Special Topic:
Deep Learning

Hello!

We are Zach Jones and Sohan Nipunage

You can find us at:
zdj21157@uga.edu
smn57958@uga.edu

Outline

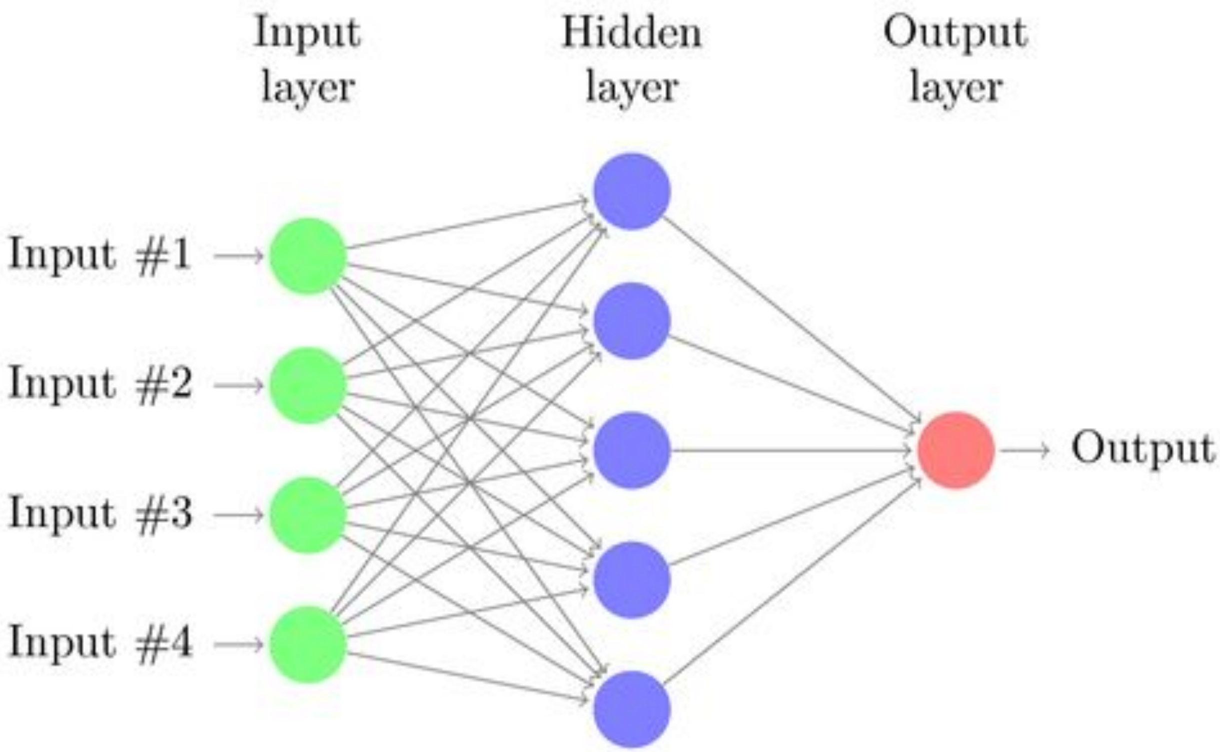
I. What is Deep Learning?
II. Why Deep Learning?
III. Common Problems
IV. Popular Use Cases
    A. Convolutional Nets
    B. Recurrent Nets
    C. Deep RL
    D. Unsupervised
V. Current Research
VI. Q & A

1.

What is Deep Learning?

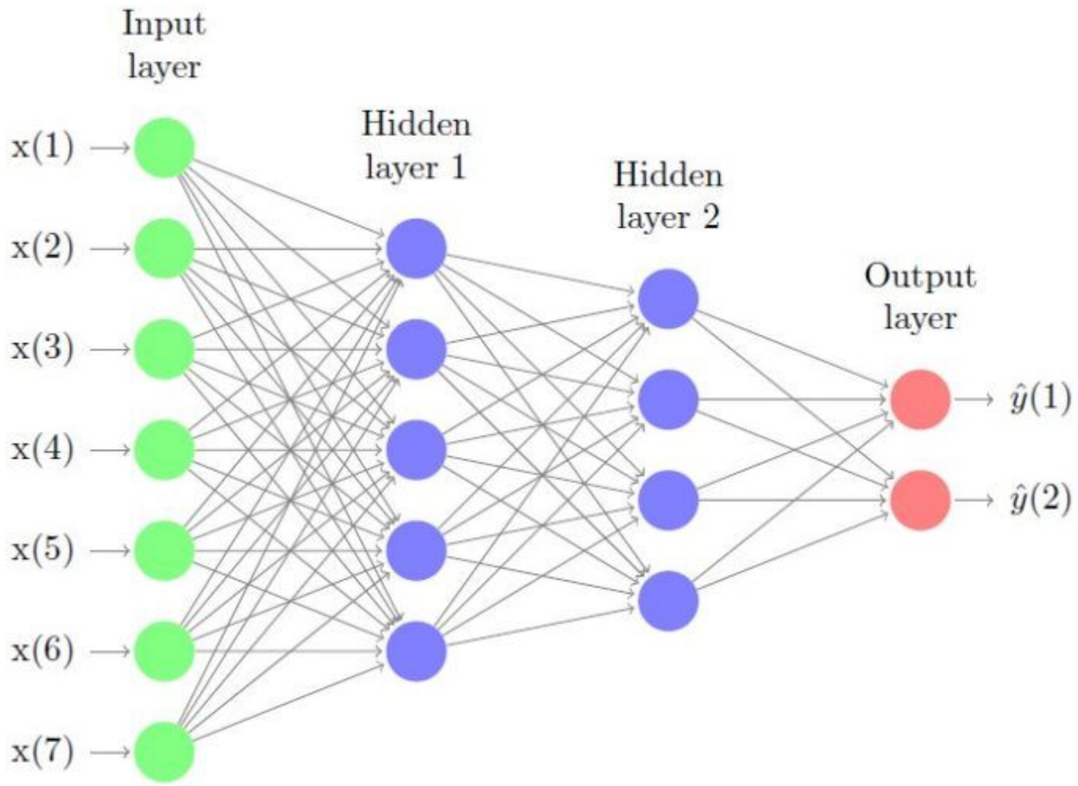
More than just a buzzword!

Neural Networks



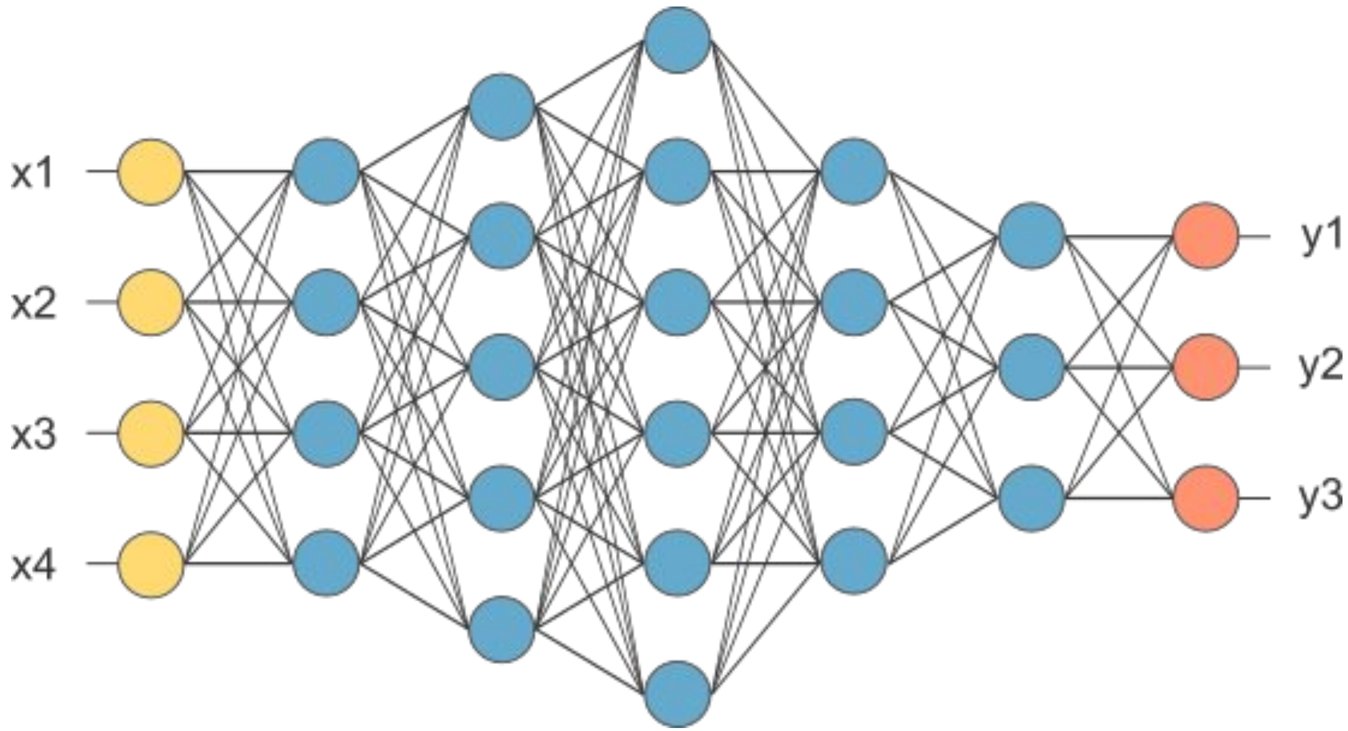
Single-layer (shallow) Neural Network

Deep Neural Networks



Deep (but not *that* deep) Neural Network

Deep Neural Networks



Deeper Neural Network

2.

Why Deep Learning?

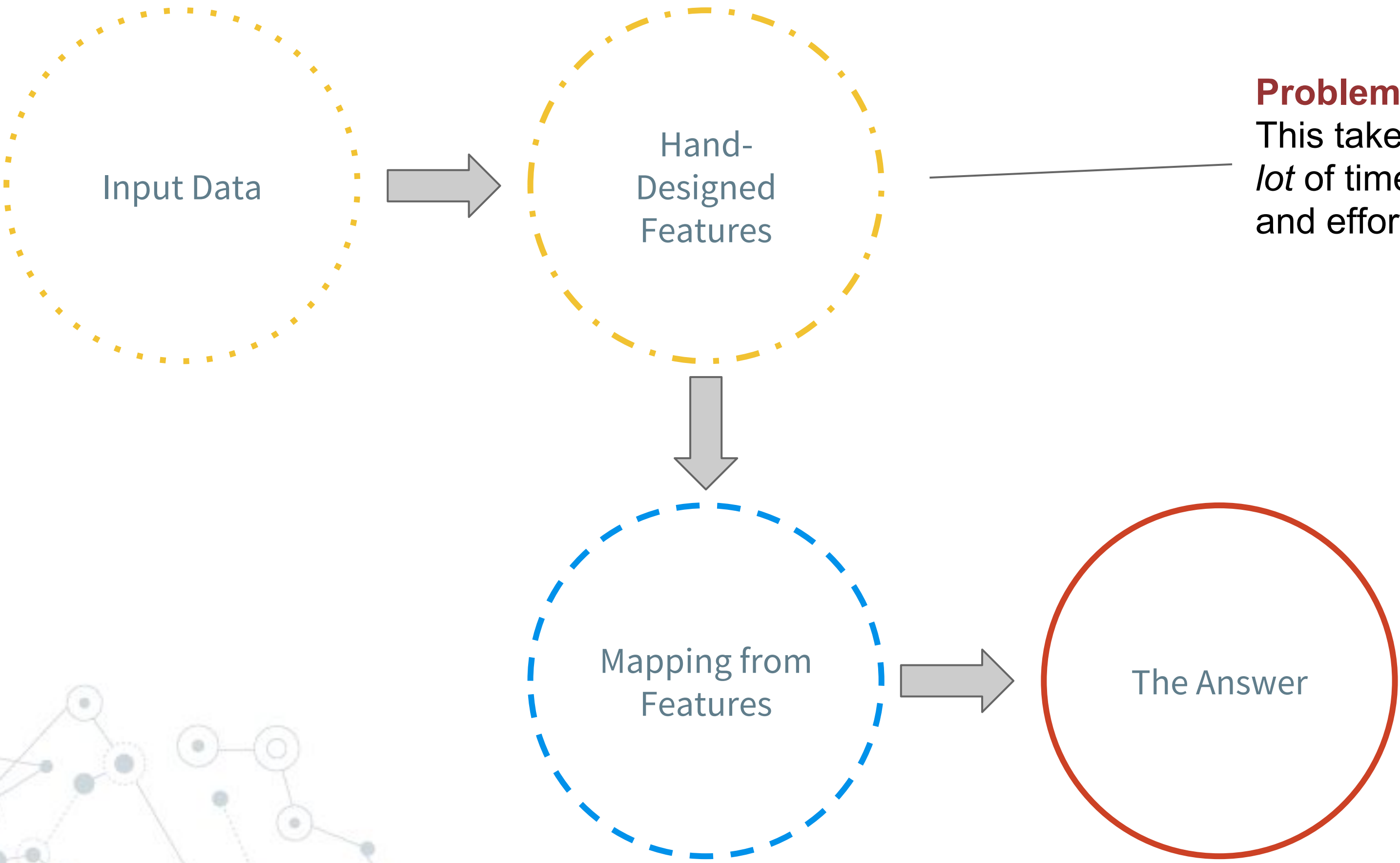
Is there a point to all of this?

