ImageNet Classification with Deep Convolutional Neural Networks

Seoyoon Choi

Introduction

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Introduction

- Simple object recognition with small dataset (MNIST, CIFAR)
  → already effective

- Real-world classification
  - need larger, more complex labeled images → LabelMe, **ImageNet**
  - need a model with a large learning capacity → **CNN**

However, it's still too expensive
to apply in large scale to high-resolution images...

Introduction

- GPUs paired with 2D convolution
  - used highly-optimized GPU for 2D convolution
  - powerful enough to facilitate the training of large CNNs

- Takes five - six days to train on two GTX 580 3GB GPUs
  - results can be improved by faster GPUs and bigger datasets someday
    → EfficientNet

Dataset

ImageNet

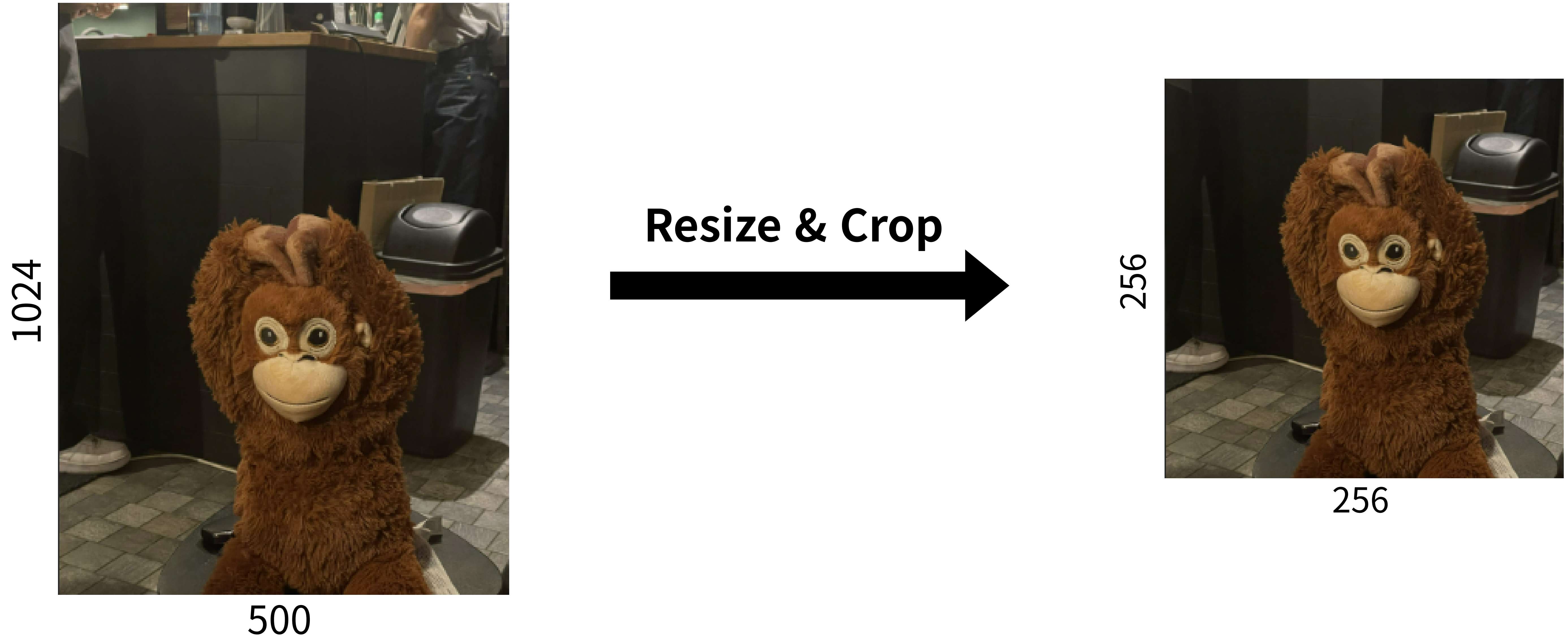
- 22,000 categories, 15 million images

- ILSVRC-2010 & ILSVRC-2012 competitions (for AlexNet)
  - subset of ImageNet
  - 1000 images in each of 1000 categories

Dataset

Data Preprocessing

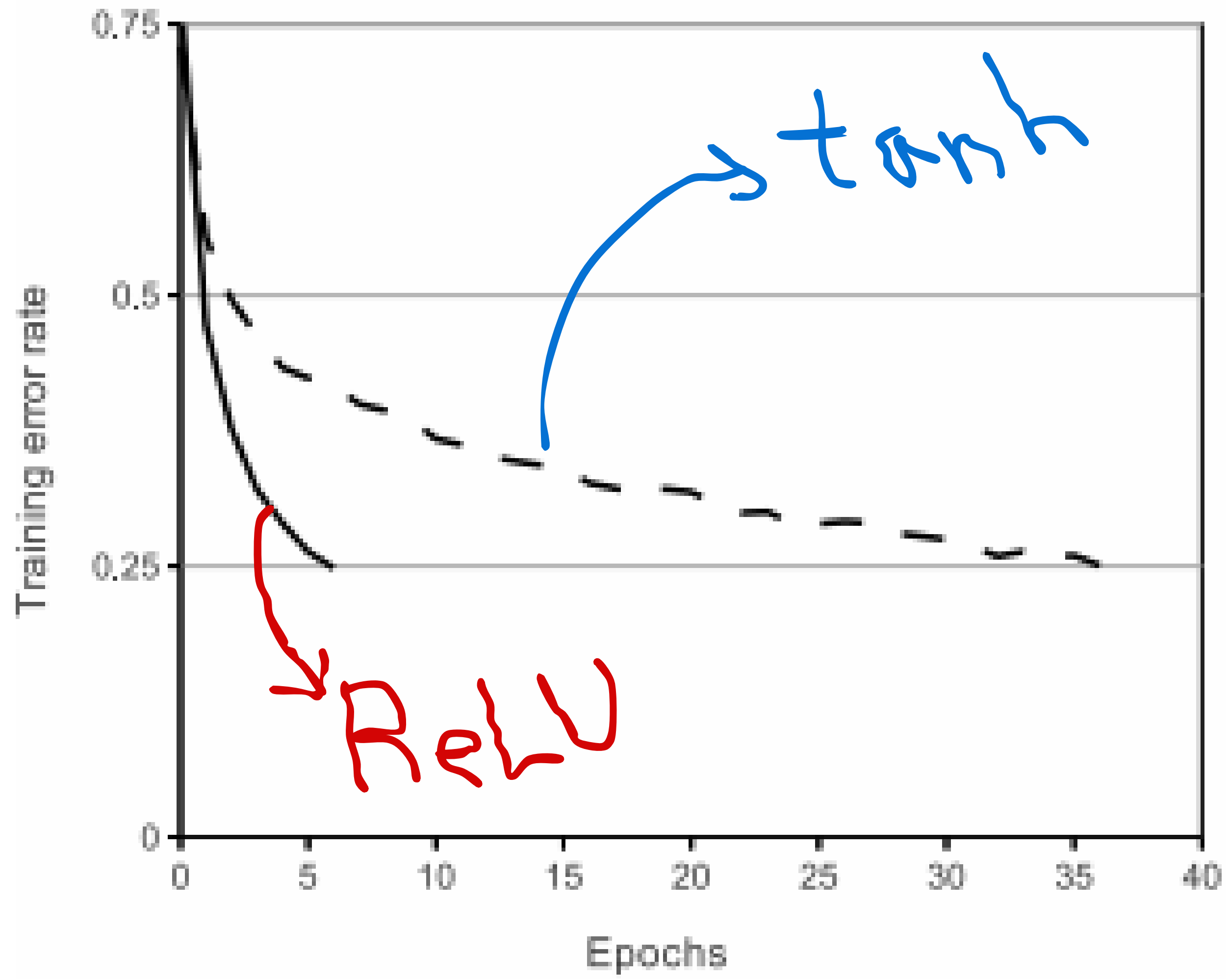
- ImageNet consists of variable-resolution images
- we need a fixed input dimensionality
  1) down-sampled to 256 x 256 (**Resize**)
  2) cropped out the central 256 x 256 patch from the resized image (**Crop**)



