NVIDIA

NEW YORK UNIVERSITY

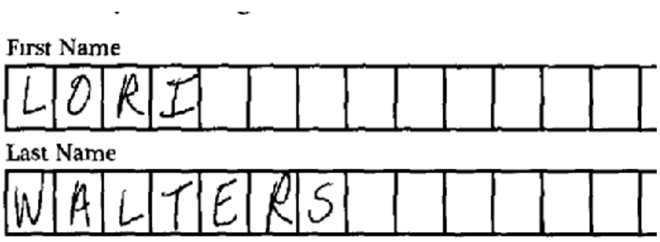
DLI Teaching Kit

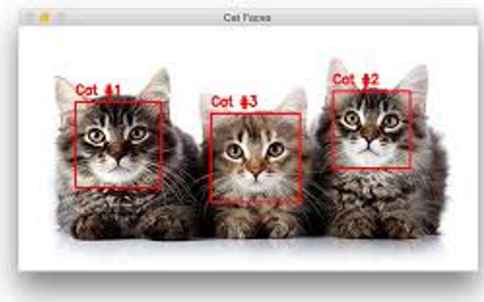
Introduction to Machine Learning

Deck credit: J. Seng

Machine Learning

- Machine Learning is the ability to teach a computer without explicitly programming it
- Examples are used to train computers to perform tasks that would be difficult to program





Types of Machine Learning

- Supervised Learning
  - Training data is labeled
  - Goal is correctly label new data
- Reinforcement Learning
  - Training data is unlabeled
  - System receives feedback for its actions
  - Goal is to perform better actions
- Unsupervised Learning
  - Training data is unlabeled
  - Goal is to categorize the observations

Applications of Machine Learning

- Handwriting Recognition
  - convert written letters into digital letters
- Language Translation
  - translate spoken and or written languages (e.g. Google Translate)
- Speech Recognition
  - convert voice snippets to text (e.g. Siri, Cortana, and Alexa)
- Image Classification
  - label images with appropriate categories (e.g. Google Photos)
- Autonomous Driving
  - enable cars to drive (e.g. Tesla,

Features in Machine Learning

- Features are the observations that are used to form predictions
  - For image classification, the pixels are the features
  - For voice recognition, the pitch and volume of the sound samples are the features
  - For autonomous cars, data from the cameras, range sensors, and GPS are features

- Extracting relevant features is important for building a model
  - Time of day is an irrelevant feature when classifying images
  - Time of day is relevant when classifying emails because SPAM often occurs at night

- Common Types of Features in Robotics
  - Pixels (RGB data)
  - Depth data (sonar, laser rangefinders)
  - Movement (encoder values)
  - Orientation or Acceleration (Gyroscope, Accelerometer, Compass)

Measuring Success for Classification

- True Positive: Correctly identified as relevant
- True Negative: Correctly identified as not relevant
- False Positive: Incorrectly labeled as relevant
- False Negative: Incorrectly labeled as not relevant

Example: Identify Cats

Prediction:

Image:

True Positive

True Negative

False Negative

False Positive

Images from the STL-10 dataset

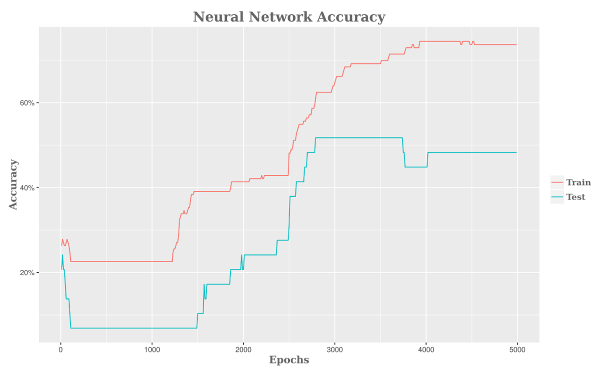
Precision, Recall, and Accuracy

- Precision
  - Percentage of positive labels that are correct
  - Precision = (# true positives) / (# true positives + # false positives)
- Recall
  - Percentage of positive examples that are correctly labeled
  - Recall = (# true positives) / (# true positives + # false negatives)
- Accuracy
  - Percentage of correct labels
  - Accuracy = (# true positives + # true negatives) / (# of samples)

Training and Test Data

- Training Data
  - data used to learn a model
- Test Data
  - data used to assess the accuracy of model

- Overfitting
  - Model performs well on training data but poorly on test data



Bias and Variance

- Bias: expected difference between model's prediction and truth
- Variance: how much the model differs among training sets

- Model Scenarios
  - High Bias: Model makes inaccurate predictions on training data
  - High Variance: Model does not generalize to new datasets
  - Low Bias: Model makes accurate predictions on training data
  - Low Variance: Model generalizes to new datasets

Supervised Learning Algorithms

- Linear Regression
- Decision Trees
- Support Vector Machines
- K-Nearest Neighbor
- Neural Networks

