Machine Learning Primer

Polina Binder

Slides credit: MIT Deep Learning for Self Driving Cars, Toronto CSC 321, ICML reinforcement learning tutorial

Outline

- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

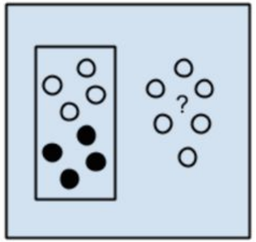
Outline

- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

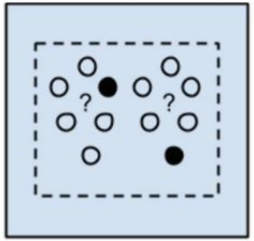
What is machine learning?

- A class of algorithms that allows us to infer rules and parameters based on example data.
- In contrast to: hand coded rules

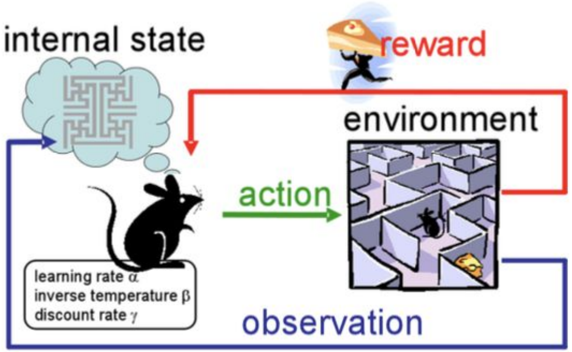
Types of Machine Learning



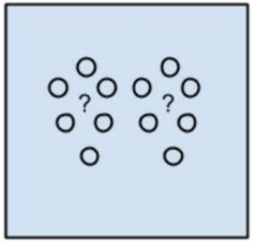
Supervised Learning



Semi-Supervised Learning

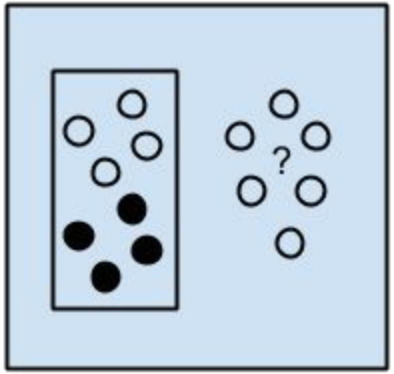


Reinforcement Learning



Unsupervised Learning

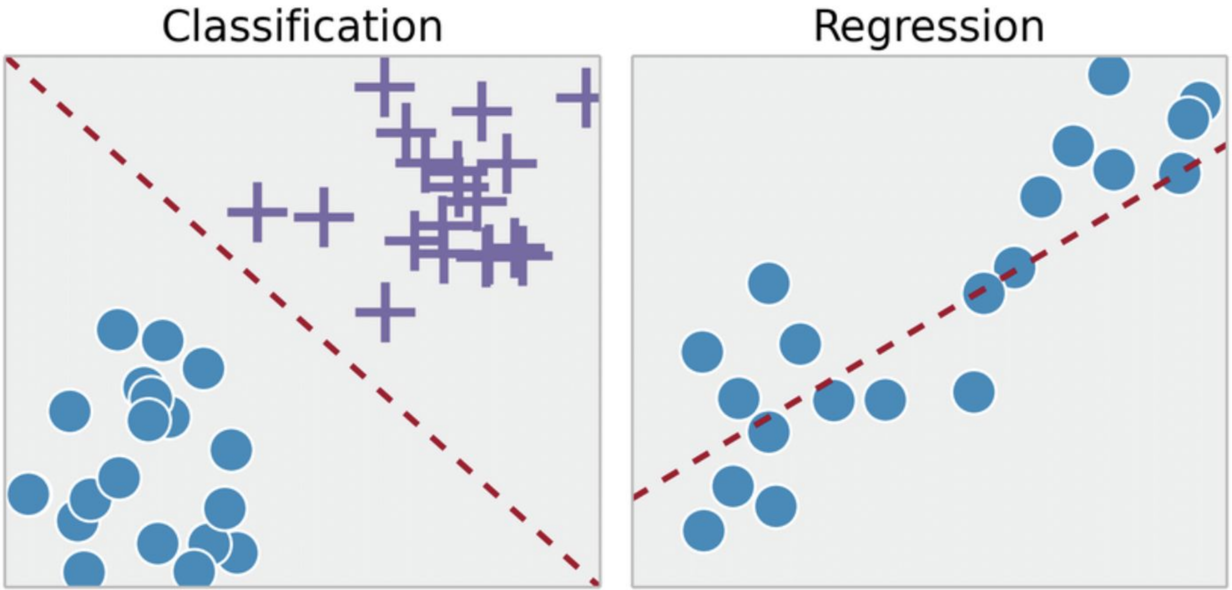
Supervised Learning



Supervised Learning
Algorithms

- Input: Training data and labels.
- Goal: Learn function to map new unlabelled data to labels

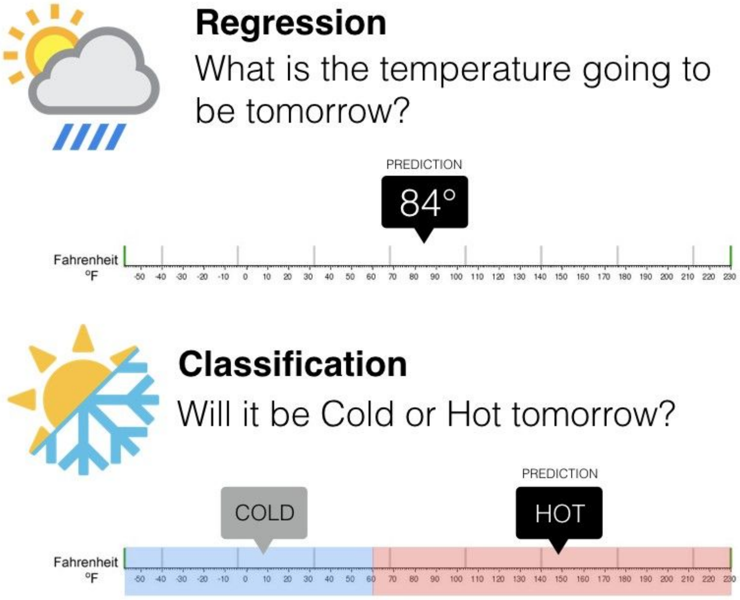
Supervised Learning: Classification vs. Regression



Input: data and discrete labels
Goal: Map data to discrete categories

Input: data and continuous values
Goal: Learn an approximate function that maps data to values

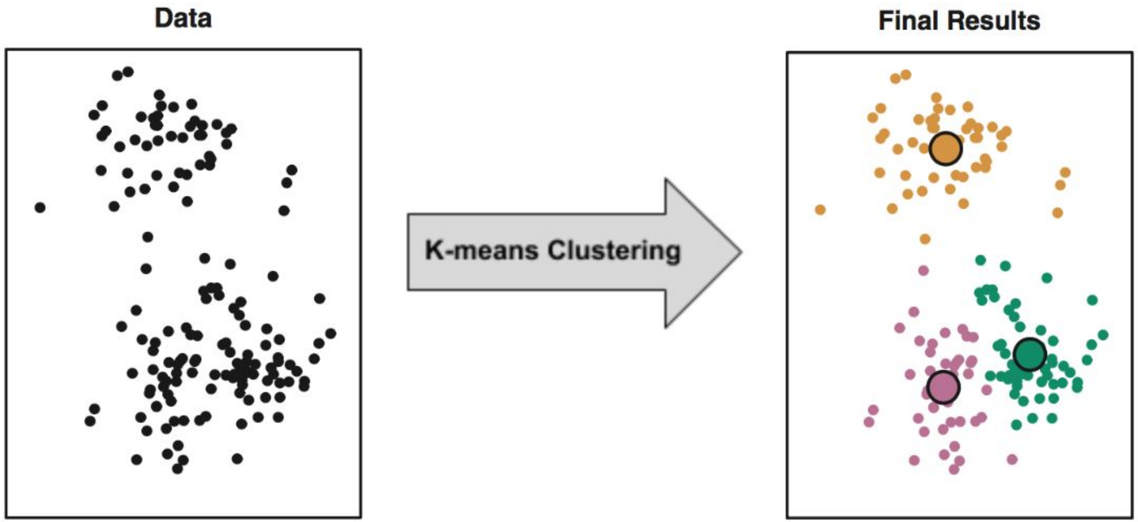
Supervised Learning: Classification vs. Regression



Regression: labels are continuous values

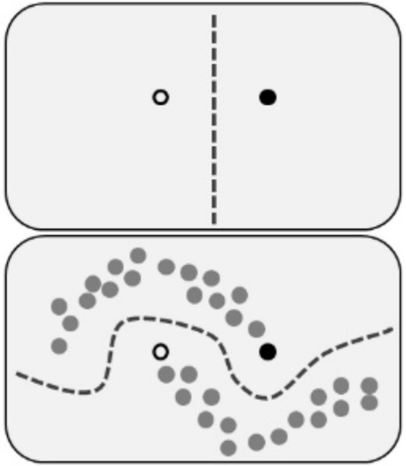
Classification: labels are discrete values

Unsupervised Learning

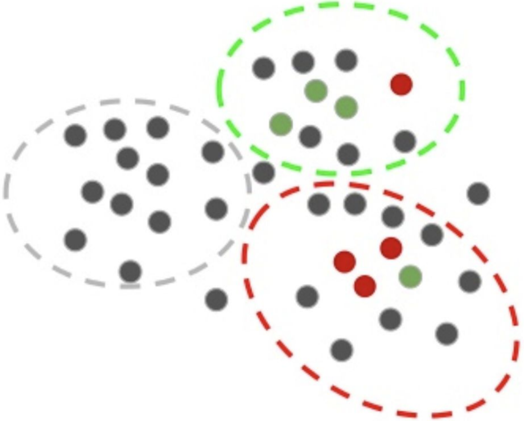


- Input: training data without labels.
- Goal: Learn structure in the data

Semi-Supervised Learning



Classification

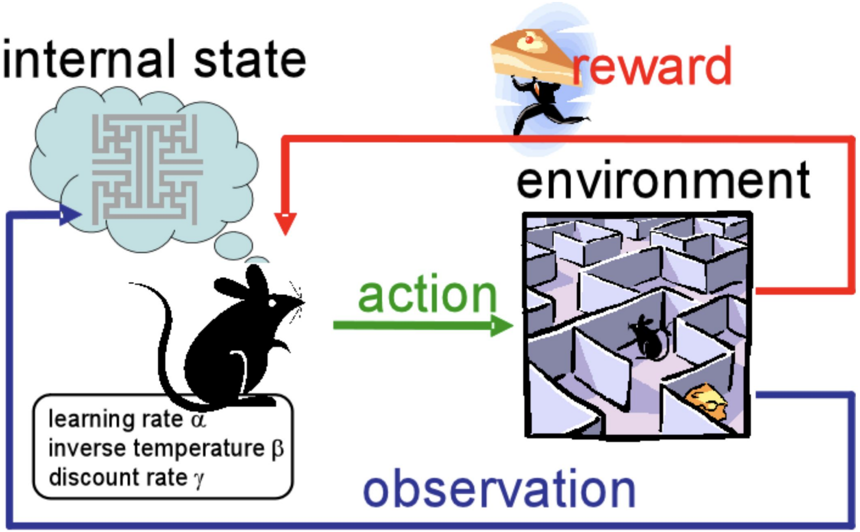


Clustering

- Input: training data, some of which is labelled
- Goal: Learn function to map new unlabelled data to labels and/ or learn structure in the data

Reinforcement Learning

- Framework for decision making
  - Agent with the capacity to act
  - Each action influences the agent's future state
  - Success is measured by a reward signal
  - Goal: Select actions to maximize future reward

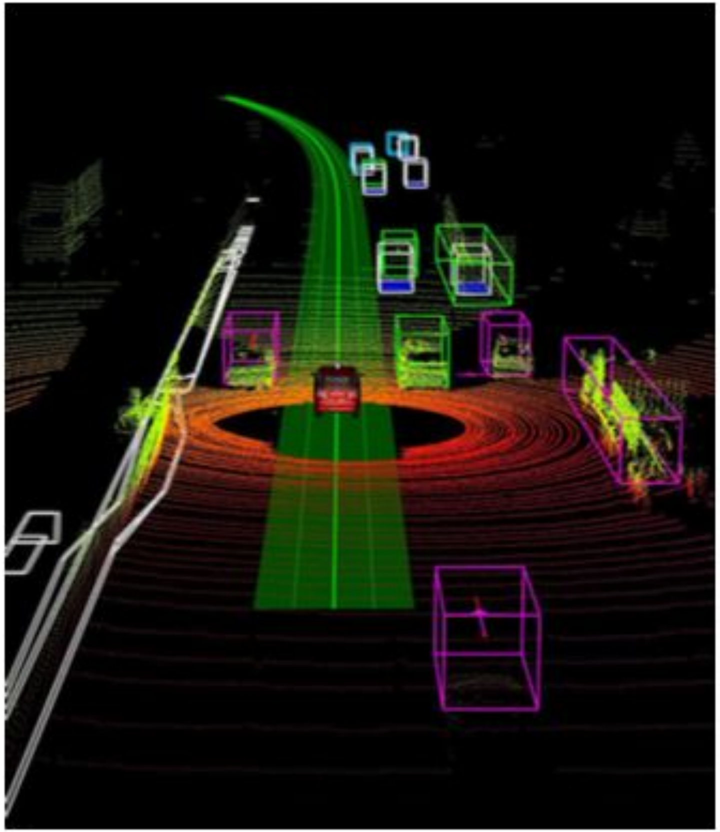


Outline

- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

Self-Driving Car Tasks

- **Localization and Mapping:** Where am I?
- **Scene Understanding:** Where is everyone else?
- **Movement Planning:** How do I get from A to B?
- **Driver State:** What's the driver up to?

