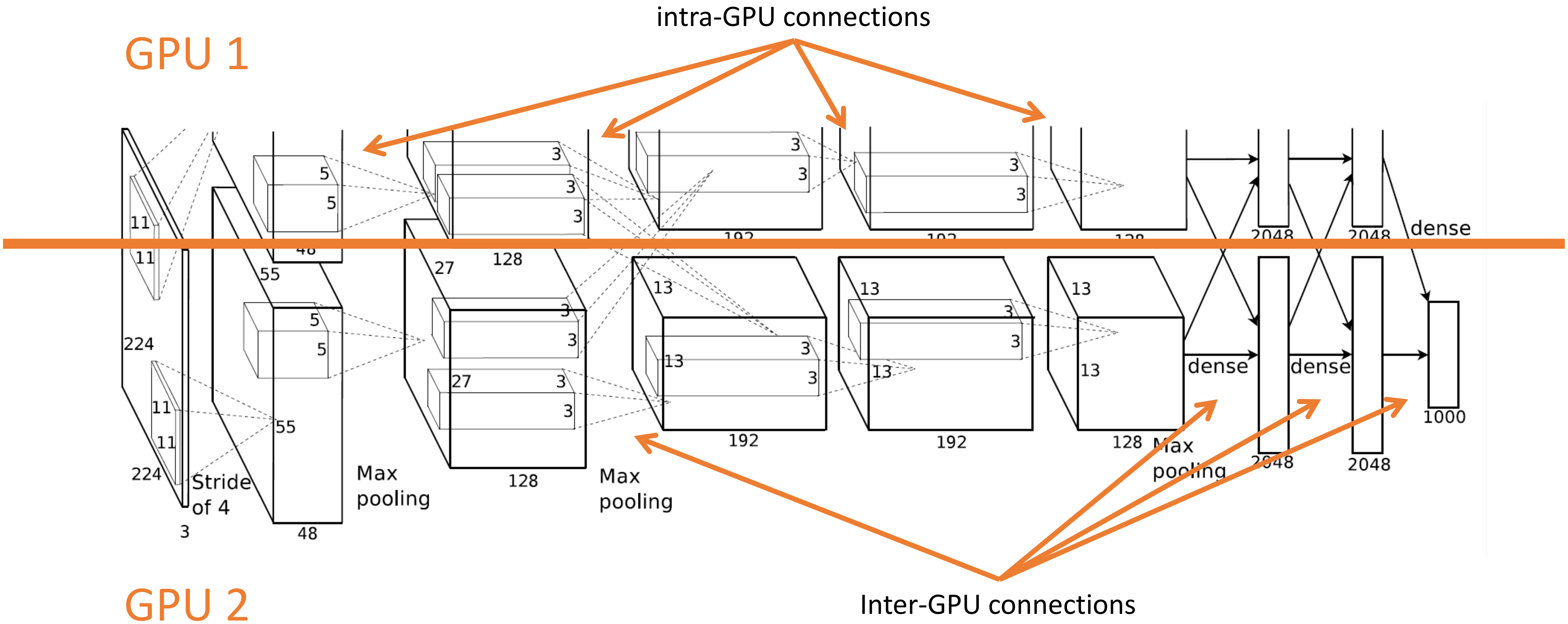
Activation function



Training on Multiple GPUs

- Half of the neurons of an certain layer are on each GPU
- GPUs communicate only in certain layers
- Improvement (as compared with a net with half as many kernels in each convolutional layer trained on one GPU):
  - top-1 error rate by 1.7%
  - top-5 error rate by 1.2%

Training on Multiple GPUs



Local Response Normalization

- ReLUs do not require input normalization to prevent them from saturating
- However, Local Response Normalization aids generalization

Activity of a neuron by applying kernel i at position (x,y)

