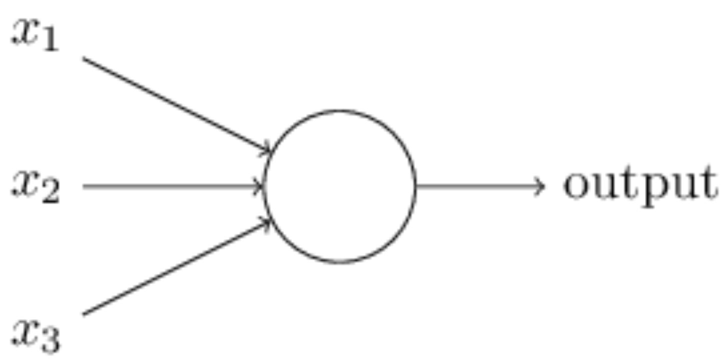
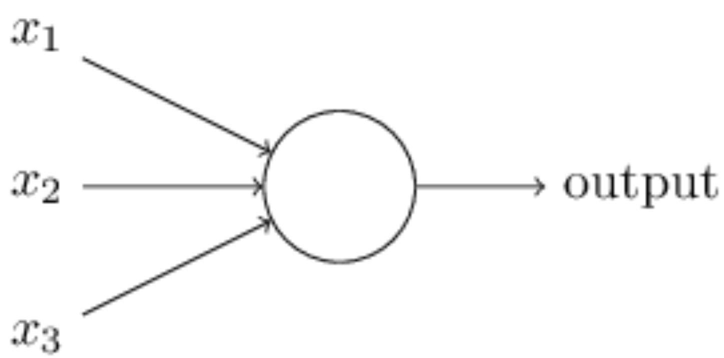
Neural networks basics





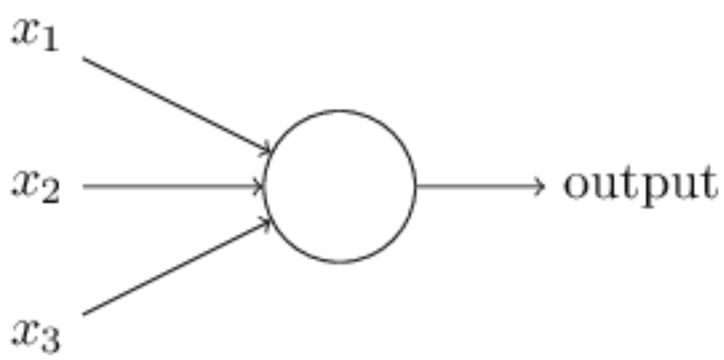
(almost) all images taken from:

