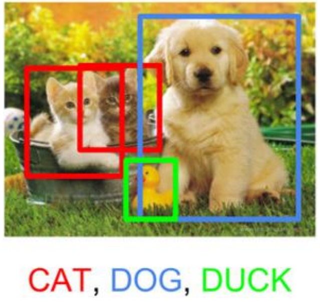
So you want to train a neural network...

Artificial neural networks

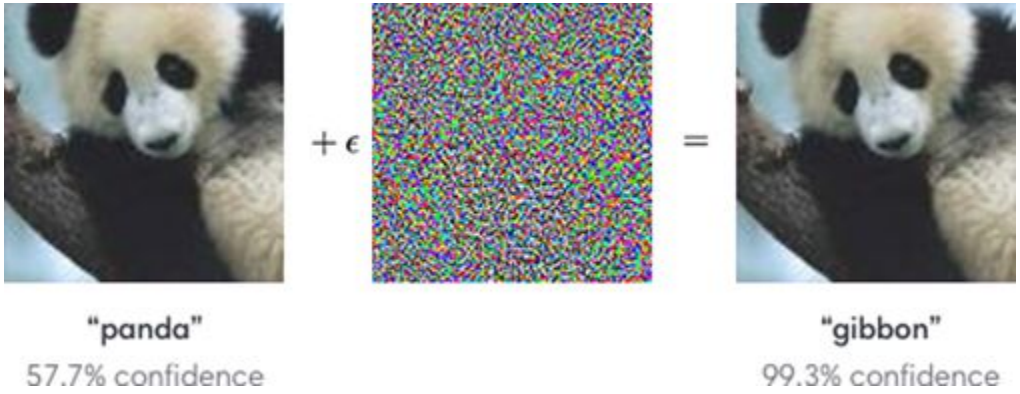
Many success stories for neural networks, old and new

- Credit card fraud detection
- Hand-written digit recognition
- Face detection
- Autonomous driving (CMU ALVINN)
- Object recognition
- Speech recognition



Points to Consider

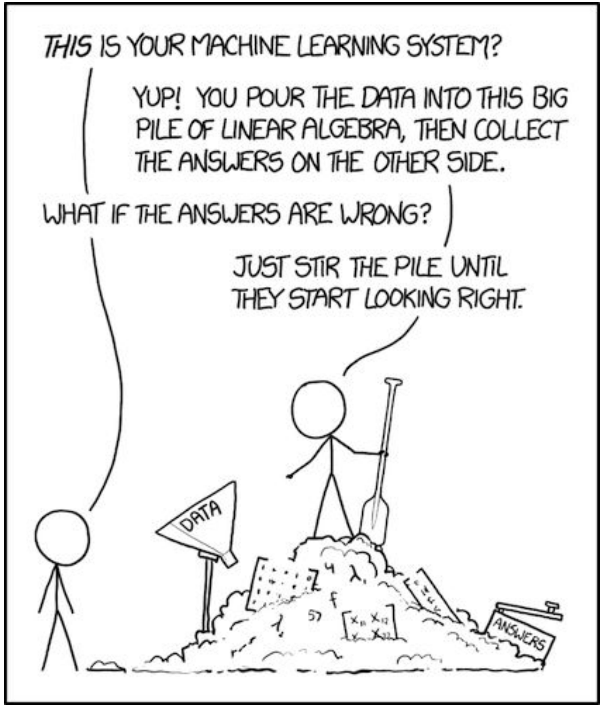
- **Interpretability**: We generally can't inspect well why a deep neural network makes the decisions it does (unlike GMMs), but see [1]



[1] https://christophm.github.io/interpretable-ml-book/cnn-features.html

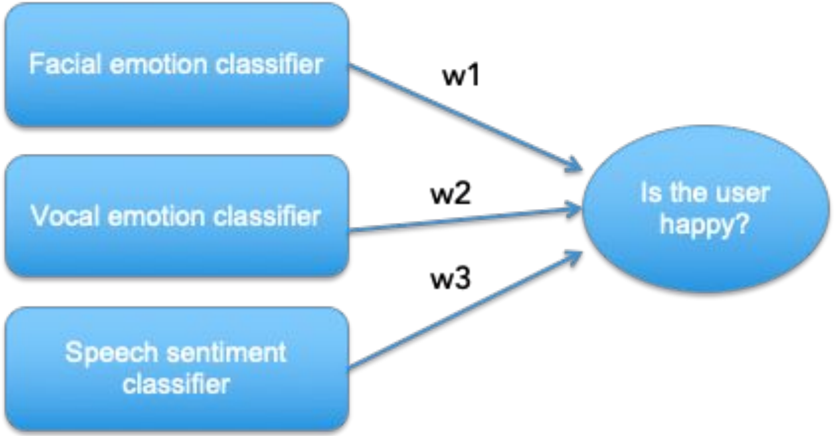
Points to Consider

- **Interpretability**: We generally can't inspect well why a deep neural network makes the decisions it does (unlike GMMs), but see [1]
- **Sample Efficiency**: Deep neural networks often require *a lot* of data, which we might not always have
- **Alchemy** (controversial): We just really don't have a solid understanding of why certain techniques do better than others



[1] https://christophm.github.io/interpretable-ml-book/cnn-features.html

Recall weighted fusion



Optimizing Weights for Weighted Sum

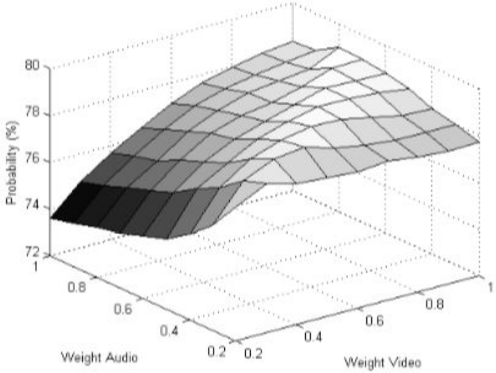


Figure 6: Influence of audio and video weights on vector fusion performance. Stable performance is observed if audio and video events are weighted in a ratio of 8 to 10.