History of Learning Systems

In the olden days: Expert Systems

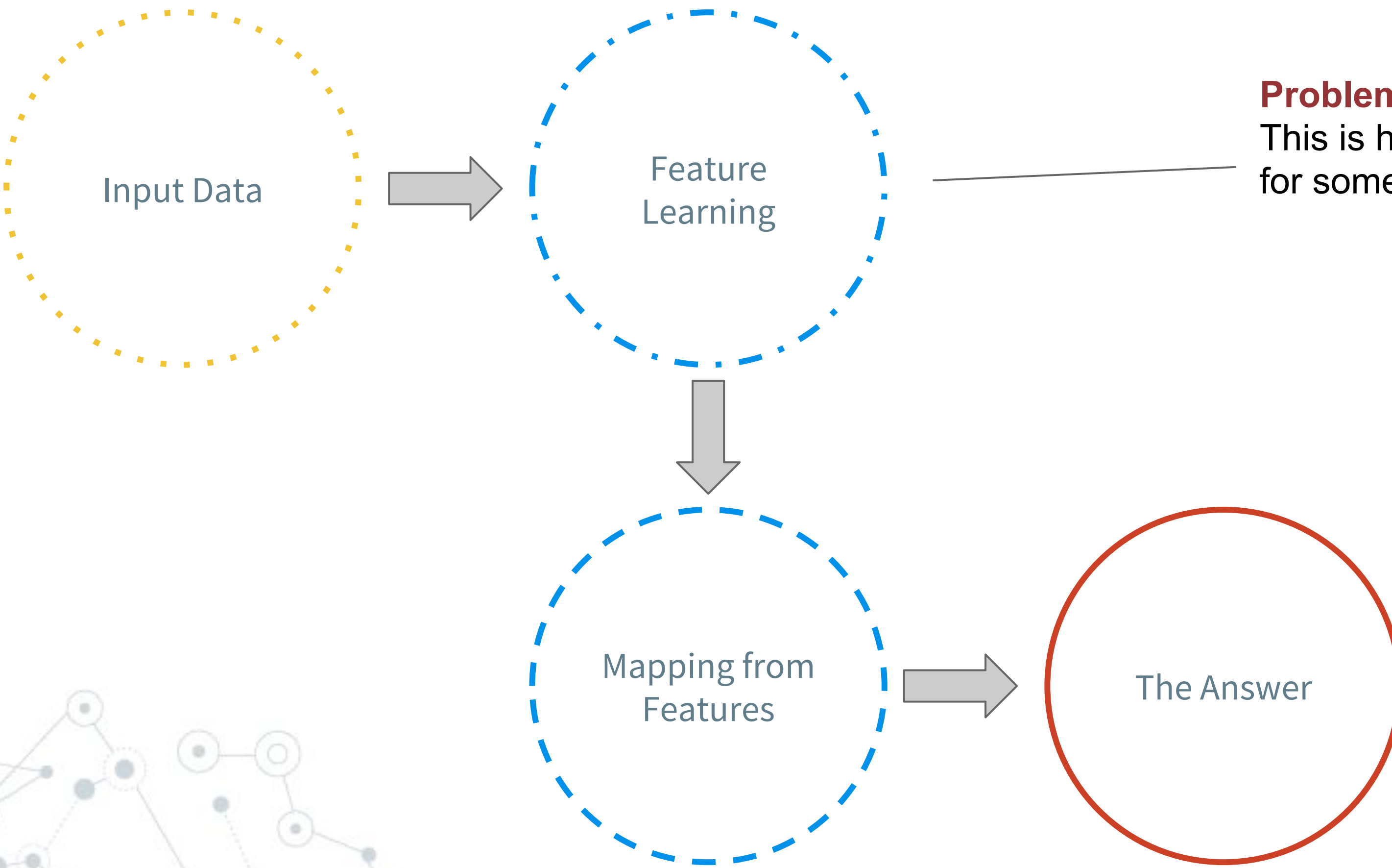
Problem:
This takes a
lot of time
and effort

Next Step: Classical Machine Learning



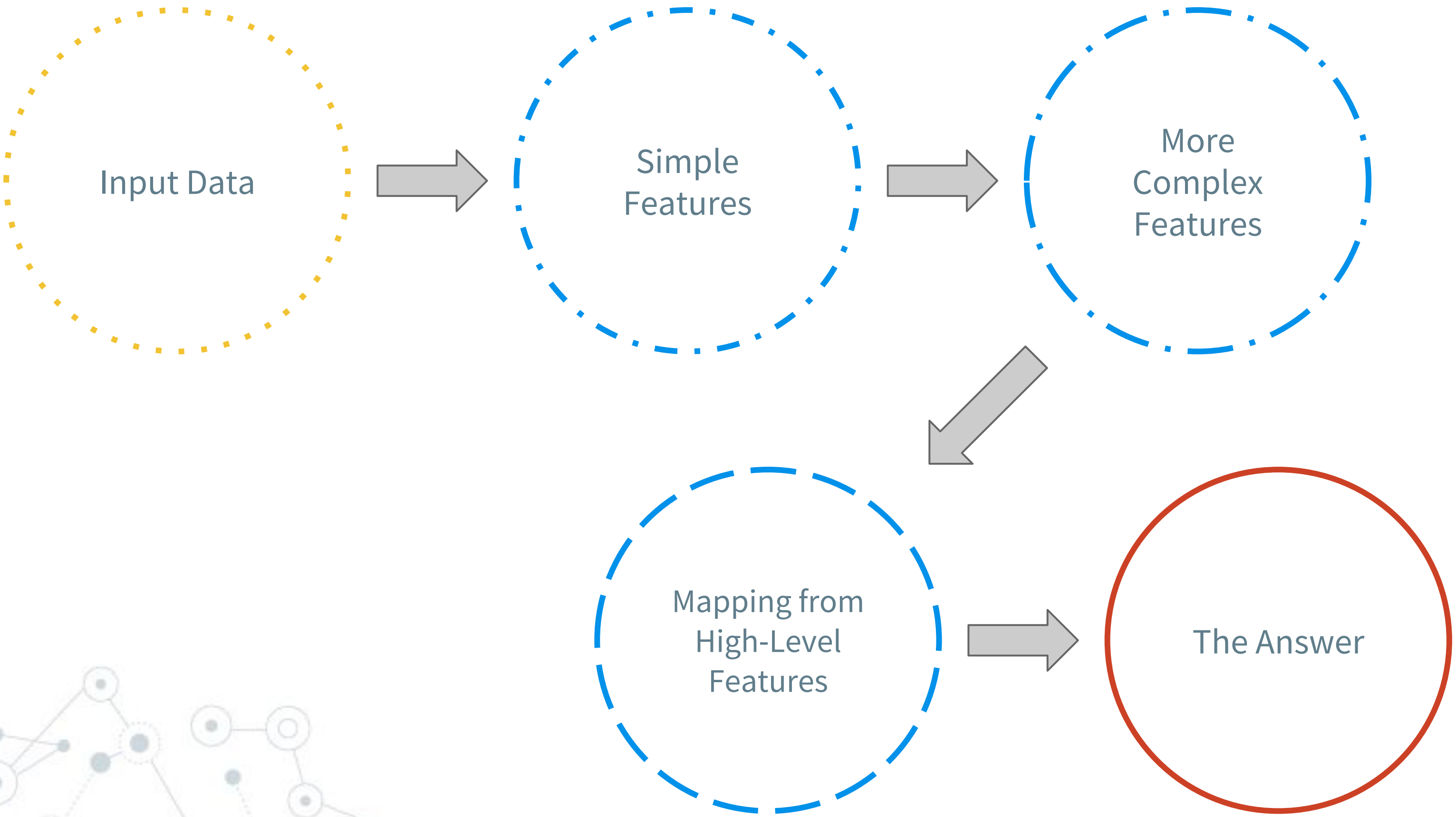
Problem:
This takes a *lot* of time and effort

Next Step: Representation Learning



Problem:
This is hard to do for some domains

The Present: Deep Learning



Why Deep Learning

- More sophisticated models
  - learn very complex non-linear functions
- Layers as a mechanism for abstraction
- **Automatic feature extraction**
- Works well in practice

Why Deep Learning

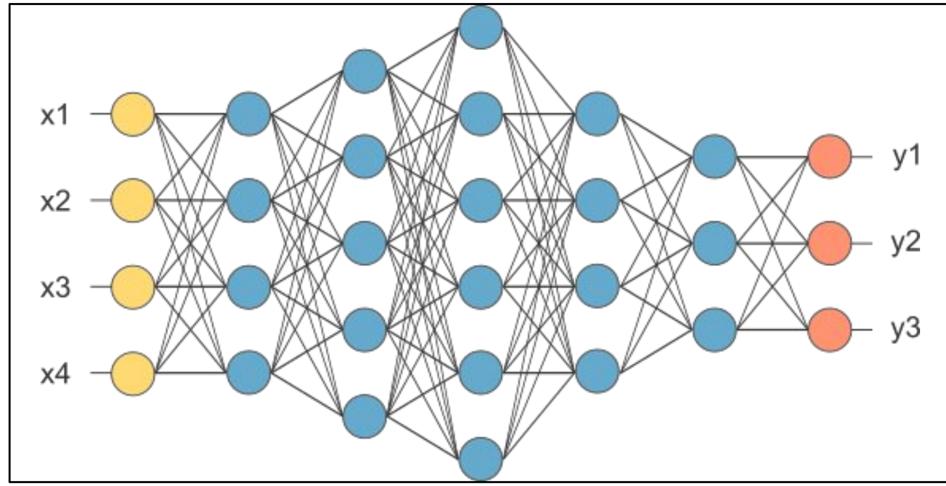
- Loads of data
- Very flexible model
  - can represent complex functions
- Powerful feature extraction
  - Defeat the curse of dimensionality

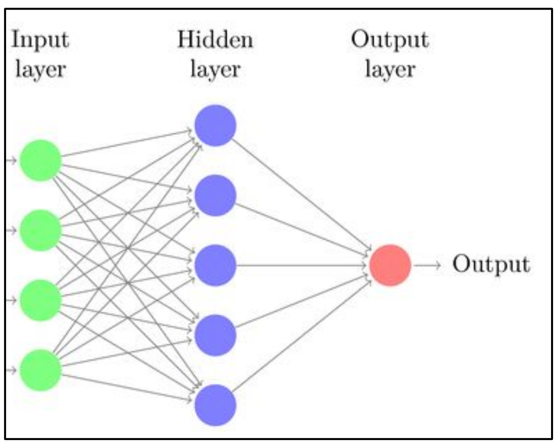
Multiple Levels of Abstraction

Capturing high-level abstractions allows us to achieve amazing results in difficult domains

No Free Lunch

Anything you can do, I can do better! I can do anything better than you!





Yes, including overfitting...

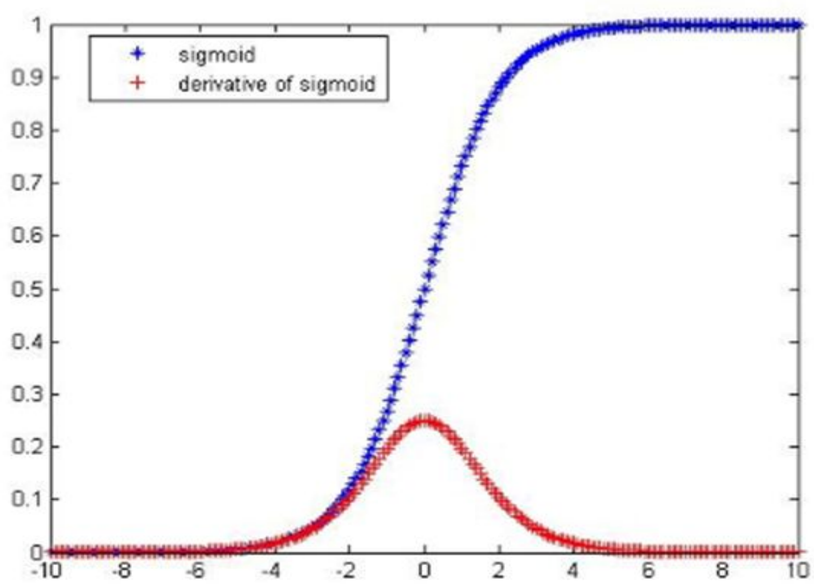
3.