Neural Networks and Deep Learning

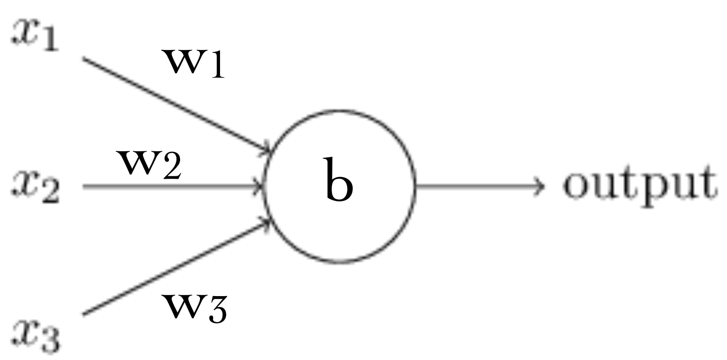
Michael Nielsen

http://neuralnetworksanddeeplearning.com/

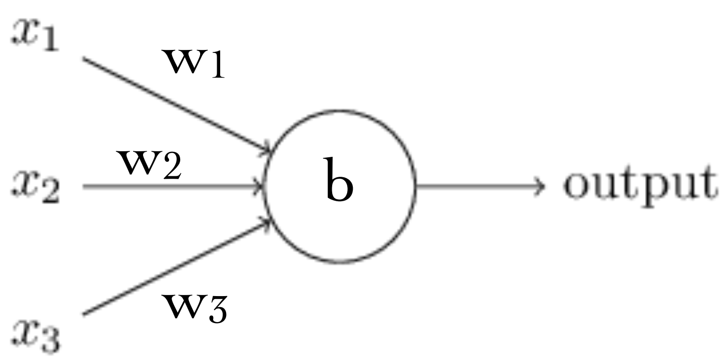
Neural networks basics



Neural networks basics

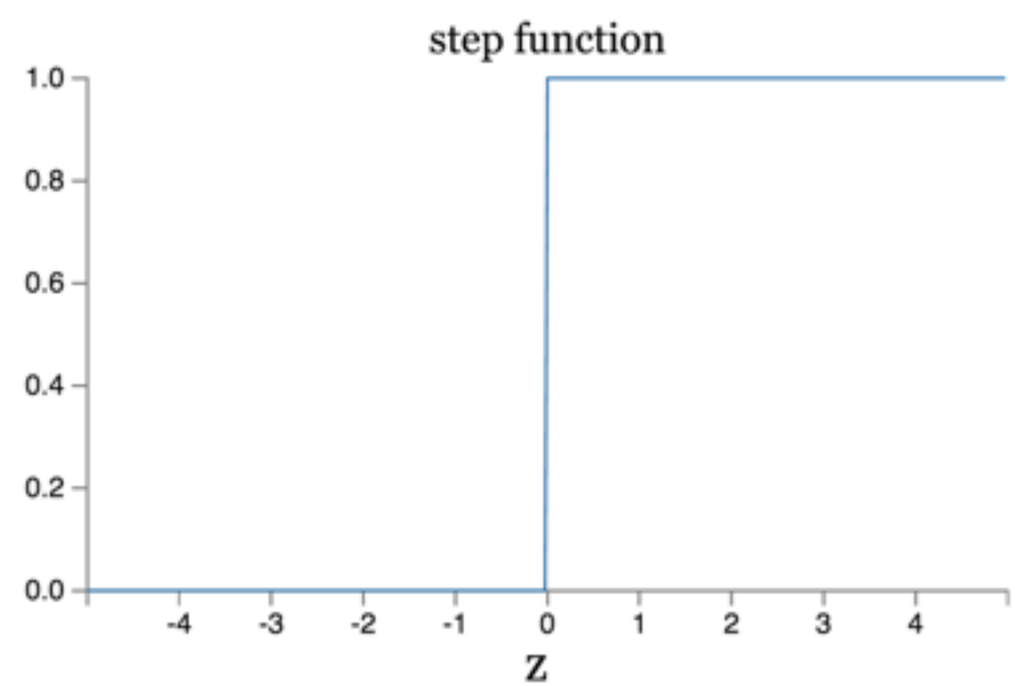


Neural networks basics

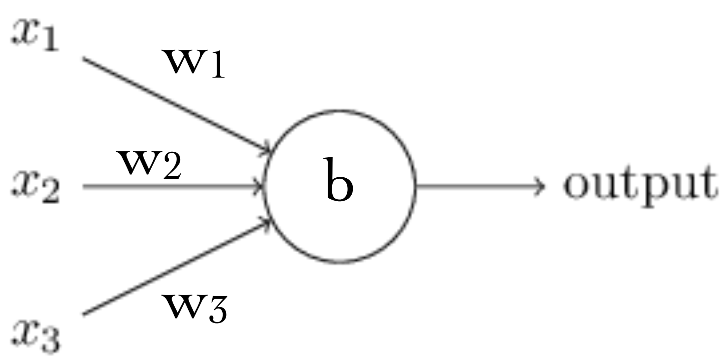


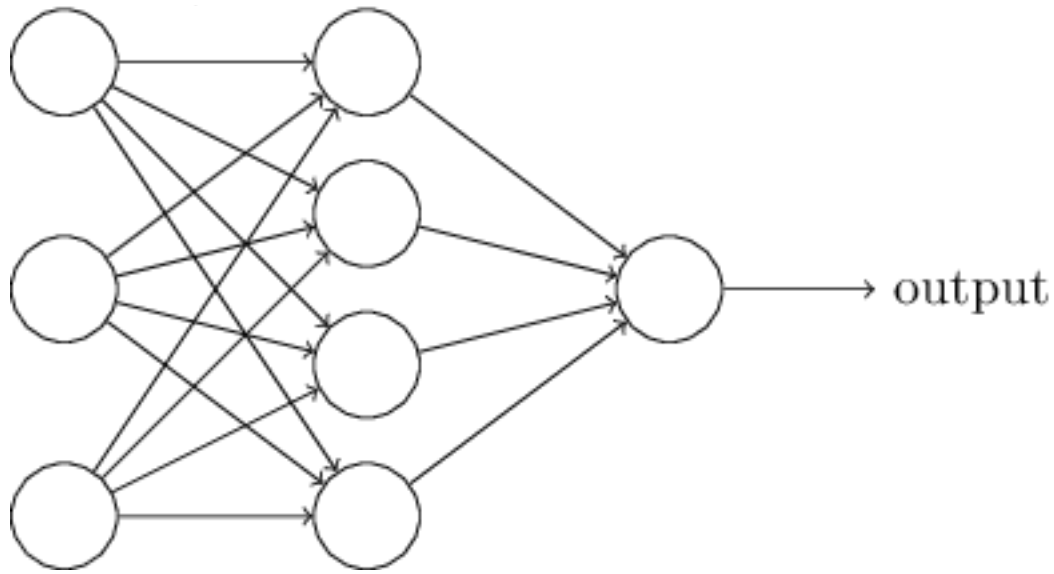
Perceptron

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$



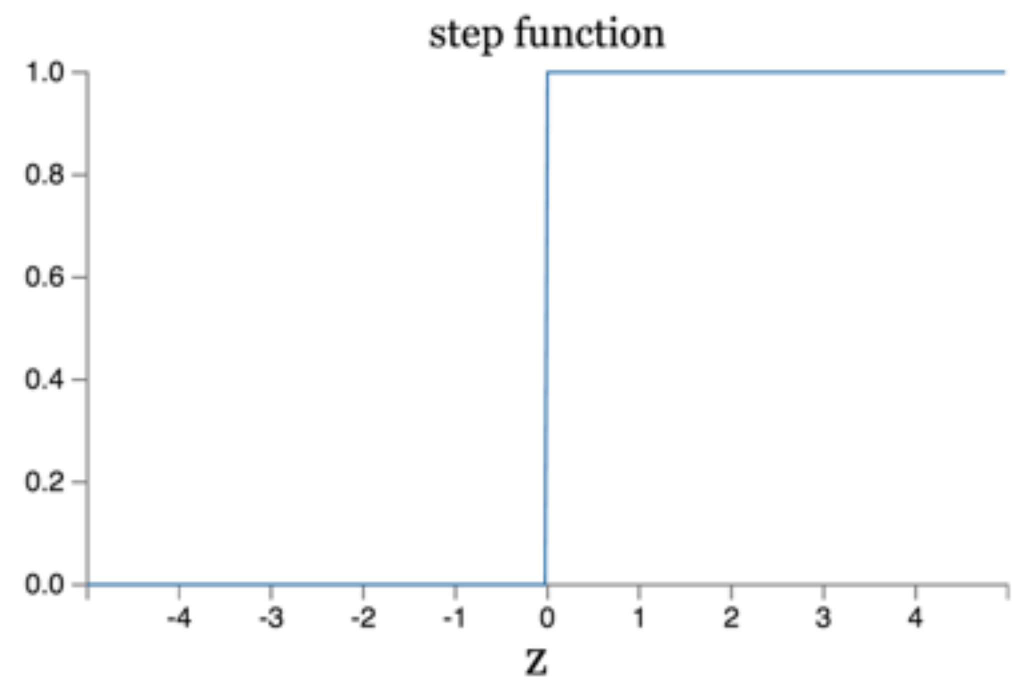
Neural networks basics



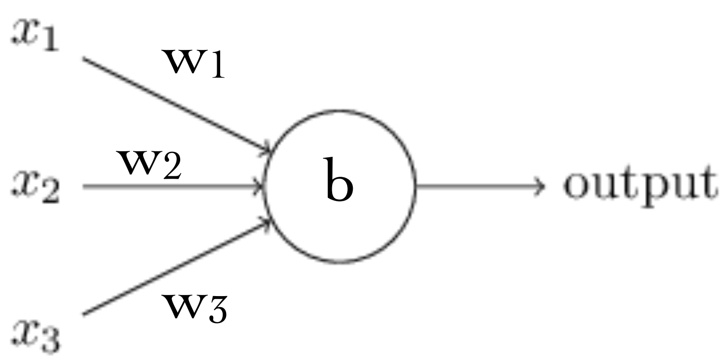


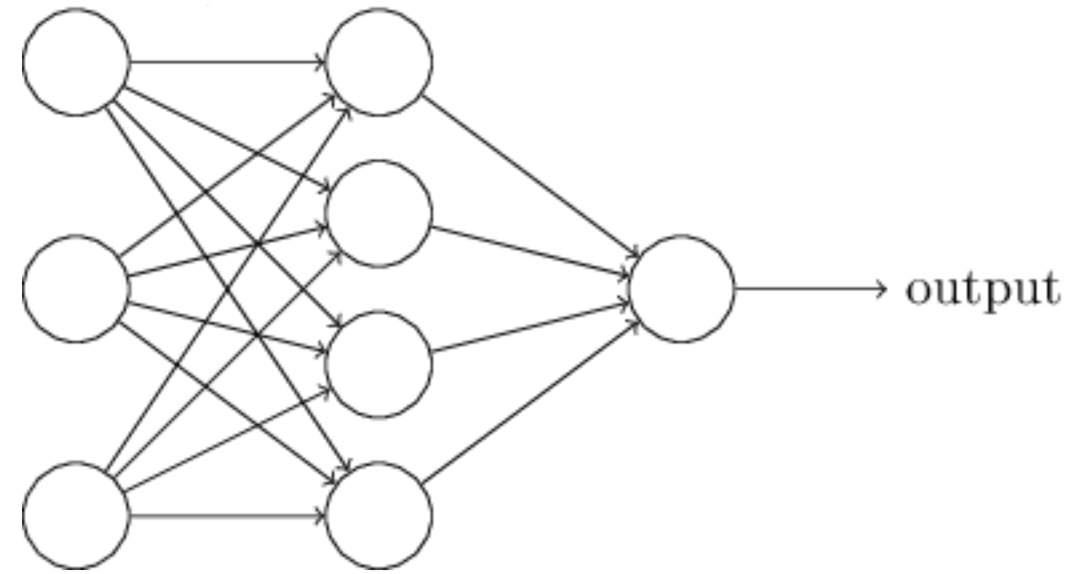
Perceptron

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$



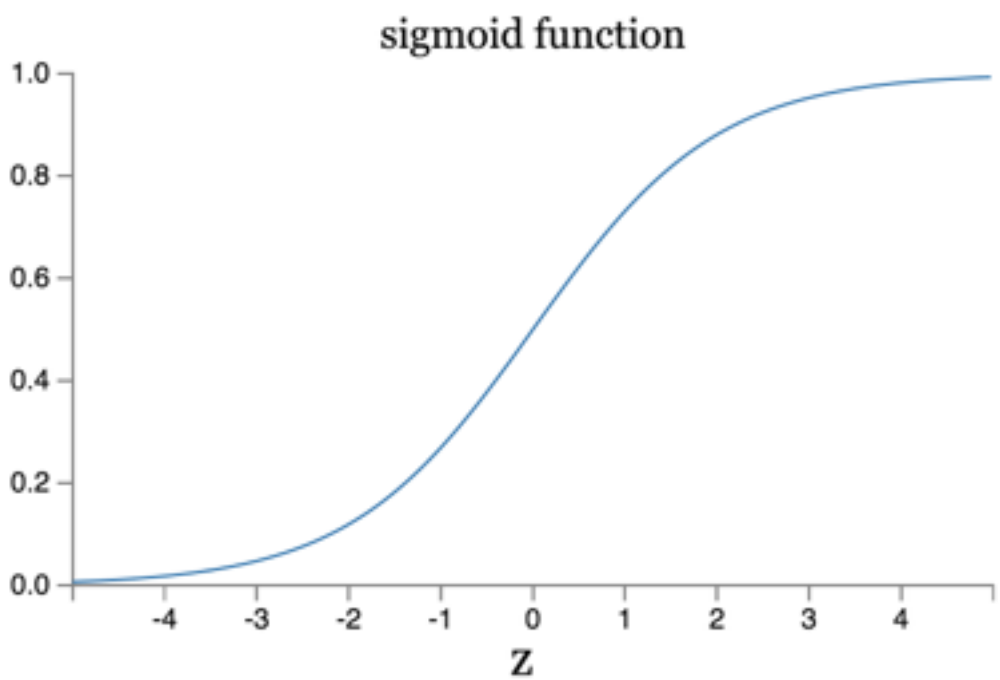
Neural networks basics





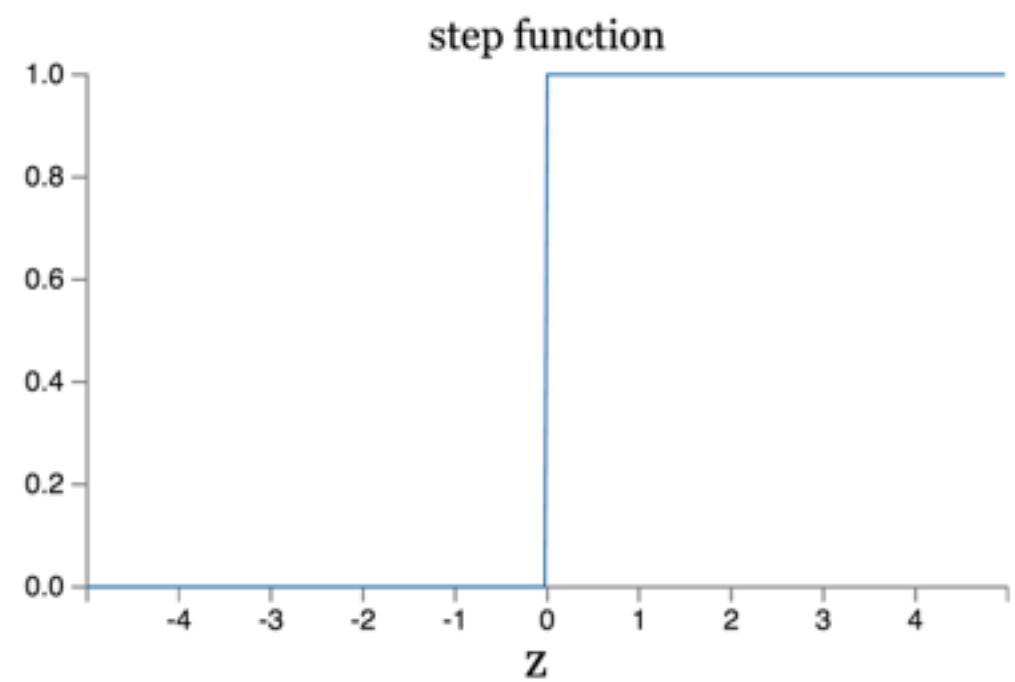
Sigmoid

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} \qquad z \equiv w \cdot x + b$$



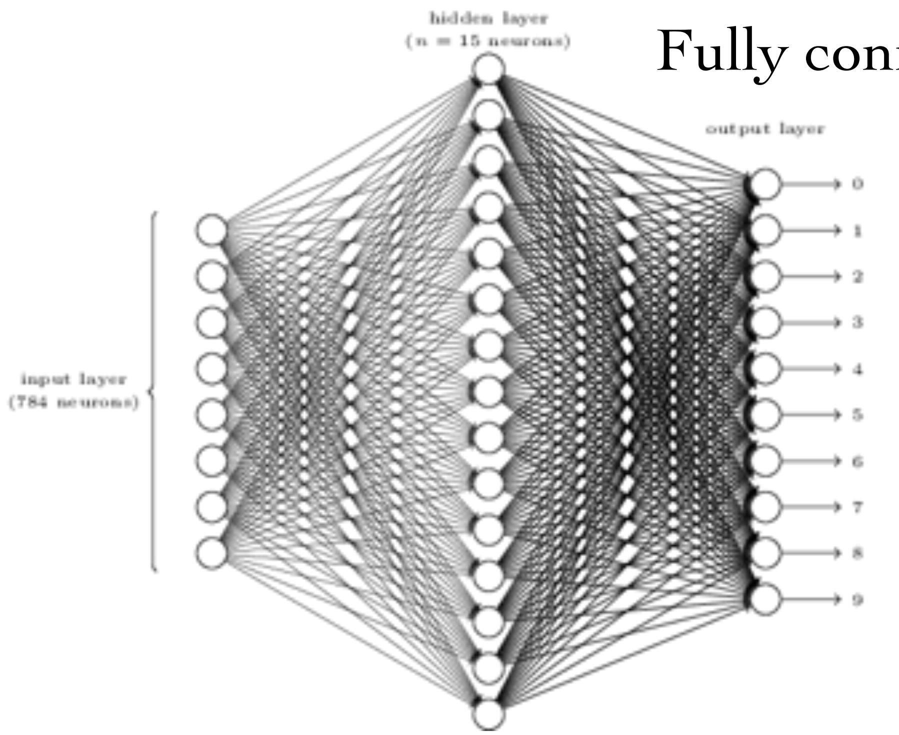
Perceptron

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$



Neural networks basics

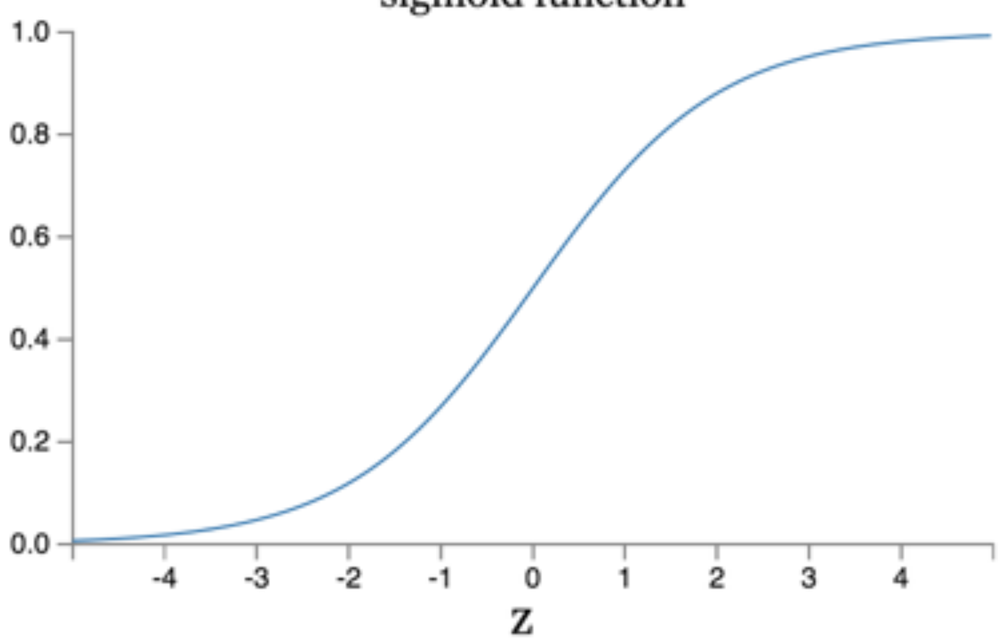
Fully connected

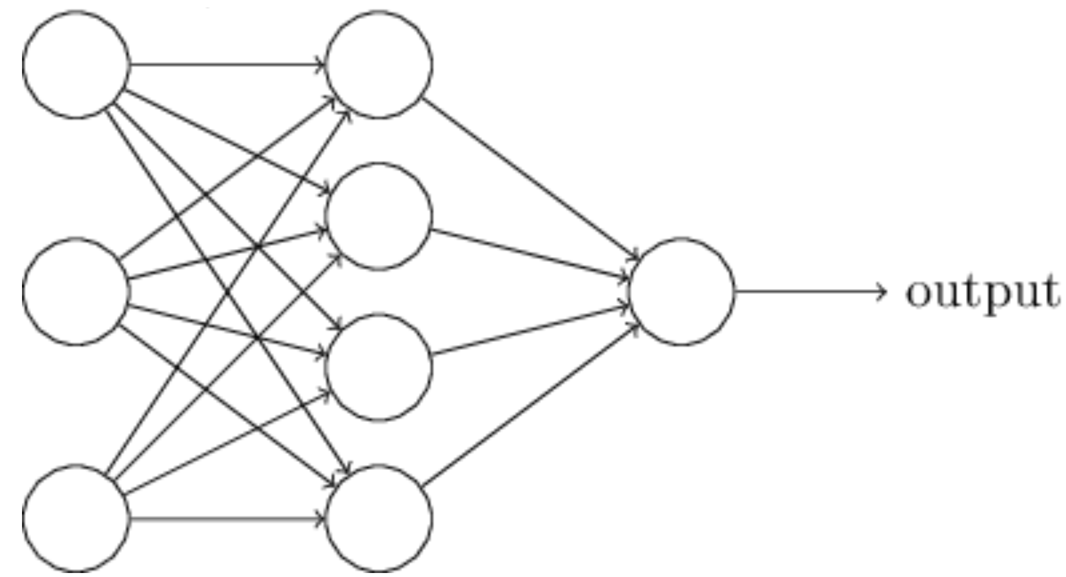


Sigmoid

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} \qquad z \equiv w \cdot x + b$$