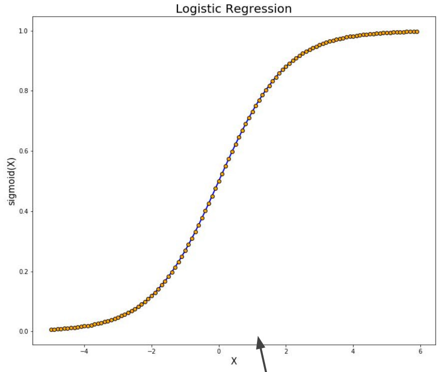
- Fusion mechanism can be voting, weighted sum or an ML approach

Linear Regression

- Simplest linear model for regression

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

  - Remember, we're learning $\boldsymbol{w}$
  - Set $\boldsymbol{w}$ so that $y(\boldsymbol{x}, \boldsymbol{w})$ aligns with target value in training data



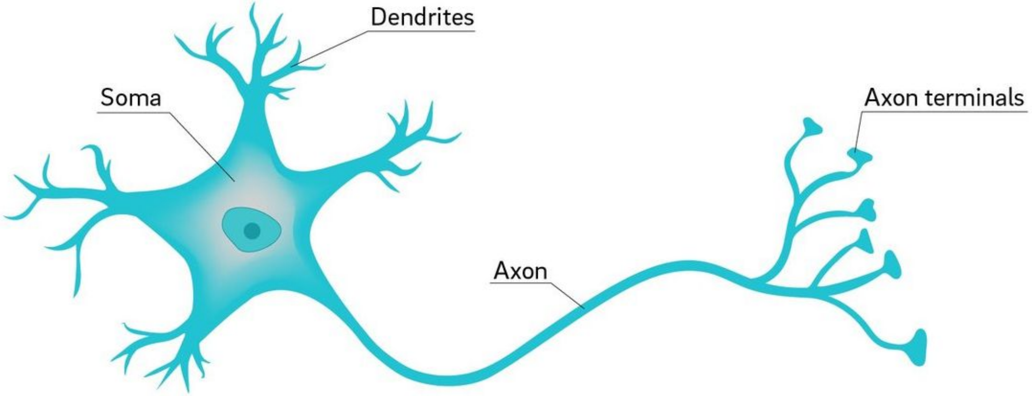
Logistic Regression adds a sigmoid activation function to map to a probability between 0 and 1.

Once we have found the weights, given new values of x, we can then predict an output.

Artificial Neural Networks

They have inspired from actual neurons in human/animal brain.

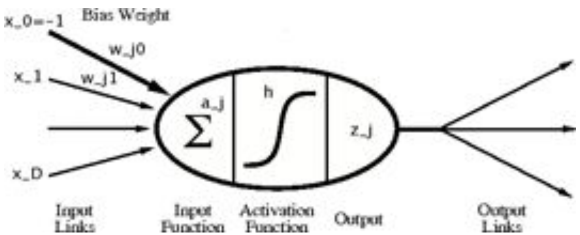
- Dendrites: Take its input from other neurons in the form of electrical impulses.
- Soma: Generates inferences from inputs and decides what action to take.
- Axon terminals: Transmit outputs in the form of impulses.

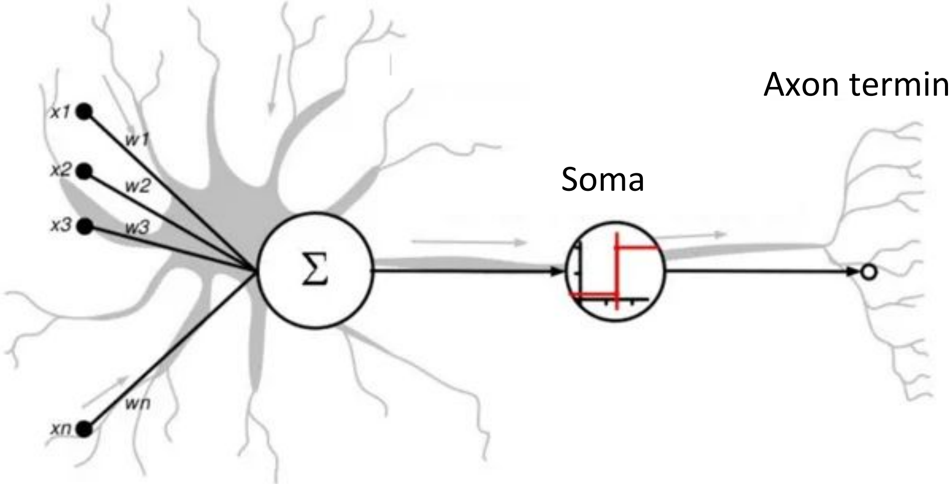


Artificial Neuron cell

$$a_j = \sum_{i=1}^{D} \left(w_{ji}^{(1)} x_i + x_{j0}^{(1)} \right)$$

$$z_j = h(a_j)$$



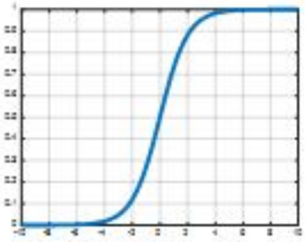


Activation function

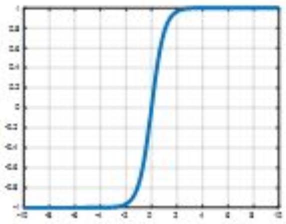
Can use a variety of activation function

- Sigmoid
- Hyperbolic tangent
- Threshold
- Rectified linear unit (ReLU): max(0,x)
- ...