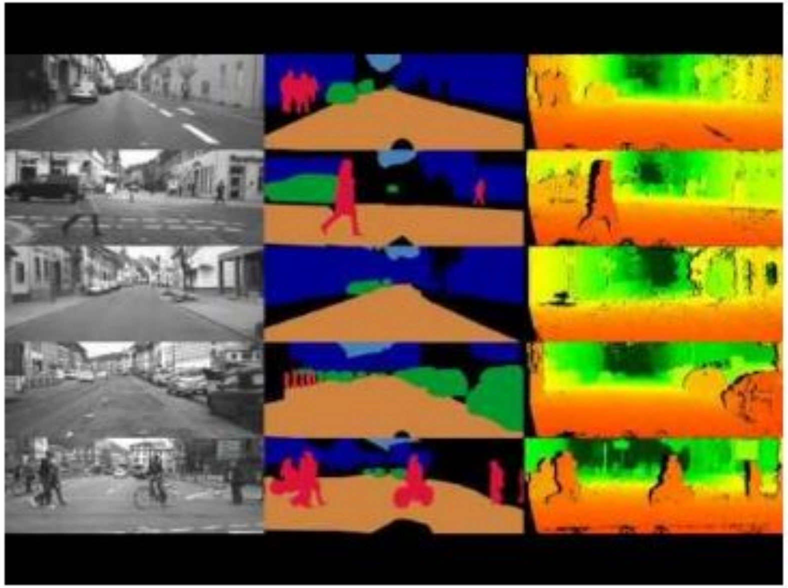


Visual Odometry

- 6-DOF: freed of movement
  - Changes in position:
    - Forward/backward: surge
    - Left/right: sway
    - Up/down: heave
  - Orientation:
    - Pitch, Yaw, Roll
- Source:
  - **Monocular:** I moved 1 unit
  - **Stereo:** I moved 1 meter
  - Mono = Stereo for far away objects
    - PS: For tiny robots everything is "far away" relative to inter-camera distance

Self-Driving Car Tasks

- **Localization and Mapping:**
  Where am I?
- **Scene Understanding:**
  Where is everyone else?
- **Movement Planning:**
  How do I get from A to B?
- **Driver State:**
  What's the driver up to?

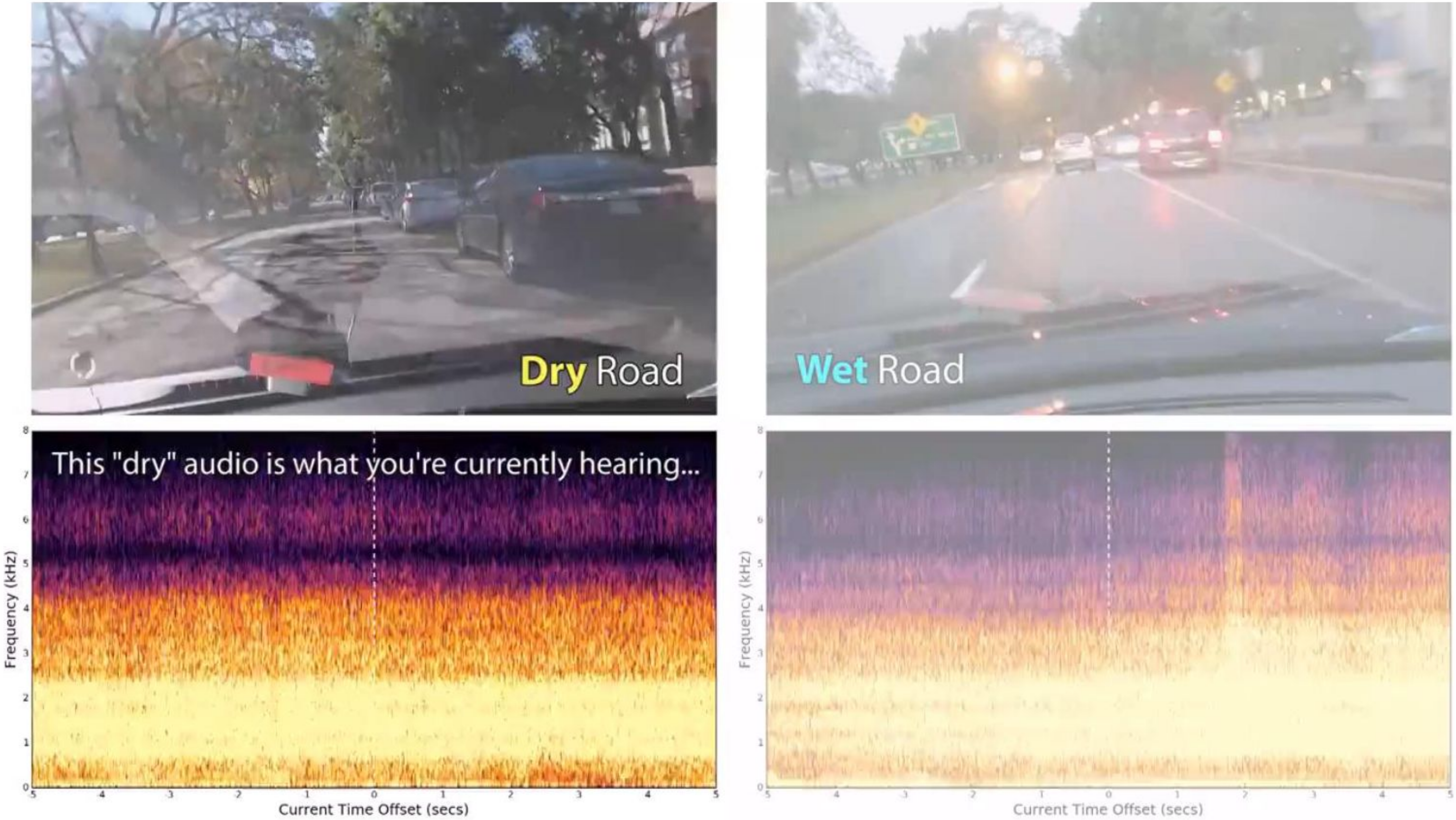


Object Detection

- Past approaches: cascades classifiers (Haar-like features)
- Where deep learning can help:
  recognition, classification, detection

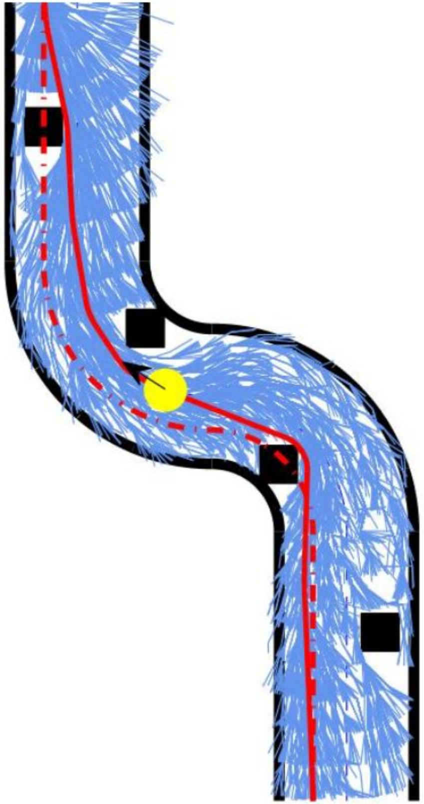
Road Texture and Condition from Audio

(with Recurrent Neural Networks)

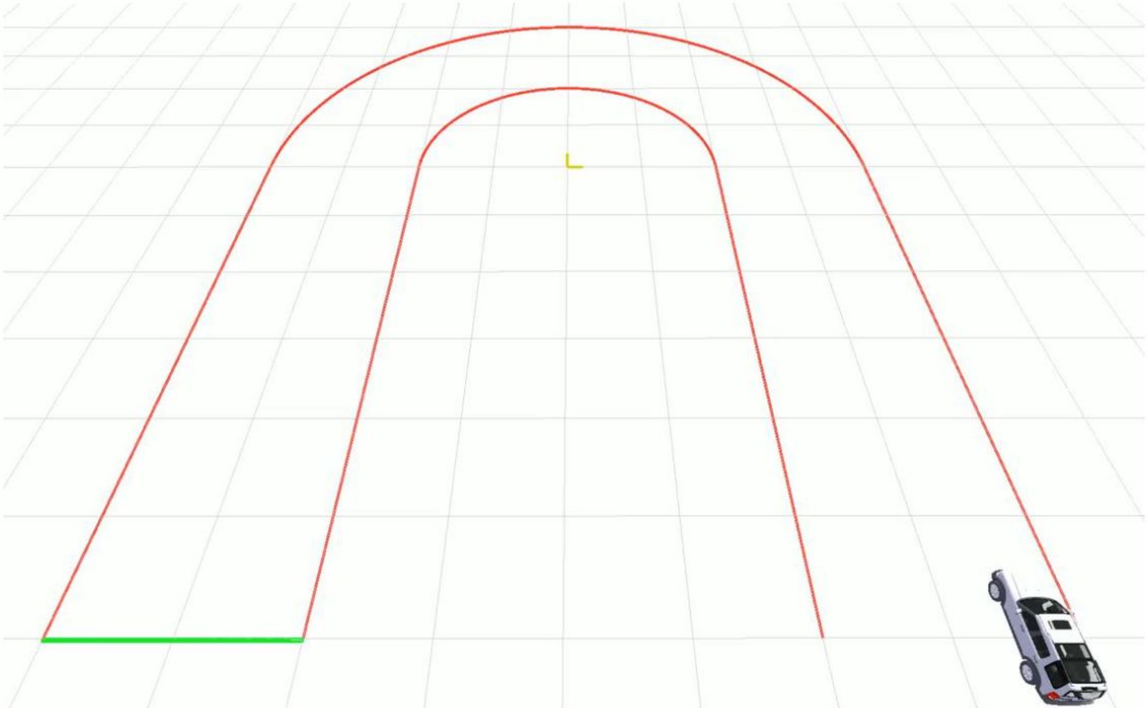


Self-Driving Car Tasks

- **Localization and Mapping:**
  Where am I?
- **Scene Understanding:**
  Where is everyone else?
- **Movement Planning:**
  How do I get from A to B?
- **Driver State:**
  What's the driver up to?



- **Previous approaches:** optimization-based control
- **Deep reinforcement learning:** give the ability to deal with under-actuated control, uncertainty, motion blur, lack of sensor calibration or prior map information.



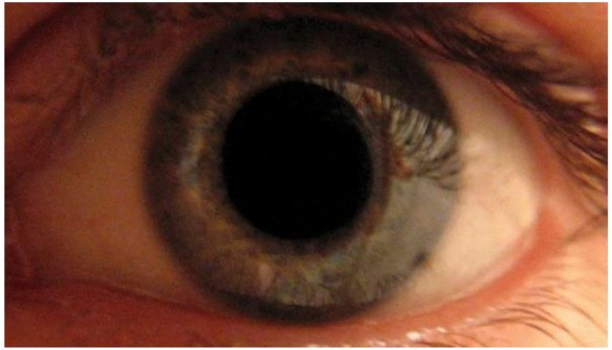
Self-Driving Car Tasks

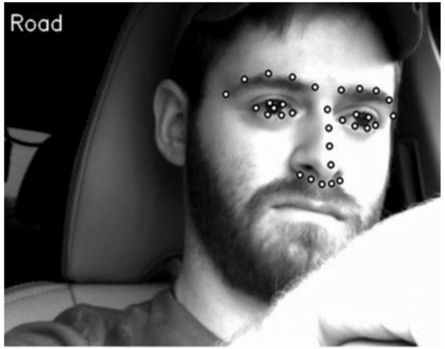
- **Localization:**
  Where am I?

- **Object detection:**
  Where is everyone else?

- **Movement planning:**
  How do I get from A to B?

- **Driver state:**
  What's the driver up to?