sigmoid function

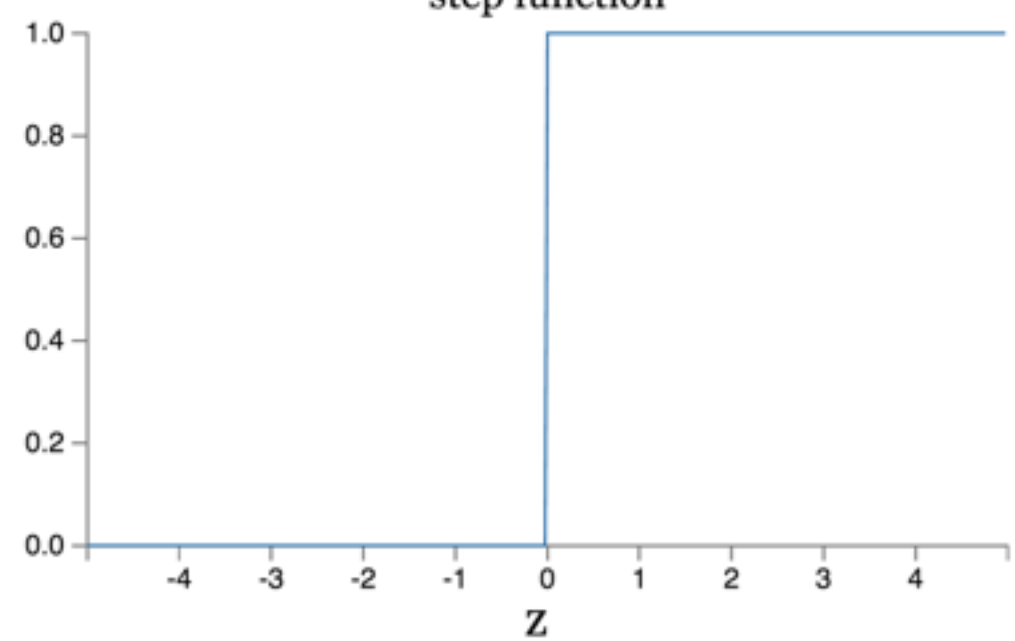




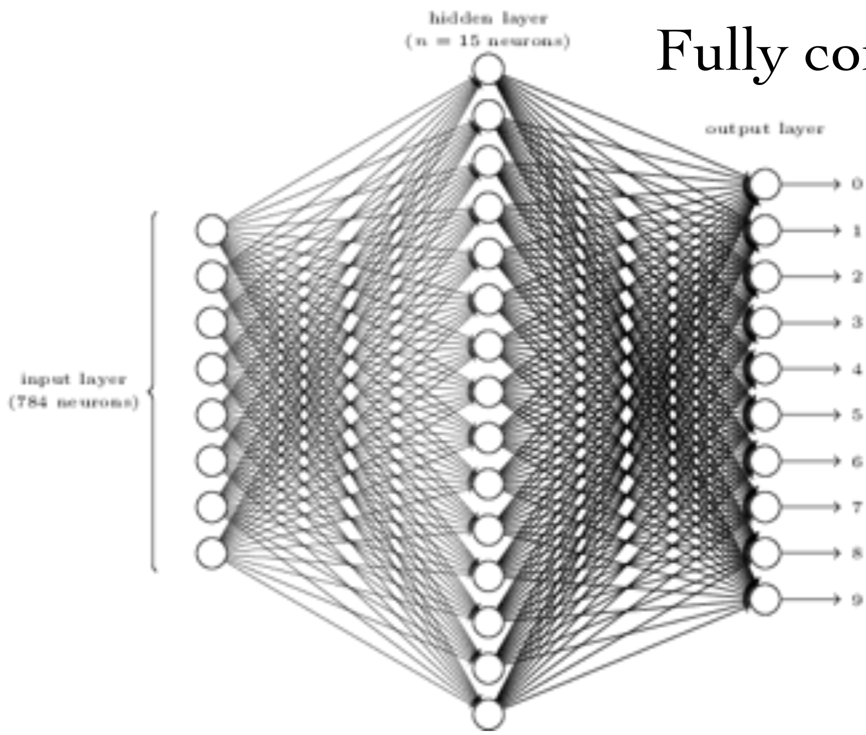
Perceptron

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

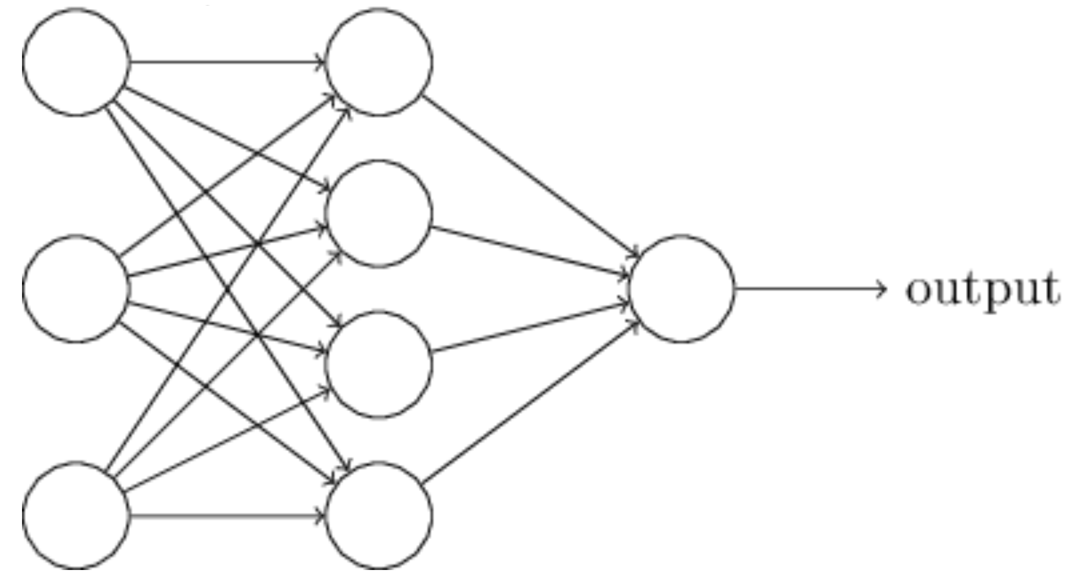
step function

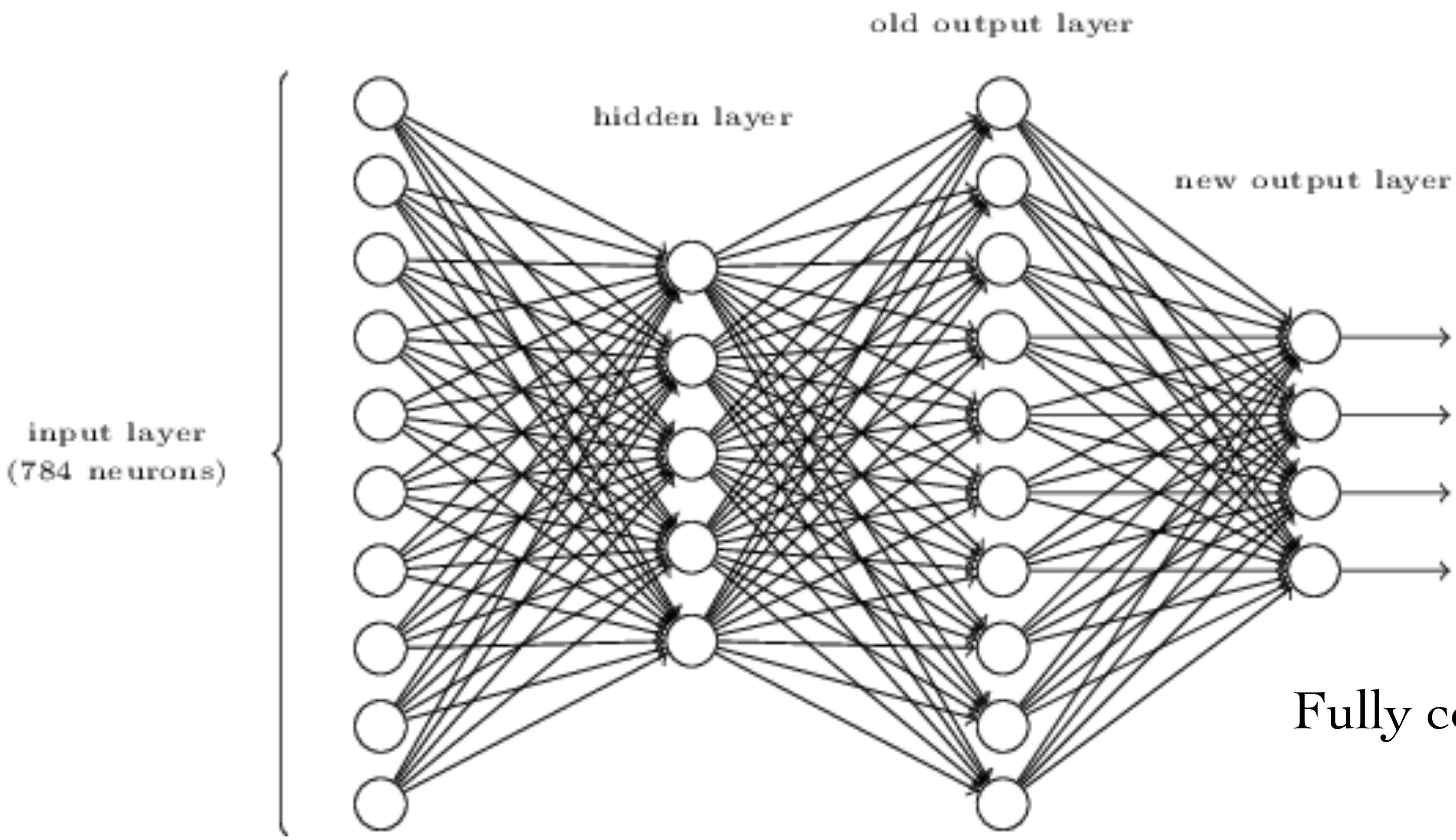


Neural networks basics



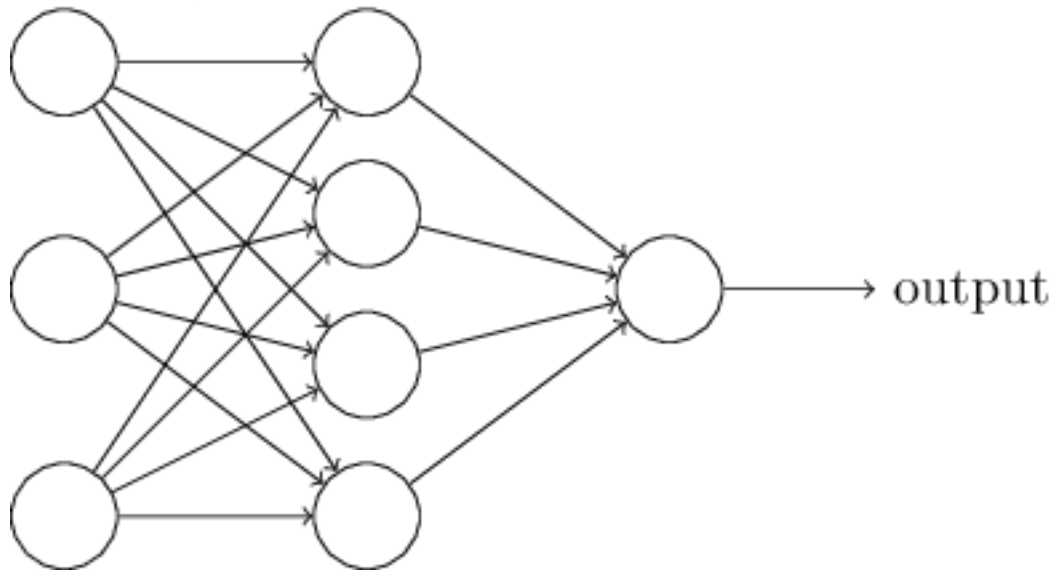
Fully connected



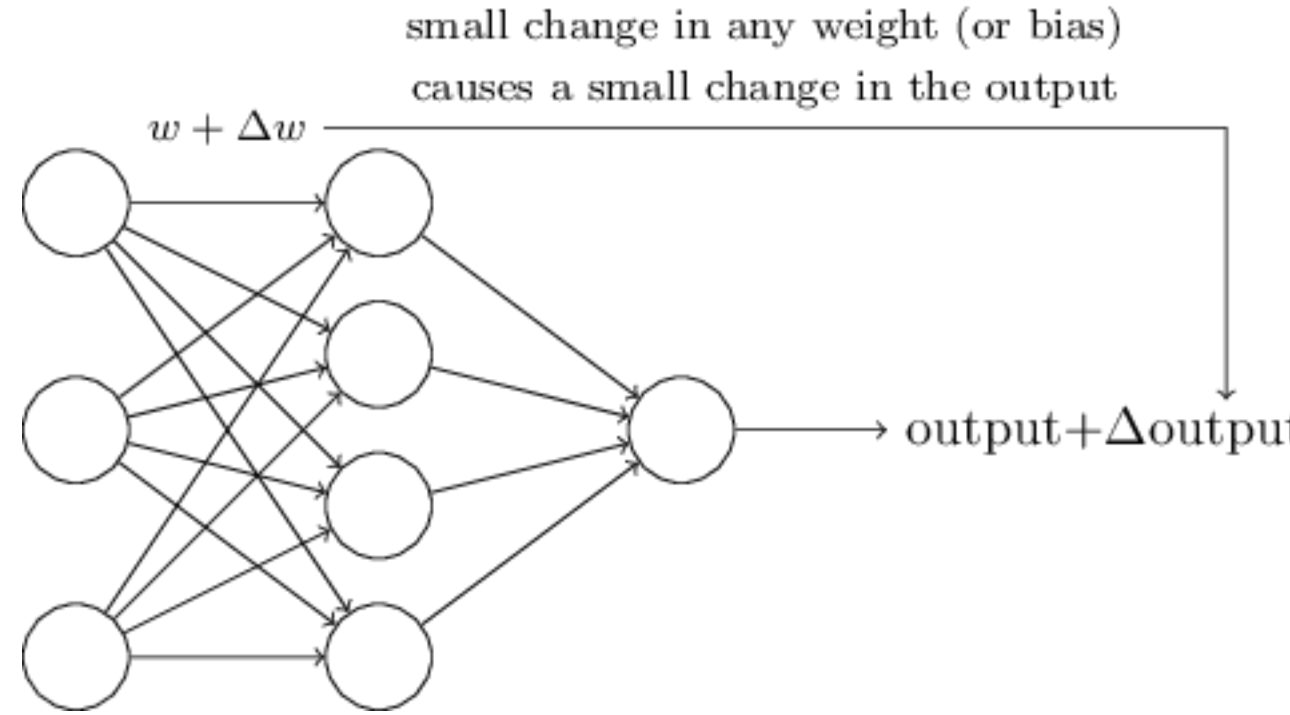


Fully connected (deep-ish)

