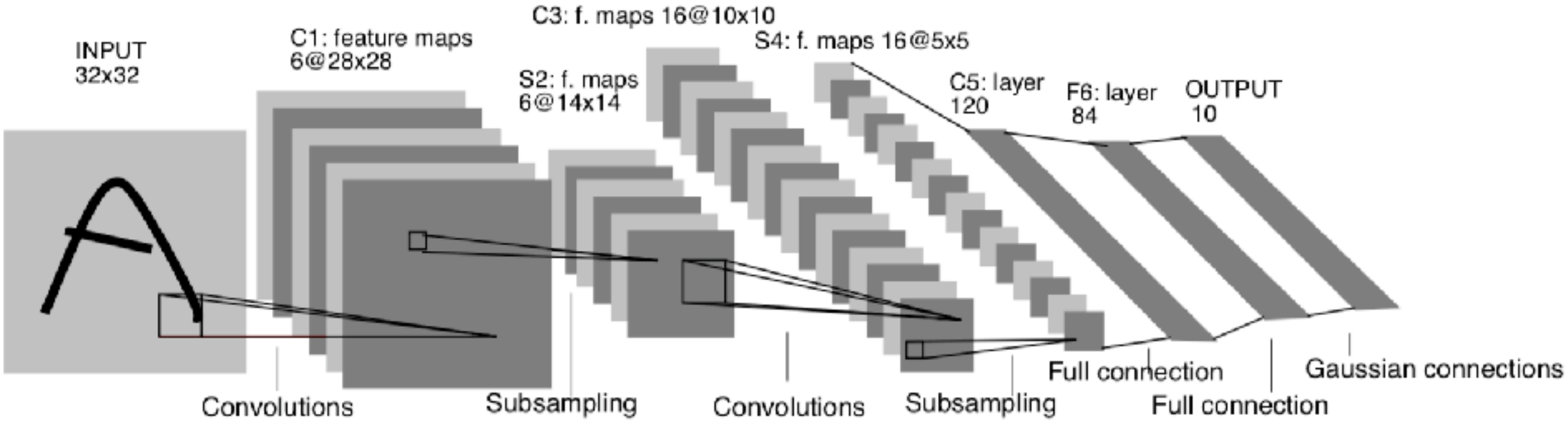



# LeNet

## Computer Vision

**Carnegie Mellon University (Kris Kitani)**

**Recall:**

‘Classical’

# Image Classification Pipeline

**Input** → **encoder** → **decoder** → **output**

image(s)

Extract features

Build descriptor

Classifier

semantic label

**Recall:**

‘Classical’

# Image Classification Pipeline

**Input** → **encoder** → **decoder** → **output**

image(s)

Extract features

Build descriptor

e.g., SIFT, Bag of visual words

Classifier

e.g., KNN, SVM, Naive Bayes

semantic label

**Recall:**

‘Classical’

# Image Classification Pipeline

**Input** → **encoder** → **decoder** → **output**

image(s)

Extract features

Build descriptor

e.g., SIFT, Bag of visual words

Classifier

e.g., KNN, SVM, Naive Bayes

Learned from the data

semantic label

**Recall:**

‘Classical’

# Image Classification Pipeline

**Input** → **encoder** → **decoder** → **output**

image(s)

Extract features

Build descriptor

e.g., SIFT, Bag of visual words

Designed by you

Painful.

Classifier

e.g., KNN, SVM, Naive Bayes

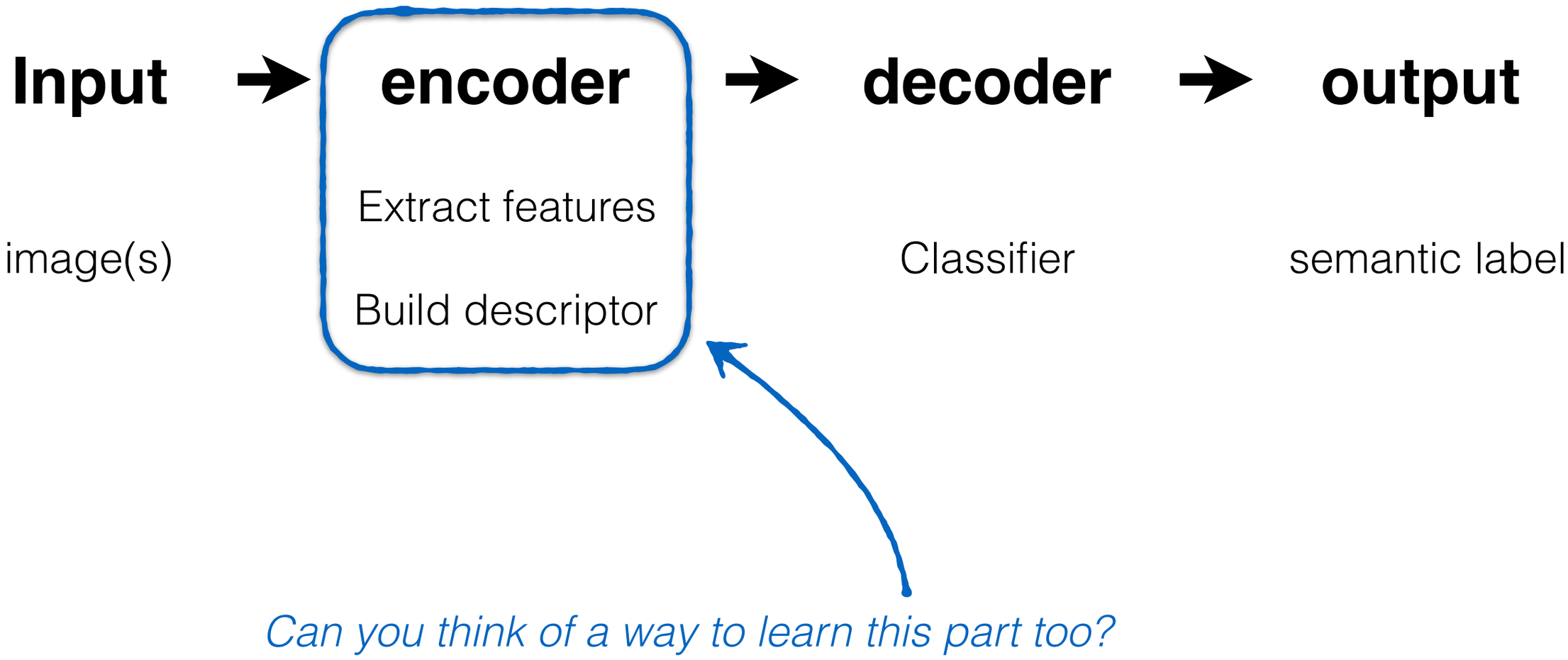
Learned from the data

semantic label

**Recall:**

‘Classical’

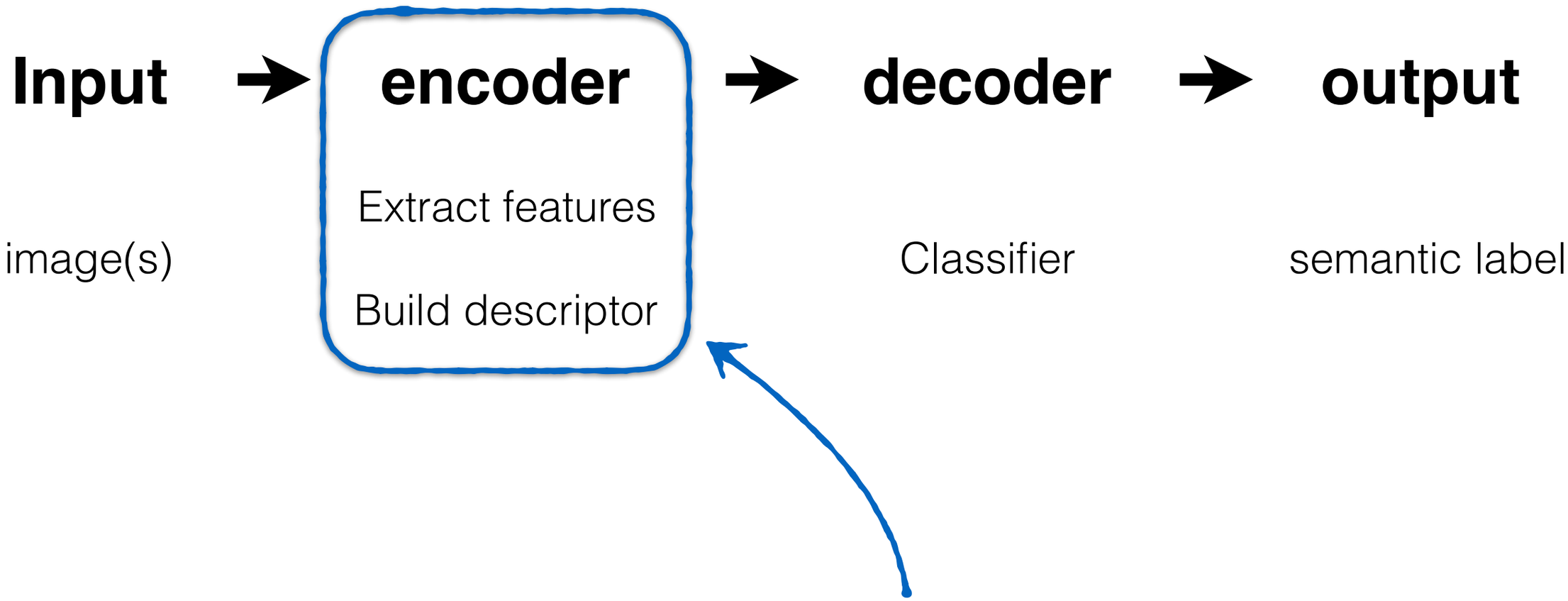
# Image Classification Pipeline



**Recall:**

‘Classical’

# Image Classification Pipeline




*How can we learn it all ‘**end to end**?’*

'modern'

~~'Classical'~~ (...revival of a classical...)

# Image Classification Pipeline

**Input** ➔ **encoder** ➔ **decoder** ➔ **output**

image(s)

~~Extract features~~

~~Build descriptor~~

Many convolutional neuron layers

~~Classifier~~

Some fully connected layers

semantic label

# 'Deep Network'

Use gradient descent to learn everything 'end to end'

(except for the architecture, which you have to design. painful.)

Before you were born…

Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communication*, pages 41–46, November 1989. invited paper.

# Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning

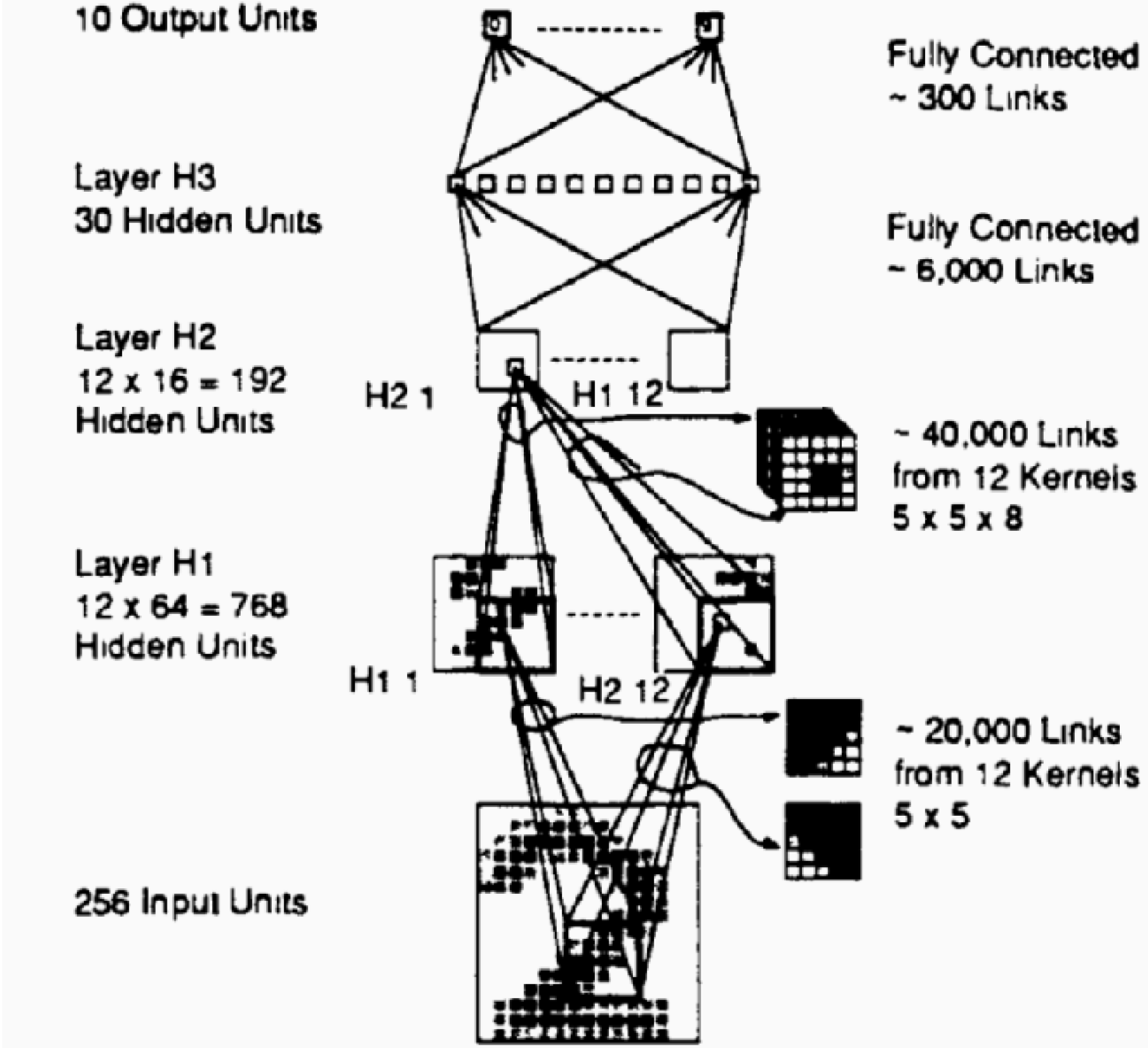
*Y. Le Cun*
*L. D. Jackel*
*B. Boser*
*J. S. Denker*
*H. P Graf*
*I. Guyon*
*D. Henderson*
*R. E. Howard*
*W. Hubbard*

THIS ARTICLE DESCRIBES TWO NEW METHODS for achieving handwritten digit recognition. The task of handwritten digit recognition was chosen for investigation not only because it has considerable practical interest, but because it is relatively well-defined and is sufficiently complex to constitute a meaningful test of connectionist methods.

Simple classification techniques applied to pixel images do not provide high recognition rates because systems based on these techniques contain little prior knowledge about the topology of the task. Knowledge can be built into the system by changing the representation of a digit from a pixel image to a predefined feature description. The first of our methods implements this idea by performing feature extraction with a neural network chip The feature representation can then be used by a relatively simple classifier, consisting of a two-layer network

is highly test-set dependent. A system may successfully recognize 99% of test data consisting of well-formed digits but score only 80% when confronted with the poorly-formed digits that are both routinely produced and easily recognized by people. We choose to perform our experiments on a rather difficult data set: isolated handwritten digits that were taken from postal zip codes. The zip code images were collected by the U.S. Postal Service from envelopes that passed through the Buffalo, NY Post Office. A postal service contractor converted the original zip code images to binary images, and segmented the digits; that is, disaggregated them into five disjointed images, one for each digit of the zip code. The resulting database consists of 9,298 binary images of isolated digits, 7,291 of which are used as the training set, while the remaining 2,007 are used as the test set. Most of the images are fairly clean; however, a significant fraction are very blotchy or incomplete. The latter defect

**1989!**




This led to a famous model known as…

# Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

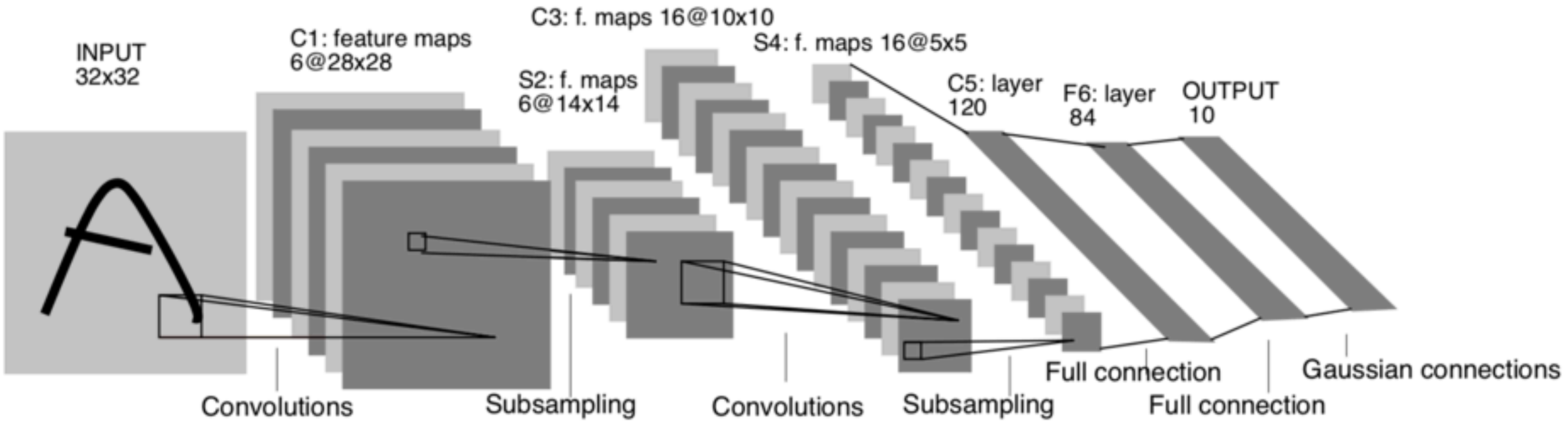


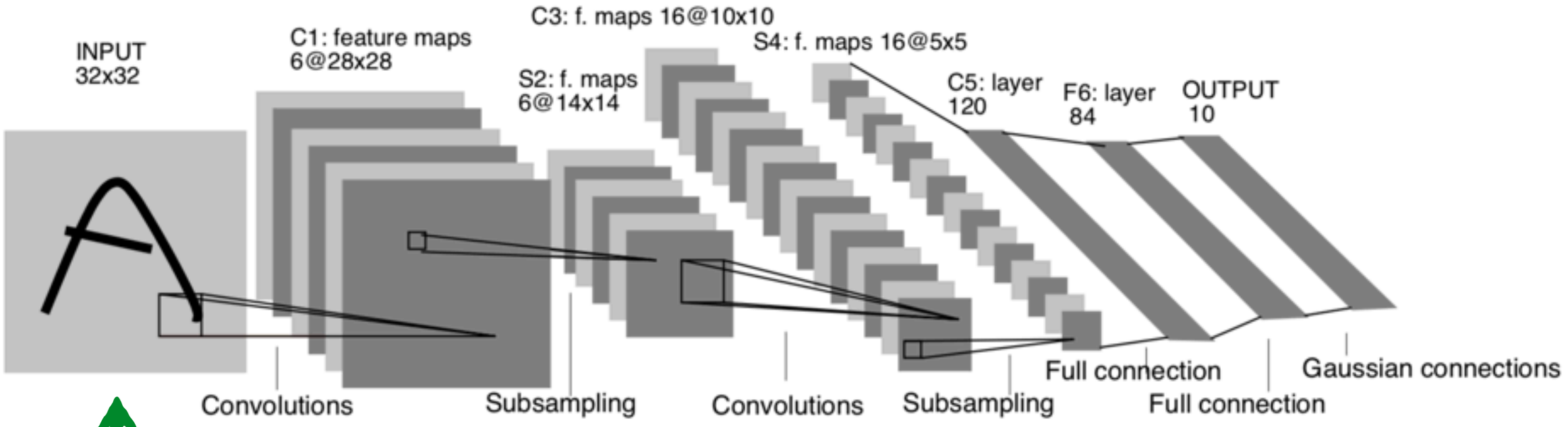

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

as the feature maps in the previous layer. The trainable coefficient and bias control the effect of the sigmoid non-linearity. If the coefficient is small, then the unit operates in a quasi-linear mode, and the sub-sampling layer merely blurs the input. If the coefficient is large, sub-sampling units can be seen as performing a "noisy OR" or a "noisy