Sigmoid

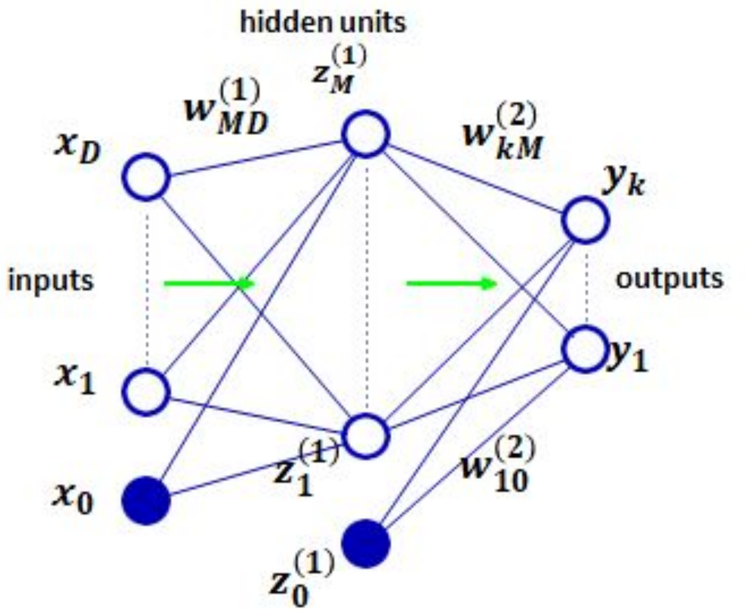


ReLu



Feed-forward Network

Connect together a number of these artificial neuron units into a feed-forward network (DAG). For instance, a network with one layer of hidden units implements the function:



$$y_k(x,w) = h^{(2)}\left(\sum_{j=1}^{M} w_{kj}^{(2)} h^{(1)}\left(\sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

Network training

Given a specified network structure, how do we set its parameters (weights)?

We define a criterion to measure how well our network performs, optimize against it.

- For regression, with training data $(x_n, t_n), t_n \in \mathbb{R}$, squared error naturally arises:

$$E(w) = \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2$$

Parameter Optimization

This was maximizing performance...

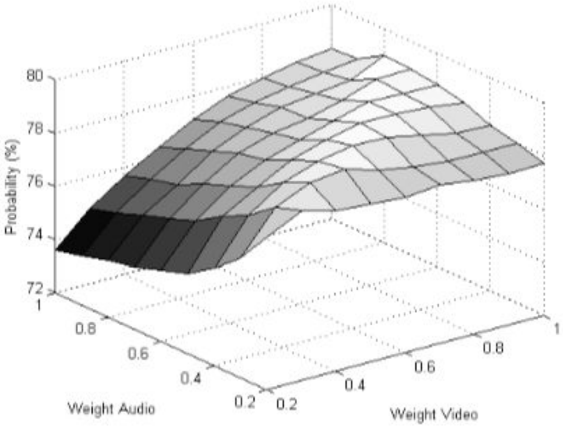
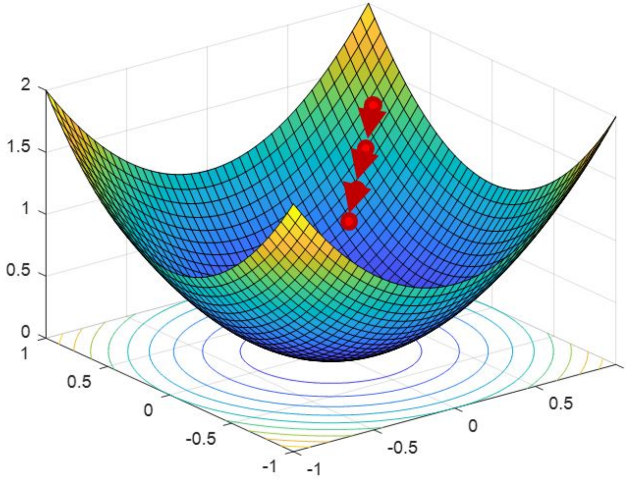


Figure 6: Influence of audio and video weights on vector fusion performance. Stable performance is observed if audio and video events are weighted in a ratio of 8 to 10.

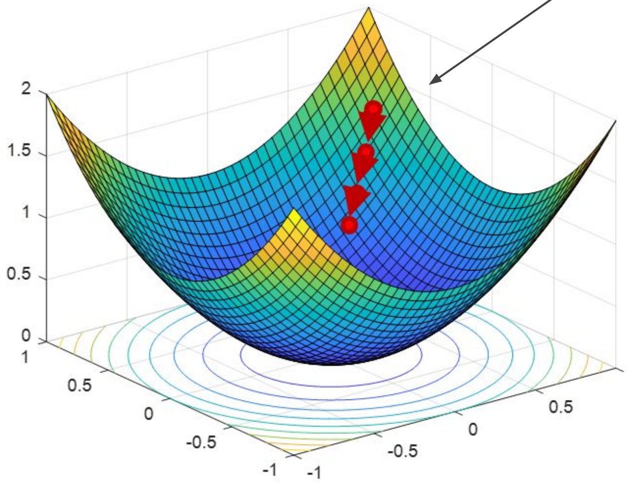
Now, we want to minimize error E(**w**).

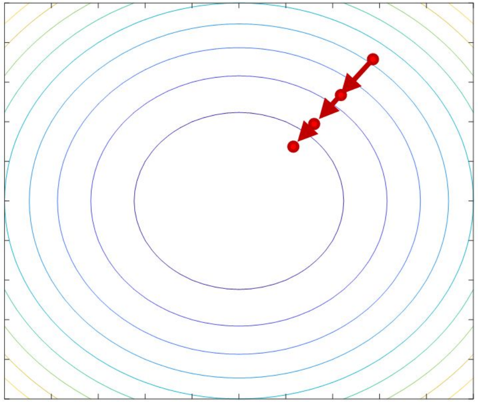


Numerical Solution: Gradient Methods

Similar to a search: which direction should we go? And by how much?