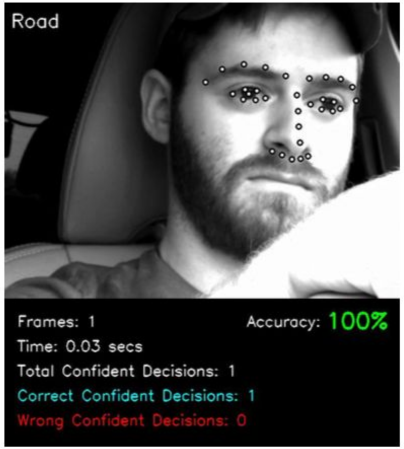


Drive State Detection:
A Multi-Resolutional View

Increasing level of detection resolution and difficulty

Body Pose | Head Pose | Blink Rate | Blink Duration | Eye Pose | Blink Dynamics | Pupil Diameter | Micro Saccades

Gaze Classification | Drowsiness | Micro Glances | Cognitive Load



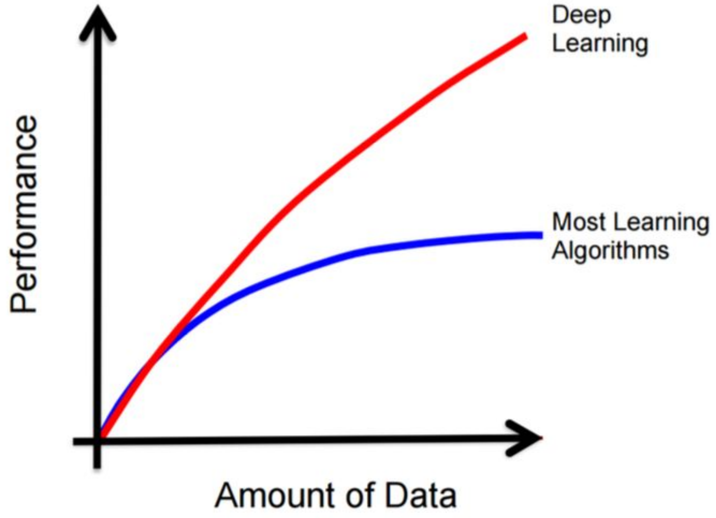
Outline

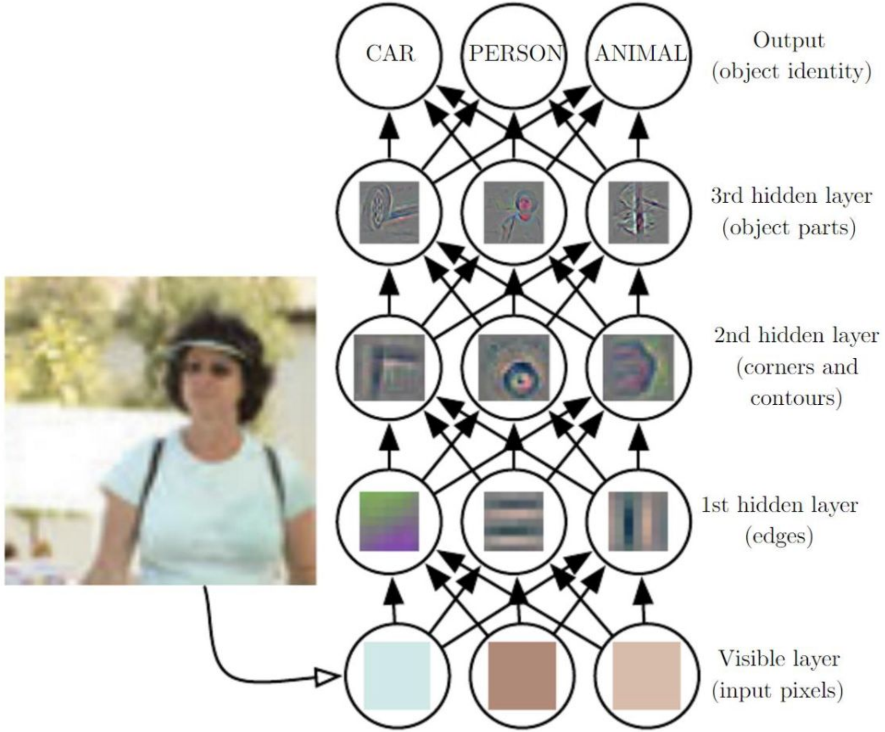
- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

Outline

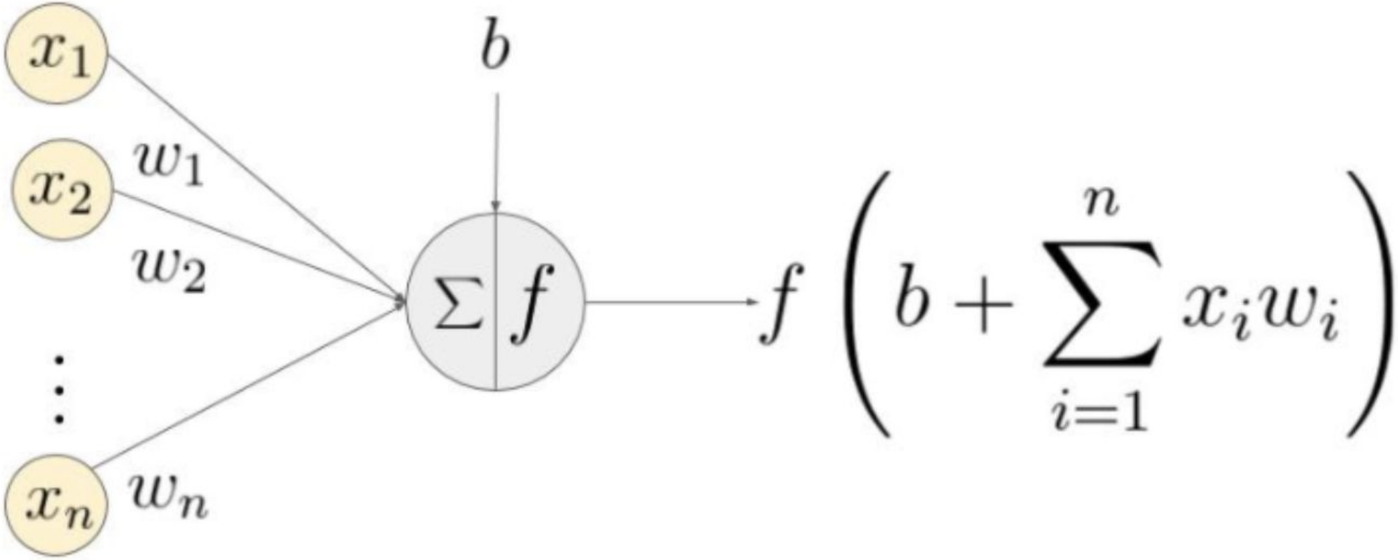
- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

Deep Learning: **Scalable** Machine Learning

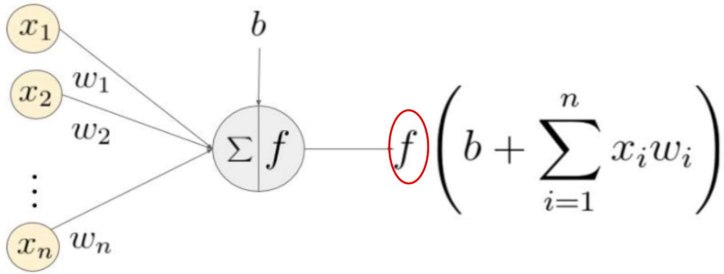


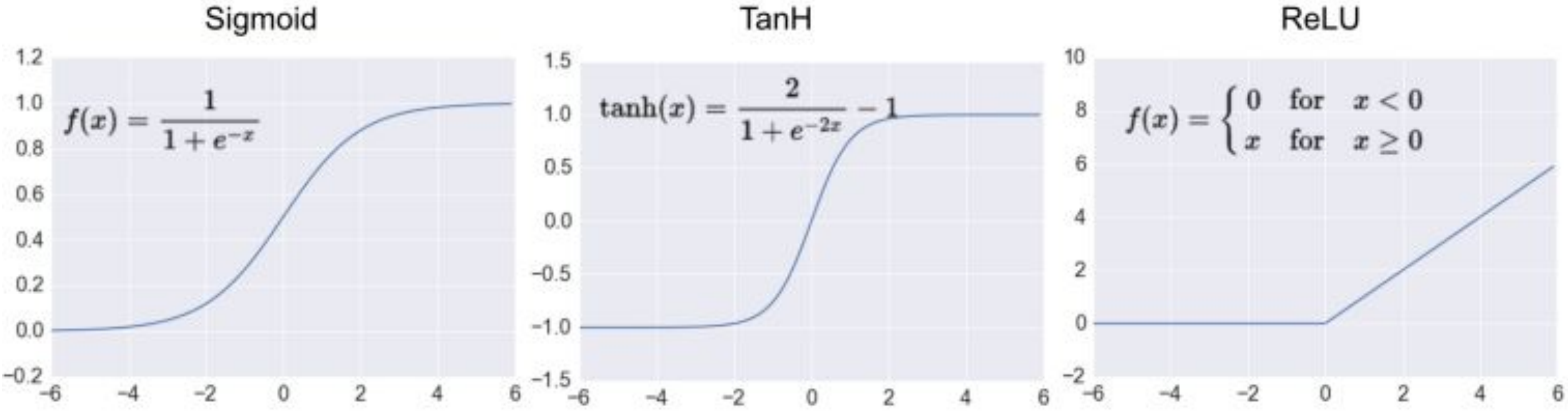


Neurons - Building block of neural networks



Activation Functions



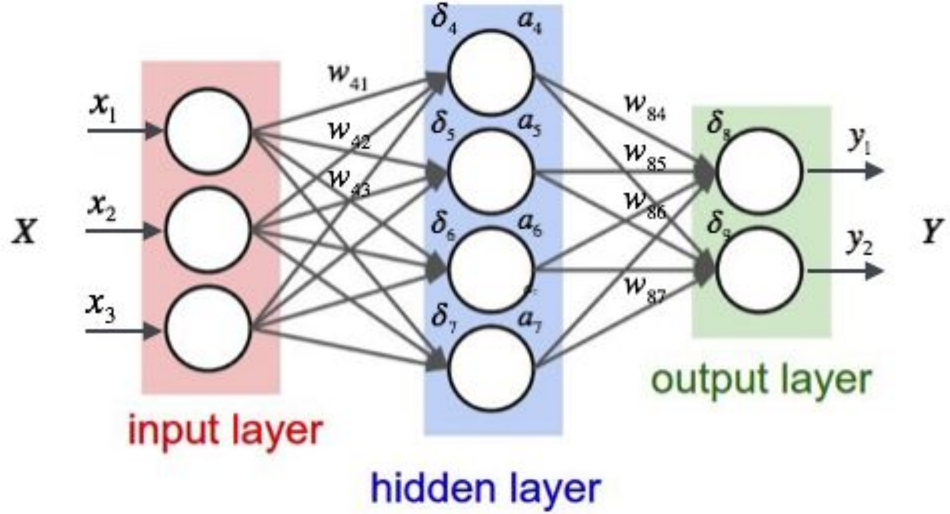


Bounded outputs

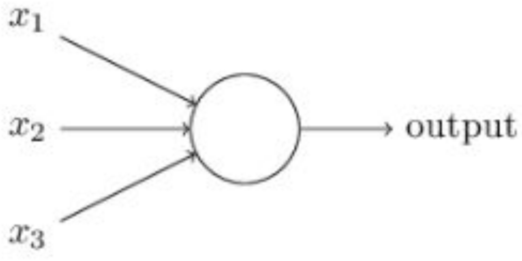
Zero-centered

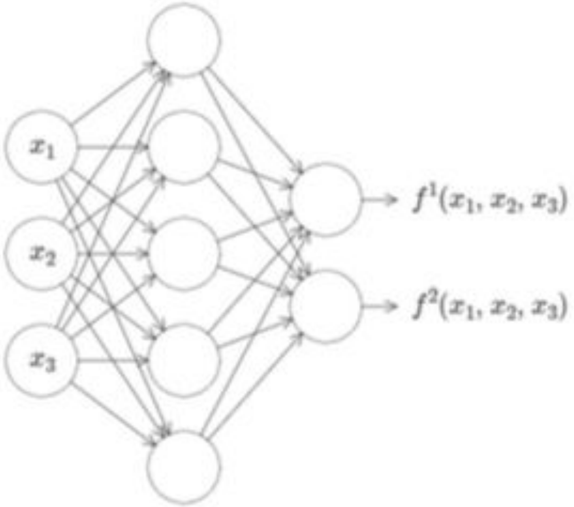
Unbounded outputs
Trains Faster

Combining neurons into layers



Combing Neurons in Hidden Layers:
The "Emergent" Power to Approximate





Universality: For any arbitrary function f(x), there exists a neural network that closely approximate it for any input x

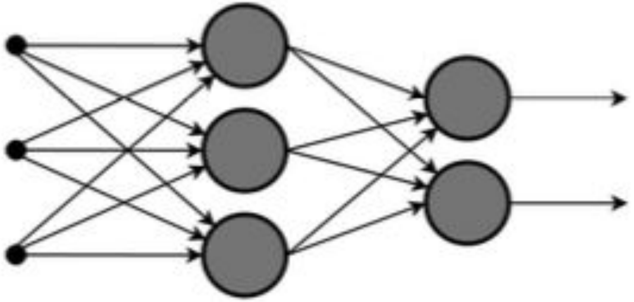
Universality is an incredible property!* And it holds for just 1 hidden layer.

* Given that we have good algorithms for training these networks.