Common Problems

Vanishing Gradients, Parameter Explosion, Overfitting, Long Training Time, and other disasters!

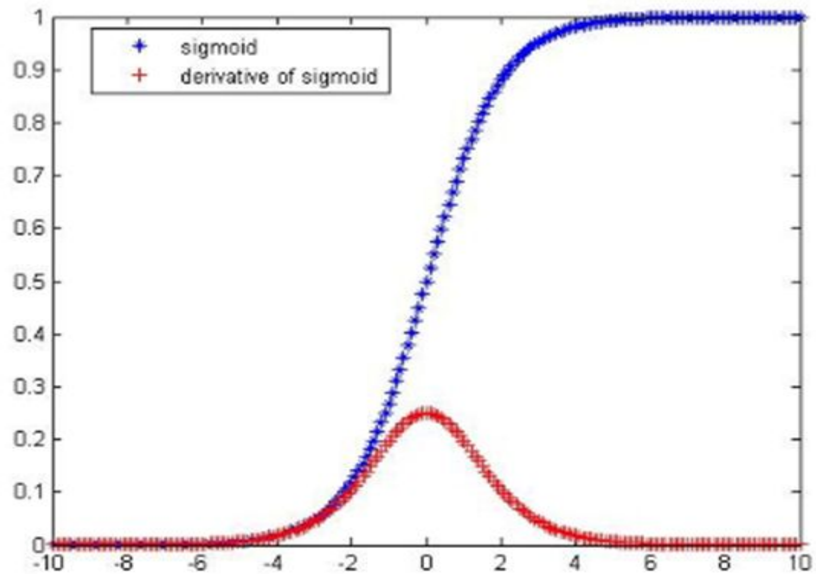
Problem: Vanishing Gradients

- Towards either end of the sigmoid function, Y values tend to respond very less to changes in X
- Gradient in that region is going to be too small.



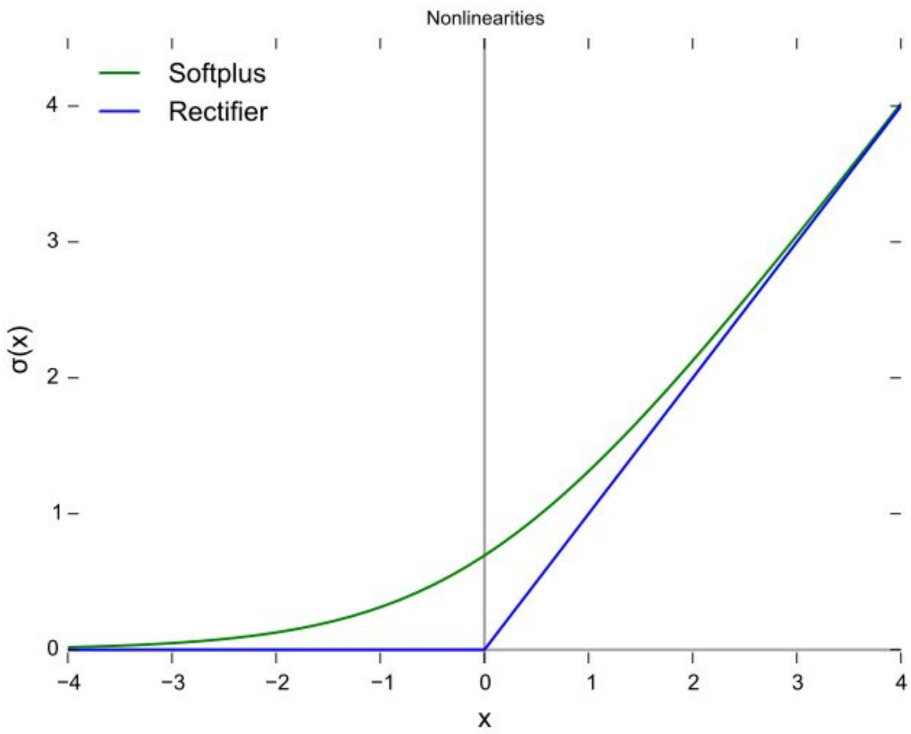
Problem: Vanishing Gradients

- Backpropagation
  - o=sig(WX+b)
  - ∂o/∂W=o(1-o) X
- Chains of sigmoid derivatives
  - Eating the gradient
  - Narrow range



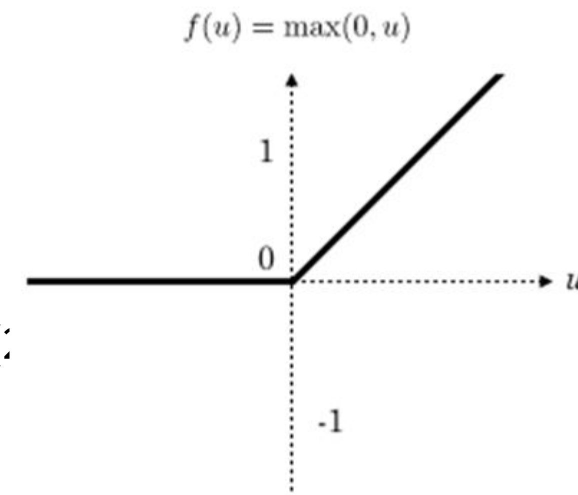
Solution: Rectified Linear Units

- Rectifier:

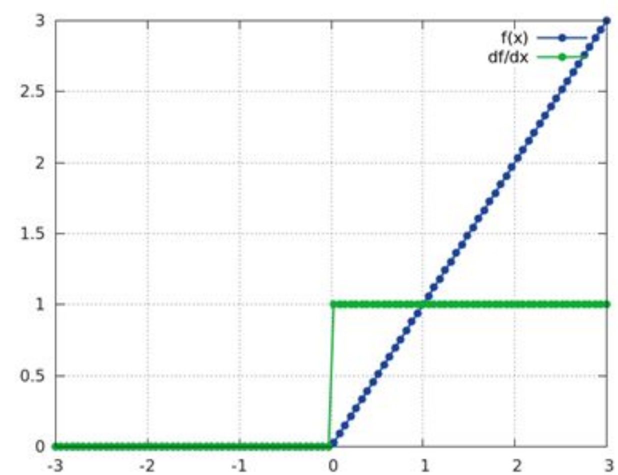


Solution: Rectified Linear Units

- Rectified Linear Units (ramp)
  - f(x)=max(0,x)
  - Derivative: All in or all out (unit step)
    - f′(x)=1 if x>0 else 0
  - First proposed as activation by Hahnloser et al (
  - Popularized by Hinton in his RBM (2010).



- Dead ReLUs
  - LeakyReLU: f(x)=max(x,0.01x)
  - PReLU: f(x)=max(x,ax)



Solution: Rectified Linear Units

Unit variance weights $Var[W] = 1$

Glorot et al (2010):

- $Var[W] = nin * Var[wi]$ (since iid)
- $Var[w_i] = \frac{1}{n_{in}}$
- Eg, sample from $U[-\frac{1}{\sqrt{n_{in}}}, +\frac{1}{\sqrt{n_{in}}}]$ or $N[0, \frac{1}{n_{in}}]$

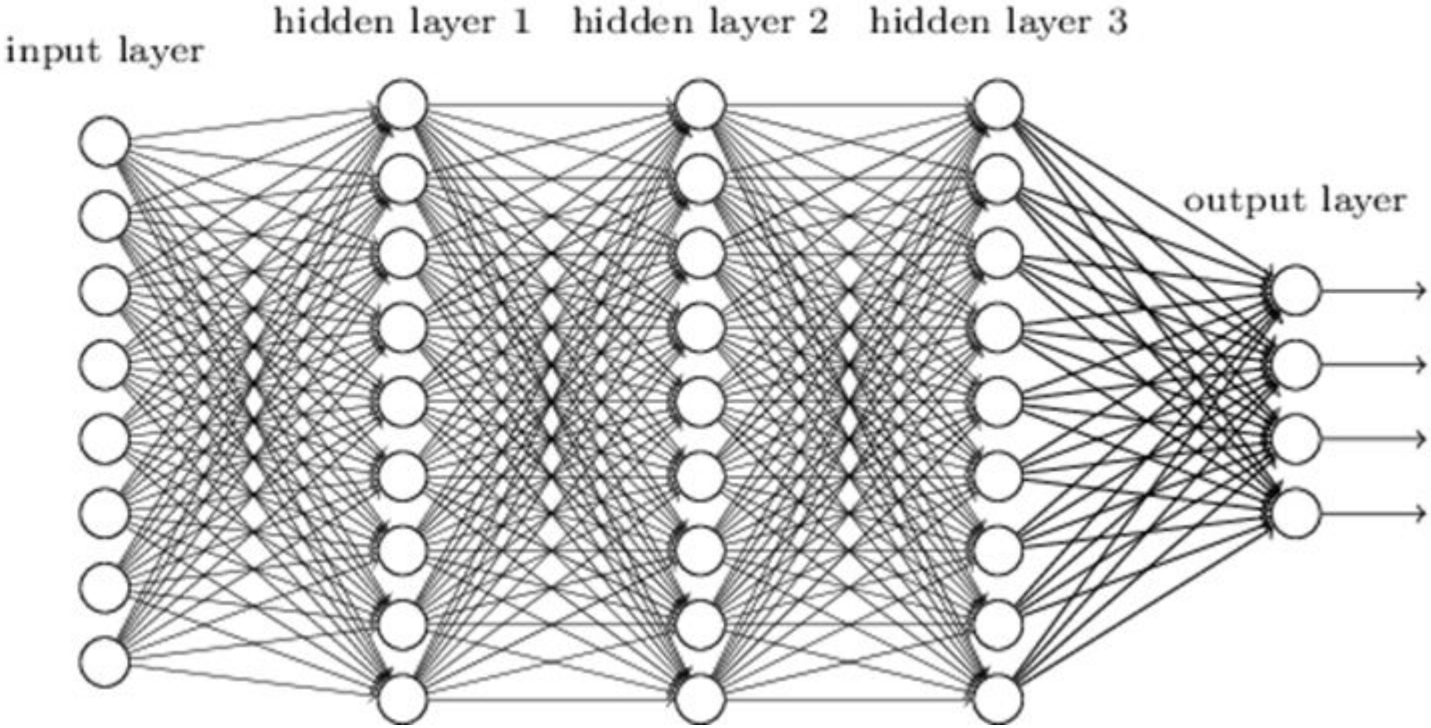
He et al (2015):

- $Var[w_i] = \frac{2}{n_{in}+nou_t}$

Solution: Rectified Linear Units

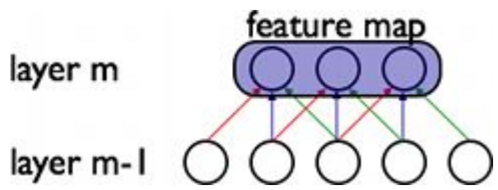
- All You Need Is A Good Init (2015):
  - Initialize from N(0,1) or U[-1,1]
  - Orthonormalize the weights (Singular Value Decomposition-SVD)
  - Unit singular values in all directions
  - Keep scaling down until unit variance

Problem: Parameter Explosion



Solution: Shared Weights

- Each filter h_i is replicated across the entire visual field.
- These replicated units share the same parameterization (weight vector and bias) and form a feature map.

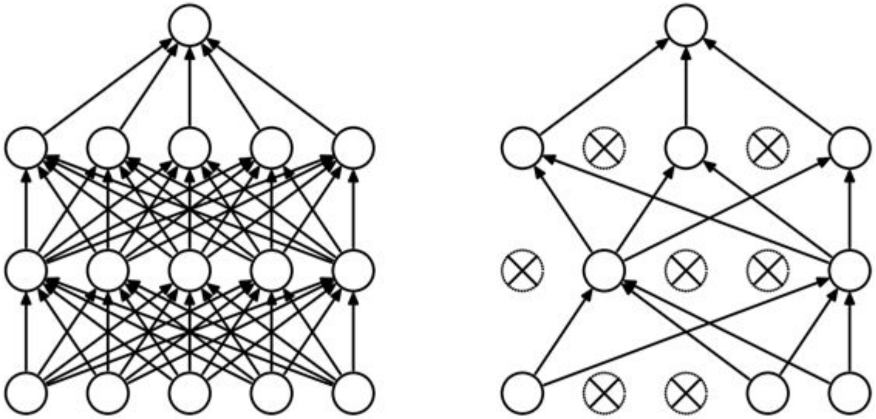


Solution: Regularization, Dropout, and Normalization

- Regularization :
  - Make some minima more appealing than others
  - Smooth the search space (less jagged)
  - Norm-based
  - L1 (sparse weights)
  - L2 (weight decay)

Solution: Regularization, Dropout, and Normalization

- Dropout:
  - Randomly deactivating units in feature maps
  - Forces all parts to be responsible for the output
  - Practically becomes an Ensemble of networks



(a) Standard Neural Net (b) After applying dropout.

