AlexNet

Computer Vision

Carnegie Mellon University (Kris Kitani)

ImageNet Classification with Deep Convolutional Neural Networks

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

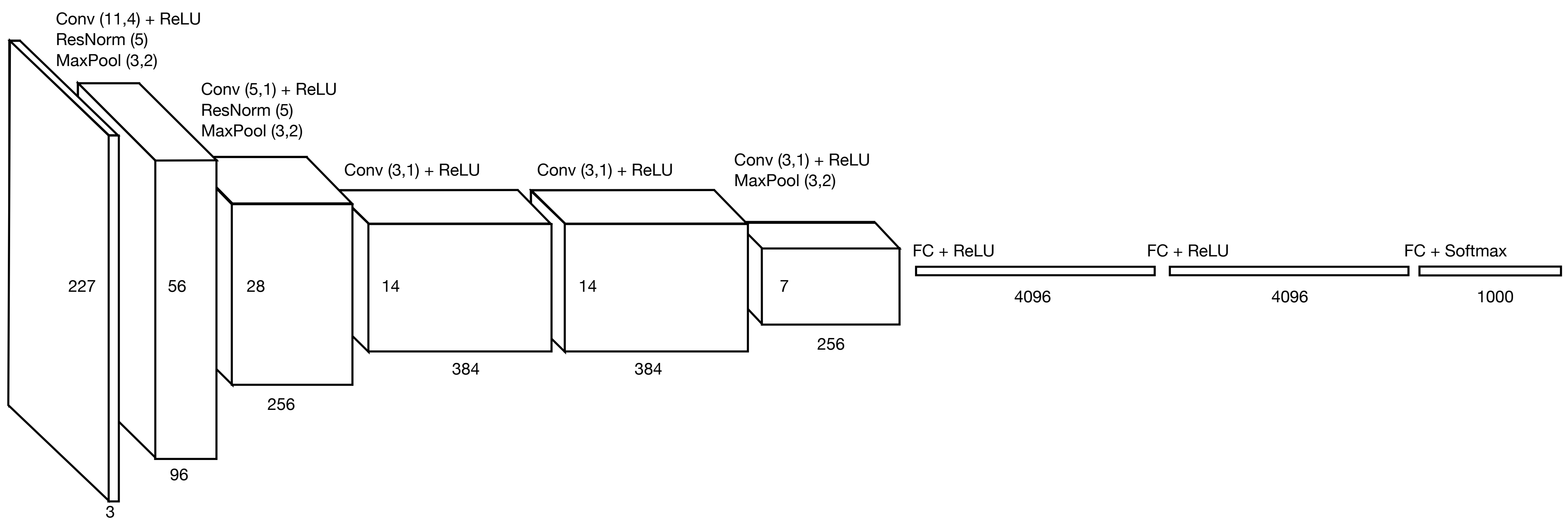
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

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

Abstract

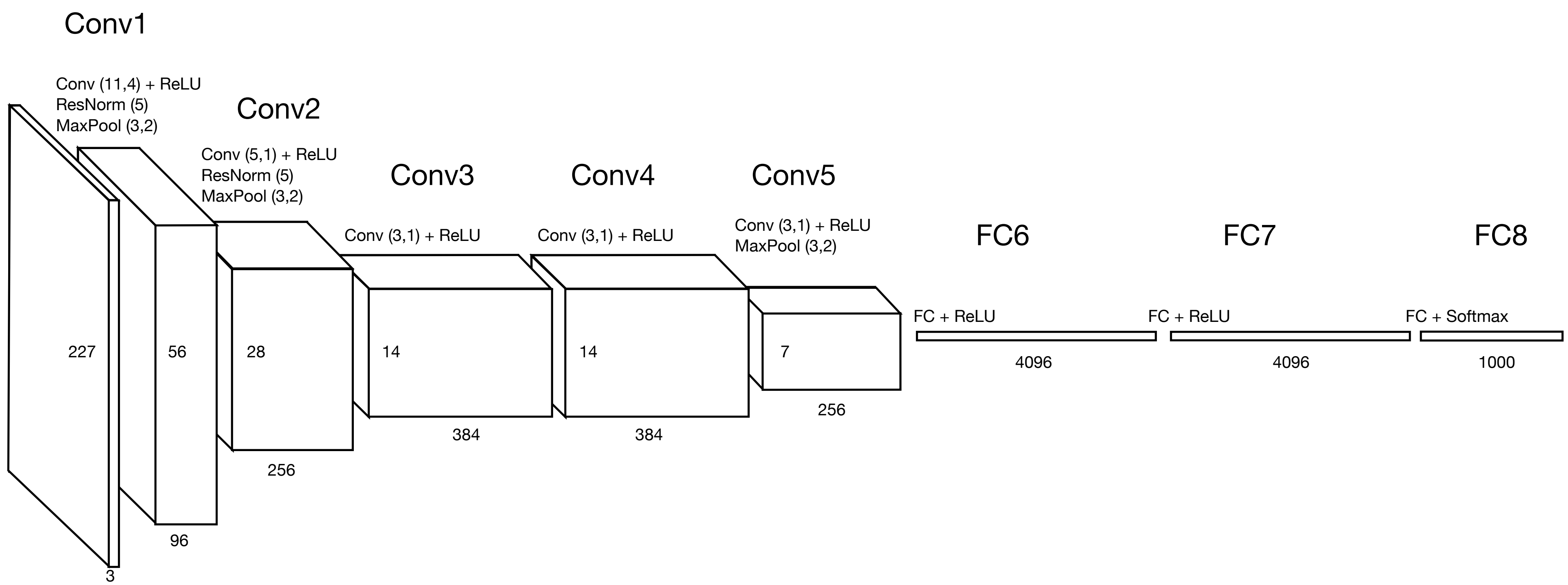
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. 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.

AlexNet

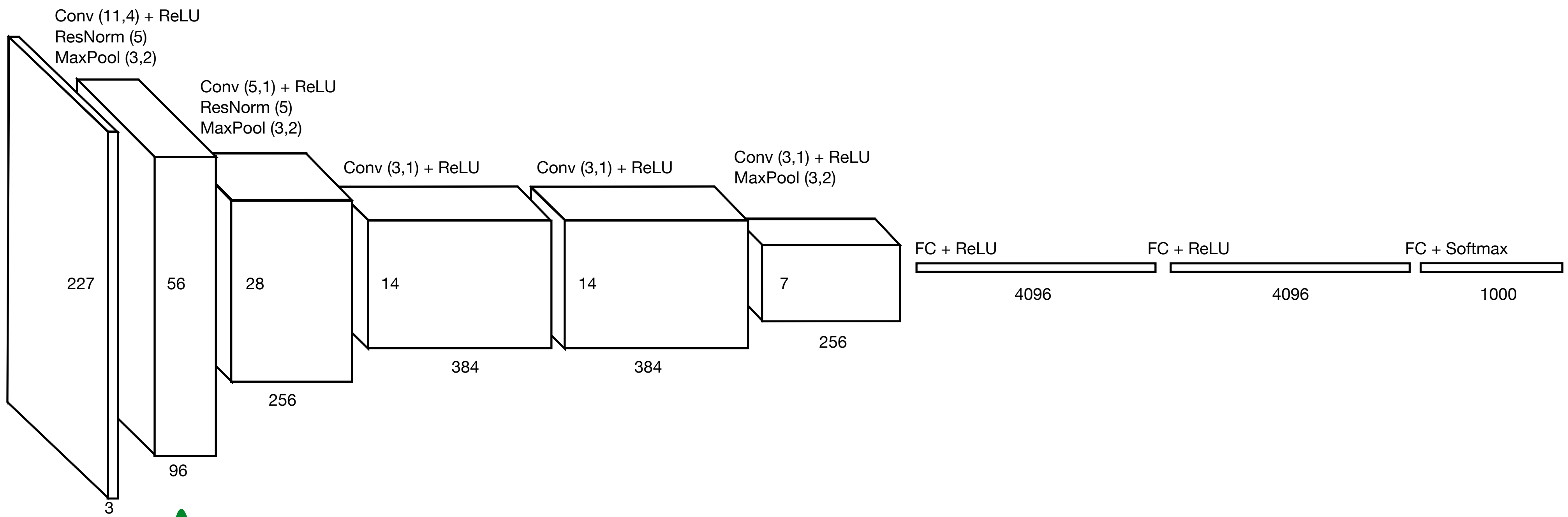


https://github.com/BVLC/caffe/blob/master/models/bvlc_alexnet/train_val.prototxt

AlexNet



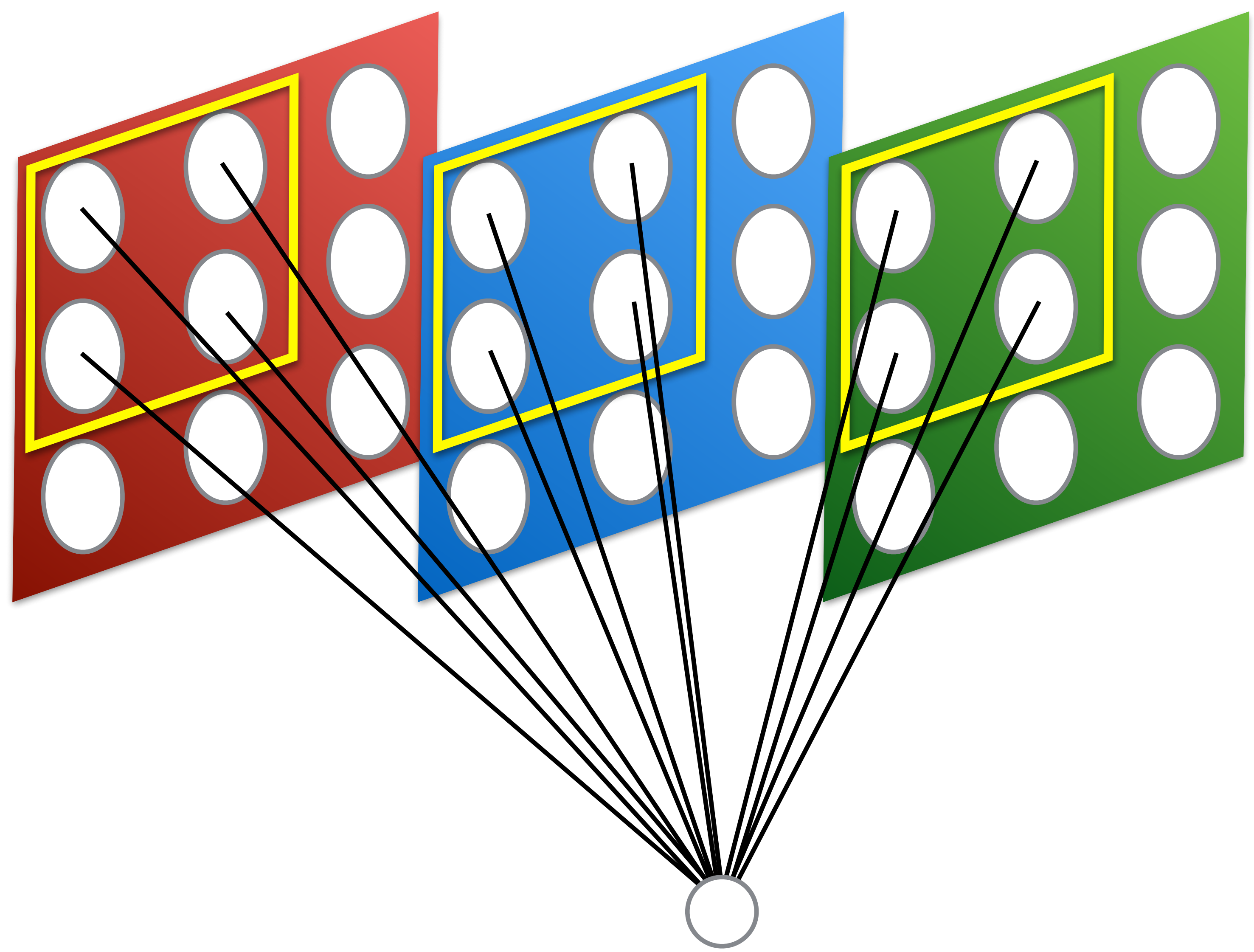
https://github.com/BVLC/caffe/blob/master/models/bvlc_alexnet/train_val.prototxt



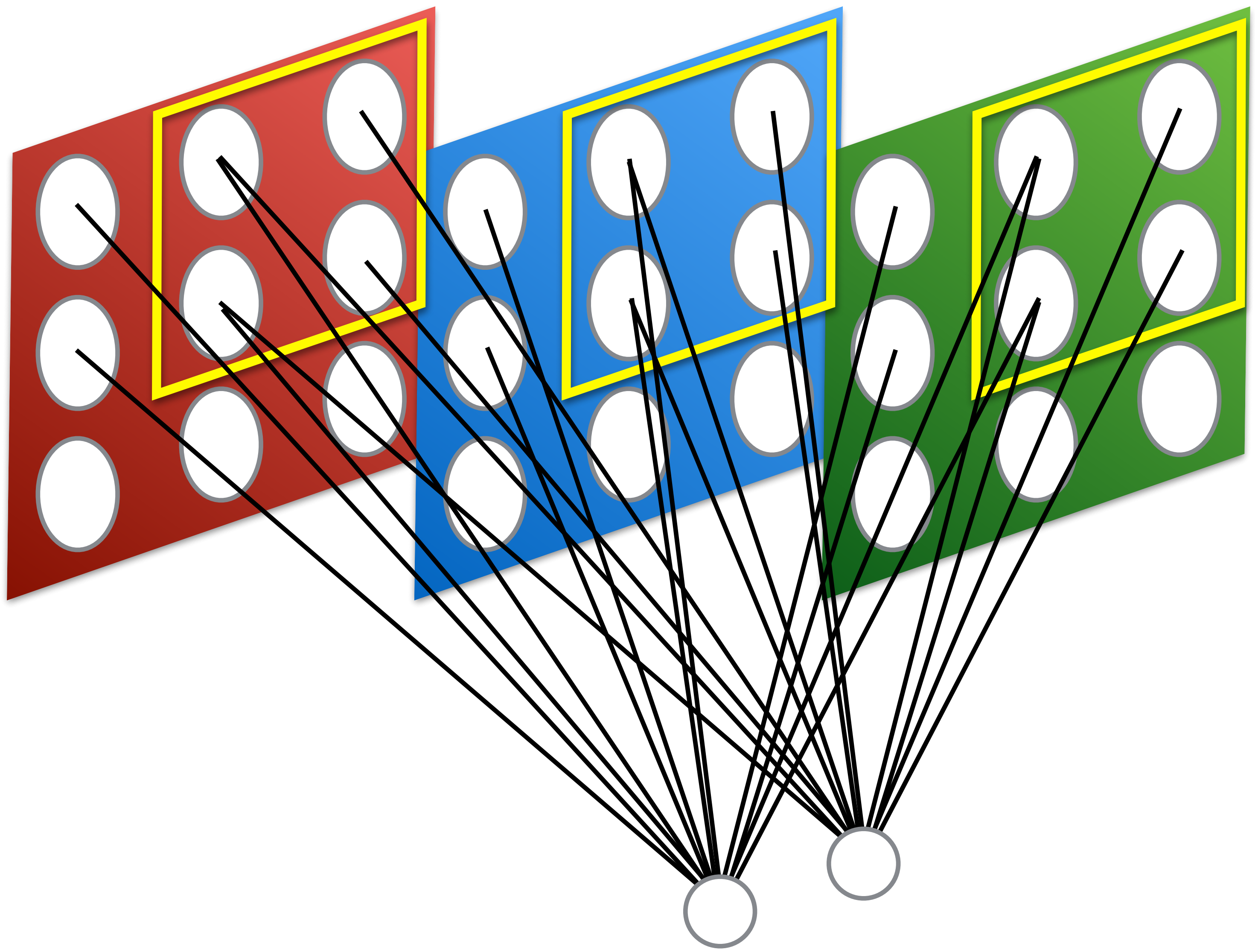
Conv1

(actually a convolution, ReLU, response normalization and max pooling)

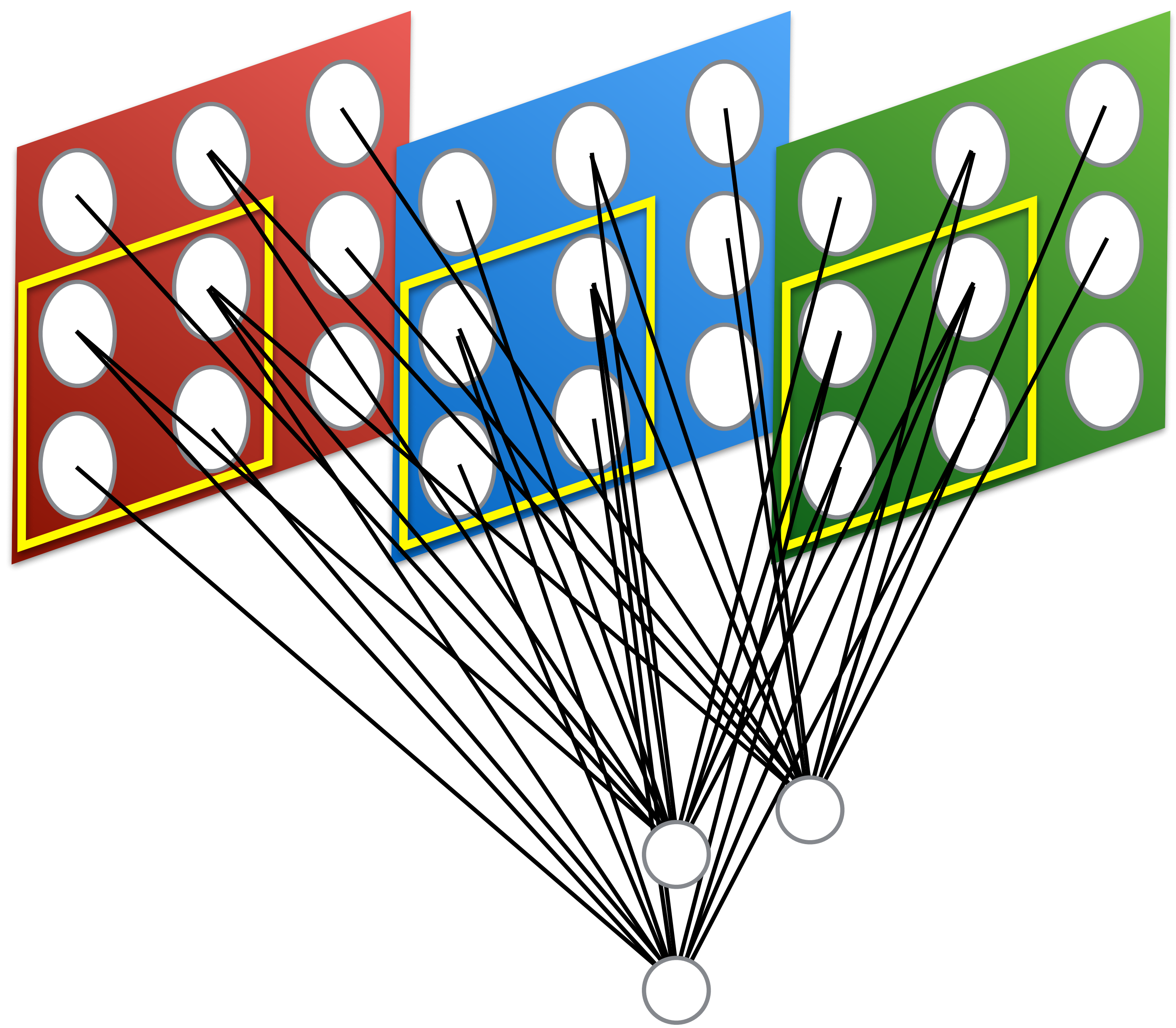
Recall:



