Georgia Tech
College of Computing

CX4240 Spring 2026

Convolution Neural Networks

Bo Dai
School of CSE, Georgia Tech
bodai@cc.gatech.edu

Based on Fei-Fei Li & Ehsan Adeli's Slides

Multi-Layer Perceptron

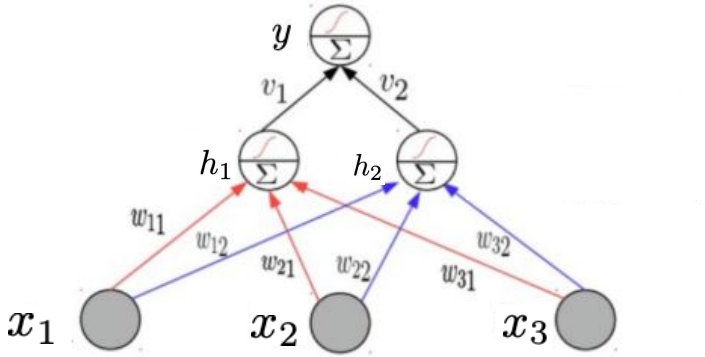
$$y = o(V g(W x))$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$



Multi-Layer Perceptron

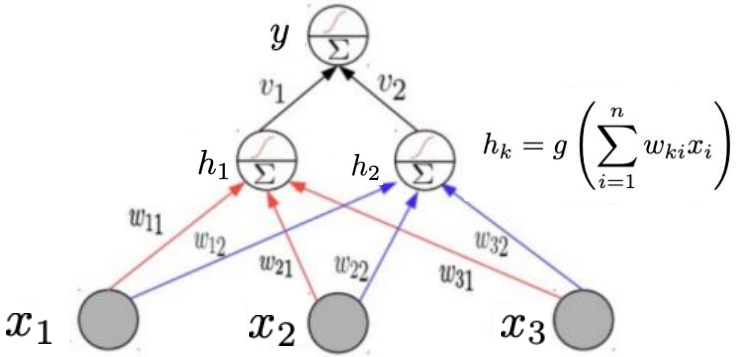
$$y = o(V g(W x))$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$



Multi-Layer Perceptron

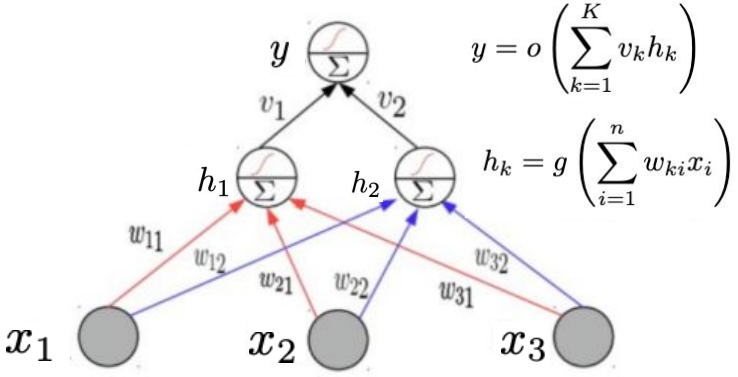
$$y = o(V g(W x))$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$



Multi-Layer Perceptron

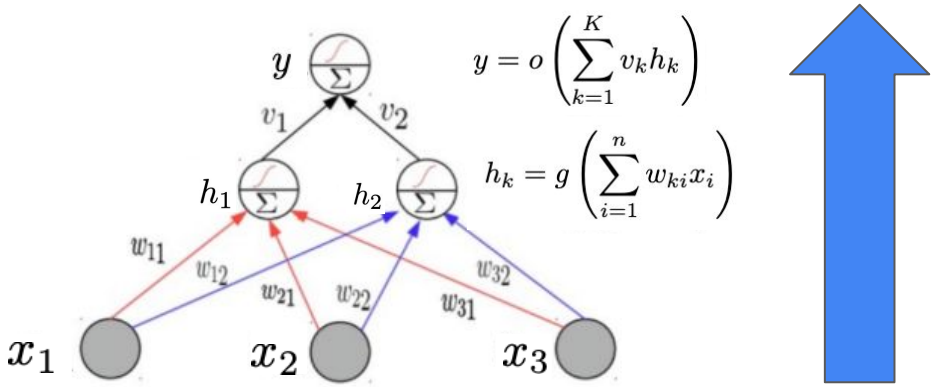
$$y = o(V g(W x))$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

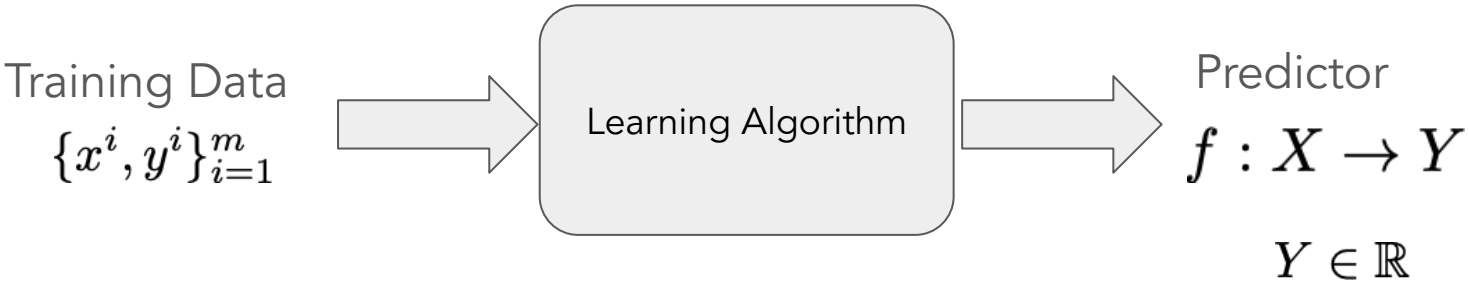
$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$x = [x_1, x_2, x_3]^\top$$



Forward Pass

Regression Algorithms

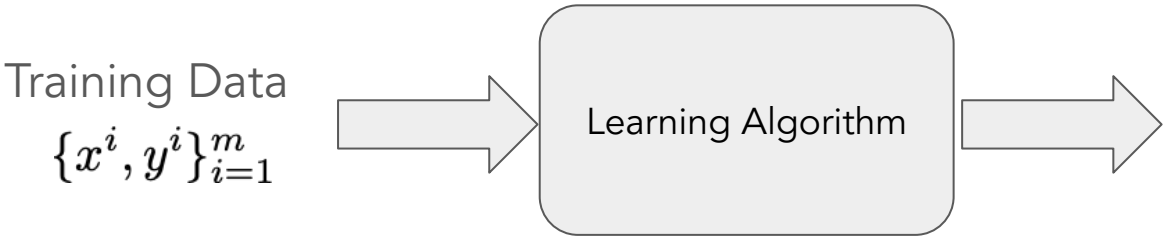


Linear Regression Pipeline

1. Build probabilistic models:
   Gaussian Distribution + Neural Network
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

$$y = o(V g(W x))$$

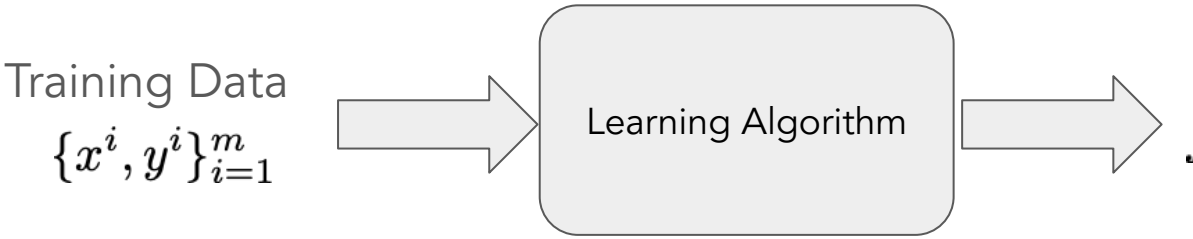
Binary Classification Algorithms



Binary Logistic Regression Pipeline

1. Build probabilistic models:
   Bernoulli Distribution + Neural Network $y = o(Vg(Wx))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Multiclass Logistic Regression Algorithms



Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Multiclass Logistic Regression Pipeline

1. Build probabilistic models:
   Categorical Distribution + Neural Network $y = o(Vg(Wx))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Select Optimizer

$$L(\theta) = \sum_{i=1}^{m} \ell(x^i, y^i, \theta) + \lambda\Omega(\theta)$$

$$\theta = [V, W]$$

$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

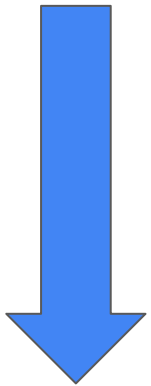
$$\ell(x^i, y^i, \theta) = -y^i \log \sigma(o(Vg(Wx^i)))$$
$$-(1 - y^i) \log(1 - \sigma(o(V_j g(Wx^i))))$$

$$\ell(x^i, y^i, \theta) = -\sum_{j=1}^{k} y^i \log \frac{\exp(o(V_j g(Wx^i)))}{\sum_{c=1}^{k} \exp(o(V_c g(Wx^i)))}$$

- (Stochastic) Gradient Descent

Backpropagation: Chain Rule on Neural Network

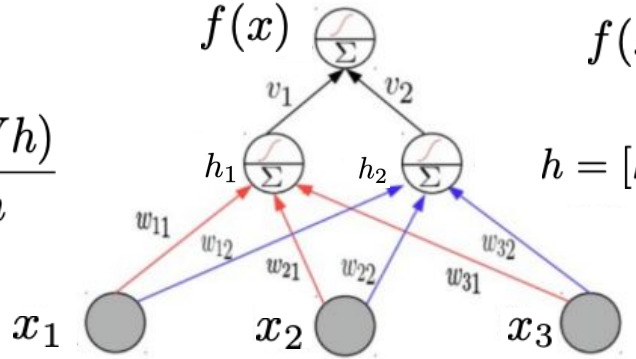
$$\ell(x^i, y^i, \theta) = (f(x^i, V, W) - y^i)^2$$



$$\frac{\partial \ell(x^i, y^i, \theta)}{\partial f}$$

$$\frac{\partial o(Vh)}{\partial V} \quad \frac{\partial o(Vh)}{\partial h}$$

$$\frac{\partial h}{\partial W}$$



$$f(x) = o(Vh)$$

$$h = [h_1, h_2]^\top = g(Wx)$$

Backward Pass

(Stochastic) Gradient Descent

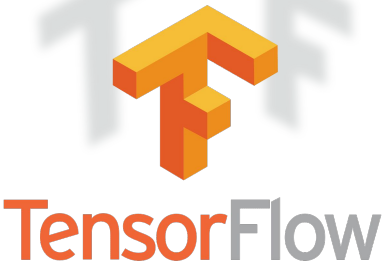
- Initialize parameter θ^0
- Sample $\{x^i, y^i\}_{i=1}^B$
- Do $$\theta^{t+1} \leftarrow \theta^t - \eta \sum_{i=1}^{B} \nabla_\theta \ell(x^i, y^i, \theta^t) - (\lambda \nabla \Omega(\theta^t))$$