Solution: Regularization, Dropout, and Normalization

- Batch Normalization:
  - Learns to adjust the mean and variance of the data
  - Helps combat overfitting by removing circumstantial data statistics
  - Helps keeping the gradients strong

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Problem: Long Training Time

- Long training time may take upto days for computing.

Solution: Modern GPUs and TPUs

- GPUs allowed for much faster training time (days to hours).
- The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks.
- cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers.
- cuDNN is part of the NVIDIA Deep Learning SDK.

Solution: Modern GPUs and TPUs

- A **tensor processing unit** (**TPU**) is an AI accelerator application-specific integrated circuit (ASIC) developed by Google specifically for neural network machine learning.
- The chip has been specifically designed for Google's TensorFlow framework

4.

Popular Use Cases

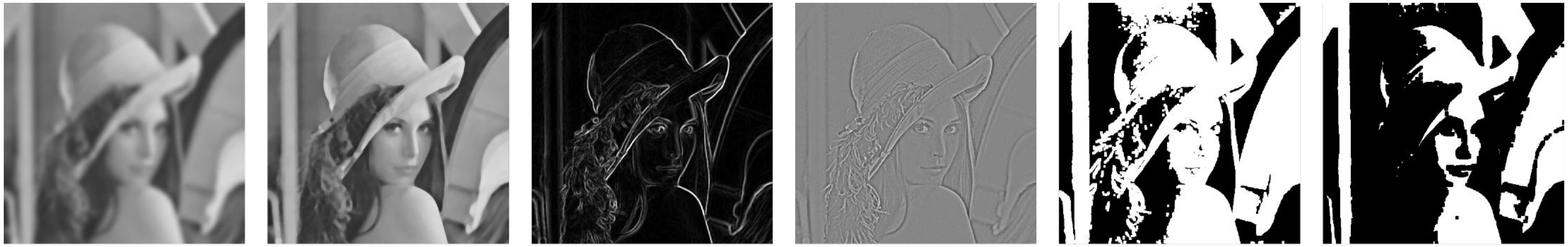
Let’s see what all the cool kids are doing...

Convolutional Neural Networks

Image and Video Processing

Image Processing

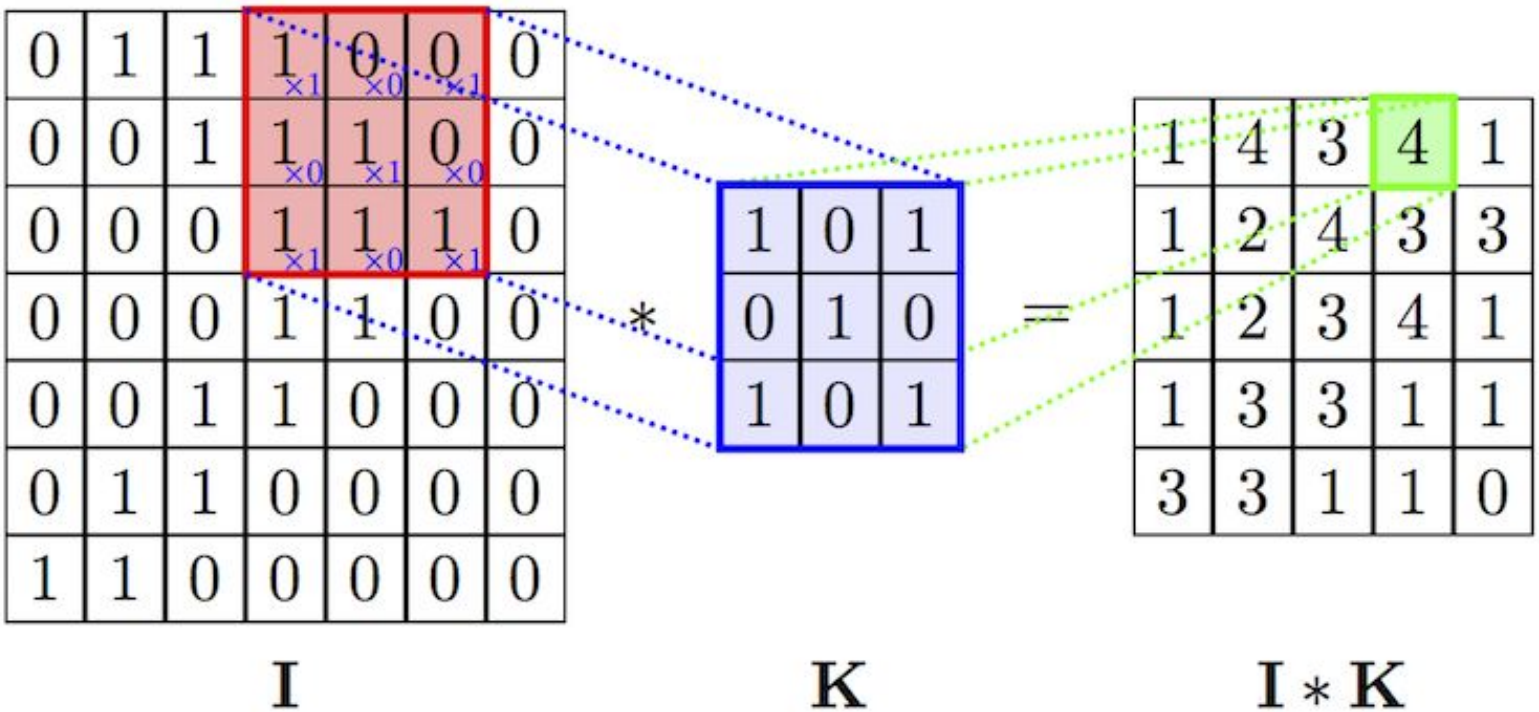
- Computer vision
  - Explosive spatial domain
  - 256 x 256 RGB image → 256 x 256 x 3 = **196,000 inputs!**
- Traditional Image processing:



What if we could learn the filters automatically?

Enter: Convolutional Neural Nets

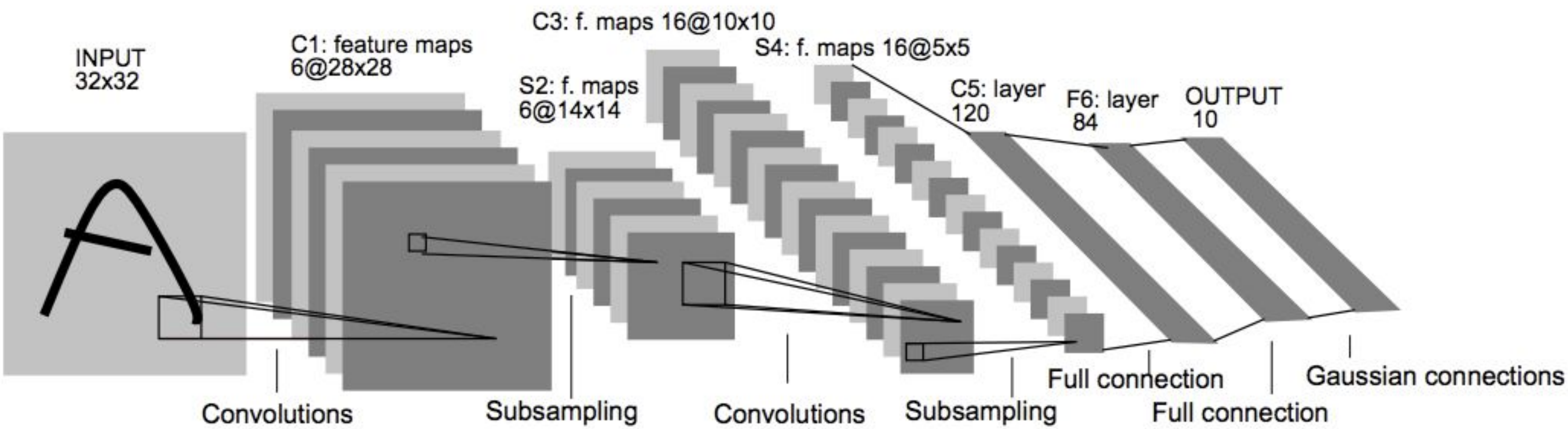
Convolution Operation



Convolutional Layers

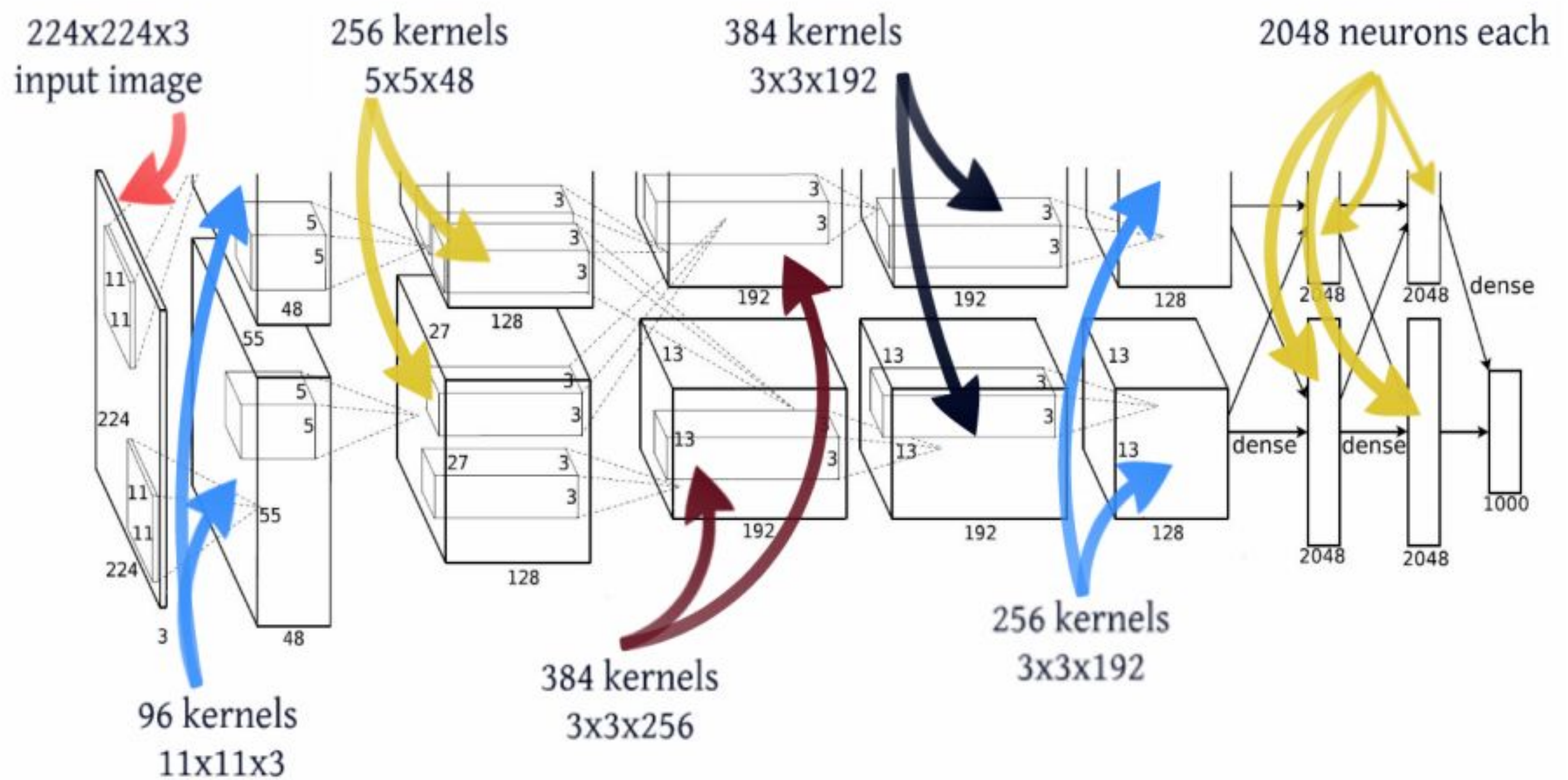
- Layer parameters consist of a set of learnable filters
- Key idea: neurons only look at small region of input
- Convolutional layer maps from 3D input to 3D output
- Output size determined by hyperparameters:
    - **receptive field**: *n* x *m* x *l* region of previous layer
    - **depth** = number of filters to apply to a region
    - **stride** = by how many pixels do we slide the receptive field