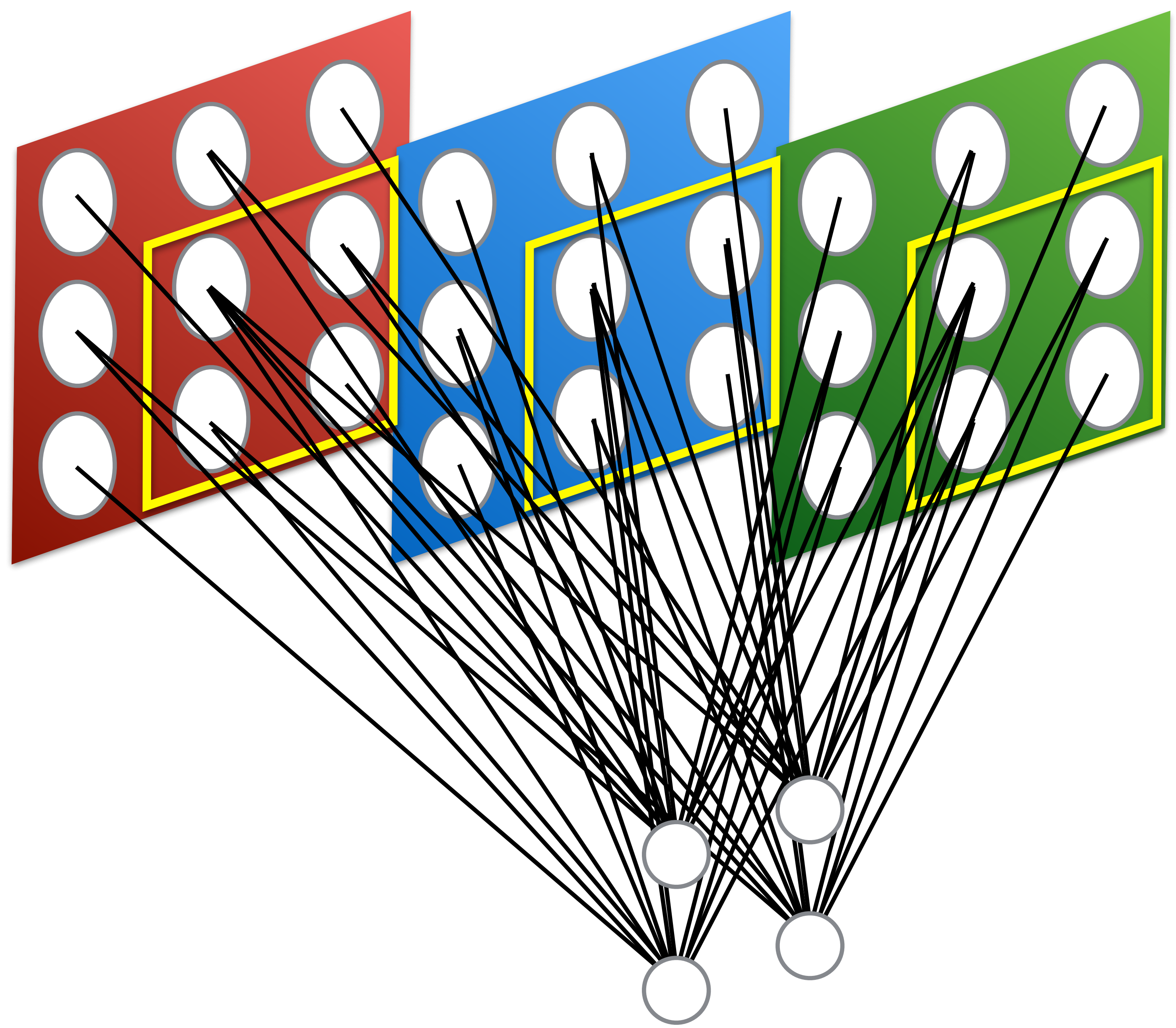
Recall:



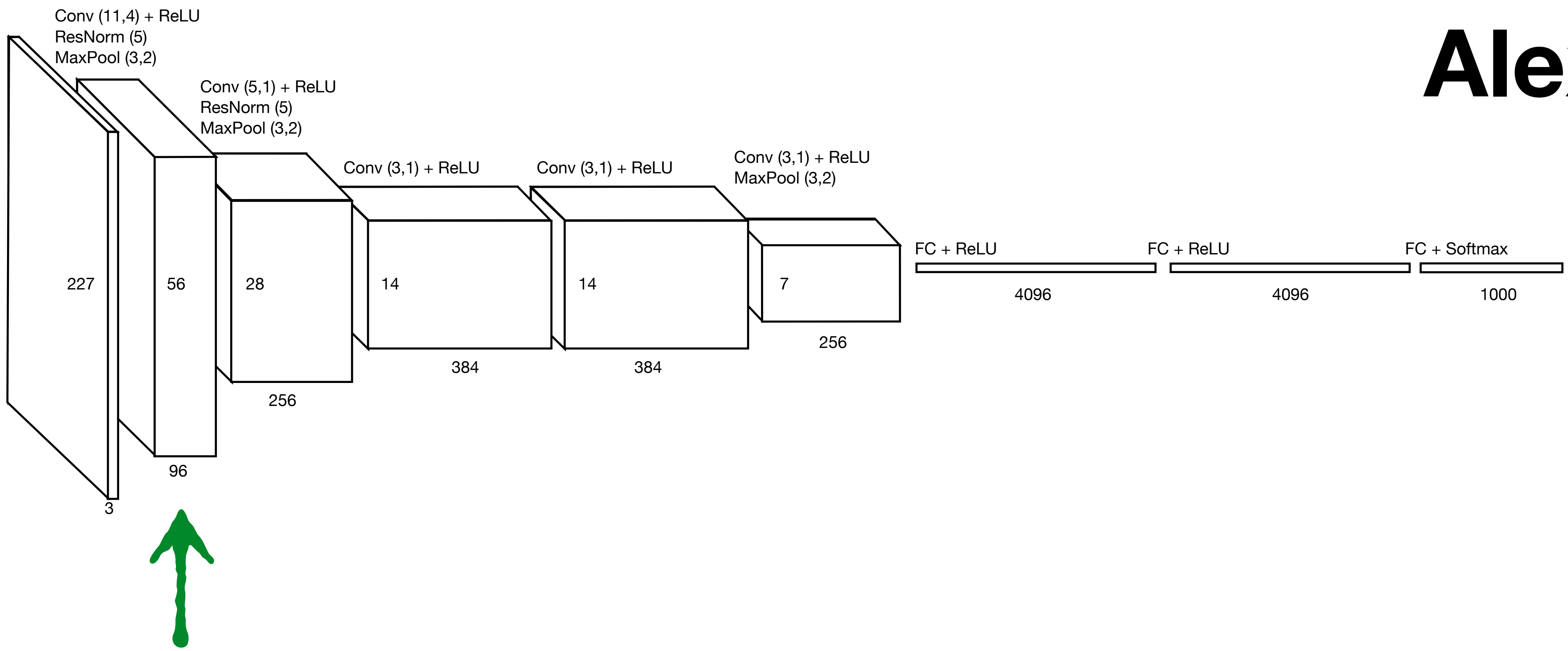
Recall:



Recall:



AlexNet



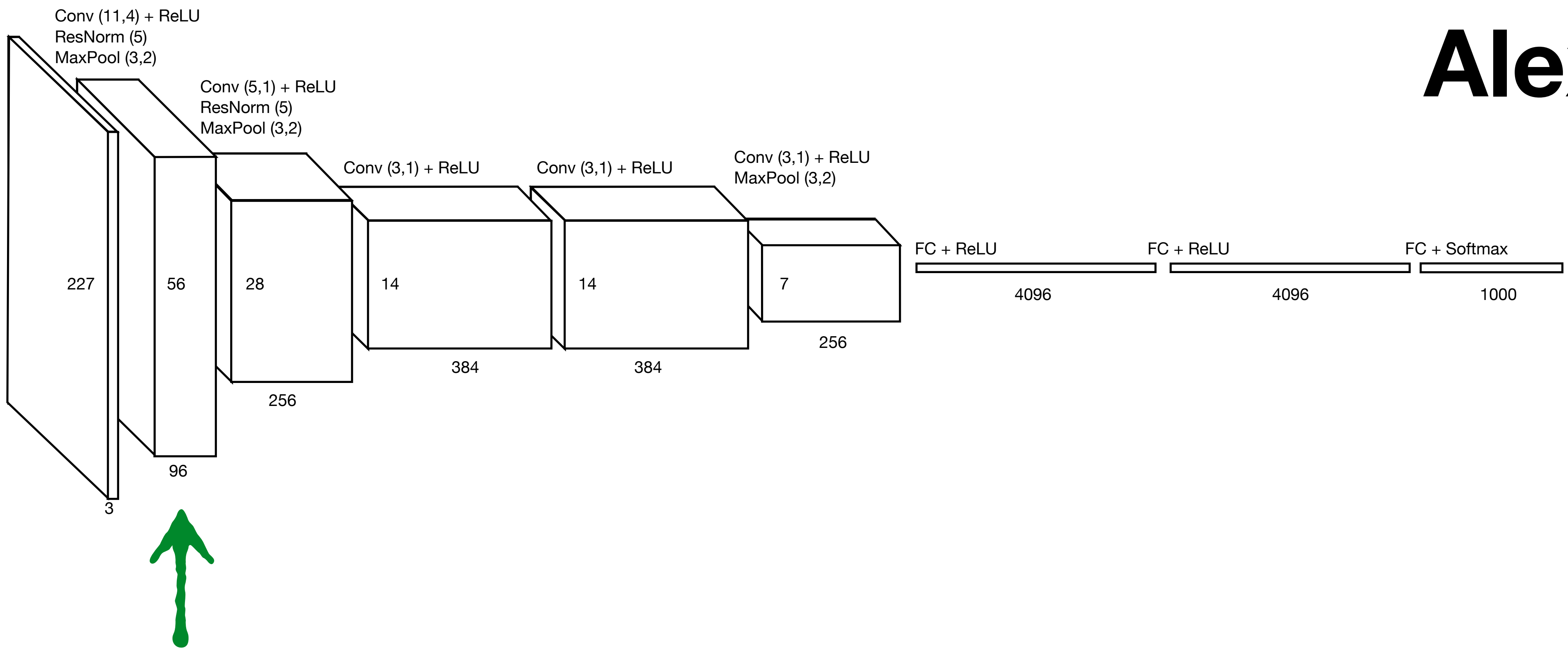
Conv1

(actually a **convolution**, ReLU, response normalization and max pooling)

Convolution with size: 11 x 11 x 3

How many filters in Conv1?

AlexNet



Conv1

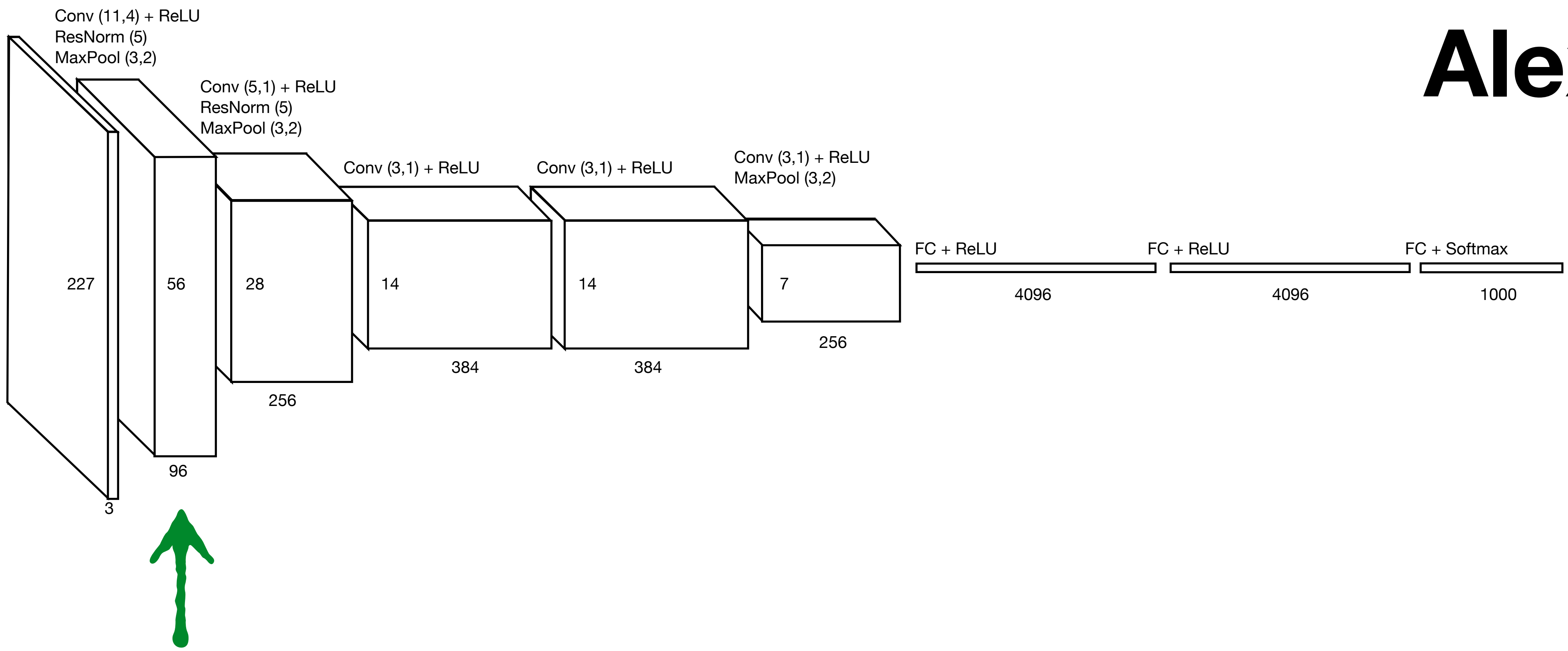
(actually a **convolution**, ReLU, response normalization and max pooling)

Convolution with size: 11 x 11 x 3

How many filters in Conv1? 96

What is the size of the response map after convolution?

AlexNet



Conv1

(actually a **convolution**, ReLU, response normalization and max pooling)

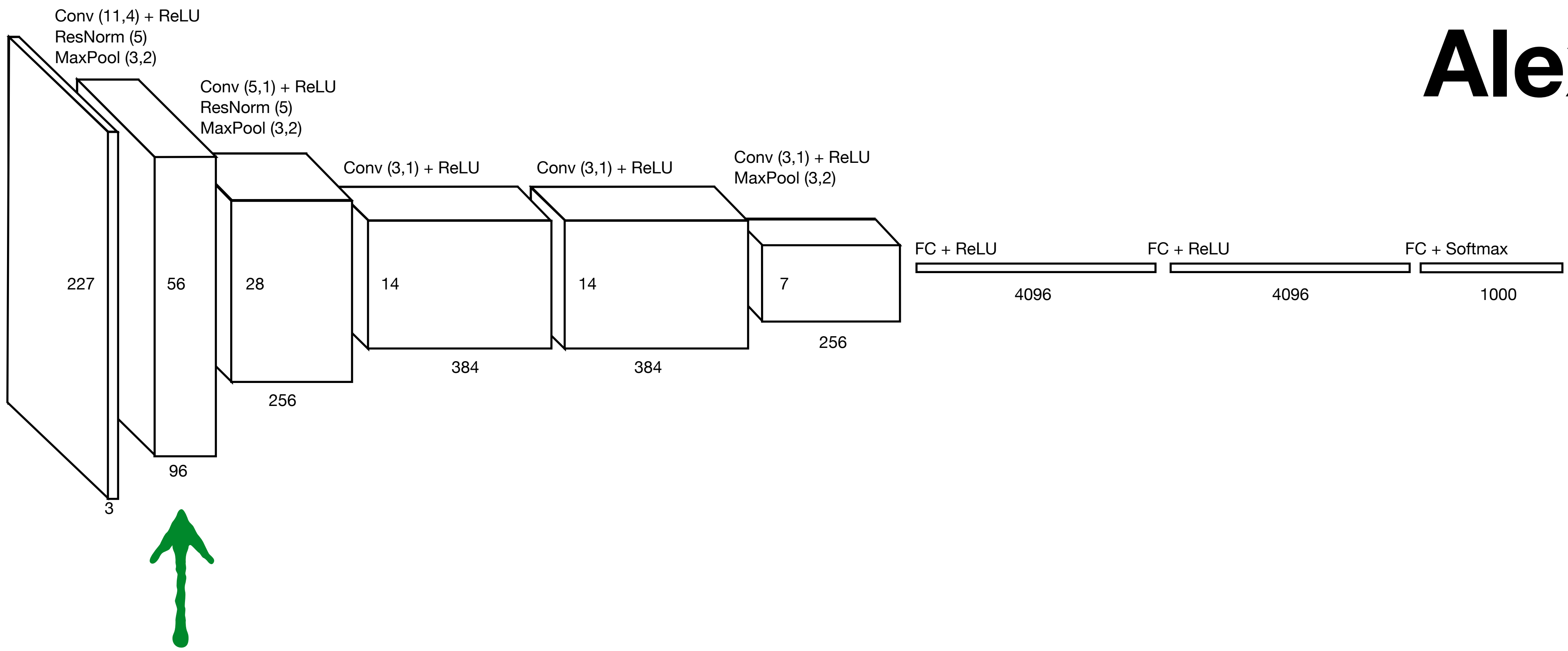
Convolution with size: 11 x 11 x 3

How many filters in Conv1? 96

What is the size of the response map after convolution? 55

What is the stride of the convolutions?