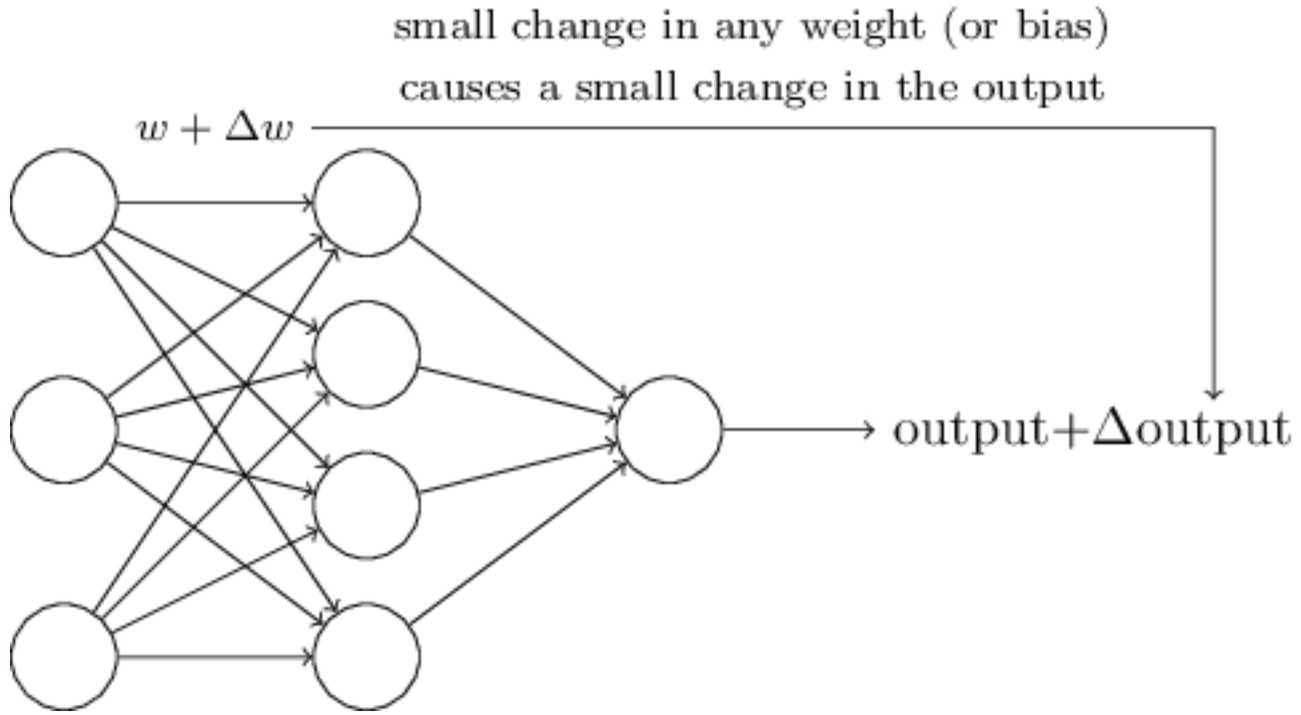
Training a Neural Net



Training a Neural Net



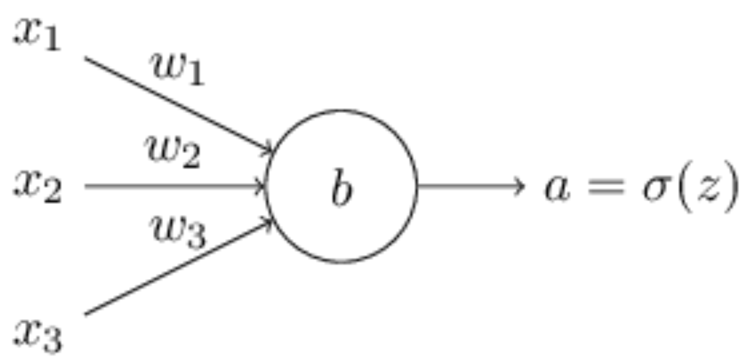
Training a Neural Net

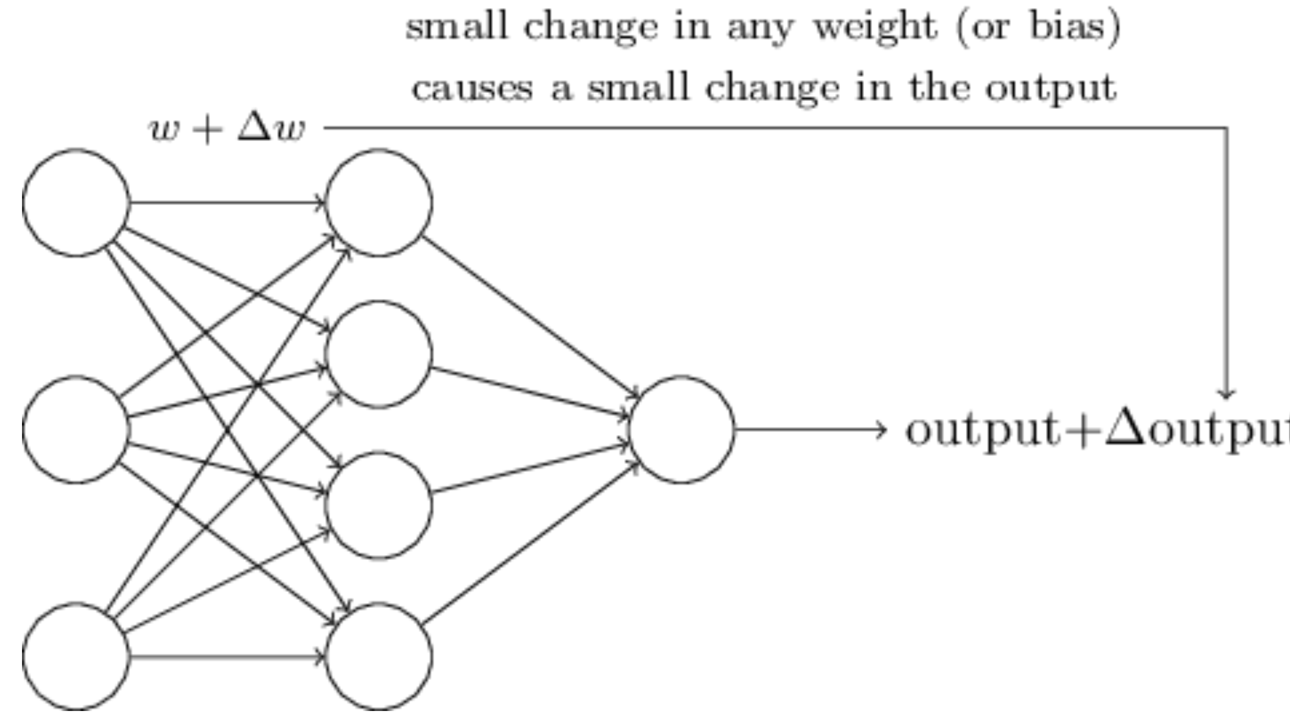


$$\Delta \text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b$$

Training a Neural Net

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$



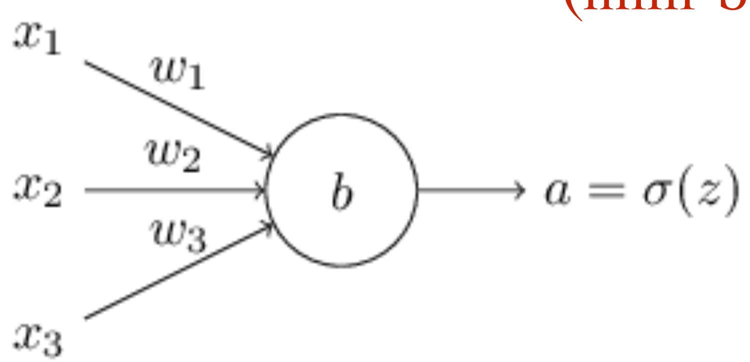


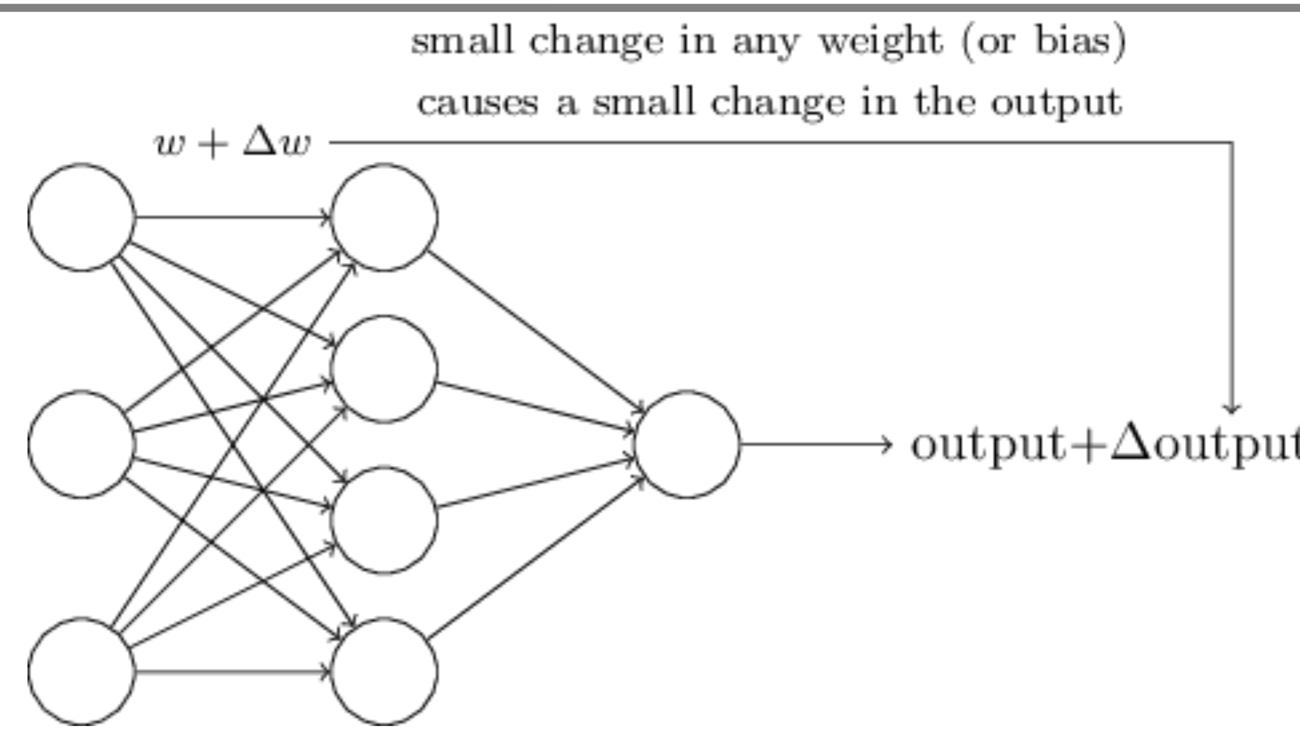
$$\Delta\text{output} \approx \sum_j \frac{\partial\,\text{output}}{\partial w_j} \Delta w_j + \frac{\partial\,\text{output}}{\partial b} \Delta b$$

Training a Neural Net

$$C(w, b) \equiv \frac{1}{2n} \sum_{x} \|y(x) - a\|^2$$

training examples
(mini-batches)