Supervised Learning Frameworks

Tool	Uses	Language
Scikit-Learn	Classification, Regression, Clustering	Python
Spark MLlib	Classification, Regression, Clustering	Scala, R, Java
Weka	Classification, Regression, Clustering	Java
Caffe	Neural Networks	C++, Python
TensorFlow	Neural Networks	Python

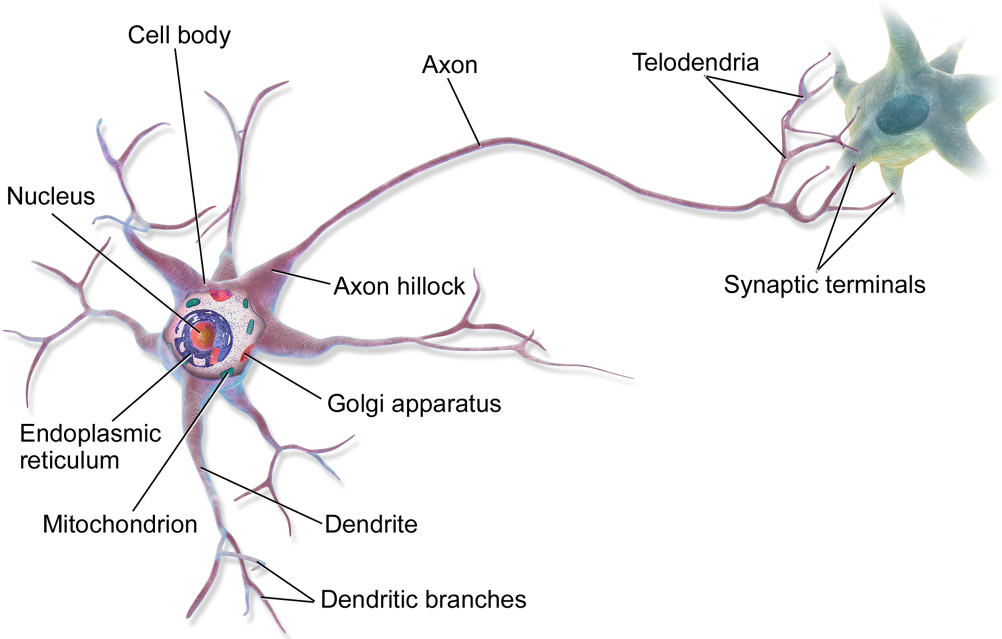
NVIDIA

NEW YORK UNIVERSITY

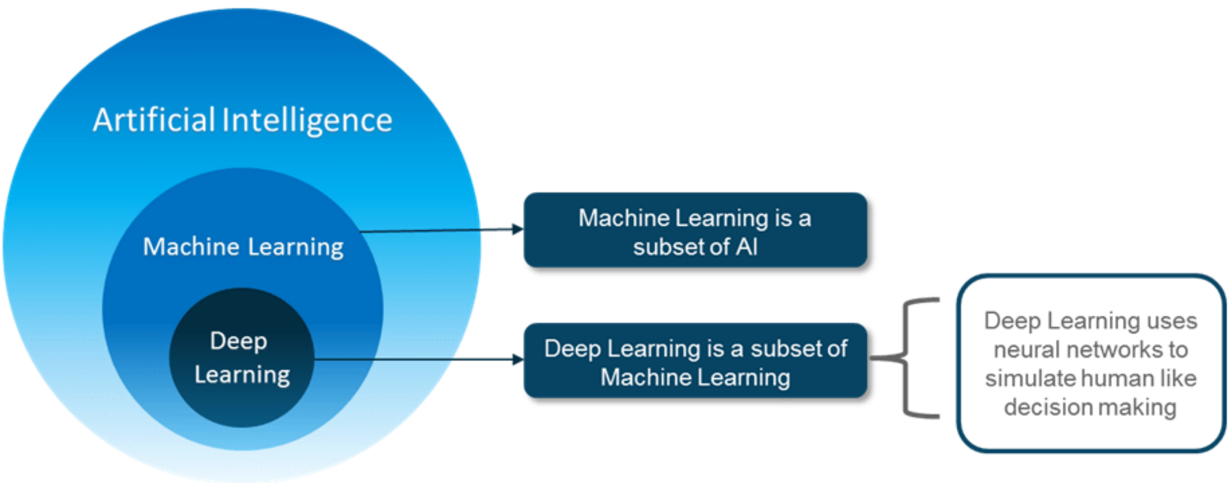
DLI Teaching Kit

Introduction to Neural Networks

Recall: Biological Inspiration



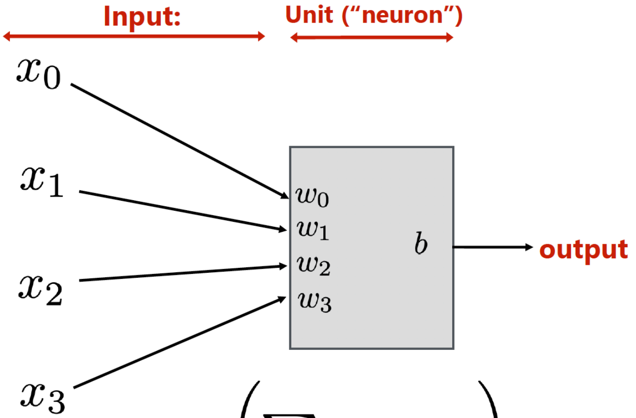
AI Taxonomy



Deep Learning

A basic unit:

