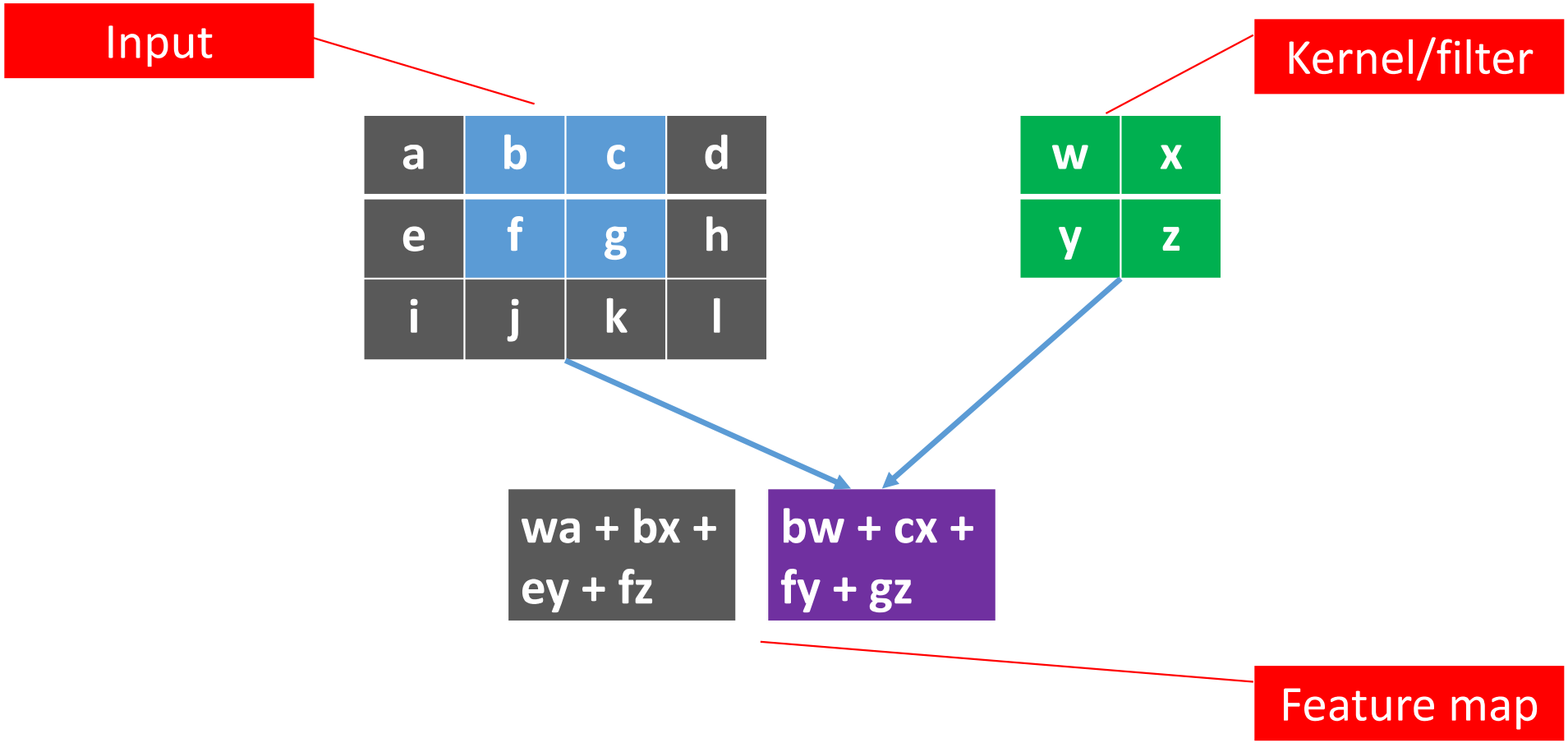
Deep Learning Basics
Lecture 6: Convolutional NN