AlexNet

Activation Function

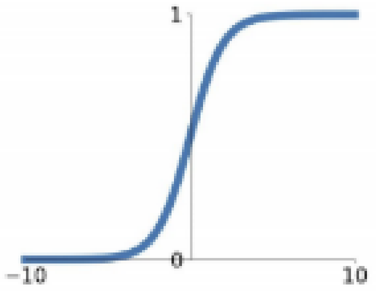
- Saturating Nonlinearity (**tanh**, **sigmoid**)
  - slower training time
  - gradient vanishing



- Non-saturating Nonlinearity (**ReLU**)
  - converges faster than tanh/sigmoid
  - computationally efficient
  - doesn't saturate (in + region)

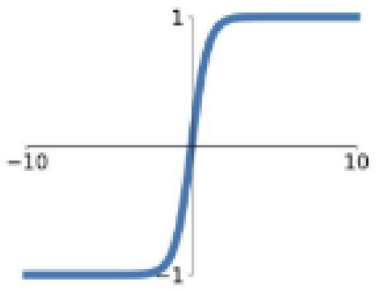
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



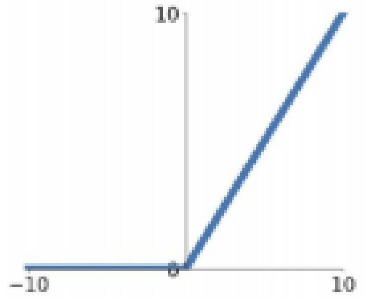
tanh

$\tanh(x)$



ReLU

$\max(0, x)$

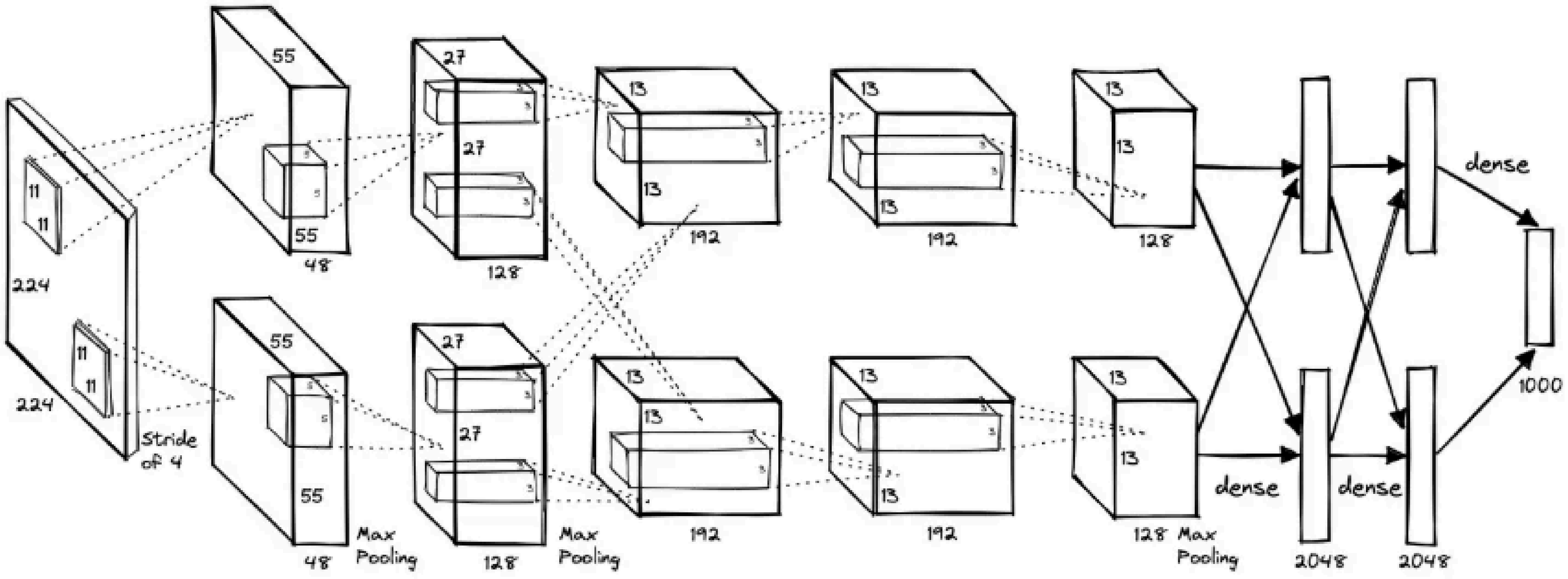


AlexNet

Multiple GPUs

- Single GTX 580 GPU has only 3GB of memory
  - limits the maximum size of the networks to train

- Spread the net across two GPUs
  - able to read from and write to one another's memory directly
  - GPUs communicate only in certain layers



AlexNet

말 그대로 지역적(local)인 응답(response)을 정규화(normalization)한다!