Auto-differentiation Packages

PyTorch

JAX

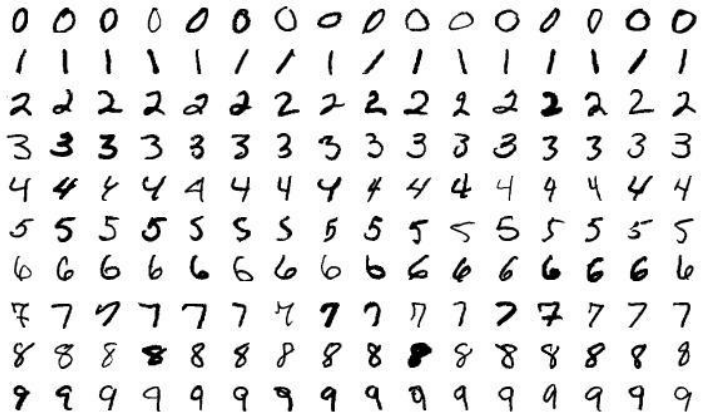
Tensorflow



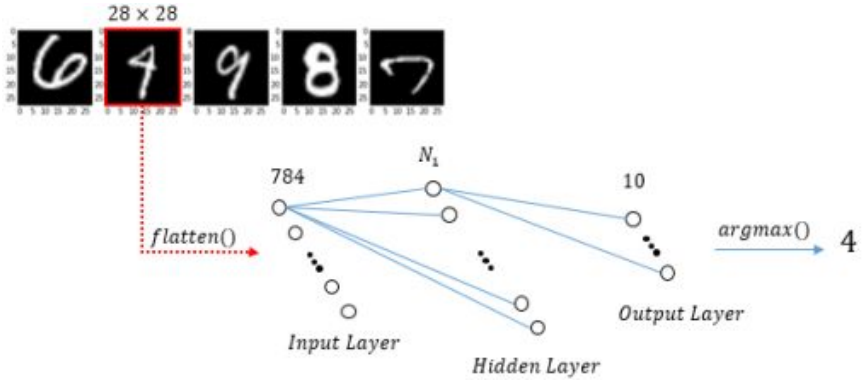
Multiclass Logistic Regression Algorithms with CNN

Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Revisit MLP for MNIST

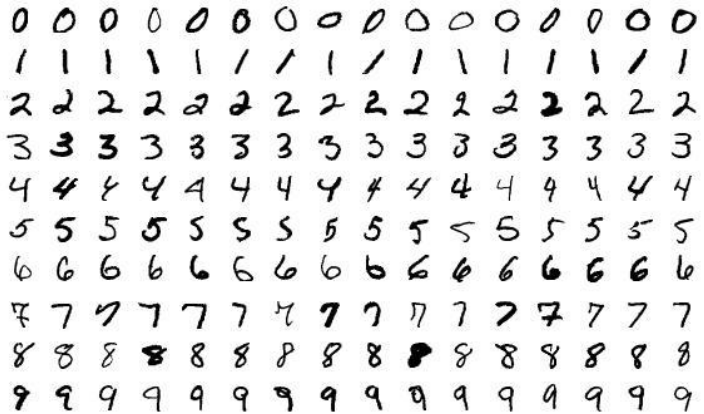


MNIST hand-written character recognition

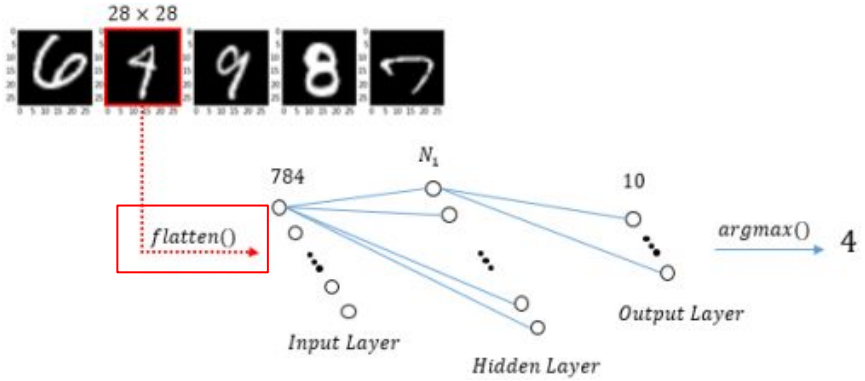


- 60,000 images
- 28x28 pixels = 784
- Grayscale, from 0 to 255 → Converted to [0,1]

Revisit MLP for MNIST

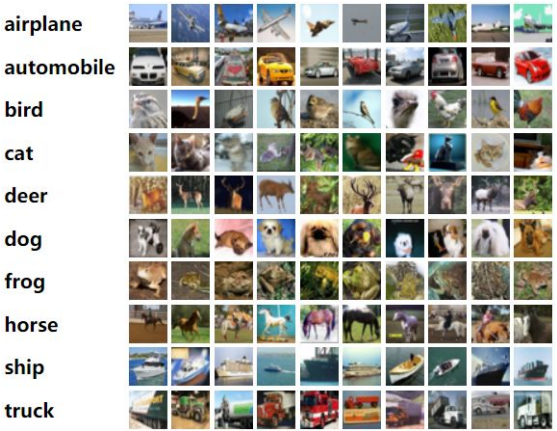


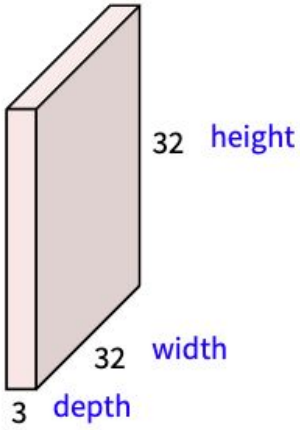
MNIST hand-written character recognition

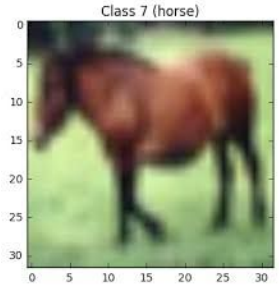


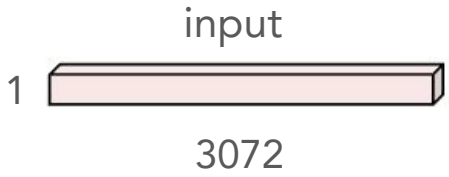
- 60,000 images
- 28x28 pixels = 784
- Grayscale, from 0 to 255 → Converted to [0,1]