Princeton University COS 495

Instructor: Yingyu Liang

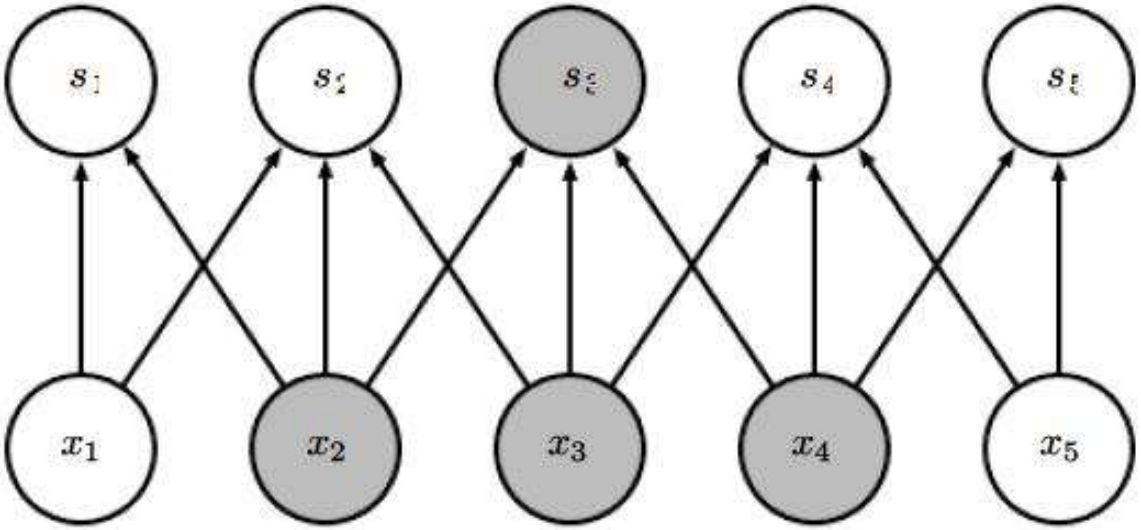
Review: convolutional layers

Convolution: two dimensional case



Convolutional layers

the same weight shared for all output nodes



m output nodes

k kernel size

n input nodes

Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Terminology

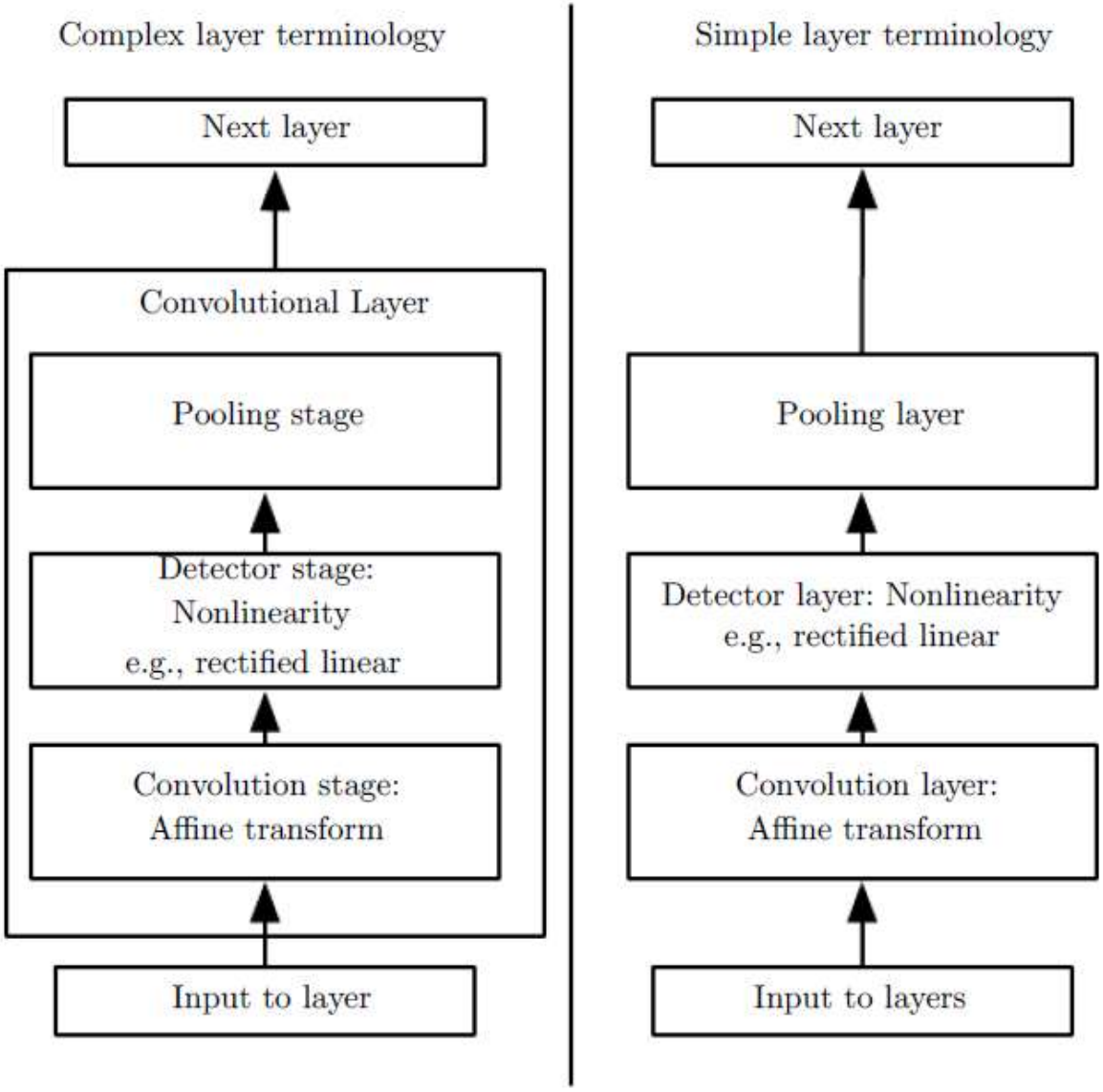


Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Case study: LeNet-5

LeNet-5

- Proposed in *"Gradient-based learning applied to document recognition", by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

LeNet-5

- Proposed in *"Gradient-based learning applied to document recognition", by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

- Apply convolution on 2D images (MNIST) and use backpropagation

LeNet-5

- Proposed in *"Gradient-based learning applied to document recognition", by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

- Apply convolution on 2D images (MNIST) and use backpropagation

- Structure: 2 convolutional layers (with pooling) + 3 fully connected layers
  - Input size: 32x32x1
  - Convolution kernel size: 5x5
  - Pooling: 2x2