AlexNet



Conv1

(actually a **convolution**, ReLU, response normalization and max pooling)

Convolution with size: 11 x 11 x 3

How many filters in Conv1? 96

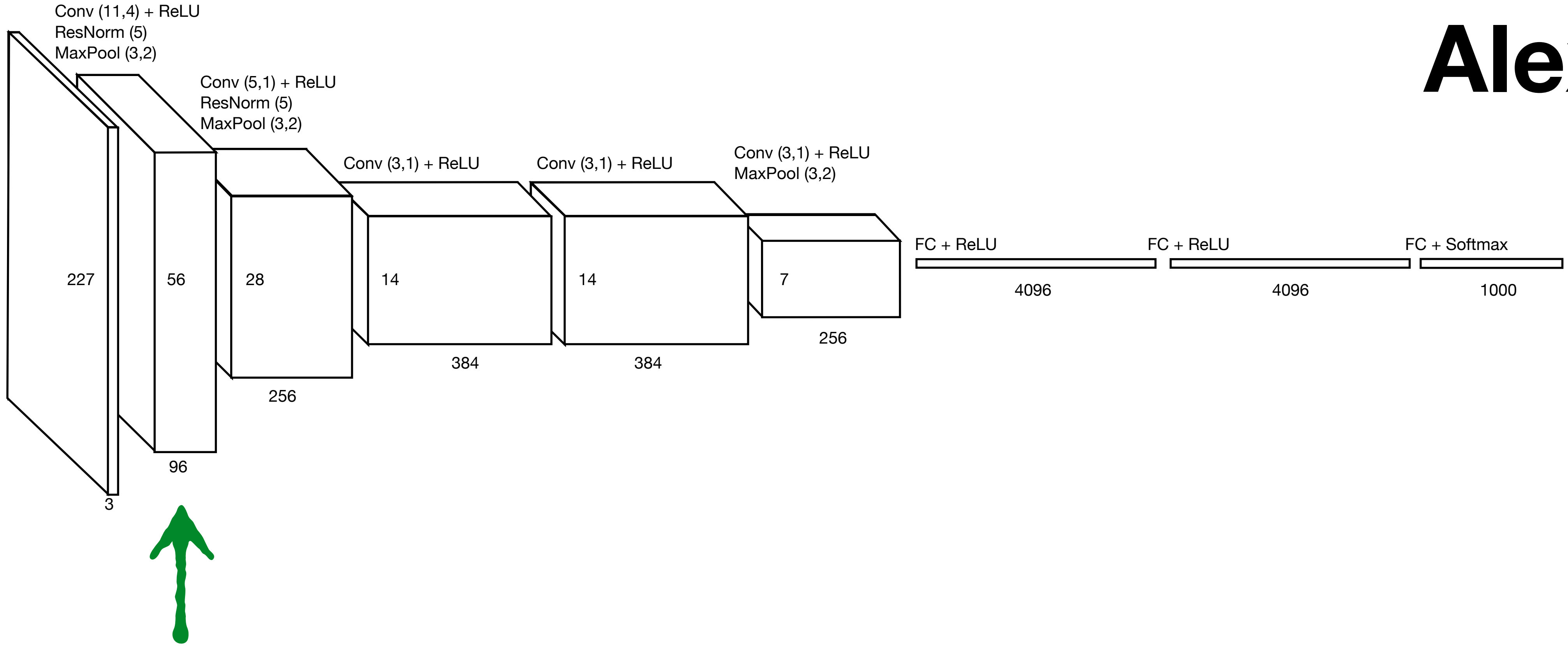
What is the size of the response map after convolution? 55

What is the stride of the convolutions? 4

Shorthand notation: Conv(11, 4)

Size Stride

AlexNet



Conv1

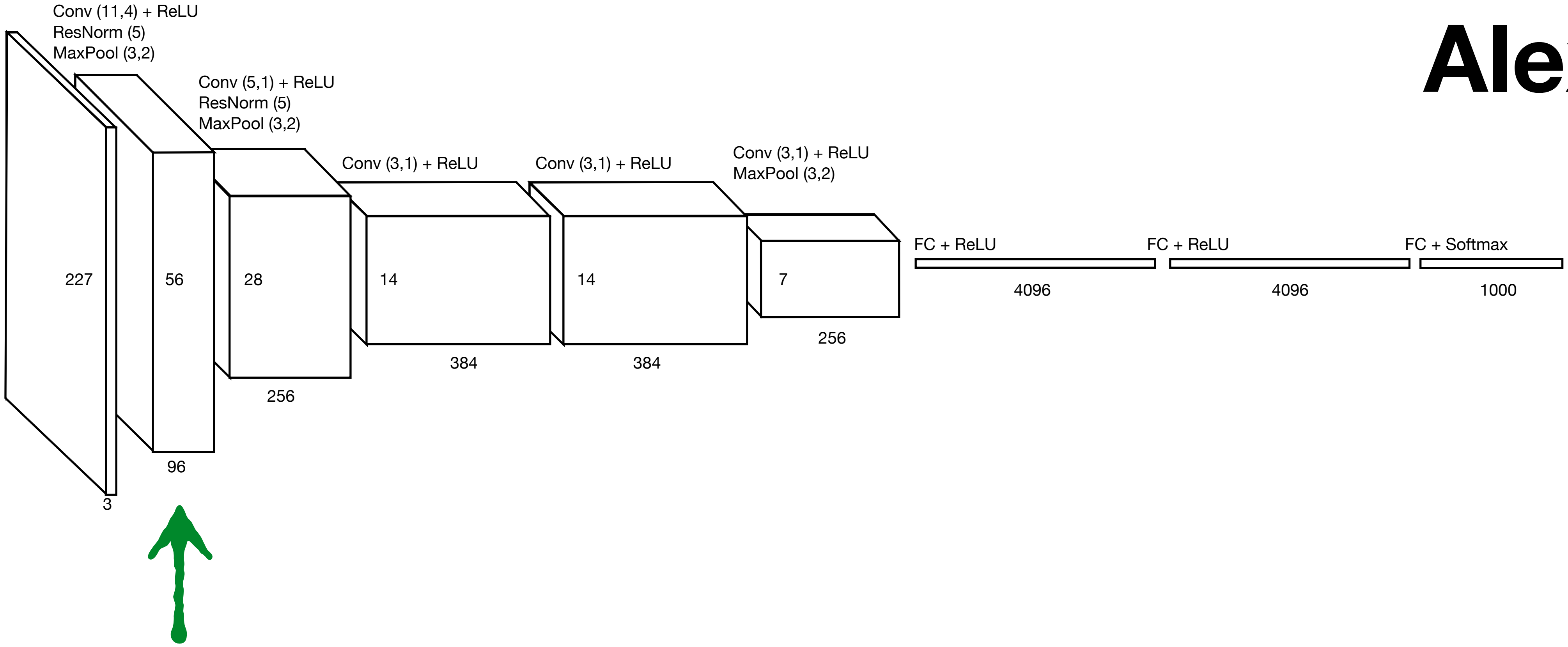
(actually a convolution, **ReLU**, response normalization and max pooling)

Regularized Linear Unit (ReLU)

$$f(x) = \max(0, x)$$

Pixel-wise operator

AlexNet



Conv1

(actually a convolution, **ReLU**, response normalization and max pooling)

Regularized Linear Unit (ReLU)

$$f(x) = \max(0, x)$$

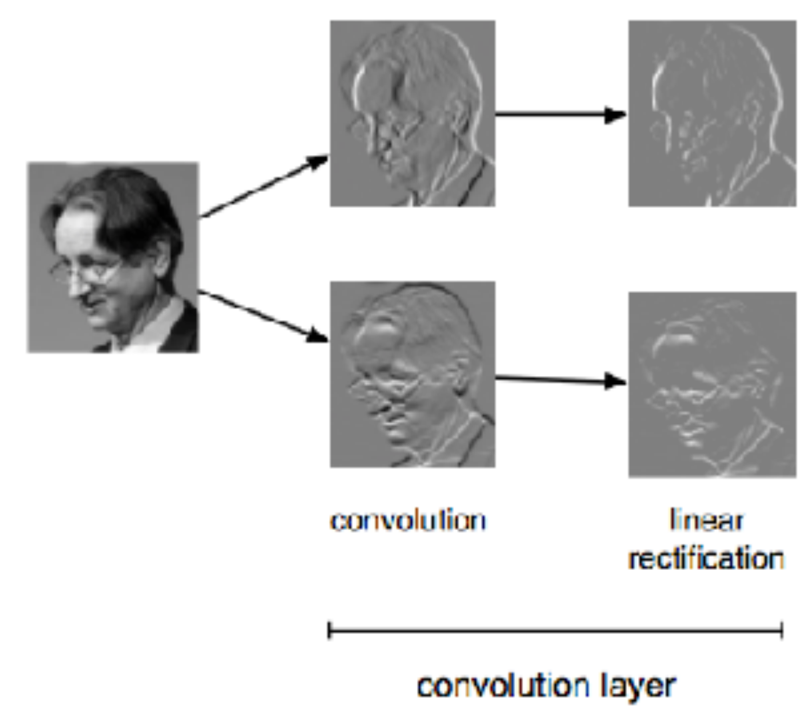
Pixel-wise operator

'Keeps all of the positive activations and throws away all negative!'

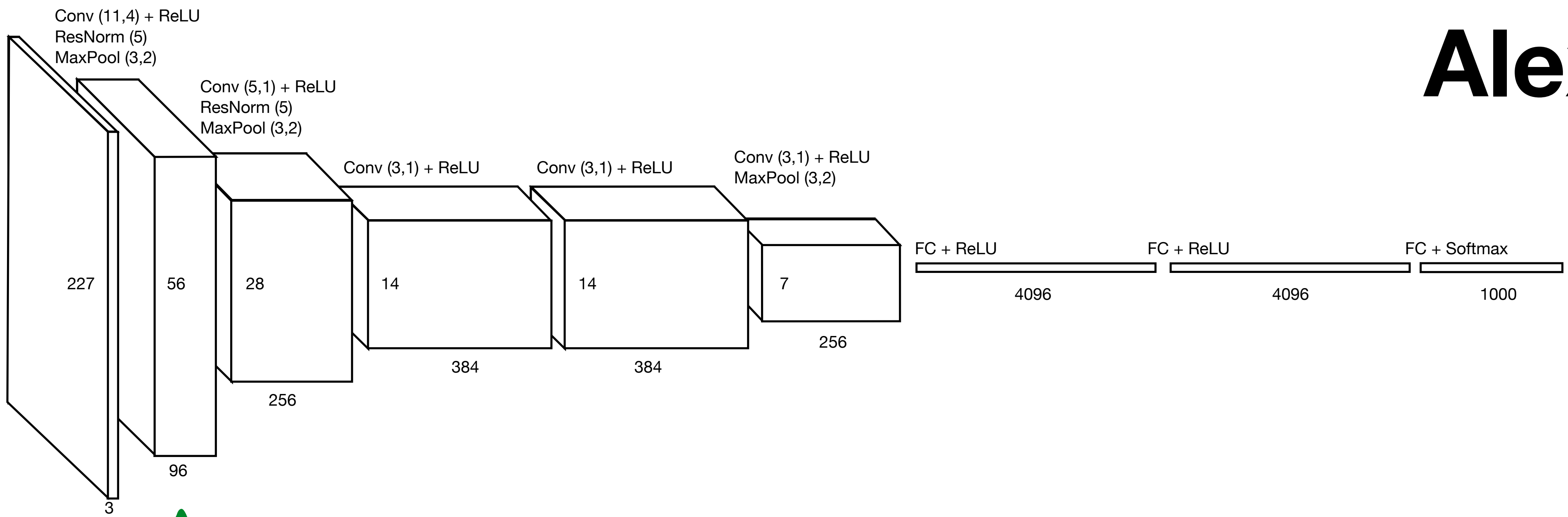
Why would we want a non-linear activation function like this?

Why use ReLU?

- Two linear functions (convolutions) is the same as one linear function (convolution).
- ReLU adds a non-linearity and is more powerful if you want to stack convolutions.
- Makes gradients sparse. Helps avoid vanishing gradients.
- Really speeds up convergence of gradient descent



AlexNet



Conv1

(actually a convolution, **ReLU**, response normalization and max pooling)

Regularized Linear Unit (ReLU)

$$f(x) = \max(0, x)$$

'Keeps all of the positive activations'

Speeds up network training significantly

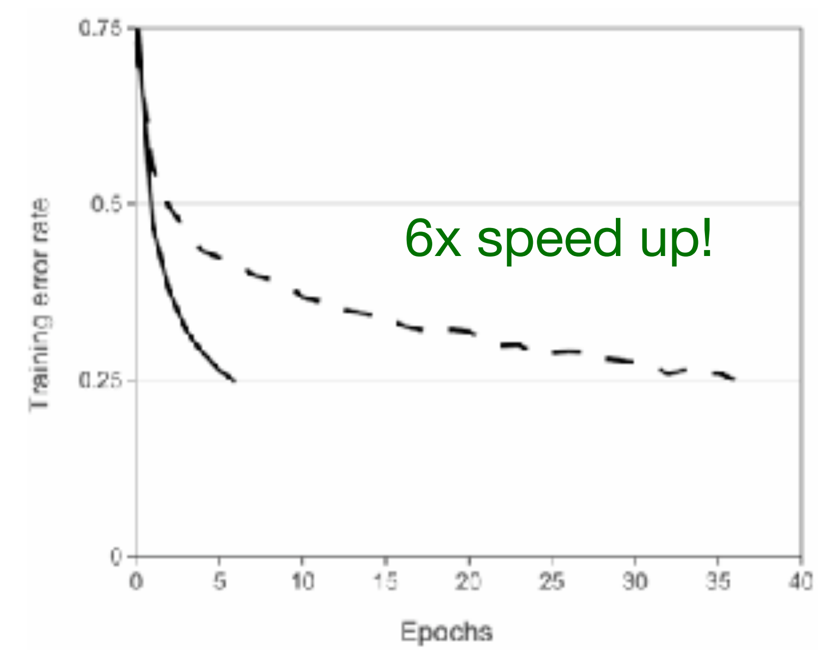
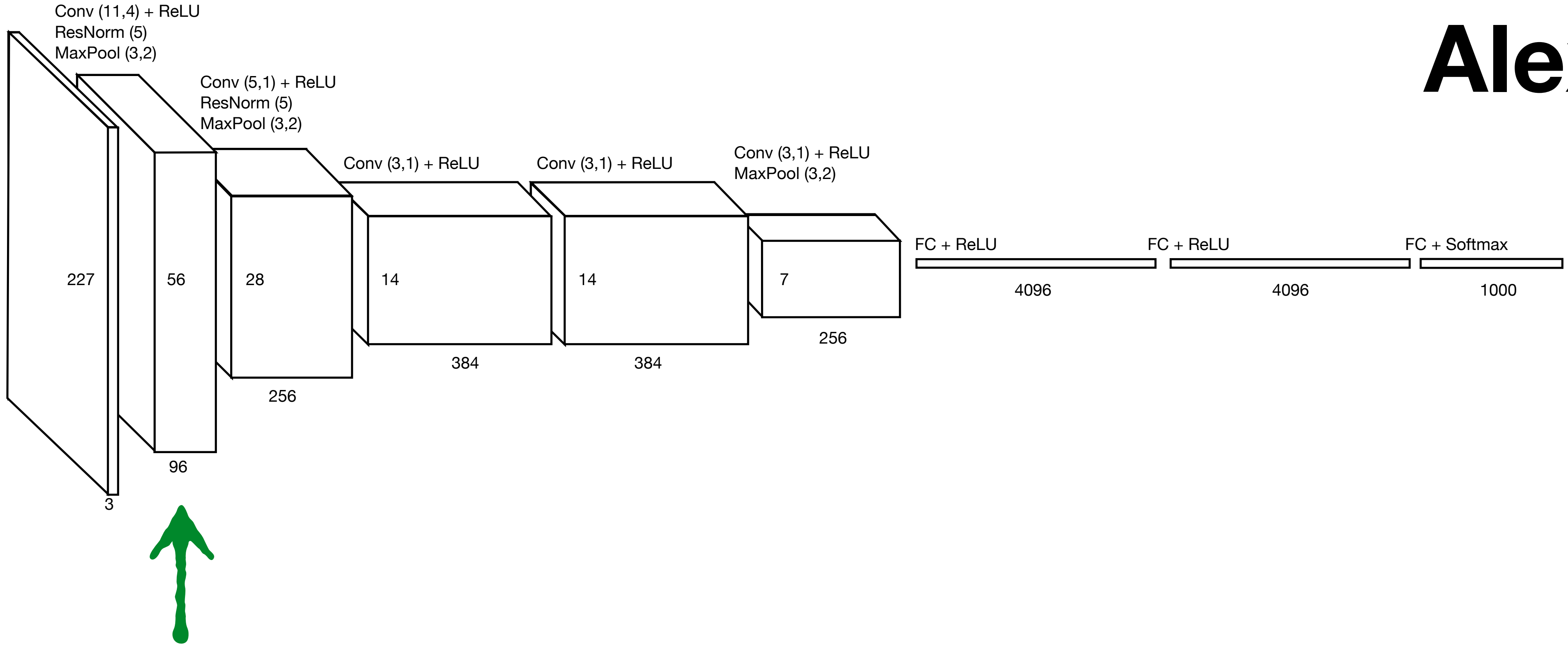


Figure 1: A four-layer convolutional neural network with ReLUs (**solid line**) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons (**dashed line**). The learning rates for each net-

V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning*, 2010.

AlexNet

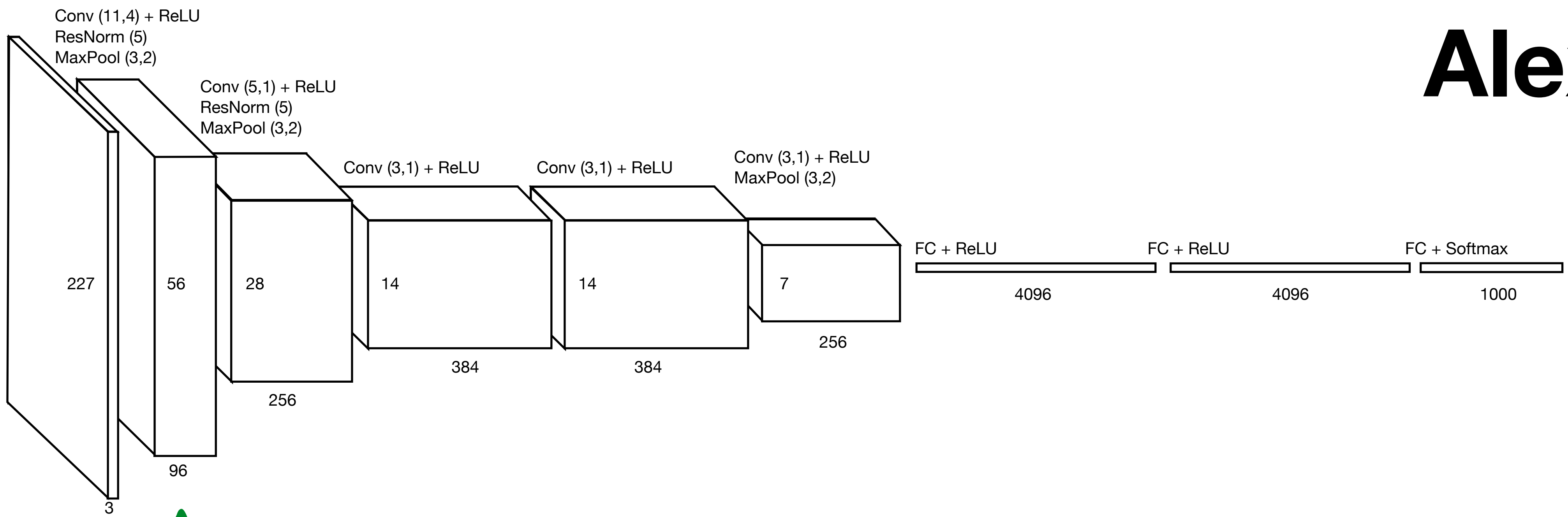


Conv1

(actually a convolution, ReLU, **response normalization** and max pooling)

Local response normalization

AlexNet



Conv1

(actually a convolution, ReLU, **response normalization** and max pooling)