Unit with *n* inputs described by *n*+1 parameters (weights + bias)



$$f\left(\sum_i x_i w_i + b\right)$$

Example f: rectified linear unit (ReLU)

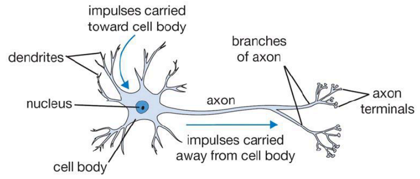
$f(x) = max(0, x)$

Basic computational interpretation:

It's just a circuit!

Biological inspiration:

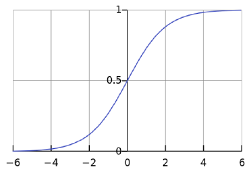
unit output corresponds loosely to activation of neuron



Machine learning interpretation:

binary classifier: interpret output as the probability of one class

$$f(x) = \frac{1}{1 + e^{-x}}$$



Deep Learning Leaders

Type to enter a caption.

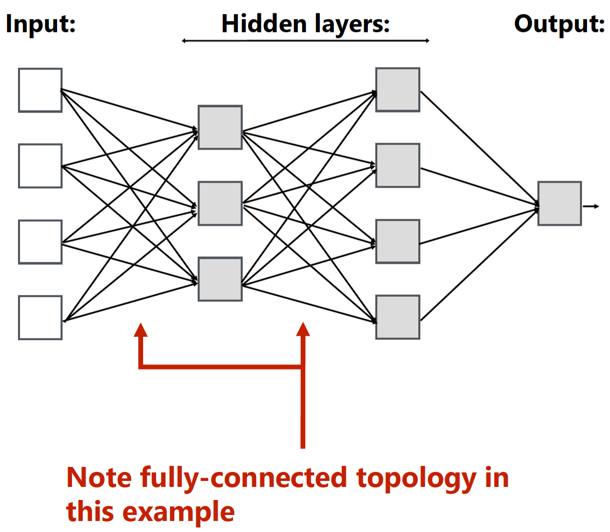
- **2019 Turing Award Winners**
  - **Yoshua Bengio**
  - **Geoff Hinton**
  - **Yann LeCun**

Two Distinct Issues with Deep Networks

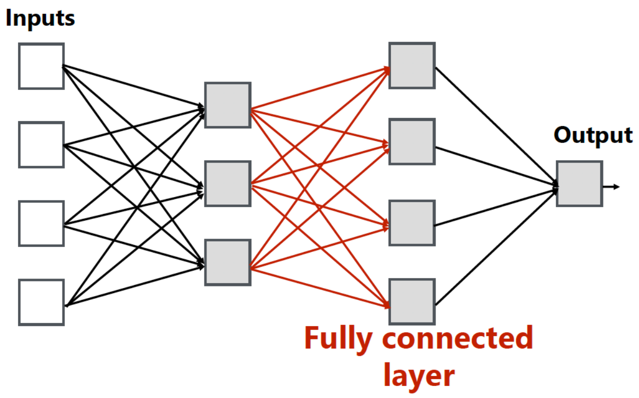
- Evaluation/Inference
  - often takes milliseconds
- Training
  - often takes hours, days, weeks

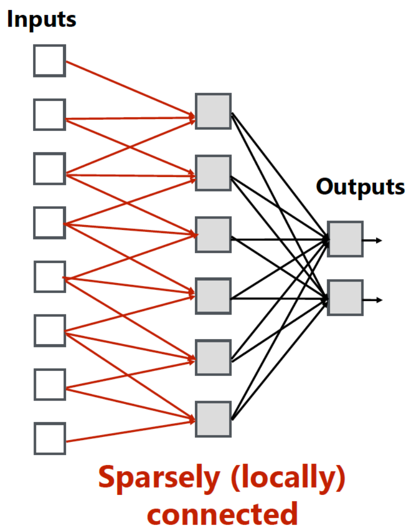
What is a deep neural network? topology

This network has: 4 inputs, 1 output, 7 hidden units
"Deep" > one hidden layer
Hidden layer 1: 3 units x (4 weights + 1 bias) = 15 parameters
Hidden layer 2: 4 units x (3 weights + 1 bias) = 16 parameters