Local Response Normalization

- Lateral inhibition (biological motivation)
  - highly active neurons tend to inhibit their neighbors
  - this creates contrast and sharpens sensory perceptions

- Goal of LRN
  - create competition across feature maps
    → enhance the most "salient" features
  - improve the model's generalization capabilities

AlexNet

Local Response Normalization

feature map의 특정 위치 (x, y)에서, 채널 방향으로 이웃한 값들을 이용해 현재 채널의 값을 정규화

$$b^i_{x,y} = a^i_{x,y} \Bigg/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

$a^i_{x,y}$: source output of kernel i applied at position(=pixel) x, y

$b^i_{x,y}$: regularized output for kernel i at position(=pixel) x, y

N : total # of kernels

n : size of the normalization neighborhood

α, β, k, n : hyperparameters

- k = 2
- n = 5
- α = 0.0001
- β = 0.75

AlexNet

Local Response Normalization

- reduced Top-1 error by 1.3% and Top-5 error by 1.1%

- Why don't we use LRN today?
  - not effective to deeper models like VGGNet
  - Batch Normalization(2015) proved to be more effective

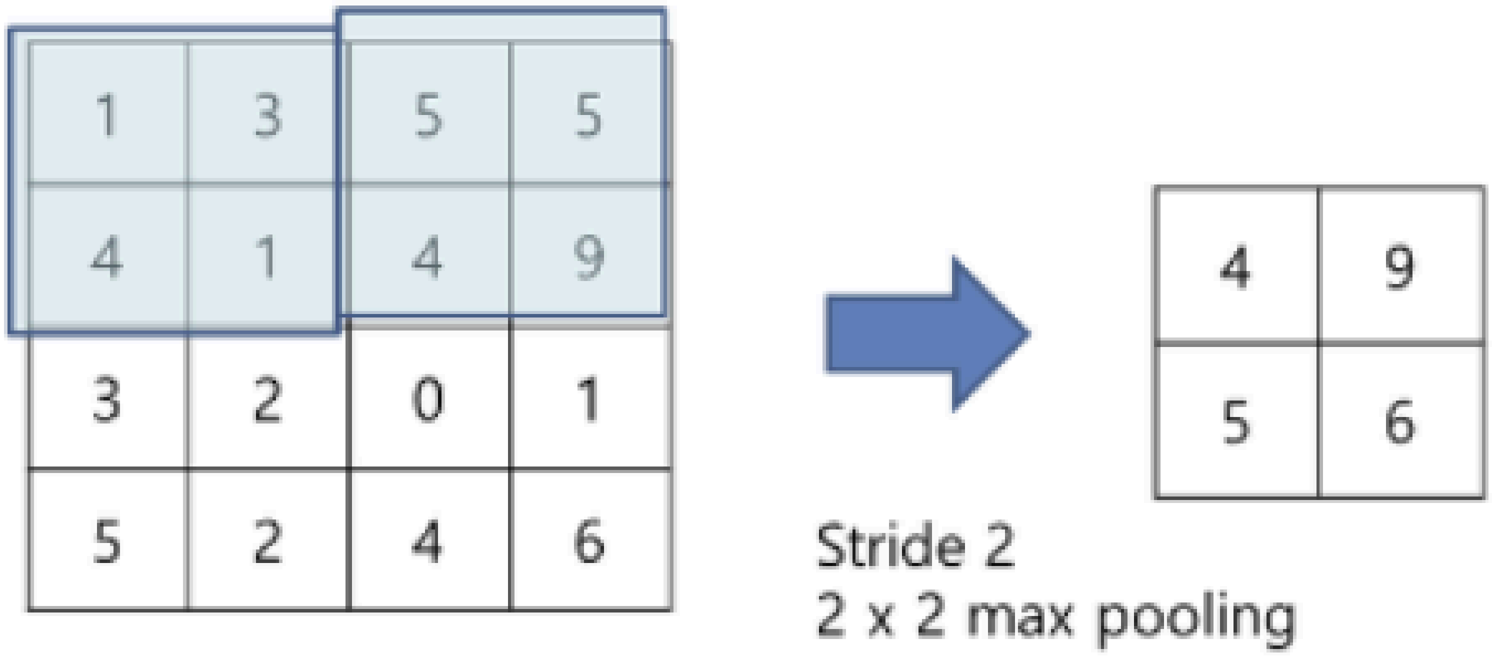
- LRN vs Batch Normalization
  - LRN: normalizes across channels
  - BN: normalizes across the batch for each individual feature
    → faster and more stable training

AlexNet

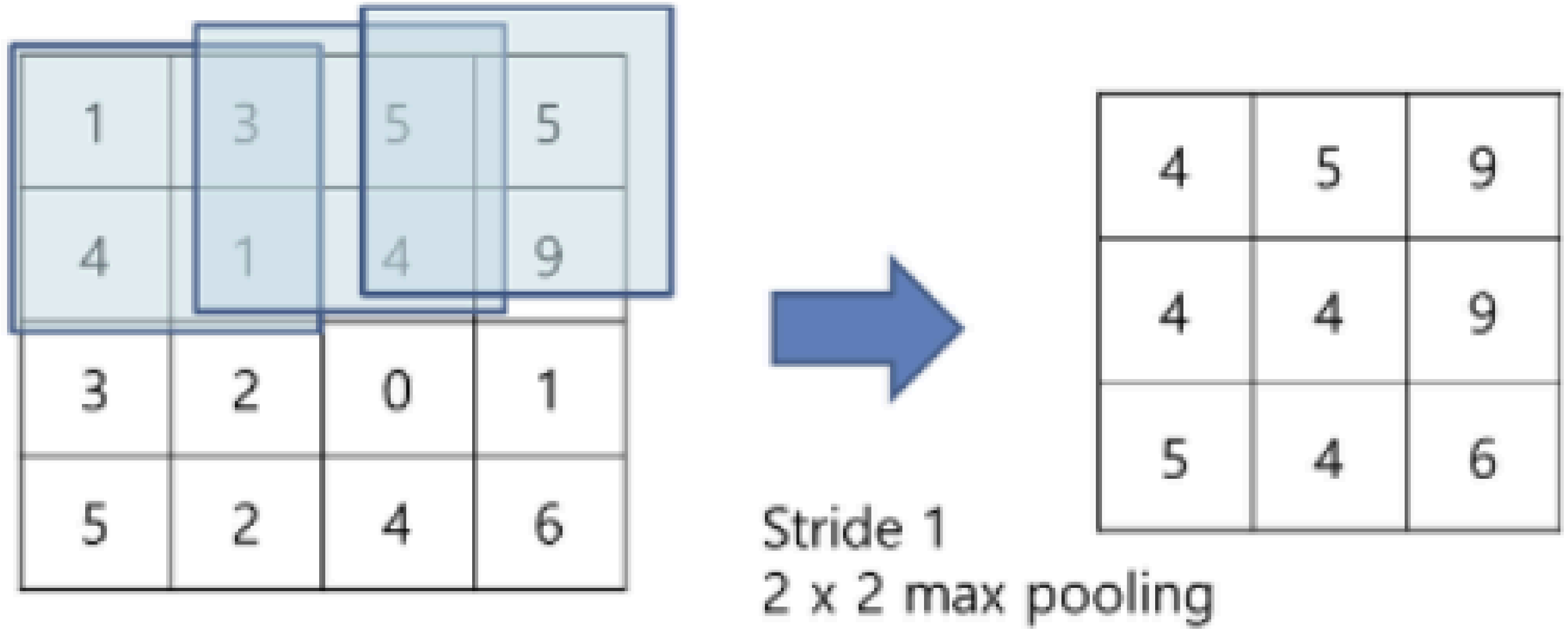
Overlapping Pooling

- s: stride
- z: size of the feature map

Overlapping pooling helps prevent overfitting more effectively than non-overlapping pooling.