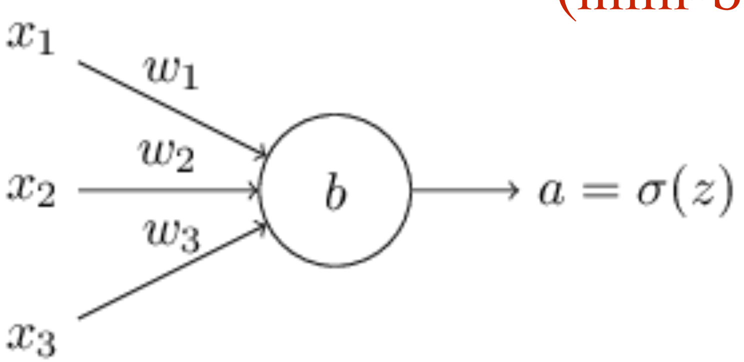


$$\Delta\text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b$$

Training a Neural Net

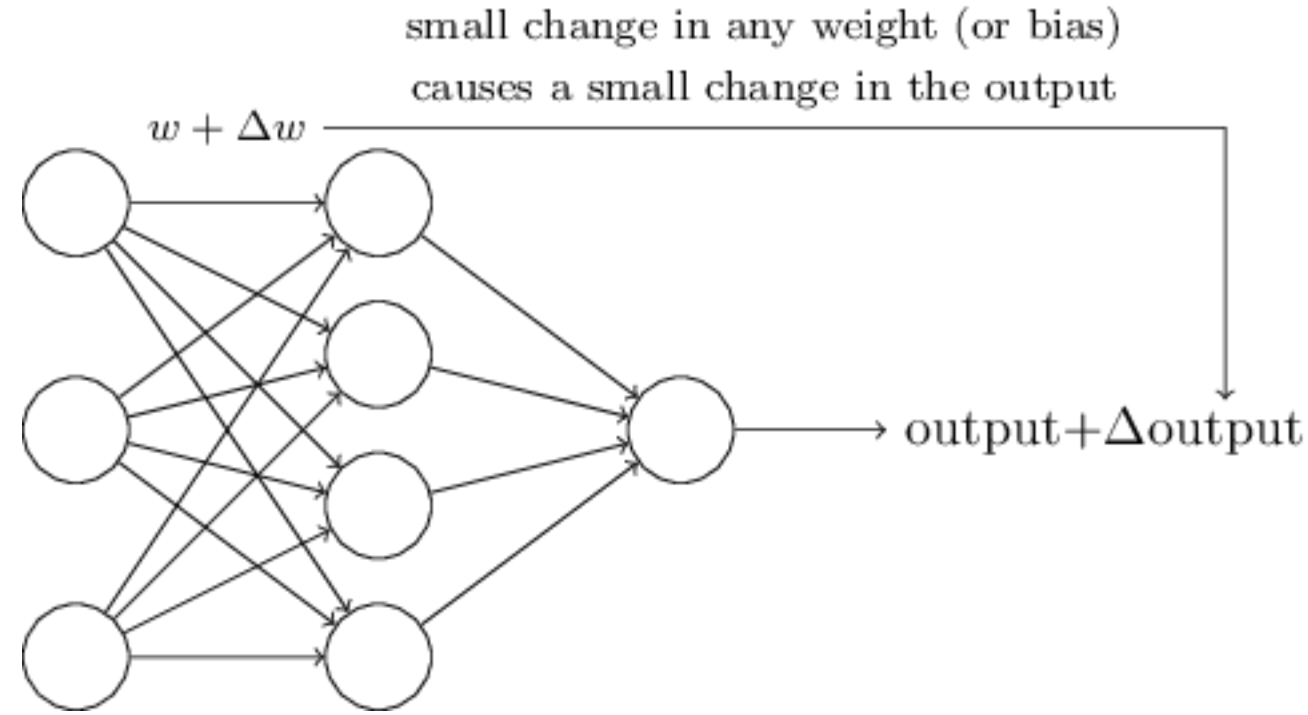
$$C(w,b) \equiv \frac{1}{2n}\sum_{x}\|y(x)-a\|^2$$

training examples
(mini-batches)



$$w_k \rightarrow w_k' = w_k - \eta\frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \eta\frac{\partial C}{\partial b_l}.$$

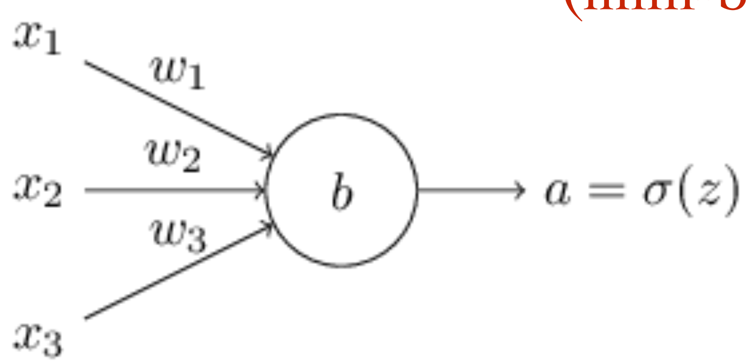


$$\Delta\text{output} \approx \sum_j \frac{\partial\,\text{output}}{\partial w_j}\Delta w_j + \frac{\partial\,\text{output}}{\partial b}\Delta b$$

Training a Neural Net

$$C(w,b) \equiv \frac{1}{2n}\sum_{x} \|y(x) - a\|^2$$

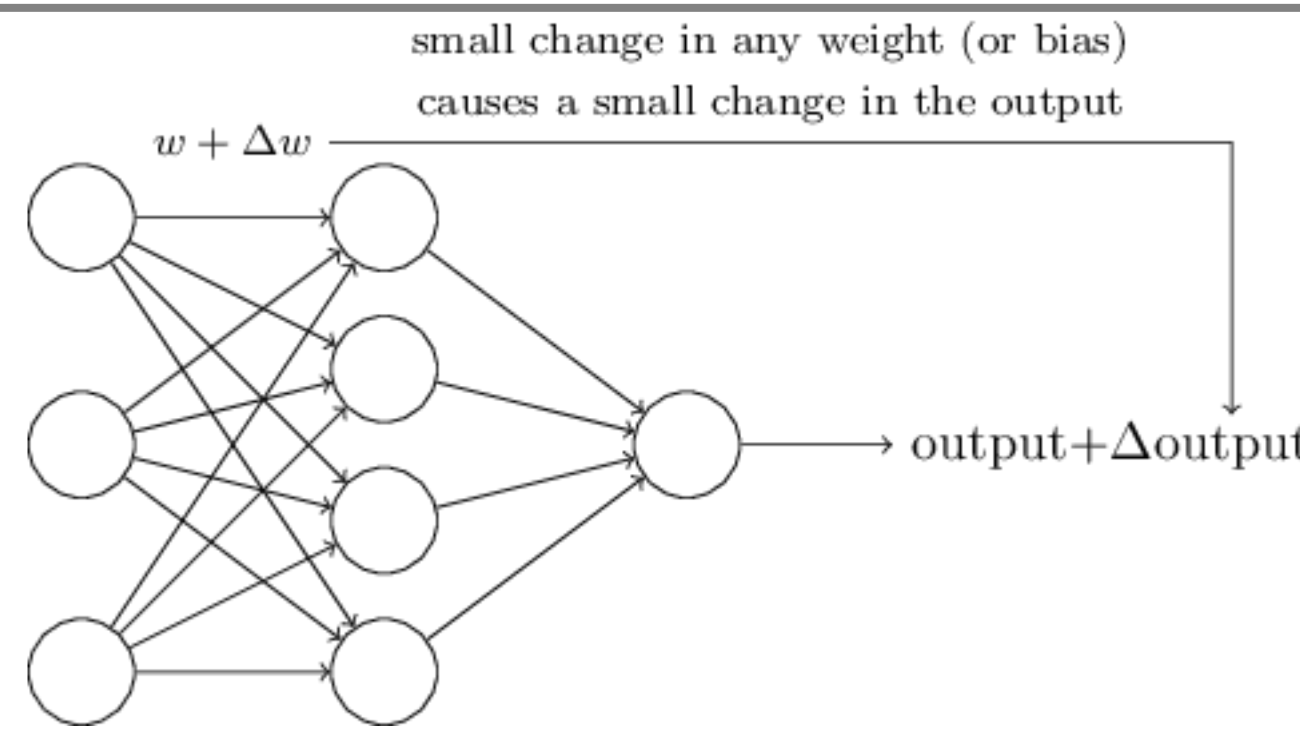
training examples
(mini-batches)



learning rate

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}.$$

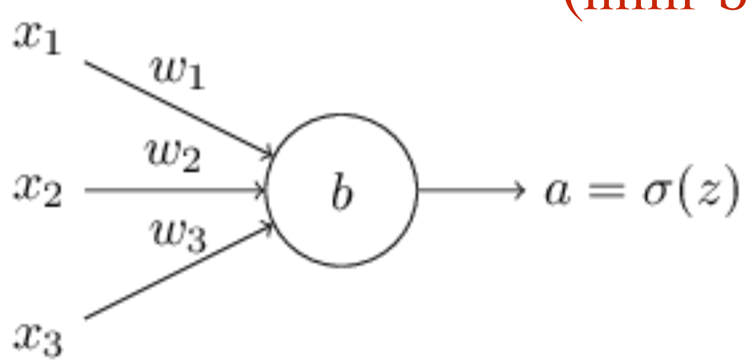


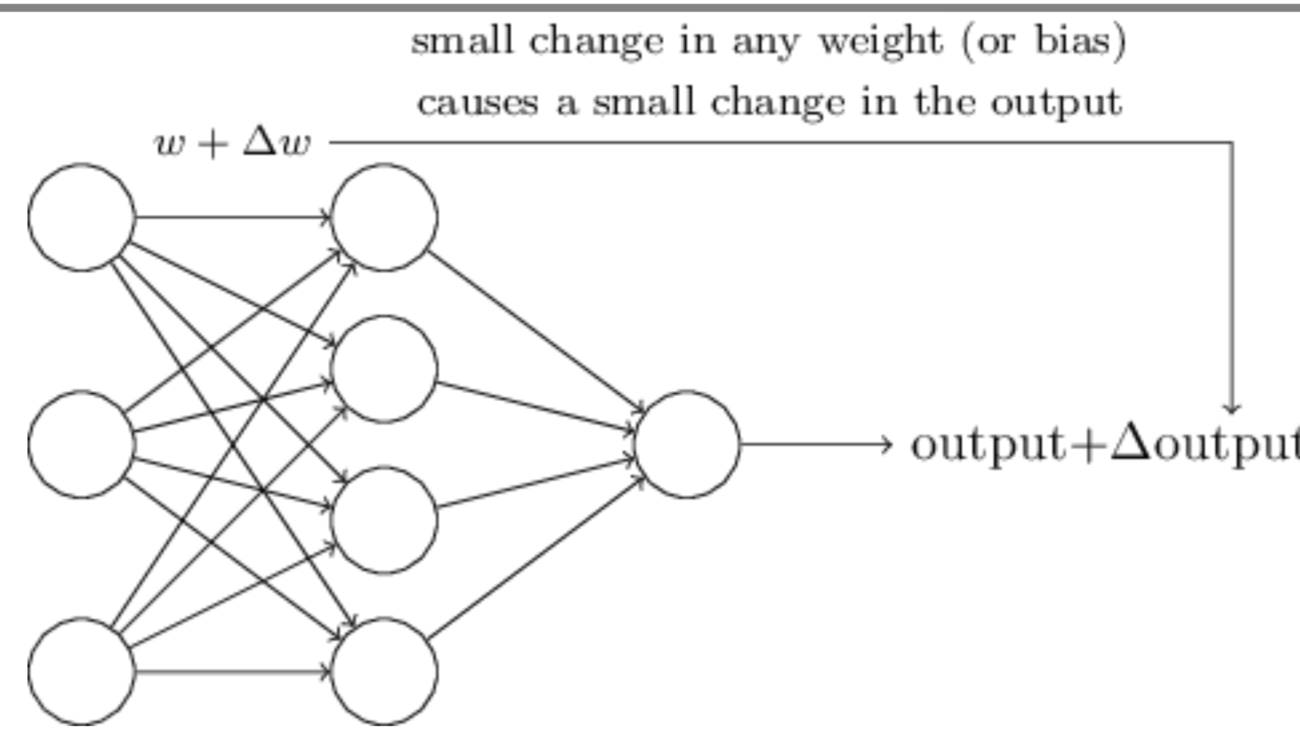
$$\Delta \text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b$$