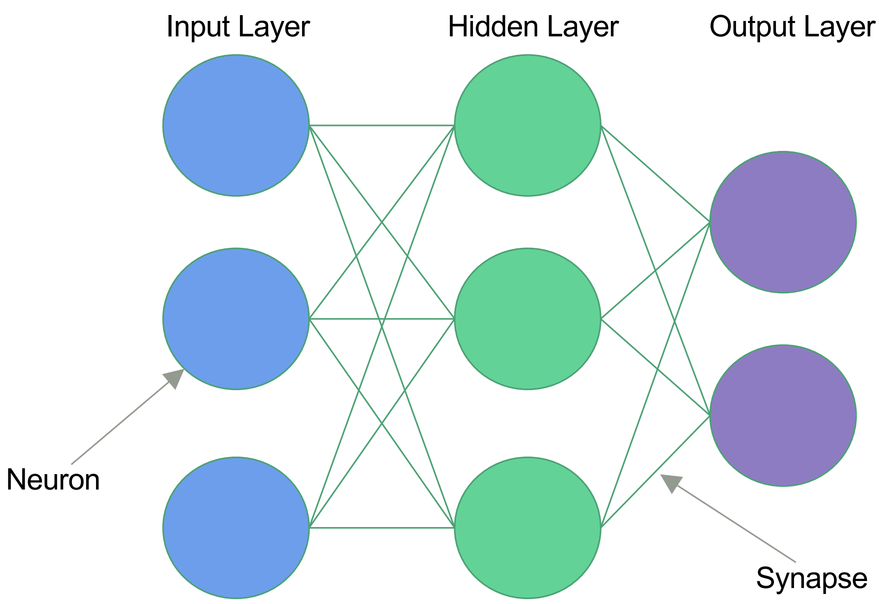


Deep Neural Networks: Topology





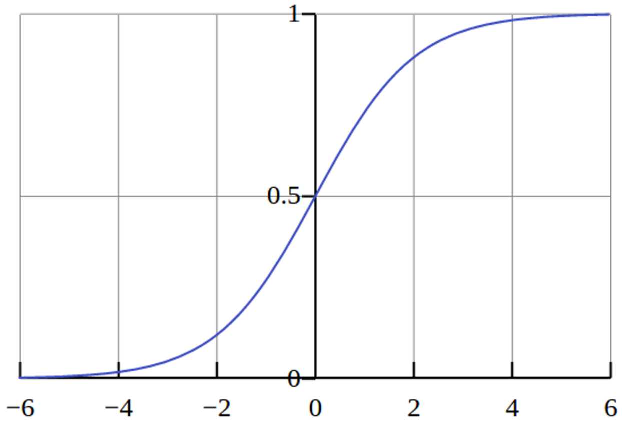
Neural Network Architecture



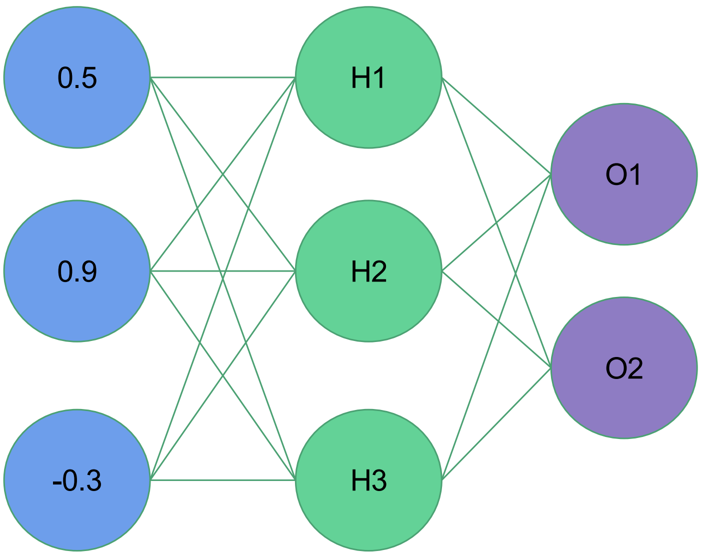
Activation Functions

– Activation Functions are applied to the inputs at each neuron
  – A common activation function is the Sigmoid