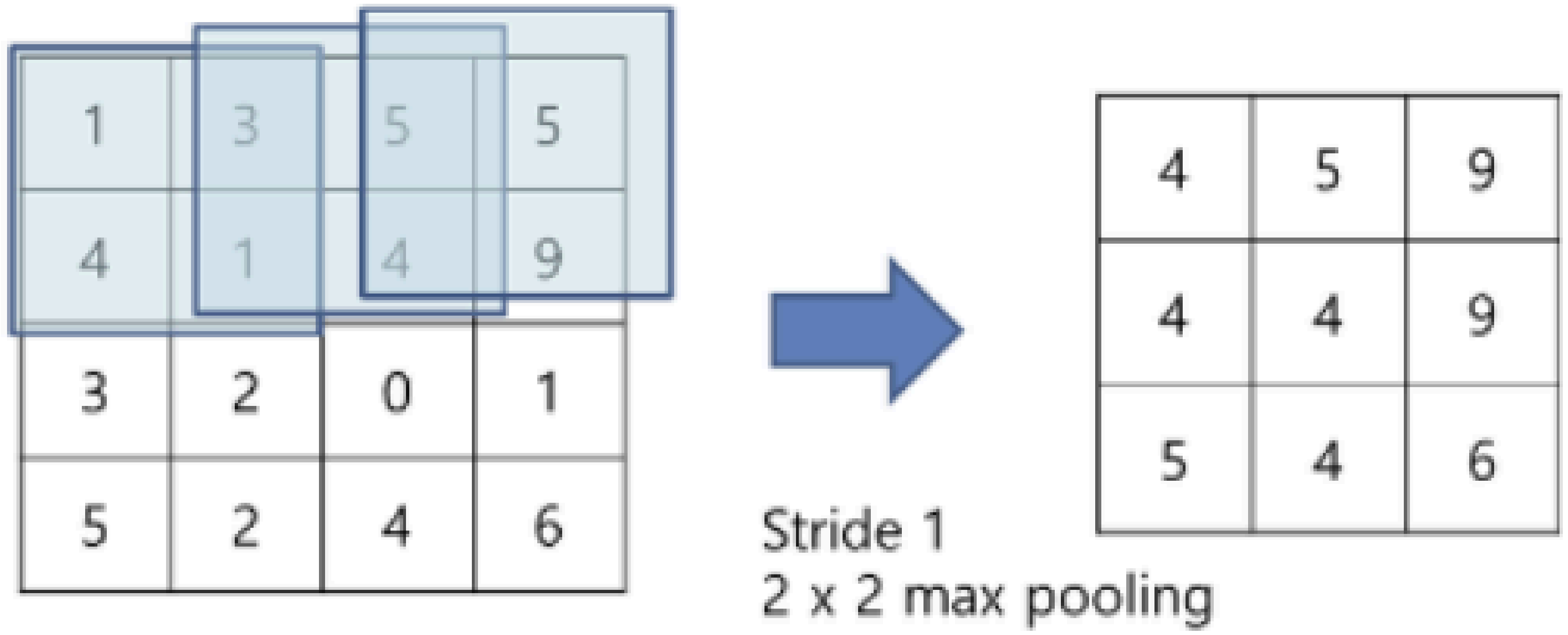
Non-overlapping pooling
(s = z)



Overlapping pooling
(s < z)

Overlapping Pooling vs Non-overlapping Pooling

- keeps more spatial information
- provides smoother feature representation
- adds a regularization effect → helps reduce overfitting

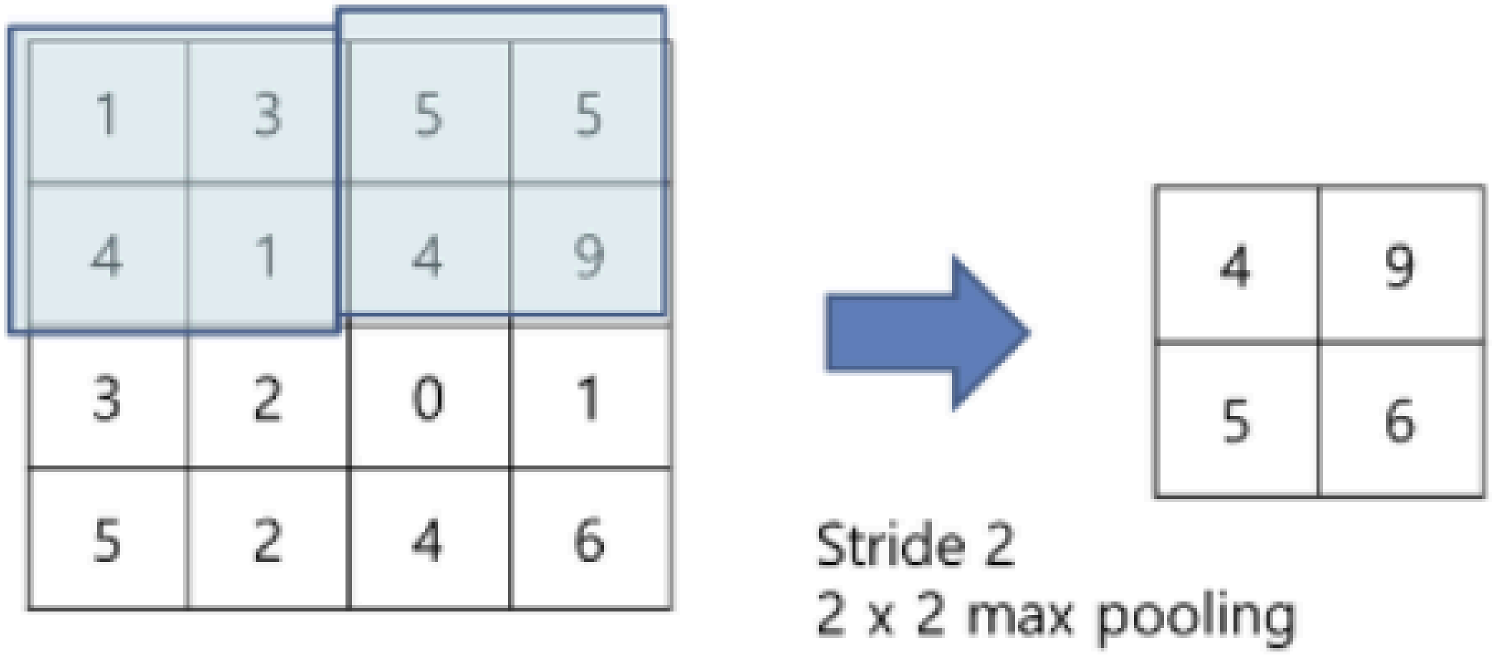


Overlapping pooling
(s < z)