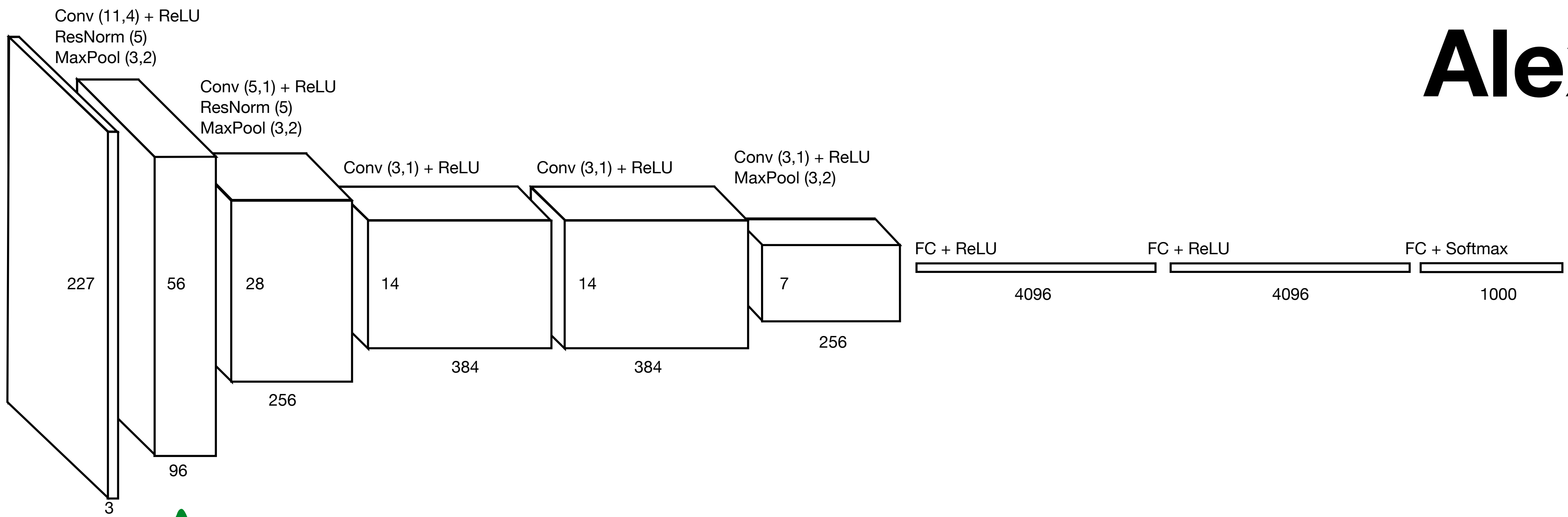
Local response normalization

channel

$$a^{i}_{x,y}$$

Activation location

AlexNet



Conv1

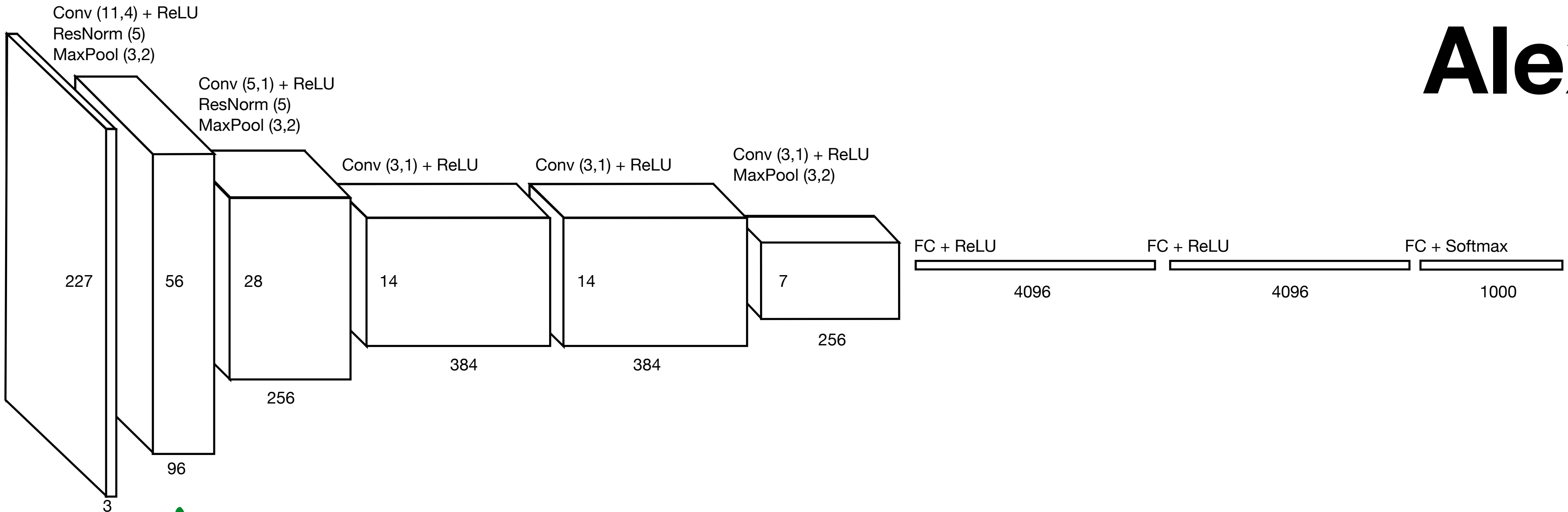
(actually a convolution, ReLU, **response normalization** and max pooling)

Local response normalization

normalized activation

$$b^i_{x,y} = \frac{a^i_{x,y}}{\quad}$$

AlexNet



Conv1

(actually a convolution, ReLU, **response normalization** and max pooling)

Local response normalization

channel

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=-2}^{2} (a^j_{x,y})^2\right)^{\beta}}$$

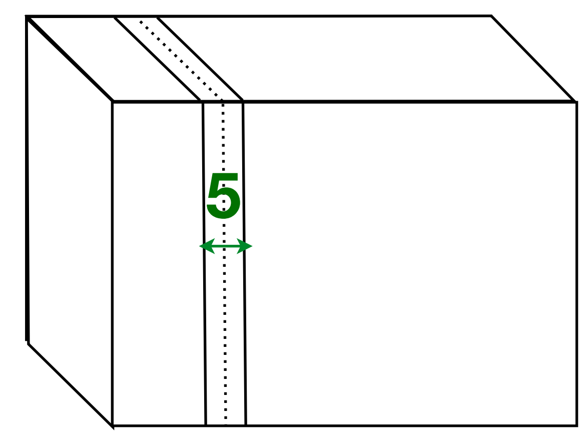
Activation location

offset (2)

Scale (10^{-4})

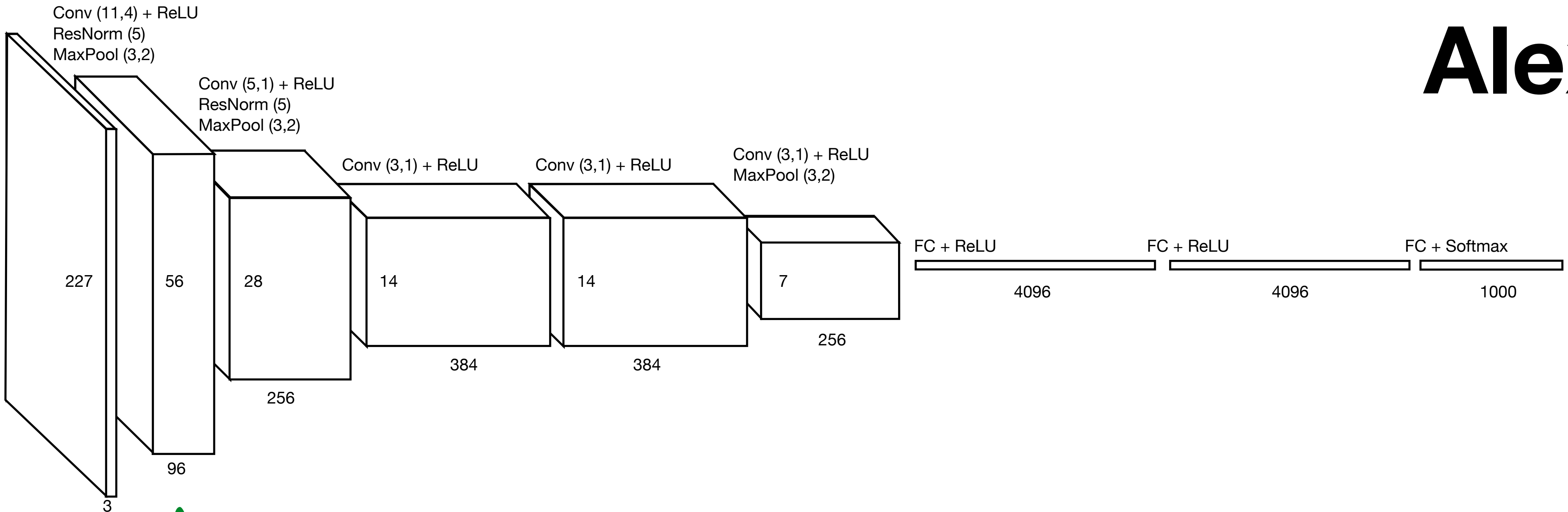
5 channel window

Non-linear scaling (0.75)



Activations averaged across neighboring channels (not spatially averaging)

AlexNet



Conv1

(actually a convolution, ReLU, **response normalization** and max pooling)

Local response normalization

channel

'ResNorm(5)'

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=-2}^{2} (a^j_{x,y})^2\right)^\beta}$$

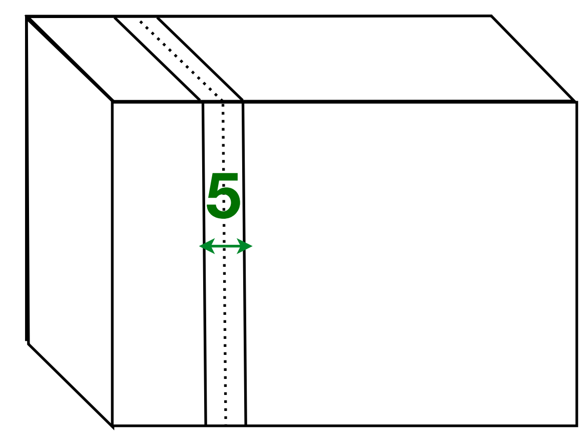
Activation location

offset (2)

Scale (10^{-4})

5 channel window

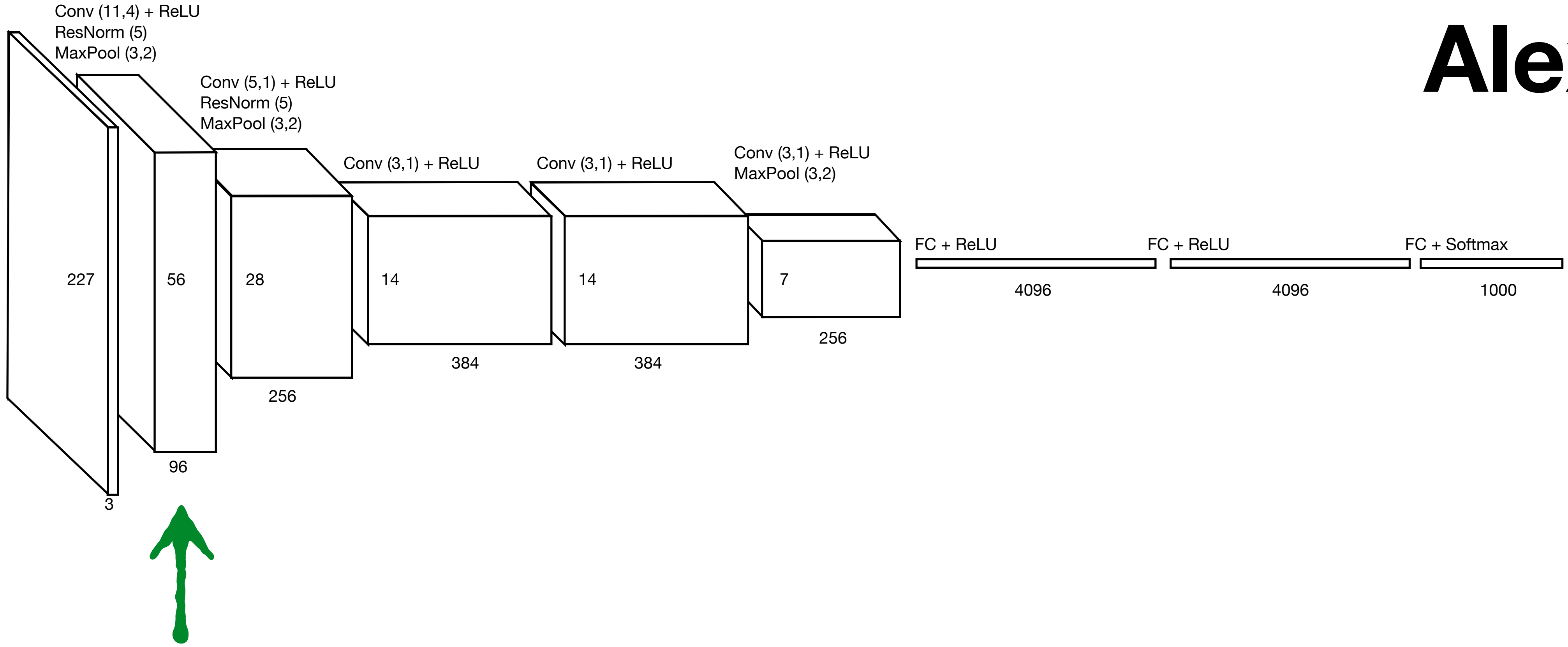
Non-linear scaling (0.75)



Activations averaged across neighboring channels (not spatially averaging)

1.2~1.4% reduction in error rate

AlexNet

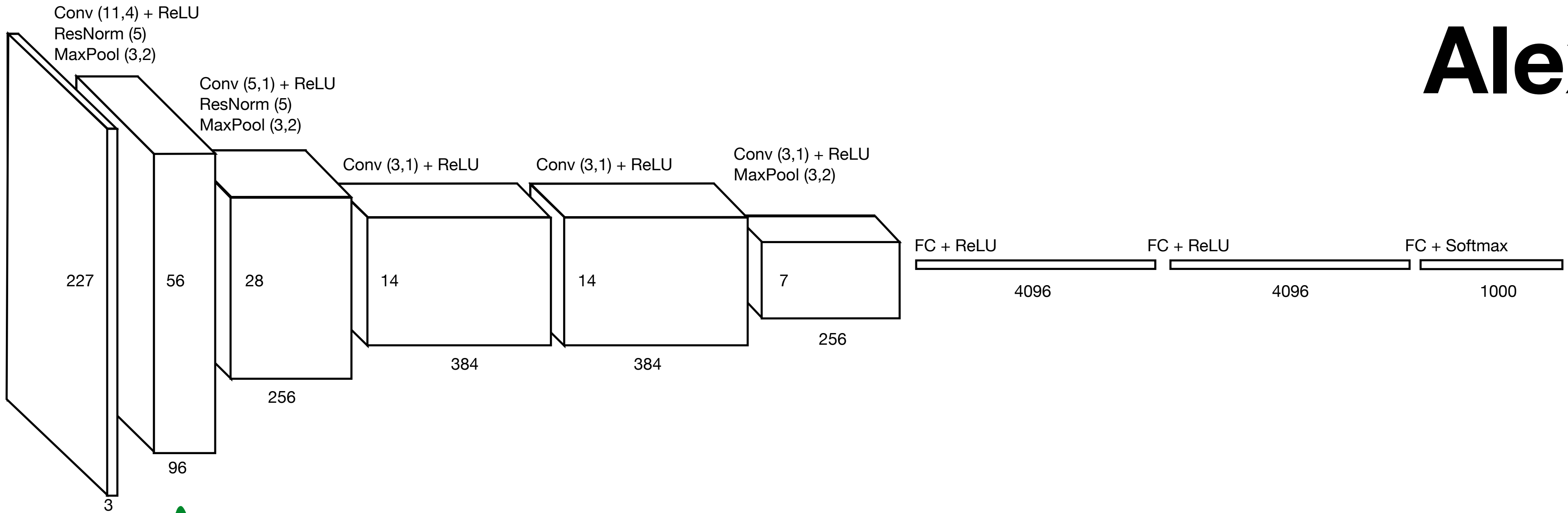


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

AlexNet

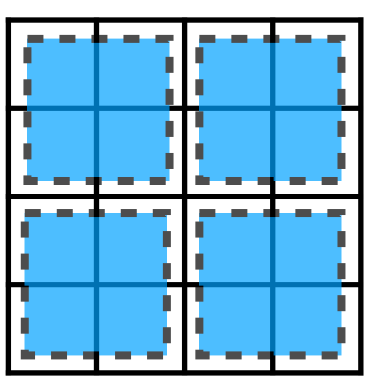


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

Traditional Pooling



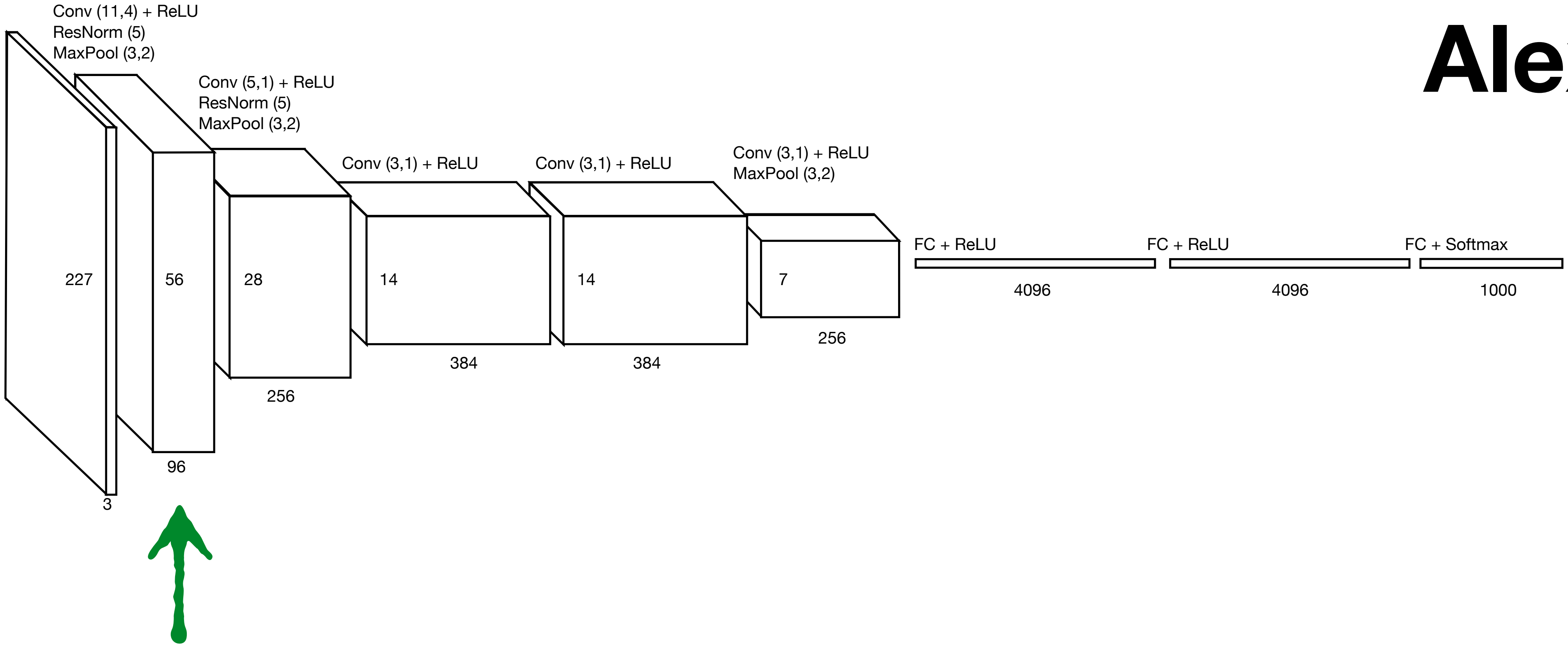
Size: 2 x 2

Stride: 2

‘MaxPool (2,2)’