MLP for CIFAR10



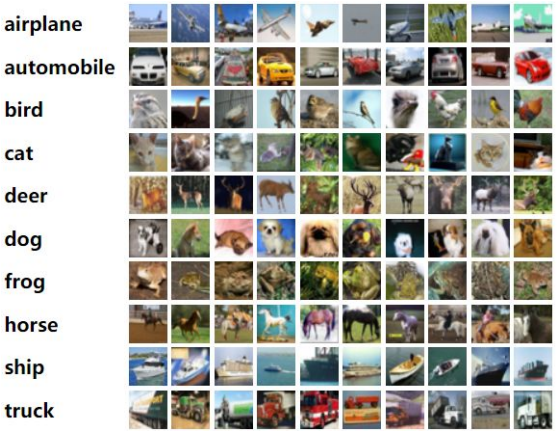


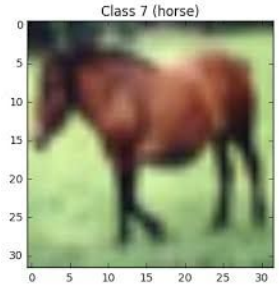


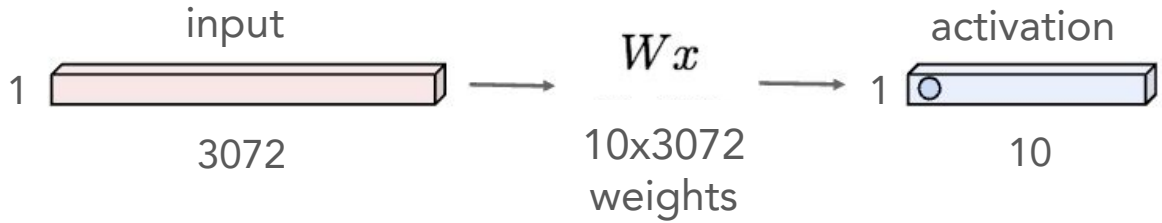


32x32x3 image → stretch to 3072x1

MLP: CIFAR10

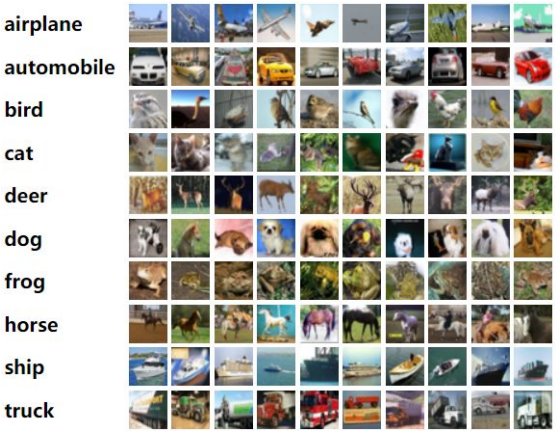


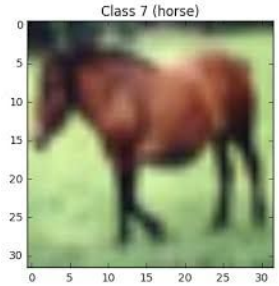




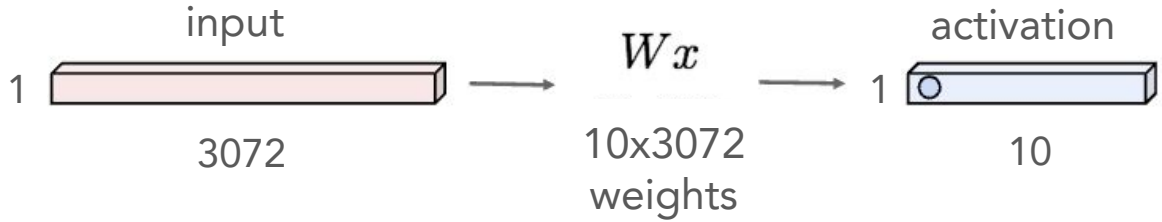
32x32x3 image → stretch to 3072x1

MLP: CIFAR10



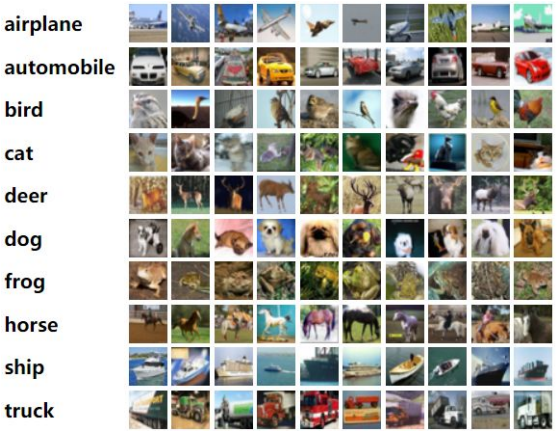


ruin the spatial structure

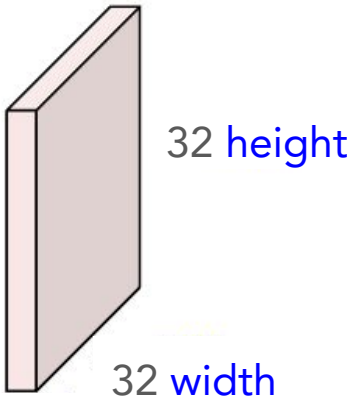


32x32x3 image → stretch to 3072x1

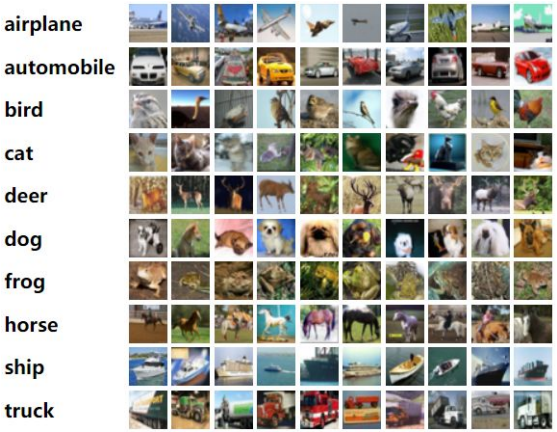
Convolution Layer



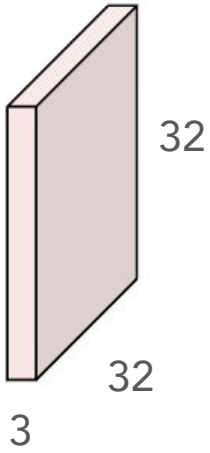
32x32x3 image → preserve spatial structure



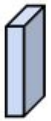
Convolution Layer



32x32x3 image

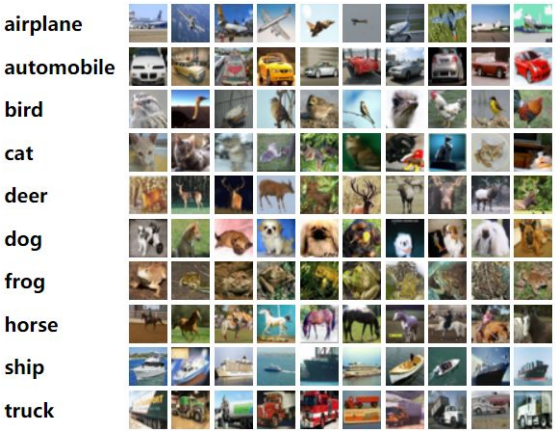


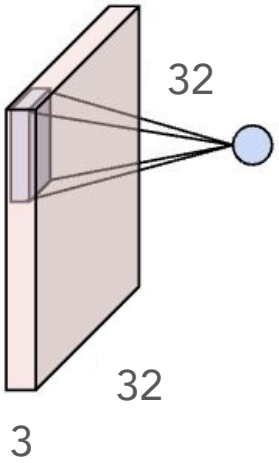
5x5x3 filter



Convolve the filter with the image, i.e., “slide over the image spatially, computing dot product”

Convolution Layer



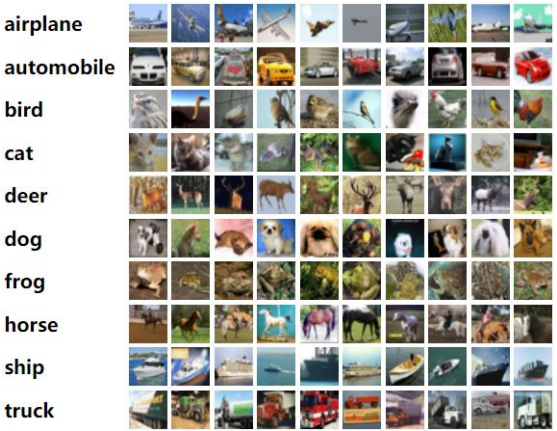


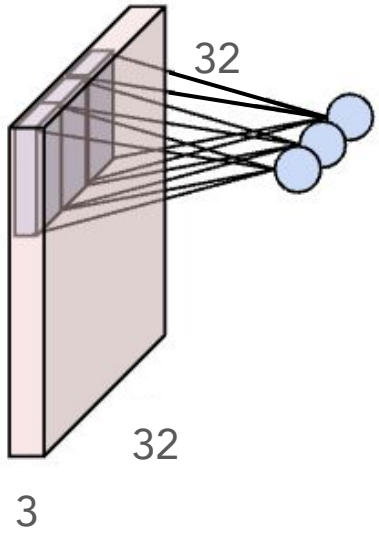
32x32x3 image
5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