AlexNet

Overlapping Pooling vs **Non-overlapping Pooling**

- more simple and computationally efficient
- larger information loss per step
- mainly for downsampling, less effective in preventing overfitting



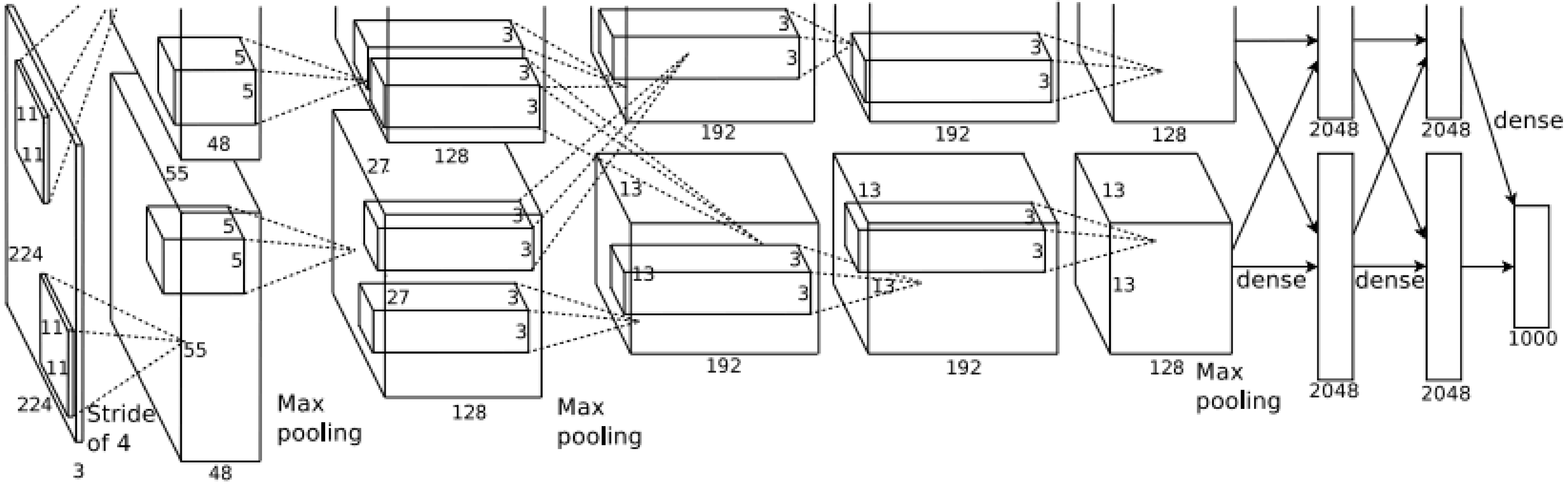
Non-overlapping pooling
(s = z)

Overlapping Pooling

- s: stride (= 2)
- z: size of the feature map (= 3x3)
- reduces top-1 & top-5 error rates by 0.4%, 0.3%