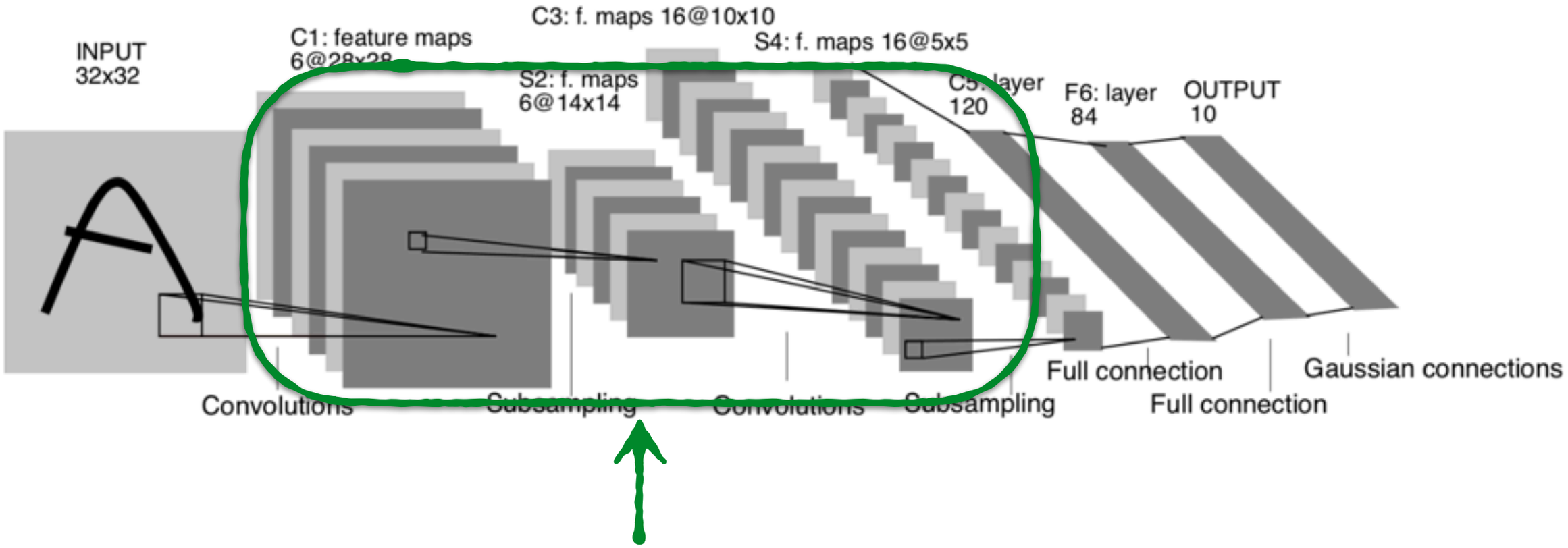
### *B. LeNet-5*

This section describes in more detail the architecture of LeNet-5, the Convolutional Neural Network used in the experiments. LeNet-5 comprises 7 layers, not counting the input, all of which contain trainable parameters (weights).

# CNN for digit recognition


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

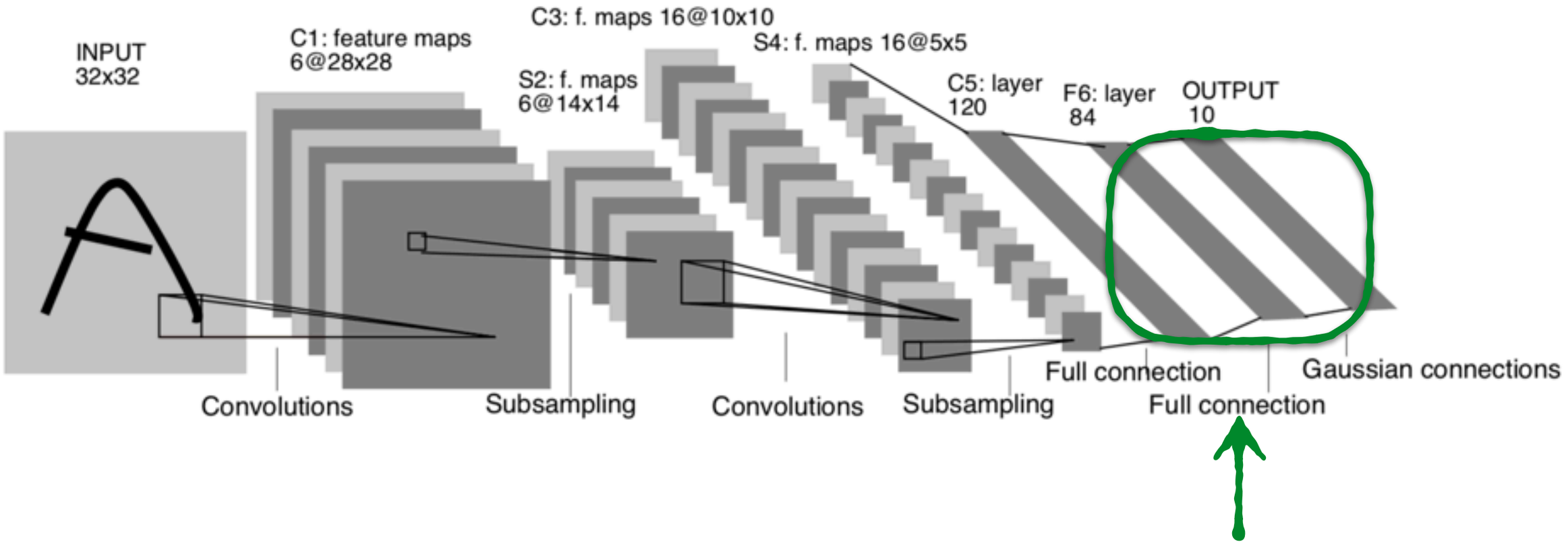
# Image input




# Encoder part.

Extracts a representational vector of the image.

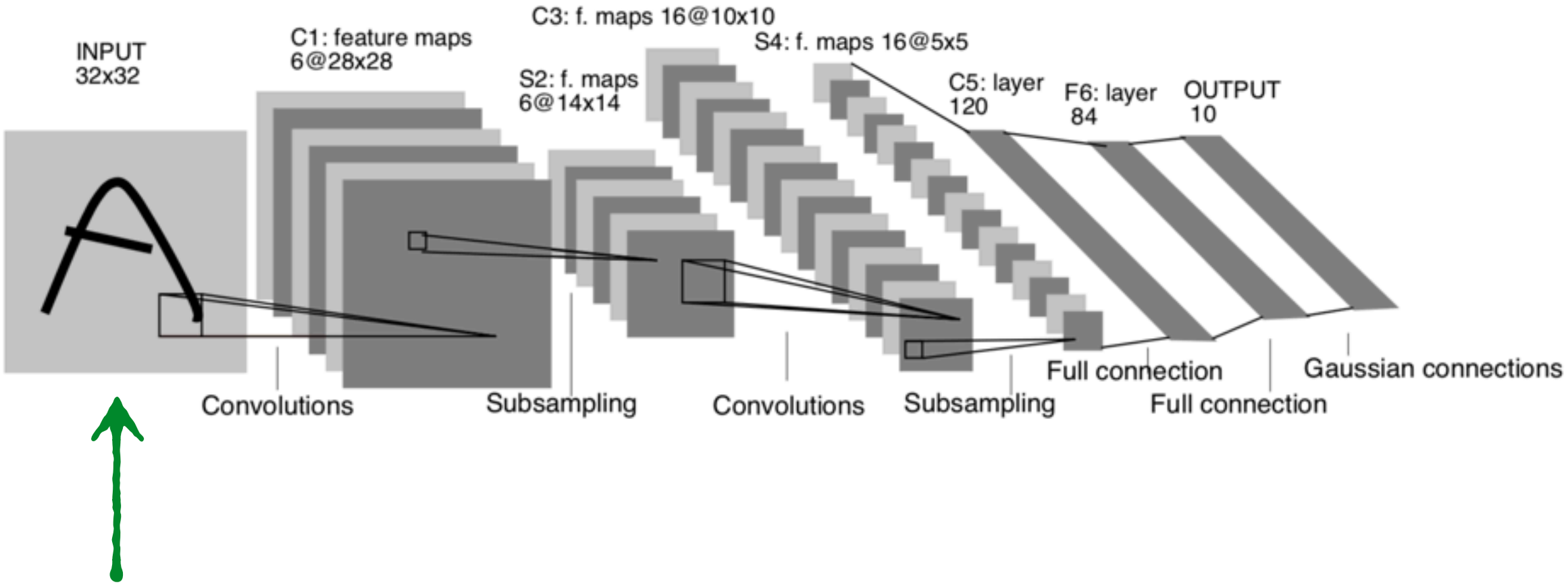
Takes place of SIFT and BOW




## Decoder part.

Maps vector to categorical probabilities.

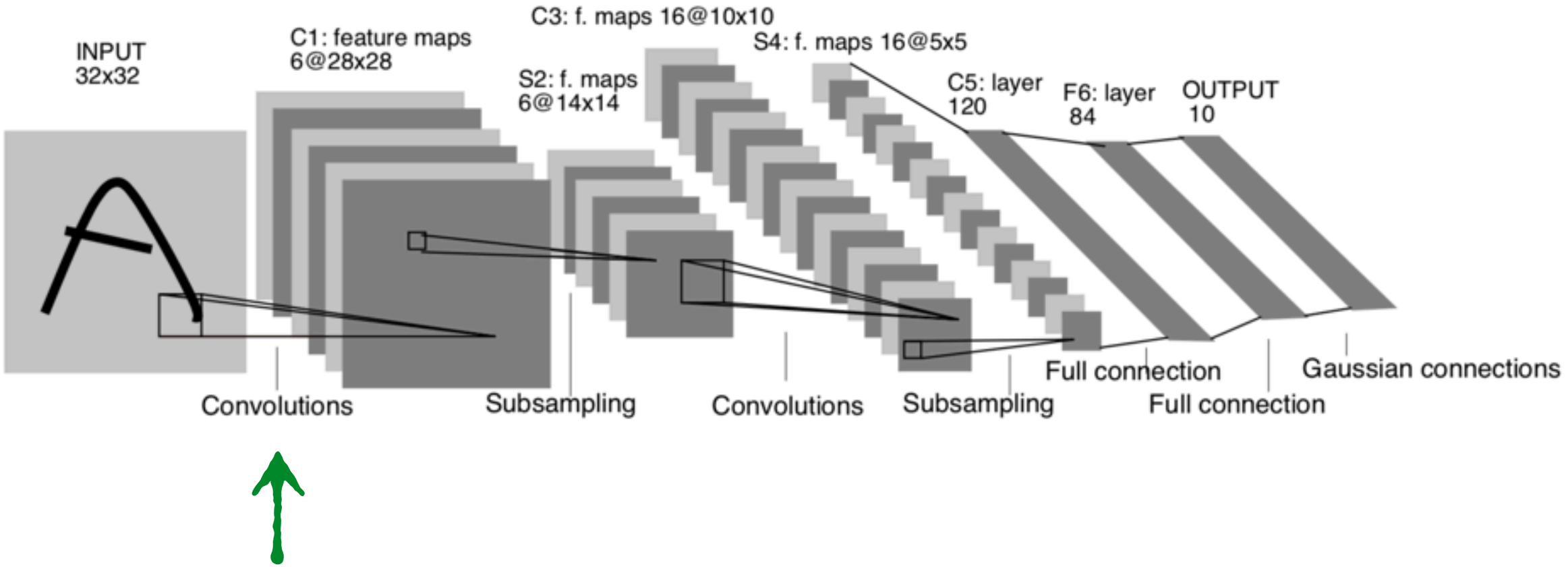
Just the multilayer perceptron covered in last session




Input image size is fixed

*What do you do if your image is too big?*

*What do you do if your aspect ration is wrong?*




Convolutions with small receptive field

*How do convolutions work?*

# Recall:

$f[\cdot, \cdot]$

image

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$h[\cdot, \cdot]$

output

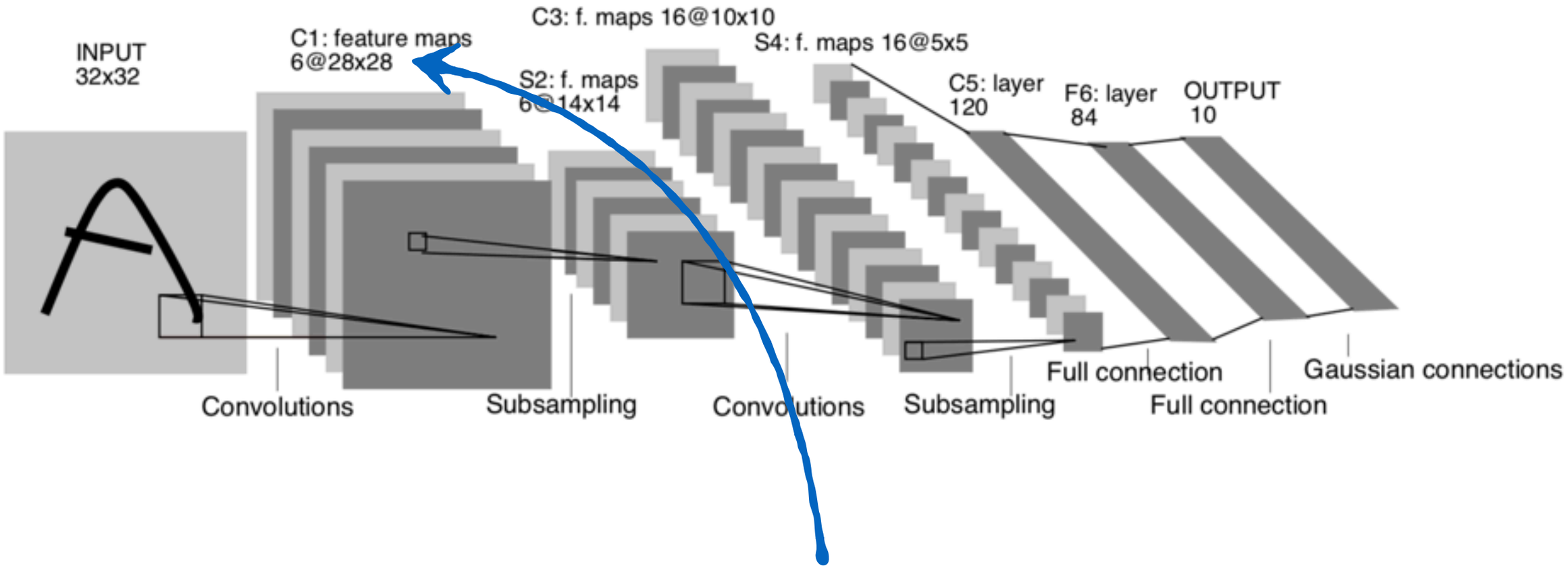
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 10 | 10 | 10 | 10 | 0 | 0 | 0 | 0 | |
| | 10 | 10 | 10 | 10 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | |

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

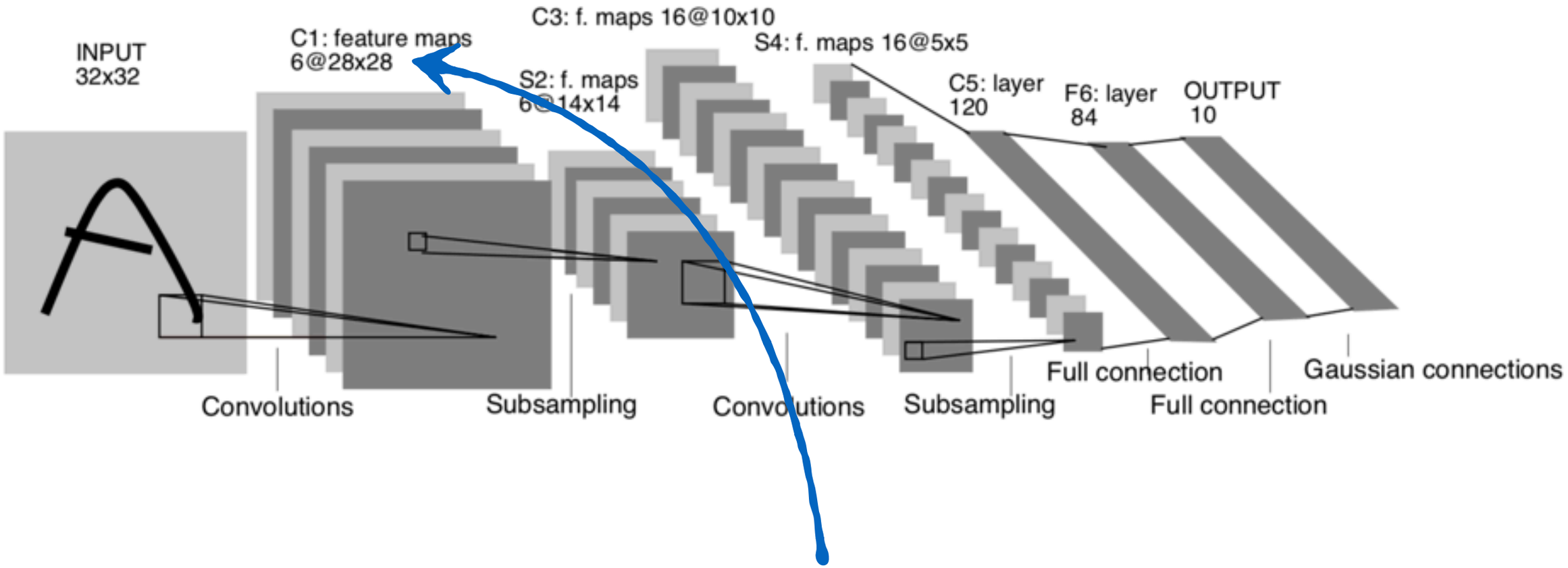
output — filter — image (signal)

Except minus sign here

* some zero values are white for visualization but they should be black




*If the feature map is 28 x 28,*
*what is the size of the convolution kernel?*



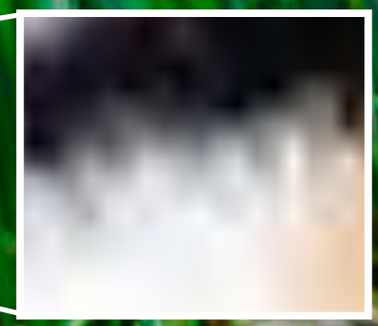

*If the feature map is 28 x 28,*
*what is the size of the convolution kernel?*

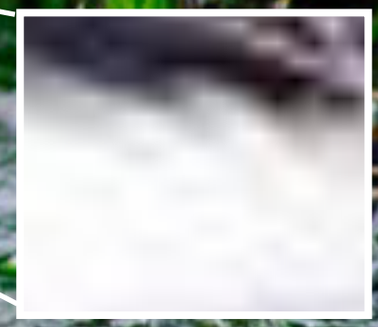
If you don't assume border padding,
then kernel size is 5 x 5.