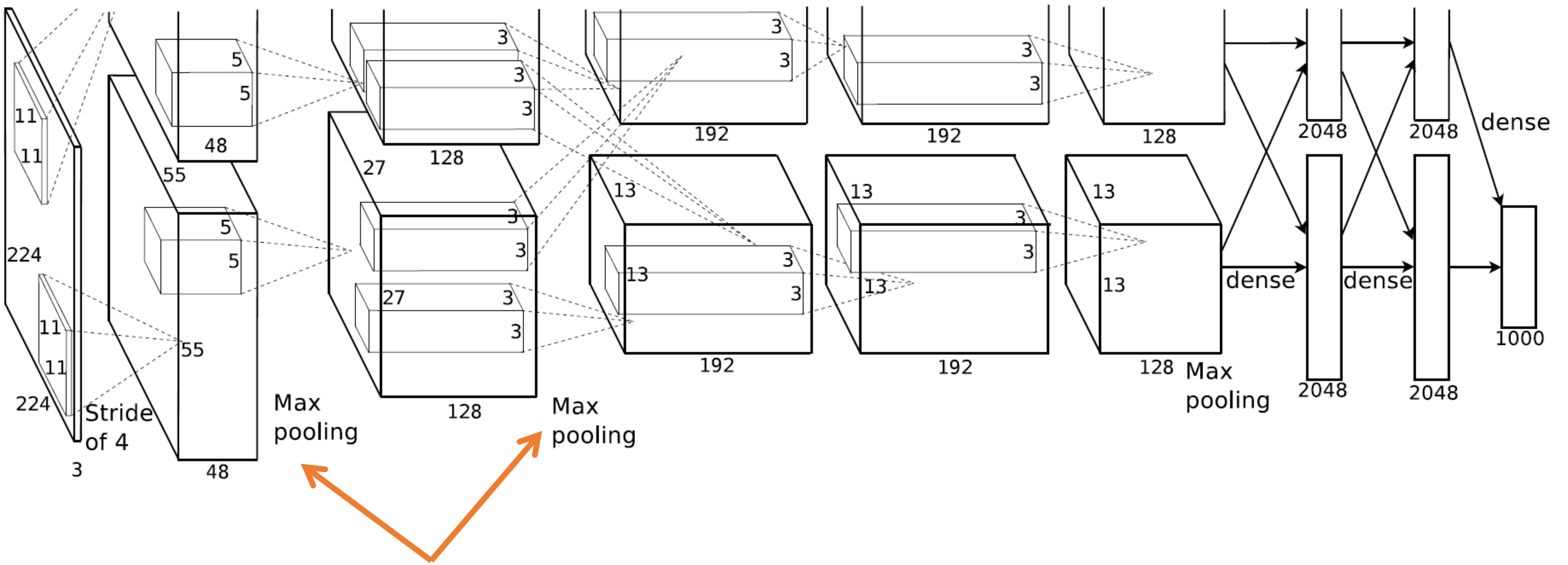
$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + \alpha \sum_{j=\max\left(0, i-\frac{n}{2}\right)}^{\min\left(N-1, i+\frac{n}{2}\right)} \left(a_{x,y}^{j}\right)^{2} \right)^{\beta}$$