0.3~0.5% reduction in error rate

AlexNet

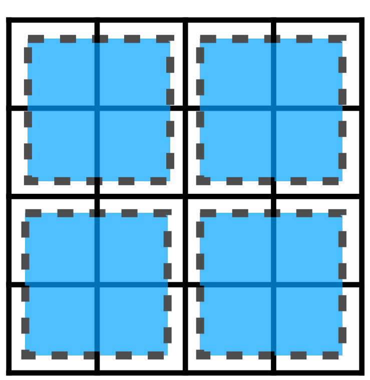


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

Traditional Pooling

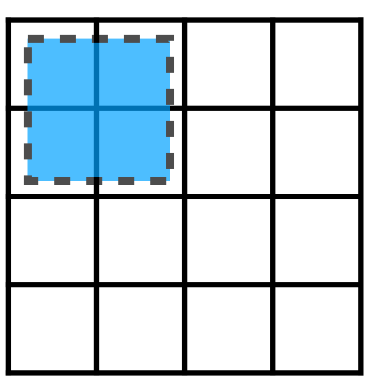


Size: 2 x 2

Stride: 2

'MaxPool (2,2)'

Overlapped Pooling



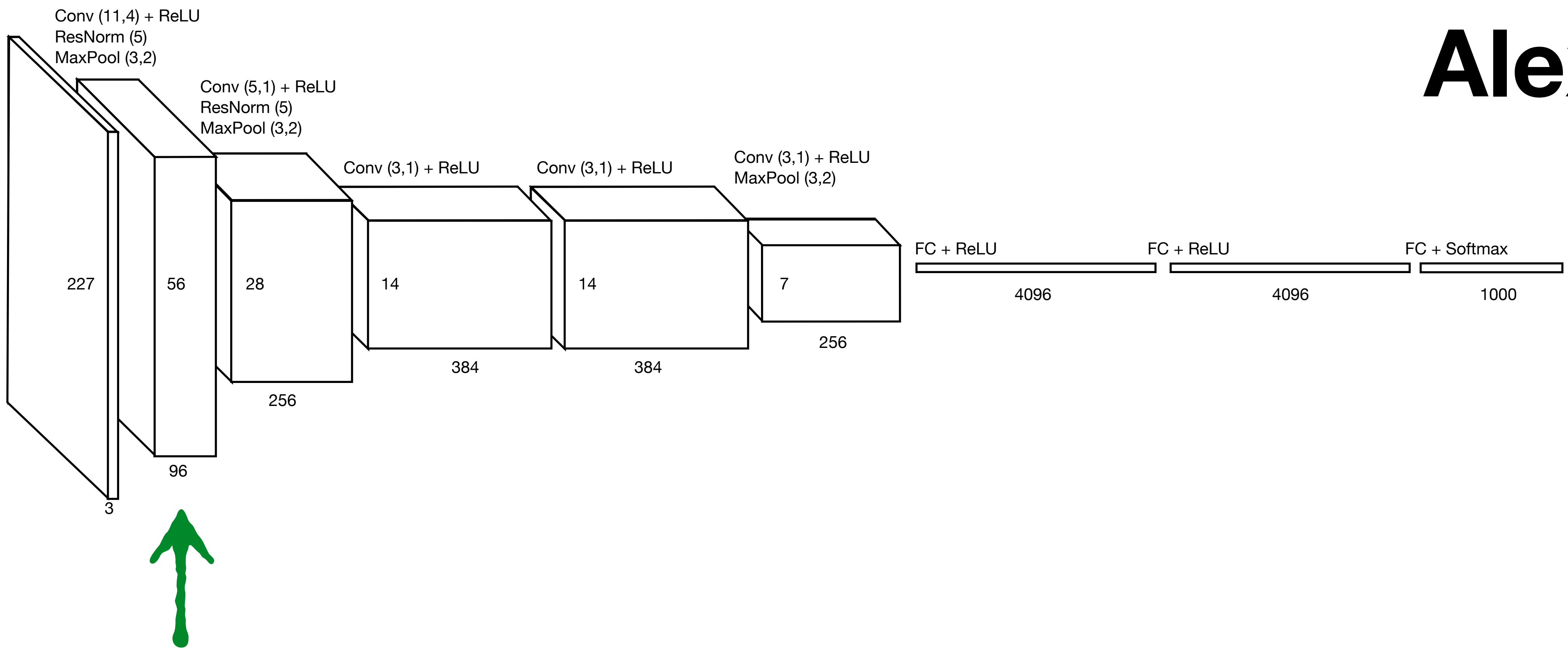
Size: 2 x 2

Stride: 1

'MaxPool (2,1)'

0.3~0.5% reduction in error rate

AlexNet

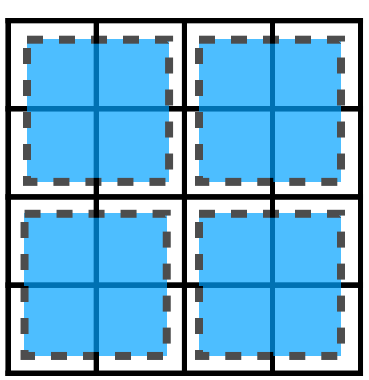


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

Traditional Pooling

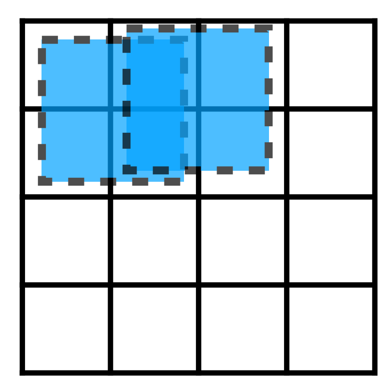


Size: 2 x 2

Stride: 2

'MaxPool (2,2)'

Overlapped Pooling



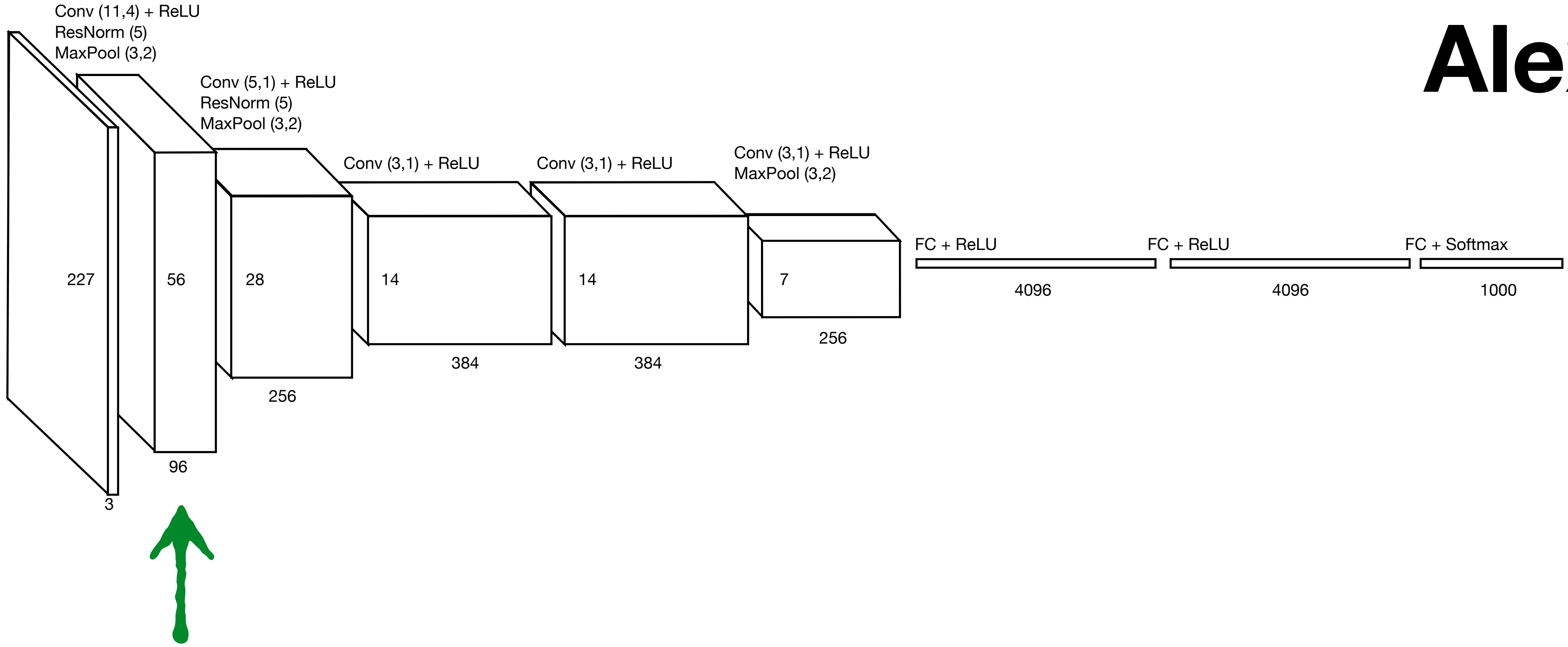
Size: 2 x 2

Stride: 1

'MaxPool (2,1)'

0.3~0.5% reduction in error rate

AlexNet

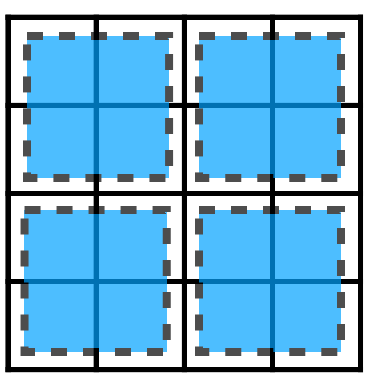


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

Traditional Pooling

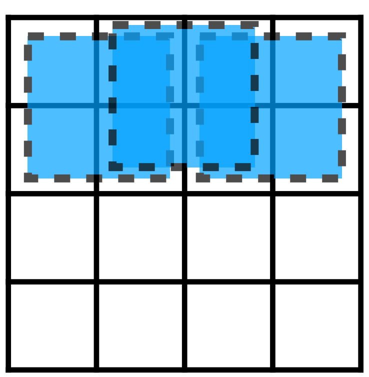


Size: 2 x 2

Stride: 2

'MaxPool (2,2)'

Overlapped Pooling



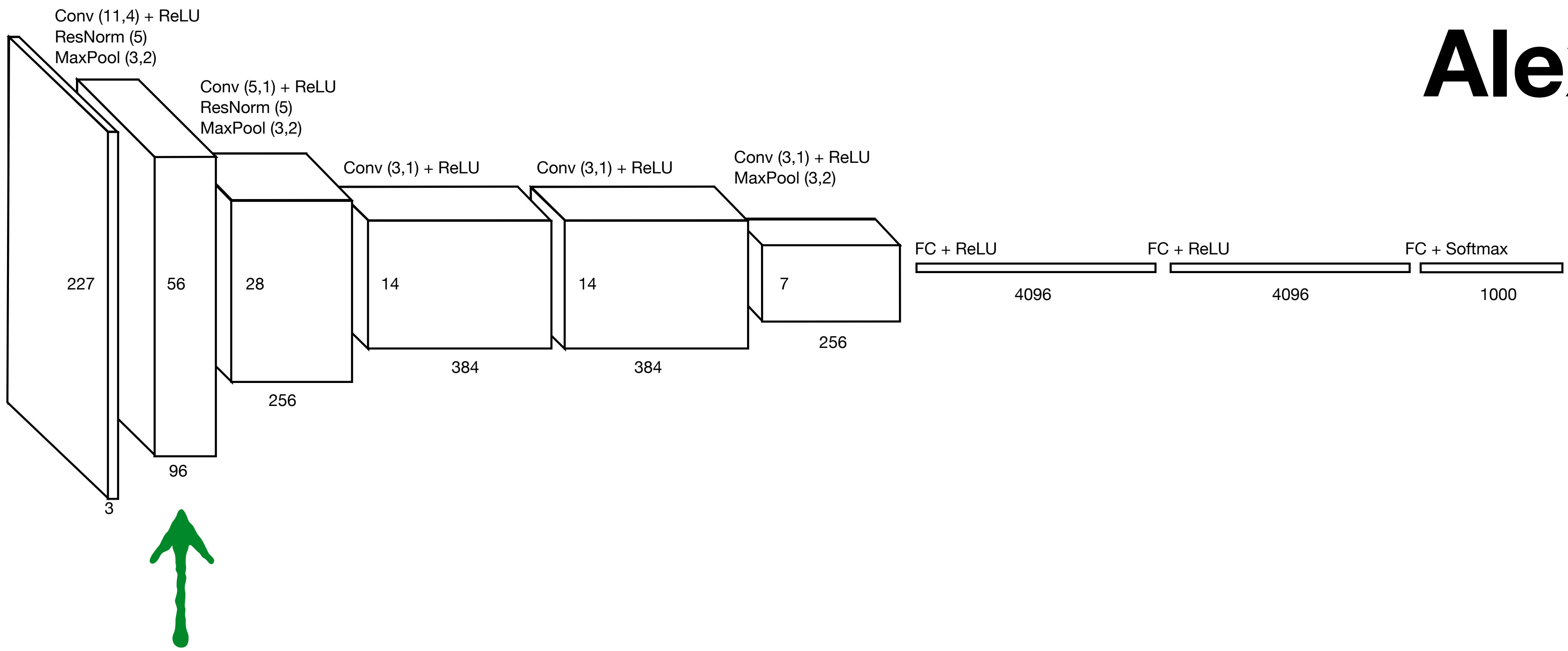
Size: 2 x 2

Stride: 1

'MaxPool (2,1)'

0.3~0.5% reduction in error rate

AlexNet

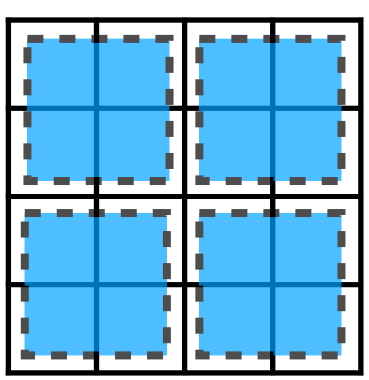


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

Traditional Pooling

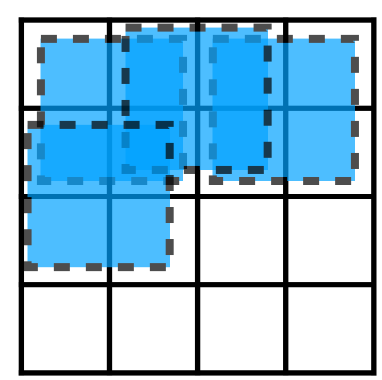


Size: 2 x 2

Stride: 2

'MaxPool (2,2)'

Overlapped Pooling



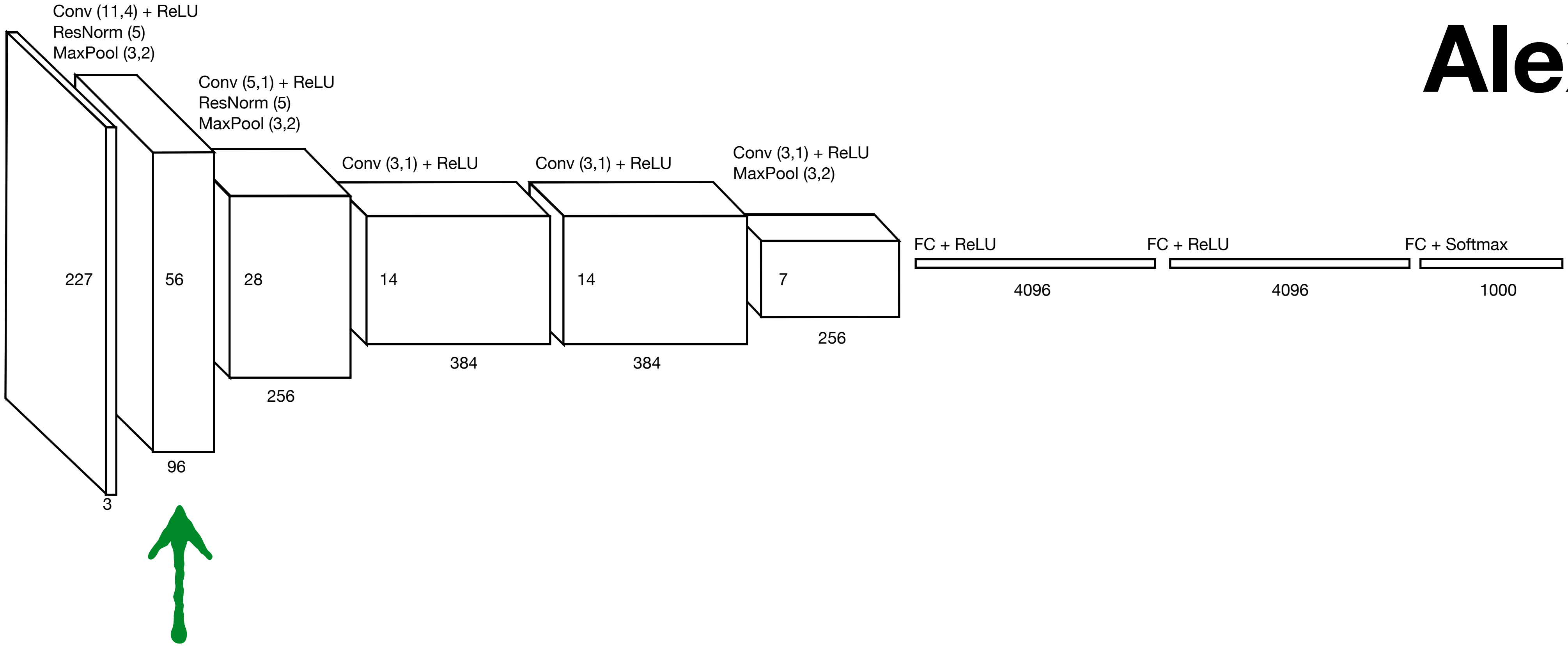
Size: 2 x 2

Stride: 1

'MaxPool (2,1)'

0.3~0.5% reduction in error rate

AlexNet

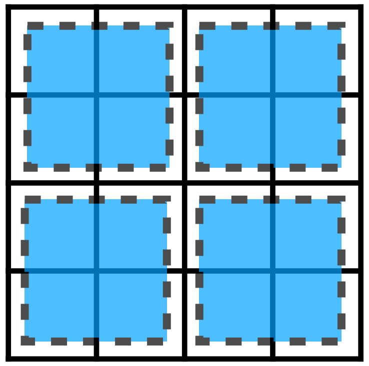


Conv1

(actually a convolution, ReLU, response normalization and **max pooling**)

Overlapping maximum pooling

Traditional Pooling

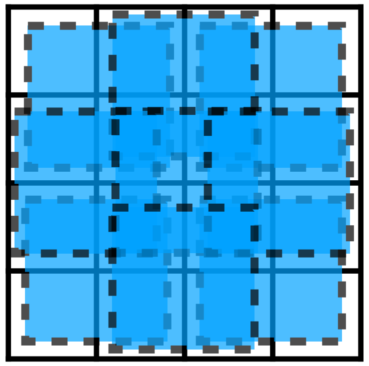


Size: 2 x 2

Stride: 2

'MaxPool (2,2)'

Overlapped Pooling



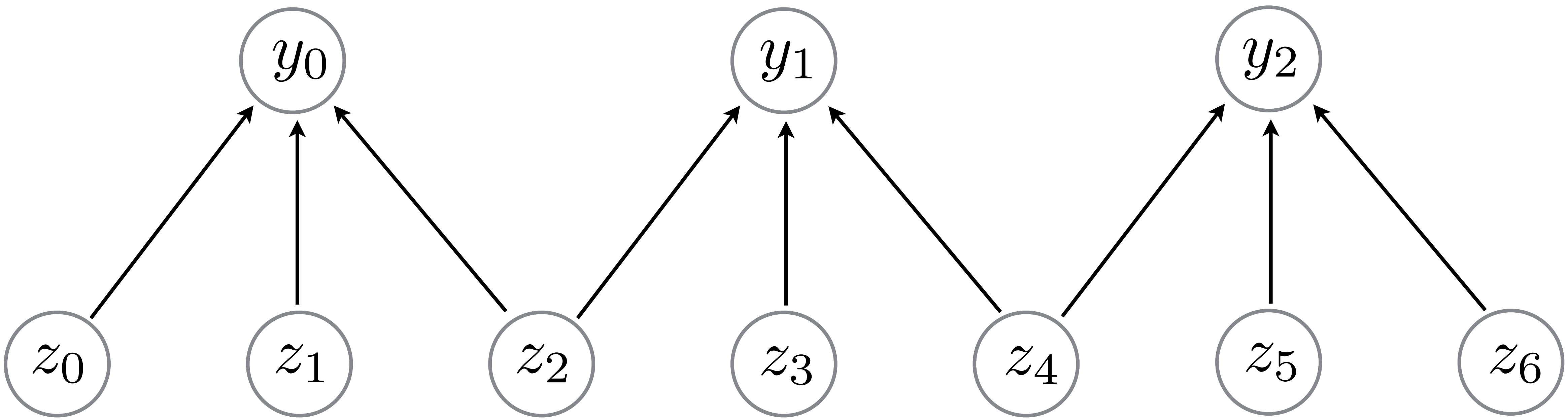
Size: 2 x 2

Stride: 1

'MaxPool (2,1)'

0.3~0.5% reduction in error rate

Pooling reduces the size the layer.
Also builds small translation invariance.