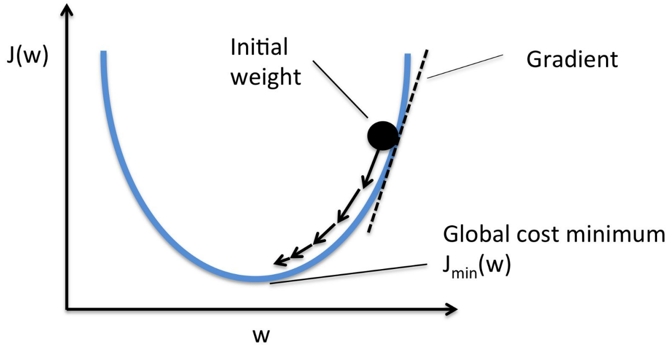
Slope = derivative!



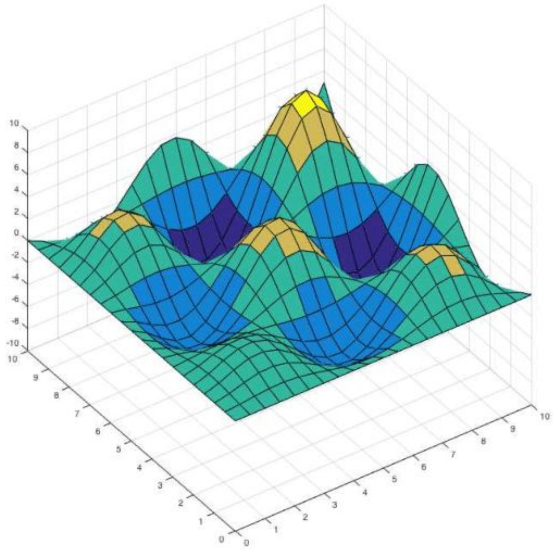


Error backpropagation

Backprop is an efficient method for computing error derivatives $\frac{\partial E_n}{\partial w_{ji}^{(m)}}$



Sometimes functions look nasty



Sometimes functions are **non-convex function** with local minima :(

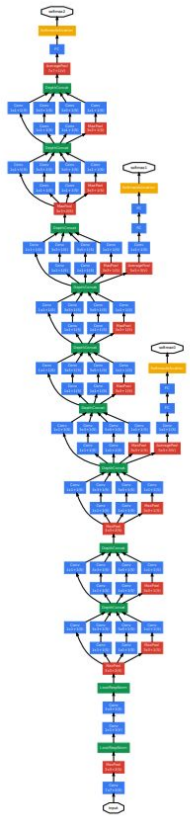
So you want to train a Deep Neural Network...

They have many layers...

Deep Neural Networks

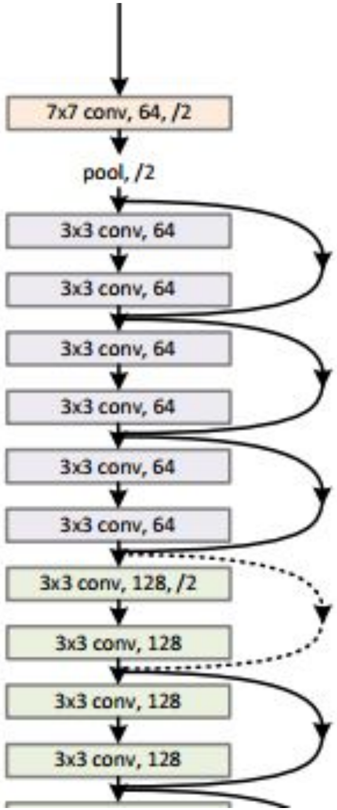
GoogLeNet

- GoogLeNet developed by Szegedy et al., CVPR 2015
- Modern deep network
- ImageNet top-5 error rate of 6.67% (later versions even better)
- Comparable to human performance (especially for fine-grained categories)



Deep Neural Networks

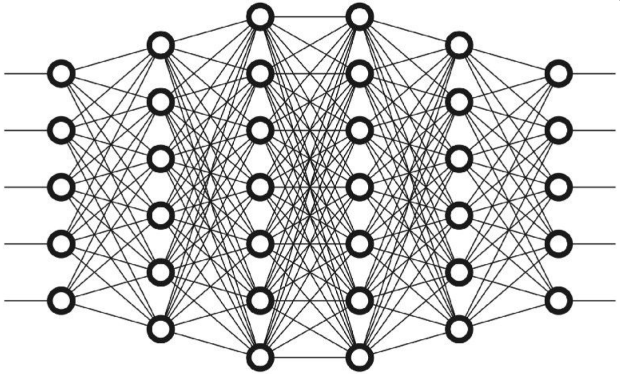
- ResNet developed by He et al., ICCV 2015
- 152 layers
- ImageNet top-5 error rate of 3.57%
- Better than human performance (especially for fine-grained categories)



Key Component 1: Many many layers

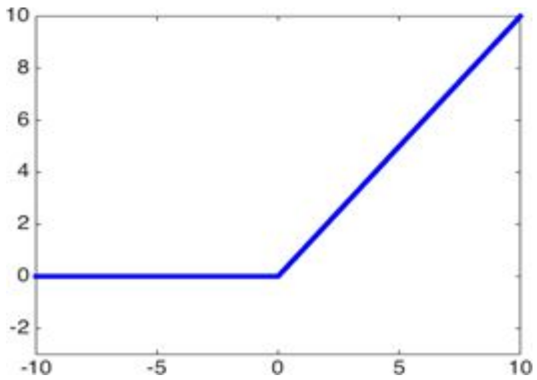
- **ResNet**: ≈152 layers (“shortcut connections”)
- GoogLeNet: ≈27 layers (“Inception” modules)
- VGG Net: 16-19 layers (Simonyan and Zisserman, 2014)
- AlexNet: 8 layers (Krizhevsky et al., 2012)