$$S(t) = \frac{1}{1 + e^{-t}}$$



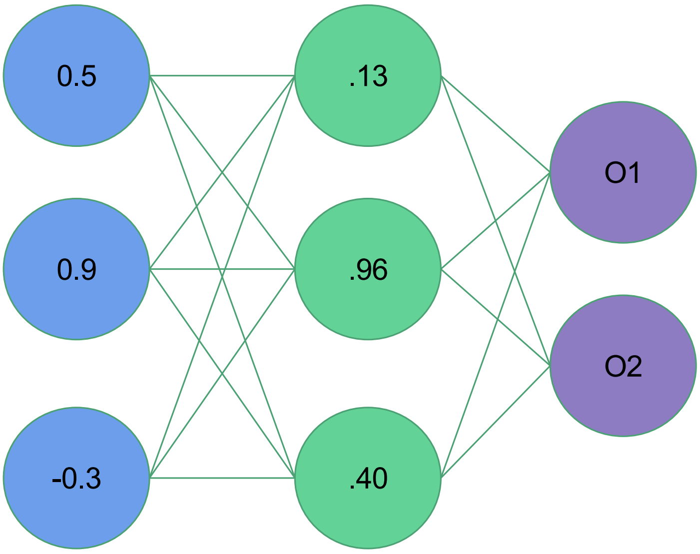
Inference



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

Inference

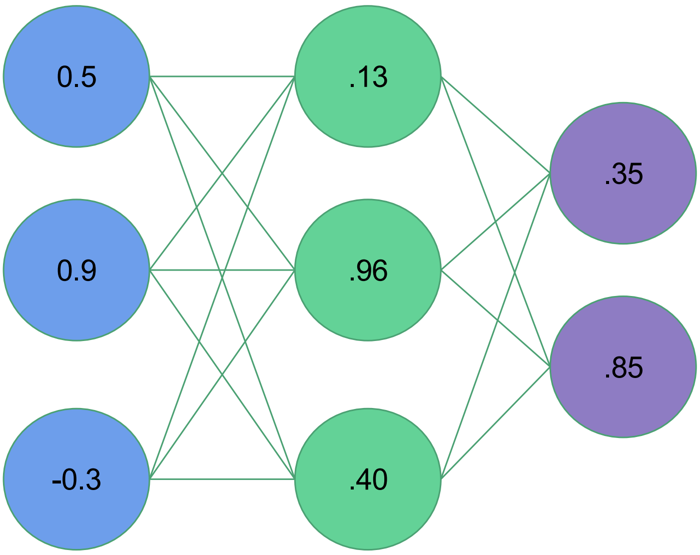


H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

H1 = S(0.5 * 1.0 + 0.9 * -2.0 + -0.3 * 2.0) = S(-1.9) = .13
H2 = S(0.5 * 2.0 + 0.9 * 1.0 + -0.3 * -4.0) = S(3.1) = .96
H3 = S(0.5 * 1.0 + 0.9 * -1.0 + -0.3 * 0.0) = S(-0.4) = .40

Inference



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

O1 = S(.13 * -3.0 + .96 * 1.0 + .40 * -3.0) = S(-.63) = .35
O2 = S(.13 * 0.0 + .96 * 1.0 + .40 * 2.0) = S(1.76) = .85

Matrix Formulation

H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

Hidden Layer Weights, Inputs, Hidden Layer Outputs

$$S\left(\begin{bmatrix} 1.0 & -2.0 & 2.0 \\ 2.0 & 1.0 & -4.0 \\ 1.0 & -1.0 & 0.0 \end{bmatrix} * \begin{bmatrix} 0.5 \\ 0.9 \\ -0.3 \end{bmatrix}\right) = S\left(\begin{bmatrix} -1.9 & 3.1 & -0.4 \end{bmatrix}\right) = \begin{bmatrix} .13 & .96 & 0.4 \end{bmatrix}$$

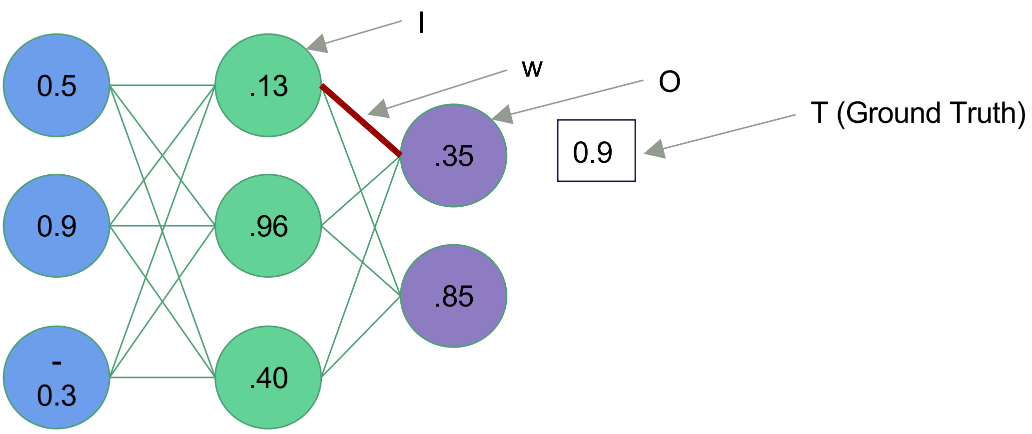
Training Neural Networks

- Procedure for training Neural Networks
  - Perform inference on the training set
  - Calculate the error between the predictions and actual labels of the training set
  - Determine the contribution of each Neuron to the error
  - Modify the weights of the Neural Network to minimize the error
- Error contributions are calculated using Backpropagation
- Error minimization is achieved with Gradient Descent