LeNet-5

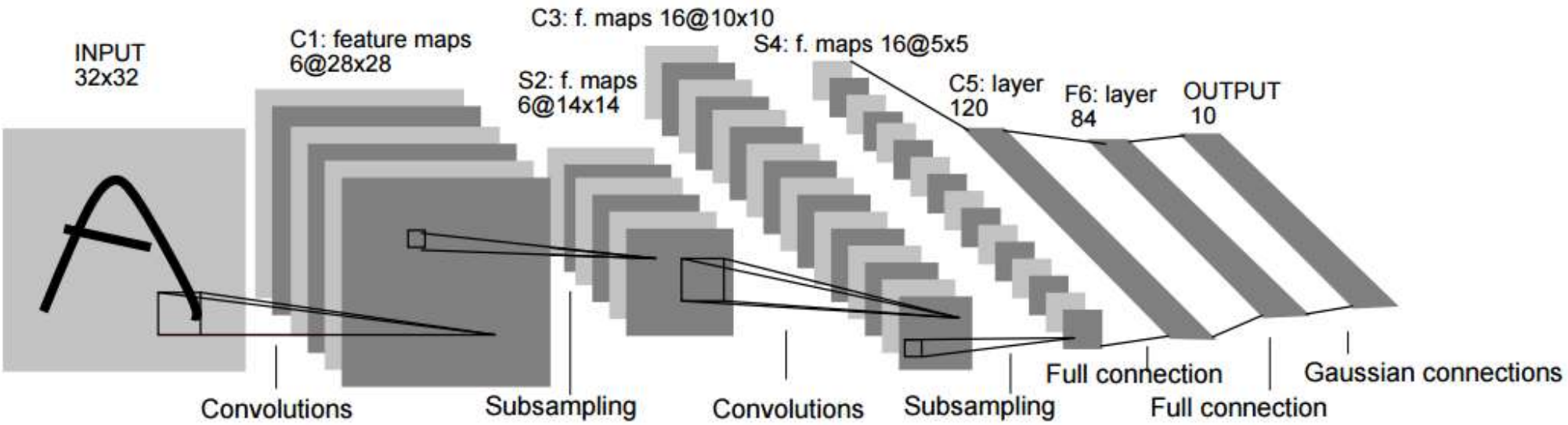


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

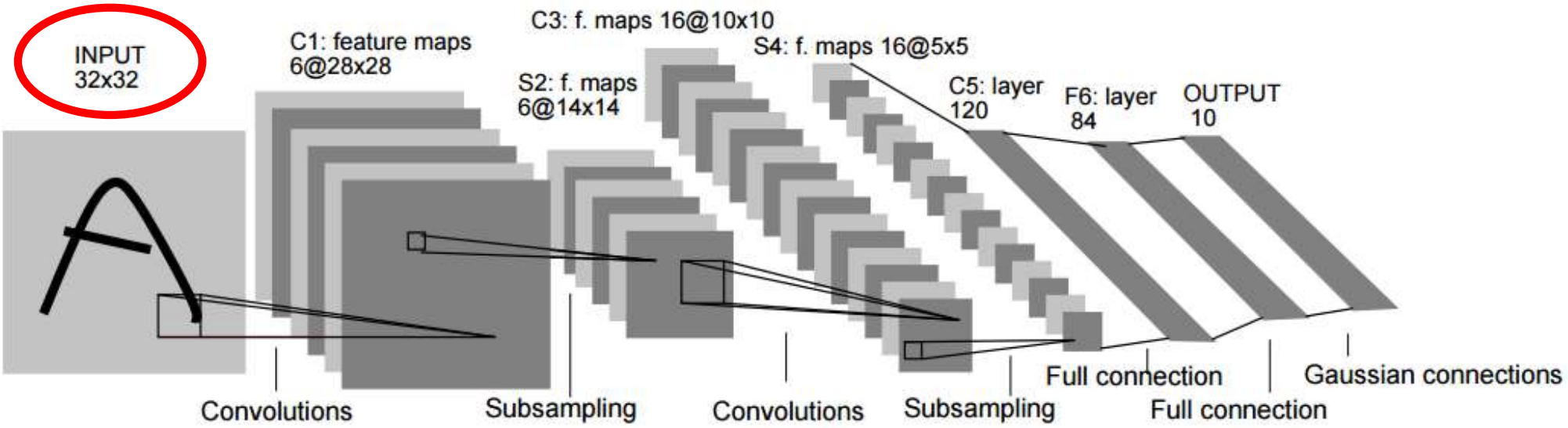


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

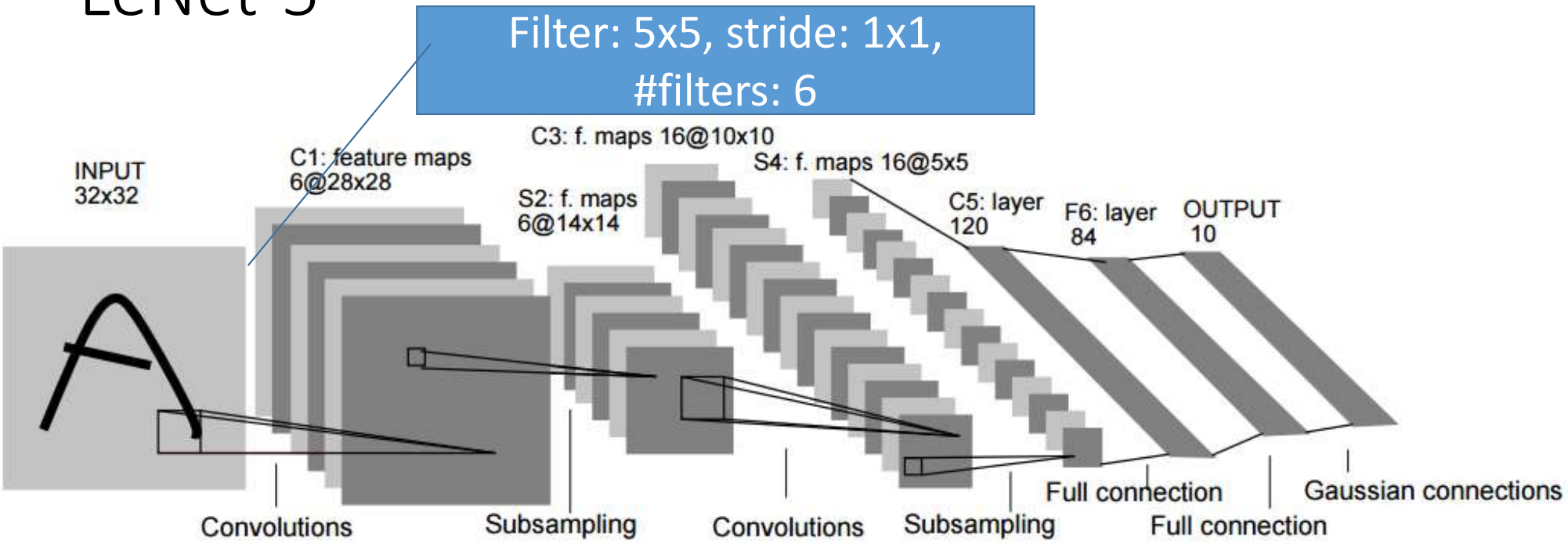


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