Allows for many tweakable weights, every layer like adding another dimension



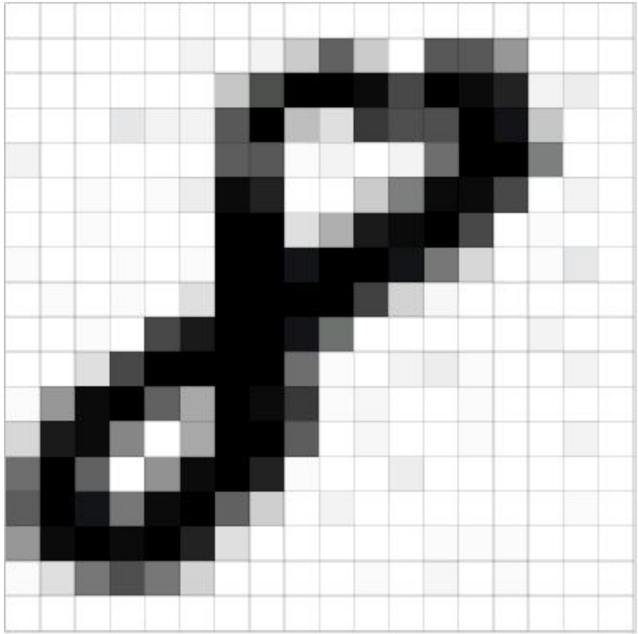
Key Component 2: ReLUs

- Vanishing gradient problem
  - If derivatives very small, no/little progress via stochastic gradient descent
  - Occurs with sigmoid function when activation is large in absolute value
- ReLU: $h(a_j) = max(0, a_j)$
- Non-saturating, linear gradients (as long as non-negative activation on some training data)
- Sparsity inducing



Key Component 3: Convolutional Filters

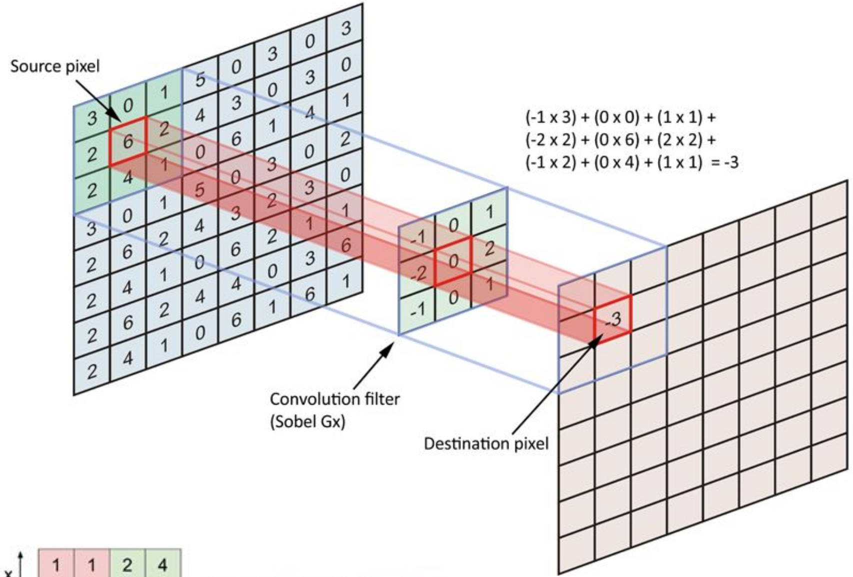
- Share parameters across network
- Reduce total number of parameters
- Provide translation invariance, useful for visual recognition

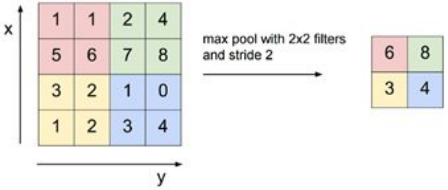


Convolutional Filters

Common Operations

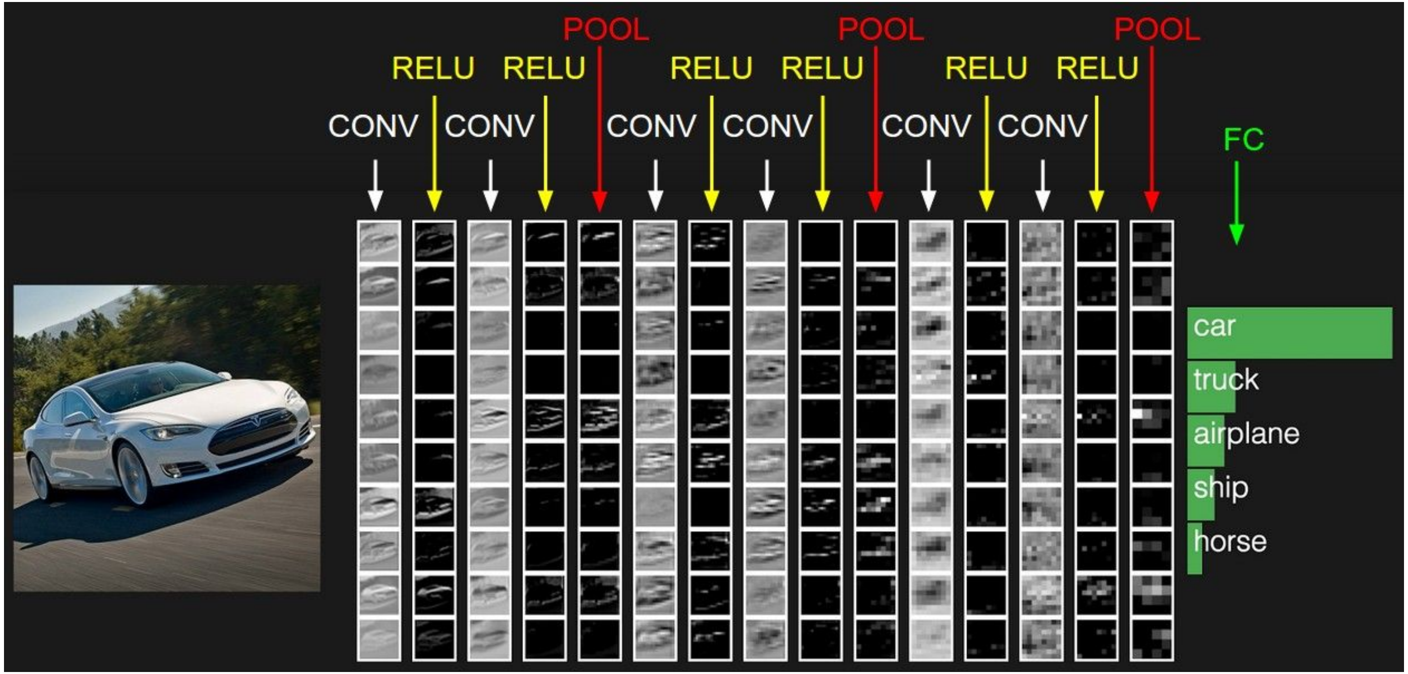
- Fully connected (dot product)
- Convolution
  - Translationally invariant
  - Controls overfitting
- Pooling (fixed function)
  - Down-sampling
  - Controls overfitting
- Nonlinearity layer (fixed function)
  - Activation functions, e.g. ReLU