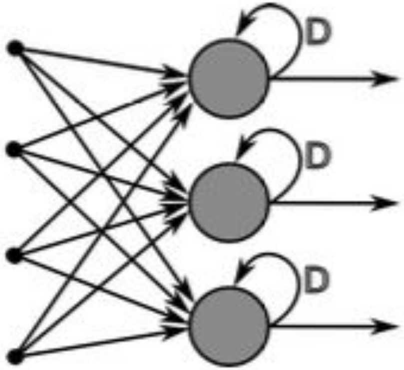
Outline

- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

Combining Neurons into Layers



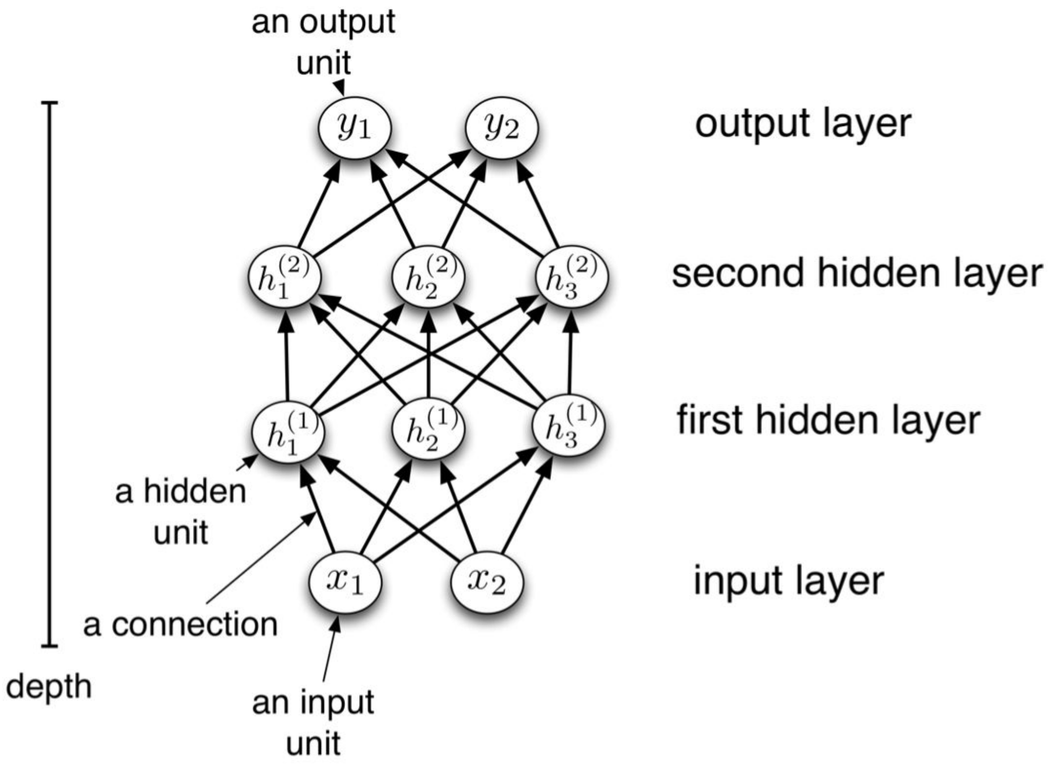
Feed Forward Neural Network



Recurrent Neural Network

- Have state memory
- Are hard to train

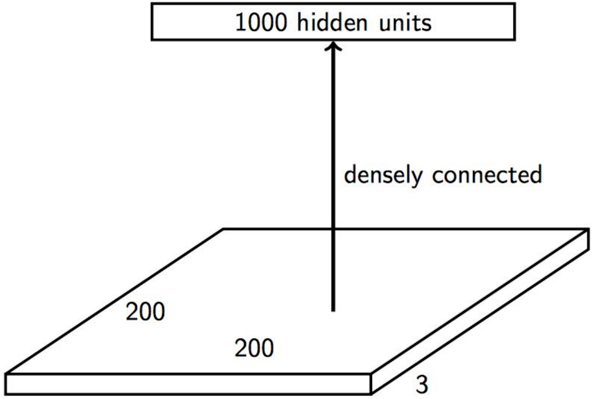
Fully Connected Neural Network



- No connections within a layer
- Each neuron is connected to all neurons in the previous layer
- Used in classification problems, sometimes image recognition, etc.

Size of Fully Connected Neural Networks

Suppose we want to train a network that takes a 200×200 RGB image as input.



What is the problem with having this as the first layer ?

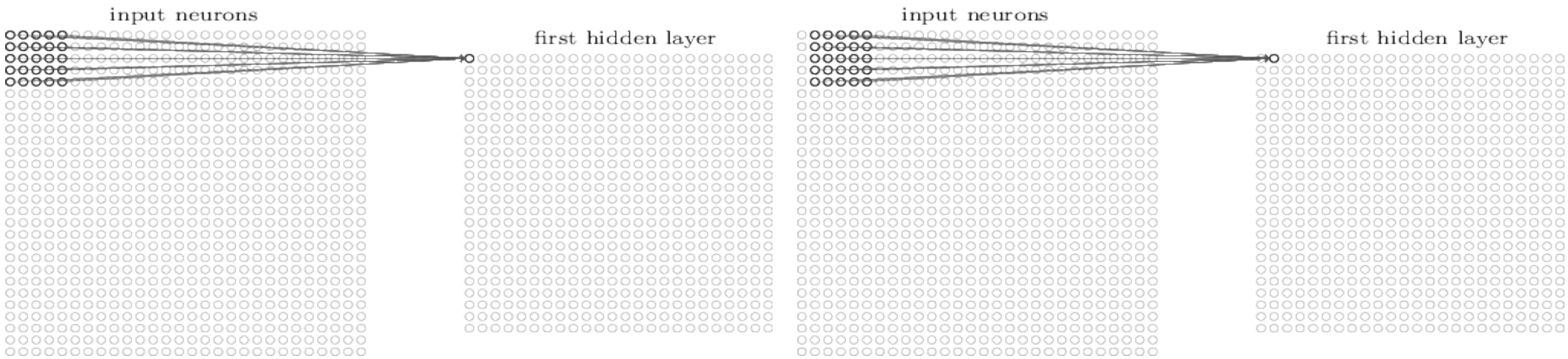
- Too many parameters! Input size $= 200 \times 200 \times 3 = 120$K. Parameters $= 120\text{K} \times 1000 = 120$ million.
- What happens if the object in the image shifts a little ?

Alternative: Convolutional Neural Networks

- Not all layers are fully connected
- Primarily used for image clustering, recognition, and classification
- Convolutional layers - apply same filter at every location in the image
- Pooling layers - reduce the size of the network and build in invariance to small transformations

Convolution

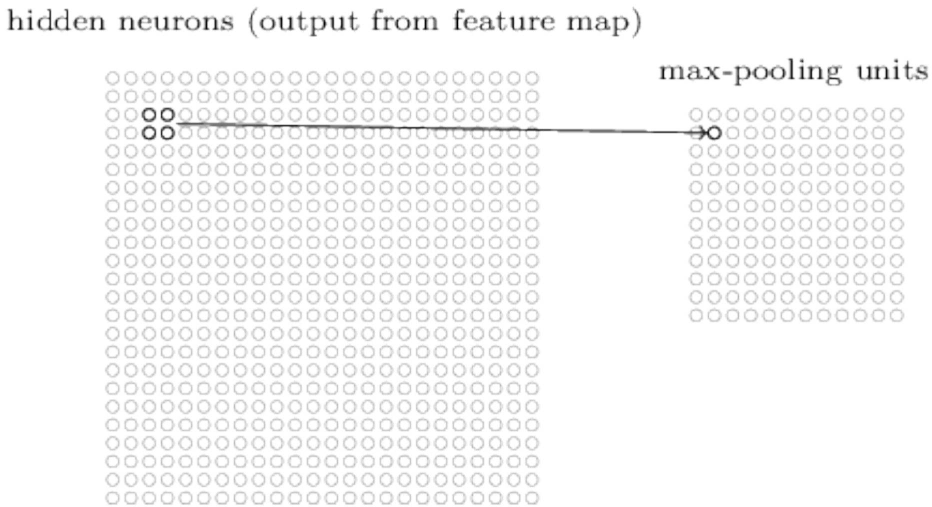
- Motivation: Learn a set of features that occur at all image locations
- Apply same weights to every region of the image
- Functions as a feature detector



- Example: 28x28 image, 5x5 filter - 25 shared weights

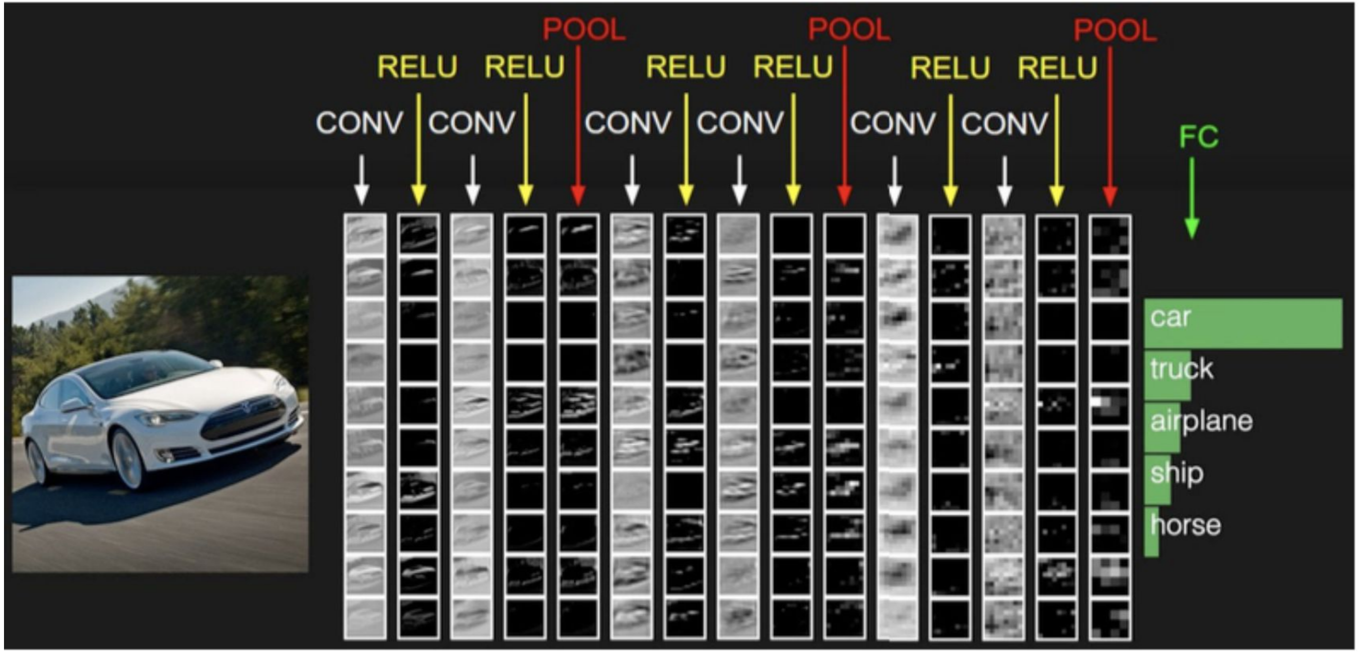
Pooling

- Summarize the output of a group of units
- Reduce the size of the representation
- Invariances to small perturbations in input.
- Example: maximum of every 2x2 region



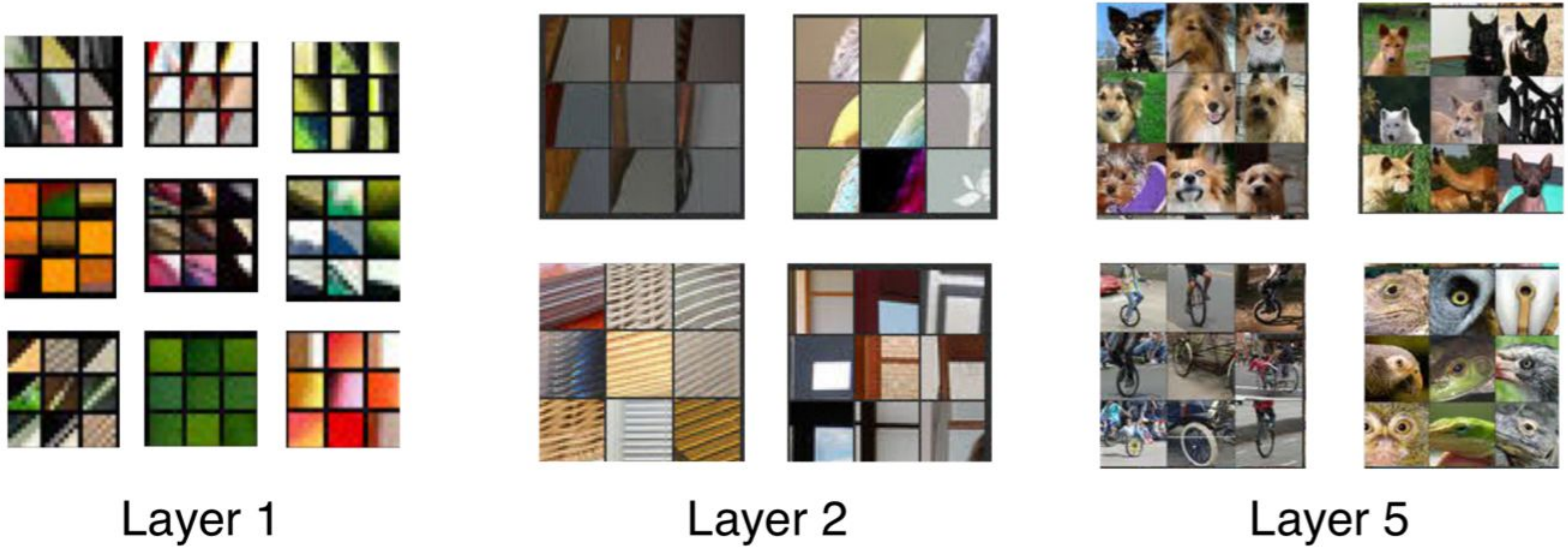
Convolutional neural networks

Putting pooling and convolutional layers together

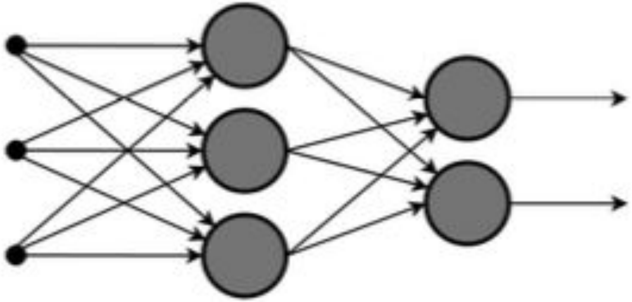


Higher layers capture more abstract information

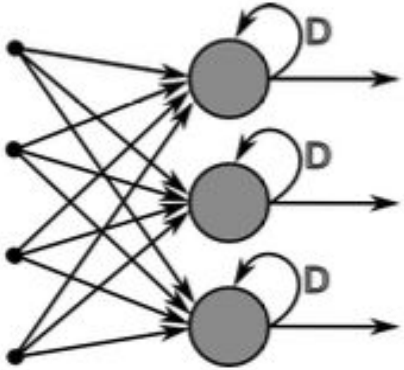
Here are the image regions that most strongly activate various neurons at different layers of the network. (Zeiler and Fergus, 2014)