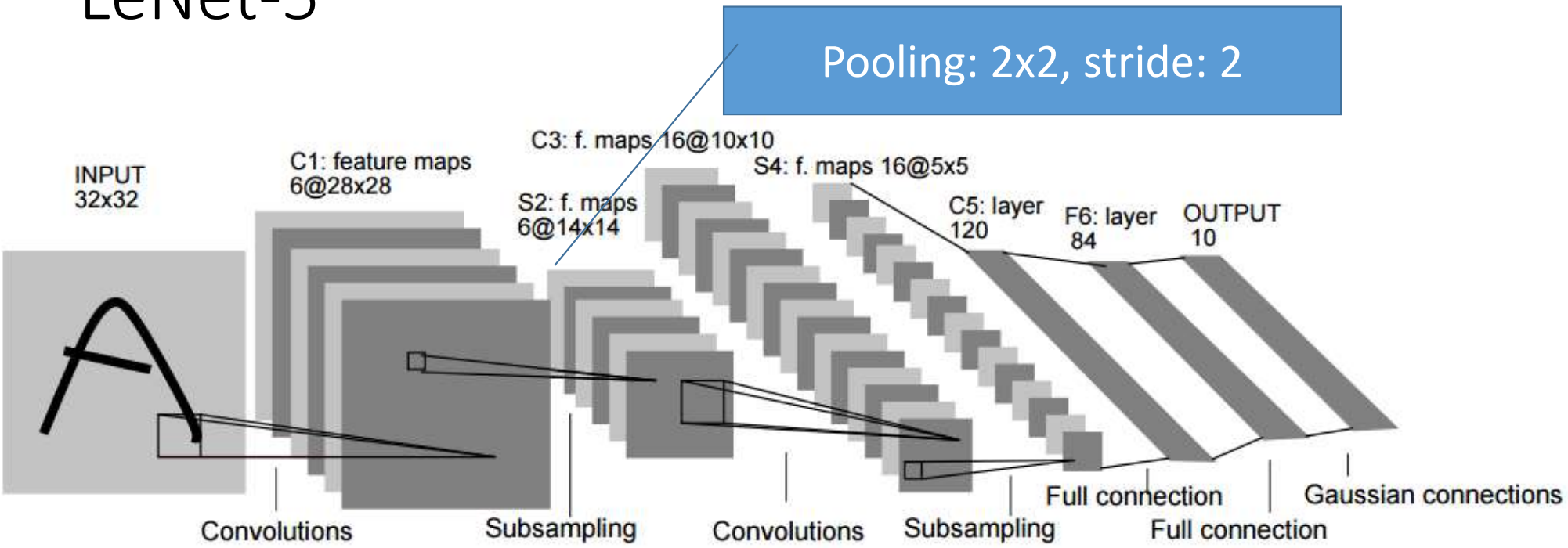


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

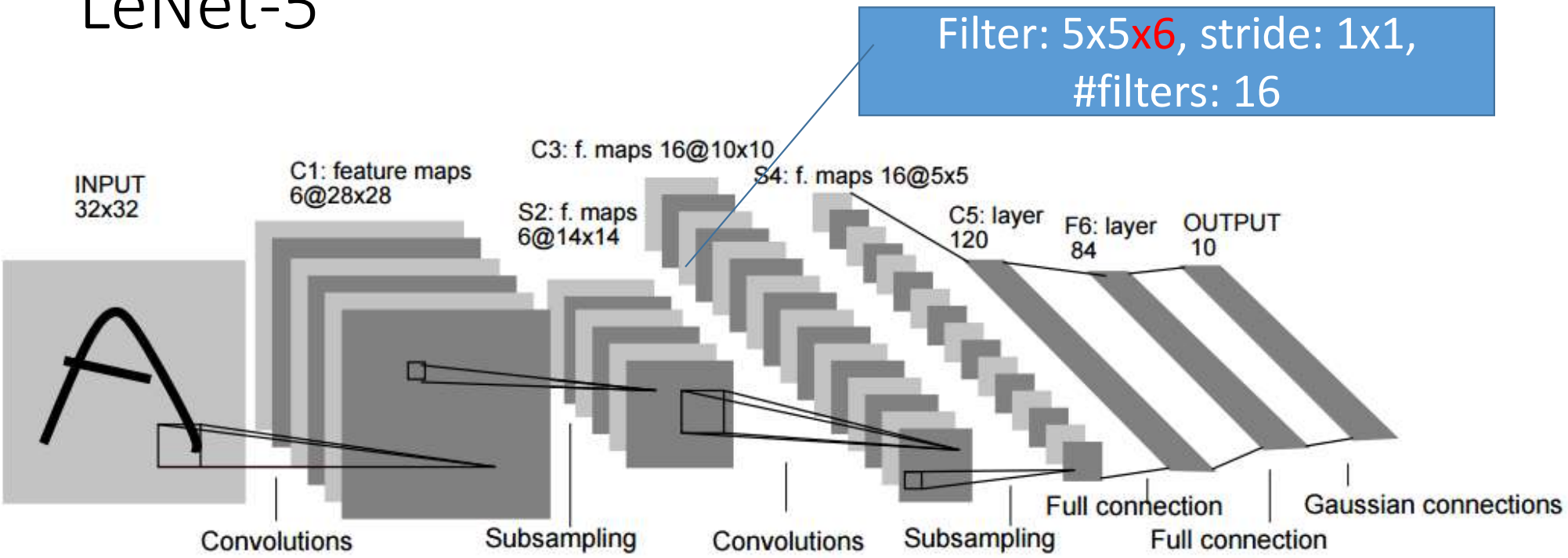


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

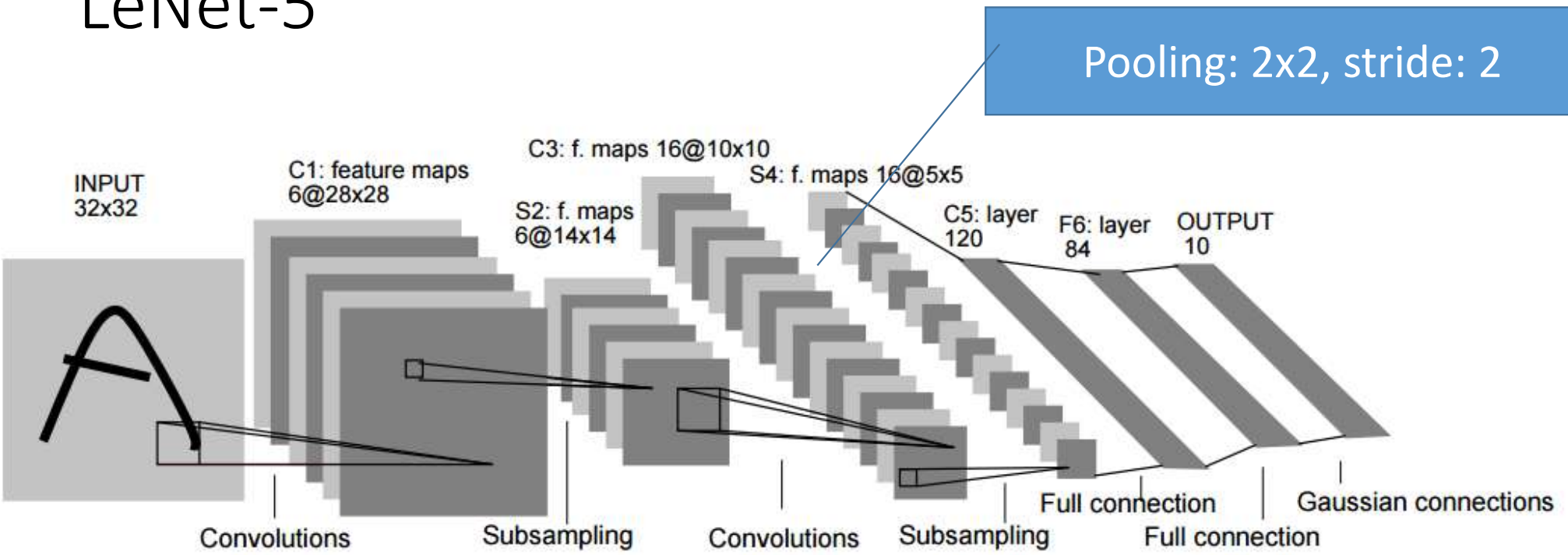