$k = 2$
$n = 5$
$\alpha = 10^{-4}$
$\beta = 0.75$

sum runs over n “adjacent” kernel maps at the same spatial position

- Improvement:
  - top-1 error rate by 1.4%
  - top-5 error rate by 1.2%

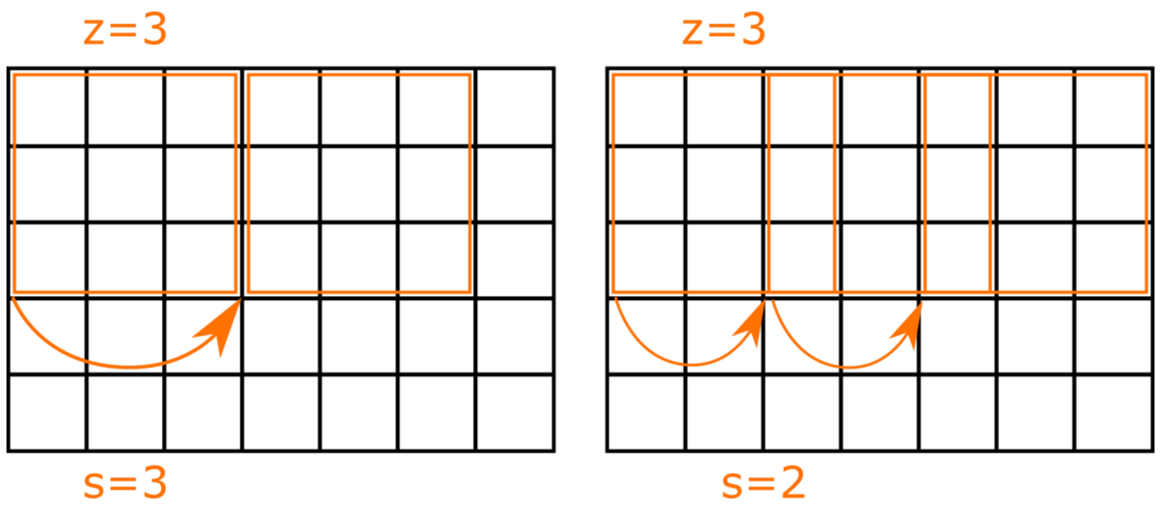
Local Response Normalization



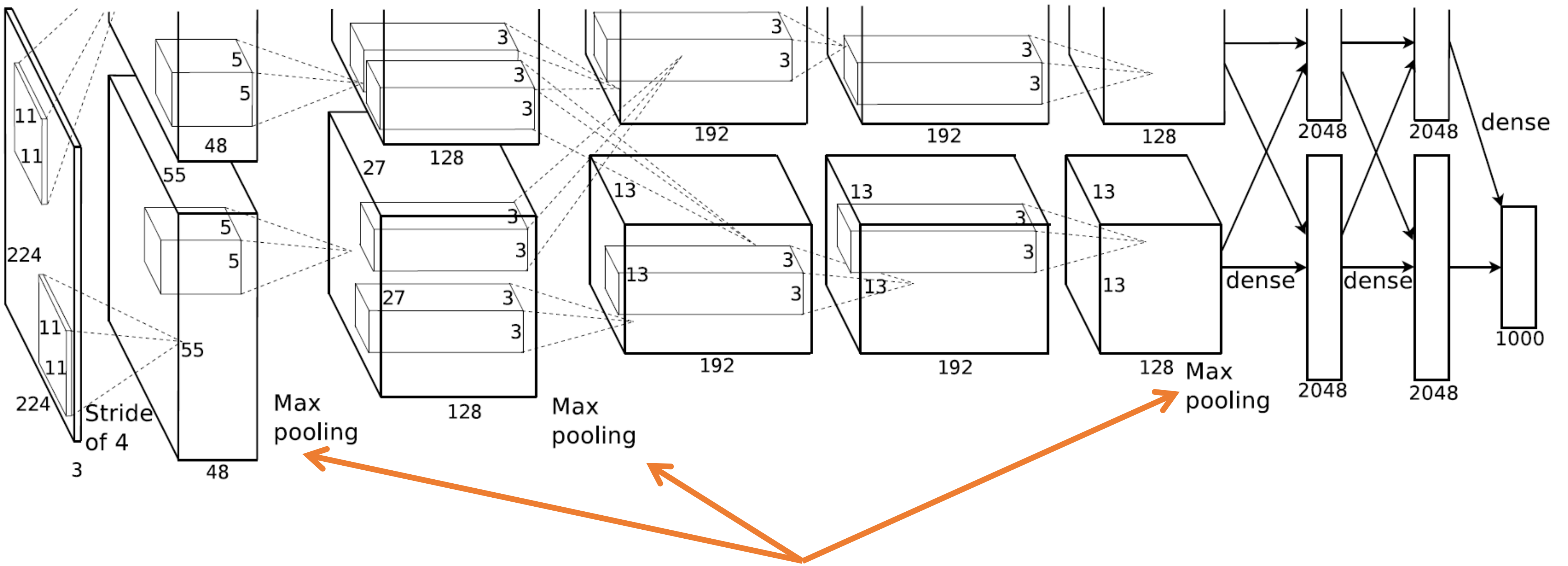
Local Response Normalization

Overlapping Pooling

- Pooling layers summarize the outputs of neighboring neurons in the same kernel map.
- Overlapping pooling → s < z
- Improvement using MaxPooling:
  - top-1 error rate by 0.4%
  - top-5 error rates by 0.3%