Convolutional Filters

Example: Small VGG Net From Stanford CS231n



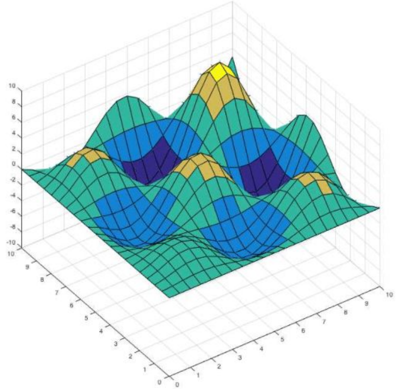
Key Component 4: Momentum

- Trick to escape plateaus / local minima
- Take exponential average of previous gradients

$$\overline{\frac{\partial E_n}{\partial w_{ji}}}^{\tau} = \overline{\frac{\partial E_n}{\partial w_{ji}}}^{\tau} + \alpha \overline{\frac{\partial E_n}{\partial w_{ji}}}^{\tau-1}$$

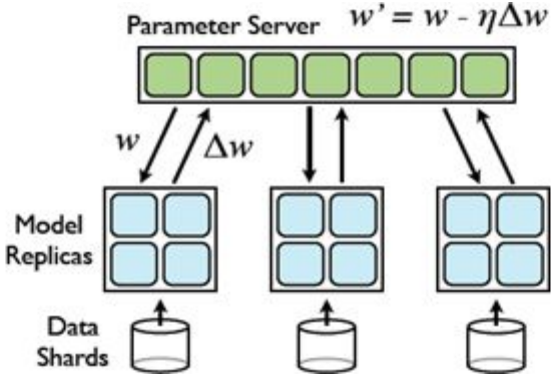
- Maintains progress in previous direction

Sometimes functions are **non-convex function** with local minima :(



Key Component 5: Asynchronous Stochastic Gradient Descent

- Big models won’t fit in memory
- Want to use compute clusters (e.g. 1000s of machines) to run stochastic gradient descent
- How to parallelize computation?
- Ignore synchronization across machines
- Just let each machine compute its own gradients and pass to a server storing current parameters
- Ignore the fact that these updates are inconsistent
- Seems to just work (e.g. Dean et al. NIPS 2012)



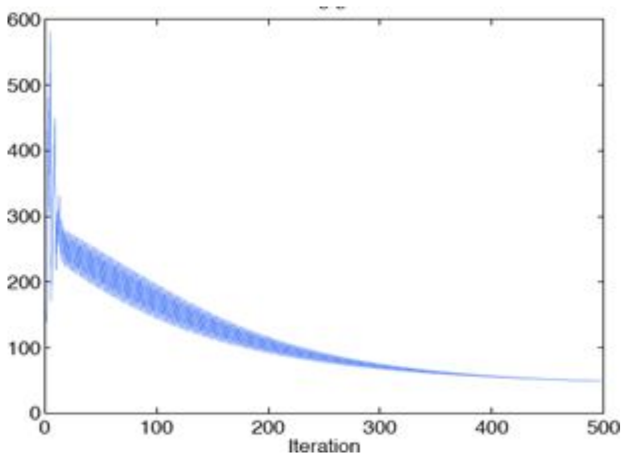
Key Component 6: Learning Rate

Step size

- How to set learning rate η?:

$$\boldsymbol{w}^{\tau} = \boldsymbol{w}^{\tau-1} + \eta \nabla \boldsymbol{w}$$

- Option 1: Run until validation error plateaus. Drop learning rate by x%
- Option 2: Adagrad, adaptive gradient. Per-element learning rate set based on local geometry (Duchi et al. 2010)



Key Component 7: Data Augmentation

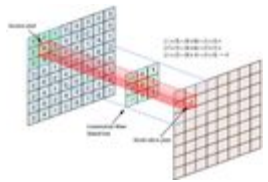
- Augment data with additional synthetic variants (10x amount of data)
- Or just use synthetic data, e.g. Sintel animated movie (Butler et al. 2012)

Key Component 8: Data and Compute

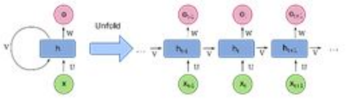
- Get lots of data (e.g. ImageNet)
- Get lots of compute (e.g. CPU cluster, GPUs)
- Cross-validate like crazy, train models for 2-3 weeks on a GPU
- Researcher gradient descent (RGD) or Graduate student descent (GSD): get 100s of researchers to each do this, trying different network structures