Backpropagation

– Problem: Which weights should be updated and by how much?
  – Insight: Use the derivative of the error with respect to weight to assign "blame"

Backpropagation Example



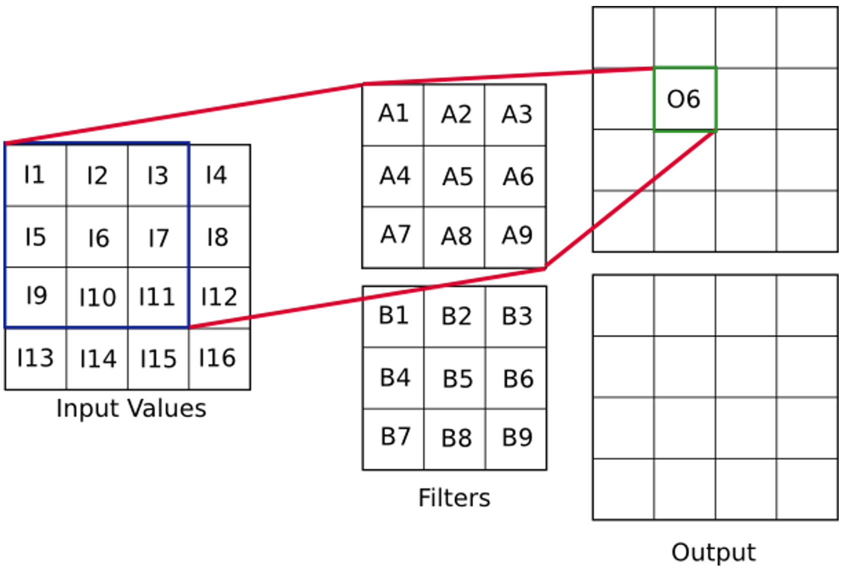
$$\frac{\partial E}{\partial w} = I \cdot (O - T) \cdot O \cdot (1 - O)$$

$$\frac{\partial E}{\partial w} = .13 \cdot (.35 - .9) \cdot .35 \cdot (1 - .35)$$

Gradient Descent

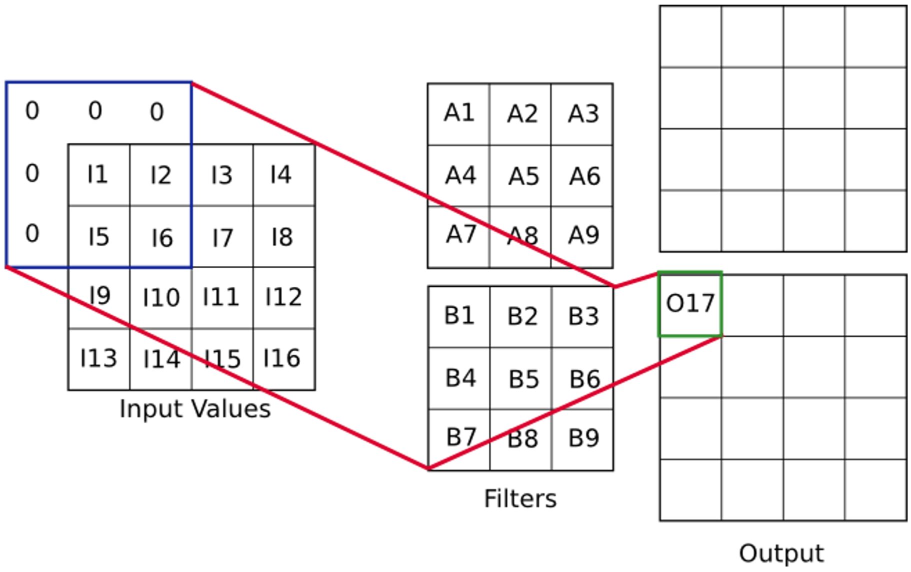
- Gradient Descent minimizes the neural network's error
  - At each time step the error of the network is calculated on the training data
  - Then the weights are modified to reduce the error
- Gradient Descent terminates when
  - The error is sufficiently small
  - The max number of time steps has been exceeded

Convolutional Neural Networks



$$O_6 = A_1 \cdot I_1 + A_2 \cdot I_2 + A_3 \cdot I_3$$
$$+A_4 \cdot I_5 + A_5 \cdot I_6 + A_6 \cdot I_7$$
$$+A_7 \cdot I_9 + A_8 \cdot I_{10} + A_9 \cdot I_{11}$$

Convolutional Neural Networks



$$O_{17} = B_5 \cdot I_1 + B_6 \cdot I_2 + B_8 \cdot I_5 + B_9 \cdot I_6$$

CNN Intuition

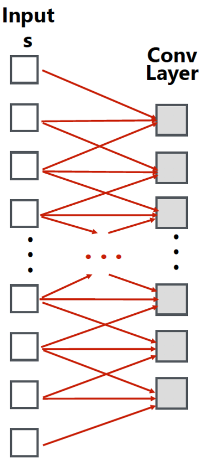
– A combination of two components:
  – feature extraction part
  – classification part
– The convolution + pooling layers perform feature extraction.
– Example
  – Given an image, the convolution layer detects features such as two eyes, long ears, four legs, a short tail and so on.
  – The fully connected layers then act as a classifier on top of these features and assign a probability for the input image being a dog.