AlexNet

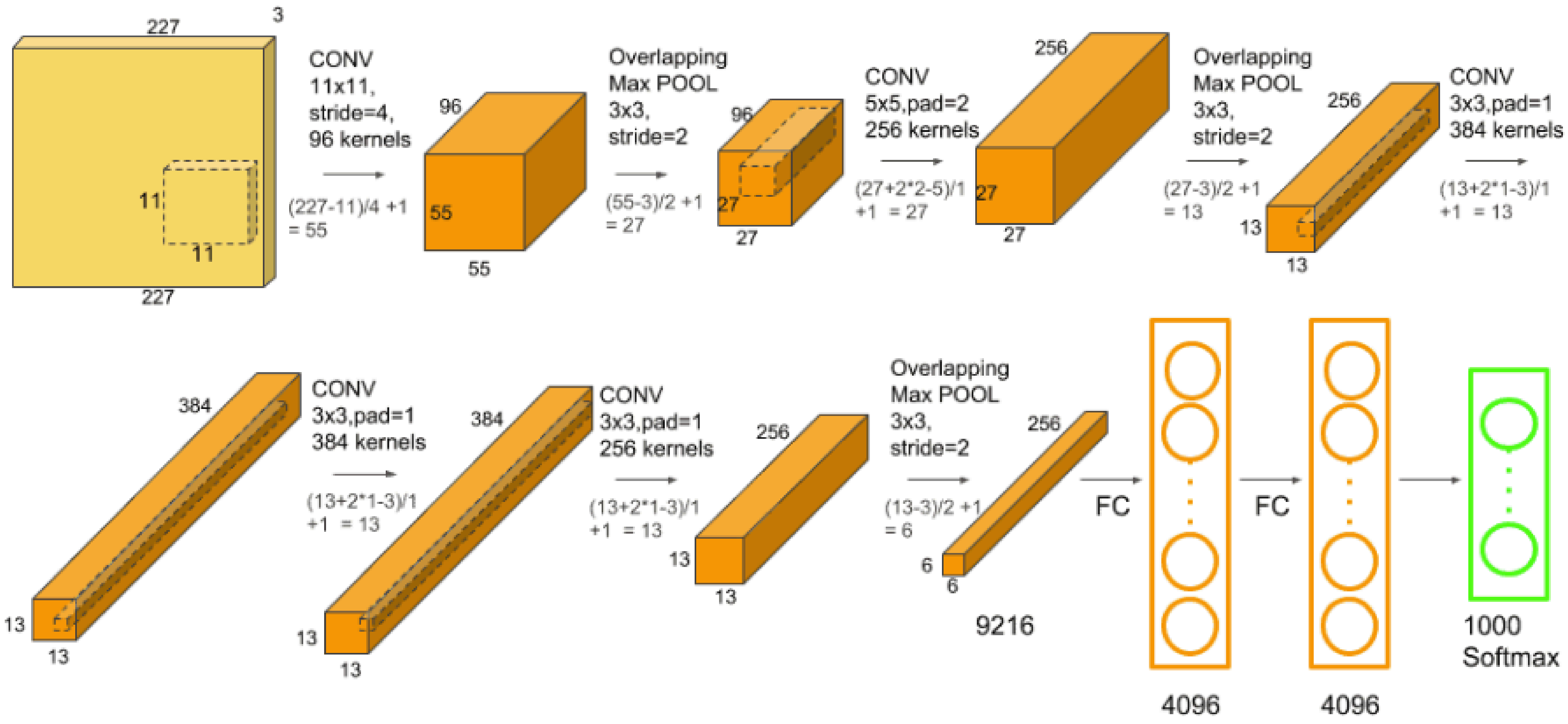
Architecture



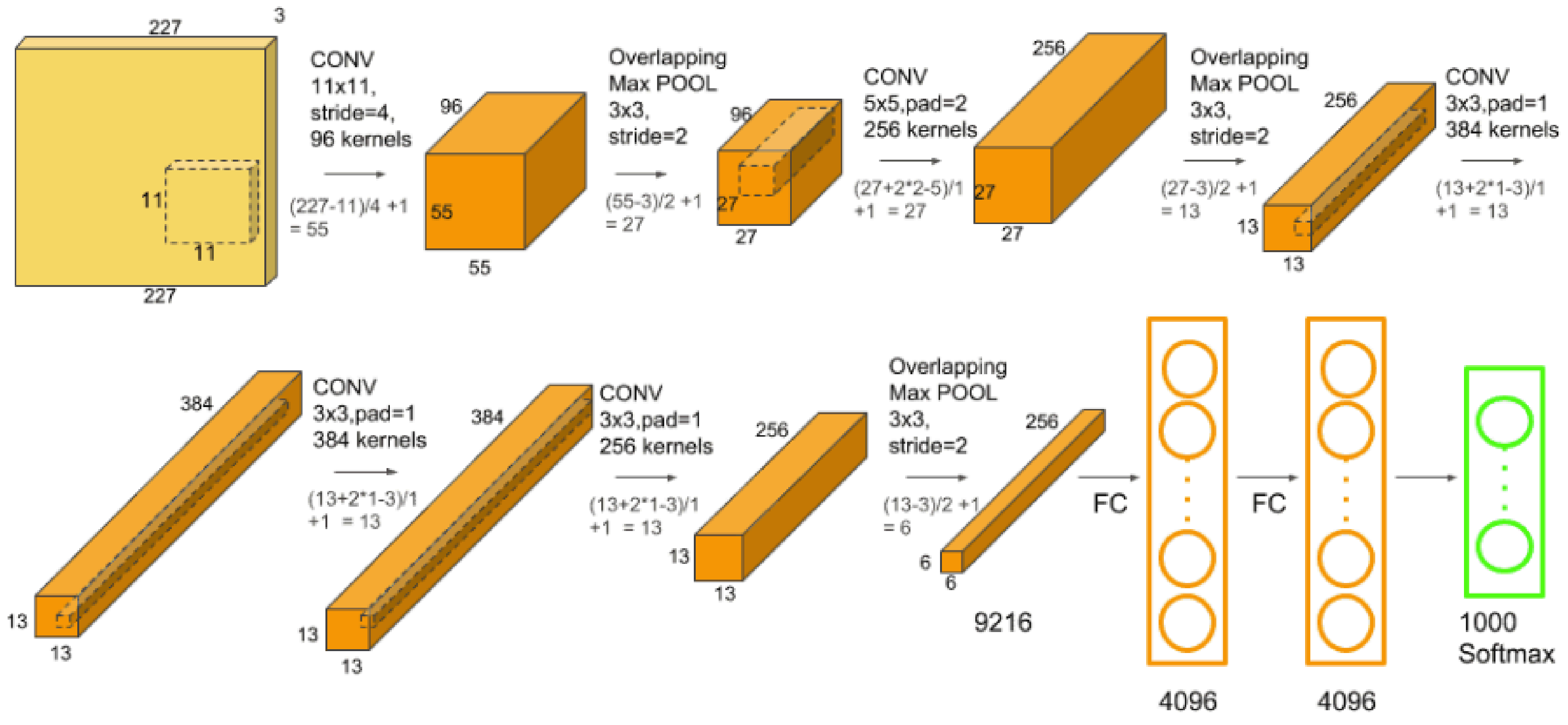
AlexNet

Architecture

- 5 Convolutional layer
- 3 Fully connected layer



AlexNet



Architecture

[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x2 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

AlexNet

Reducing Overfitting

1) Data Augmentation

- Image translations and horizontal reflections
- RGB alterations (PCA)

AlexNet

Reducing Overfitting

1) Data Augmentation

- Image translations and horizontal reflections
- RGB alterations (PCA)

$$[I_{xy}^R, I_{xy}^G, I_{xy}^B]^T + [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

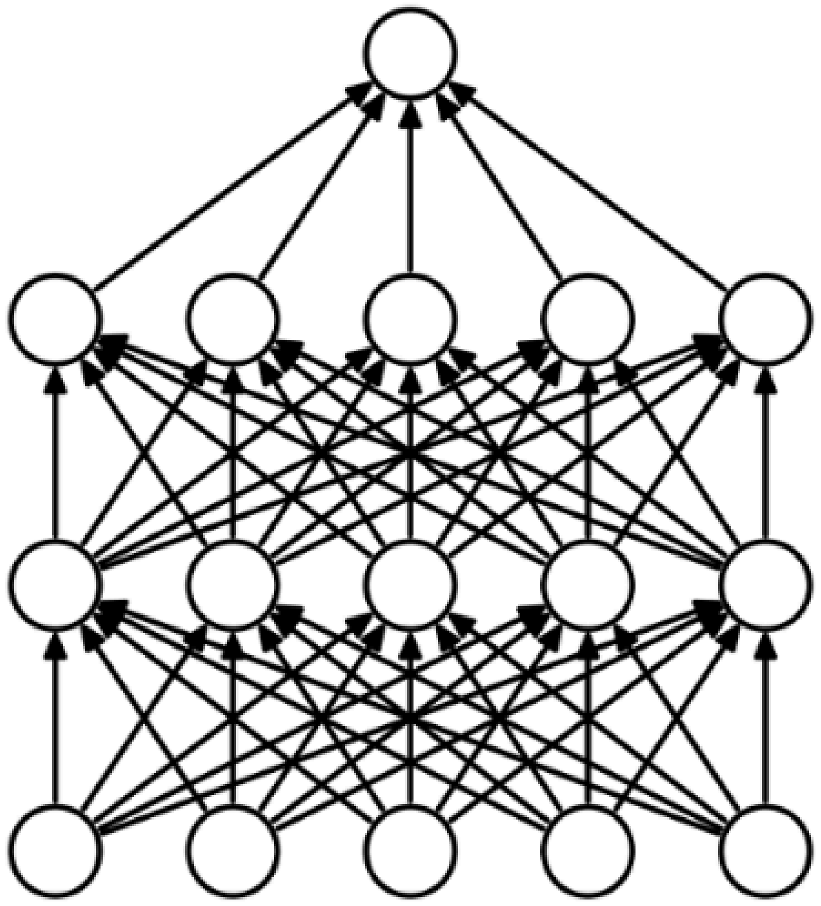
- x, y = each pixel from image
- p_i : eigenvector of the 3x3 covariance matrix of RGB pixel values
- λ_i : eigenvalue of the 3x3 covariance matrix of RGB pixel values
- α_i : random variable