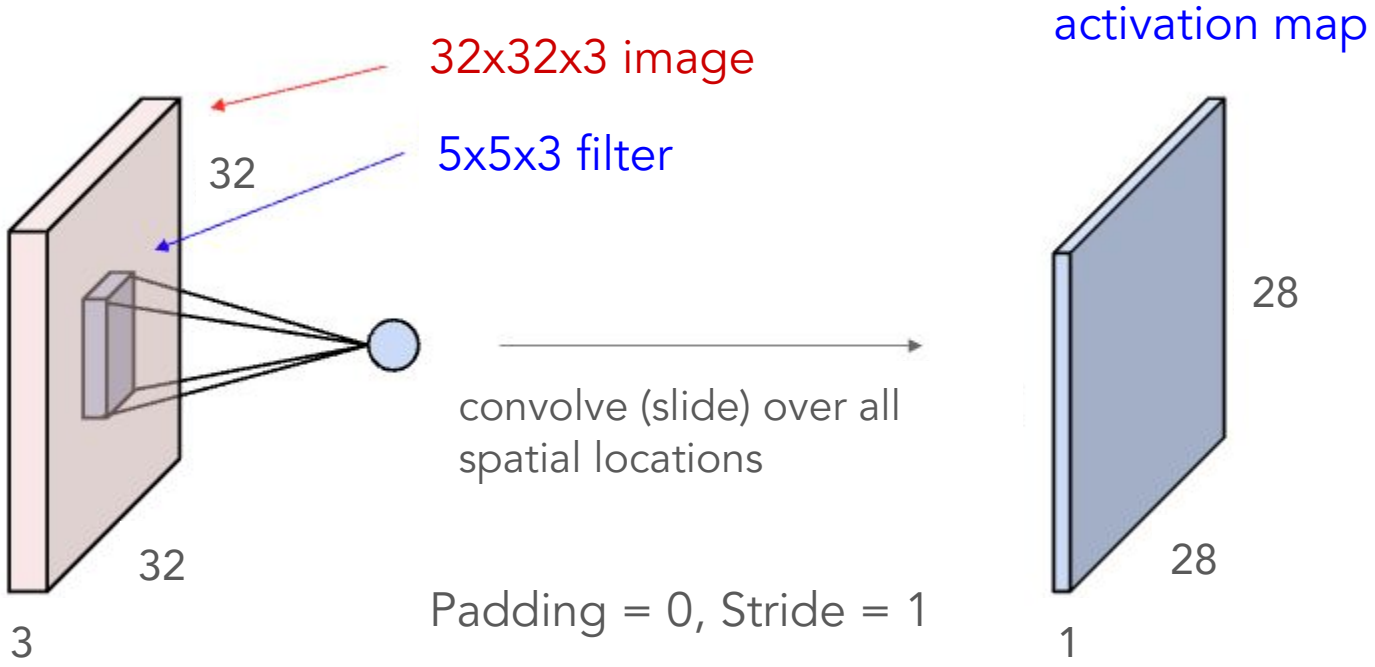
$$w^T x + b$$

Convolution Layer

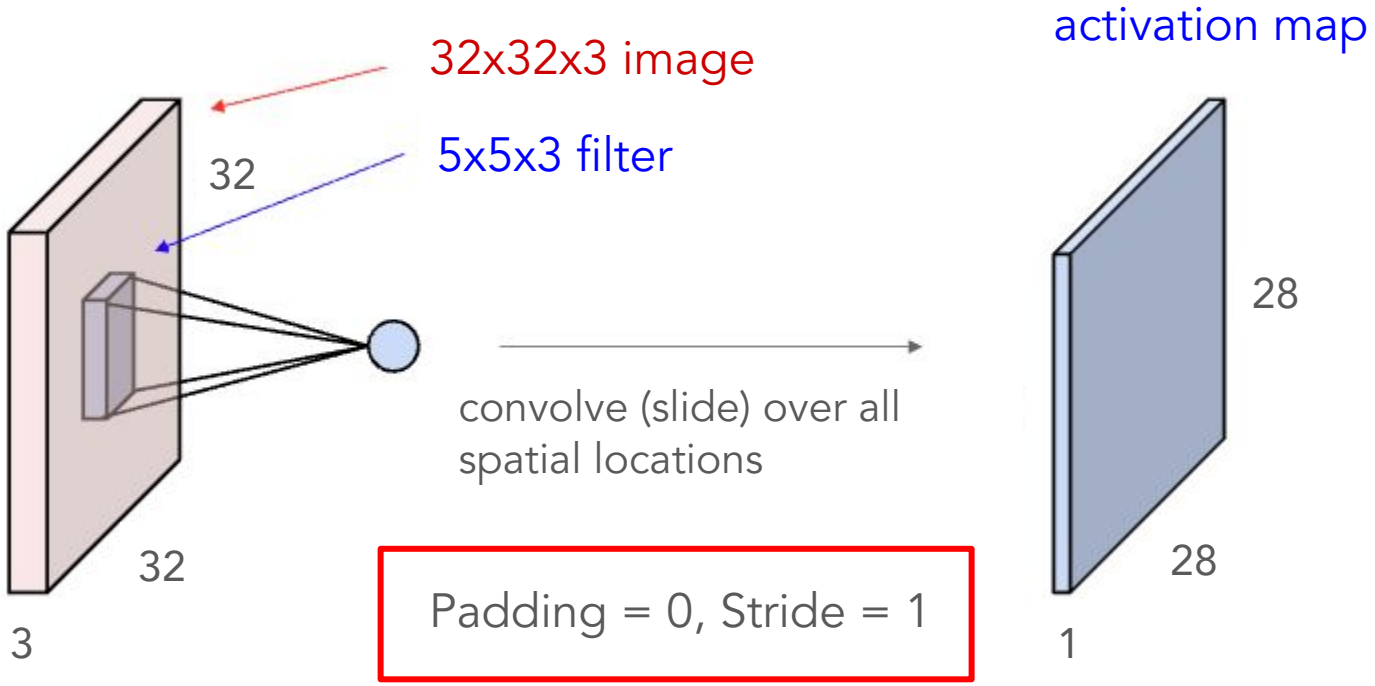




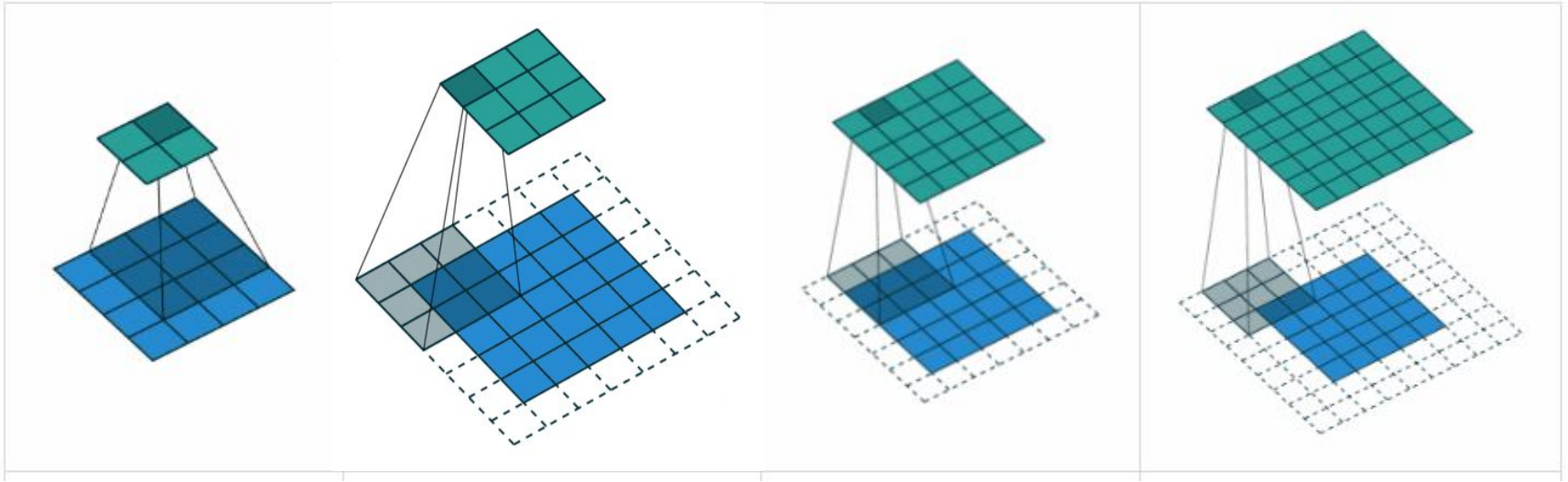
Convolution Layer



Convolution Layer



Convolution Layer



padding = 0, stride = 1

padding = 1, stride = 2

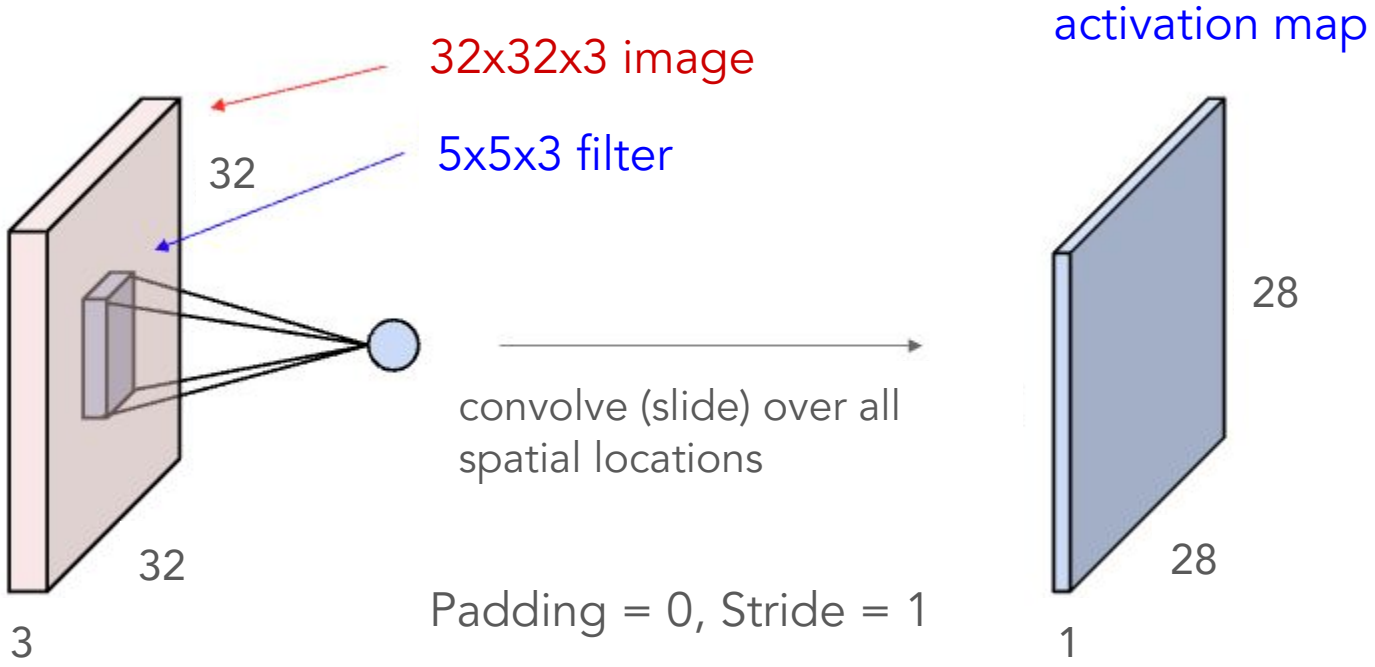
padding = 1, stride = 1

padding = 2, stride = 1

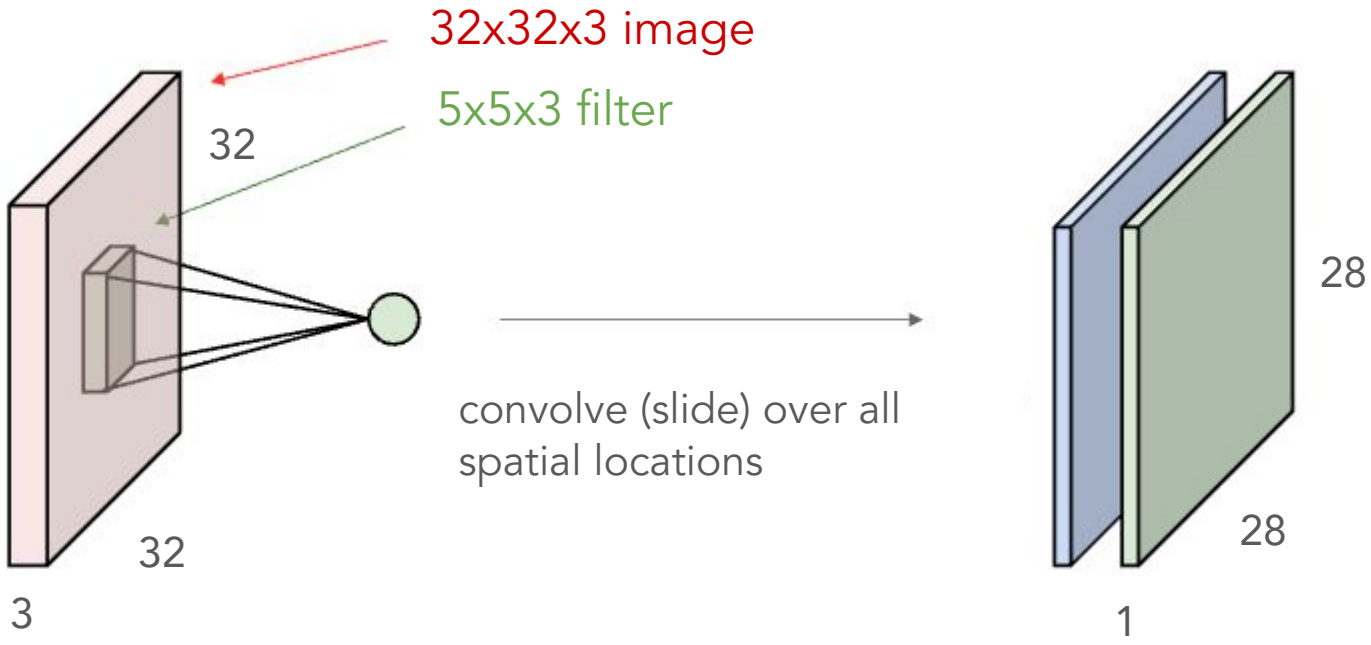
$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Animation from Hochschule der Medien