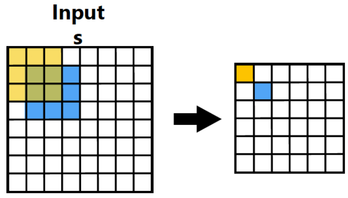
Image Convolution: 3x3

```
int WIDTH = 1024;
int HEIGHT = 1024;
float input[(WIDTH+2) * (HEIGHT+2)];
float output[WIDTH * HEIGHT];

float bias = 0.f;
float weights[] = {1.0/9, 1.0/9, 1.0/9,
                   1.0/9, 1.0/9, 1.0/9,
                   1.0/9, 1.0/9, 1.0/9};


for (int j=0; j<HEIGHT; j++) {
  for (int i=0; i<WIDTH; i++) {
    float tmp = bias;
    for (int jj=0; jj<3; jj++)
      for (int ii=0; ii<3; ii++)
        tmp += input[(j+jj)*(WIDTH+2) + (i+ii)] * weights[jj*3 + ii];
    output[j*WIDTH + i] = tmp;
  }
}
```





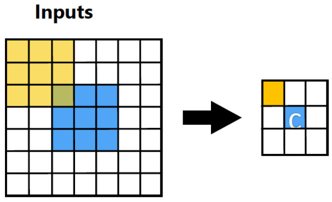
Convolutional layer: locally connected AND all units in layer share the same parameters (same weights + same bias):
(note: network diagram only shows links due to one iteration of ii loop)

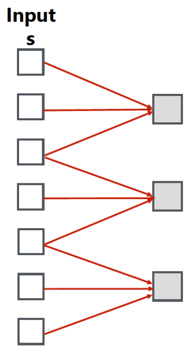
Strided 3x3 Convolution

```
int WIDTH = 1024;
int HEIGHT = 1024;
int STRIDE = 2;
float input[(WIDTH+2) * (HEIGHT+2)];
float output[(WIDTH/STRIDE) * (HEIGHT/STRIDE)];

float bias = 0.f;
float weights[] = {1.0/9, 1.0/9, 1.0/9,
                   1.0/9, 1.0/9, 1.0/9,
                   1.0/9, 1.0/9, 1.0/9};


for (int j=0; j<HEIGHT; j+=STRIDE) {
  for (int i=0; i<WIDTH; i+=STRIDE) {
    float tmp = bias;
    for (int jj=0; jj<3; jj++)
      for (int ii=0; ii<3; ii++) {
         tmp += input[(j+jj)*(WIDTH+2) + (i+ii)] * weights[jj*3 + ii];
      output[(j/STRIDE)*WIDTH + (i/STRIDE)] = tmp;
  }
```



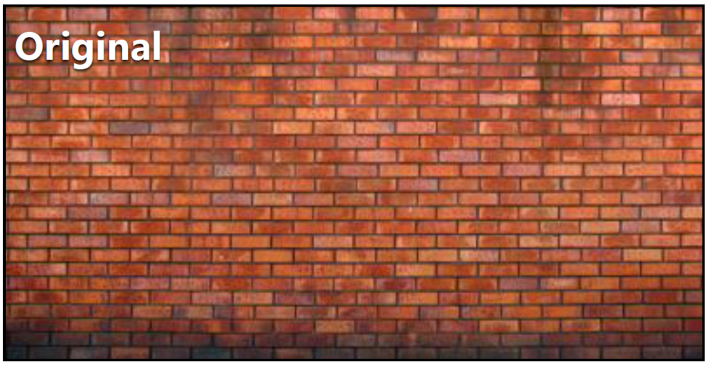


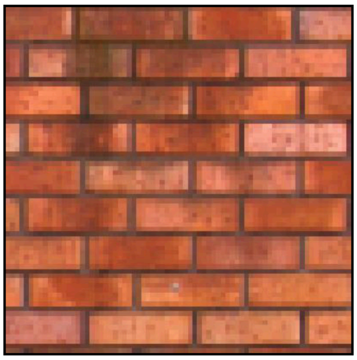
Convolutional layer with stride 2

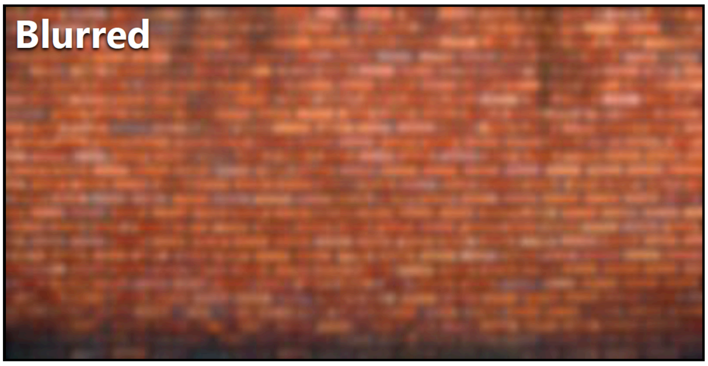
What does convolution using these filter weights do?