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

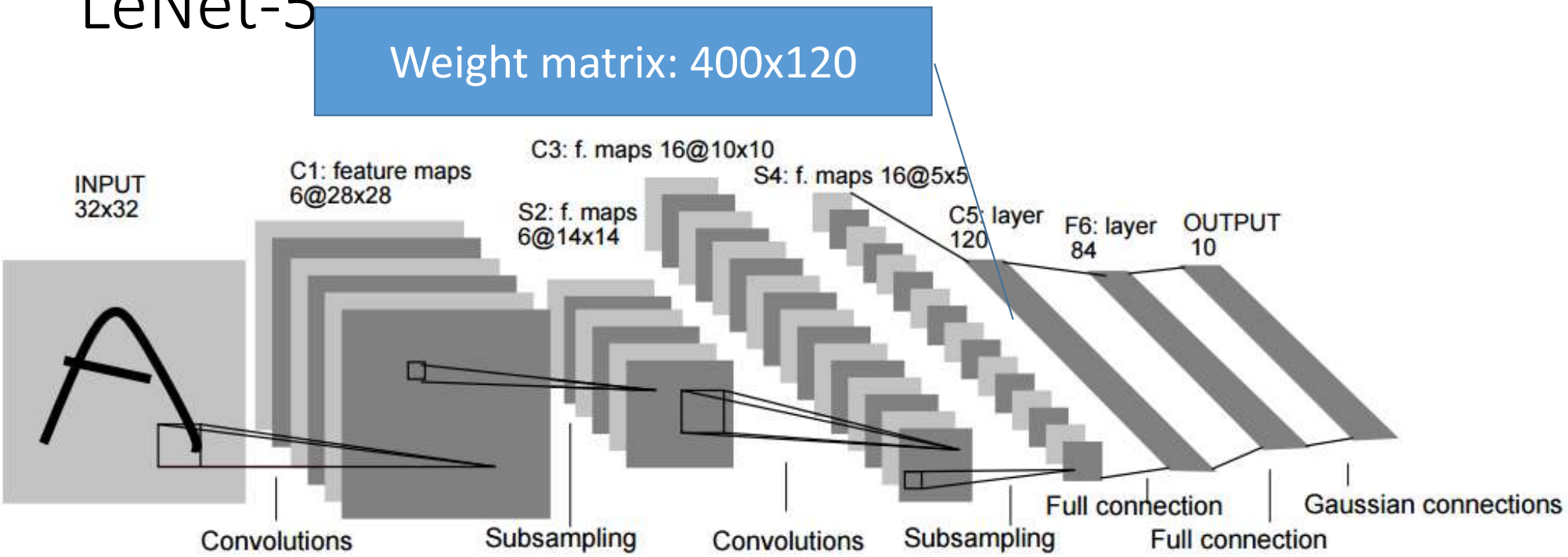


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

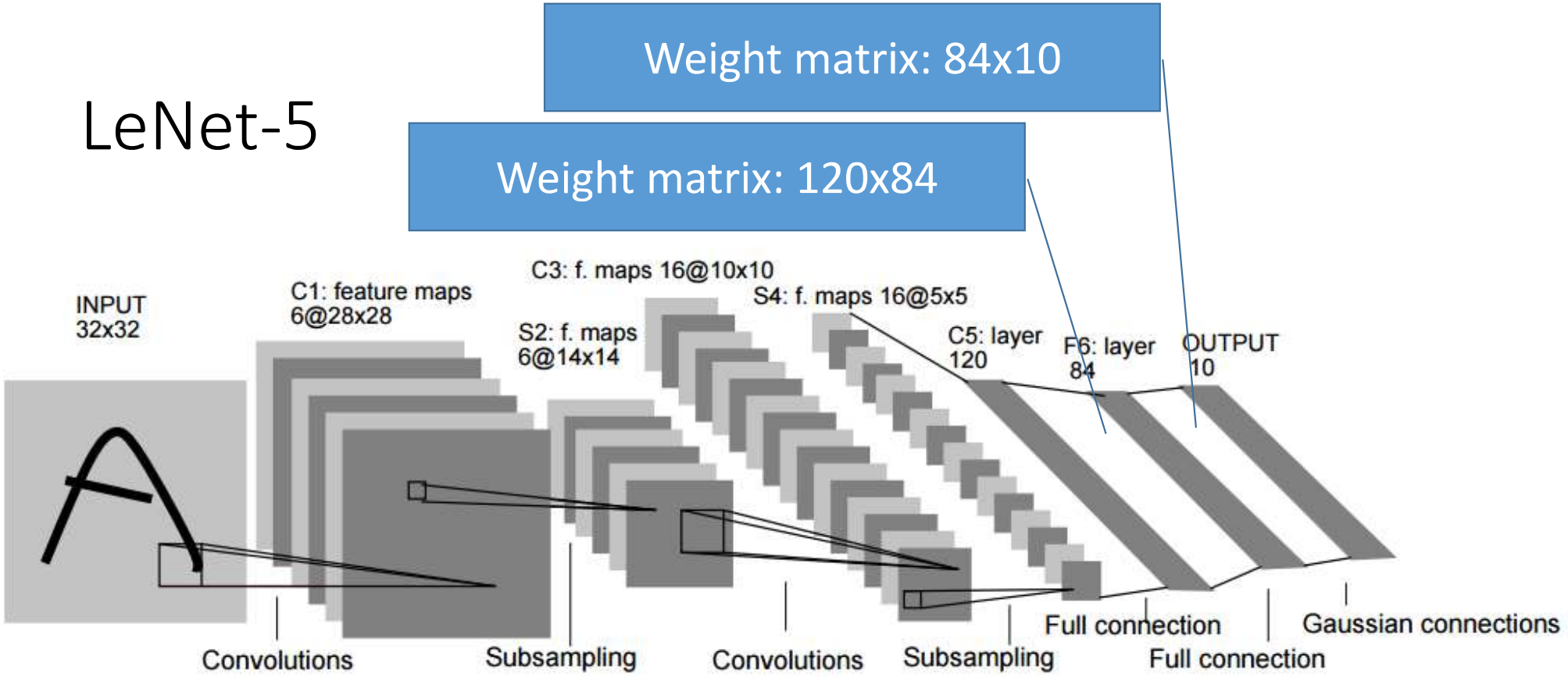
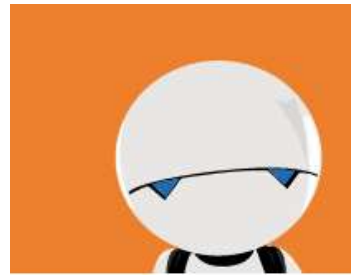


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