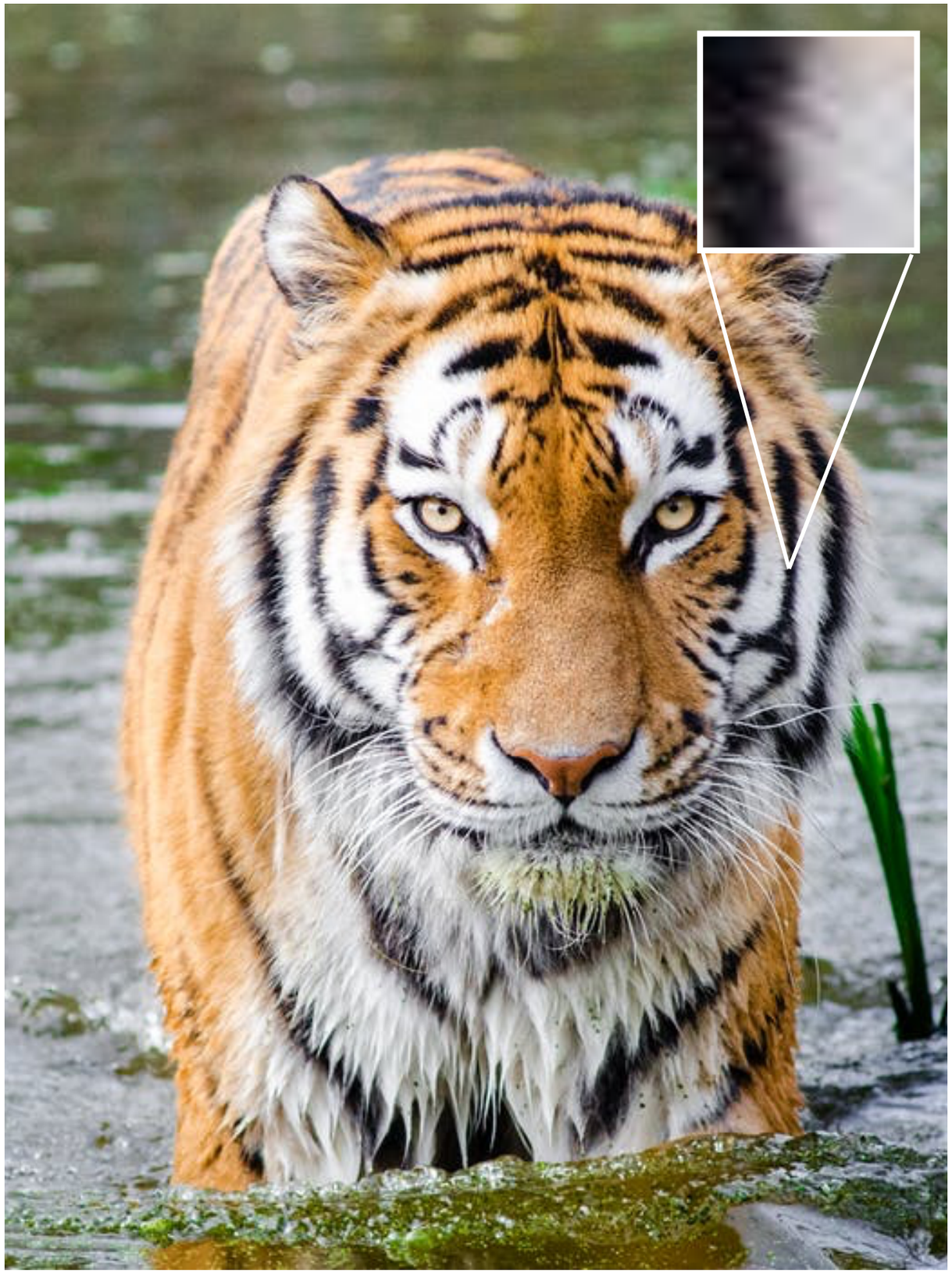
Can't compute convolution for 2 pixels around the border.

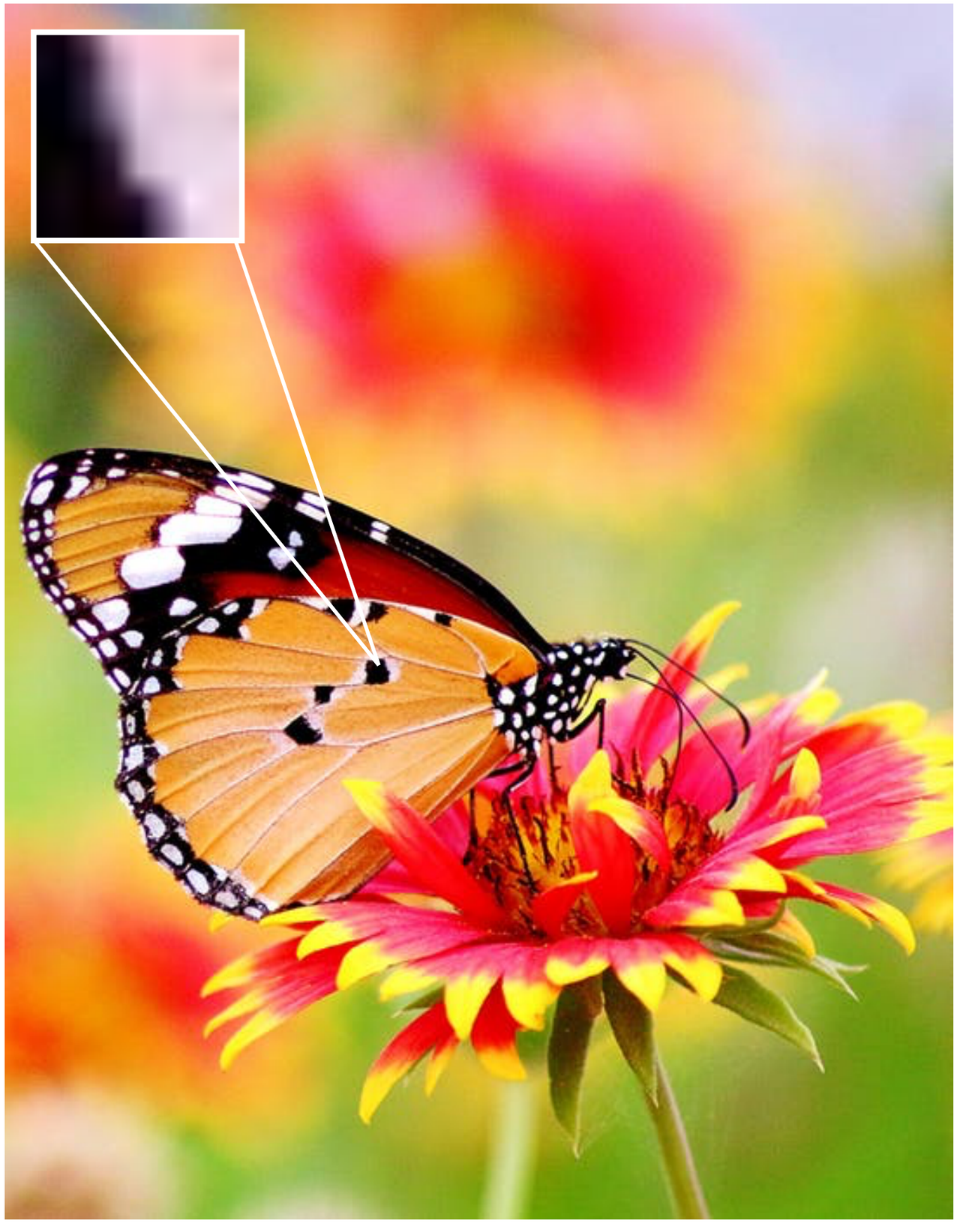
*Why not use a bigger kernel size?*

# On a micro-scale, many features of an object look similar









# .. features also similar across objects

**Recall:**

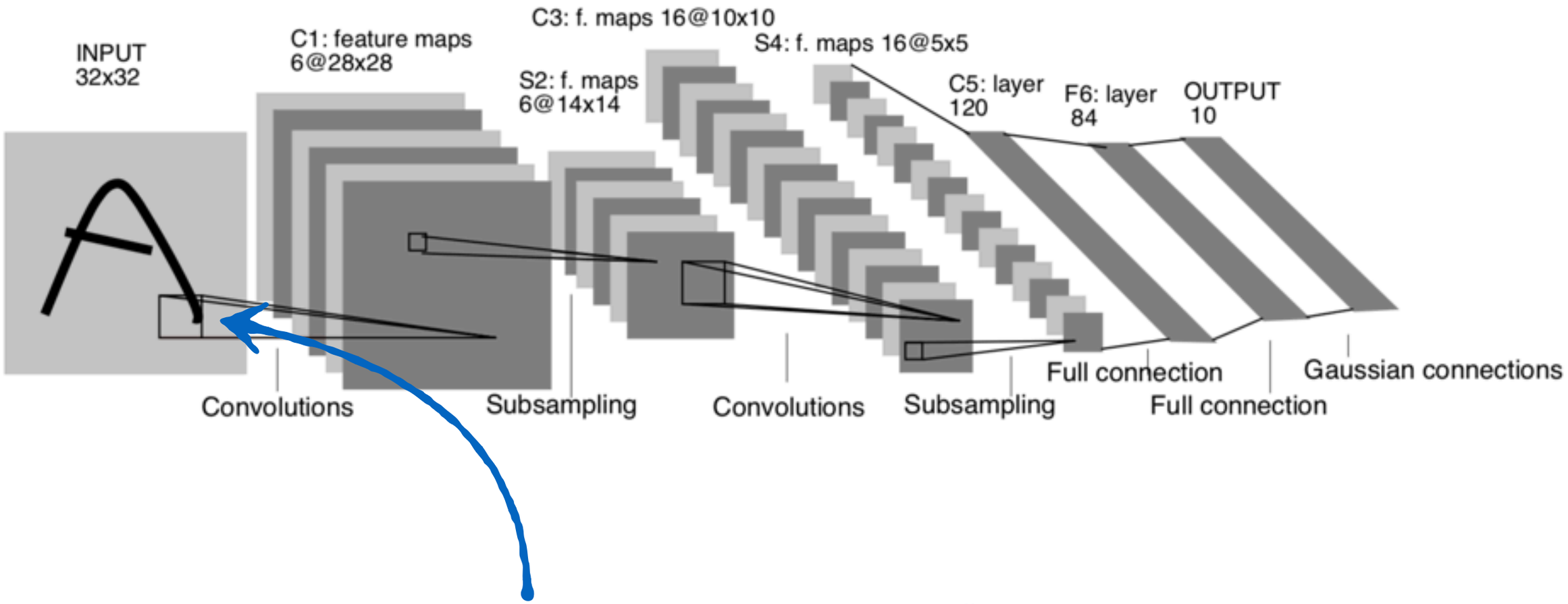
# Harris Detector

C.Harris and M.Stephens. “A Combined Corner and Edge Detector.”1988.

Gradient covariance matrix

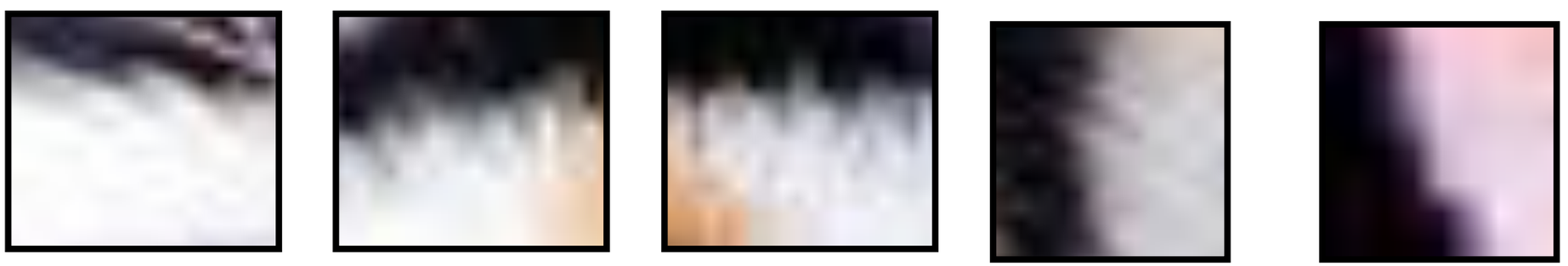
$$\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

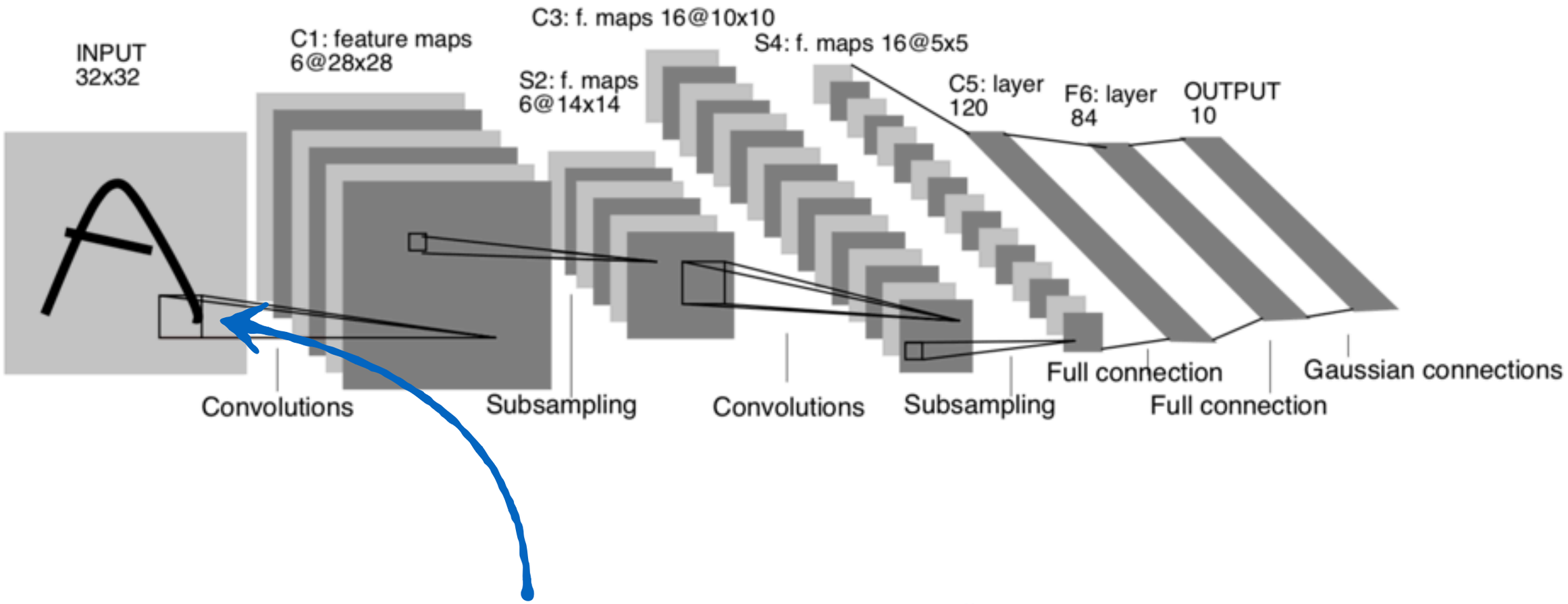
Harris corner detector also used
a **small image region P** to compute corners




*Why not use a bigger kernel size?*

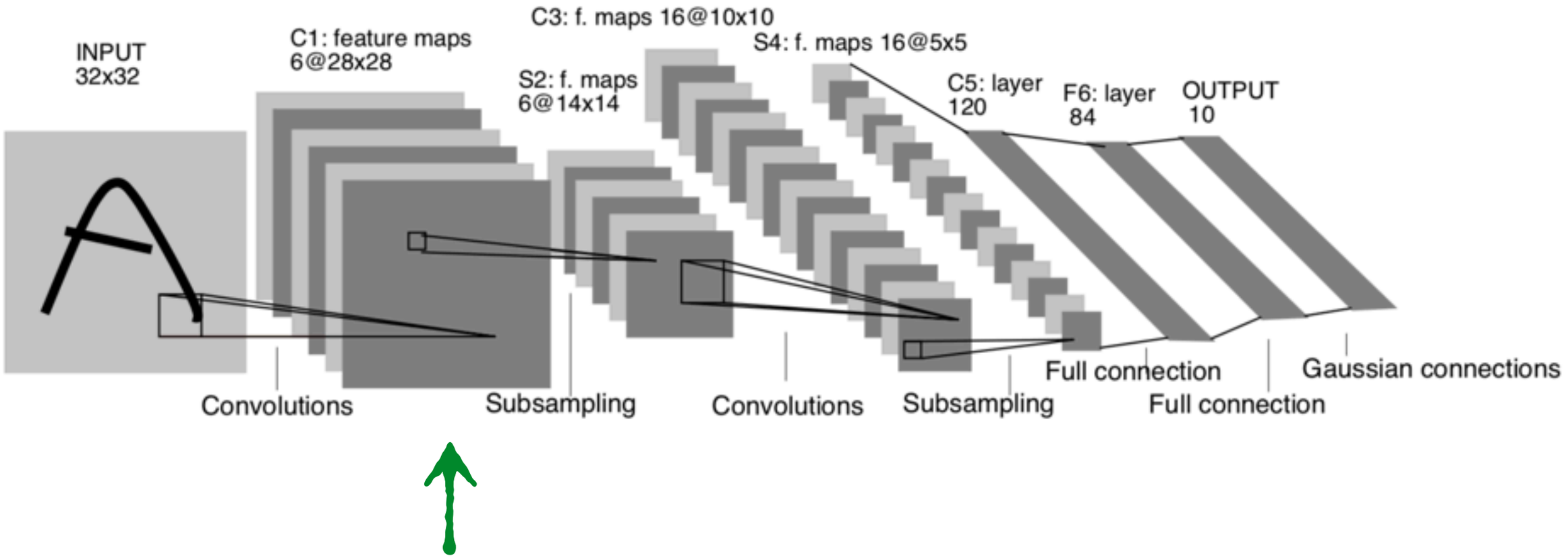
A small kernel can capture common primitive patterns across a wide range of images





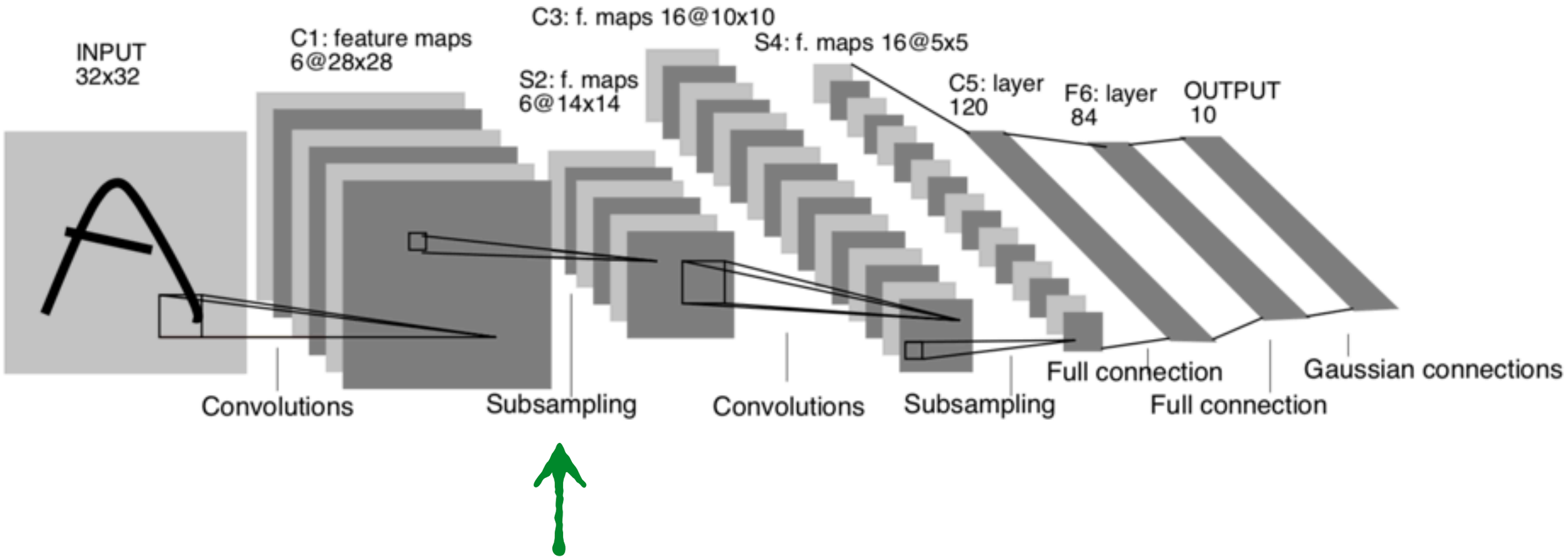

*Why not use a bigger kernel size?*

Can be used reused over the entire image
(shared weights)