Overlapping Pooling



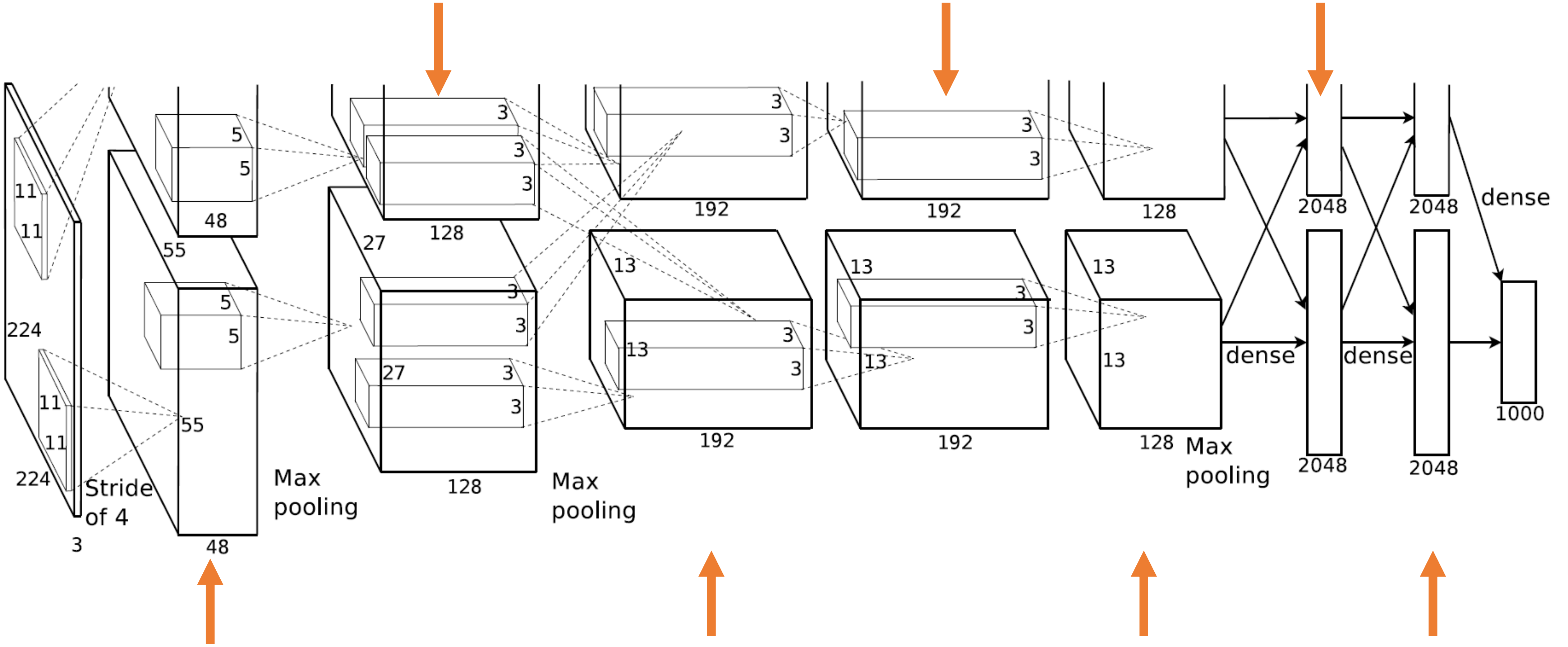
Overlapping Pooling

Overall Architecture

256 kernels (5x5x48)

384 kernels (3x3x192)

4096 neurons



96 kernels (11x11x3)

384 kernels (3x3x256)

256 kernels (3x3x192)

4096 neurons

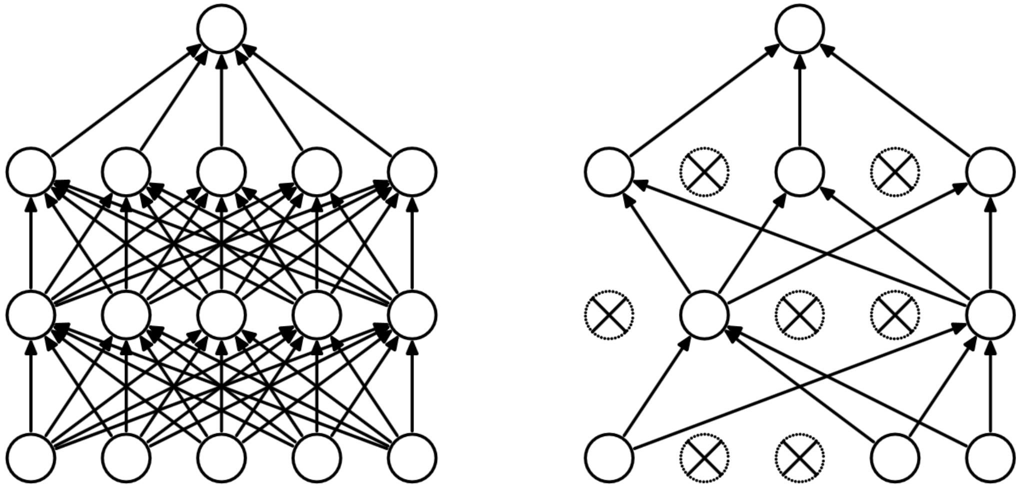
Reducing Overfitting - Data Augmentation

- 1st : image translations and horizontal reflections
  - random 224x224 patches + horizontal reflections from the 256x256 images
  - Testing: five 224x224 patches + horizontal reflections → averaging the predictions over the ten patches
- 2nd : change the intensity of RGB channels
  - PCA on the set of RGB pixel values throughout the ImageNet training set
  - To each RGB image pixel $\mathrm{I_{xy}} = \left[I_{xy}^R, I_{xy}^G, I_{xy}^B\right]$ following is added

$$\underbrace{[p_1, p_2, p_3]}_{\text{eigenvectors}}[\alpha_1 \underbrace{\lambda_1}_{\text{eigenvalues}}, \alpha_2 \underbrace{\lambda_2}_{\text{eigenvalues}}, \alpha_3\, \underbrace{\lambda_3}_{\text{eigenvalues}}]^T \quad | \alpha_i \sim N(0, 0.1)$$