Neural Network Architectures

Check DeepMind lectures



- Convolutional neural network (CNN)
  - Has translational invariance properties from convolution
  - Common used for computer vision
- Recurrent neural network (RNN)
  - Has feedback loops to capture temporal or sequential information
  - Useful for handwriting recognition, speech recognition,reinforcement learning
  - Long short-term memory (LSTM): special type of RNN with advantages in numerical properties
- Others
  - General feedforward networks, variational autoencoders (V VAEs, generative adversarial networks



Training Neural Networks

- Data preprocessing
  - Removing bad data
  - Transform input data (e.g. rotating, stretching, adding noise)

- Training process (optimization algorithm)
  - Choice of loss function (eg. L1 and L2 regularization)
  - Dropout: randomly set neurons to zero in each training iteration
  - **Learning rate** (step size) and other hyperparameter tuning

- Software packages: efficient gradient computation
  - Caffe, Torch, Theano, TensorFlow

More Information

- https://sites.google.com/site/_deeplearningsummerschool
- http://tutorial.caffe.berkeleyvision.org/
- ufldl.stanford.edu/eccv10-tutorial
- http://www.image-net.org/challenges/LSVRC/_2012/supervision.pdf
- Courses: Deep Learning, Natural Language Processing, Computer Vision

Sequential Neural Network Models

For temporal or dynamic data

Example

Predict where the person will go next based on their previous locations and body pose.

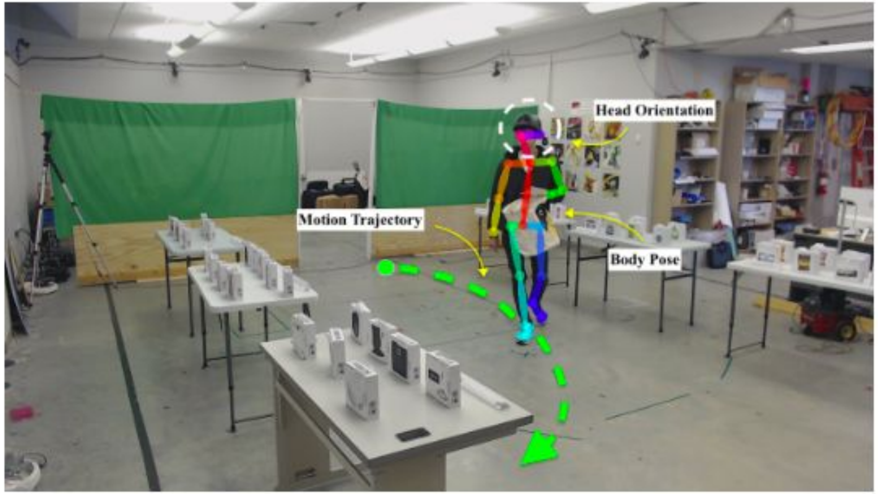


Figure 1: Use of human motion, body pose and head orientation to infer navigational intent.

Zhang et al.

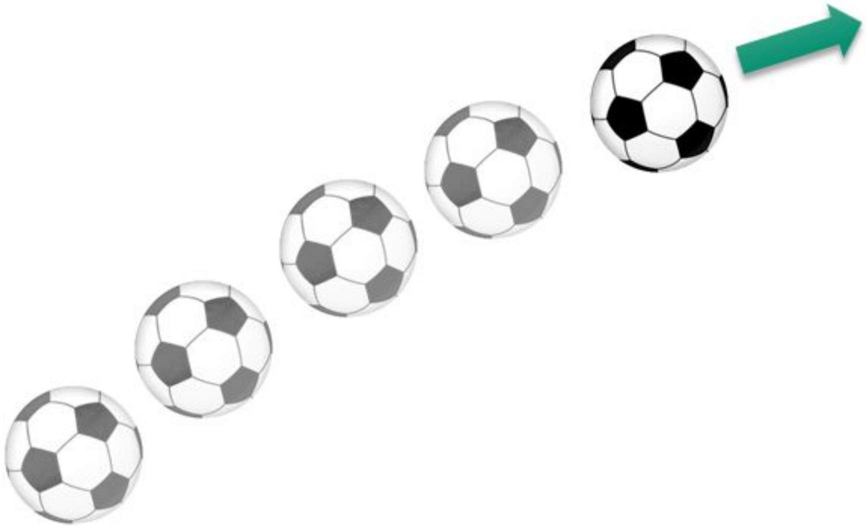
ROSIE LAB

Recurrent Neural Networks

- Sequential input / output
  - Many inputs, many outputs $x_{1:T} \rightarrow y_{1:s}$
    - e.g. object tracking, speech recognition with HMMs; online/batch processing
  - One input, many outputs $x \rightarrow y_{1:s}$
    - e.g. image captioning
  - Many inputs, one output $x_{1:T} \rightarrow y$
    - e.g. video classification