Software platforms for CNN

Updated in April 2016; checked more recent ones online

Platform: Marvin (marvin.is)



Marvin

A minimalist GPU-only N-dimensional ConvNet framework

Learn more | Questions?

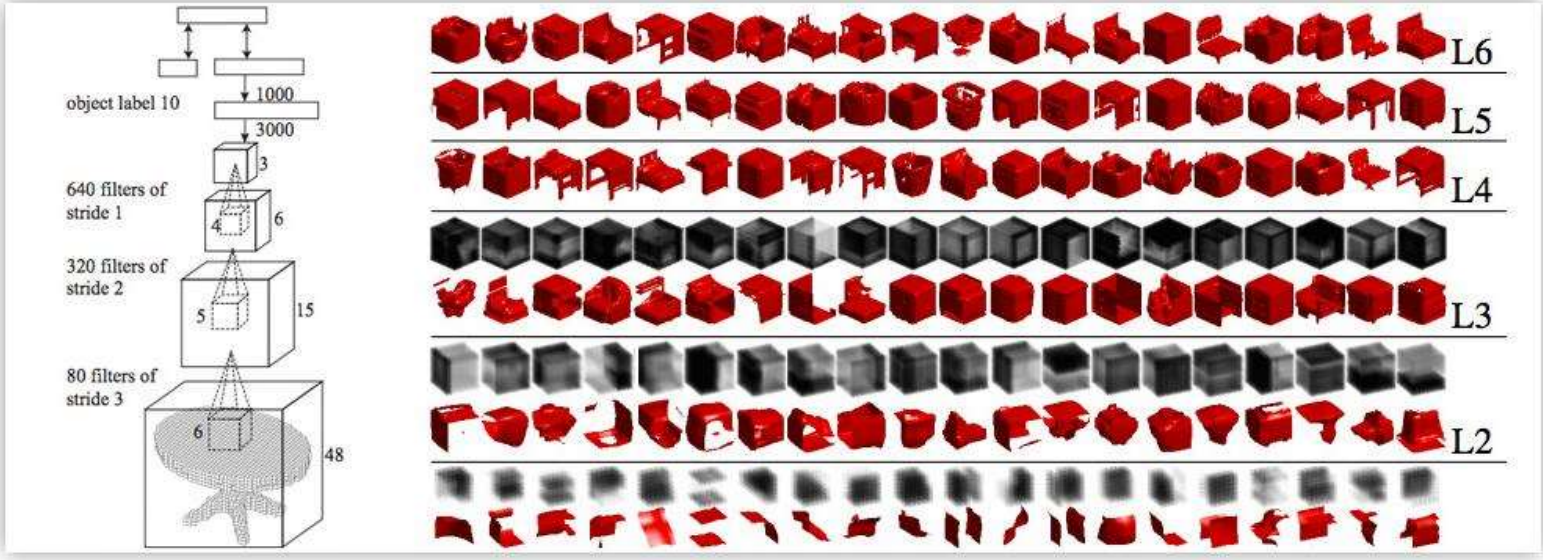
Marvin thinks, therefore Marvin is.