  - Improvement:
    - top-1 error rate by 1%

Reducing Overfitting - Dropout

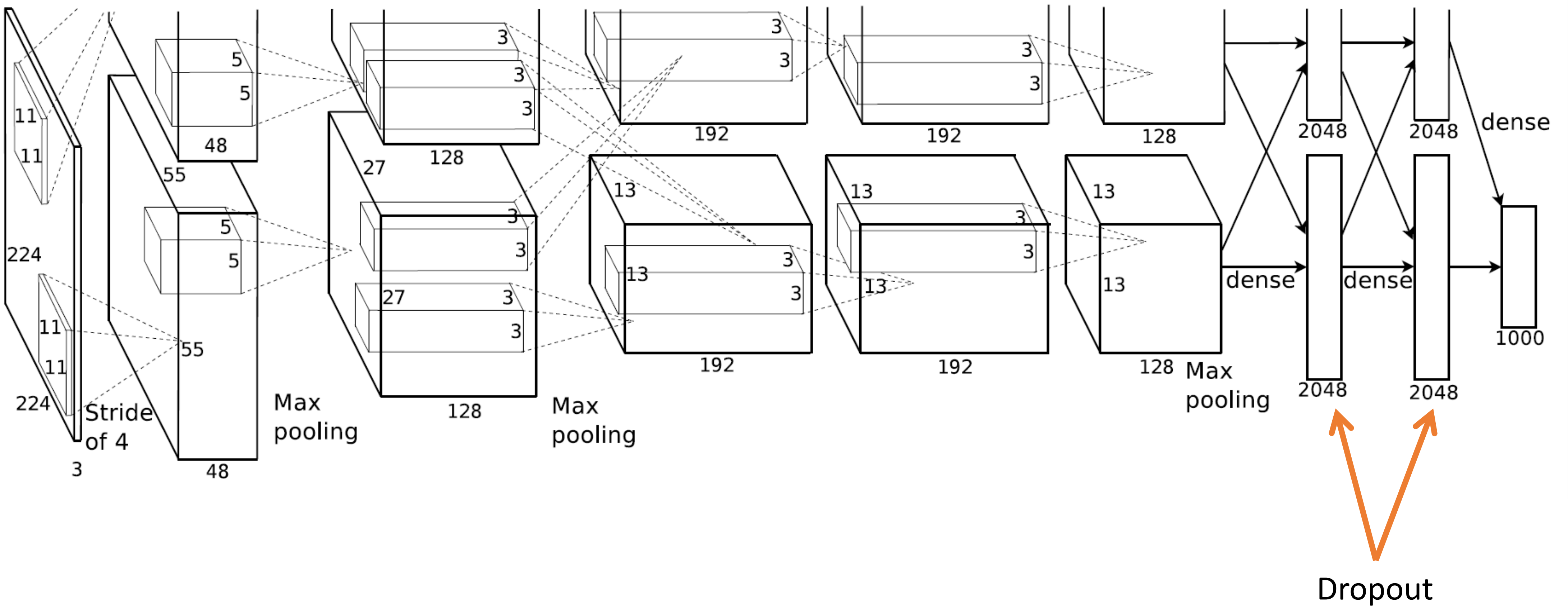


(a) Standard Neural Net (b) After applying dropout.

- Output of each hidden neuron is set to zero with probability 0.5
- Learning more robust features
- Doubles the number of iterations required to converge
- Applied in the first two fully connected layers

[N. Srivastava et al., *Dropout: A Simple Way to Prevent Neural Networks from Overfitting,* 2014]

Reducing Overfitting - Dropout



Stochastic Gradient Descent

- Training process
  - Minimizing the cross-entropy loss function:

$$L(w) = \sum_{i=1}^{N} \sum_{c=1}^{1000} -y_{ic} \log f_c(x_i) + \epsilon \|w\|_2^2$$

predicted probability of class c for image x

indicator that example i has label c

Stochastic Gradient Descent

- SGD with a batch size of 128
- Learning rate initialized at 0.01; divided by 10 if validation error rate stopped improving
- Update rule for weight w:

weight decay

learning rate

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

momentum

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

Gradient of Loss

- ~ 90 cycles → five to six days on two NVIDIA GTX 580 3GB GPUs