LeNet (1998)

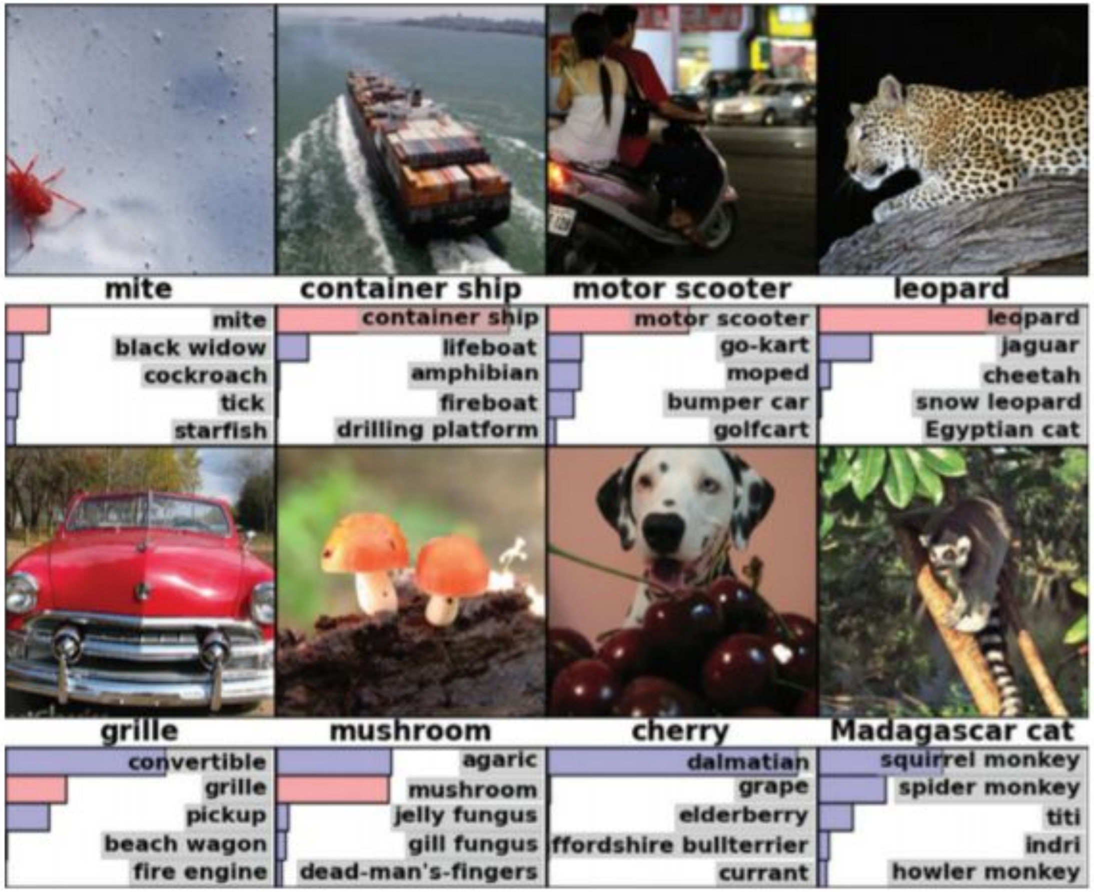


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

AlexNet (2012)

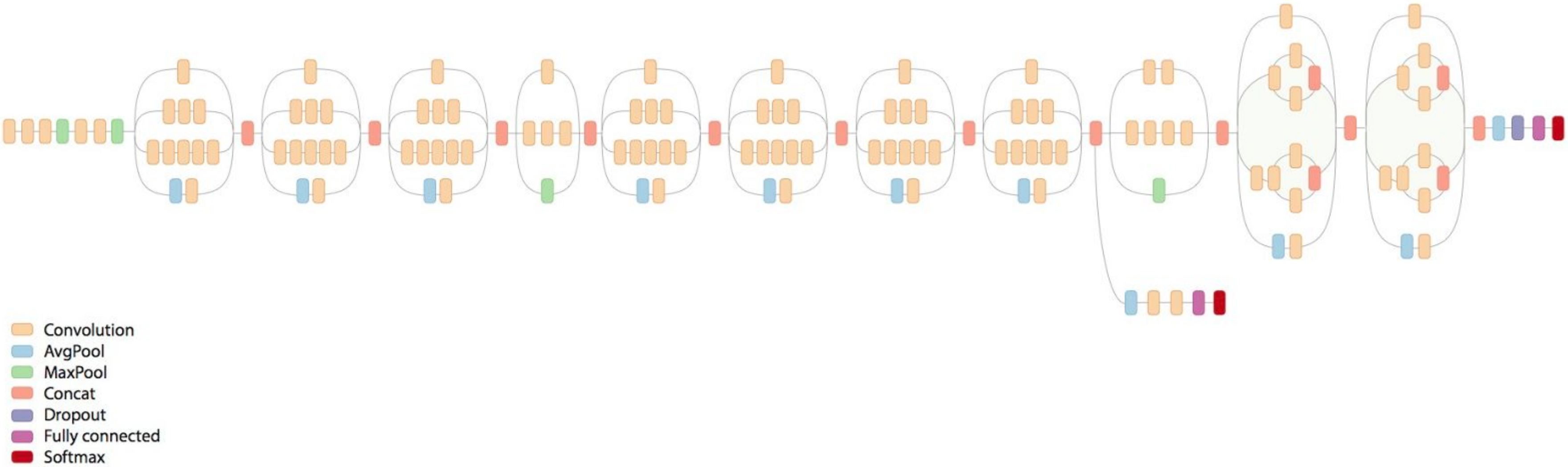


AlexNet Classifications



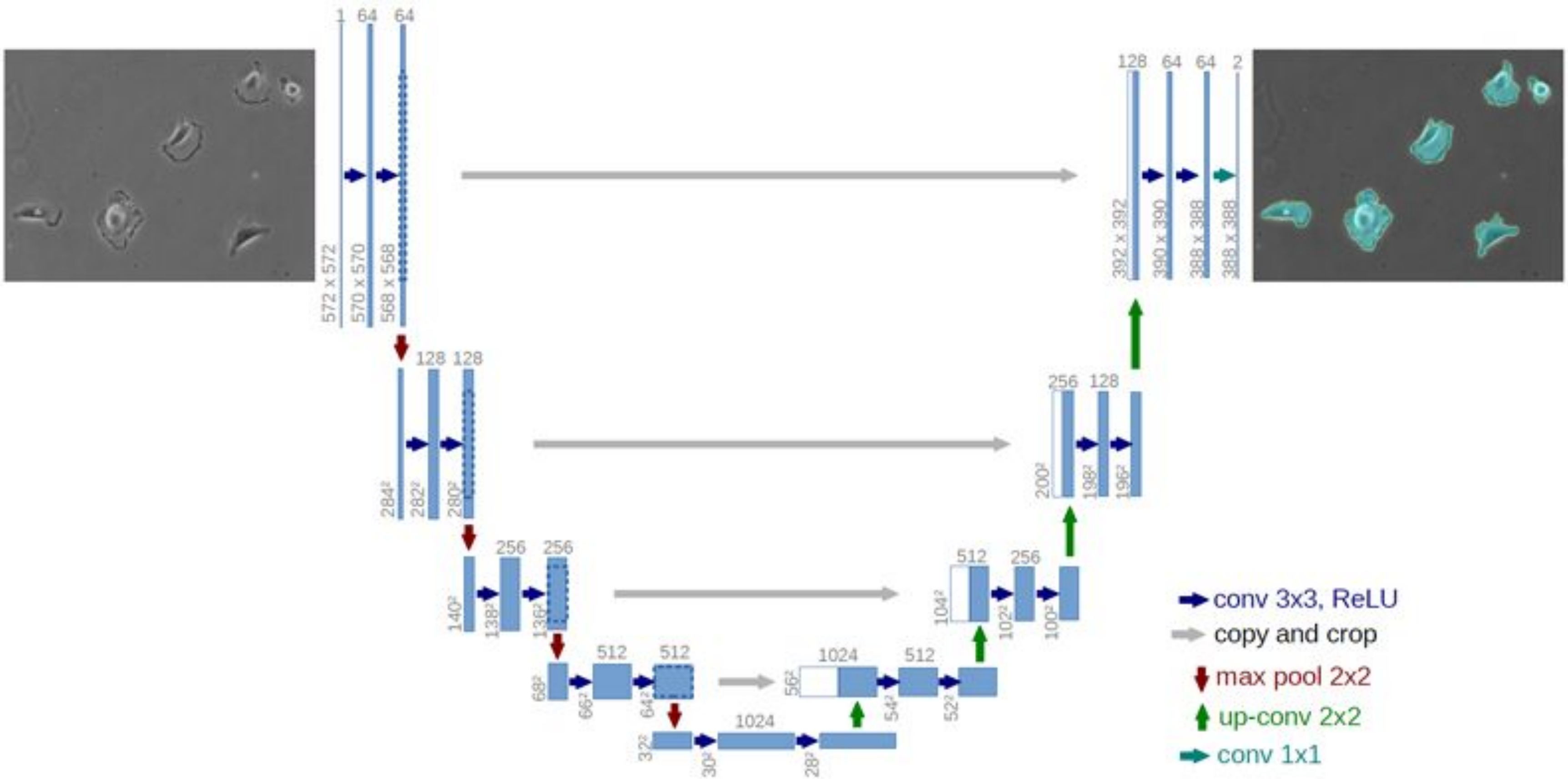
Top-5 Error Rate: 15.3%

Google “Inception” Network (2015)



Top-5 Error Rate: 6.67%

U-Net (2015)



1 64 64
572 x 572
570 x 570
568 x 568
128 128
284^2
282^2
280^2
256 256
140^2
138^2
136^2
512 512
68^2
66^2
64^2
1024
32^2
30^2
28^2
1024 512
56^2
54^2
52^2
512 256
104^2
102^2
100^2
256 128
200^2
198^2
196^2
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

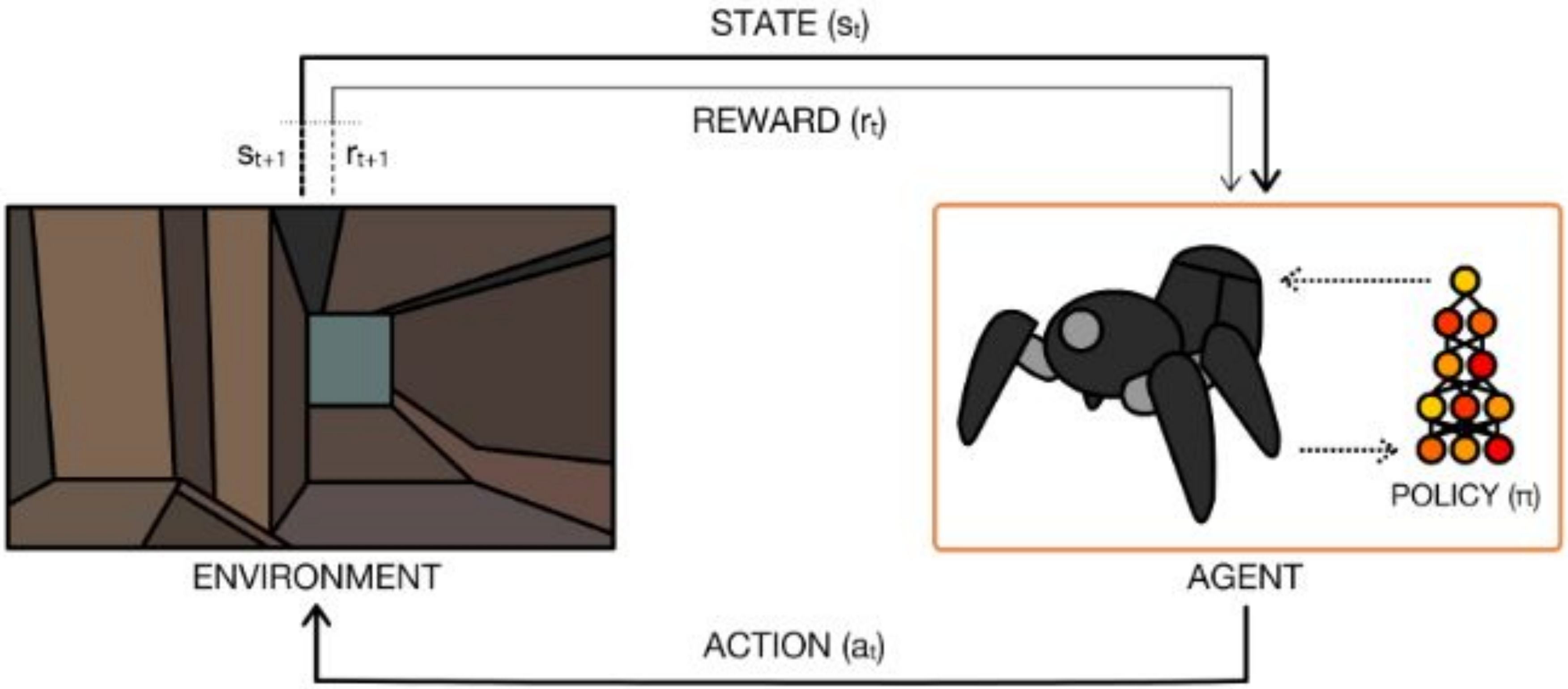
More Applications

- Text Classification [5]
  - Words are also spatially correlated!
- Music Recommendation [6]