Combining Neurons into Layers



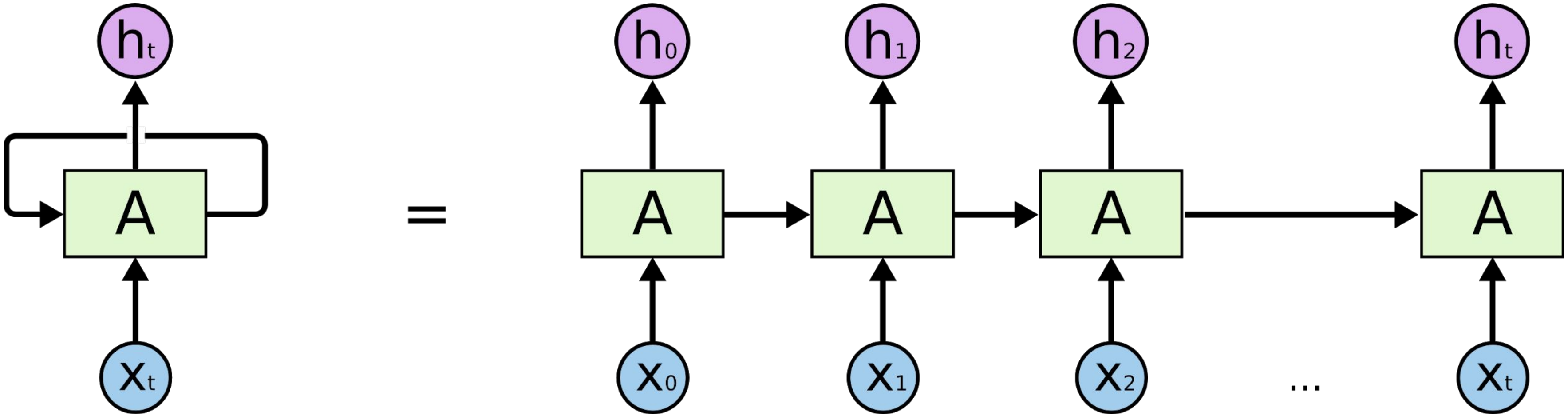
Feed Forward Neural Network



Recurrent Neural Network

- Have state memory
- Are hard to train

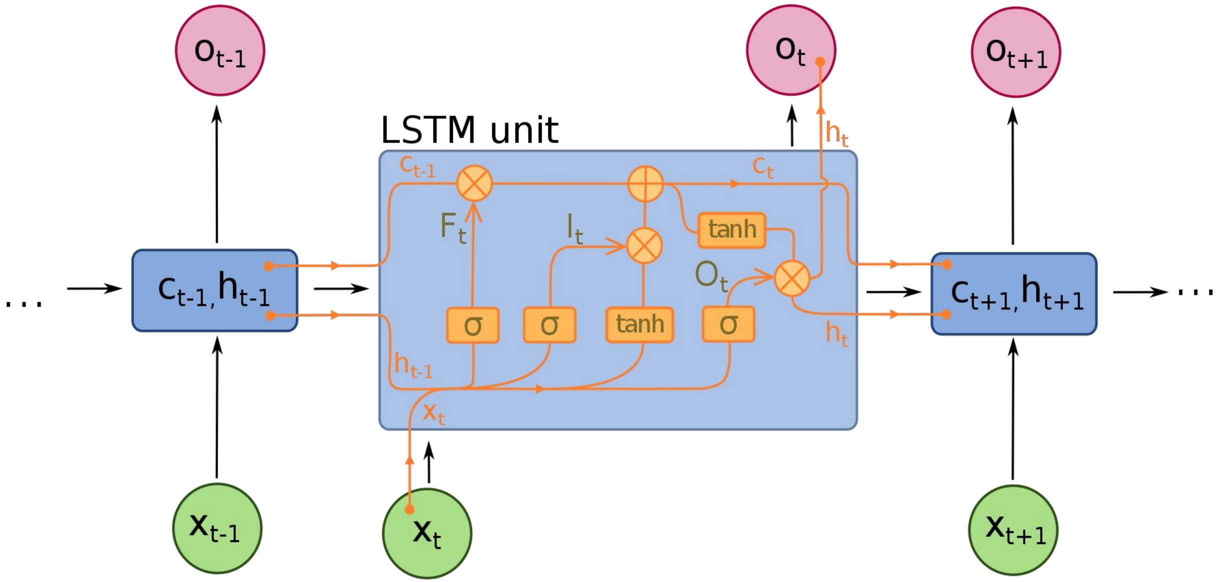
Recurrent Neural Networks



- Often used for language modelling
- Hard to train long term dependencies, e.g. remembering what happened hundreds of words ago.

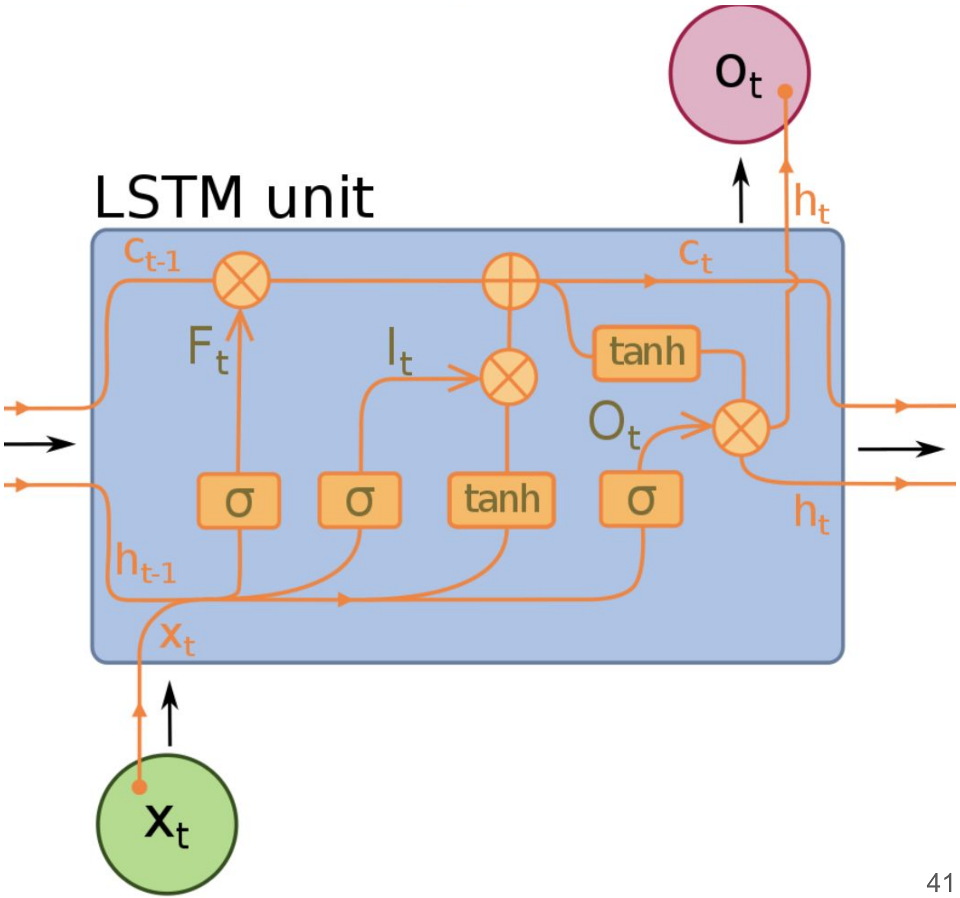
Long Short-Term Memory (LSTM)

- Capable of learning long-term dependencies
- Composed of memory cells which have controllers saying when to store or forget information.
- Used for time series data
- Example application: text generation



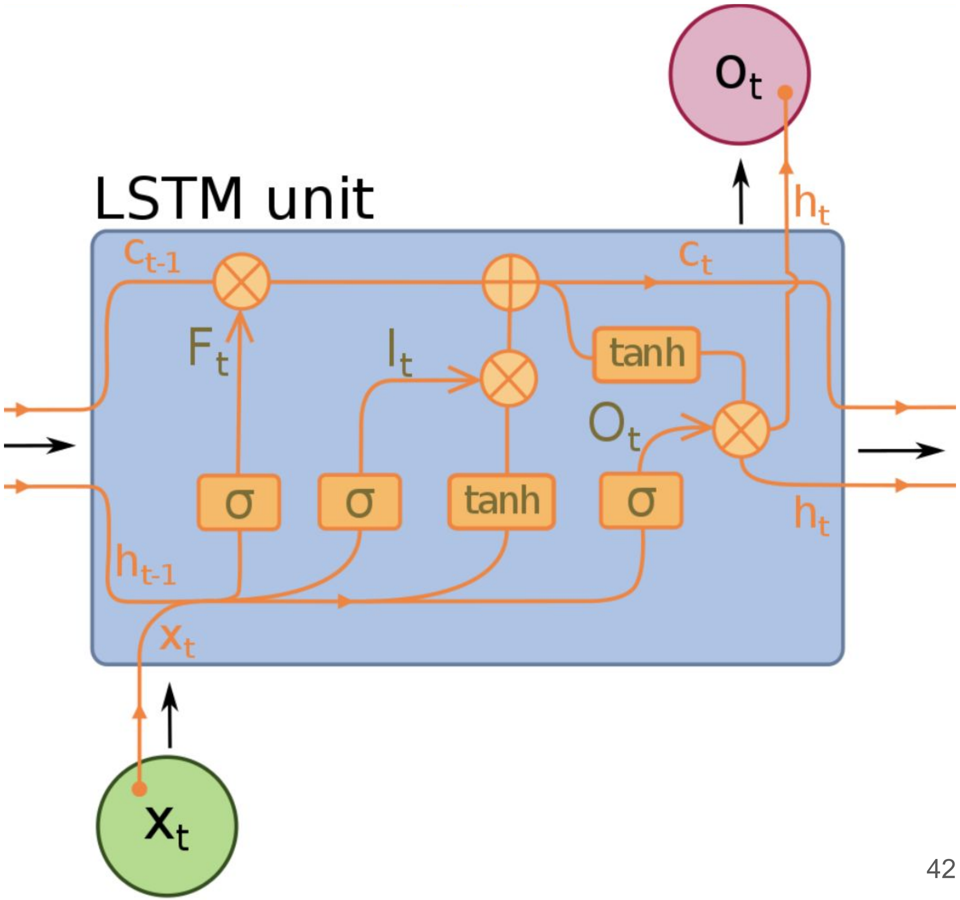
LSTM Components

- x - input to LSTM
- h - hidden state (output vector)
- c - cell state vector: (carries information down the sequence of the LSTM)
- F - forget gate activation
- I - input gate activation
- O - output gate activation



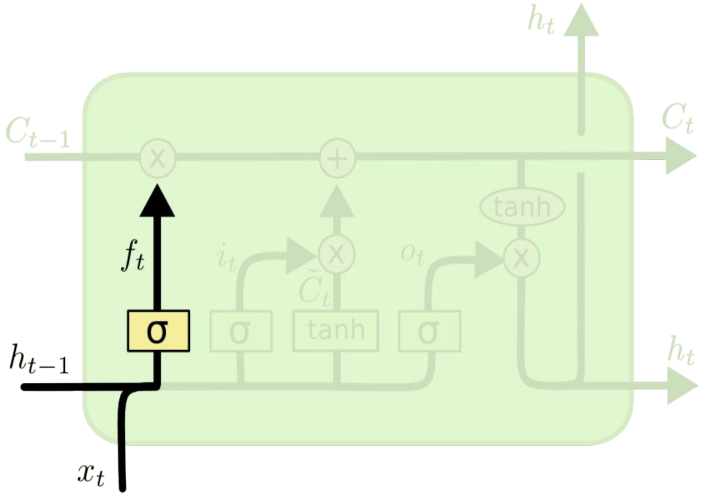
LSTM general behaviour

- I = 0, F = 1: Remember previous value
- I = 1, F = 1: Add to previous value
- I = 0, F = 0; Erase the value
- I = 1, F = 0: Overwrite the value



Forget Gates

f_t between 0 (forget previous input) and 1 (keep) previous input
Function of previous and current input