AlexNet

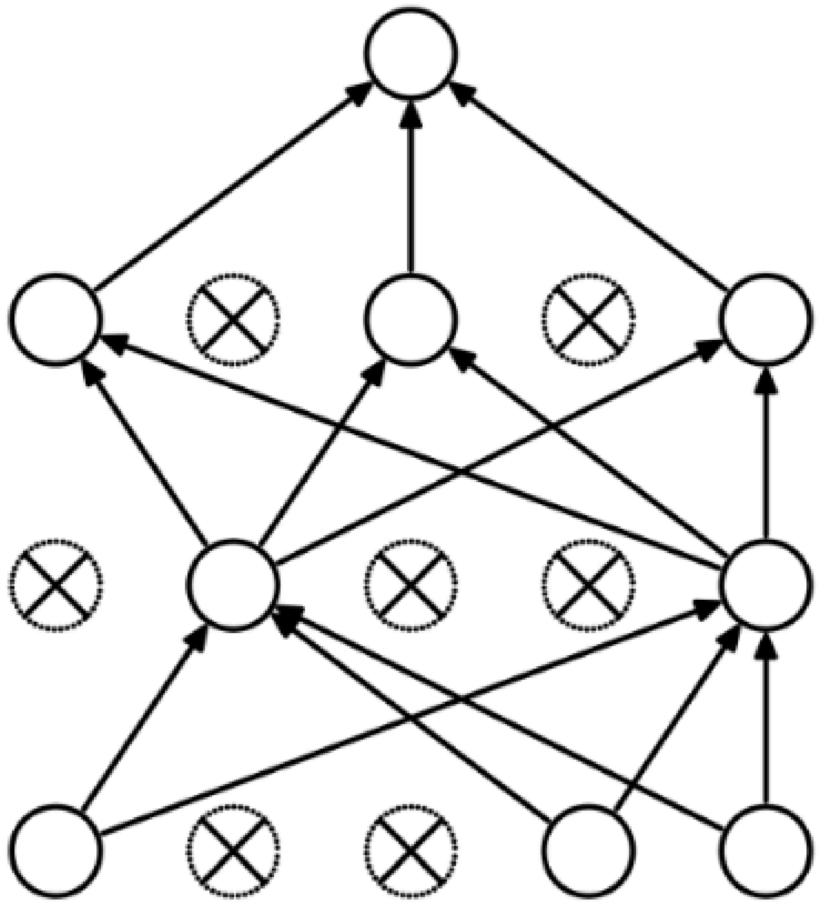
Reducing Overfitting

2) Dropout

- reduces complex connections of neurons
- forced to learn stronger features
- works well with many random neuron groups



Before dropout



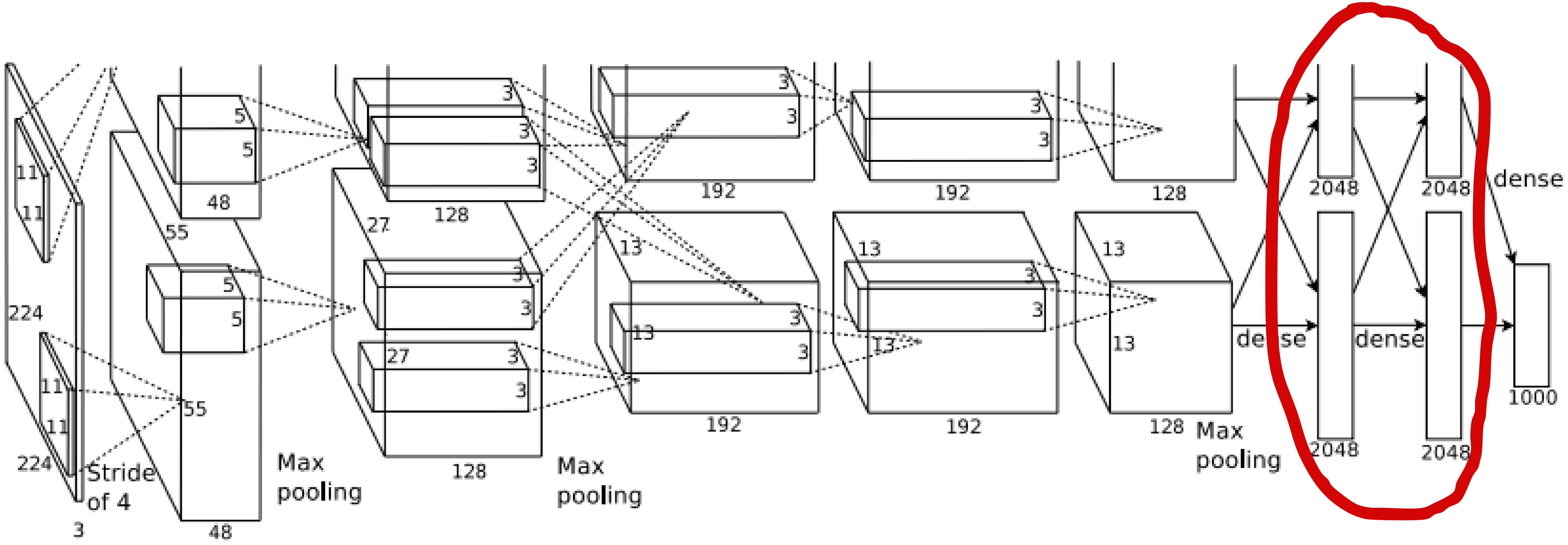
After dropout

AlexNet

Reducing Overfitting

2) Dropout

- dropout rate = 0.5



AlexNet

- Details of learning
  - batch size 128
  - SGD Momentum 0.9
  - learning rate 1e-2, reduced by 10
  - L2 weight decay 5e-4
    → reduces the model's training error
  - weight initialization
    → zero-mean Gaussian distribution & standard deviation 0.01
  - bias initialization
    → 2, 4, 5 conv layer & fc layer with constant 1
    → remaining layers with constant 0