Deep Reinforcement Learning

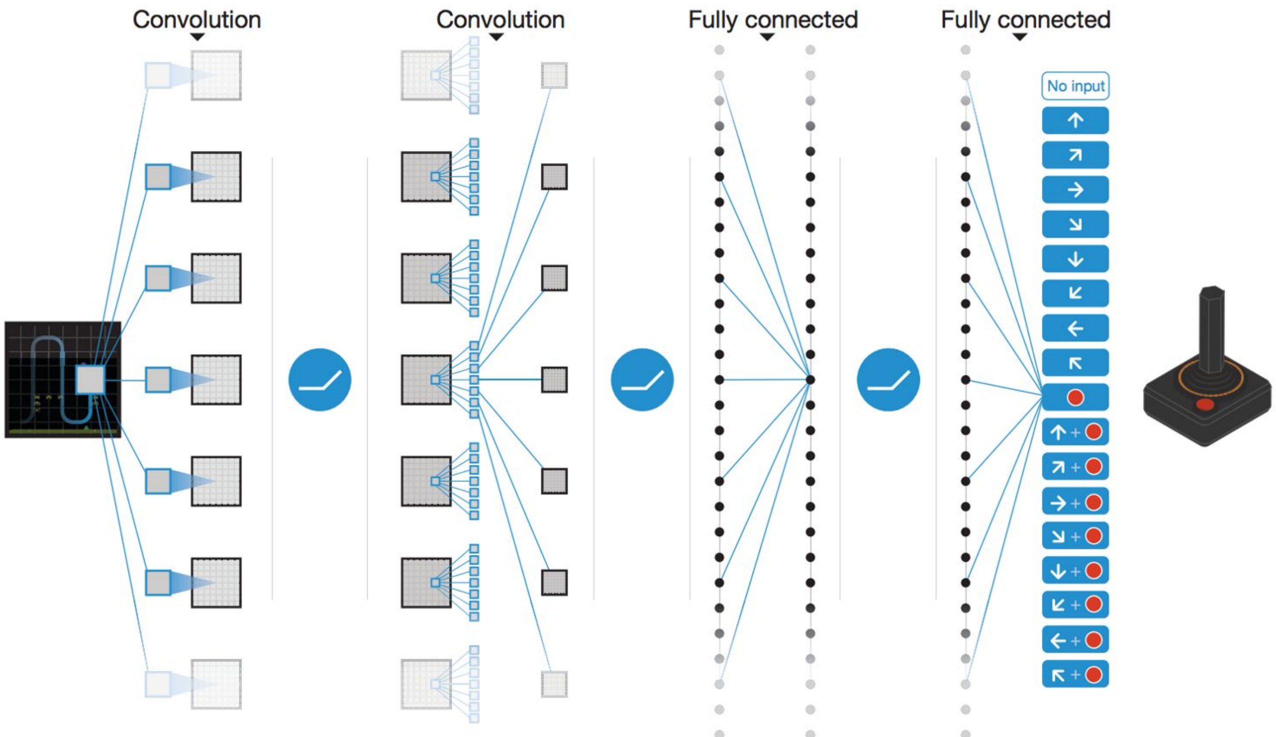
Decision Making in complex, unsearchable domains

Reinforcement Learning



- If we know the reward function, then it is easy!
- What if we don't?
- Idea: Learn the reward function using a deep neural network
  - Capable of inferring complicated reward structure

DQN (2015)



Deep Q-Learning for Arcade Games

AlphaGo Zero

- Policy Network
  - Where should I search?
- Value Network
  - What is the value of each state?
- Trained through self-play
  - Beat reigning Go champions after four days of training

Recurrent Neural Networks

Making sense of sequential data

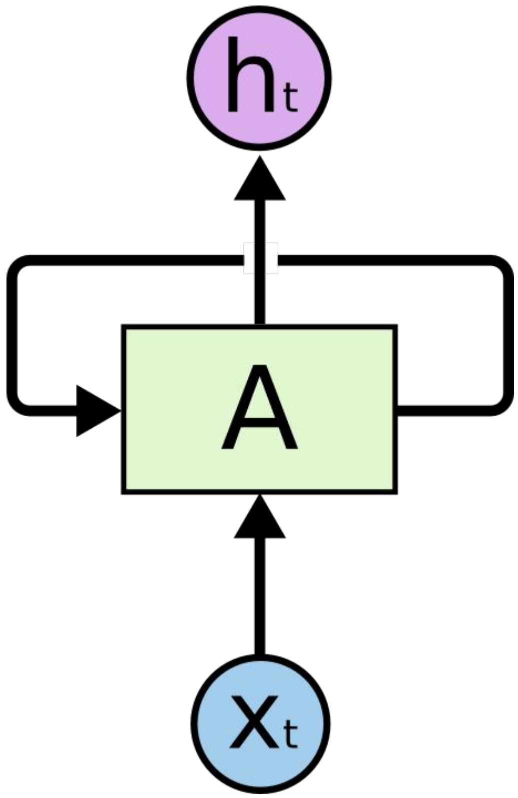
Recurrent Neural Networks

- For visual datasets: features are spatially correlated

- What if features are correlated over *time*?
  - Text Classification
  - Speech Recognition
  - Handwriting Recognition

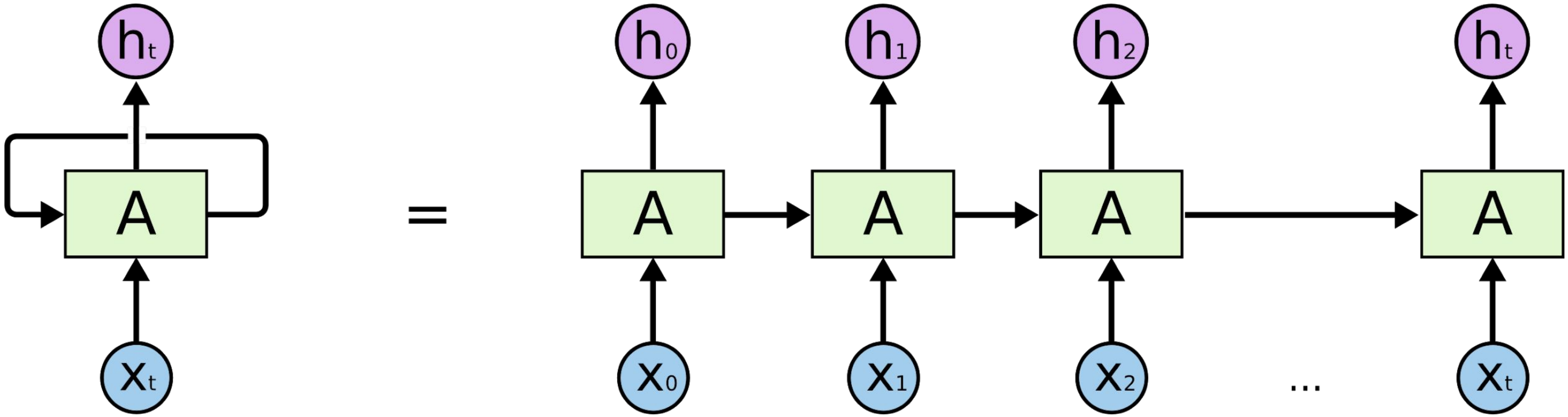
- Solution: *Recurrent Neural Networks*

Recurrent Neural Networks



Recurrent Neural Networks have *back-connections*

Recurrent Neural Networks



Recurrent Neural Network unrolled over time

“

Basic Recurrent Neural Nets work well for ***short term dependencies***

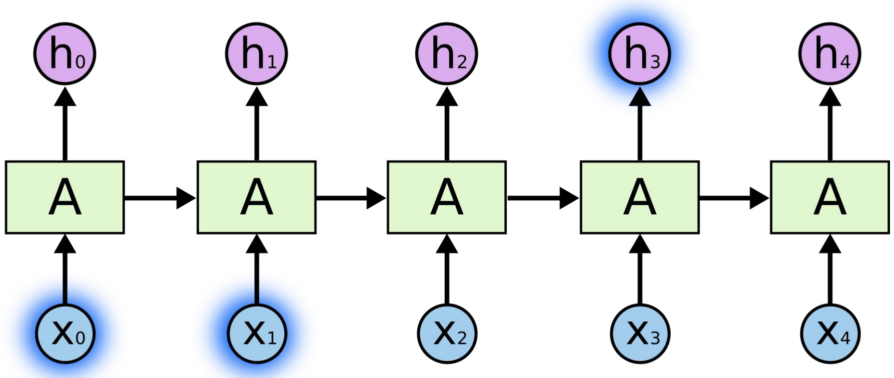


Image source:
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Basic Recurrent Neural Nets break down when data has **long term dependencies**

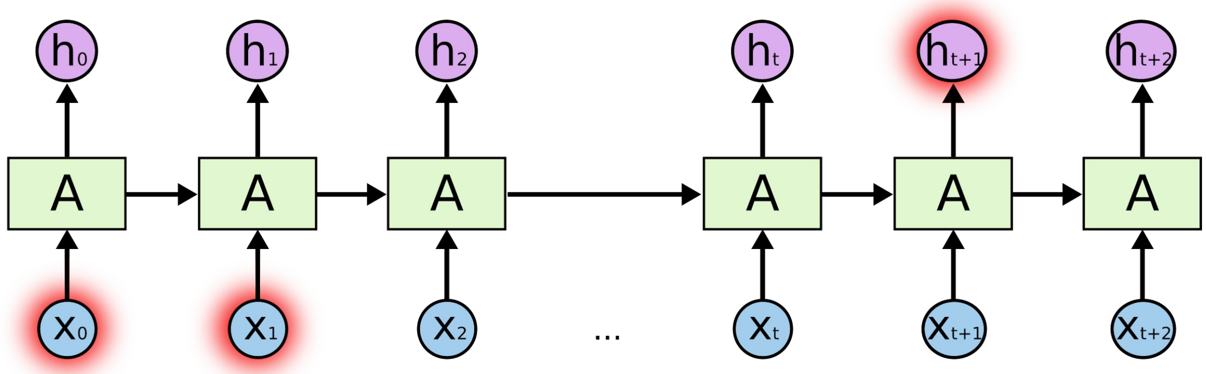


Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (LSTM)