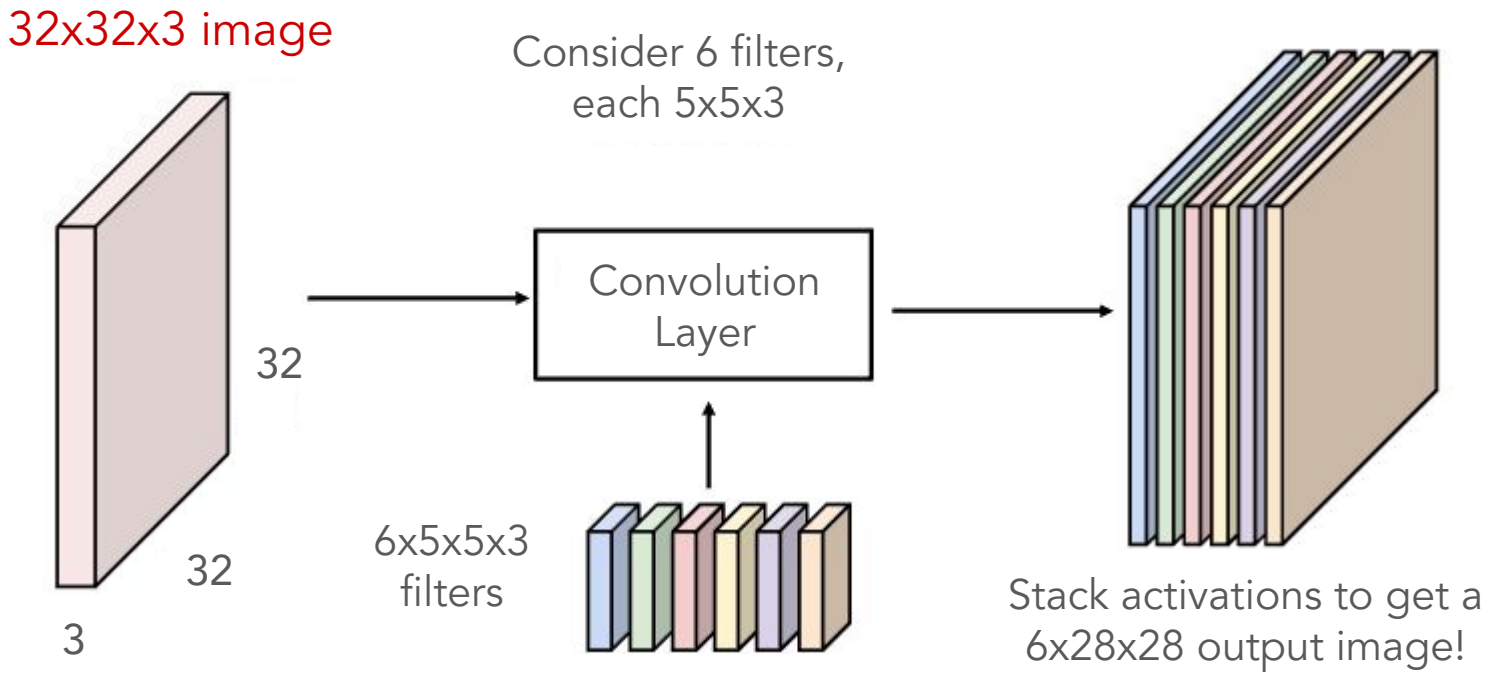
Convolution Layer



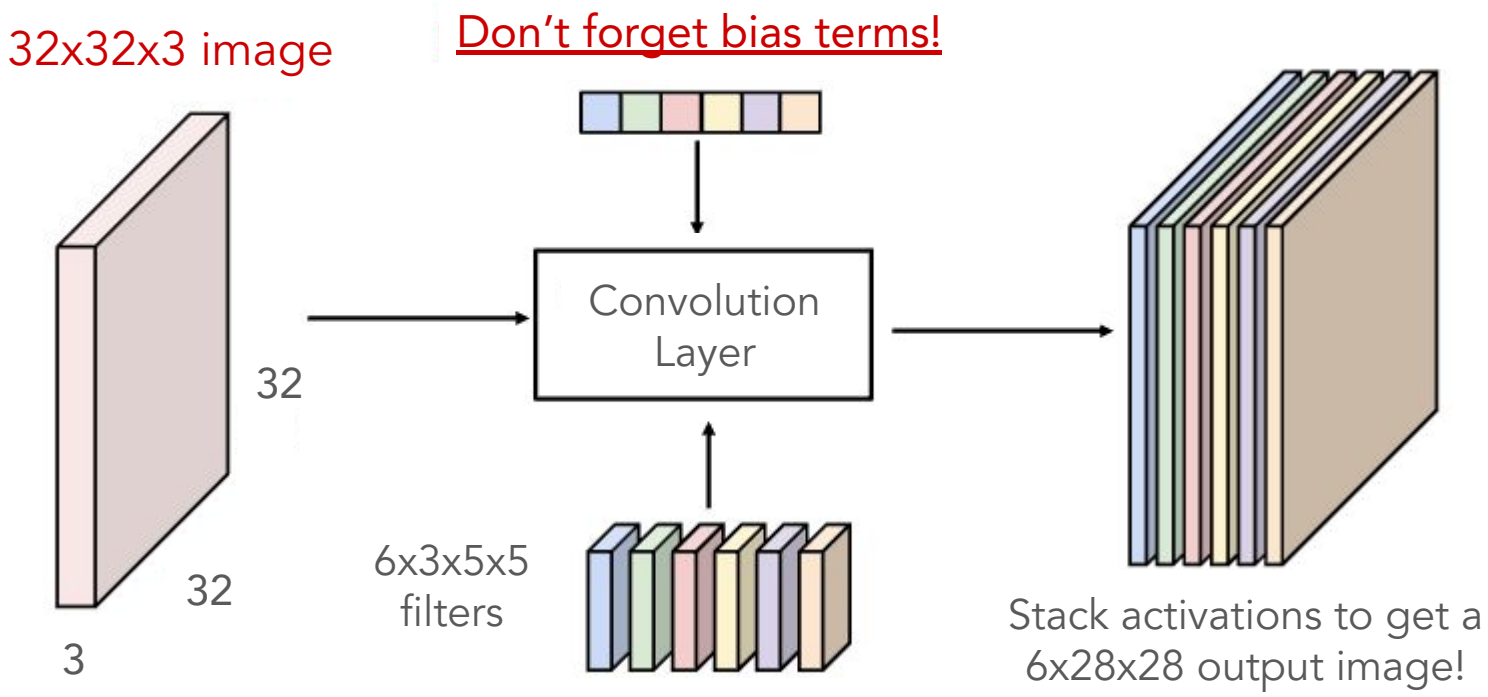
Convolution Layer



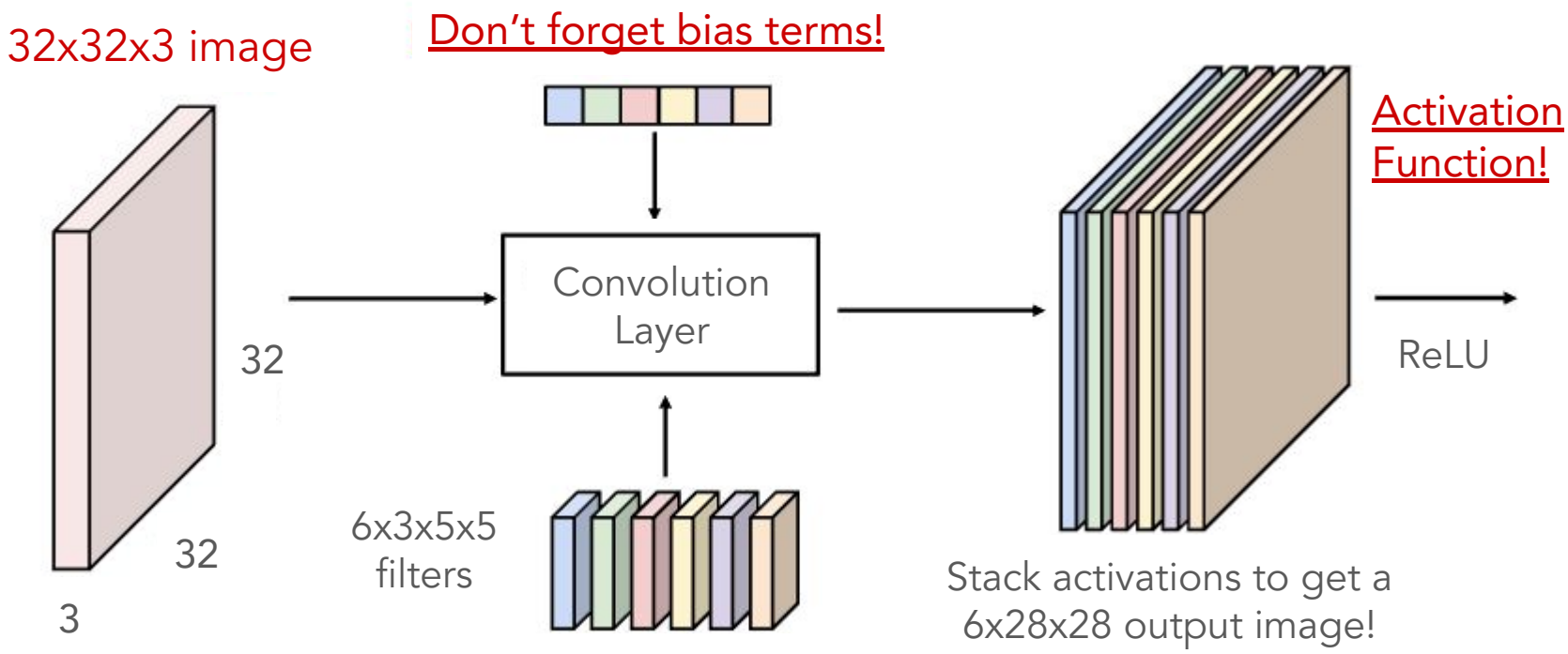
Convolution Layer



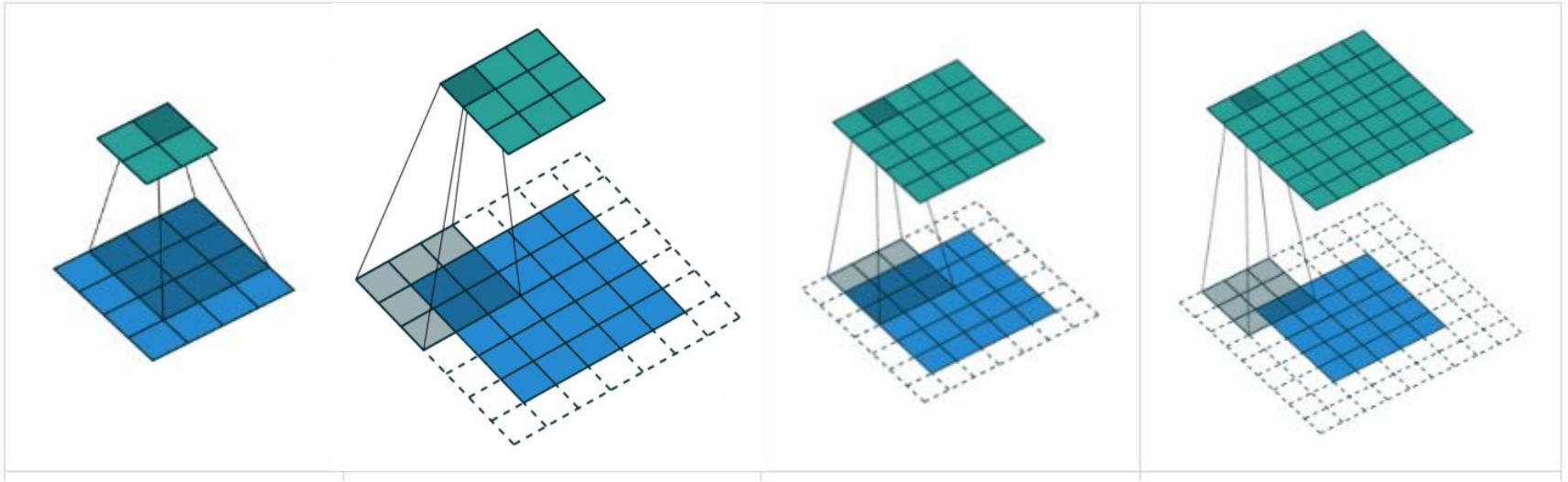
Convolution Layer



Convolution Layer



Convolution Layer



padding = 0, stride = 1

padding = 1, stride = 2

padding = 1, stride = 1

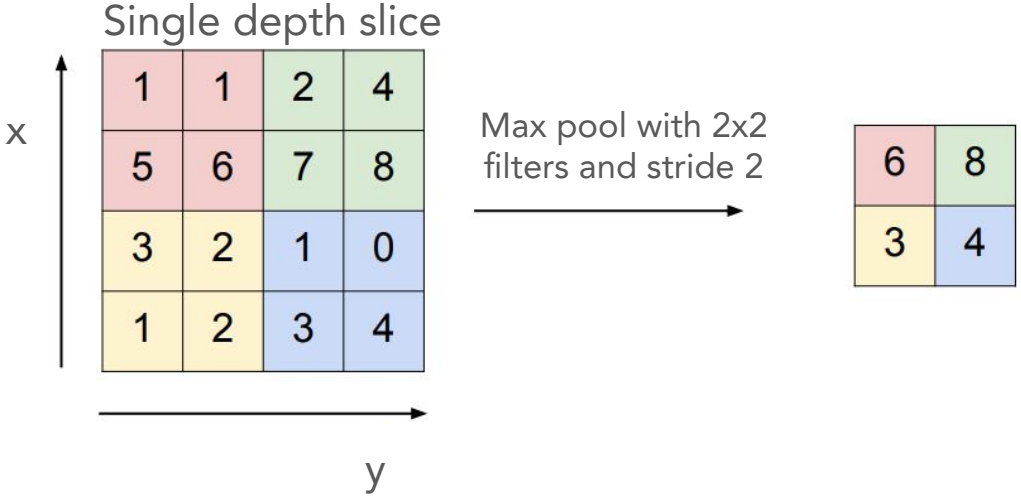
padding = 2, stride = 1

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Animation from Hochschule der Medien