Never before has it been so easy to learn so deeply. Marvin was born to be hacked, relying on few dependencies and basic C++. All code lives in two files (`marvin.hpp` and `marvin.cu`) and all numbers take up two bytes (FP16). Win friends and influence people in four easy steps:

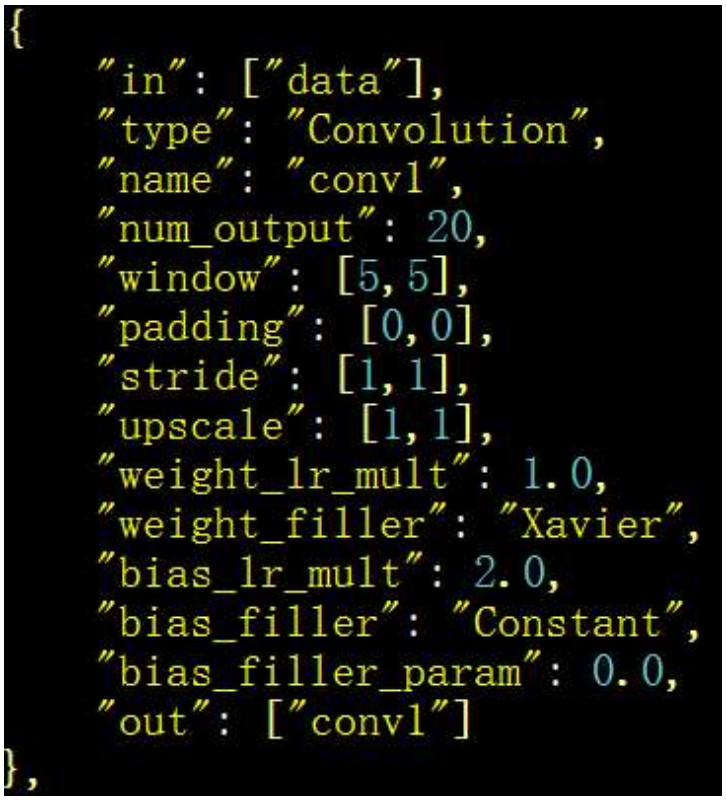
Platform: Marvin by

PRINCETON VISION GROUP

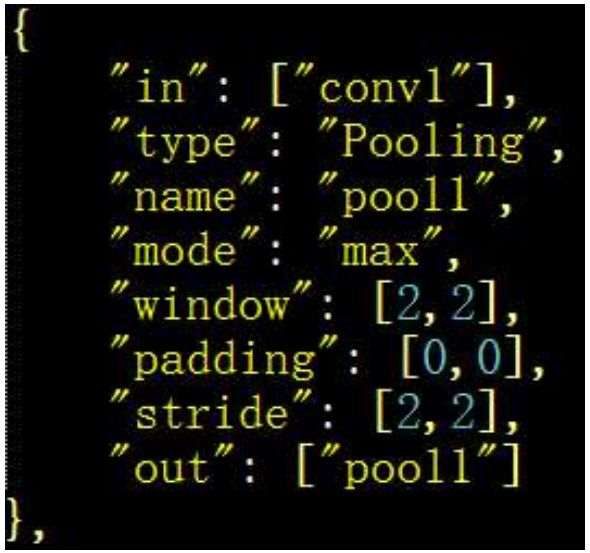
Home Research Publications Code Courses People



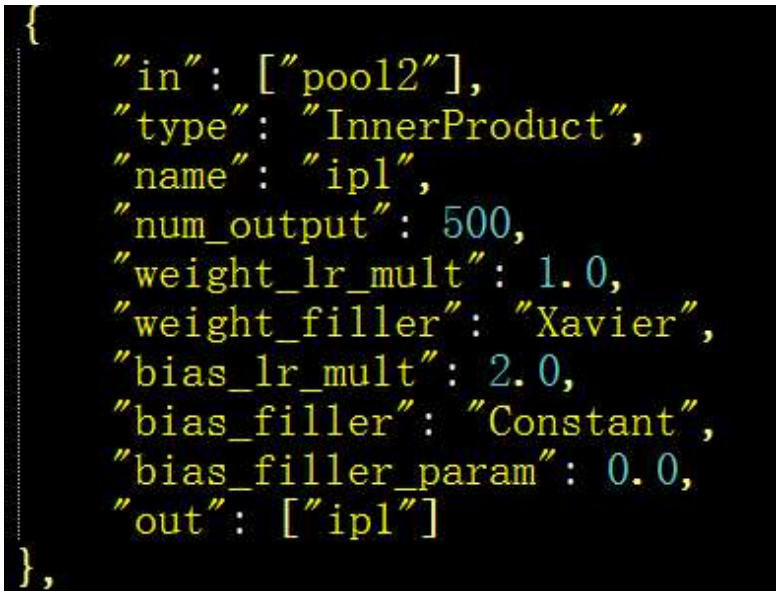
LeNet in Marvin: convolutional layer



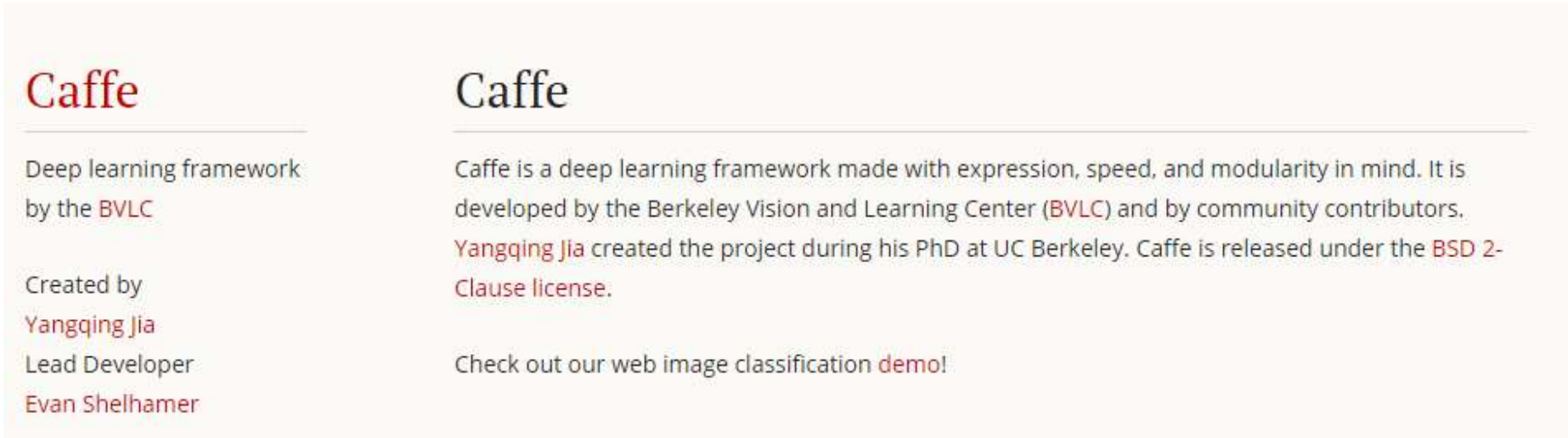
LeNet in Marvin: pooling layer



LeNet in Marvin: fully connected layer



Platform: Caffe (caffe.berkeleyvision.org)