Training a Neural Net

$$C(w,b) \equiv \frac{1}{2n} \sum_{x} \|y(x) - a\|^2$$

training examples (mini-batches)





$$\Delta \text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b$$

$$w_k \to w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

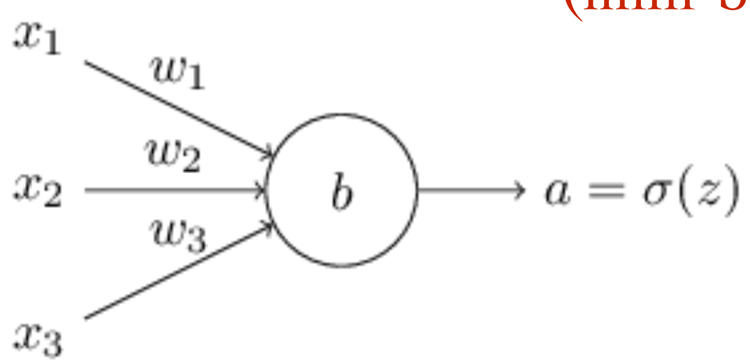
$$b_l \to b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

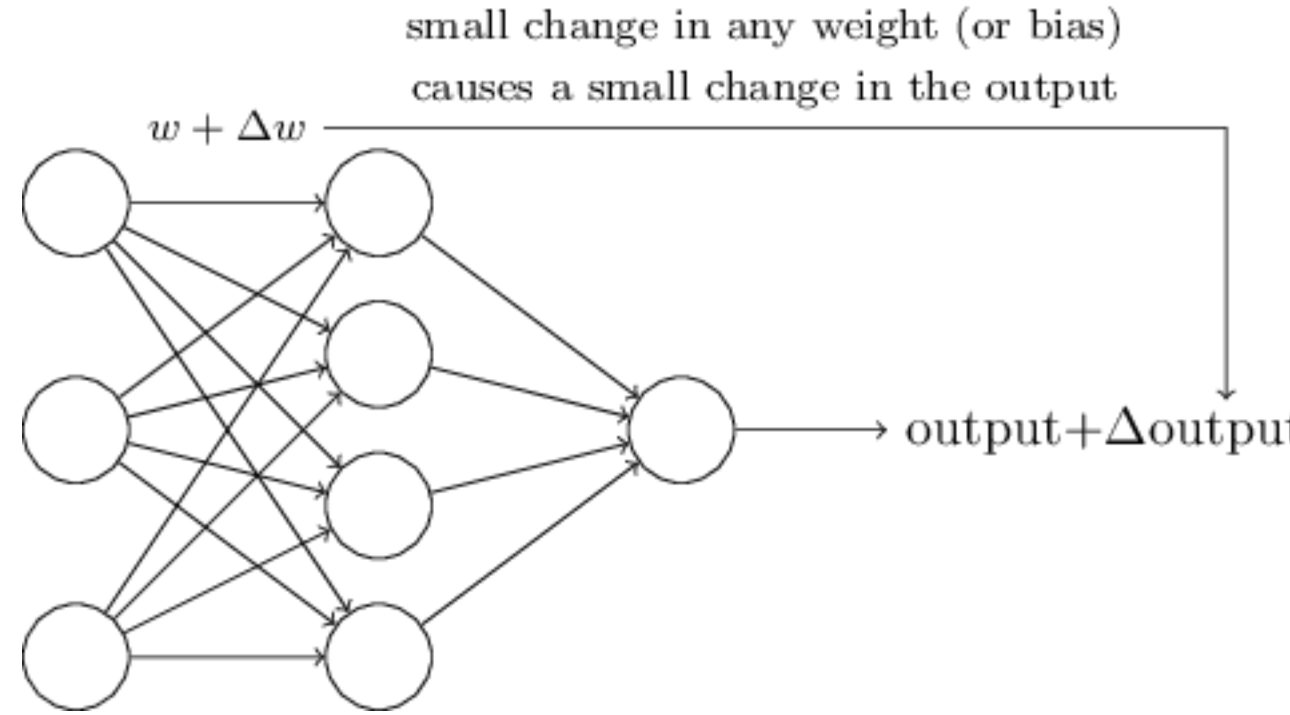
Stochastic Gradient Descent

Training a Neural Net

$$C(w,b) \equiv \frac{1}{2n}\sum_{x}\|y(x)-a\|^2$$

training examples (mini-batches)





$$\Delta\text{output} \approx \sum_j \frac{\partial\,\text{output}}{\partial w_j}\Delta w_j + \frac{\partial\,\text{output}}{\partial b}\Delta b$$

$$w_k \rightarrow w_k' = w_k - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

potentially difficult to compute

Stochastic Gradient Descent

Backpropagation

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Backpropagation

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Quadratic

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$ Quadratic

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$ Cross entropy

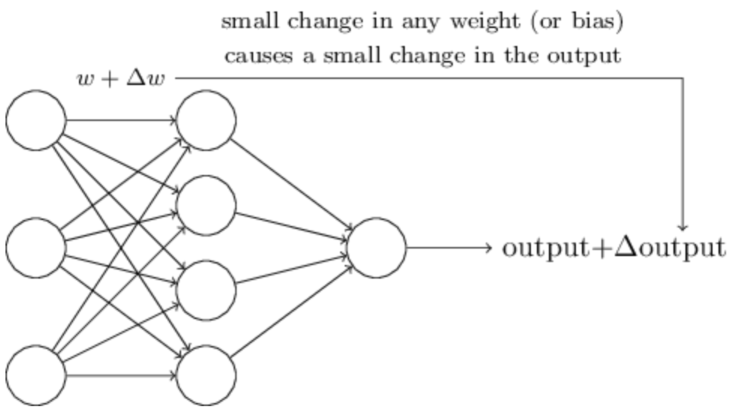
Backpropagation

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Quadratic

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$

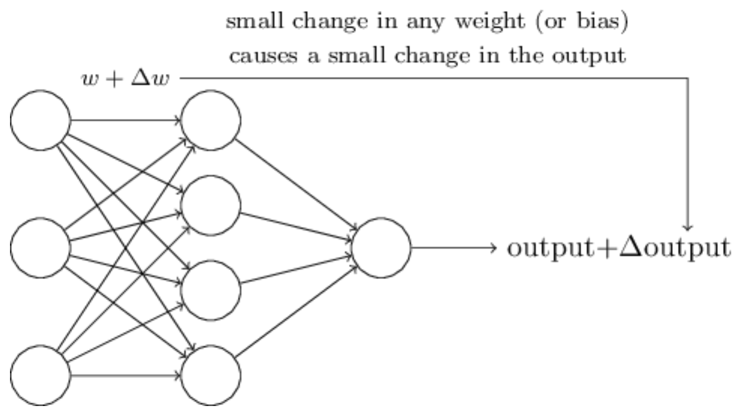
Cross entropy



Backpropagation

$$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x)-a\|^2$$ Quadratic

$$C = -\frac{1}{n}\sum_x \left[y \ln a + (1-y)\ln(1-a)\right]$$ Cross entropy



$$\boxed{\begin{aligned} \delta^L &= \nabla_a C \odot \sigma'(z^L) \\ \delta^l &= ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \\ \frac{\partial C}{\partial b^l_j} &= \delta^l_j \\ \frac{\partial C}{\partial w^l_{jk}} &= a^{l-1}_k \delta^l_j \end{aligned}}$$

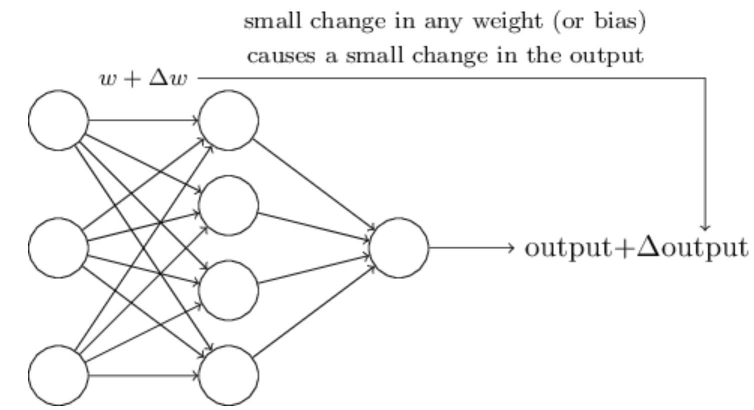
Backpropagation

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Quadratic

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$

Cross entropy



$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

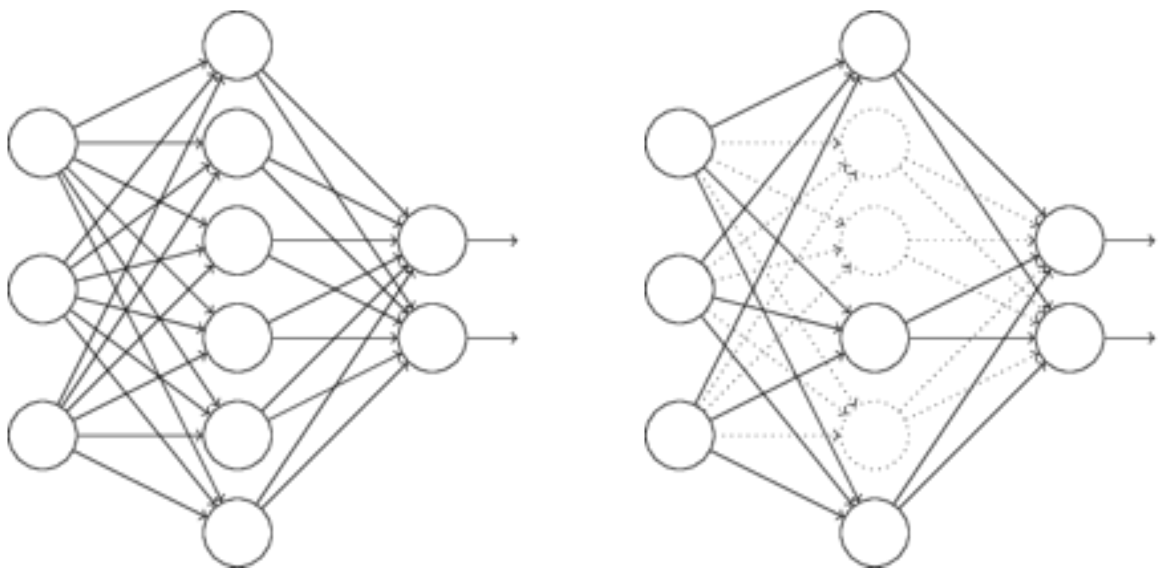
1. **Input x:** Set the corresponding activation a^1 for the input layer.
2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
3. **Output error δ^L:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.
4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

Guarding against overfitting and saturation

- Regularization (weight decay)

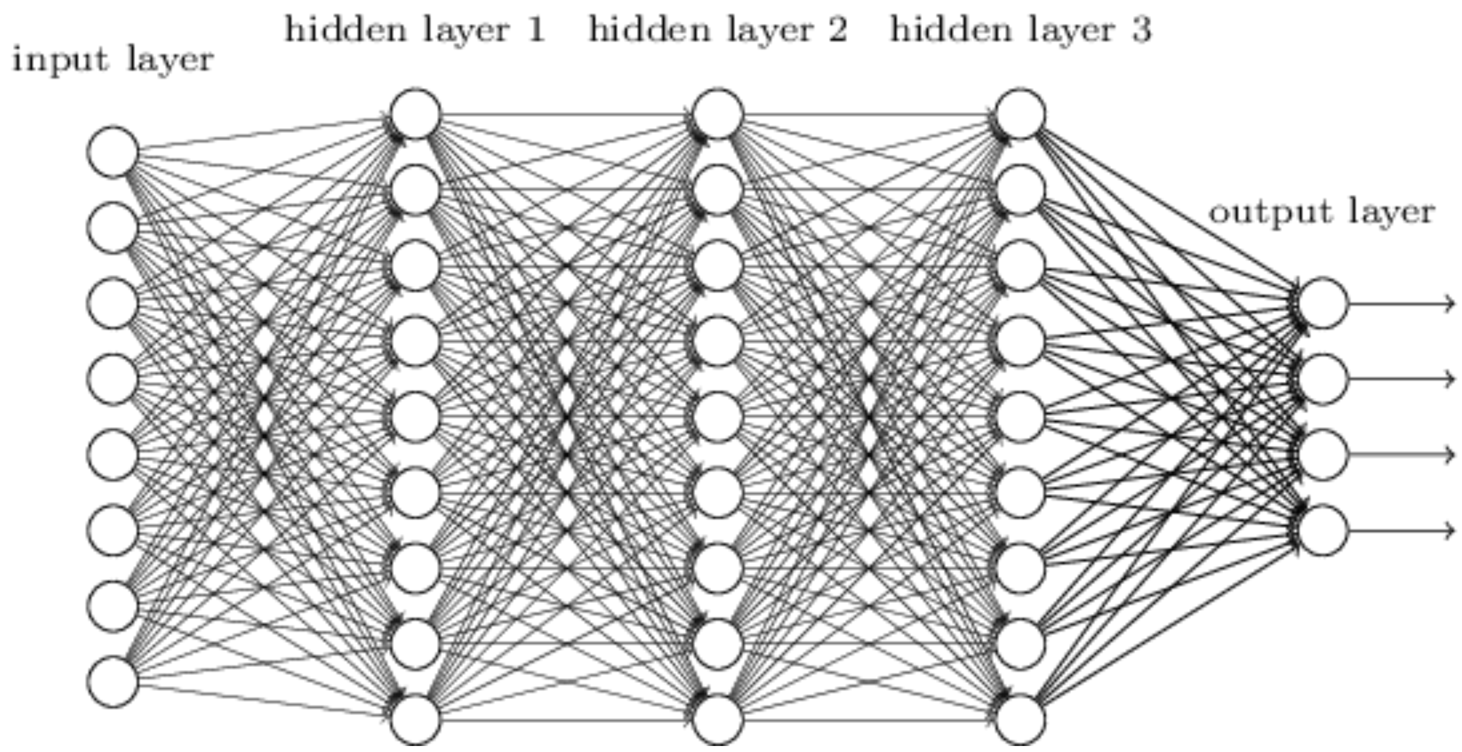
$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] + \frac{\lambda}{2n}\sum_w w^2$$