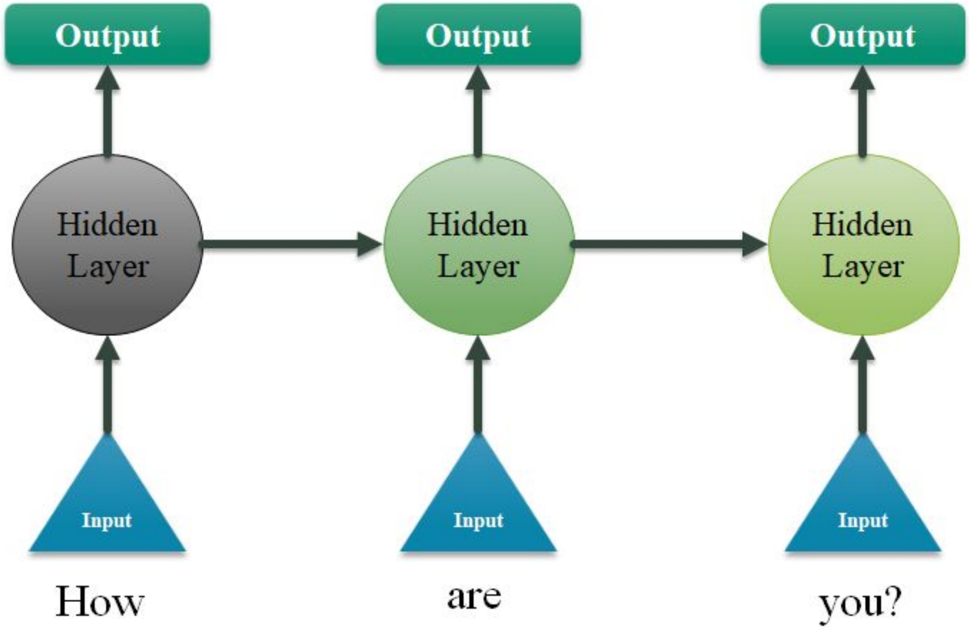
Recurrent Neural Networks

- Basic idea: maintain a state h_t
- State at time t depends on input x_t and previous state x_{t-1}



Recurrent Neural Networks

- Basic idea: maintain a state h_t
- State at time t depends on input x_t and previous state x_{t-1}



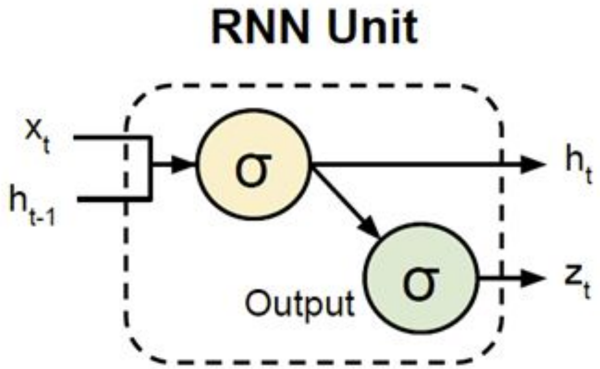
Recurrent Neural Networks

- Basic idea: maintain a state h_t
- State at time t depends on input x_t and previous state x_{t-1}
- It’s a neural network, so relation is non-linear function of these inputs and some parameters **W**:

$$h_t = f_x(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{W}) = \sigma(W_x x_t + W_h h_{t-1})$$

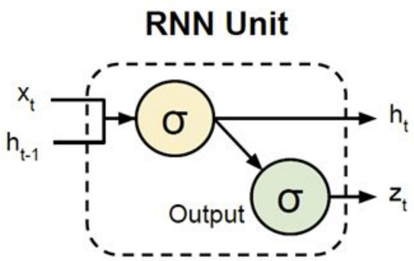
- Parameters **W** and function f(·) reused at all time steps
- Output z_t also depends on the hidden state:

$$z_t = f_z(\boldsymbol{h}_t; \boldsymbol{W}_z) = \sigma(W_z h_t)$$



Recurrent Neural Networks

- Has feedback loops to capture temporal or sequential information
- Has the ability to learn tasks that require “memory” of events from many time steps ago
- Long short-term memory (LSTM): special type of RNN with advantages in numerical properties

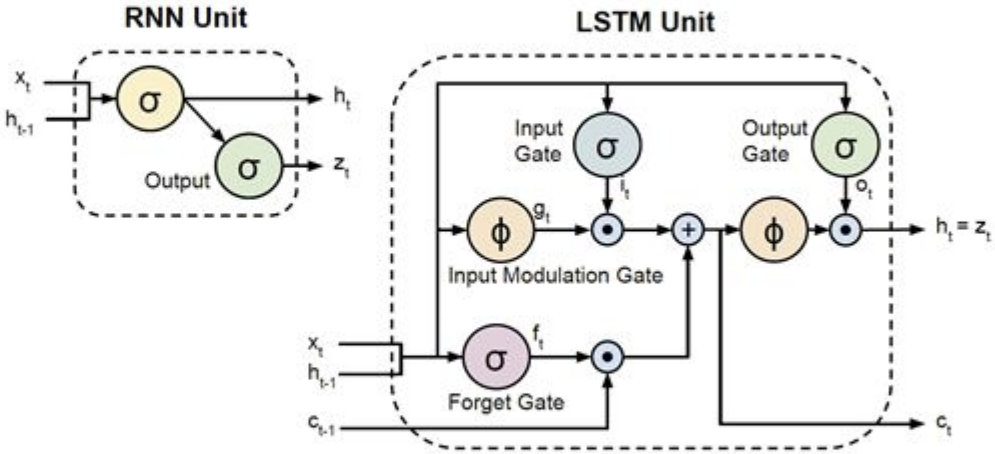


Recurrent Neural Networks

- Basic RNN not very effective
- Need many time steps / complex model for challenging tasks
- Gradients in learning are a problem
  - Too large: can be handled with gradient clipping (truncate gradient magnitude)
  - Too small: can be handled with network structures / gating functions (LSTM, GRU)

Long Short-Term Memory

- Hochreiter and Schmidhuber, Neural Computation 1997
- (Figure from Donohue et al. CVPR 2015)
- Gating functions g(·), f(·), o(·) reduce vanishing gradients



More Information

DeepMind x UCL Deep Learning Lectures

- Readings:
    - http://www.deeplearningbook.org/contents/rnn.html
    - https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577
    - https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html
- Recurrent neural networks, can model sequential inputs/outputs
    - Input includes state (output) from previous time
    - Different structures:
    - RNN with multiple inputs/outputs
    - Gated recurrent unit (GRU)
    - Long short-term memory (LSTM)
    - Error gradients back-propagated across entire sequence