LeNet in Caffe

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
```

Platform: Tensorflow (tensorflow.org)

Platform: Tensorflow (tensorflow.org)

```
conv = tf.nn.conv2d(data,
                    conv1_weights,
                    strides=[1, 1, 1, 1],
                    padding='SAME')
# Bias and rectified linear non-linearity.
relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
# Max pooling. The kernel size spec {ksize} also follows the layout of
# the data. Here we have a pooling window of 2, and a stride of 2.
pool = tf.nn.max_pool(relu,
                      ksize=[1, 2, 2, 1],
                      strides=[1, 2, 2, 1],
                      padding='SAME')
```

Platform: Tensorflow (tensorflow.org)

```
# Fully connected layer. Note that the '+' operation automatically
# broadcasts the biases.
hidden = tf.nn.relu(tf.matmul(reshape, fc1_weights) + fc1_biases)
# Add a 50% dropout during training only. Dropout also scales
# activations such that no rescaling is needed at evaluation time.
if train:
  hidden = tf.nn.dropout(hidden, 0.5, seed=SEED)
```

Others

- **Theano** – CPU/GPU symbolic expression compiler in python (from MILA lab at University of Montreal)
- **Torch** – provides a Matlab-like environment for state-of-the-art machine learning algorithms in lua
- **Lasagne** - Lasagne is a lightweight library to build and train neural networks in Theano

- See: http://deeplearning.net/software_links/

Optimization: momentum

Basic algorithms

- Minimize the (regularized) empirical loss

$$\hat{L}_R(\theta) = \frac{1}{n}\sum_{t=1}^{n} l(\theta, x_t, y_t) + R(\theta)$$

  where the hypothesis is parametrized by θ

- Gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \nabla \hat{L}_R(\theta_t)$$

Mini-batch stochastic gradient descent

- Instead of one data point, work with a small batch of b points

$$(x_{tb+1}, y_{tb+1}), \ldots, (x_{tb+b}, y_{tb+b})$$

- Update rule

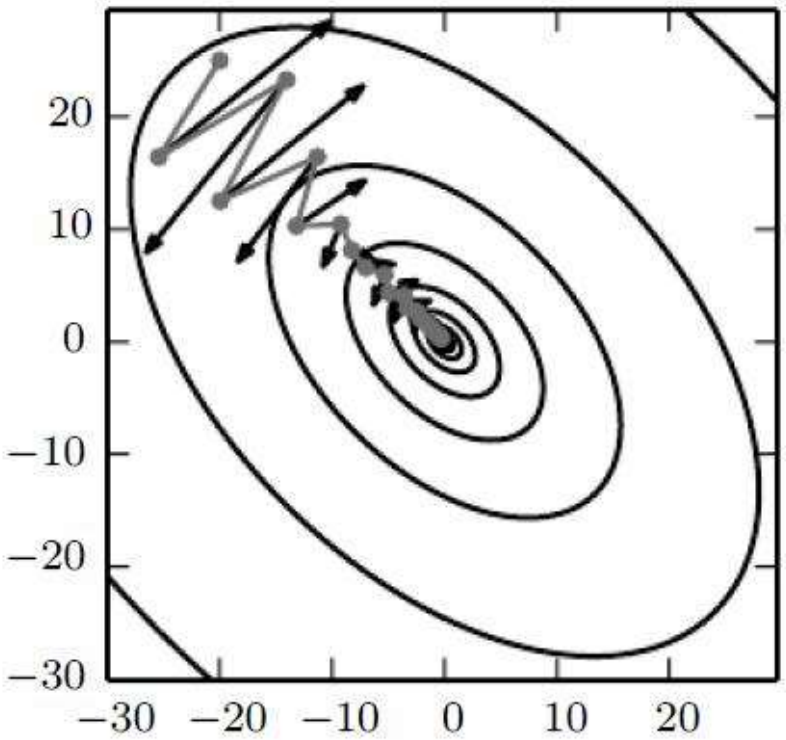
$$\theta_{t+1} = \theta_t - \eta_t \nabla \left(\frac{1}{b} \sum_{1 \le i \le b} l(\theta_t, x_{tb+i}, y_{tb+i}) + R(\theta_t) \right)$$

Momentum

- Drawback of SGD: can be slow when gradient is small

- Observation: when the gradient is consistent across consecutive steps, can take larger steps
- Metaphor: rolling marble ball on gentle slope

Momentum



Contour: loss function
Path: SGD with momentum
Arrow: stochastic gradient

Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Momentum

- work with a small batch of b points

$$(x_{tb+1}, y_{tb+1}), \ldots, (x_{tb+b}, y_{tb+b})$$

- Keep a momentum variable v_t, and set a decay rate α
- Update rule

$$v_t = \alpha v_{t-1} - \eta_t \nabla \left(\frac{1}{b} \sum_{1 \le i \le b} l(\theta_t, x_{tb+i}, y_{tb+i}) + R(\theta_t) \right)$$

$$\theta_{t+1} = \theta_t + v_t$$

Momentum

- Keep a momentum variable v_t, and set a decay rate α
- Update rule

$$v_t = \alpha v_{t-1} - \eta_t \nabla \left(\frac{1}{b} \sum_{1 \le i \le b} l(\theta_t, x_{tb+i}, y_{tb+i}) + R(\theta_t) \right)$$

$$\theta_{t+1} = \theta_t + v_t$$

- Practical guide: α is set to 0.5 until the initial learning stabilizes and then is increased to 0.9 or higher.