Max-pooling:

$$y_i = \max_{j \in J_i} z_j$$

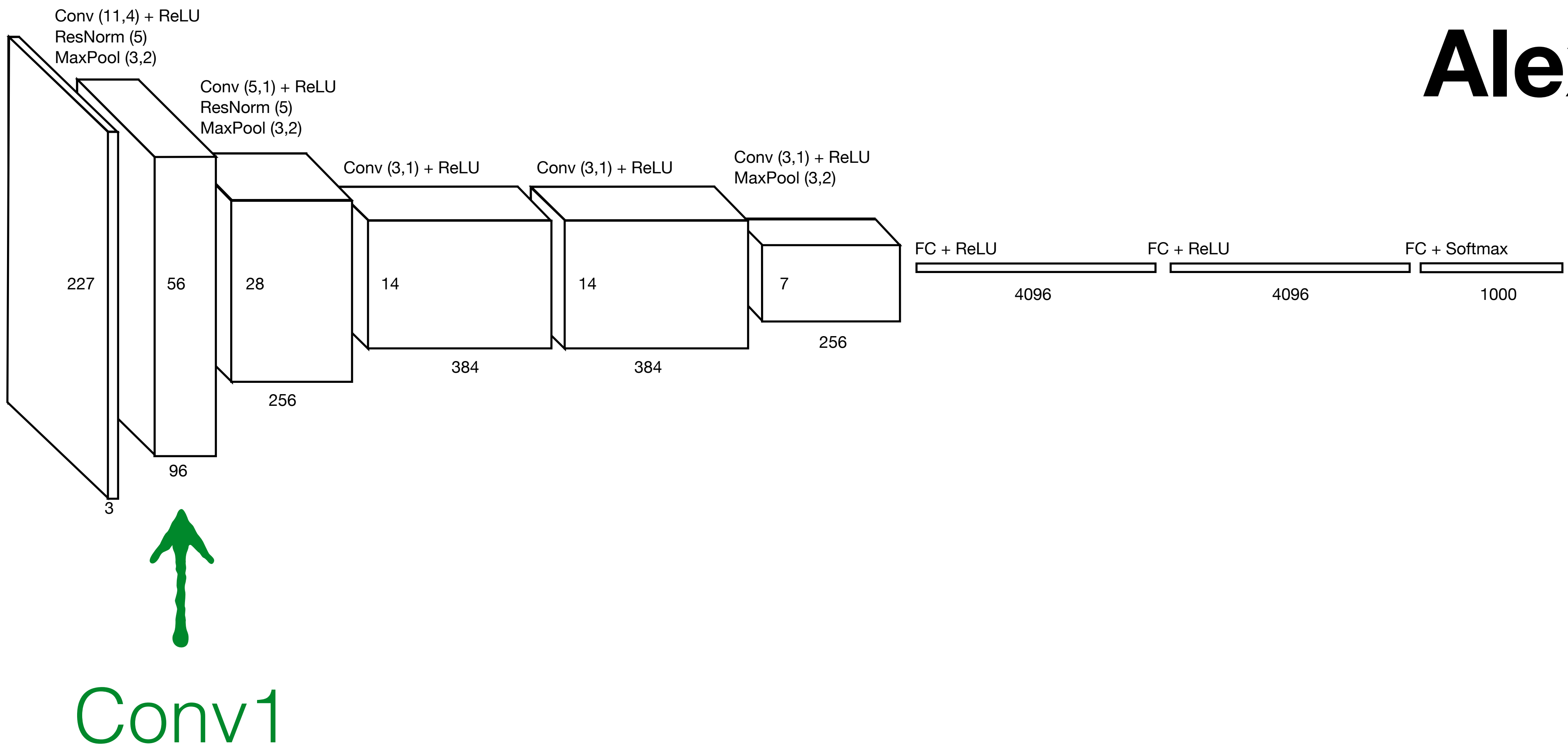
max in the pooling group

Average-pooling:

$$y_i = \mathrm{avg}_{j \in J_i} z_j$$

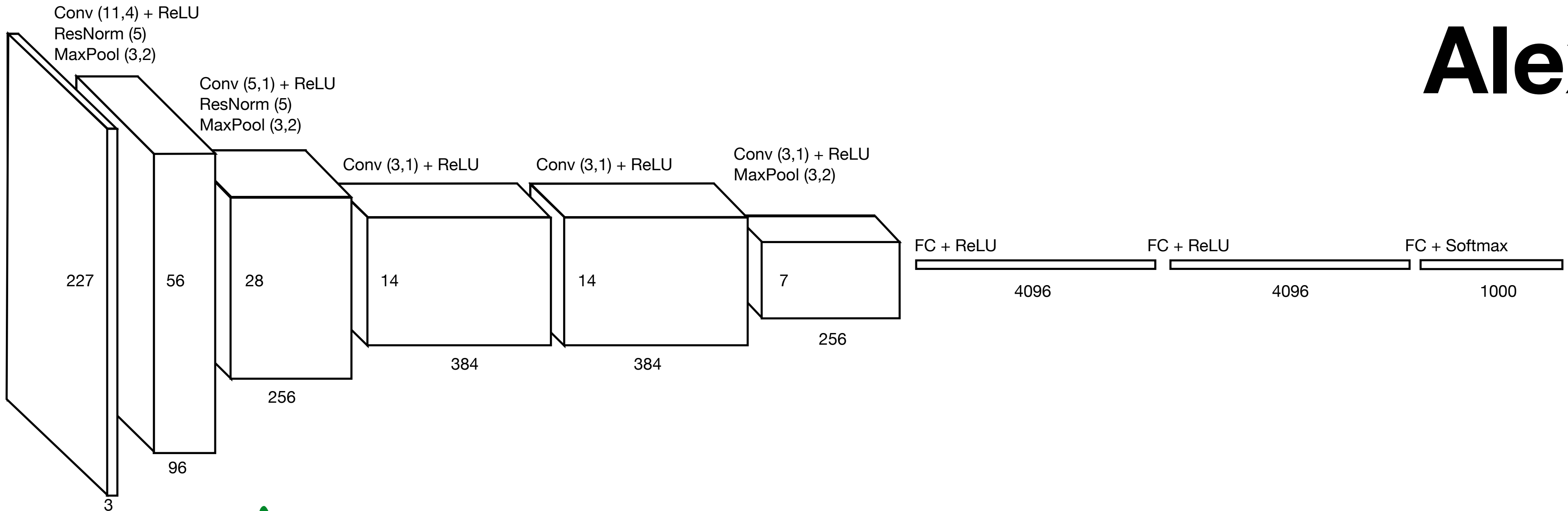
average of the pooling group

AlexNet



Conv (11,4) + ReLU
ResNorm (5)
MaxPool (3,2)

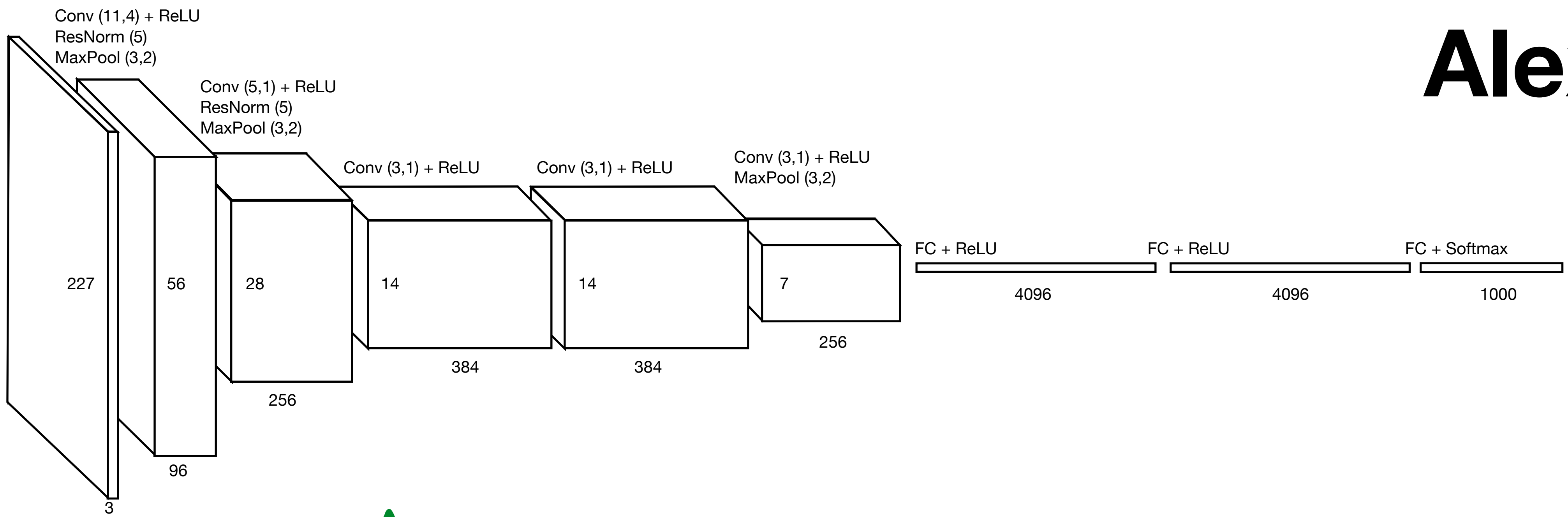
AlexNet



Conv2

Conv (5,1) + ReLU
ResNorm (5)
MaxPool (3,2)

AlexNet

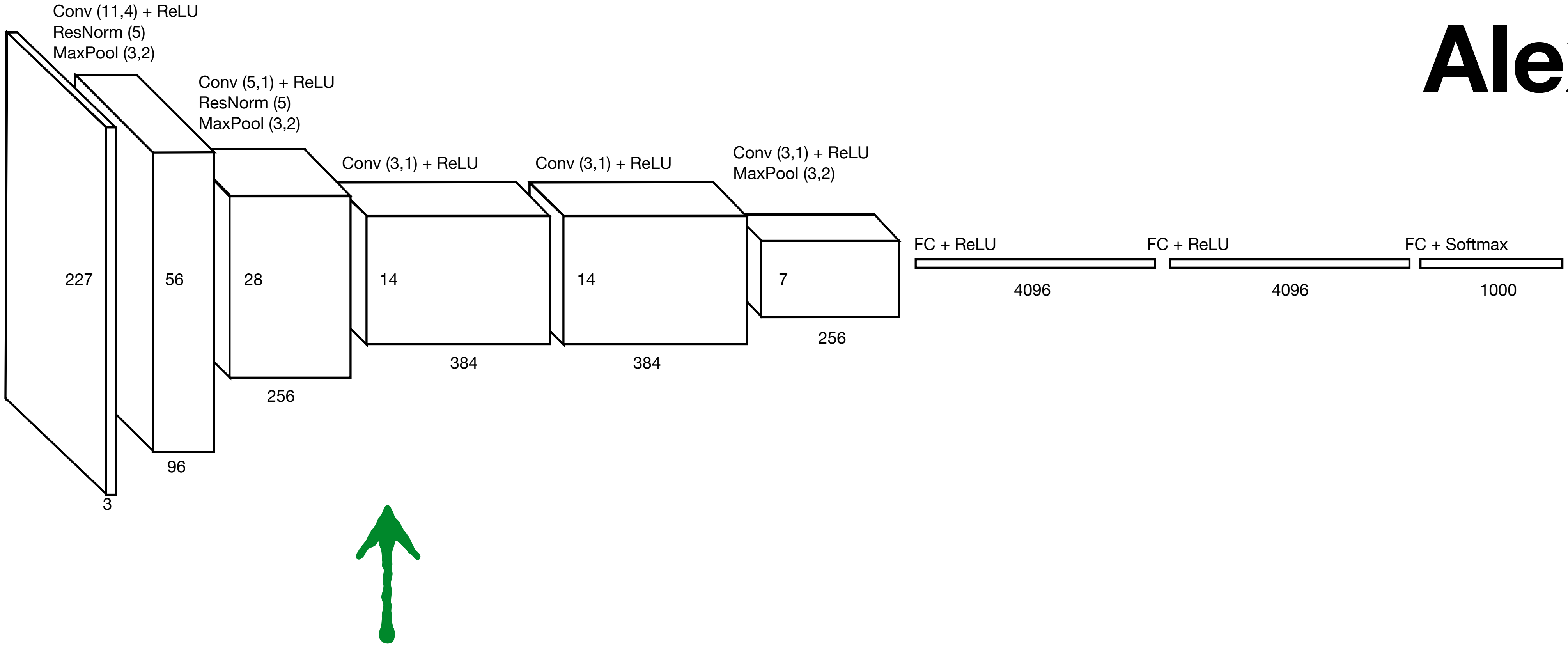


Conv3

Conv (3,1) + ReLU

Why is the response map reduced from 27 to 13 after convolution and ReLU?

AlexNet



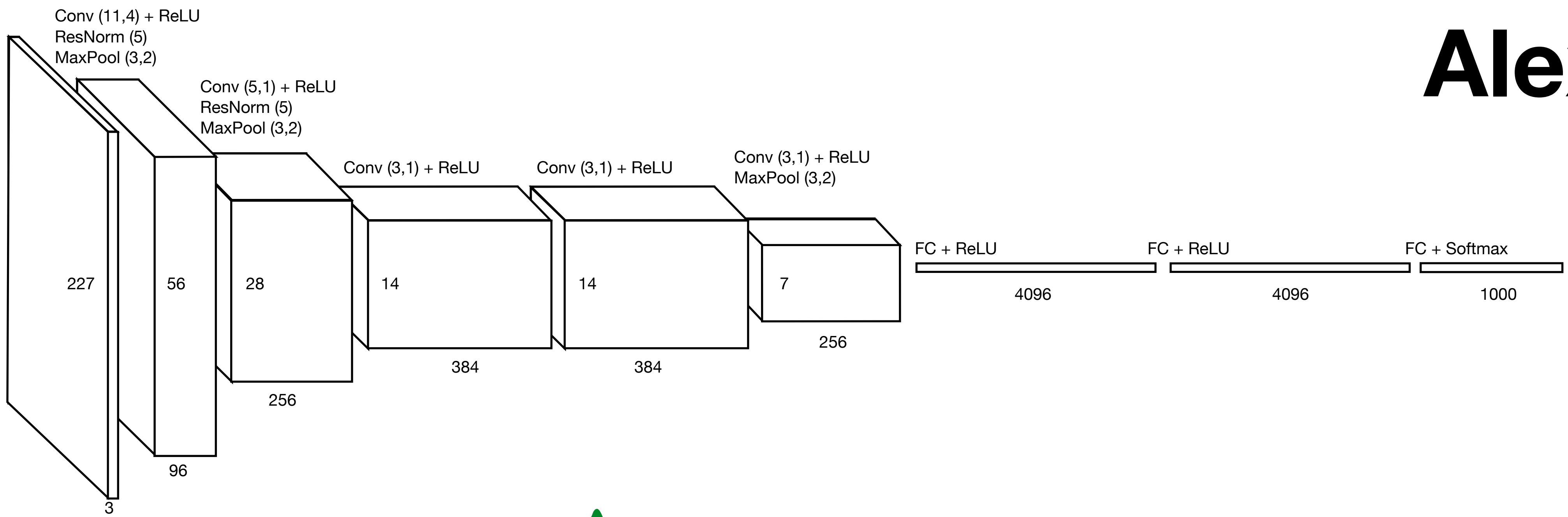
Conv3

Conv (3,1) + ReLU

Why is the response map reduced from 27 to 13 after convolution and ReLU?

The max-pool in conv2 reduced the response size by half.