Pooling (Subsampling)

MAX POOLING

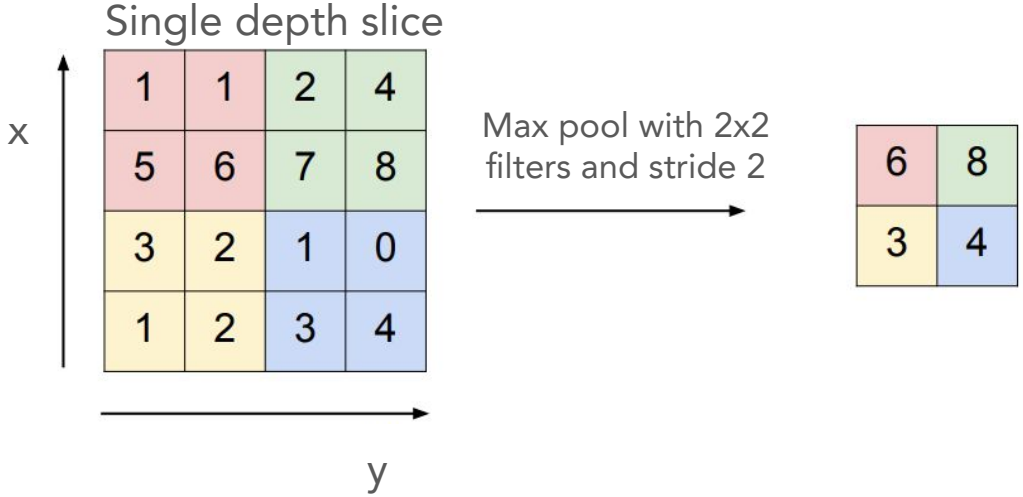


Pooling (Subsampling)

MAX POOLING

(Wx - F)/Stride + 1

(Wy - F)/Stride + 1

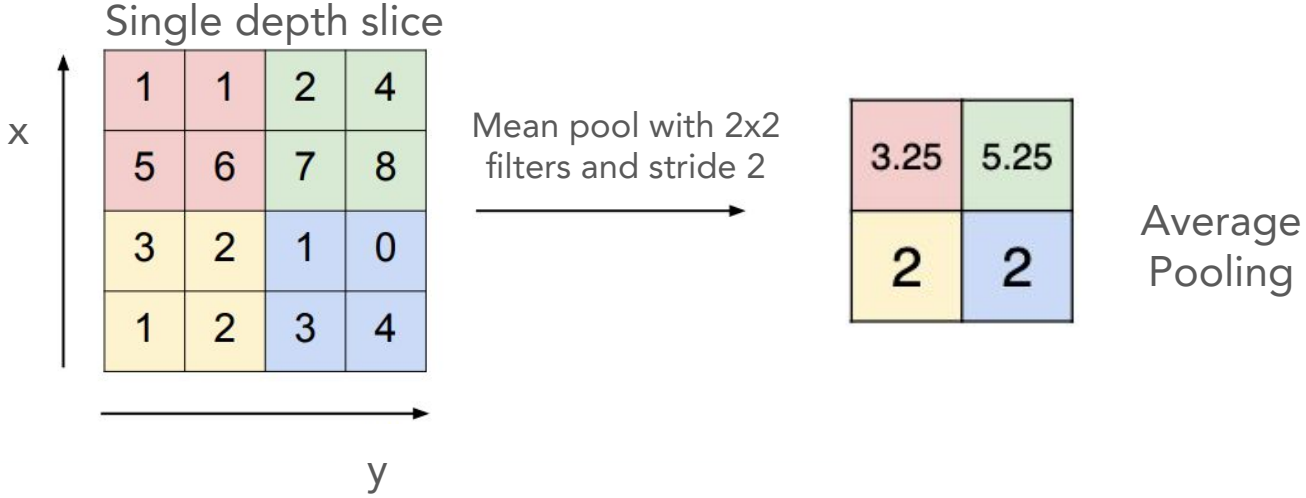


Pooling (Subsampling)

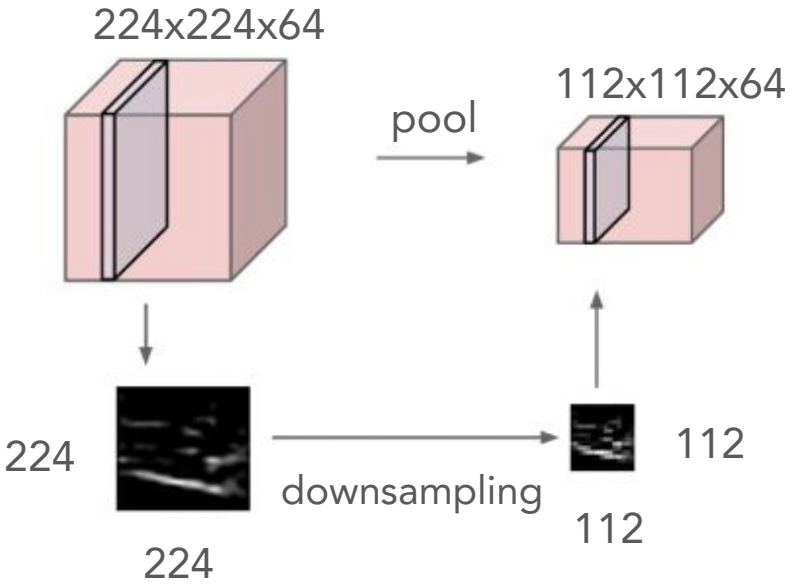
MEAN POOLING

(Wx - F)/Stride + 1

(Wy - F)/Stride + 1

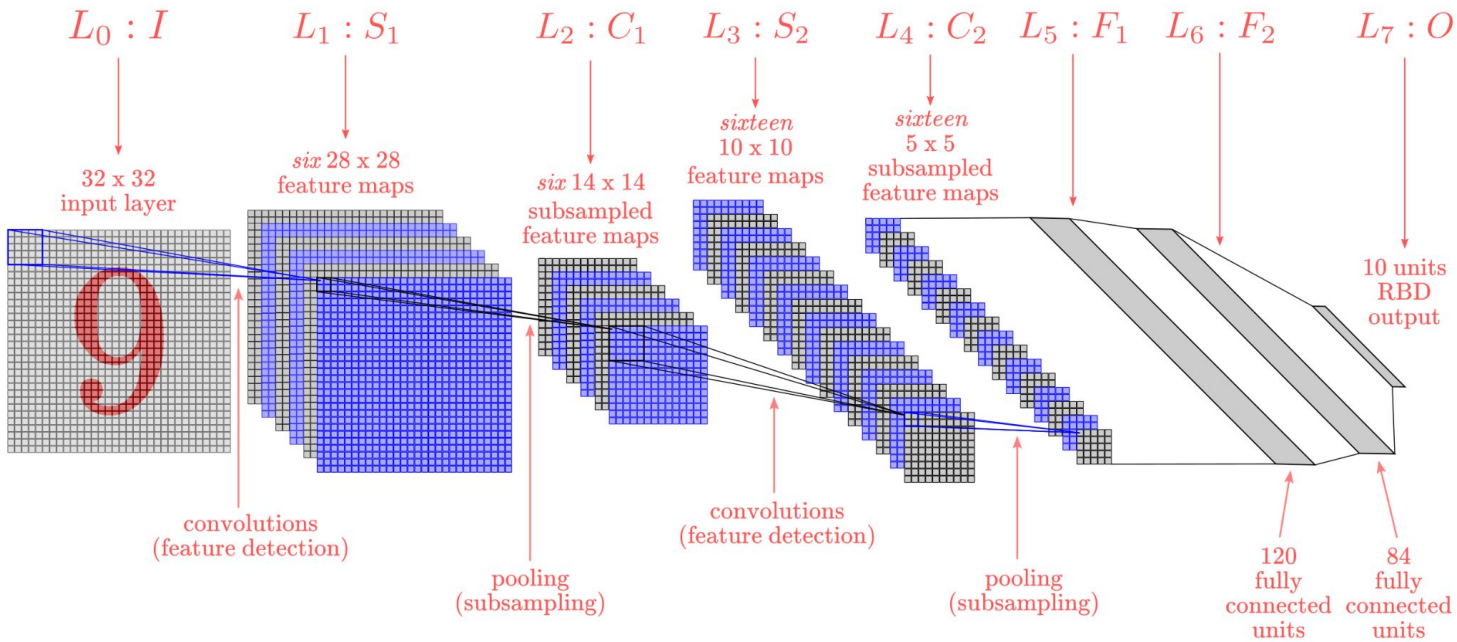


Pooling (Subsampling)



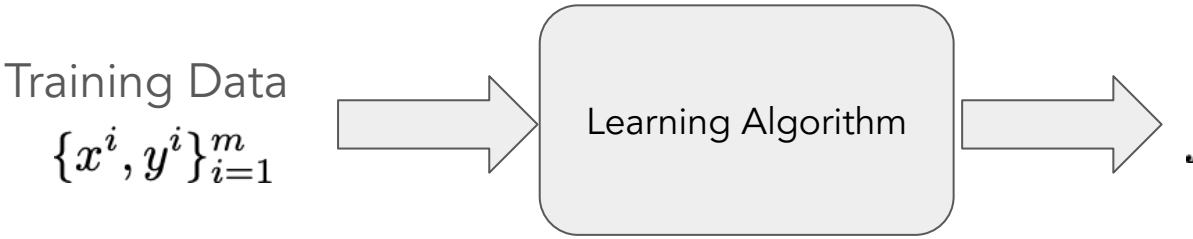
- Pooling layers simplify / subsample / compress the information in the output from the convolutional layer
- Reduce parameters

Put Everything Together



[LeNet-5, LeCun 1980]

Multiclass Logistic Regression Algorithms



Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Multiclass Logistic Regression Pipeline