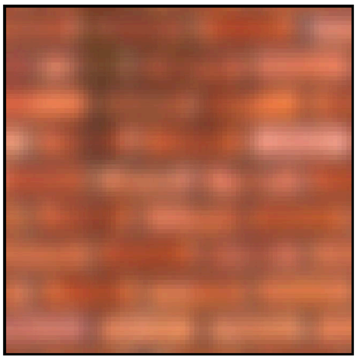
$$\begin{bmatrix} .075 & .124 & .075 \\ .124 & .204 & .124 \\ .075 & .124 & .075 \end{bmatrix}$$

"Gaussian Blur"









What does convolution with these filters do?

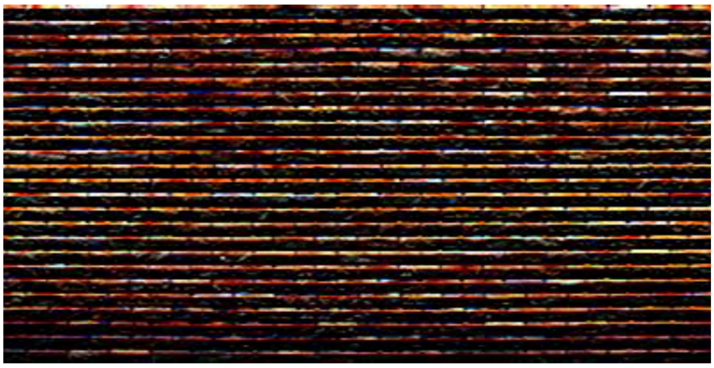
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Extracts horizontal gradients

Extracts vertical gradients

Gradient Detection Filter

Horizontal gradients



Vertical gradients

Note: You can think of a filter as a "detector" of a pattern, and the magnitude of a pixel in the output image as the "response" of the filter to the region surrounding each pixel in the input image

Emerging architectures for deep learning

- NVIDIA Pascal (most recent GPU)
  - Adds double-throughput 16-bit floating point ops
  - Feature that is already common on mobile GPUs
- Google TensorFlow Processing Unit
  - Hardware accelerator for array computations
  - Used in Google data centers
- Apple Neural Engine
  - On A11 & A12 processor chips in iPhones & iPads
- NOR Networks
  - Reduce weights & data to single bits
- FPGAs, ASICs?
  - Microsoft"BrainWave"on FPGAs within data centers
  - Not new: FPGA solutions have been explored for years
- And a million startups…

Programming frameworks for deep learning

- Heavyweight processing (low-level kernels) carried out by target-optimized libraries (NVIDIA cuDNN, Intel MKL)
- Popular frameworks use these kernel libraries
  - Caffe, Torch, TensorFlow, MxNet, Keras, …
- DNN application development = constructing novel network topologies
  - Programming by constructing networks
  - Significant interest in new ways to express network construction

Max-Pooling

1	0	2	2
1	3	0	1
3	1	4	1
2	0	2	1

2x2 Max Pooling →

3	2
3	4

Training/evaluating deep neural networks

Technique leading to many high-profile AI advances in recent years

Speech recognition/natural language processing

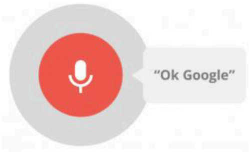
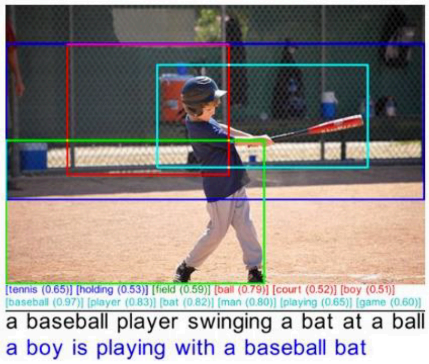
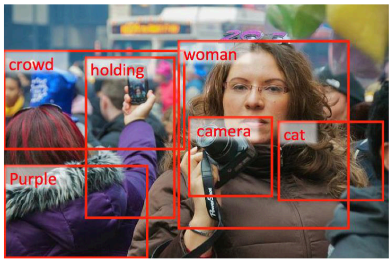


Image interpretation and understanding





DLI Teaching Kit

Thank you