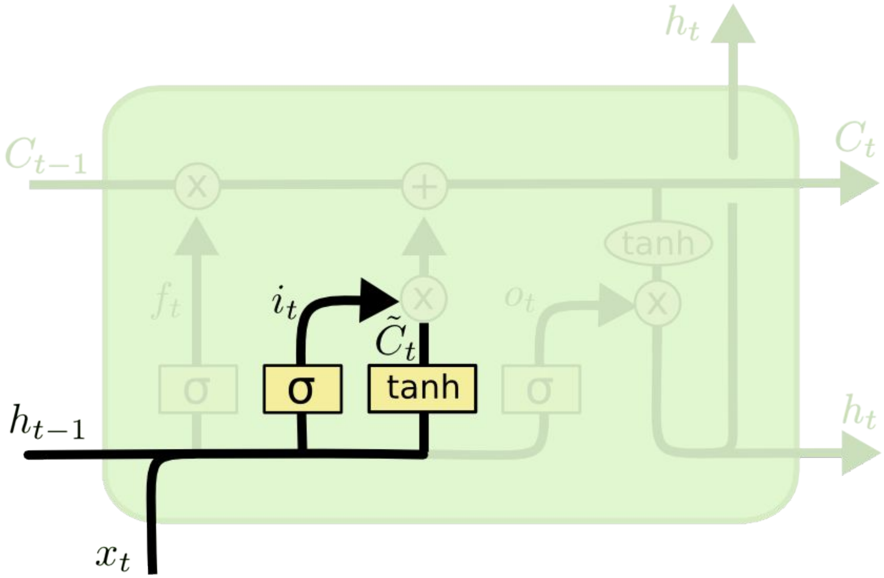


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Ignore gate and temporarily cell state

i - ignore or keep new inputs

C - proposed new cell state

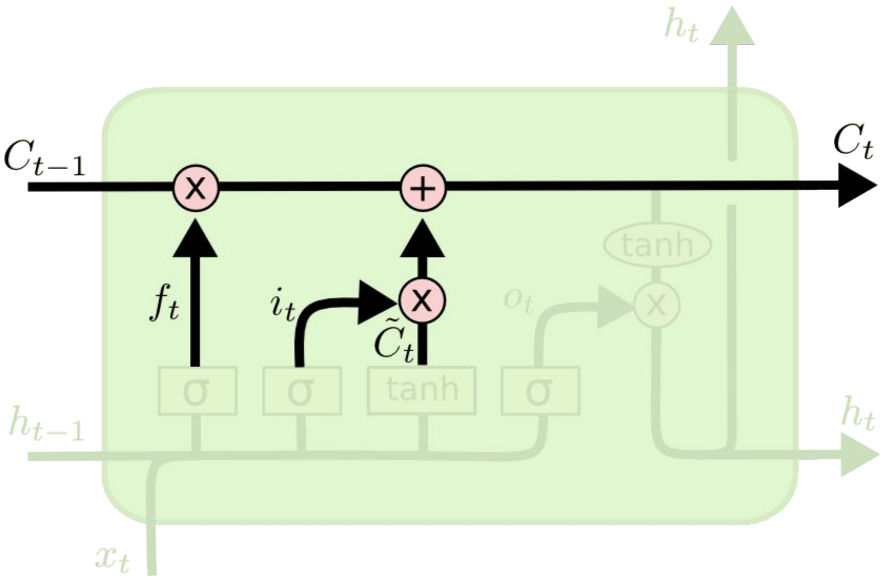


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Outputting Data

Cell state: output a combination of previous and new cell state.

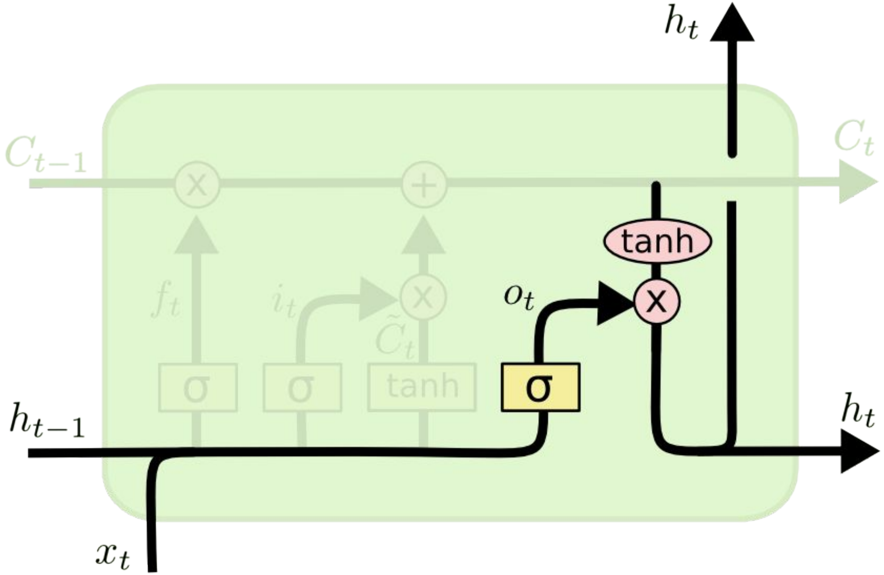


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTMs

o - Output gate's activation vector. Decides what the next hidden/output state should be.

h - Output vector of the LSTM

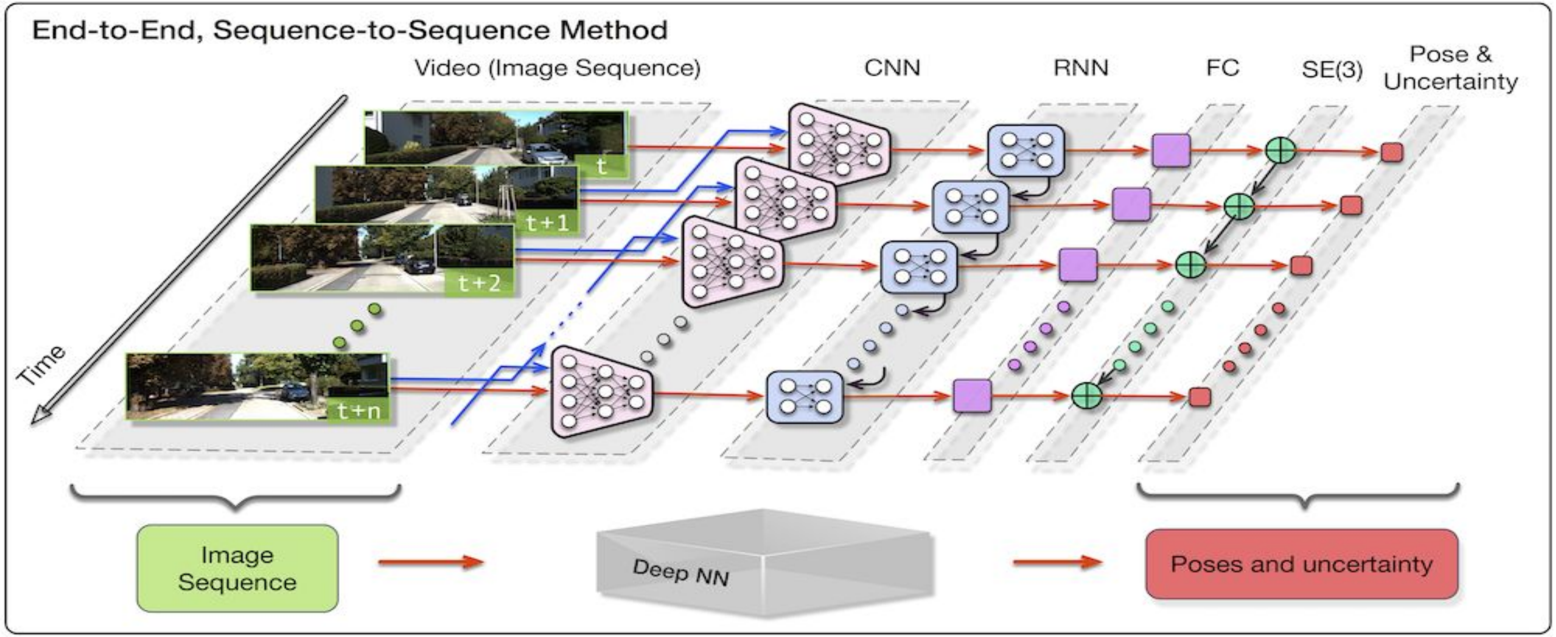


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

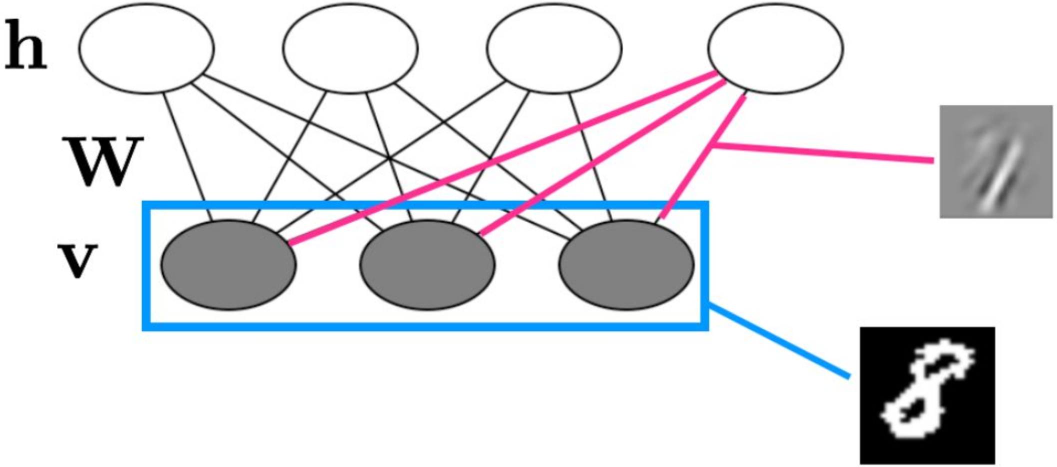
Application: Visual Odometry

- Combines Convolutional and Recurrent layers



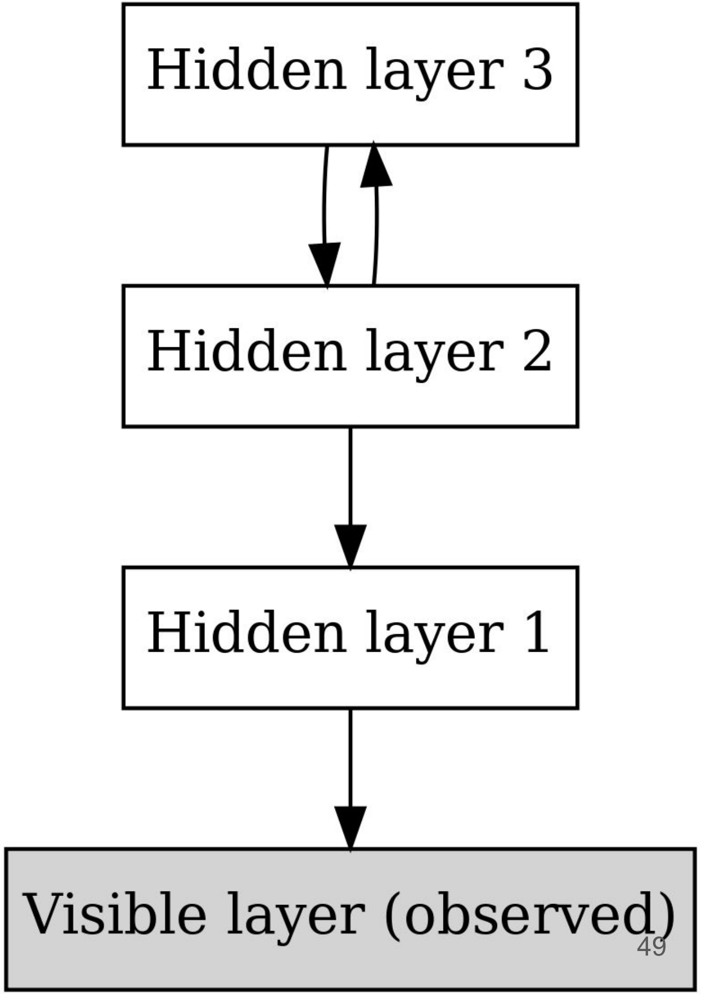
Restricted Boltzmann Machine

- Bipartite Graph over hidden and visible nodes
- Model the joint distribution of the data and the hidden layers.
- Unsupervised and semi-supervised learning
- Generative graphical model



Deep Belief Networks

Similar to RBMs with multiple hidden layers.