Results

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Results

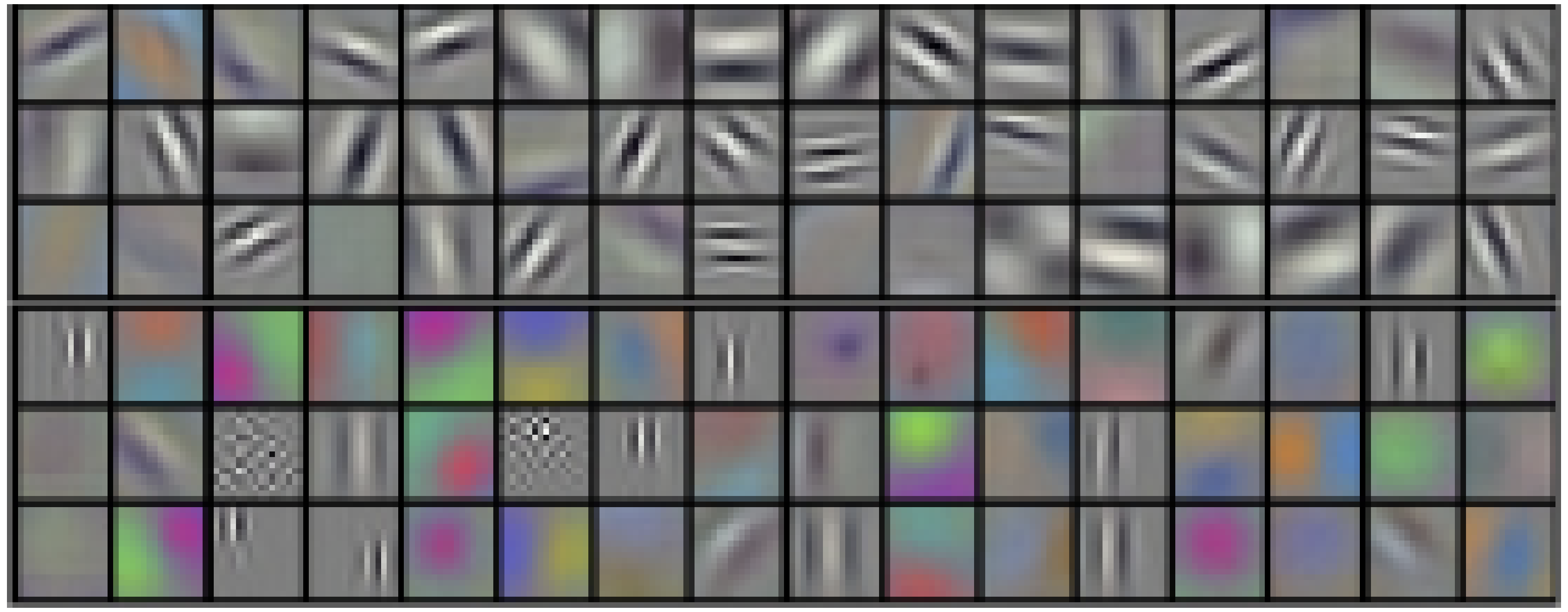
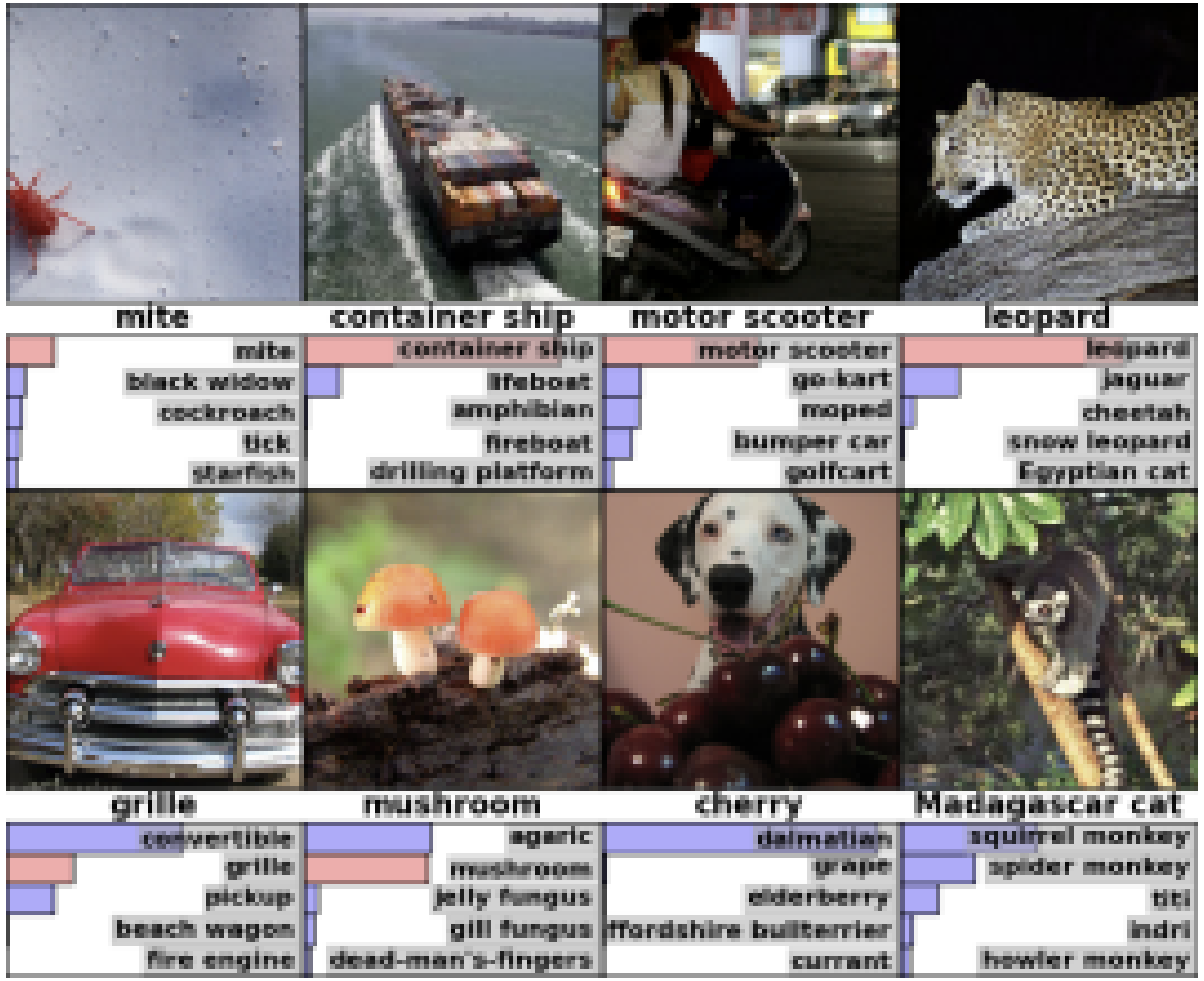


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

Results


mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Thank You