AlexNet



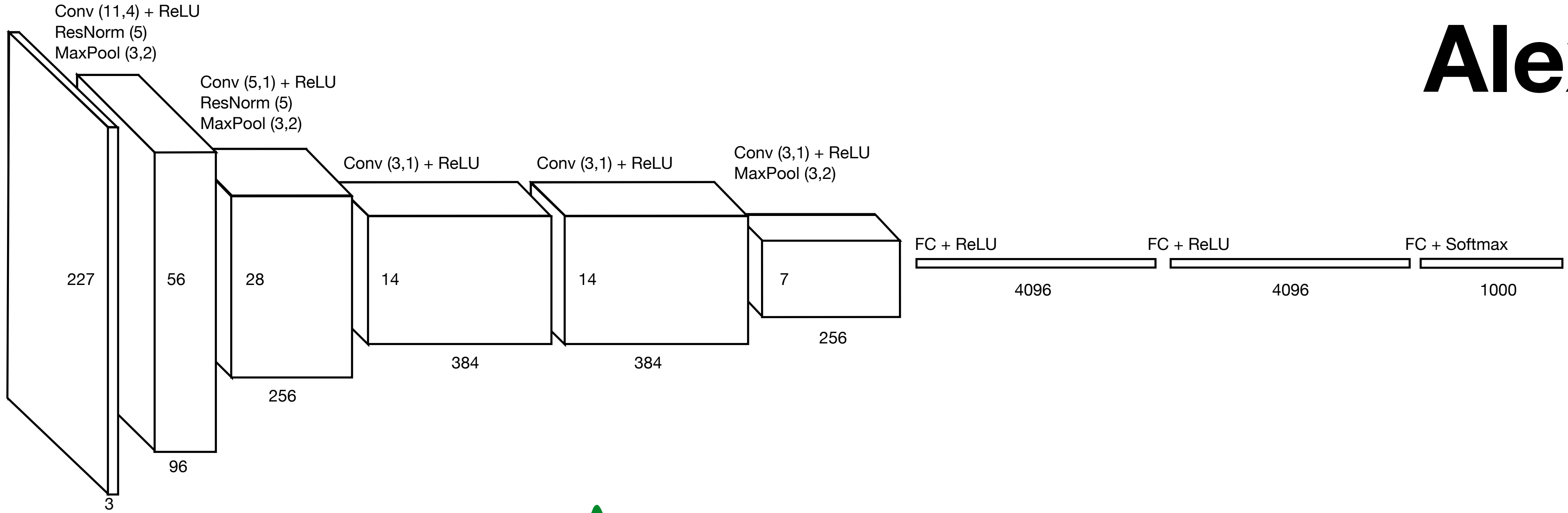
Conv4

Conv (3,1) + ReLU

Why do we want to convolve with the same size twice?

Why not convolve with a 5 x 5 filter?

AlexNet



Conv4

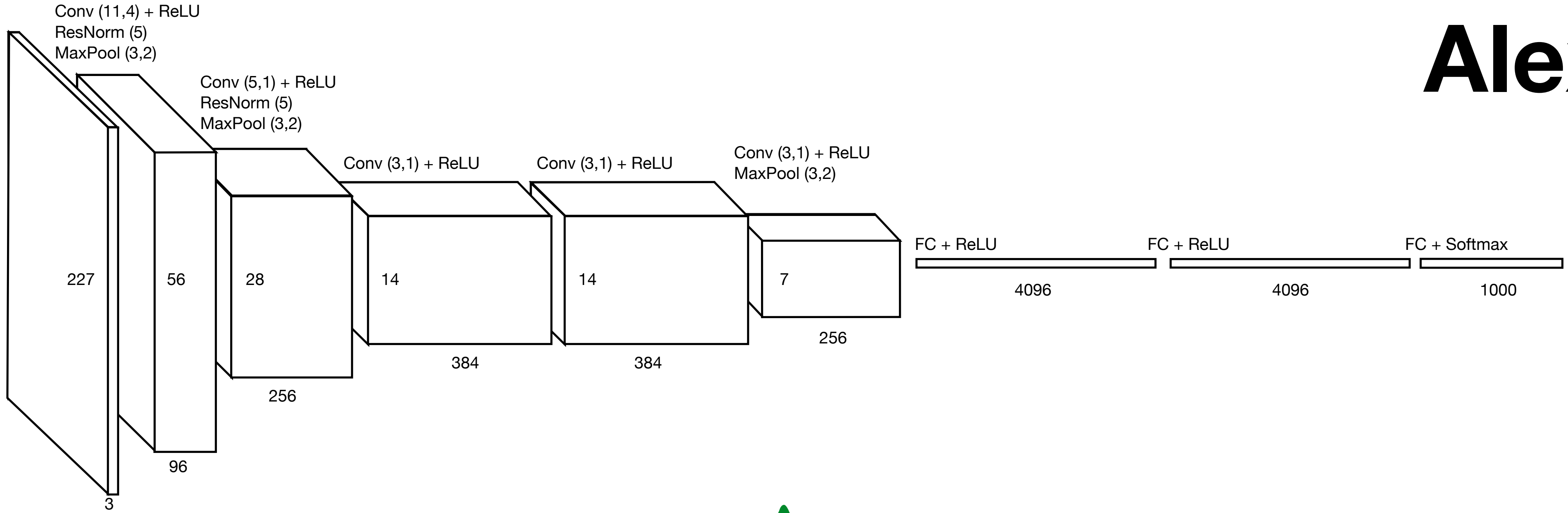
Conv (3,1) + ReLU

Why do we want to convolve with the same size twice?

Why not convolve with a 5 x 5 filter?

Learns a non-linear combination of non-linear lower level features

AlexNet

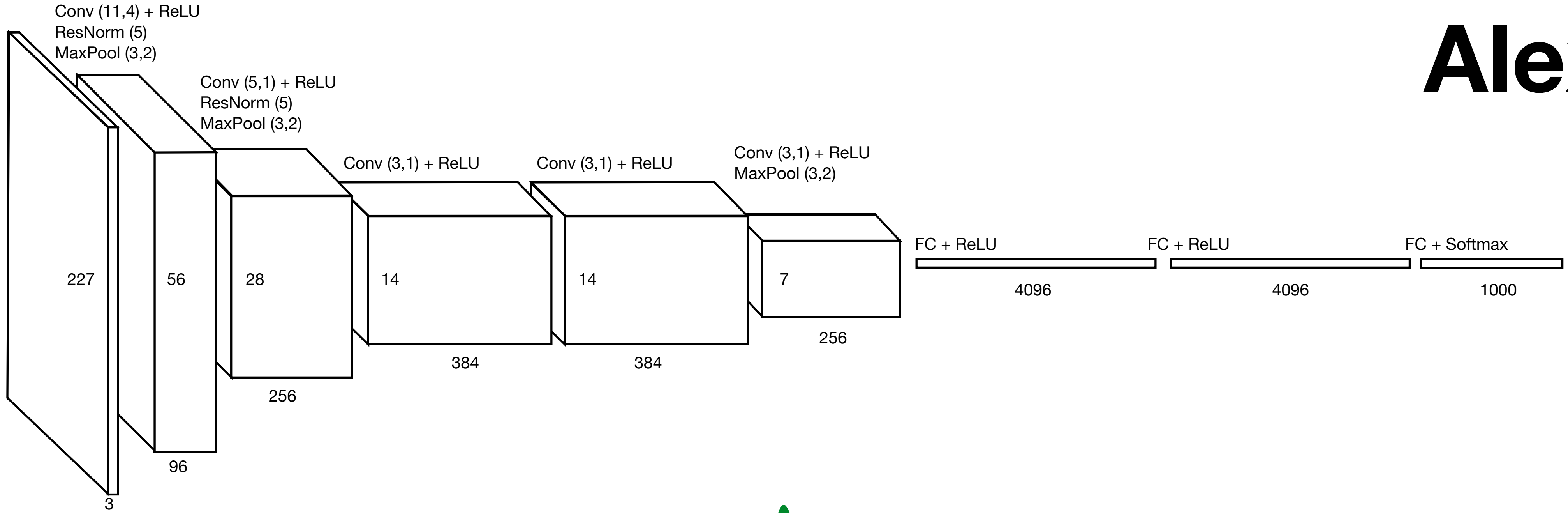


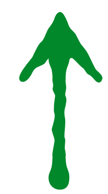
Conv5

Conv (3,1) + ReLU
MaxPool (3,2)

What is the size of the response map after conv5?

AlexNet





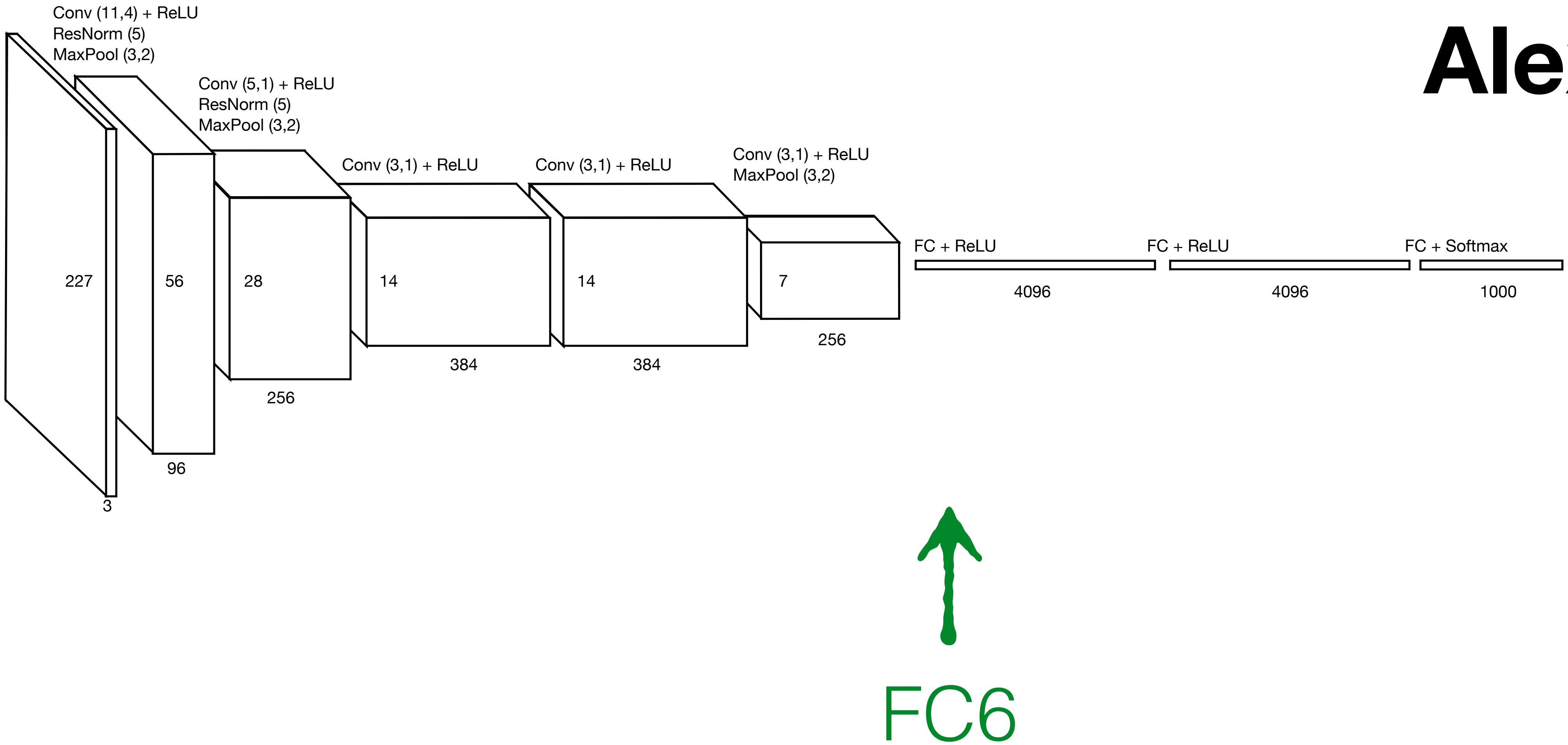
Conv5

Conv (3,1) + ReLU
MaxPool (3,2)

What is the size of the response map after conv5?

7 x 7 x 256

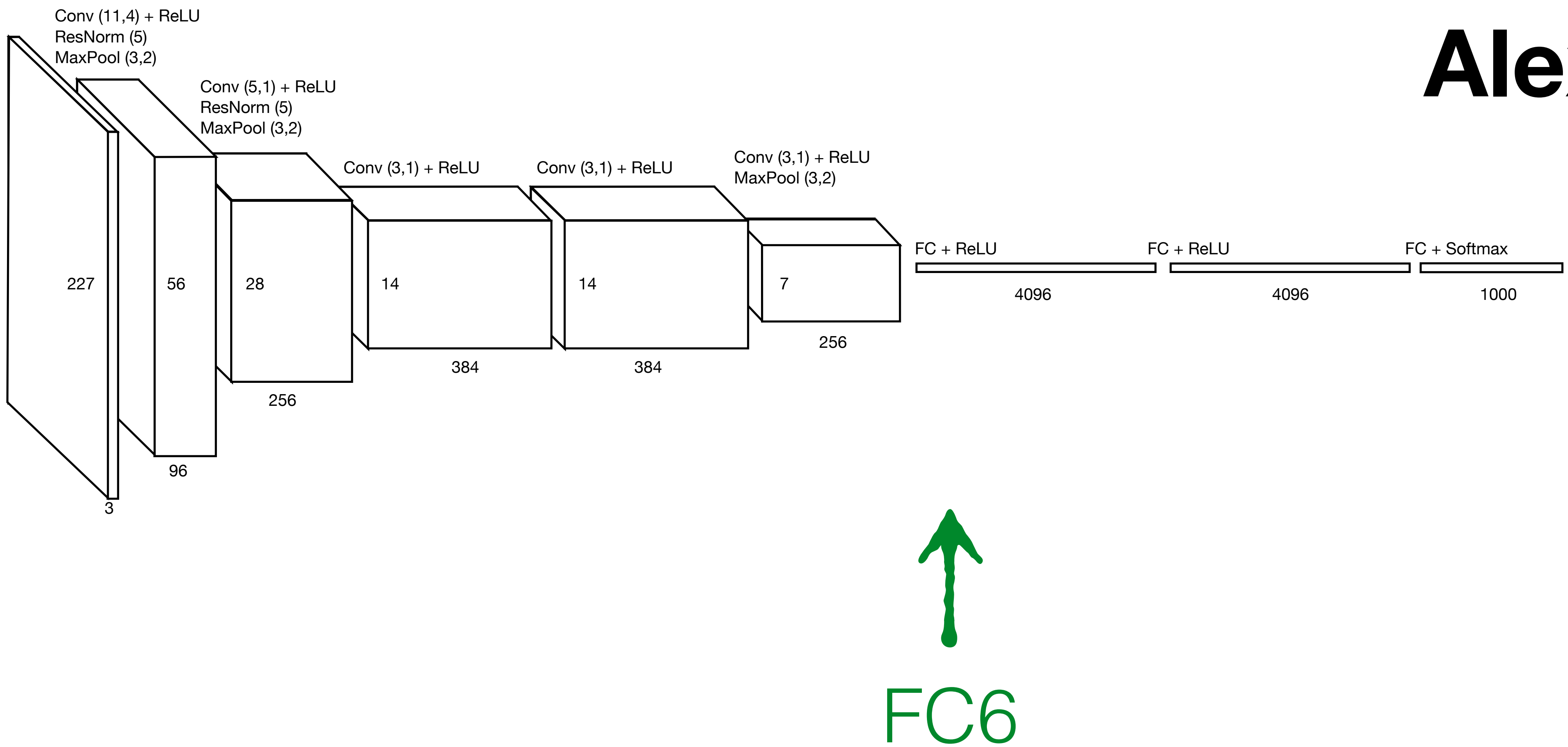
AlexNet



Fully Connected Layer (FC)
ReLU

How many connections in the fully connect layer?

AlexNet



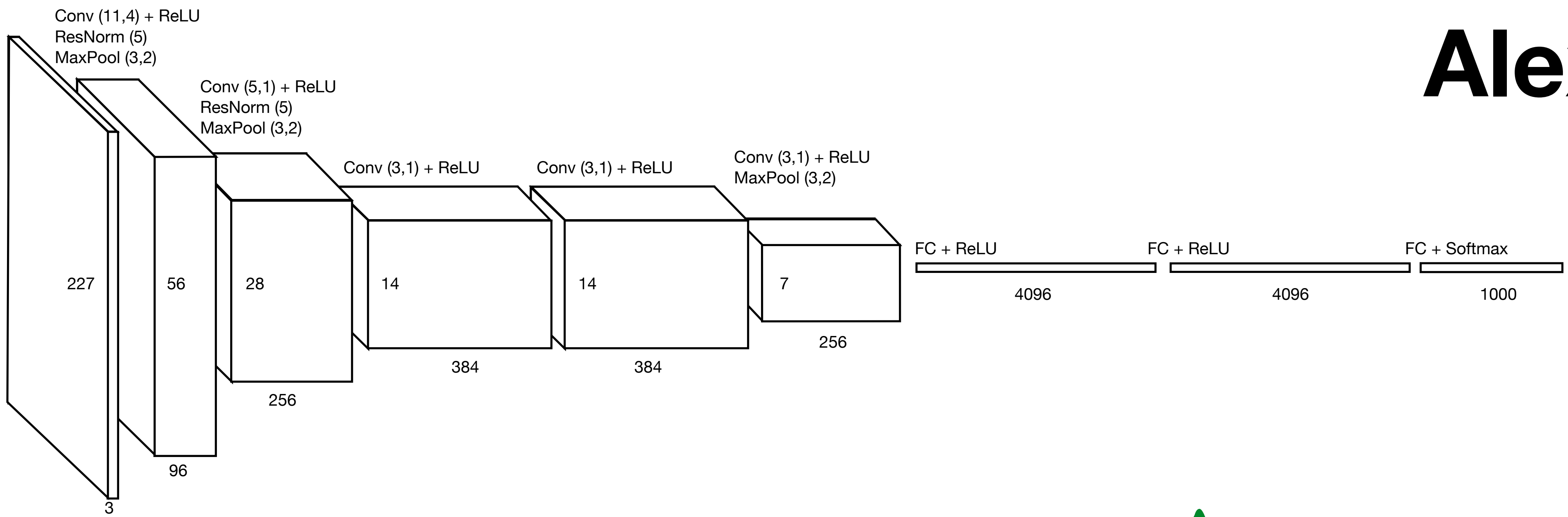
Fully Connected Layer (FC)
ReLU

How many connections in the fully connect layer?