- Dropout

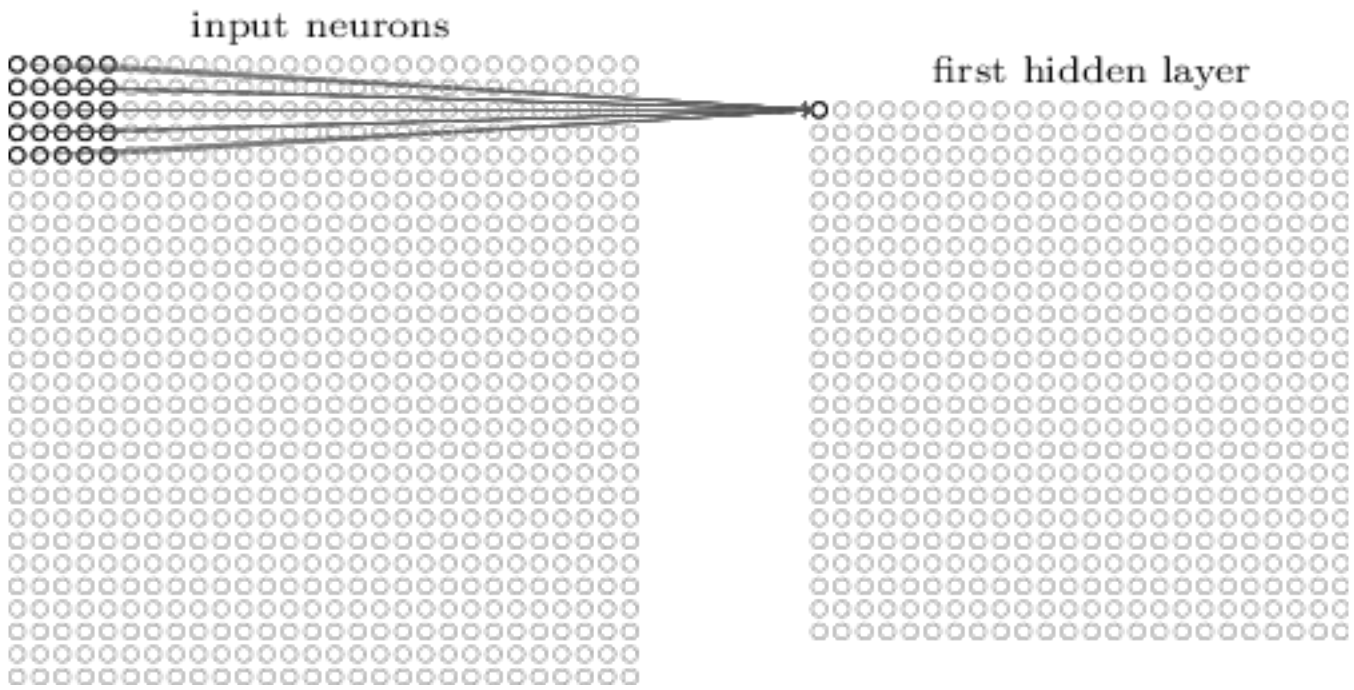


- Expanding the training data
  - reflections
  - translations
  - rotations
  - contrast
  - skewness