1. Build probabilistic models:
   Categorical Distribution + Conv NN
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Select Optimizer

$$L(\theta) = \sum_{i=1}^{m} \ell(x^i, y^i, \theta) + \lambda\Omega(\theta)$$

$$\theta = [W_1, W_2, W_3 W_4]$$

$$\ell(x^i, y^i, \theta) = -\sum_{j=1}^{k} y^i \log \frac{f(x^i;\theta)_j}{\sum_{c=1}^{k} \exp(o(V_c g(W x^i)))} \exp(o(V_j g(W x^i)))$$

$$f(x^i;\theta)$$

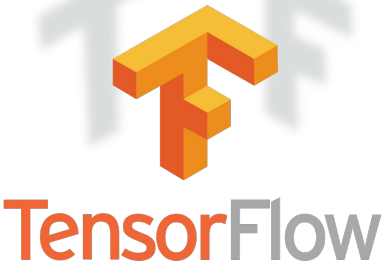
- (Stochastic) Gradient Descent

Auto-differentiation Packages

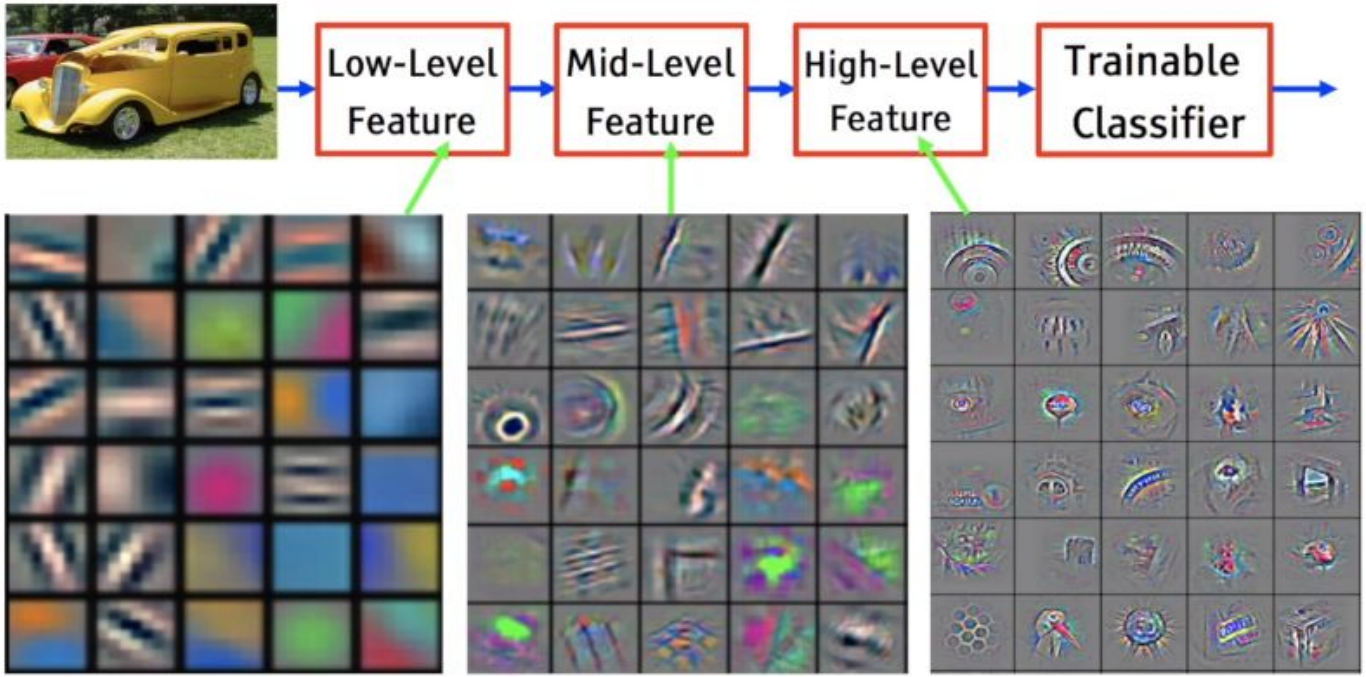
PyTorch

JAX

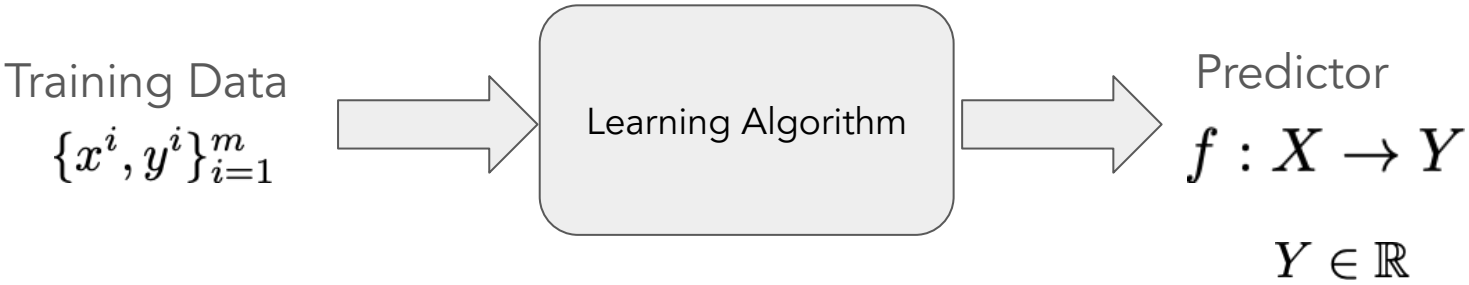
Tensorflow



Convolution Features



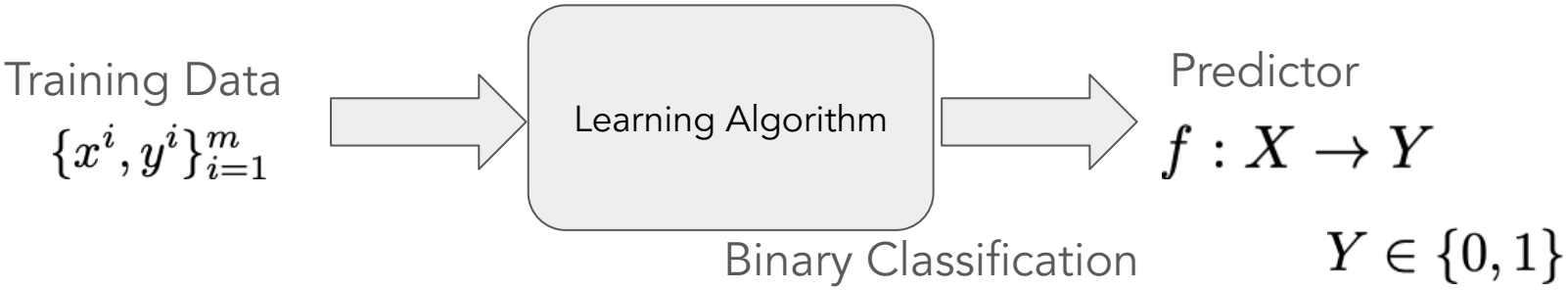
Regression Algorithms



Linear Regression Pipeline

1. Build probabilistic models:
   Gaussian Distribution + Conv NN
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Binary Classification Algorithms

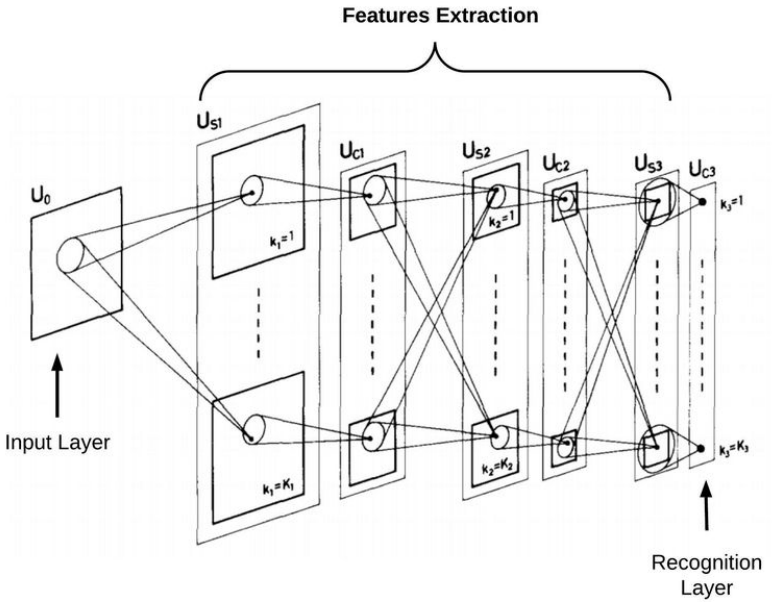


Binary Logistic Regression Pipeline

1. Build probabilistic models:
   Bernoulli Distribution + Conv NN
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

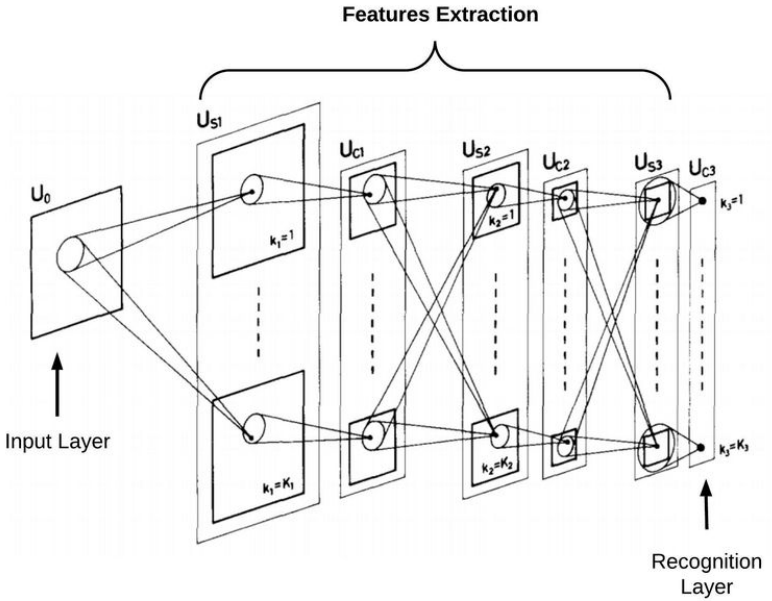
A Bit of History

- Neocognitron [Fukushima 1980]



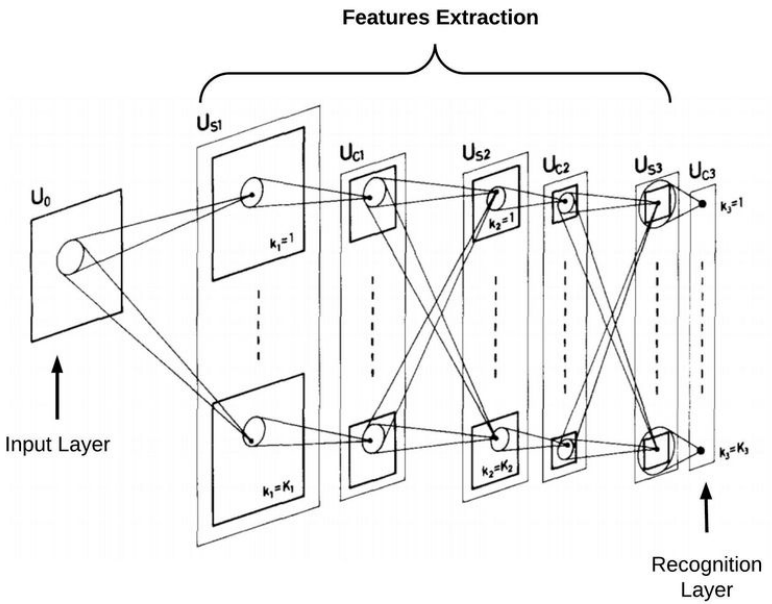
A Bit of History

- Neocognitron [Fukushima 1980]
- Backpropagation [Rumelhart et al., 1986, Werbos 1974]