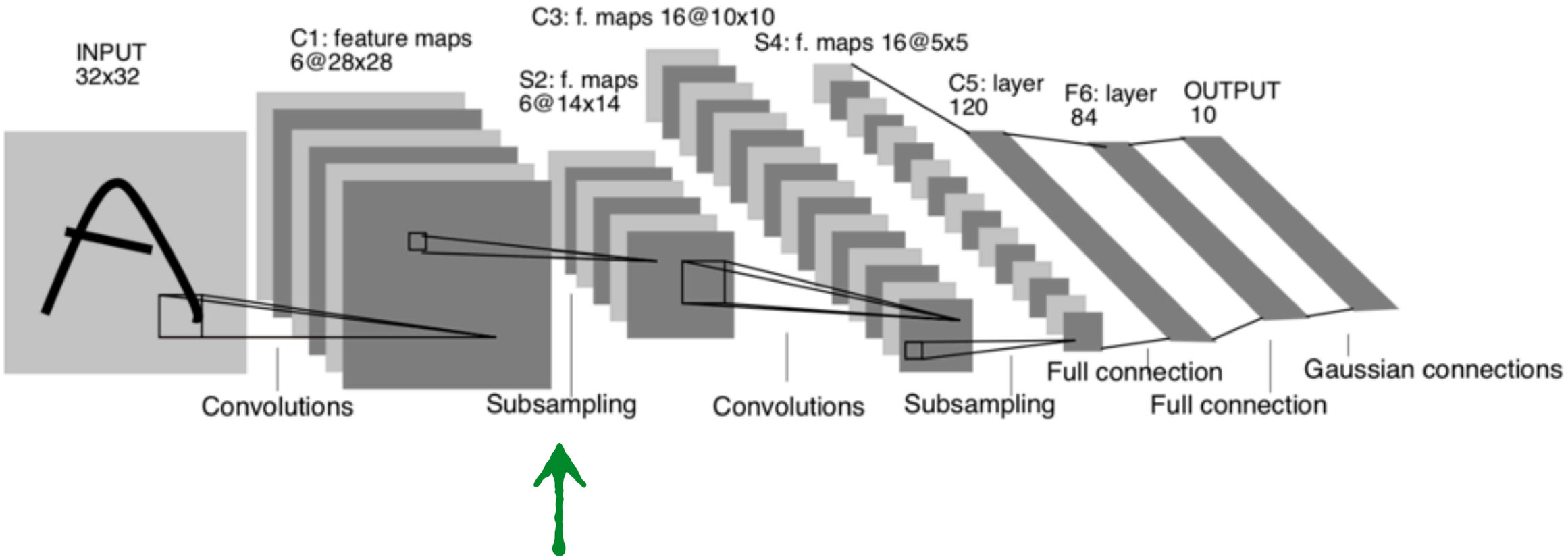

One convolution kernel
results in one feature map

*How many kernels did we use here?*




# Subsampling reduces the size of feature maps

(actually average pooling with scale and bias, sigmoid function)



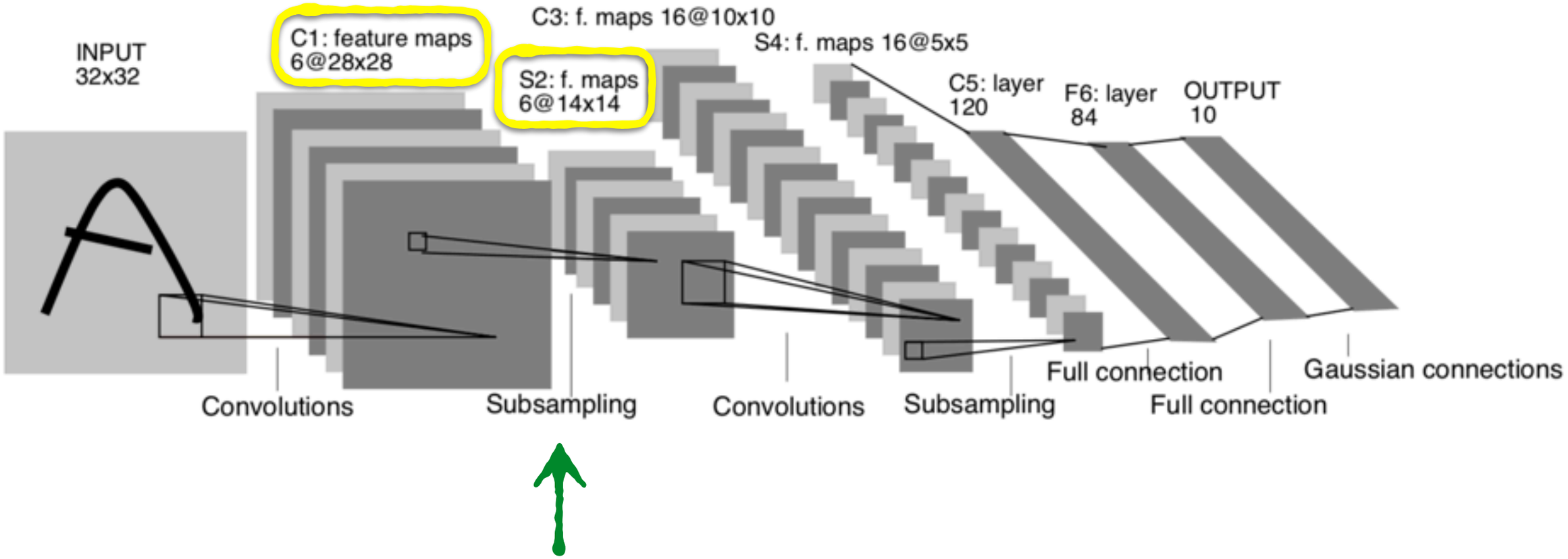

# Subsampling reduces the size of feature maps

(actually average pooling with scale and bias, sigmoid function)

$$h[i,j] = \text{sigmoid}\left(\alpha \cdot \sum_{u,v} g[u,v] f[i-u, j-v] - \beta\right)$$

scale — ‘Average Pooling’ — bias

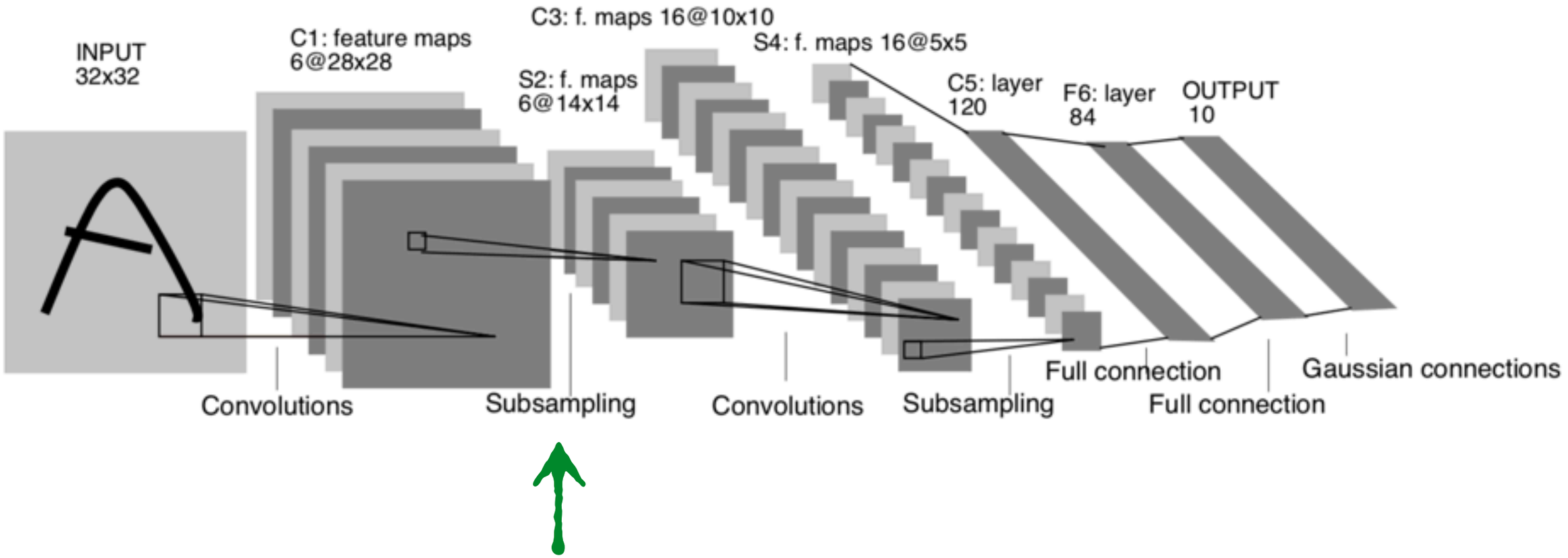
Box filter — Feature map




Subsampling reduces the size of feature maps

(average pooling with scale and bias, sigmoid function)

*What is the step size and receptive field of subsampling?*

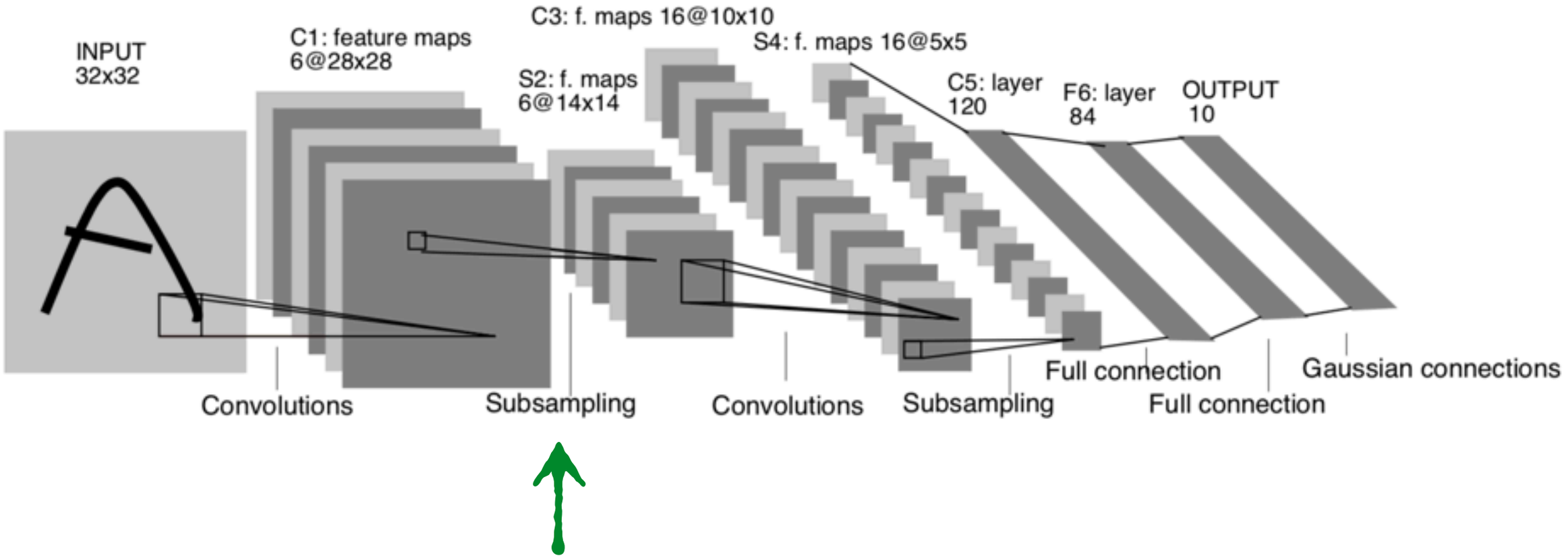



Subsampling reduces the size of feature maps

(average pooling with scale and bias, sigmoid function)

*What is the step size and receptive field of subsampling?*

Step size: 2, Receptive field: 2 x 2



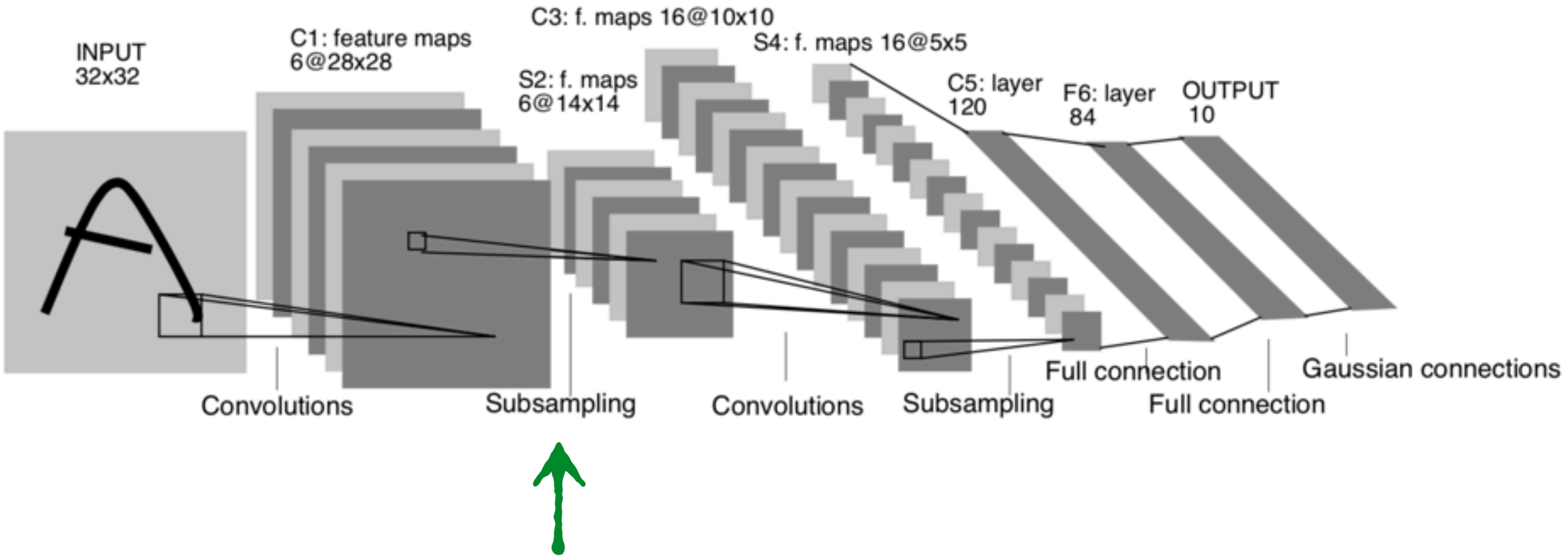

Subsampling reduces the size of feature maps

(average pooling with scale and bias, sigmoid function)

*What is the step size and receptive field of subsampling?*

Step size: 2, Receptive field: 2 x 2

*How does subsampling affect the image representation?*




Subsampling reduces the size of feature maps

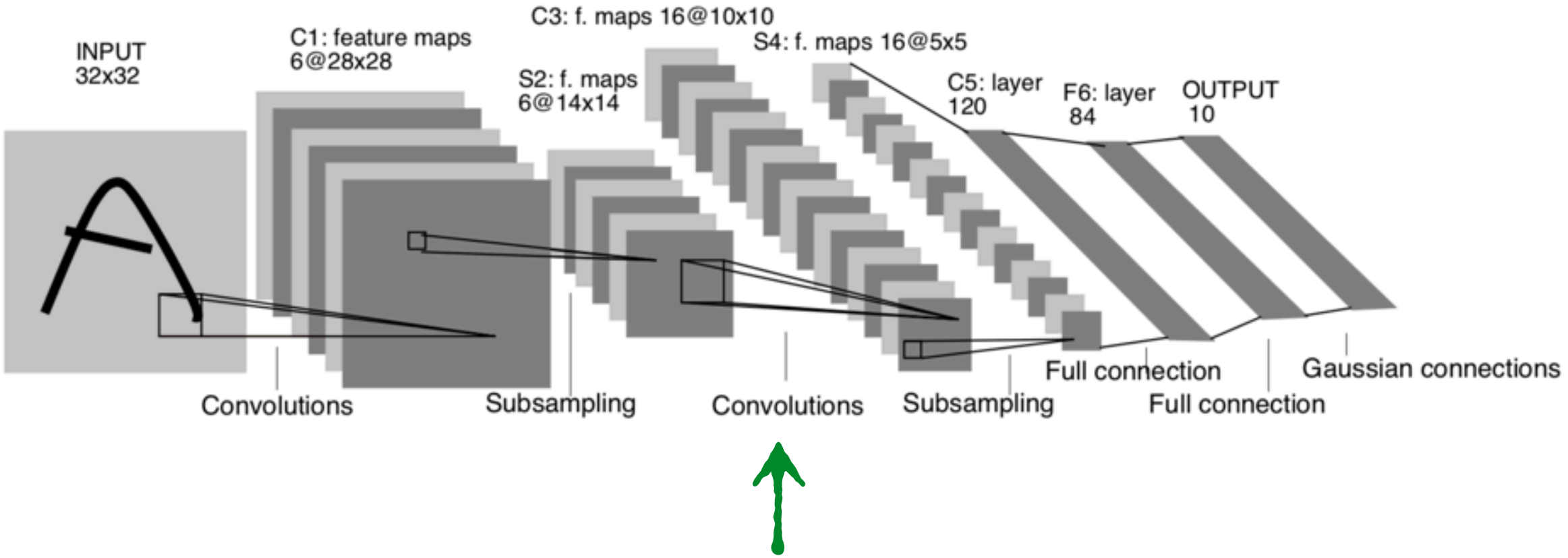
(average pooling with scale and bias, sigmoid function)

*What is the step size and receptive field of subsampling?*

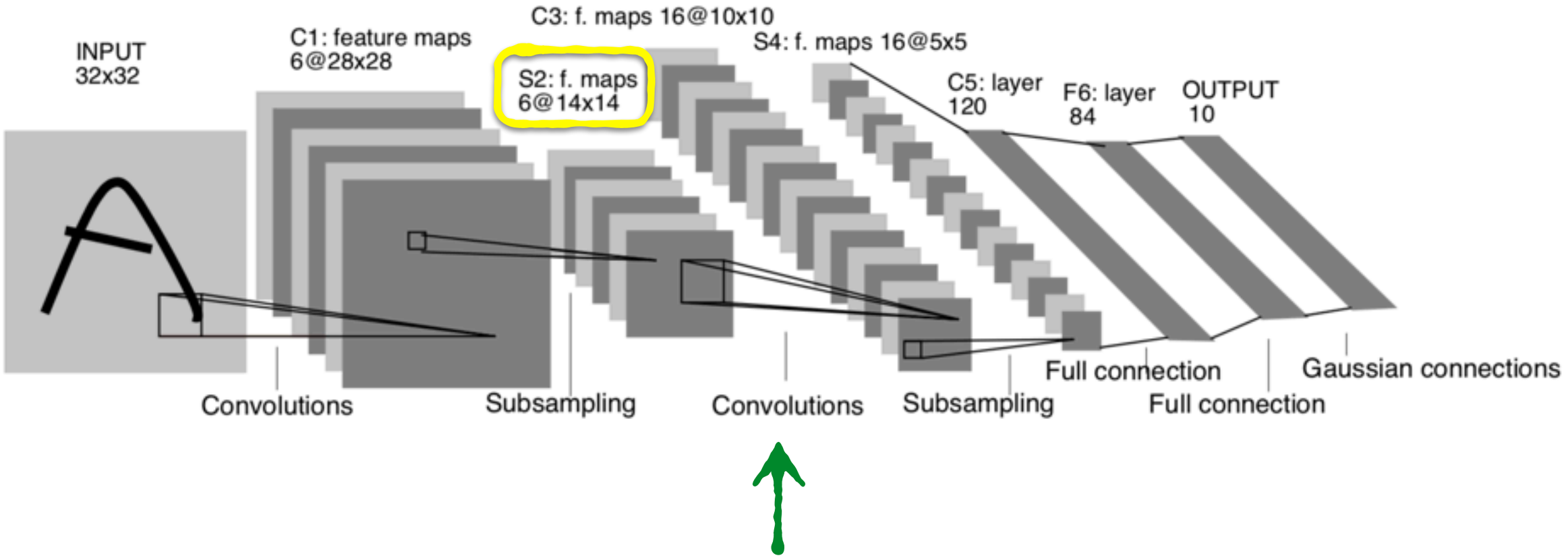
Step size: 2, Receptive field: 2 x 2

*How does subsampling affect the image representation?*

Low pass filter. ‘Blurs’ the image.