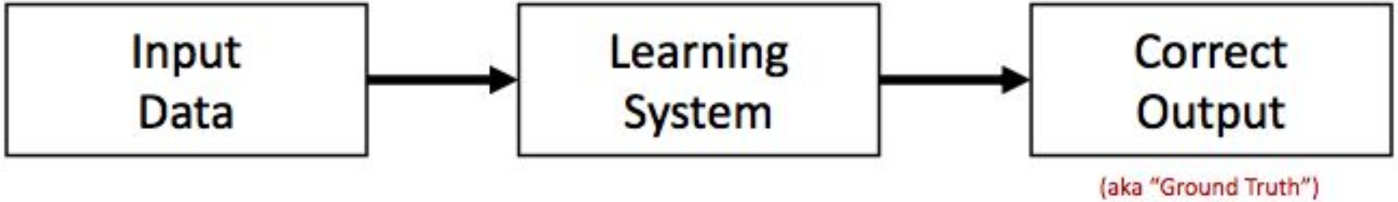


Outline

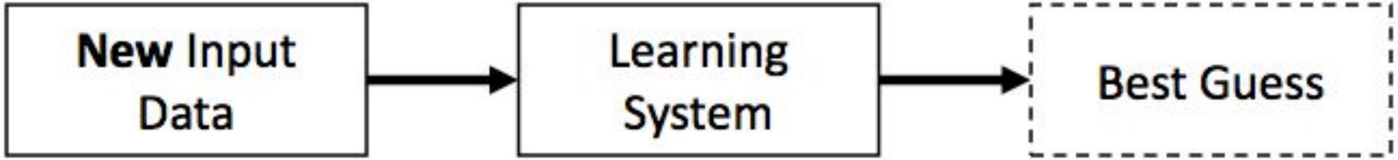
- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

Deep Learning: Training and Testing

Training Stage:

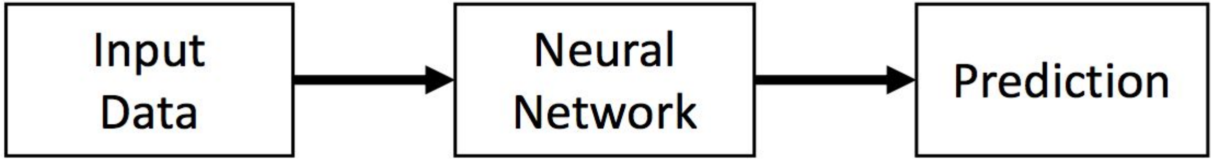


Testing Stage:

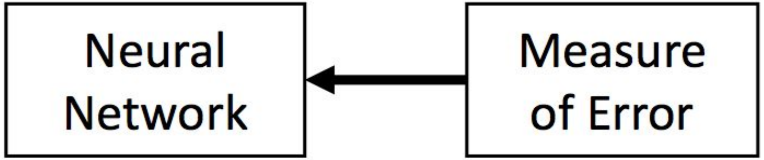


How Neural Networks Learn: Backpropagation

Forward Pass:



Backward Pass (aka Backpropagation):



Adjust to Reduce Error

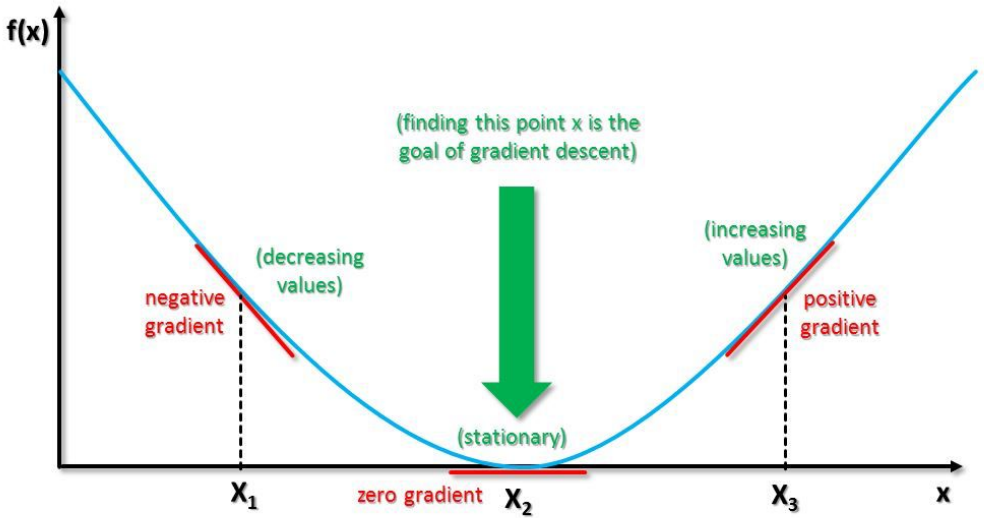
Loss Function

- Example Loss Function: $$MSE = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2$$

- $\hat{y}_i$ Predicted labels are a function of the weights and biases in the neural network.
- Loss Function is a function of weights and biases in the neural network.
- Weights and biases can be optimized by gradient descent.

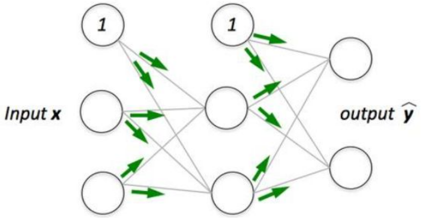
Gradient Descent: Example

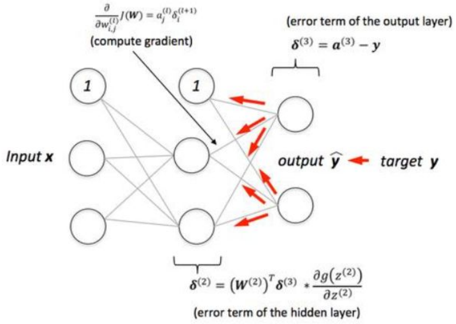
- Loss function: C = f(w), w is a weight
- Weight's gradient: dC/dw
- dC/dw > 0: Decreasing w increases C
- dC/dw < 0: Increasing w increases C
- For small s > 0, updating w' = w - s * dC/dw decreases C
- Repeatedly adjust weights looking for local minimum in C



Key Concepts:

Backpropagation





Task: Update the **weights** and **biases** to decrease **loss function**

Subtasks:

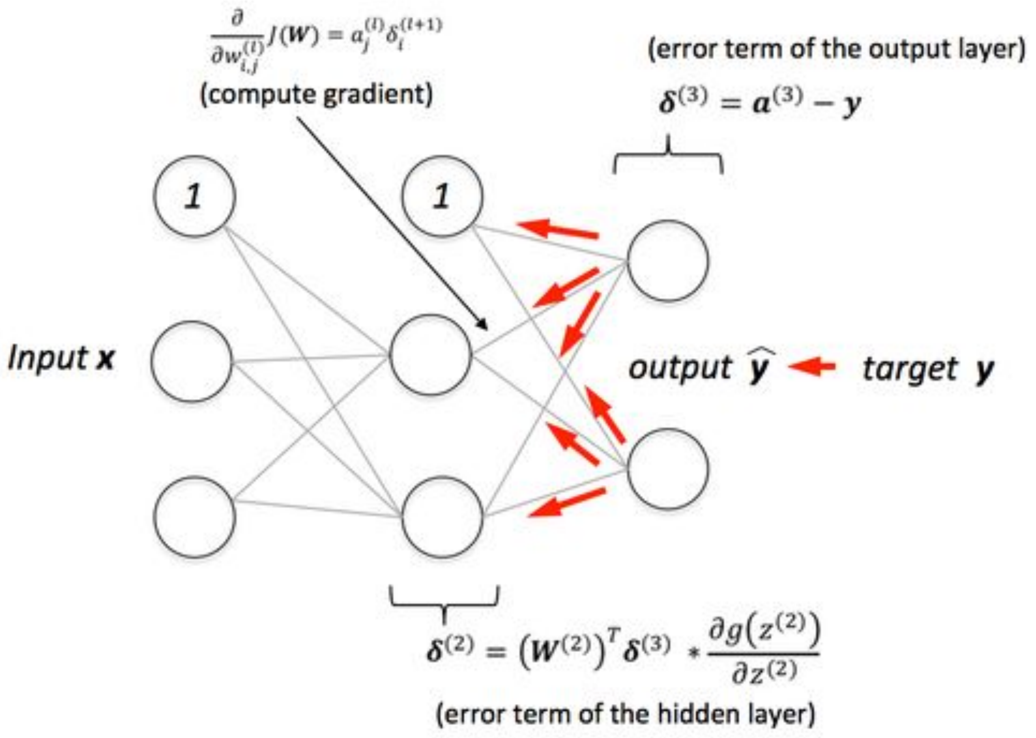
1. Forward pass to compute network output and "error"
2. Backward pass to compute gradients
3. A fraction of the weight's gradient is subtracted from the weight.

Learning Rate

Outline

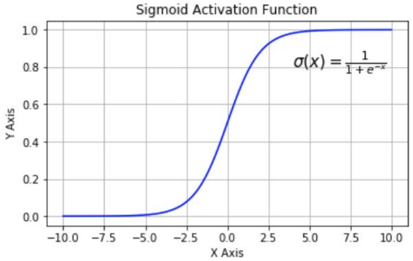
- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

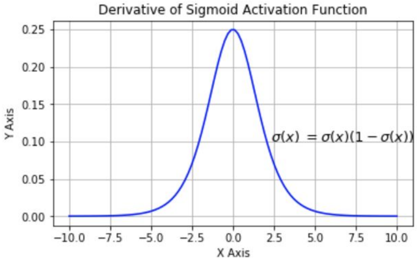
Backpropagation and gradients



- Gradients at layer n-1 are functions of gradients in layer n.
- Gradients are multiplied as they're passed through the network
- Leads to vanishing gradients: Gradients in lower levels are close to 0.
- Exploding gradients: Update too strongly, or have numerical overflow

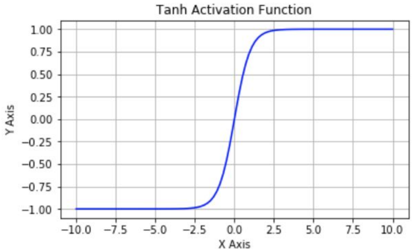
Activation Functions

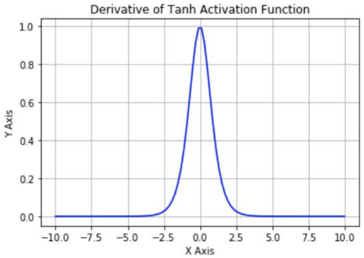




Sigmoid

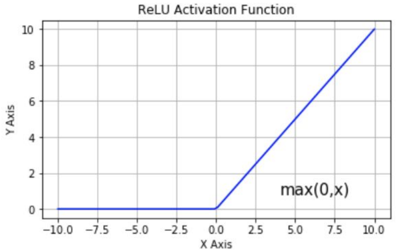
- Vanishing gradients
- Not zero centered

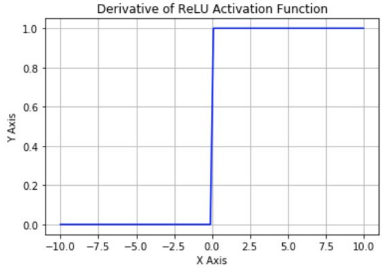




Tanh

- Vanishing gradients



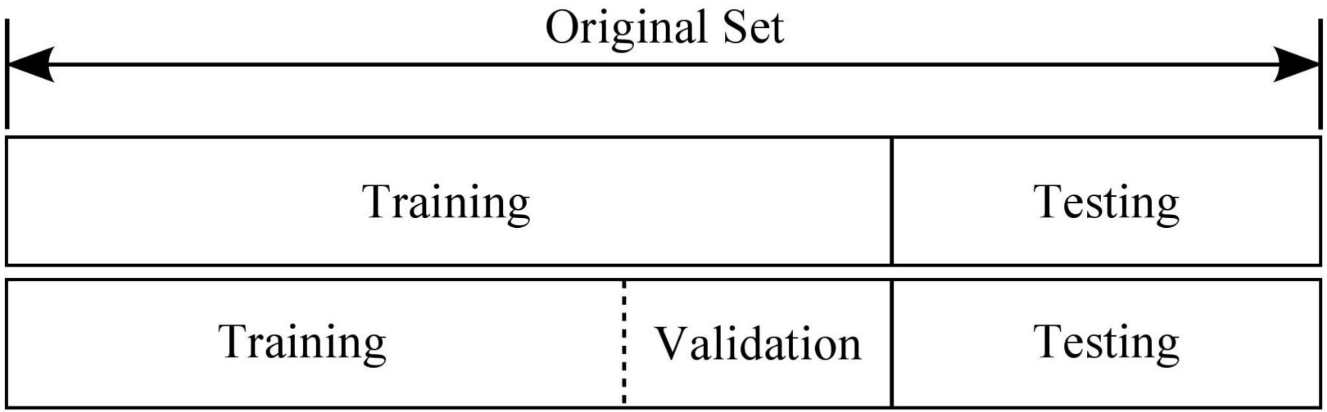


ReLU

- Not zero centered

Key Concepts:

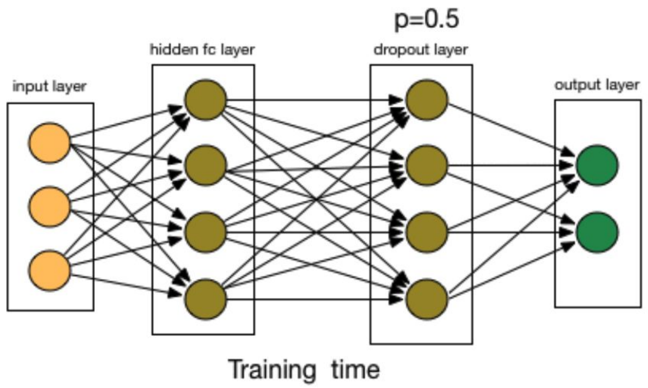
Regularization: Early Stoppage



- Create “validation” set (subset of the training set).
  - Validation set is assumed to be a representative of the testing set.
- **Early stoppage:** Stop training (or at least save a checkpoint) when performance on the validation set decreases

Key Concepts:

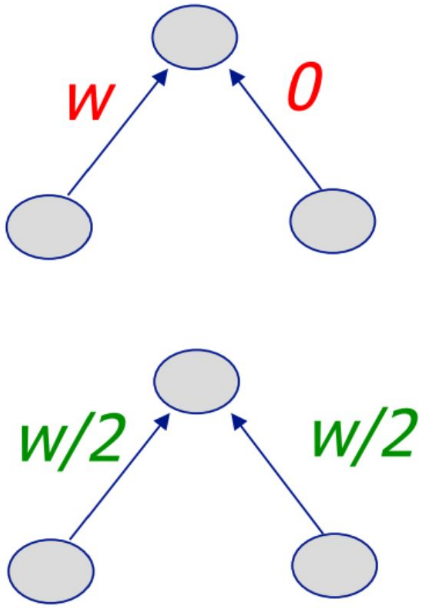
Regularization: Dropout



- **Dropout:** Randomly remove some nodes in the network (along with incoming and outgoing edges)
- Notes:
  - Usually *p >= 0.5* (*p* is probability of keeping node)
  - Input layers *p* should be much higher (and use noise instead of dropout)
  - Most deep learning frameworks come with a dropout layer