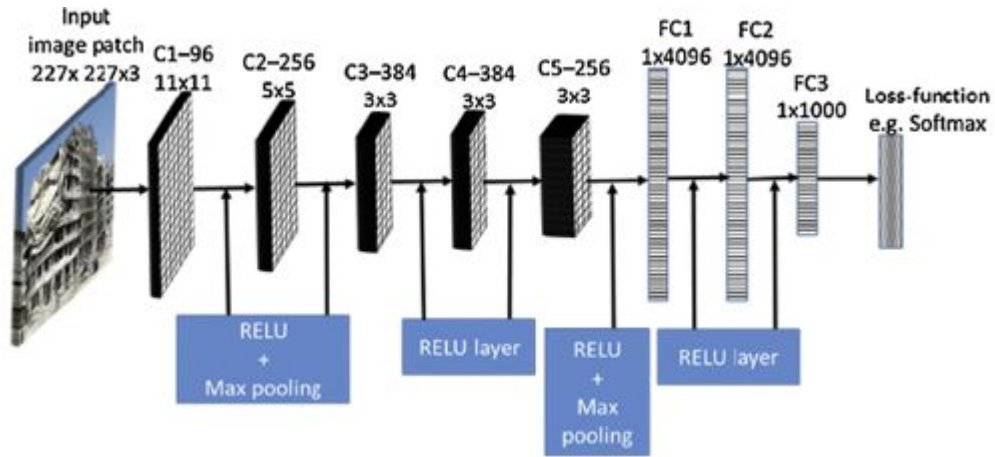
A Bit of History

- Neocognitron [Fukushima 1980]
- Backpropagation [Rumelhart et al., 1986, Werbos 1974]
- BP on CNN [LeCun et al., 1989]

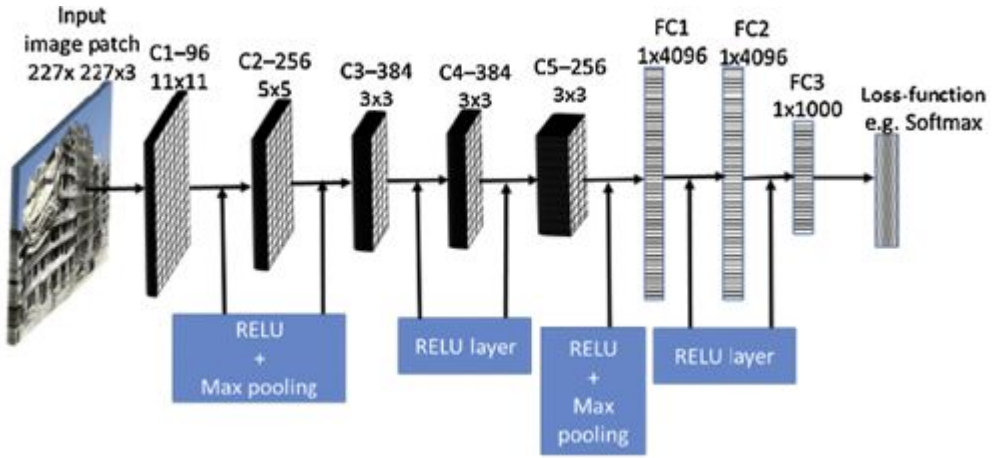


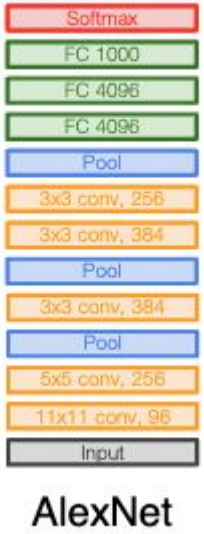
AlexNet (2012)

- AlexNet was one of the first deep convolutional on ImageNet competition with an accuracy of 84.7% as compared to the second-best with an accuracy of 73.8%.
- The activation used is the Rectified Linear Unit (ReLU).
- The training set had 1.2 million images. It was trained for 90 epochs, which took five to six days on two NVIDIA GTX 580 3GB GPUs.



AlexNet (2012)

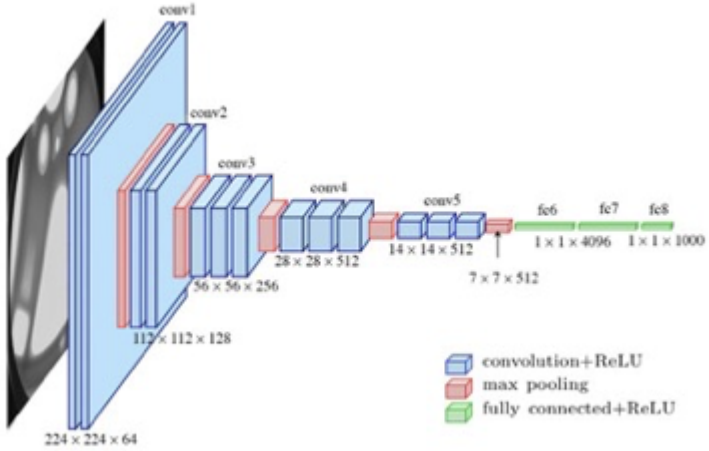




AlexNet

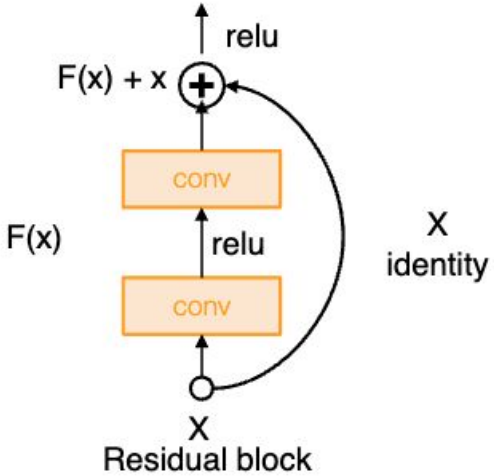
VGGNet (2014)

- Very Deep CNN
- With only 3*3 conv filters
  - Fewer parameters, deeper nonlinear layers



ResNet (2015)

- Very Deep CNN with residual connections
  - 152-layer model for ImageNet
  - ILSVRC'15 classification winner (3.57% top 5 error)
  - Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



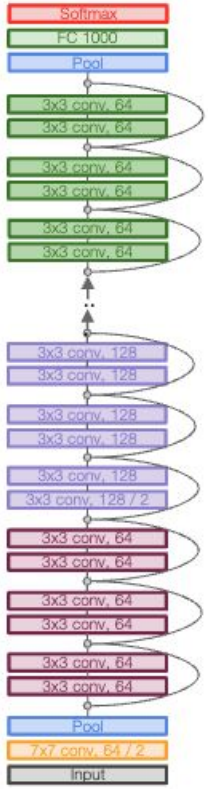
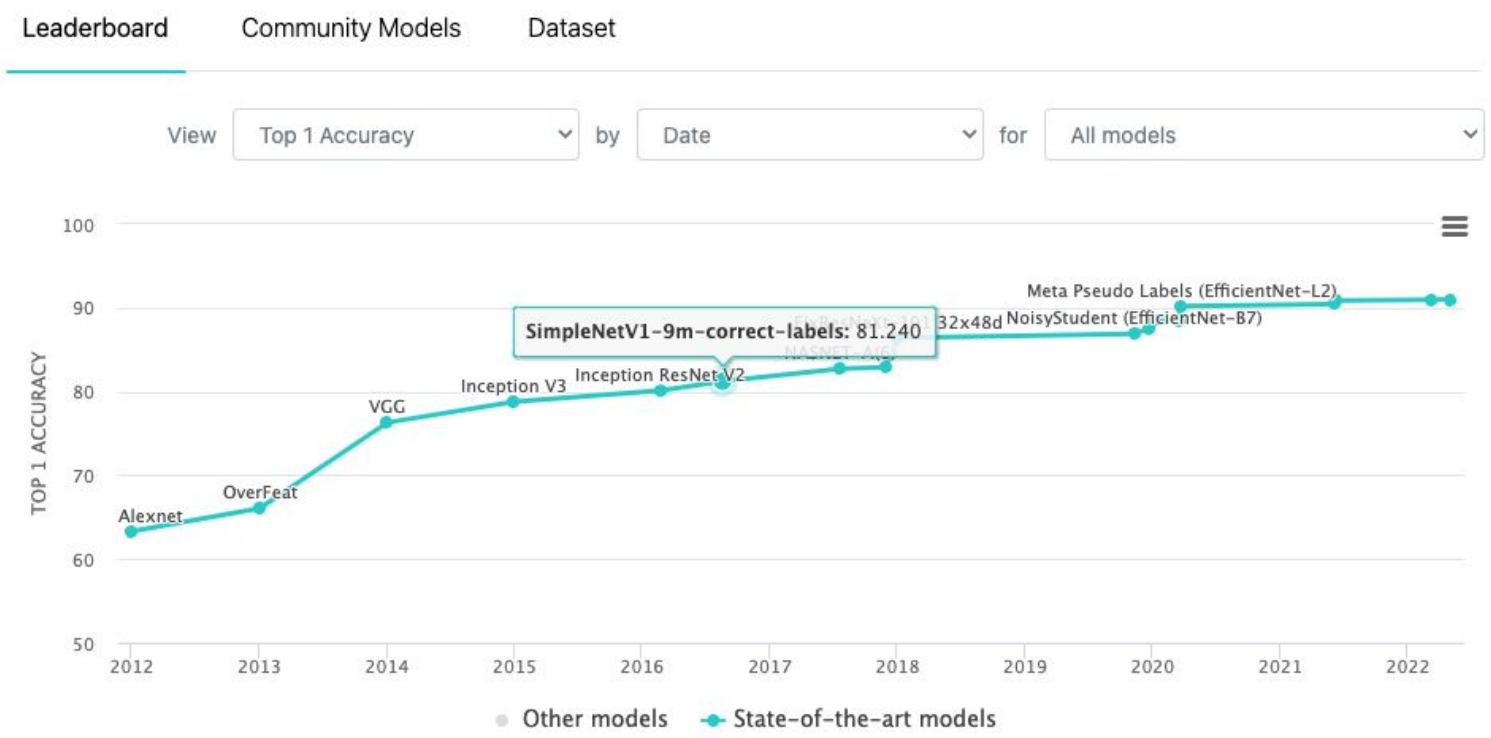


Image Classification on ImageNet



Q&A

HW2 Due Today!

No HW until Mar. 11th
Use the time wisely for your projects