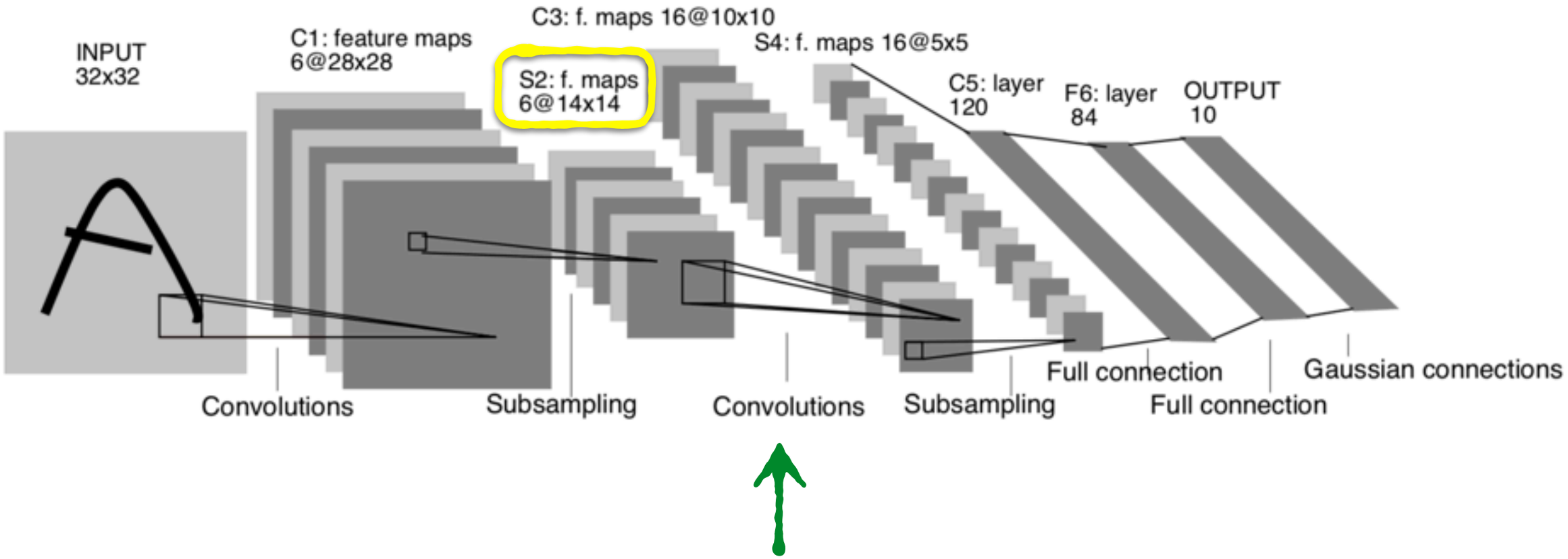



# Convolution is now multichannel



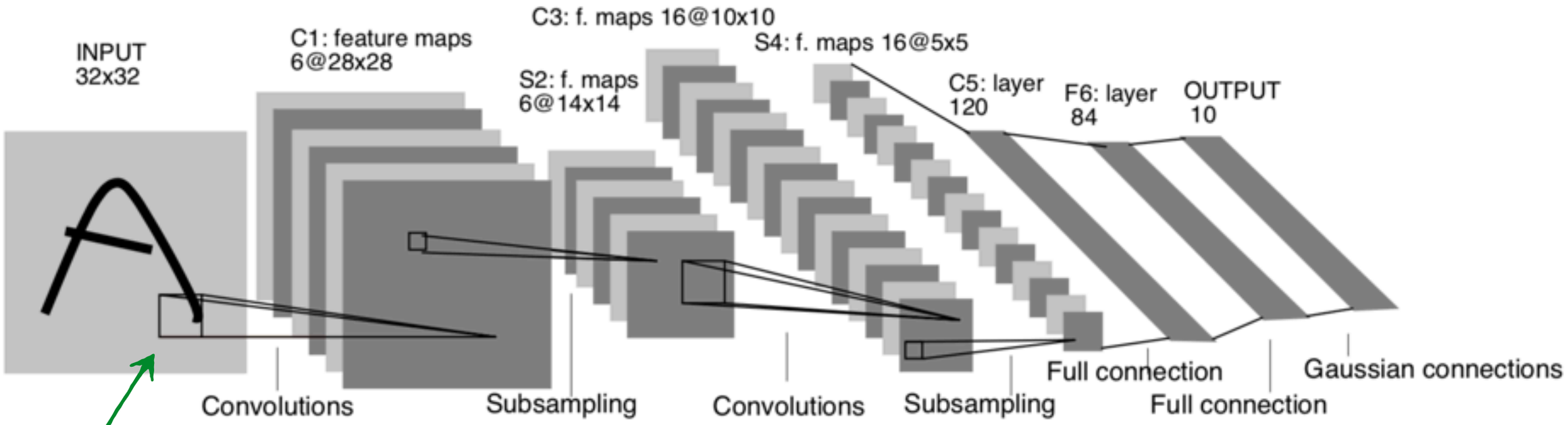

Convolution is now multichannel

*How many channels is one convolutional filter?*




Convolution is now multichannel

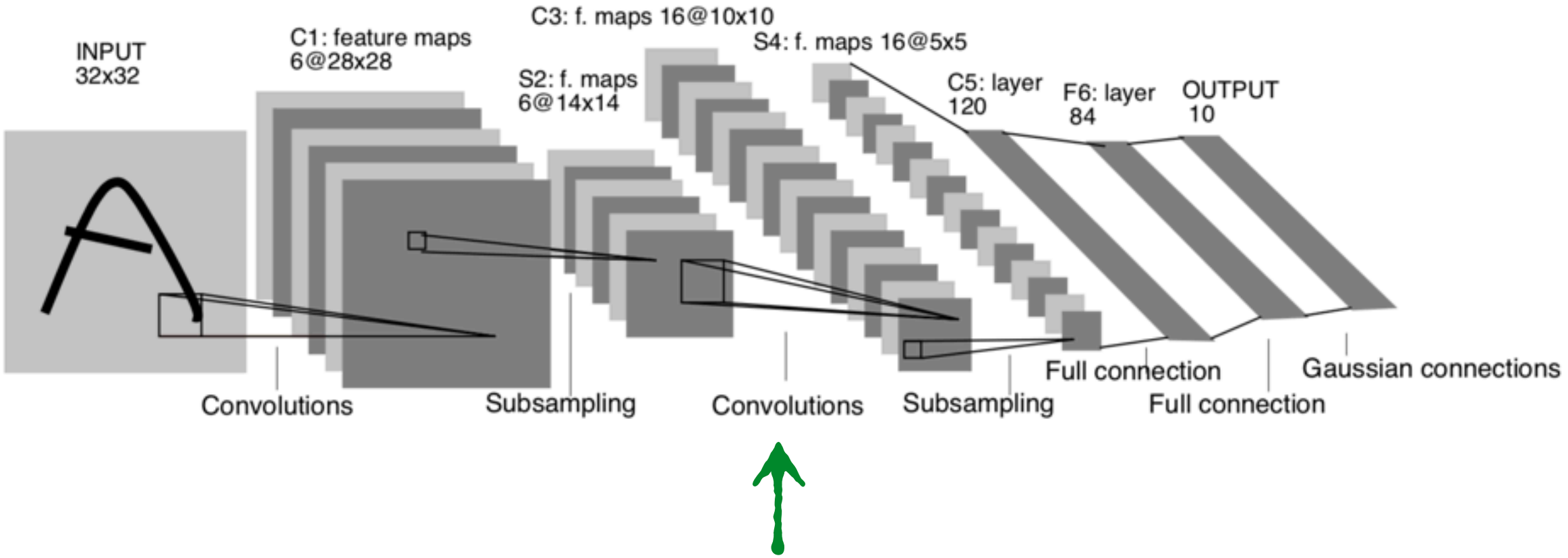
$$h[i,j] = \sum_k \sum_u \sum_v g[u,v,k] \; f[i-u, j-v, k]$$




Convolution is now multichannel

$$h[i,j] = \sum_k \sum_u \sum_v g[u,v,k] \; f[i-u, j-v, k]$$

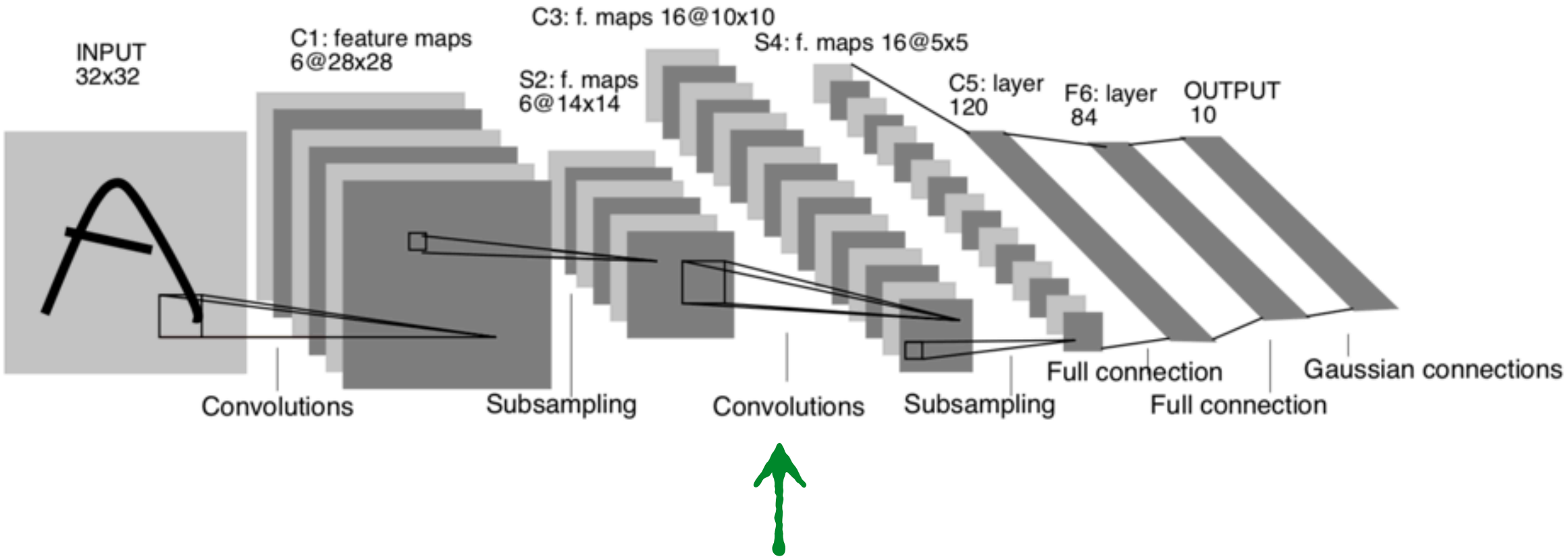
k was there before … it was just k=1




Convolution is now multichannel

$$h[i,j] = \sum_k \sum_u \sum_v g[u,v,k] \; f[i-u, j-v, k]$$

*Is this a 3D convolution?*

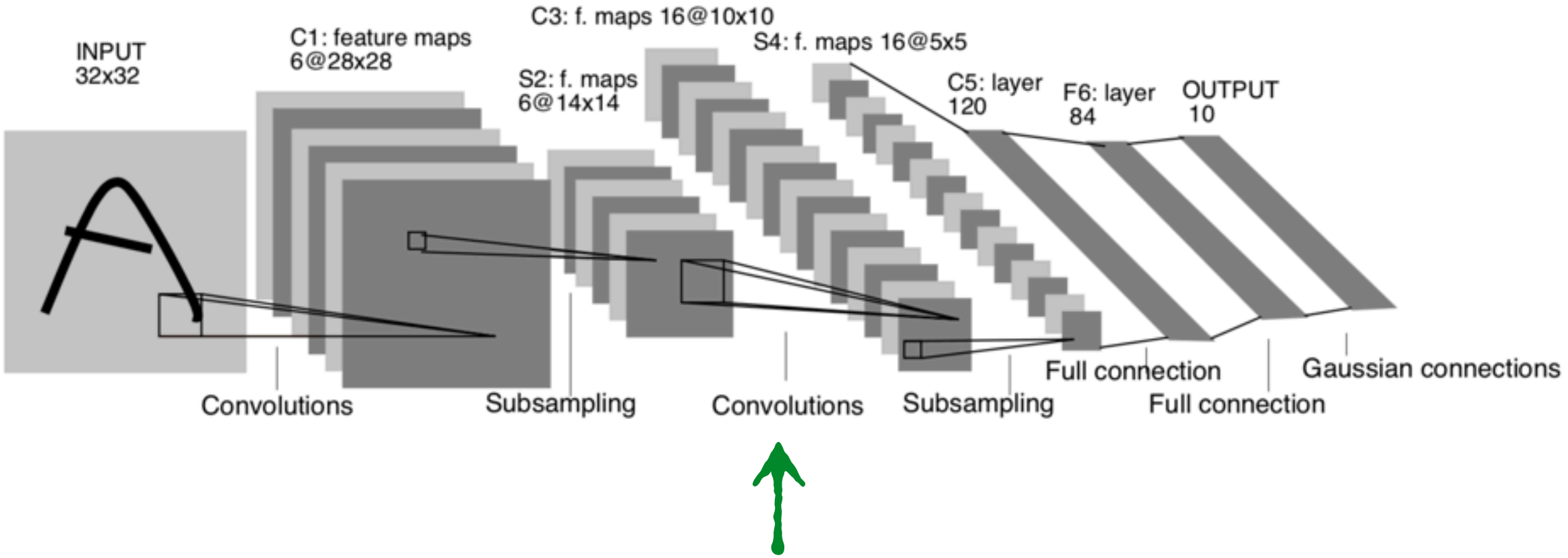



Convolution is now multichannel

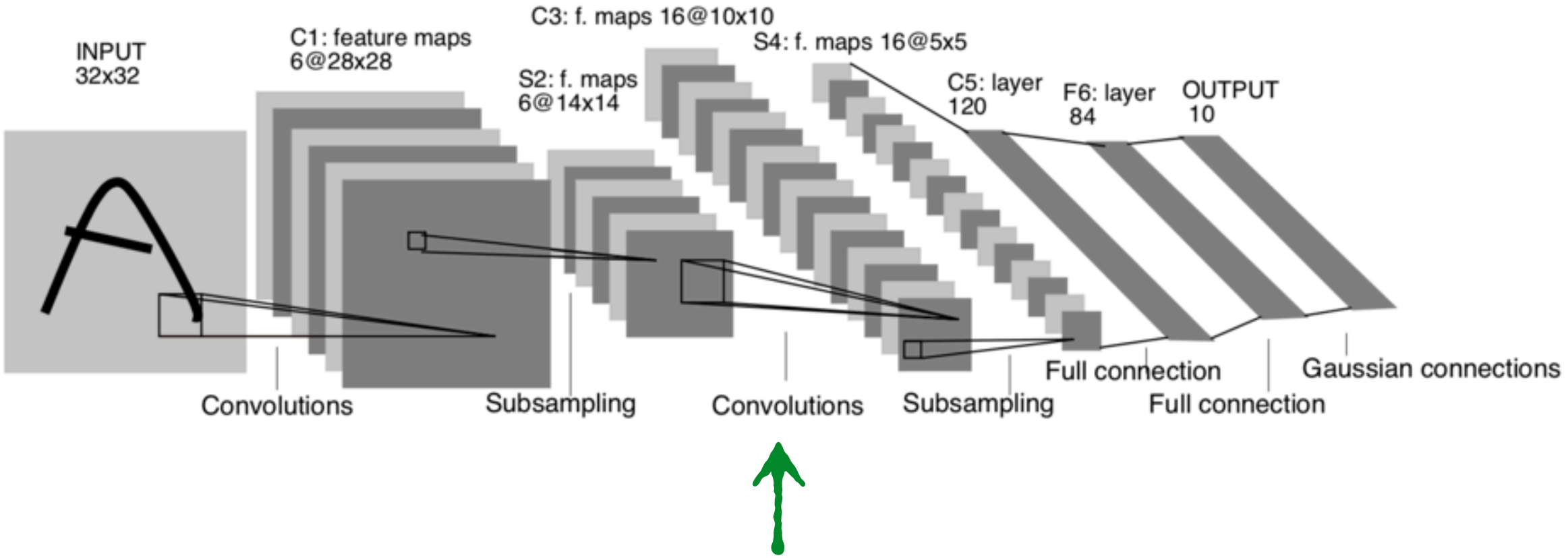
$$h[i,j] = \sum_k \sum_u \sum_v g[u,v,k] \; f[i-u, j-v, k]$$

*Is this a 3D convolution?*

No. Why?