Key Concepts:

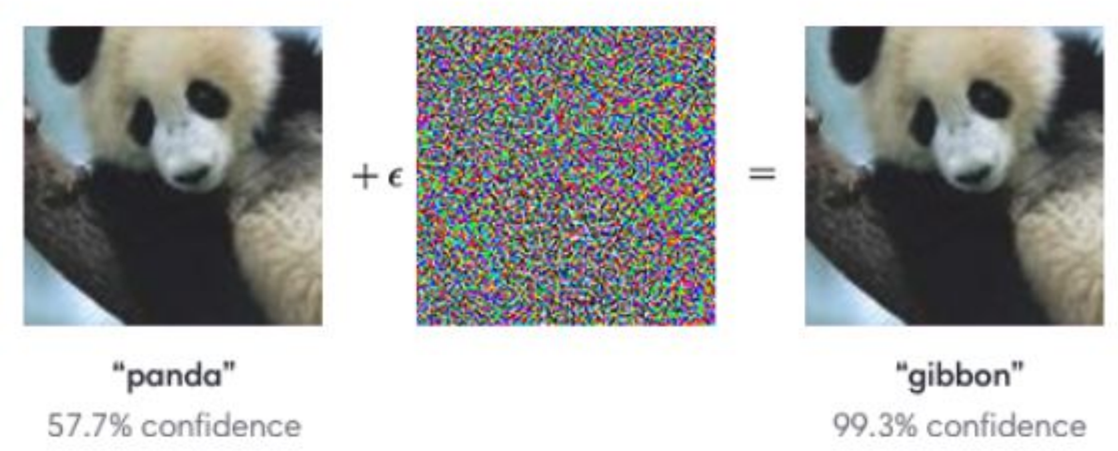
Regularization: Weight Penalty *(aka Weight Decay)*



- **L2 Penalty:** Penalize squared weights. Result:
  - Keeps weight small unless error derivative is very large.
  - Prevent from fitting sampling error.
  - Smoother model (output changes slower as the input change).
  - If network has two similar inputs, it prefers to put half the weight on each rather than all the weight on one.

- **L1 Penalty:** Penalize absolute weights. Result:
  - Allow for a few weights to remain large.

Adversarial examples



Noise is set to be a function of the gradient in the neural network

Adversarial Stickers

Misclassified as speed signs

Environment Modeling Challenge – Uncertainty and Unknowns

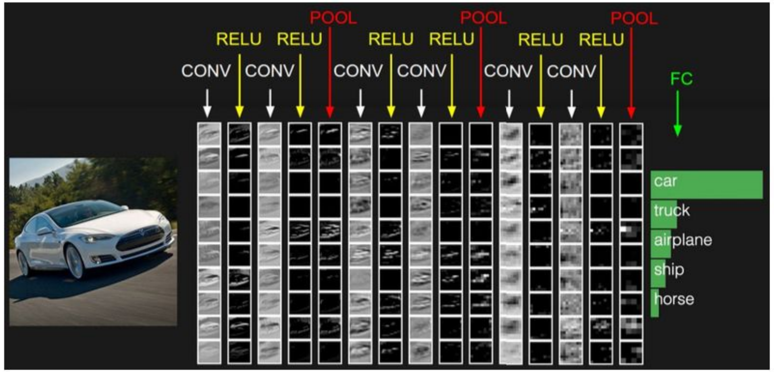
Self-Driving Vehicles: Interact with Humans in Complex Environments; Significant use of machine learning!

Known Unknowns and Unknown Unknowns!!

Cannot represent all possible environment scenarios

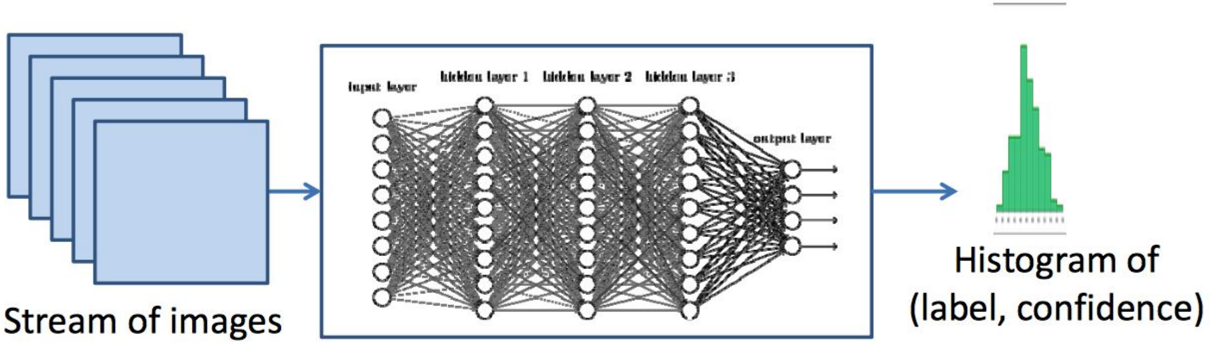
What's the Specification for Perception Tasks?

Convolutional Neural Network trained to recognize cars



How do you formally specify "a car"?

Modeling Learning Systems with High-Dimensional Input & State Space



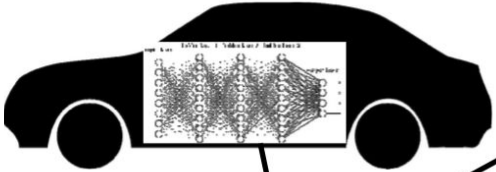
Input Space: ~10^6 dimensions for single time point

System Parameters: >1M, continuous+discrete

Need New Methods for *Abstraction* and *Modular Reasoning*!

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.



System **S** → Does S || E satisfy φ? → YES [+ proof]

Environment **E** →

Specification φ → NO [+ counterexample]

Design Correct-by-Construction instead? How?

Counterexamples, Inputs, etc. from High-Dimensional Signal Spaces

Outline

- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning

Reinforcement Learning in a nutshell

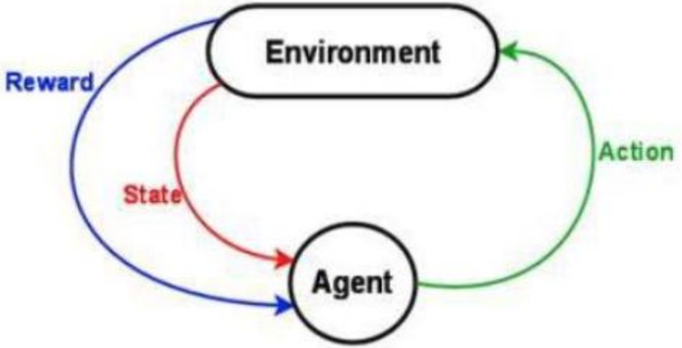
RL is a general-purpose framework for decision-making

- RL is for an agent with the capacity to act
- Each action influences the agent's future state
- Success is measured by a scalar reward signal
- Goal: select actions to maximise future reward

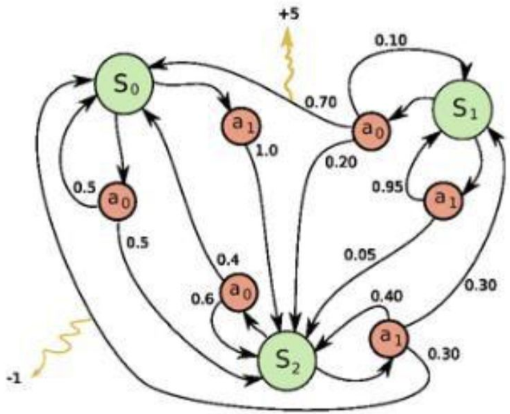
Agent and Environment

- At each step the agent:
  - Executes action
  - Receives observation (new state)
  - Receives reward

- The environment:
  - Receives action
  - Emits observation (new state)
  - Emits reward



Markov Decision Process



$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

state

action

reward

Terminal state

Examples of Reinforcement Learning

Bin Packing

- **Goal** - Pick a device from a box and put it into a container
- **State** - Raw pixels of the real world
- **Actions** - Possible actions of the robot
- **Reward** - Positive when placing a device successfully, negative otherwise

Major Components

- An RL agent may include one or more of these components:
  - Policy: agent's behaviour function
  - Value function: how good is each state and/or action
  - Model: agent's representation of the environment

Policy

- A policy is the agent's behaviour
- It is a map from state to action:
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = \mathbb{P}[a|s]$

Value Function

- A value function is a prediction of future reward
  - "How much reward will I get from action a in state s?"
- Q-value function gives expected total reward
  - from state s and action a
  - under policy π
  - with discount factor γ

reward at time t: r_t

$$Q^{\pi}(s, a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

Approaches to Reinforcement Learning

Value-based RL

- Estimate the optimal value function $Q^*(s, a)$
- This is the maximum value achievable under any policy

Policy-based RL

- Search directly for the optimal policy π^*
- This is the policy achieving maximum future reward

Model-based RL

- Build a model of the environment
- Plan (e.g. by lookahead) using model

Optimal Case

- An optimal value function is the maximum achievable value

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^*}(s,a)$$

- Once we have Q^* we can act optimally,

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\ Q^*(s,a)$$

- Optimal value maximises over all decisions. Informally:

$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \ldots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

Q-Learning: Value Iteration

Learning Rate   Discount Factor

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$$

New State   Old State   Reward

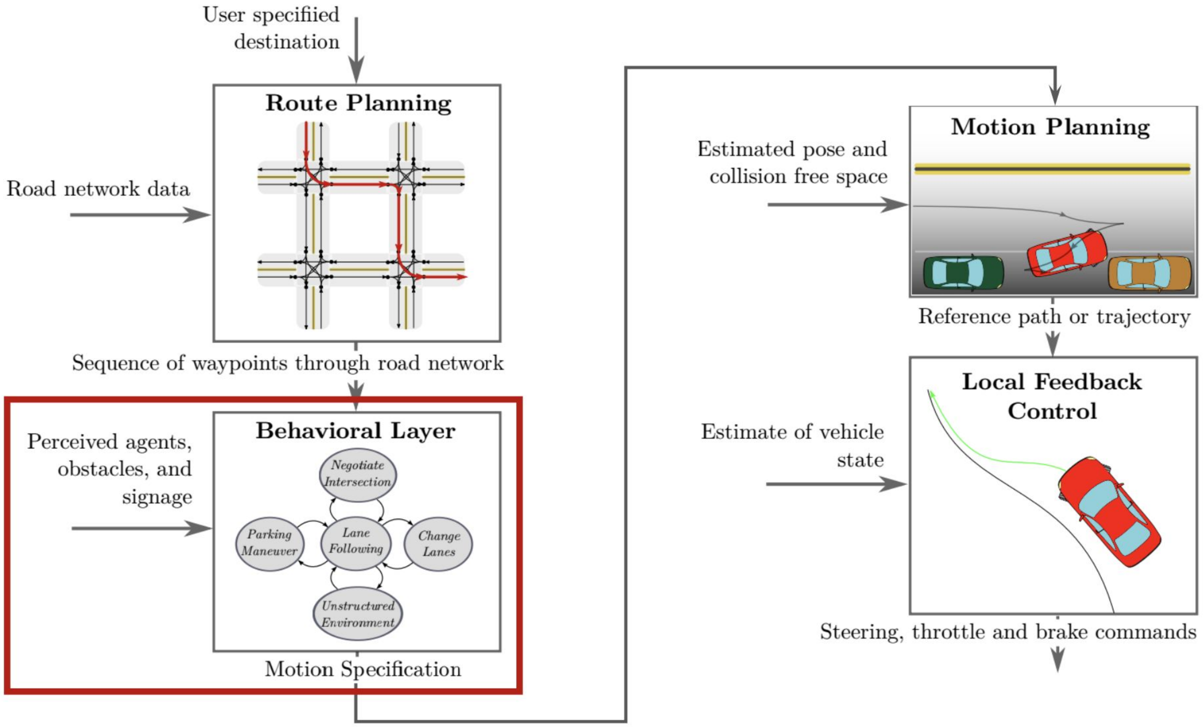
	A1	A2	A3	A4
S1	+1	+2	-1	0
S2	+2	0	+1	-2
S3	-1	+1	0	-2
S4	-2	0	+1	+1

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ maxa' Q[s',a'] - Q[s,a]
      s = s'
until terminated
```

What is Deep Reinforcement Learning?

- Deep reinforcement learning is standard reinforcement learning where a deep neural network is used to approximate either a policy or a value function
- Deep neural networks require lots of real/simulated interaction with the environment to learn
- Lots of trials/interactions are possible in simulated environments, as done in ADS

Autonomous Driving: A Hierarchical View



Paden B, Čáp M, Yong SZ, Yershov D, Frazzoli E. "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles." IEEE Transactions on Intelligent Vehicles 1.1 (2016): 33-55.

Summary

- Introduction to machine learning
- How machine learning is used in self-driving cars
- Deep learning
  - Neural Network Basics
  - Structures in Neural Networks
  - Training Neural Networks
  - Challenges with deep learning
- Reinforcement learning