- Solution: Long short-term memory cells

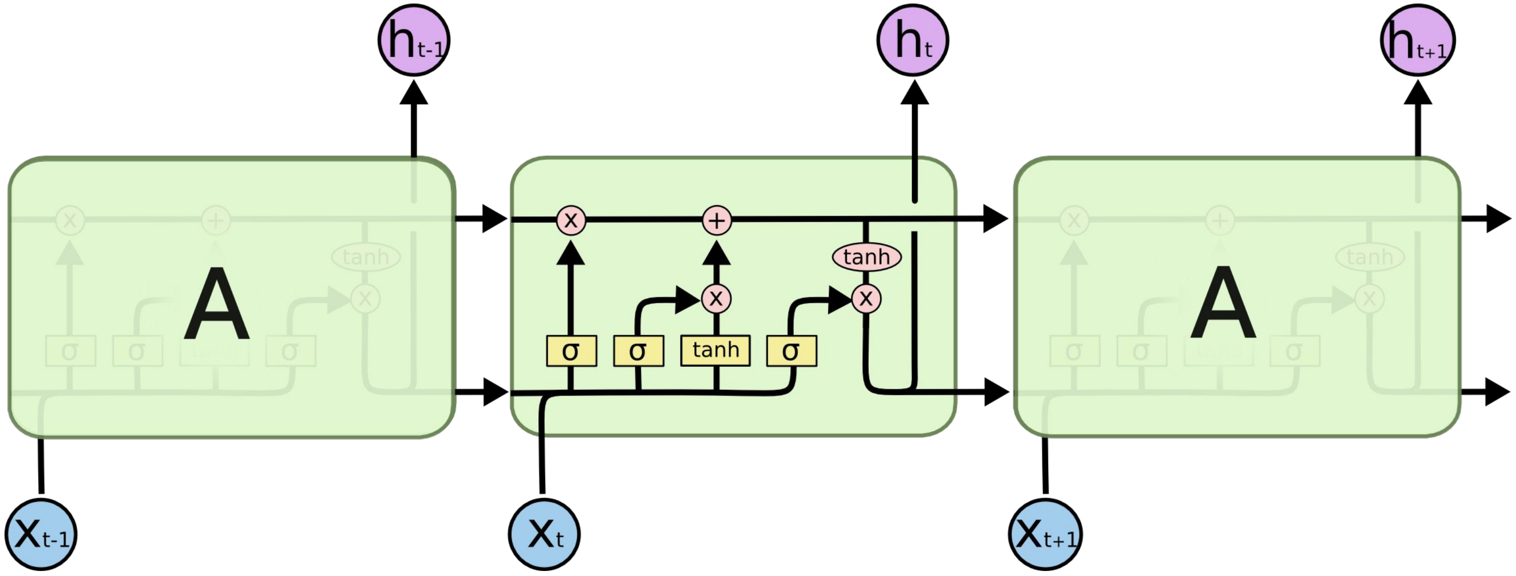


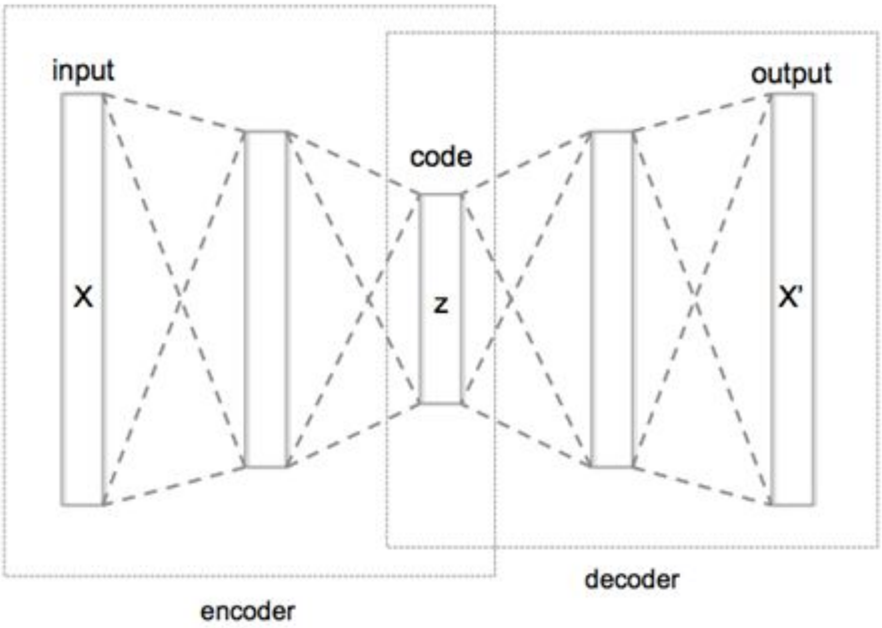
Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unsupervised Learning

Dimensionality Reduction, Generative Models, and Clustering

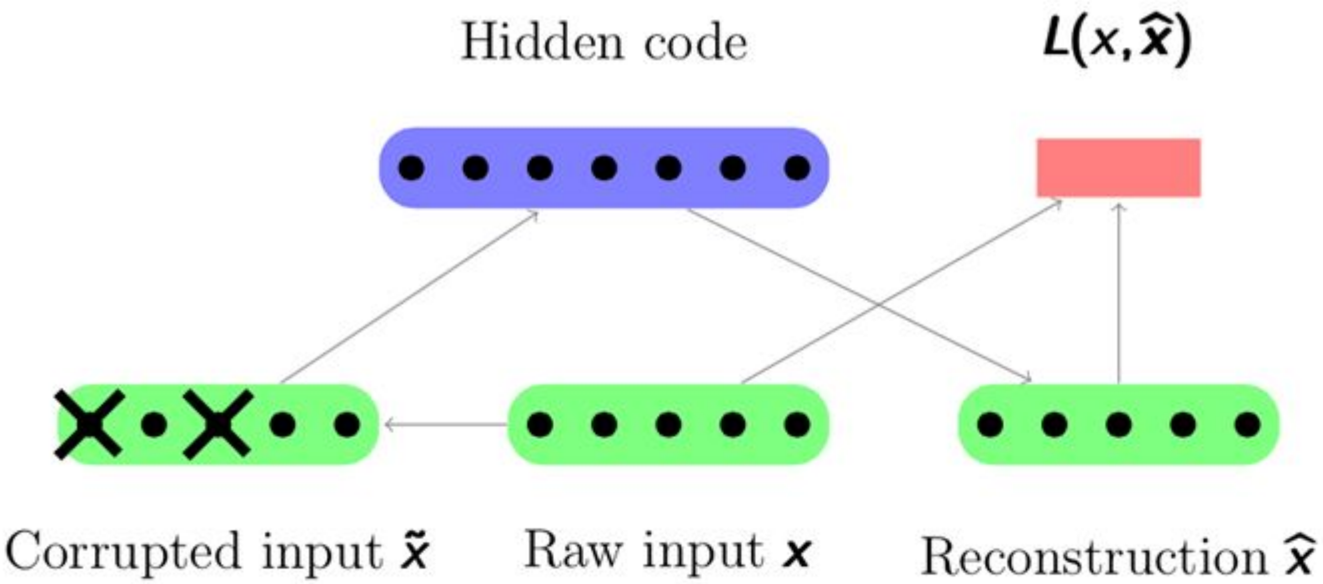
Unsupervised- Dimensionality reduction

- Autoencoders
  - Impose constraints on the code (eg, sparse)



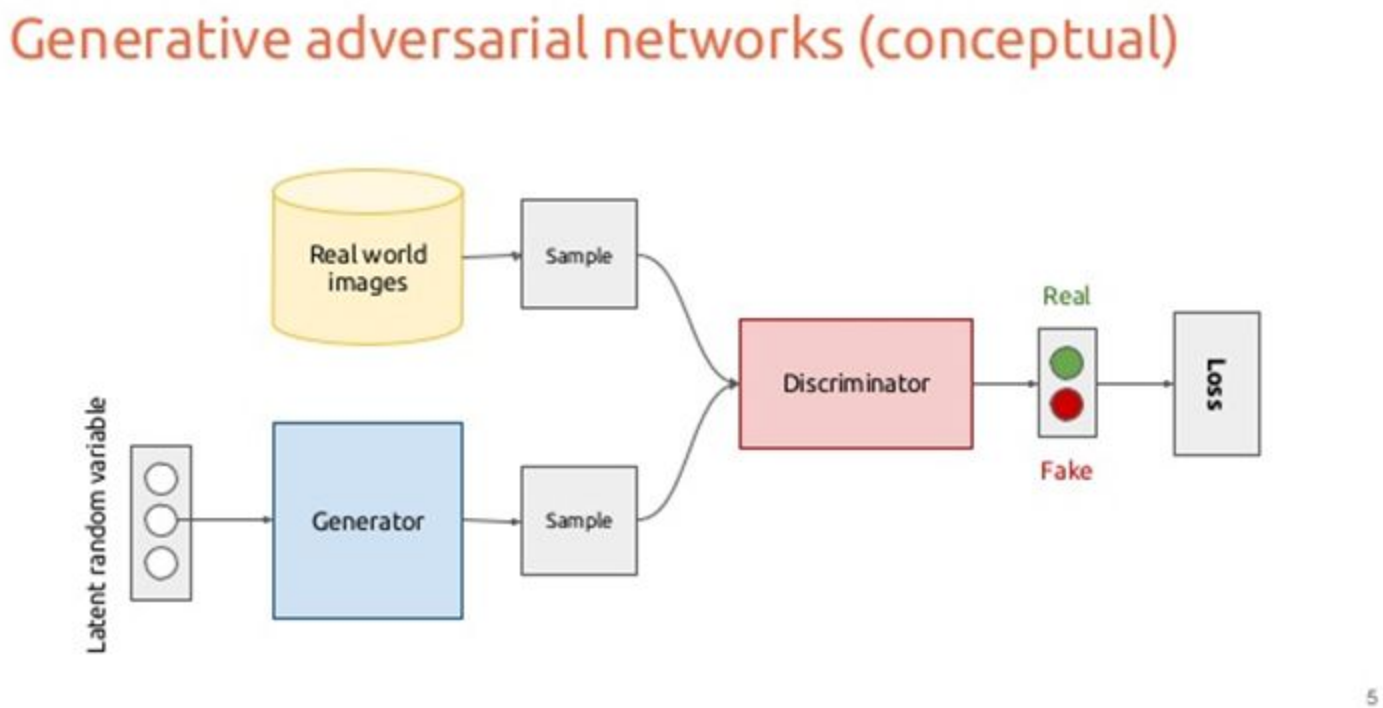
Unsupervised- Dimensionality reduction

- Denoising Autoencoders



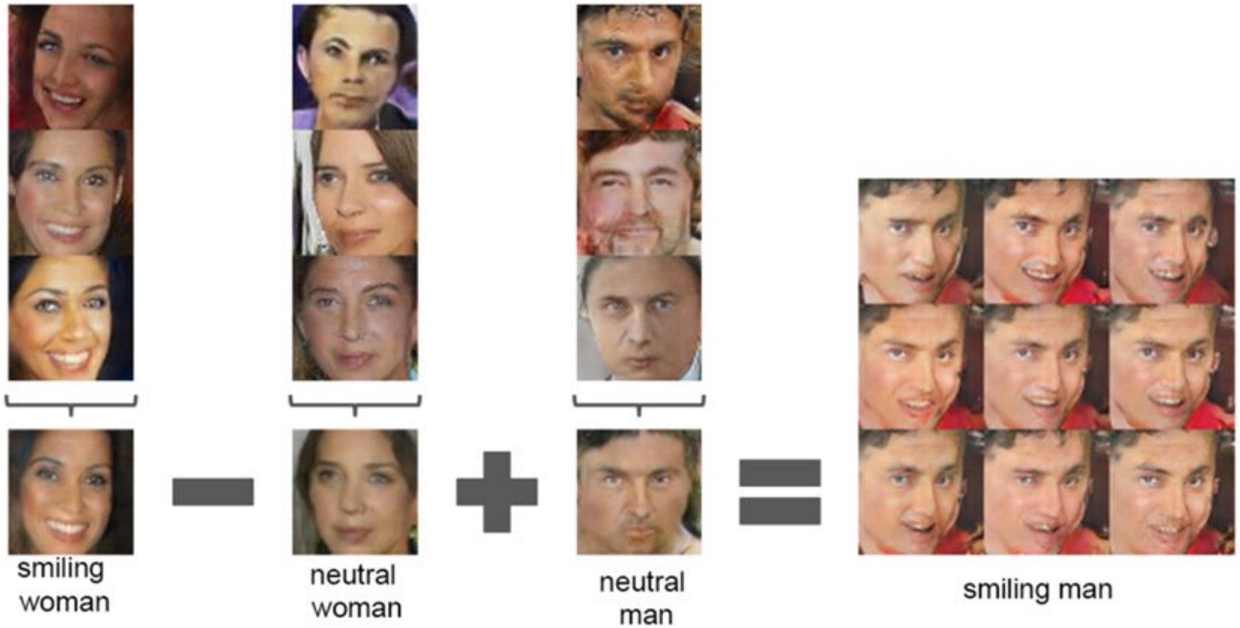
Unsupervised- Generative models

- Generative Adversarial Networks (2014)