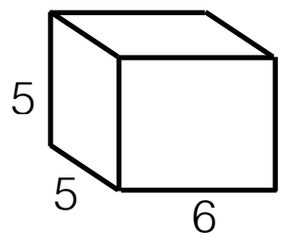



*What is the size of the convolutional filter in LeNet5 at C3?*



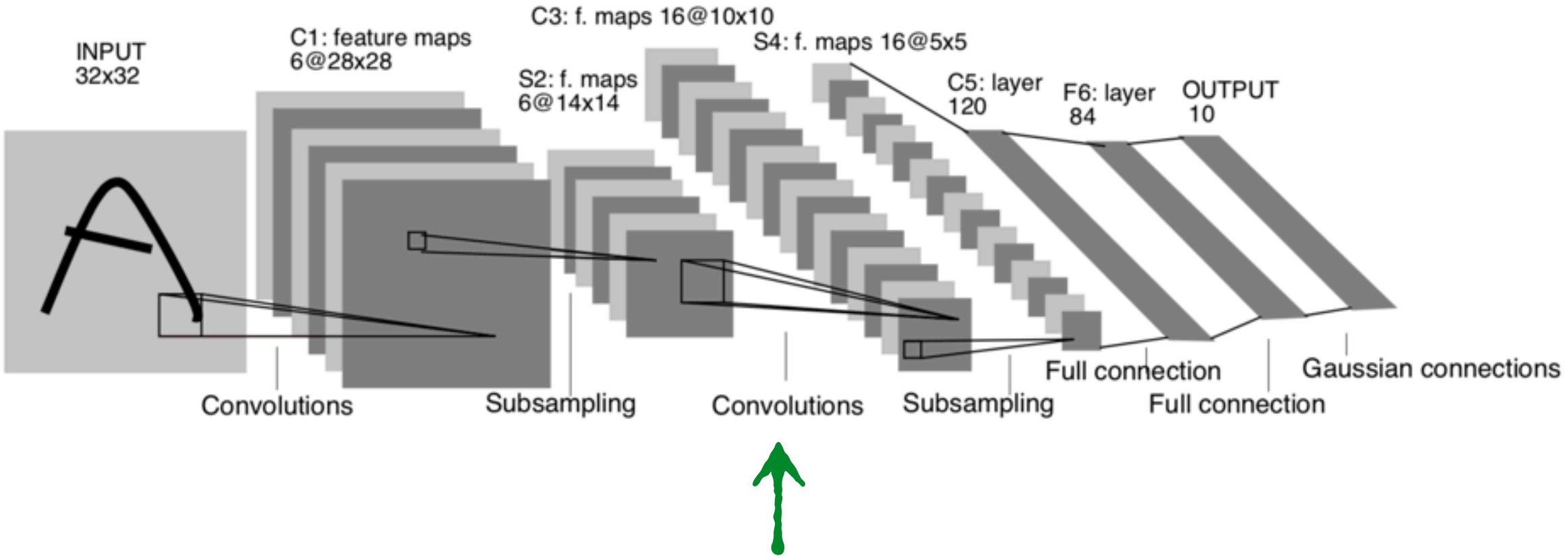

*What is the size of the convolutional filter in LeNet5 at C3?*

5 x 5 x 6



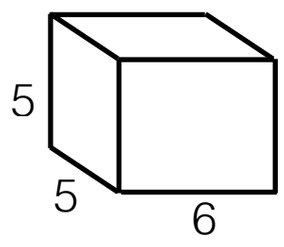

BUT not all channels are used!

*Why?*




# *What is the size of the convolutional filter in LeNet5 at C3?*

5 x 5 x 6




BUT not all channels are used!

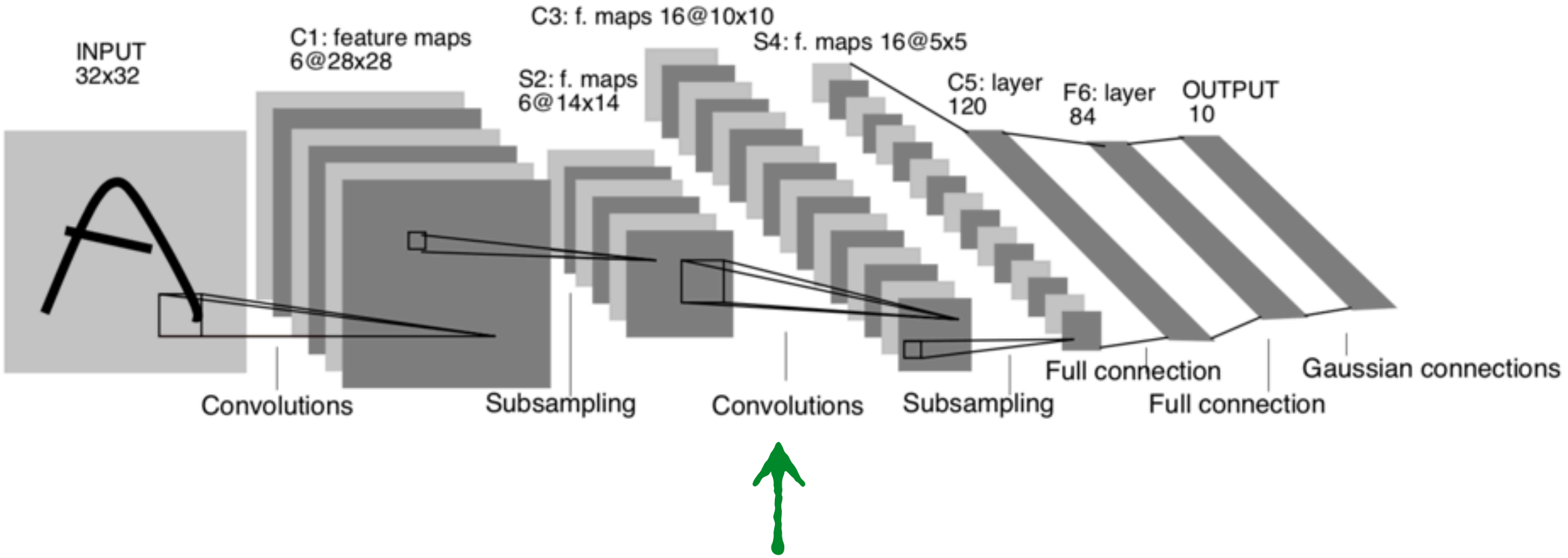
*Why?*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | X | | | | X | X | X | | | X | X | X | X | | X | X |
| 1 | X | X | | | | X | X | X | | | X | X | X | X | | X |
| 2 | X | X | X | | | | X | X | X | | | X | | X | X | X |
| 3 | | X | X | X | | | X | X | X | X | | | X | | X | X |
| 4 | | | X | X | X | | | X | X | X | X | | X | X | | X |
| 5 | | | | X | X | X | | | X | X | X | X | | X | X | X |

TABLE I

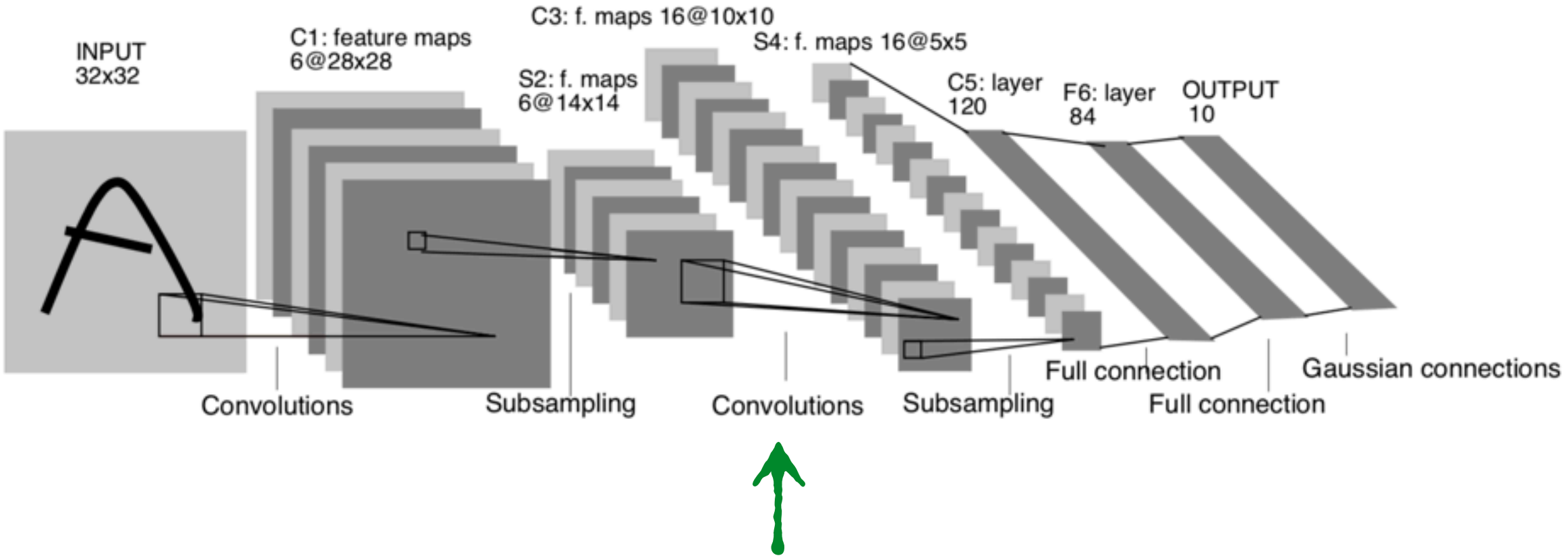
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Forces network to learn different features




Convolution is now multichannel

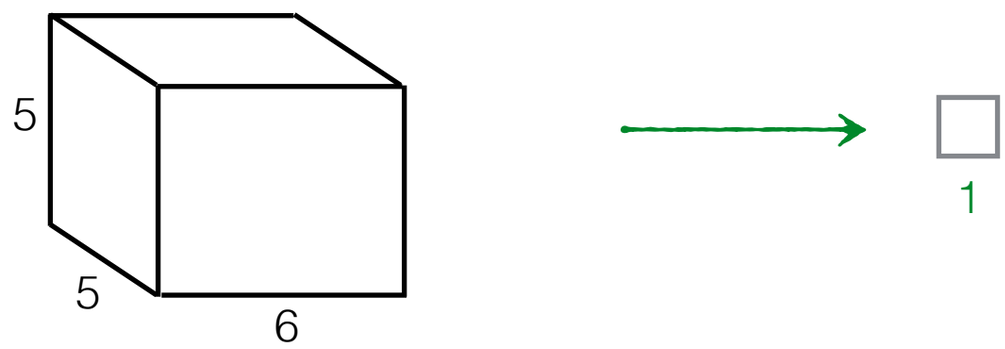
*How many features maps for one multi-channel convolution?*

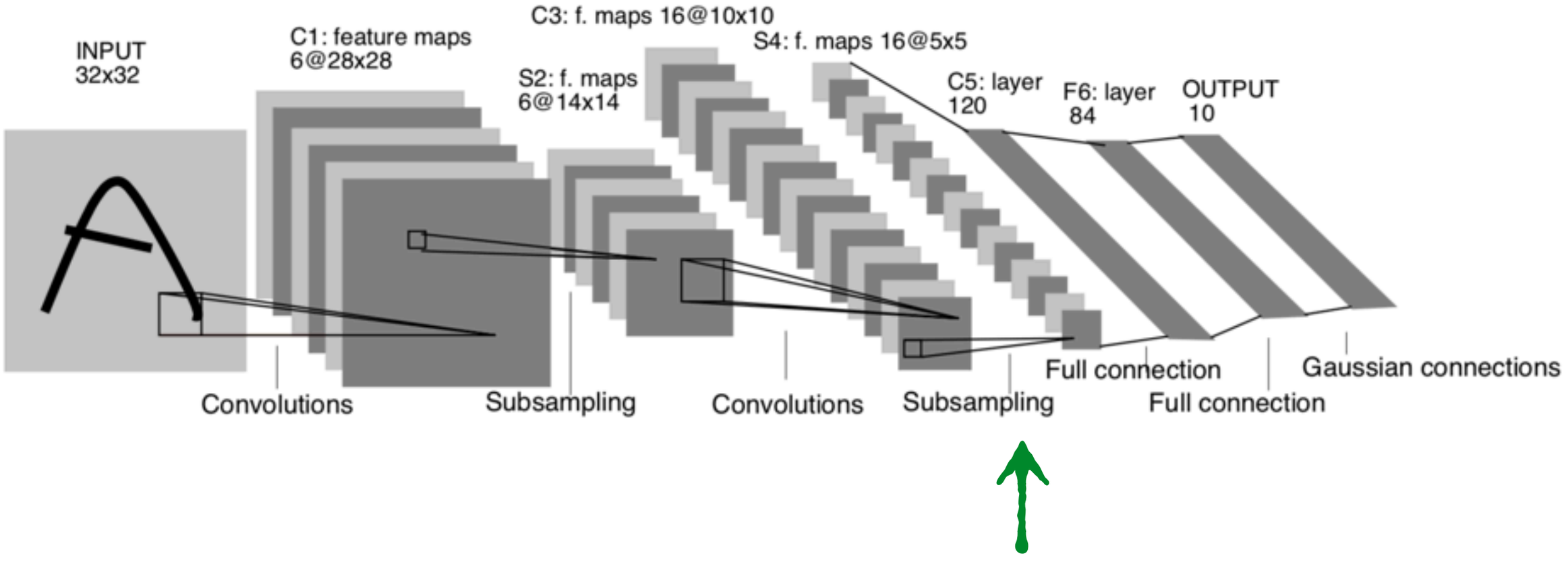



# Convolution is now multichannel

*How many features maps for one multi-channel convolution?*

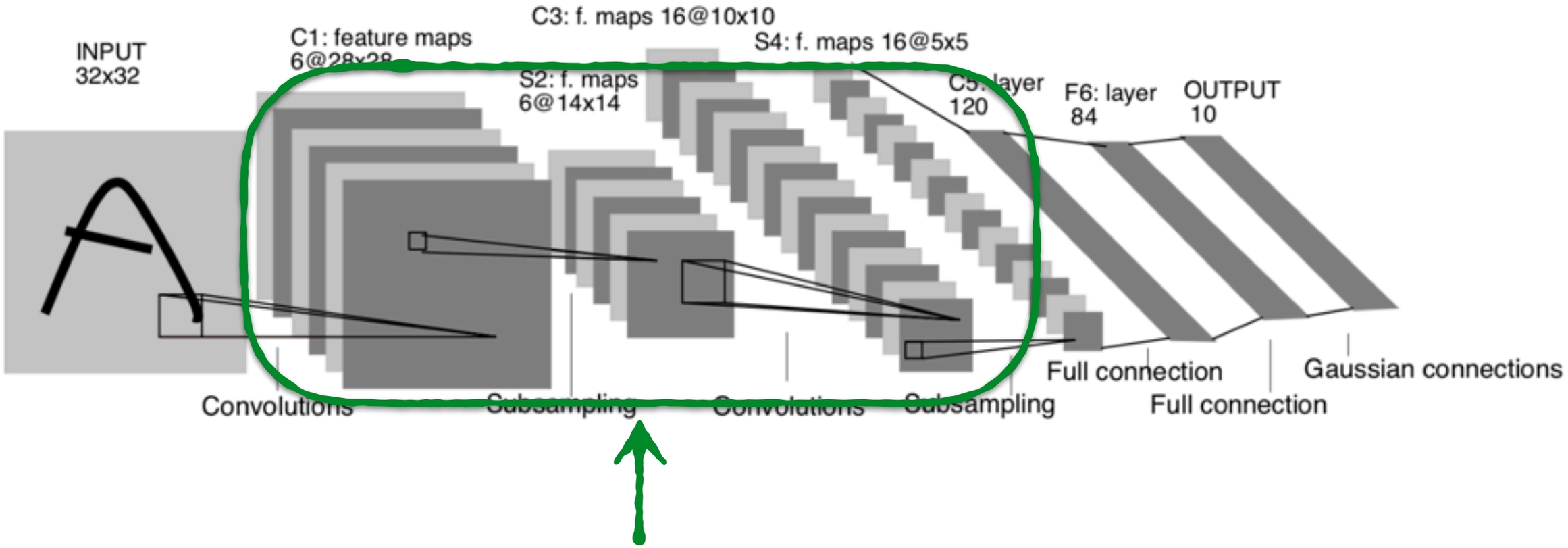
5 x 5 x 6





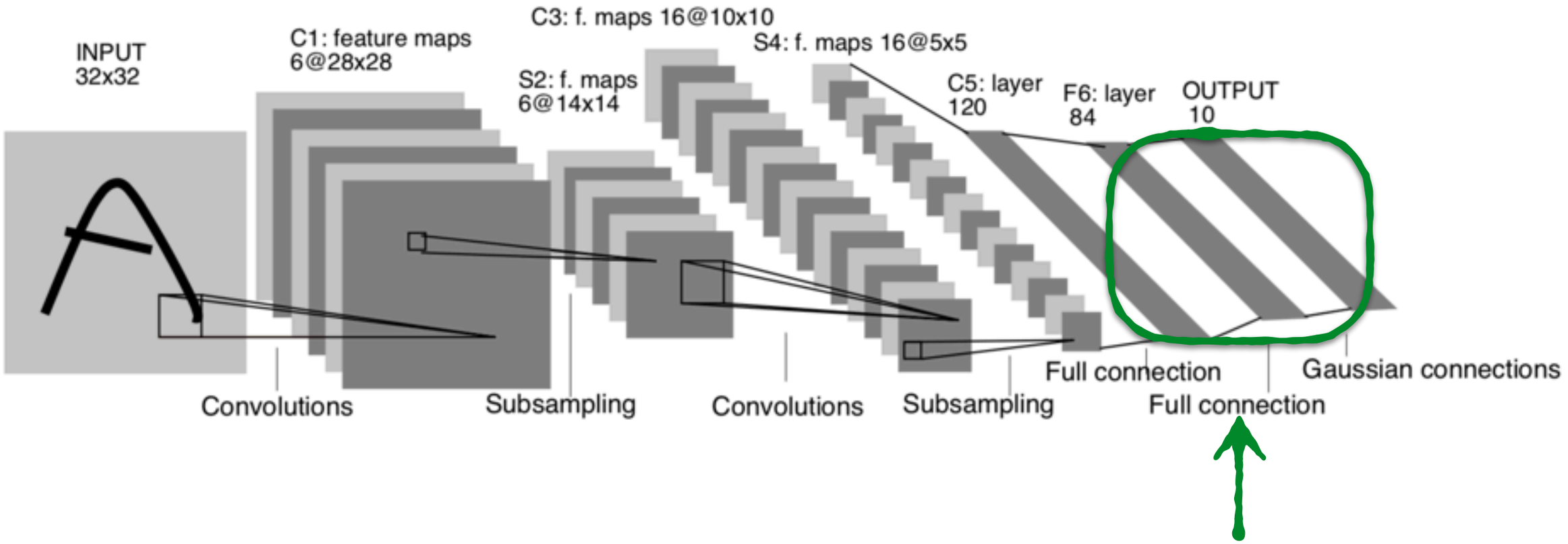

# Subsampling.

Same as before: average pooling + sigmoid



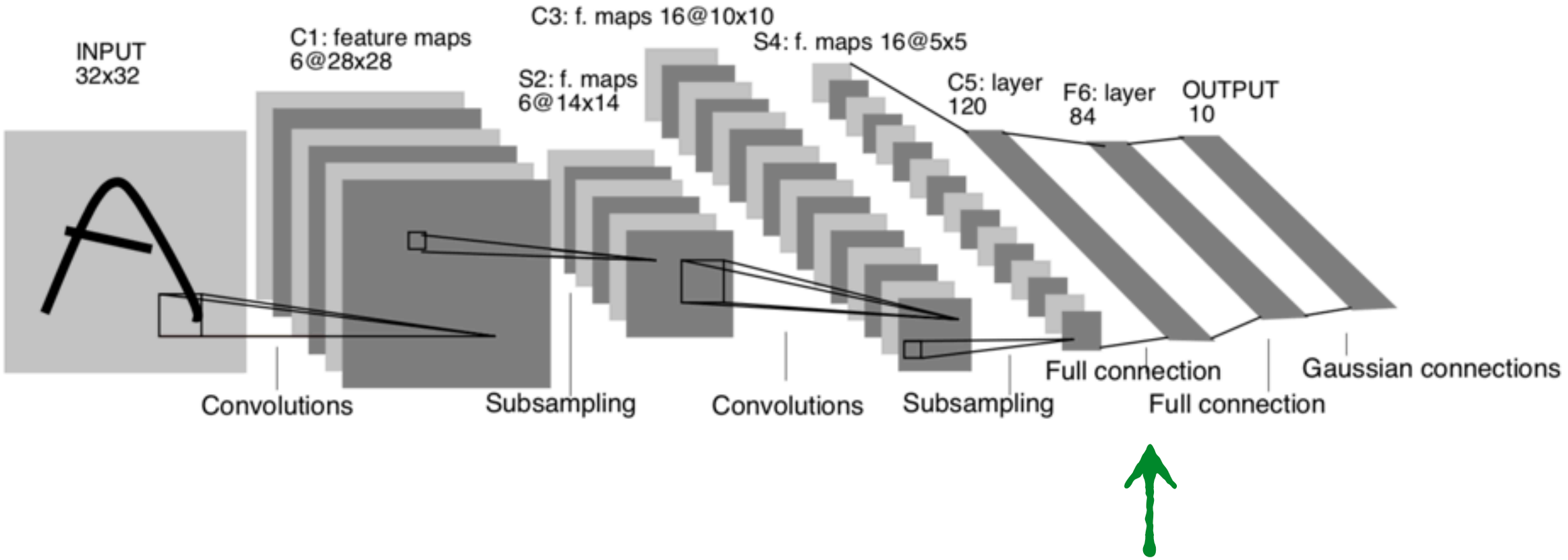

Encoder part.