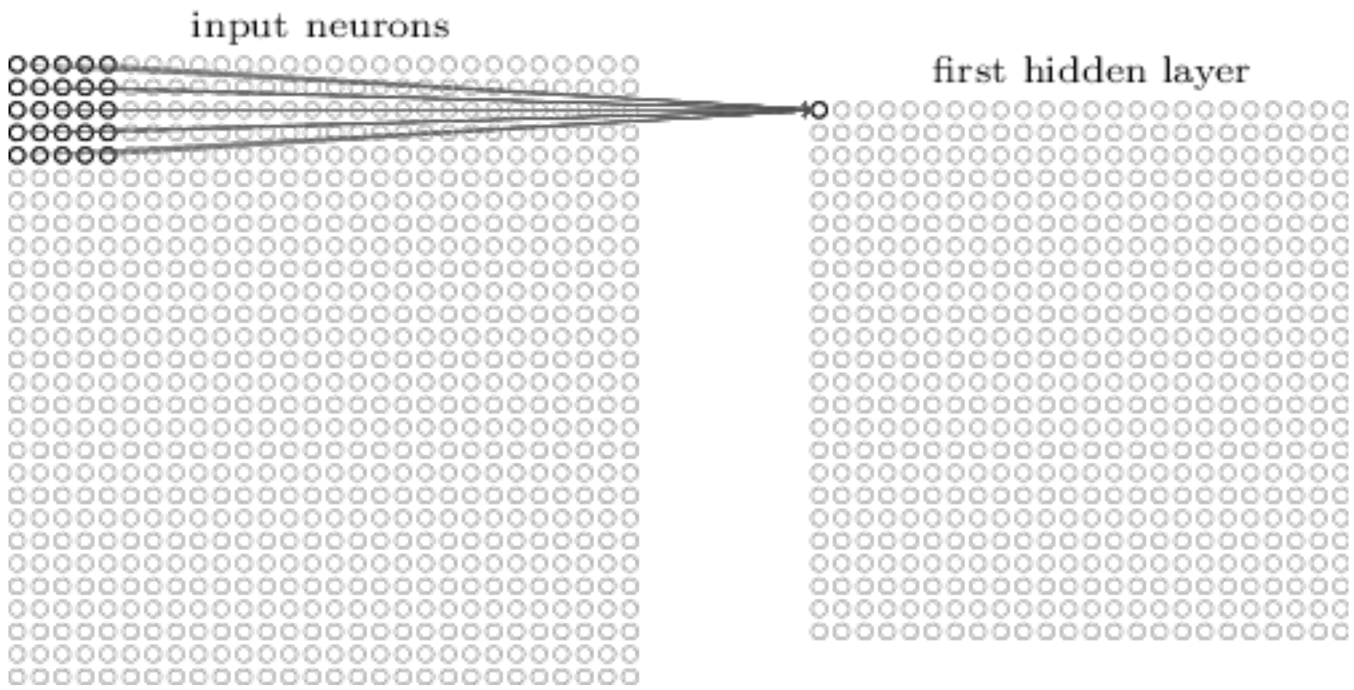
Convolutional nets

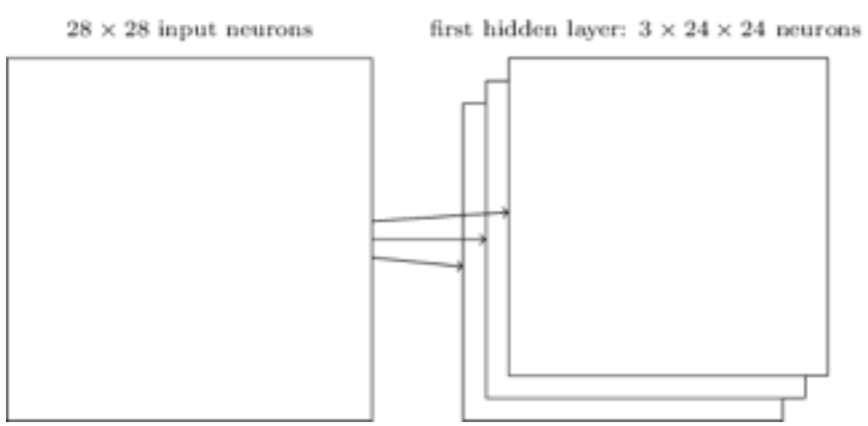


Convolutional nets

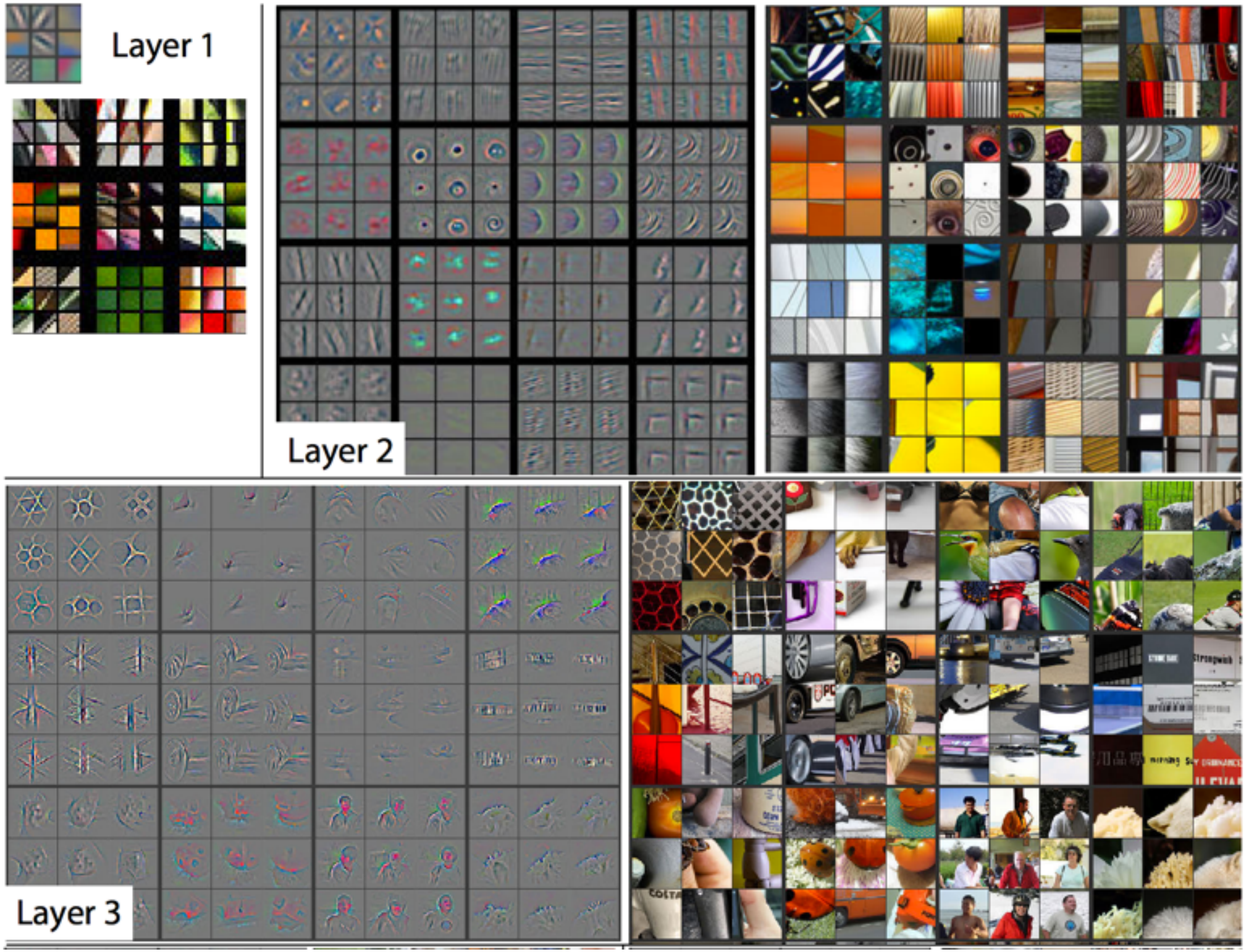


Convolutional nets



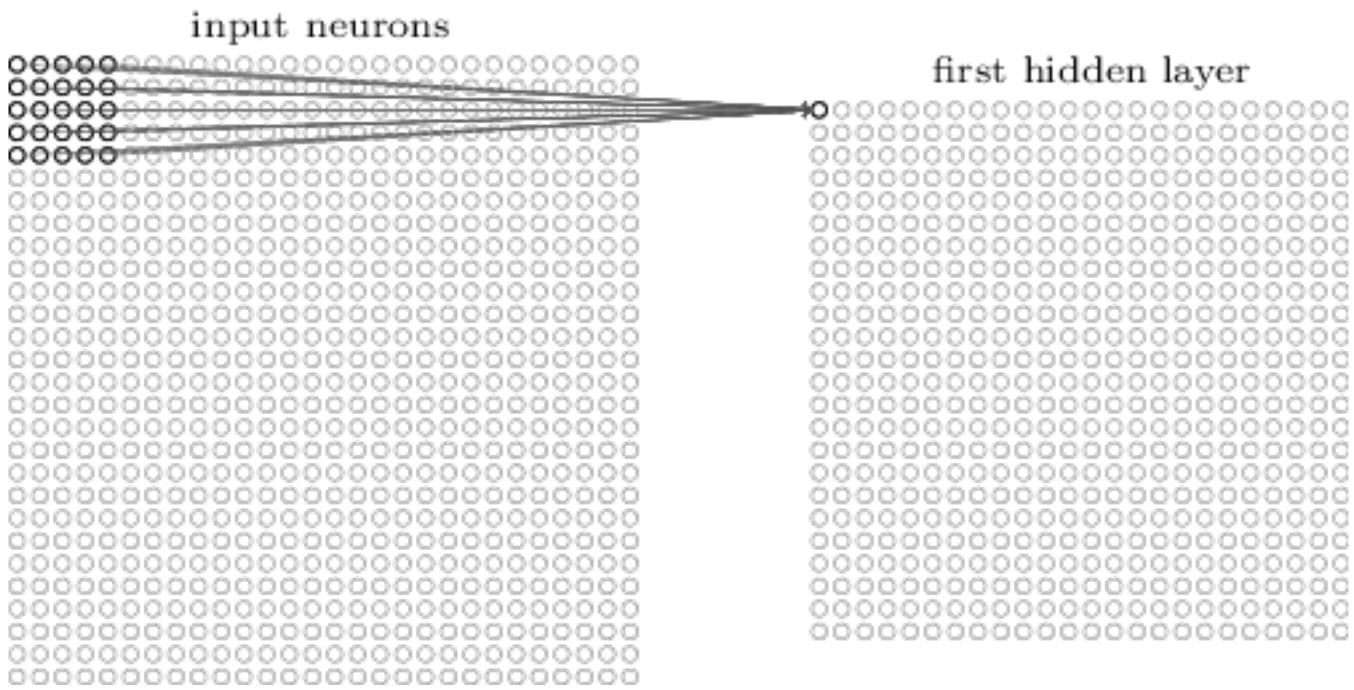


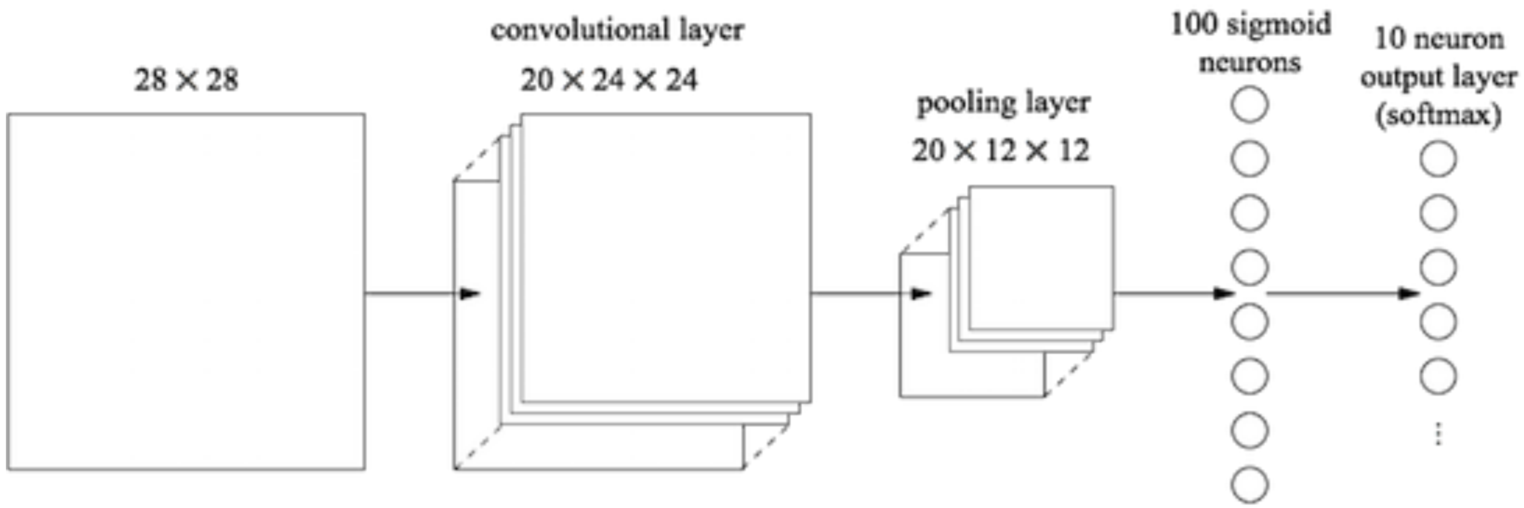
Convolutional nets

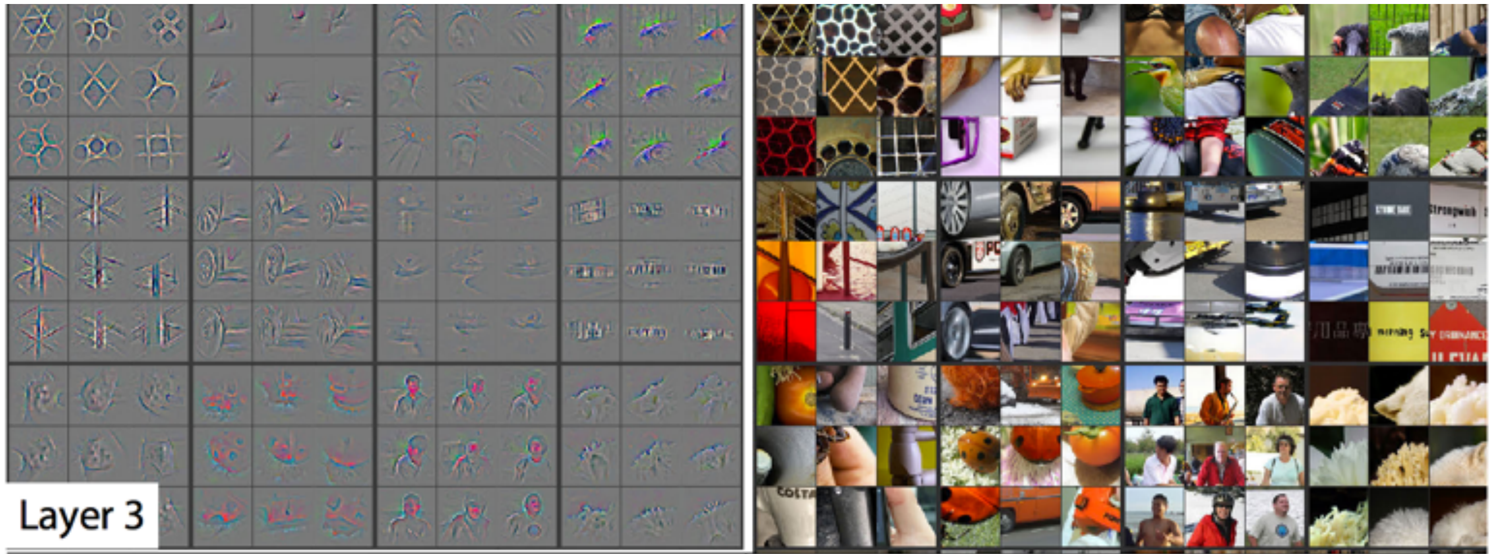


Zeiler & Fergus (2013)

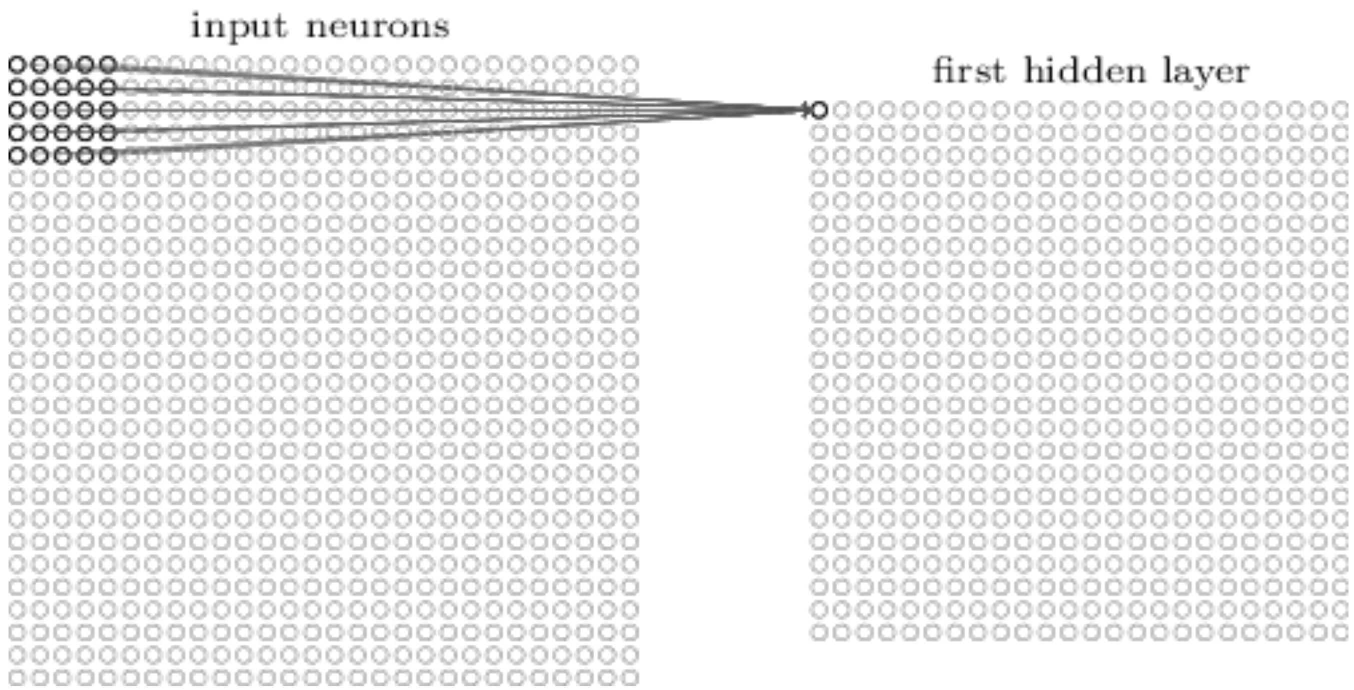
Convolutional nets

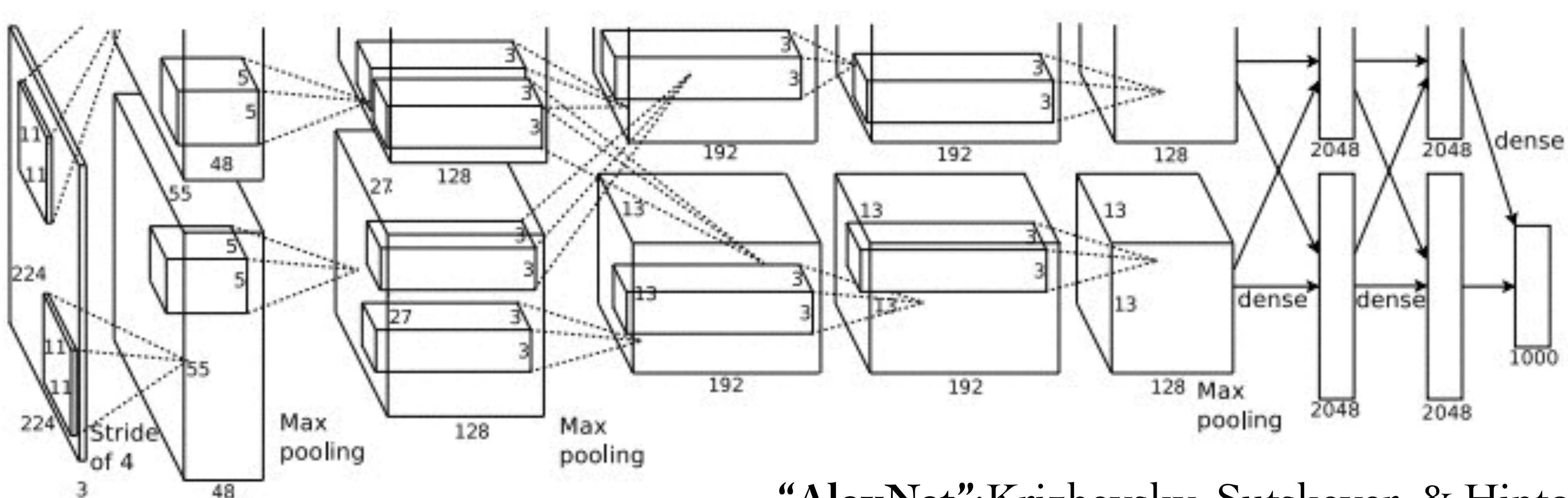






Convolutional nets





"AlexNet":Krizhevsky, Sutskever, & Hinton (2012)