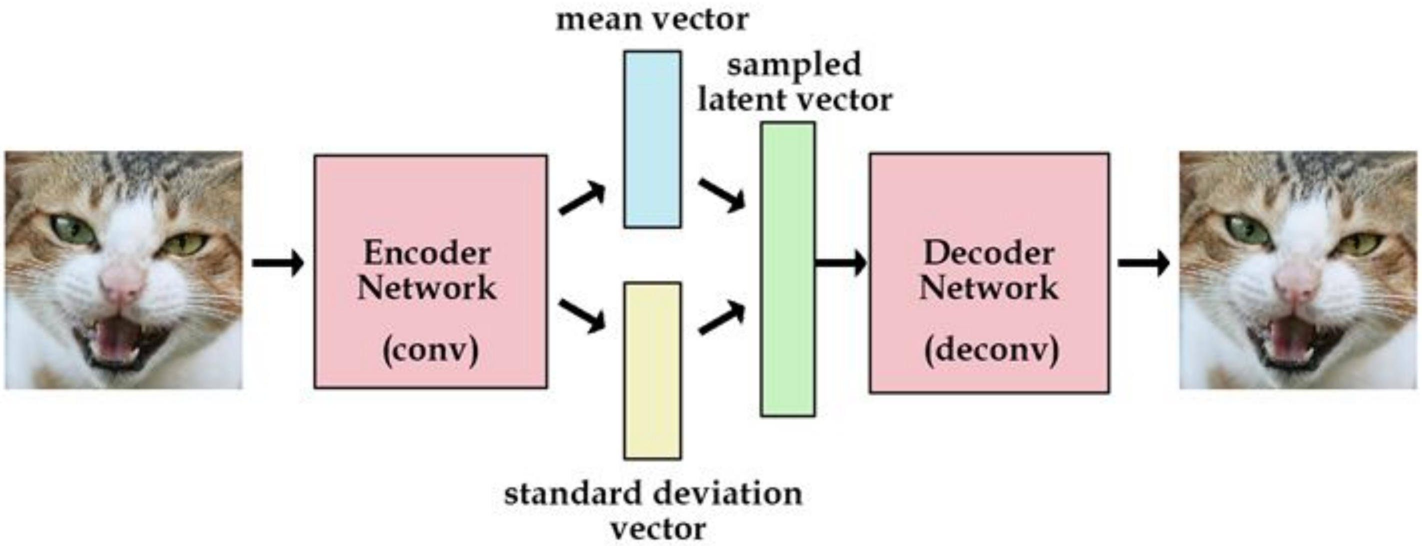
Unsupervised- Generative models

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015



Unsupervised- Generative models

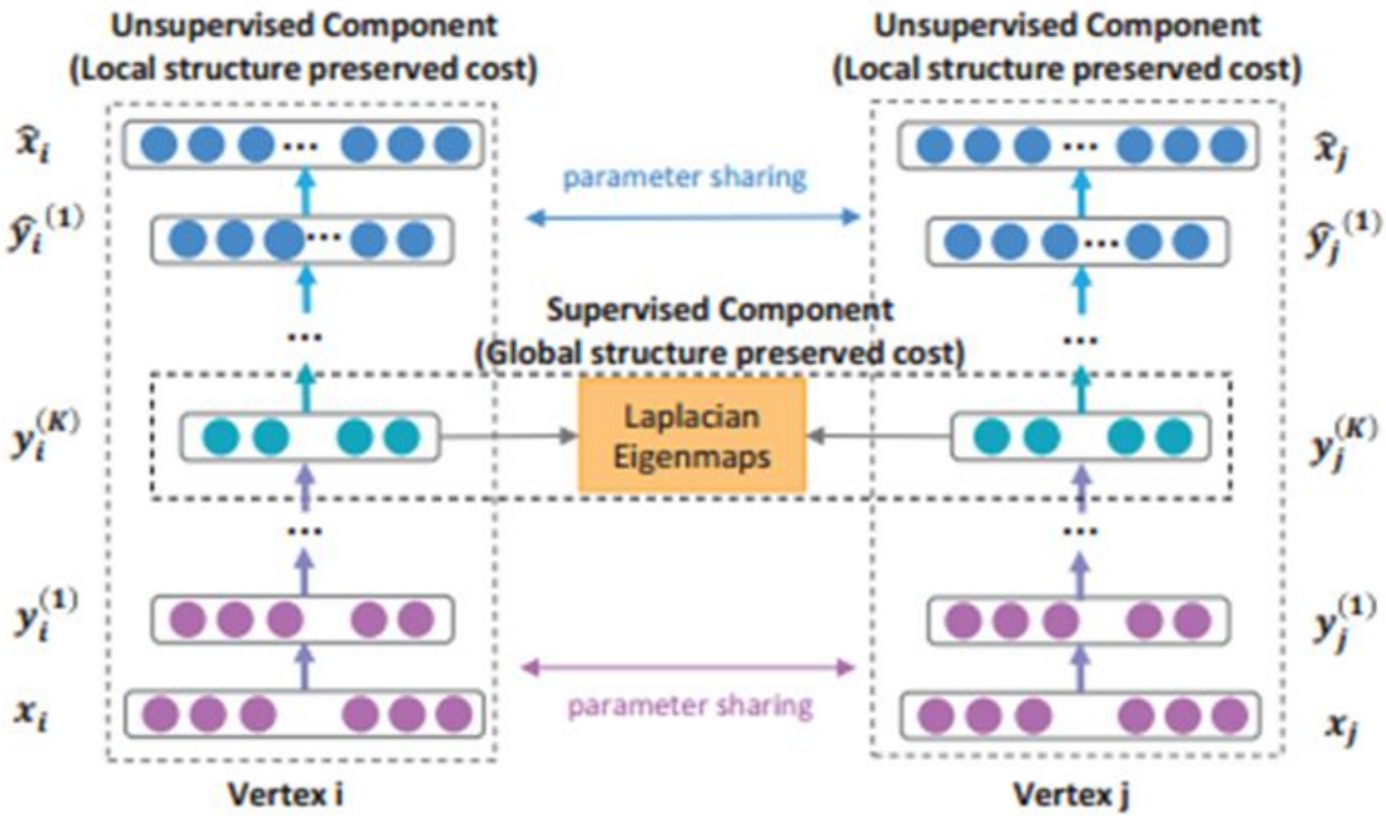
- Variational Auto Encoders (2014)
  - Concerned more about the distributions



Unsupervised- Clustering

- Spectral clustering:
    - Formulate pairwise similarity between datapoints (kernel matrix)
    - Eigendecompose the kernel matrix
    - Retain only the largest k-eigenvectors (Laplacian eigenmaps)
    - Apply k-means
- Eckart-Young-Mirsky theorem:
    - First k-eigenvectors of a matrix M reconstruct the optimal low-rank (k) version of M
- Autoencoders are all about reconstruction

Unsupervised- Clustering



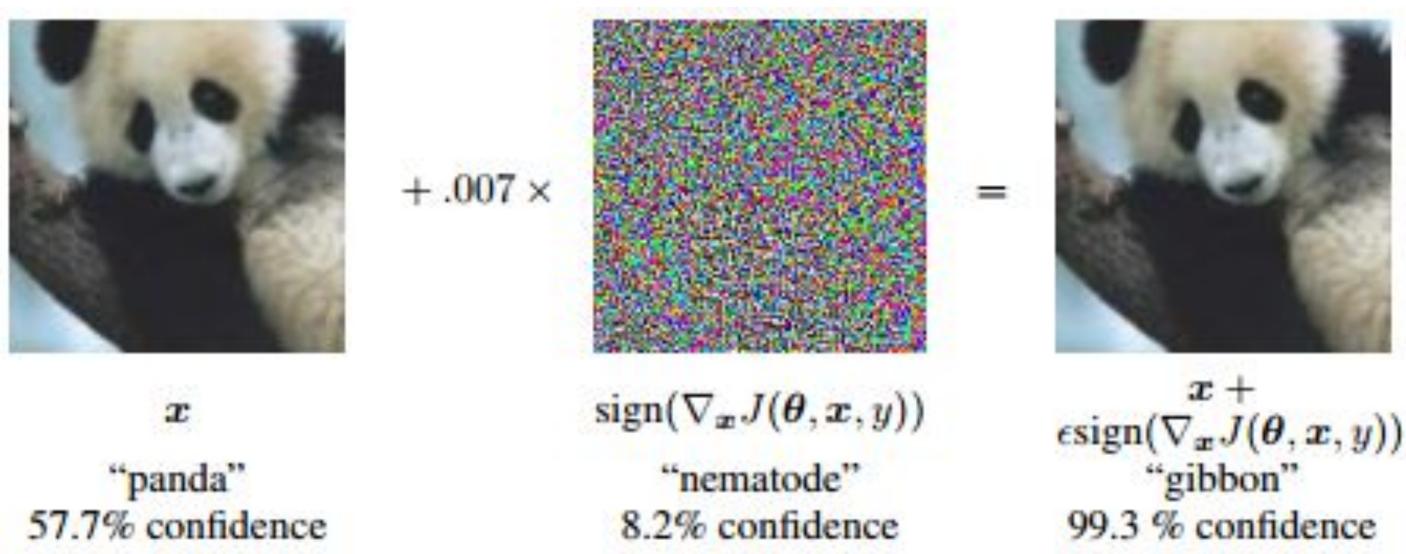
5.

Current Research

This could be you!

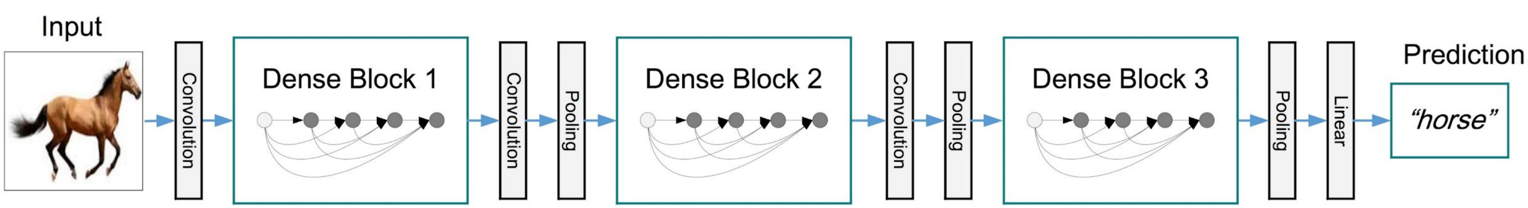
Adversarial Attacks

- CNN classifiers are easy to “trick”



Dense Nets

- Deep Neural Nets have *tons* of parameters
- Can we reduce the parameters without hurting accuracy?



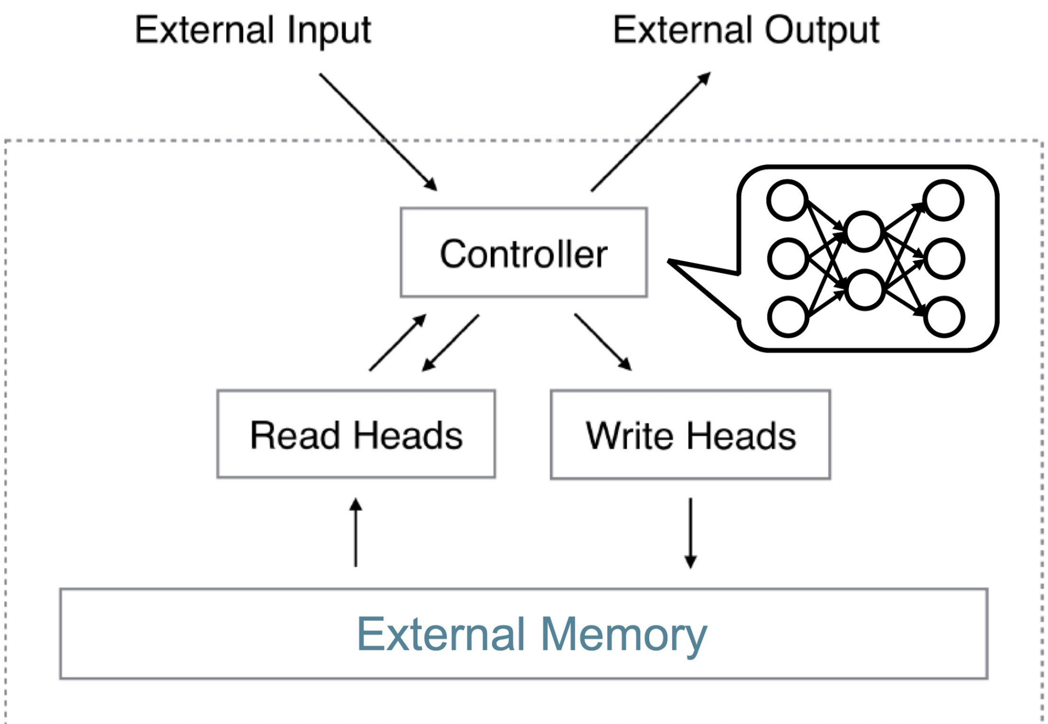
Distributed Learning

- Learning involves updating weights
- Can we avoid the expensive gradient broadcast every iteration?

Memory-Augmented Neural Nets

- Meta-learning
  - Can we learn to learn?
- Make use of long-term external memory
- One-shot Learning

Memory-Augmented Neural Nets



MANN structure

Thanks!

Any questions?

You can find us at:

zdj21157@uga.edu
smn57958@uga.edu

Credits

Papers referenced (in order of appearance):

1. LeNet (Yann LeCun)
2. AlexNet (Krishevsky et. al.)
3. Inception (Szegedy et. al.)
4. U-Net (Ronneberger et. al.)
5. CNNs for Sentence Classification (Yoon Kim)
6. Deep Content-Based Music Recommendation (van den Oord et. al.)
7. Playing Atari Games with DQN (Mnih et. al.)
8. AlphaGo Zero (Silver et. al.)

Credits

Materials used:

- Presentation template by SlidesCarnival
- Bahaa's Original Deep Learning Presentation
- Yoshua Bengio's Lecture on Deep Learning