Extracts a representational vector of the image.



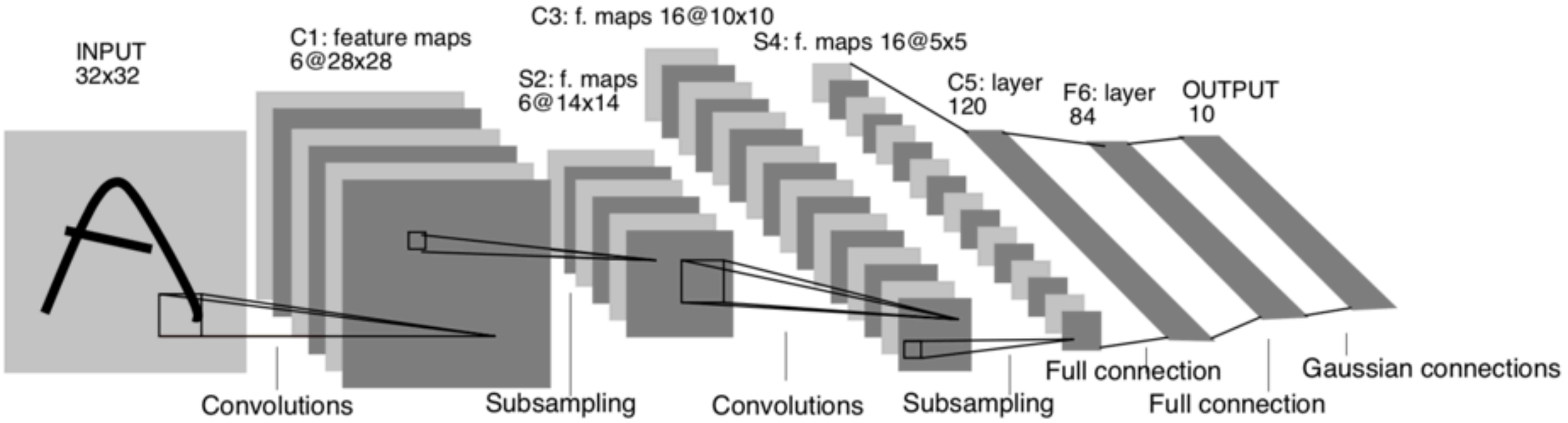

# Decoder part.

Maps vector to categorical probabilities.



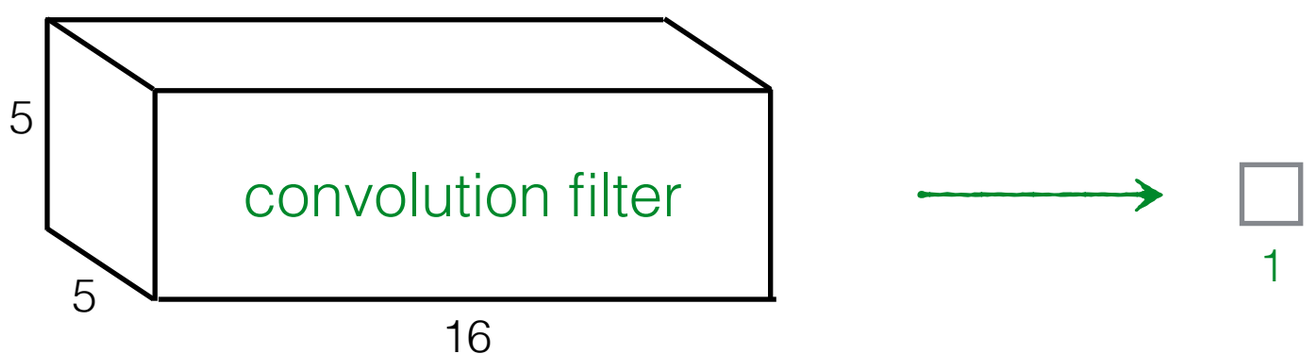

Full connection.

Actually more like a convolution.

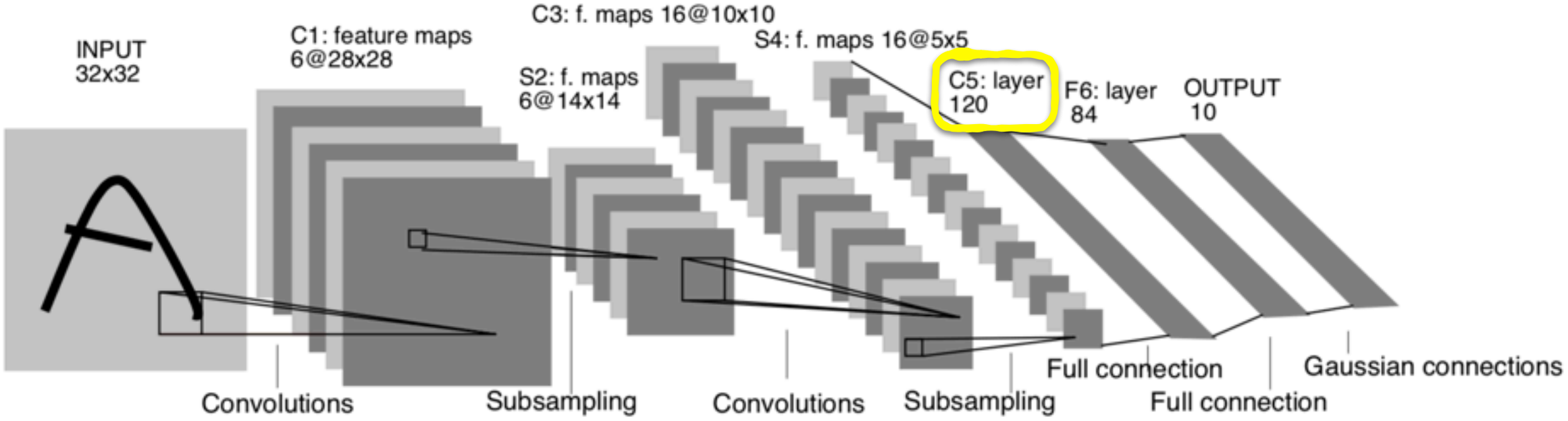



# Full connection.

Actually more like a convolution.

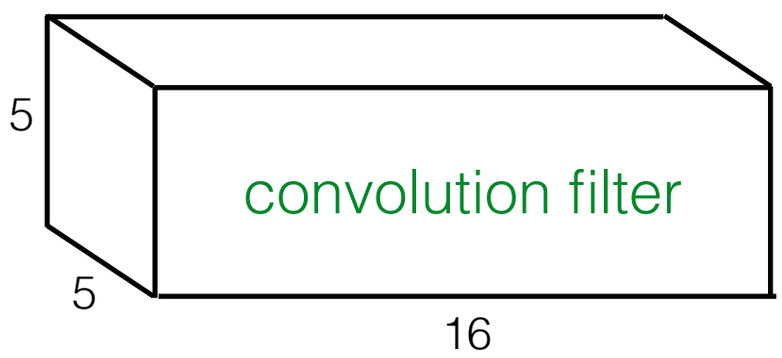



All channels used this time

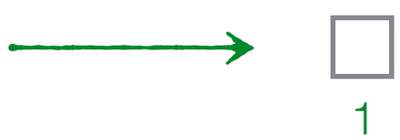



## Full connection.

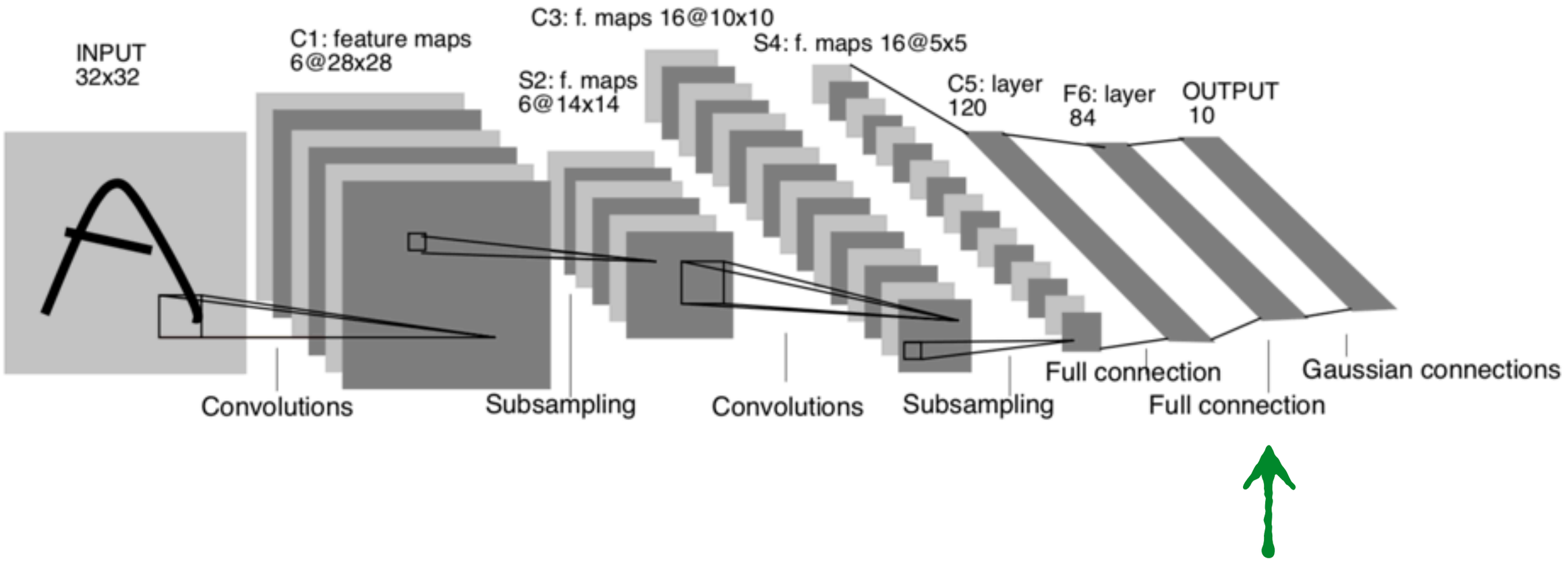
Actually more like a convolution.




All channels used this time

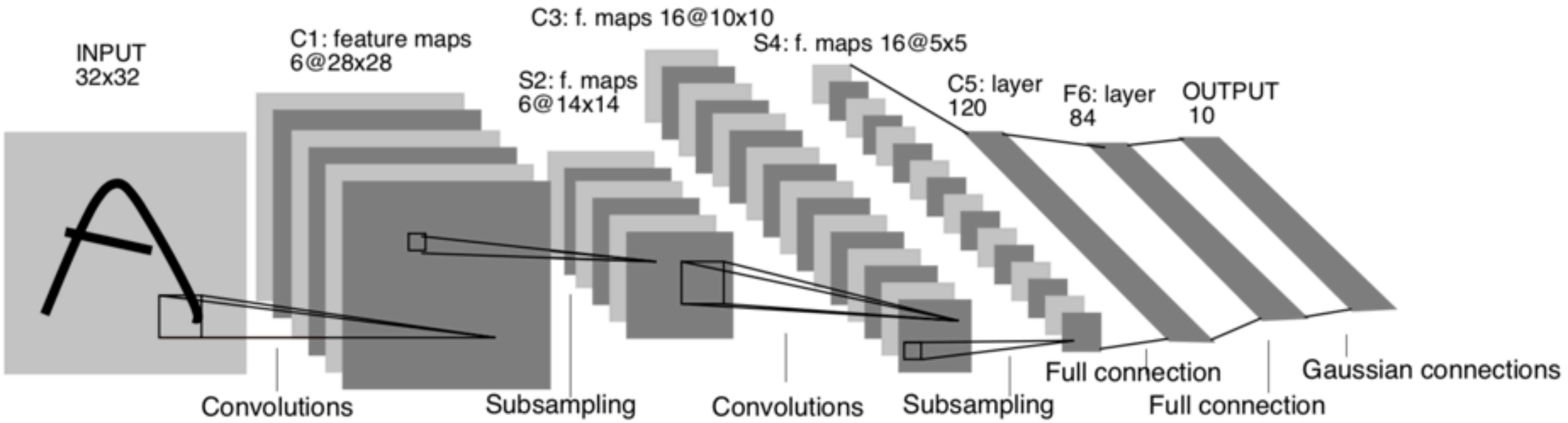



*How many convolution filters to generate C5?*




# Full connection.

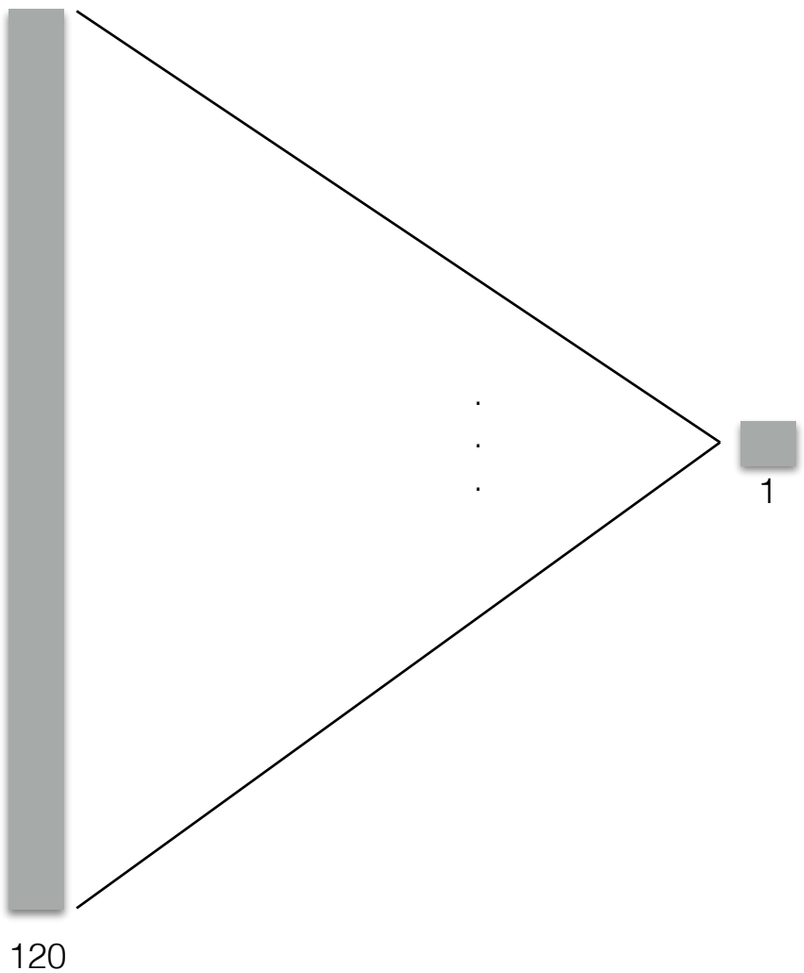
(actually just another special convolution)




Full connection.

(actually just another special convolution)

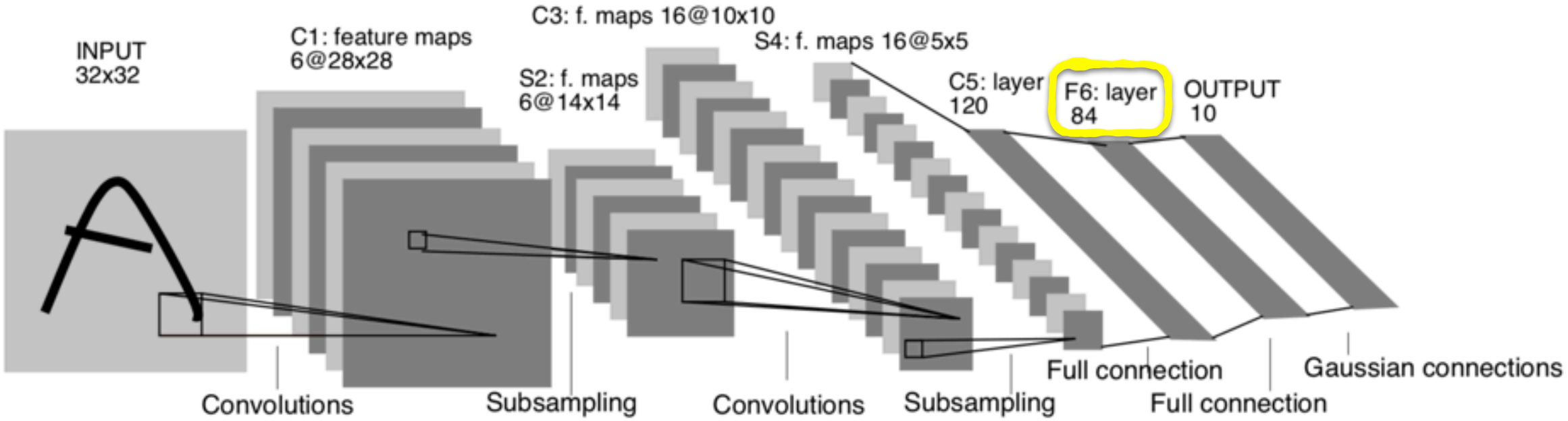
One huge perceptron!
(don't forget the bias)




amplitude scale

$$f(a) = A \cdot \tanh(Sa)$$

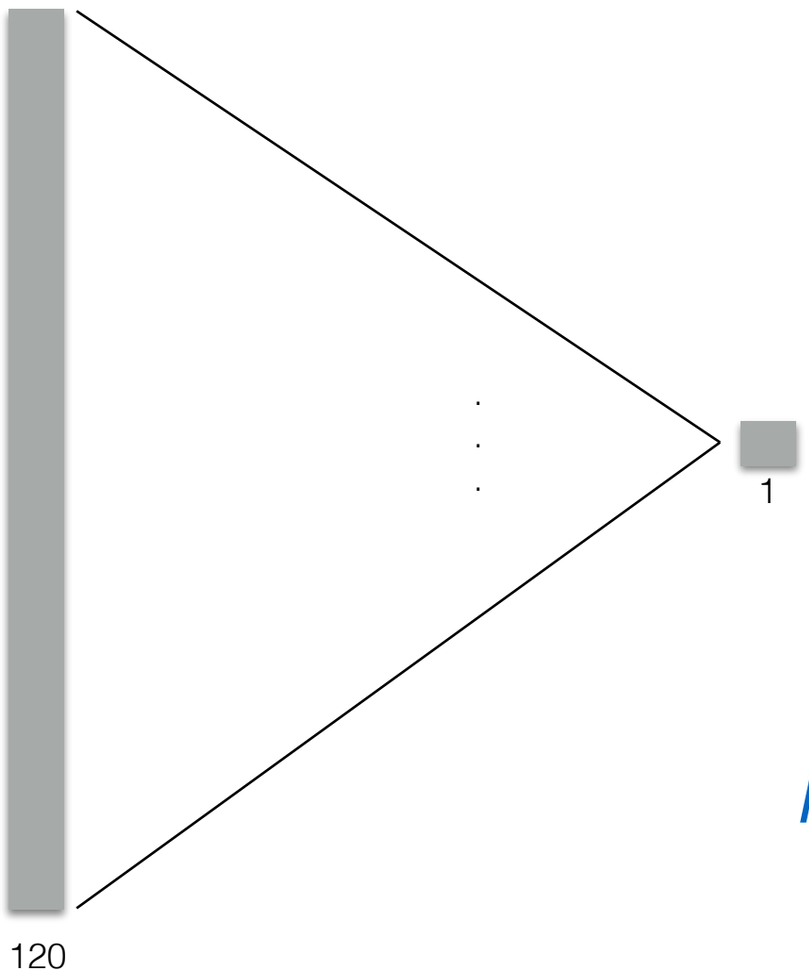
weighted sum




Full connection.