$$(7 \times 7 \times 256 + 1) \times 4096$$

bias term

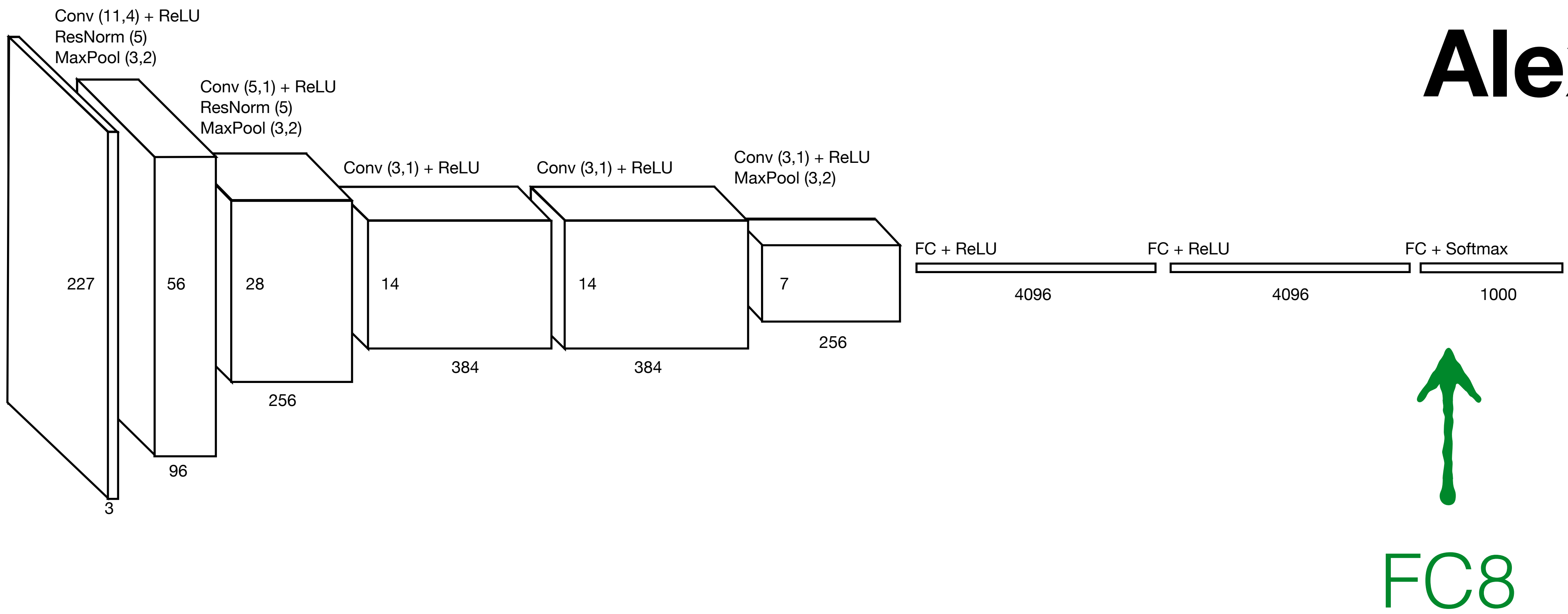
AlexNet



FC7

Fully Connected Layer (FC)
ReLU

AlexNet



Fully Connected Layer (FC)
Softmax

Softmax Function

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Final softmax converts FC output values to probabilities

Multinomial Logistic Regression

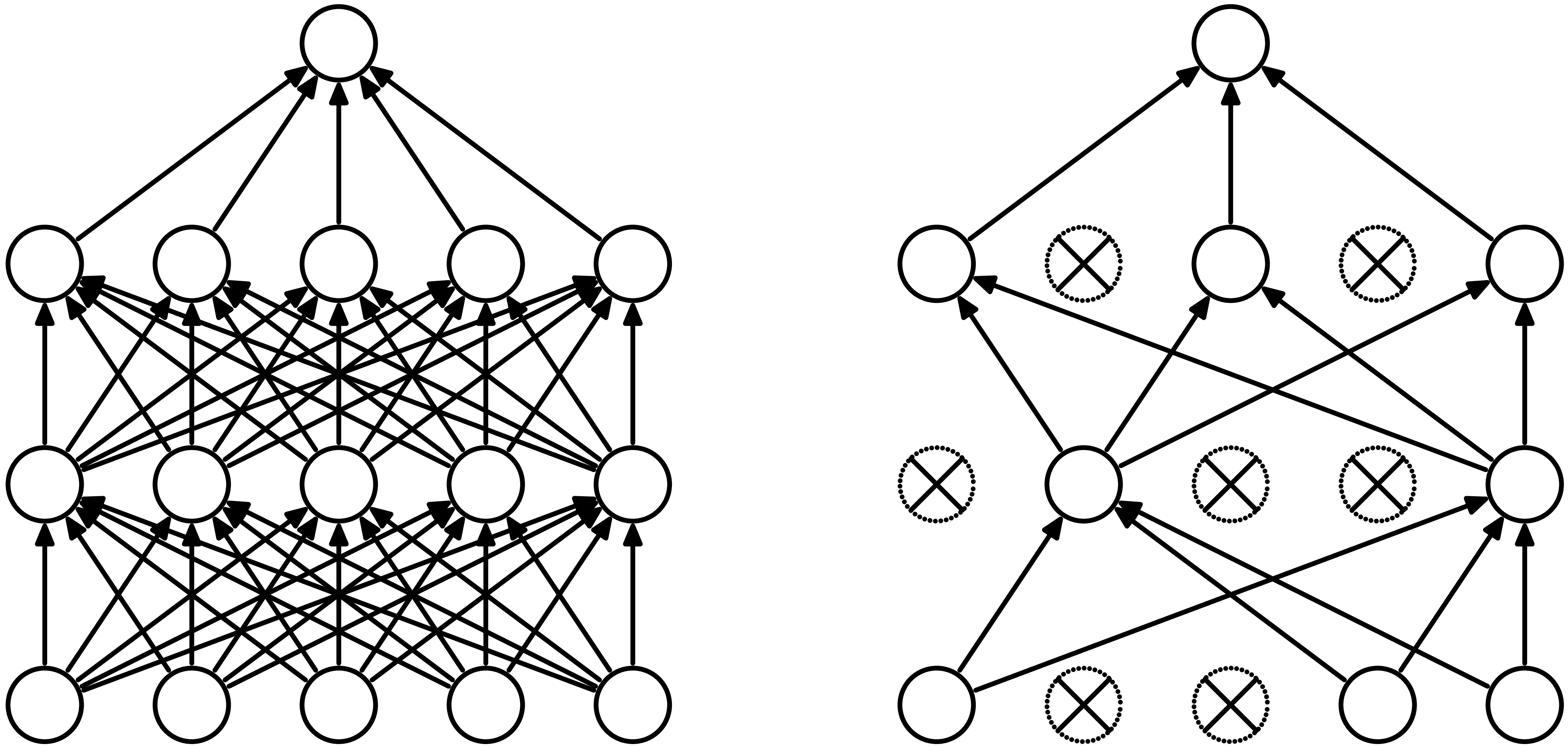
$$p(y_i|\boldsymbol{x}) = \frac{e^{\boldsymbol{x}^\top \boldsymbol{w}_i}}{\sum_j e^{\boldsymbol{x}^\top \boldsymbol{w}_j}}$$

FC + softmax is just (multi-nomial) logistic regression

Tricks for Training AlexNet

- Dropout
- Data augmentation
- SGD with momentum

DropOut



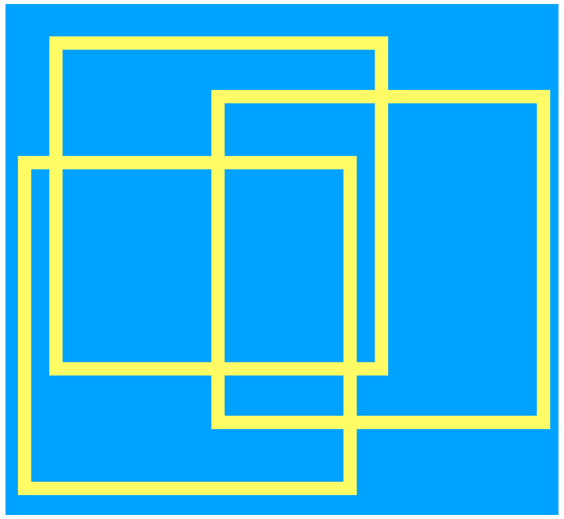
(a) Standard Neural Net

(b) After applying dropout.

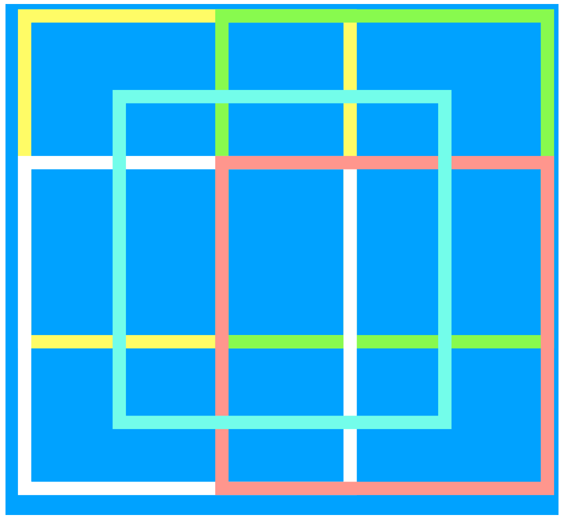
- During training certain nodes are randomly dropped out in the FC
- Forces the network to learn with less parameters
- Reduces overfitting (doubles training time)
- Has the effect of model averaging
- Use all nodes at test time

Data Augmentation

1. Random translation and
horizontal flipping
(2048x increase in data)



A test time, prediction is
averaged over 10 patches
(corners, centers and flips)



Prevents overfitting

Data Augmentation

2. Color augmentation

PCA over a bunch of pixels in training set

$$\begin{bmatrix} I_{xy}^R \\ I_{xy}^G \\ I_{xy}^B \end{bmatrix} = \underbrace{\overbrace{[\boldsymbol{e}_1 \ \ \boldsymbol{e}_2 \ \ \boldsymbol{e}_3]}^{\text{eigenvectors}}}_{\text{average color}} \begin{bmatrix} \alpha_1\lambda_1 \\ \alpha_2\lambda_2 \\ \alpha_3\lambda_3 \end{bmatrix}^\top$$

eigenvalues

multiplier

$$\alpha_i \sim \mathcal{N}(0, \sigma = 0.1)$$ sampled once per training image

Changes the color and intensity of the image

Reduces top-1 error rate by 1%

Gradient Update

expected gradient
of the loss

momentum

weight decay

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

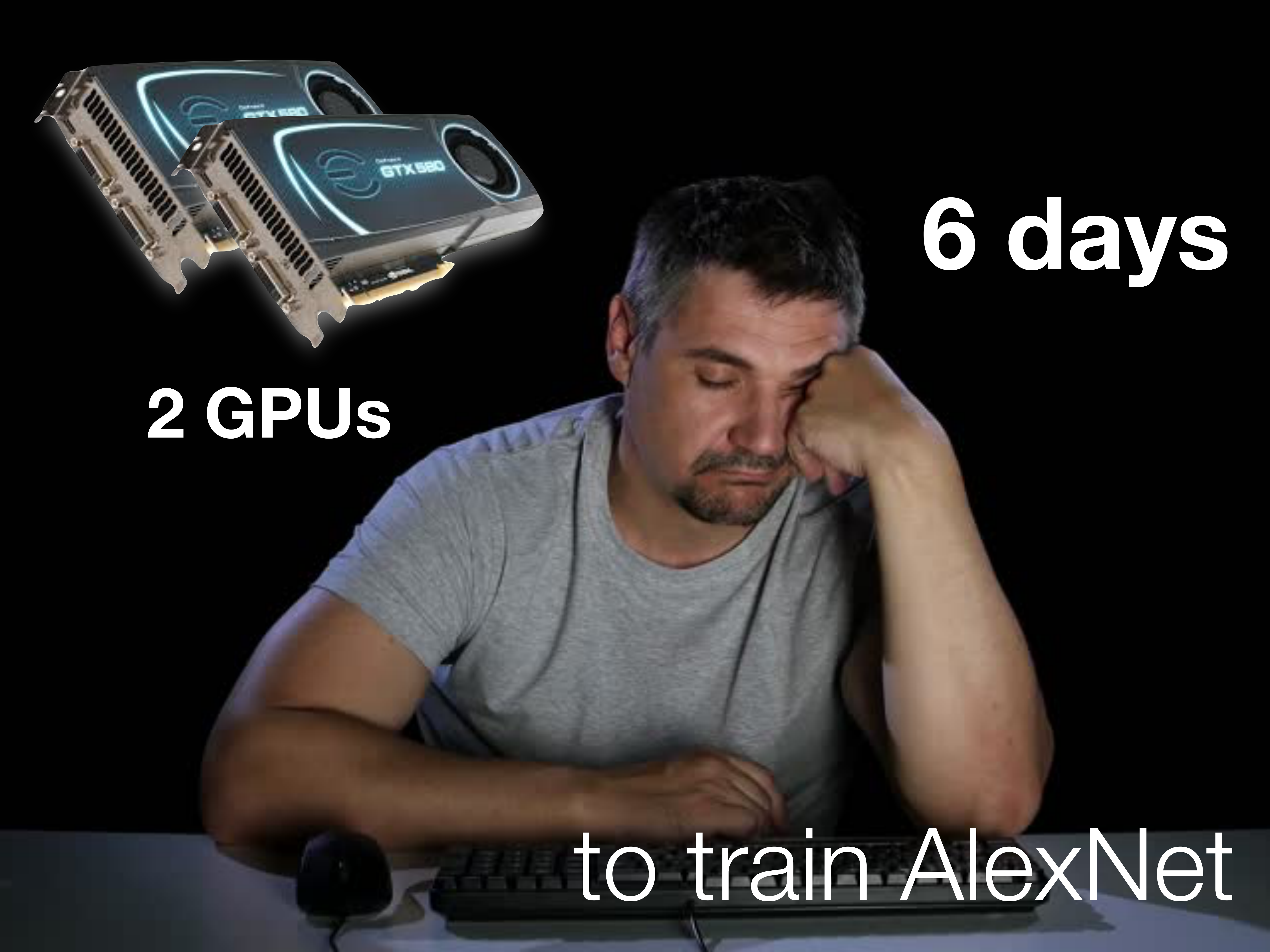
'keep going'

learning rate

mini-batch

update equation

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


2 GPUs
6 days
to train AlexNet

Results

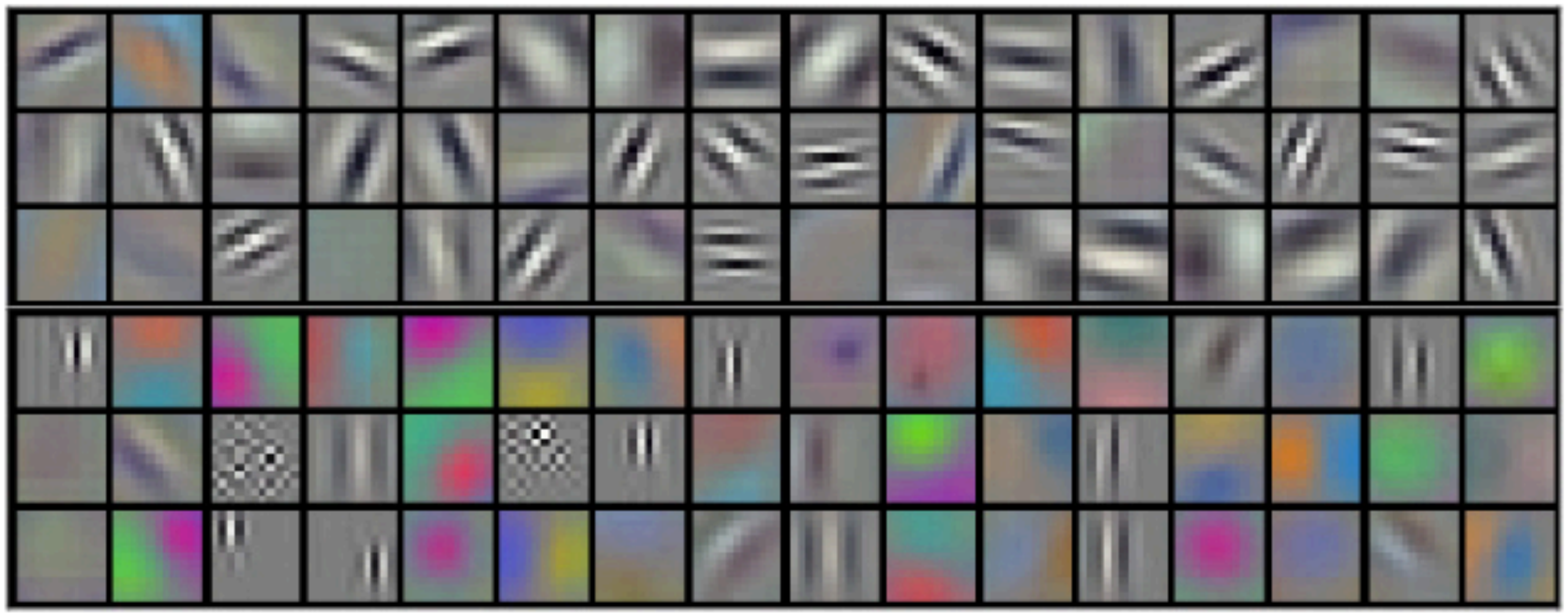


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

ILSVRC 2010

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

ILSVRC 2012

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%**

huge win!

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.

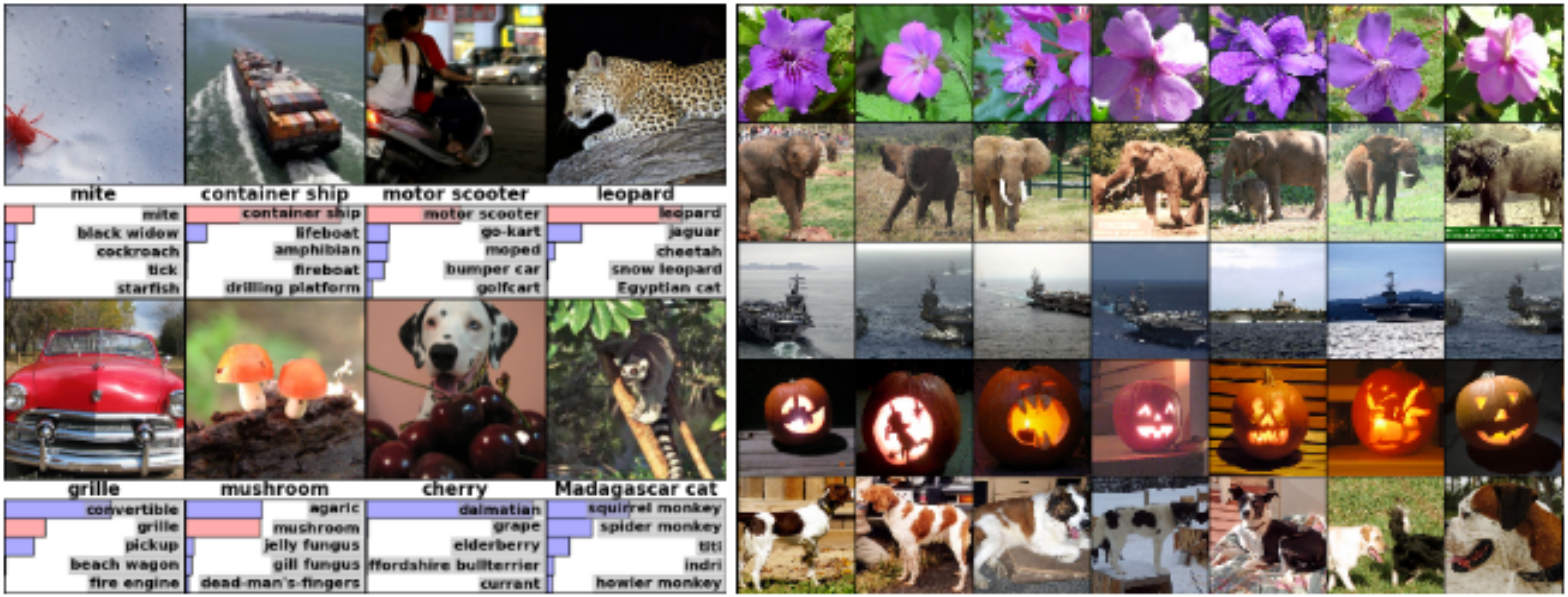


Figure 4: (**Left**) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (**Right**) Five ILSVRC-2010 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.

Important Concepts

- Overlapped Pooling
- Response Normalization
- ReLU
- Softmax
- Training trick: Dropout
- Training trick: Data Augmentation
- Training trick: momentum and decay