(actually just another special convolution)

One huge perceptron!
(don't forget the bias)

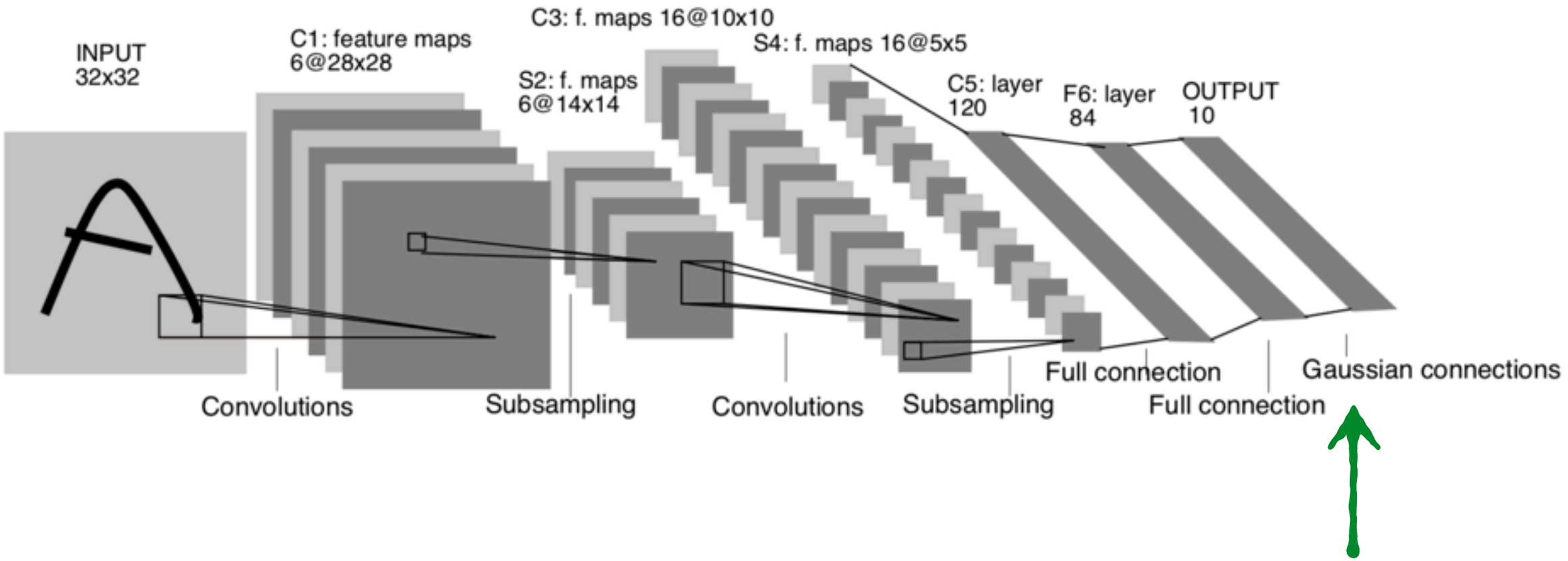



amplitude scale

$$f(a) = A \cdot \tanh(Sa)$$

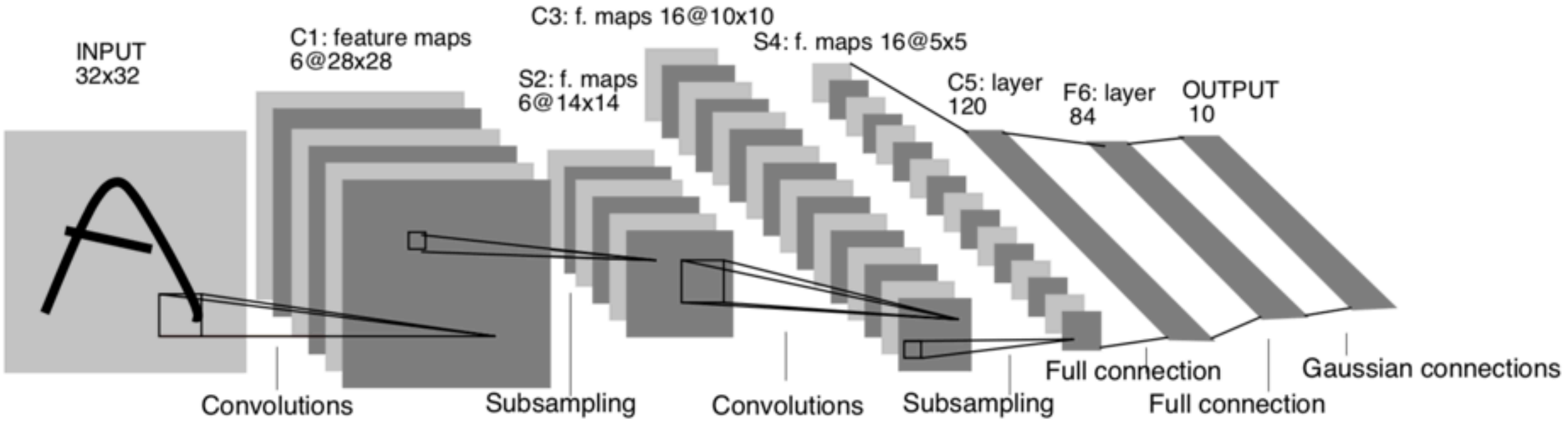
weighted sum

*How many perceptrons to generate F6?*




RBF

Radial Basis Function




# Euclidean Radial Basis Function

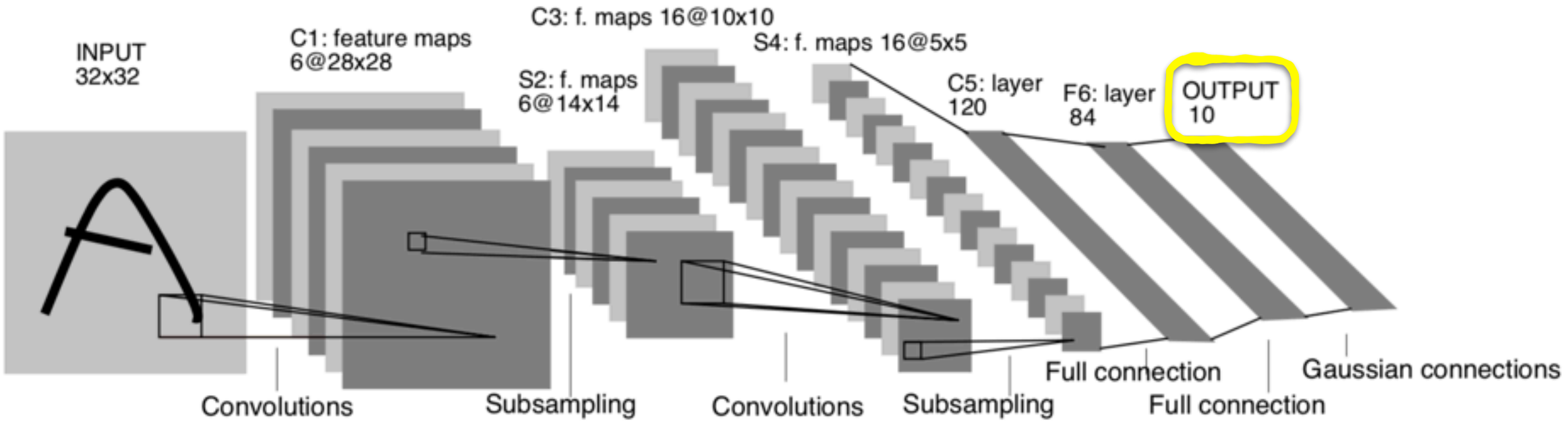
## Unnormalized negative log-likelihood of a Gaussian

$$y_i = \sum_{j}^{84} (x_j - w_{ij})^2$$

single class output

FC6

parameter
set manually
(-1, +1)




# Euclidean Radial Basis Function

## Unnormalized negative log-likelihood of a Gaussian

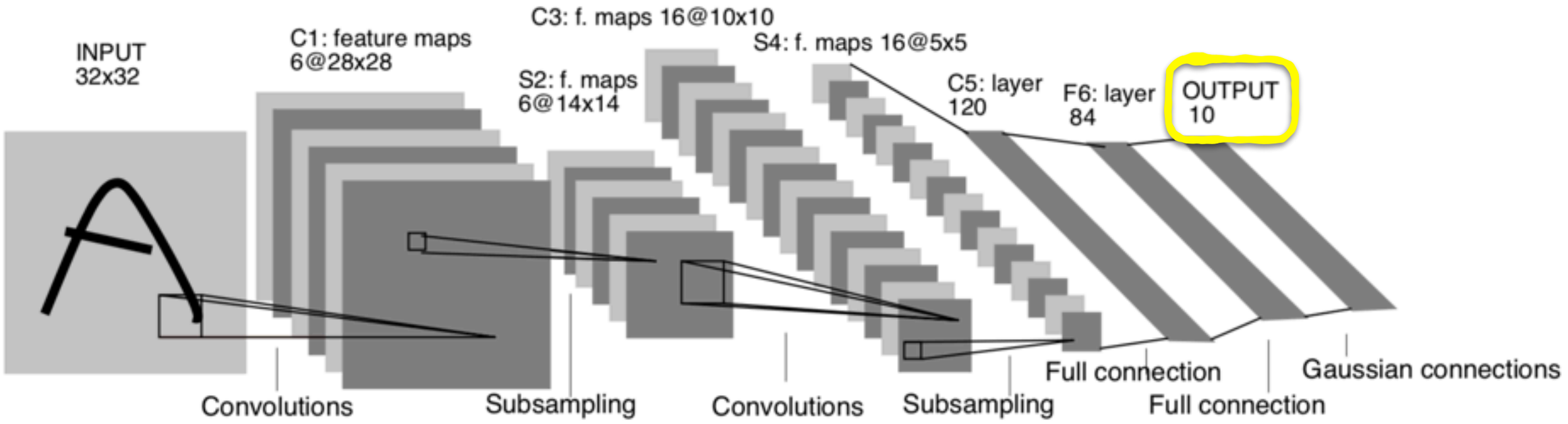
$$y_i = \sum_{j}^{84} (x_j - w_{ij})^2$$

single class output

FC6

parameter
set manually
(-1, +1)

*What is the range of i?*



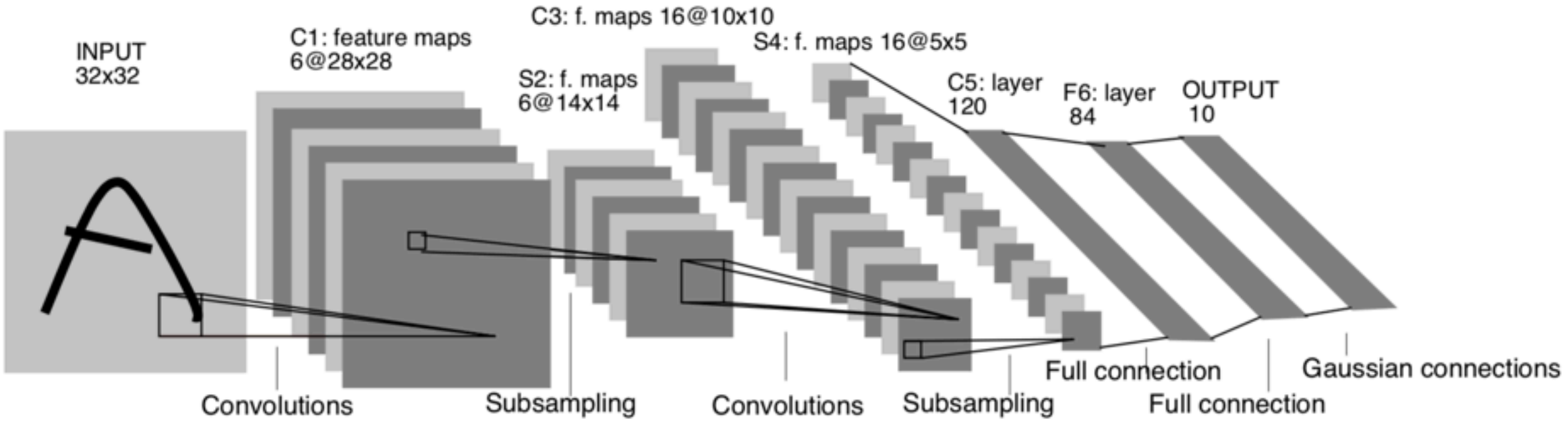

# Euclidean Radial Basis Function

## Unnormalized negative log-likelihood of a Gaussian

$$y_i = \sum_{j}^{84} (x_j - w_{ij})^2$$

- $y_i$: single class output
- $x_j$: FC6
- $w_{ij}$: parameter, set manually (-1, +1)

*What happens when x matches w?*



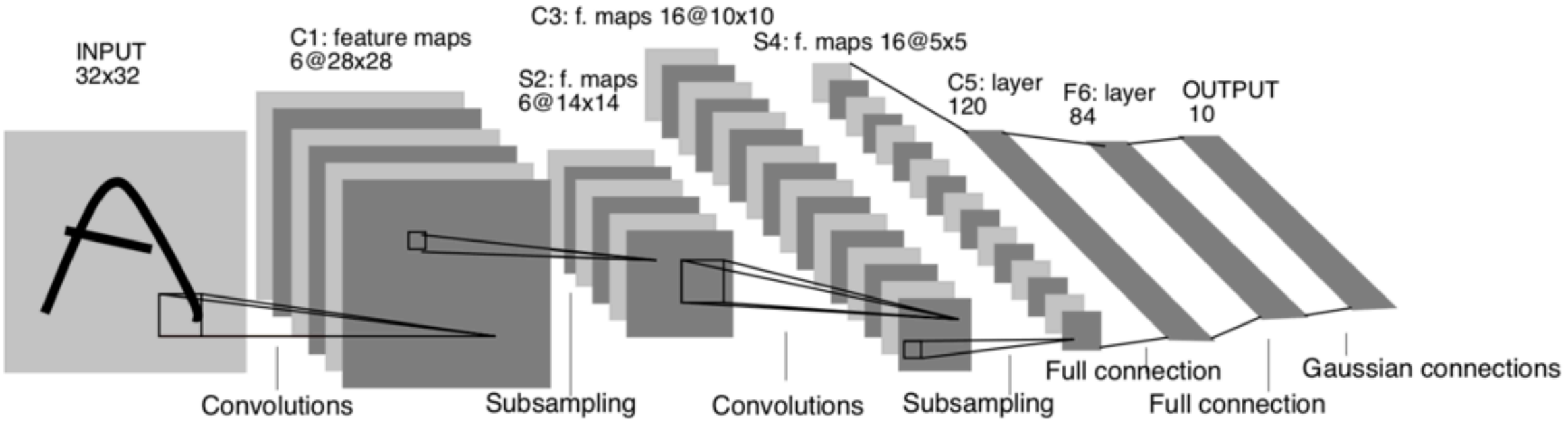

# Euclidean Radial Basis Function

## Unnormalized negative log-likelihood of a Gaussian

$$y_i = \sum_{j}^{84} (x_j - w_{ij})^2$$

- $y_i$: single class output
- $x_j$: FC6
- $w_{ij}$: parameter, set manually (-1, +1)

*What is the range of j?*




# Euclidean Radial Basis Function

## Unnormalized negative log-likelihood of a Gaussian

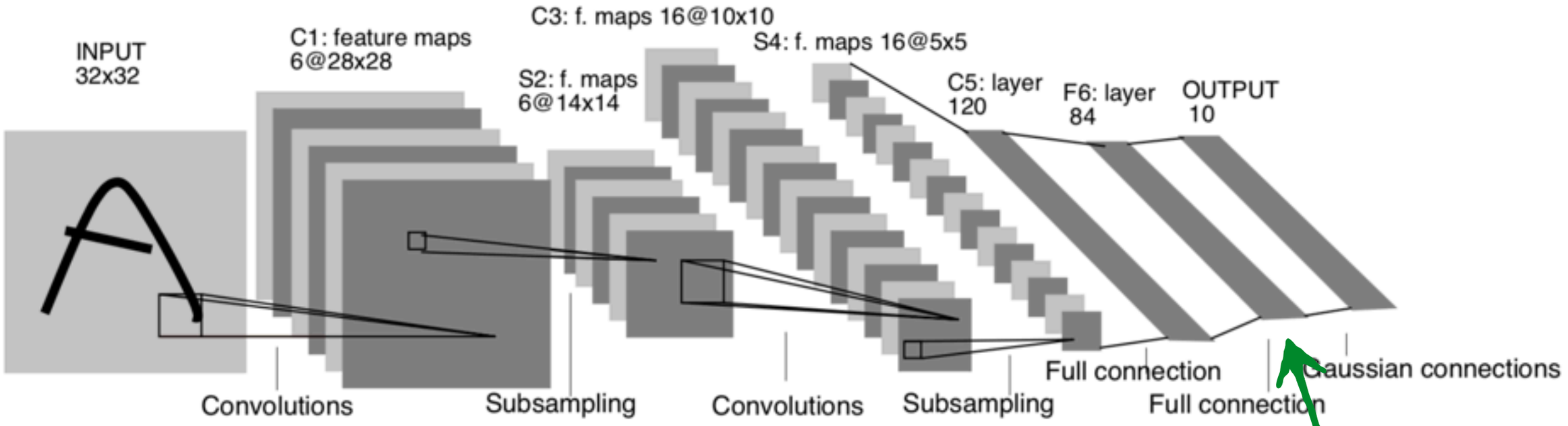
$$y_i = \sum_{j}^{84} (x_j - w_{ij})^2$$

single class output

FC6

parameter set manually (-1, +1)

*How do you set w?*




# Euclidean Radial Basis Function

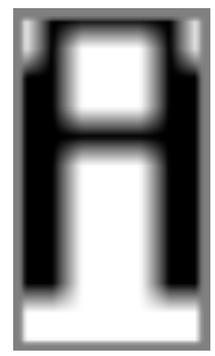
## Unnormalized negative log-likelihood of a Gaussian

$$y_i = \sum_j^{84} (x_j - w_{ij})^2$$

single class output

FC6

w is set manually to match ASCII character

$w =$ 

7

12

White: -1
Black: +1

LeNet 5 is trained to draw ASCII characters!

# Important Concepts

- Small filters capture common patterns
- Small filters can be reused and are efficient
- Multi-channel 2D Convolution
- Average pooling
- Convolution+Pooling+Non-linearity Module