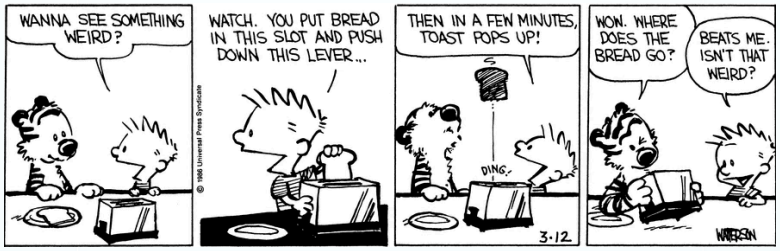


Deep Learning 2

It's like magic

Mark Dredze

Machine Learning
CS 601.475

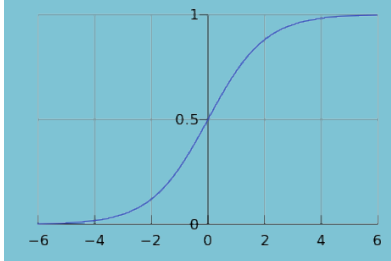
Outline

- Lecture 1: Neural Networks
  - Nonlinearities
  - Objective functions
  - Training
  - Gradient Computations
- Lecture 2: Deep Learning 1
  - Deep networks
  - Backpropogation
  - Training options
- Lecture 3: Deep Learning 2
  - Activation functions
  - Regularization
  - Dropout
  - Architecture examples

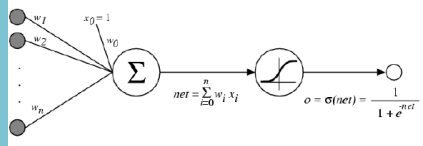
Activation Functions

Sigmoid / Logistic Function

$$\text{logistic}(u) \equiv \frac{1}{1+e^{-u}}$$



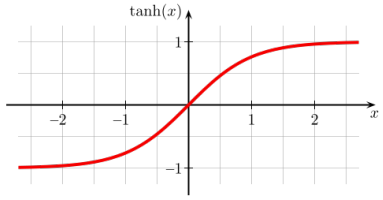
So far, we've assumed that the activation function (nonlinearity) is always the sigmoid function…



Slide from Matt Gormley

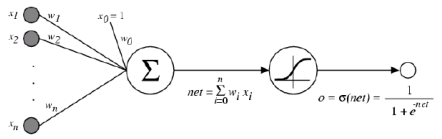
Activation Functions

- A new change: modifying the nonlinearity
  - The logistic is not widely used in modern ANNs



Alternate 1: tanh

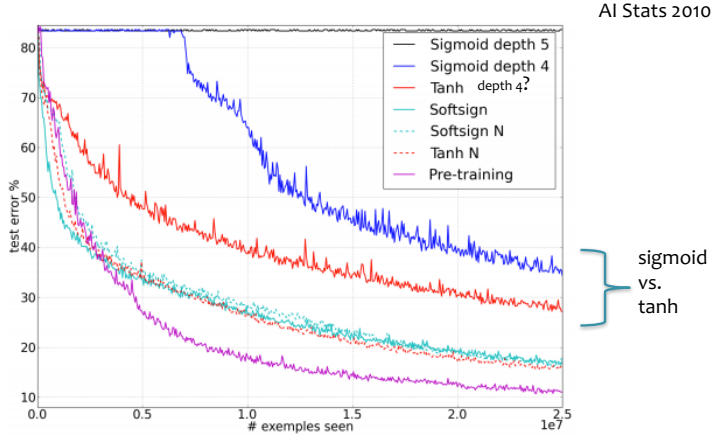
Like logistic function but shifted to range [-1, +1]



Slide from William Cohen

Understanding the difficulty of training deep feedforward neural networks

AI Stats 2010

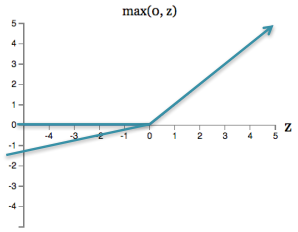


sigmoid vs. tanh

Figure from Glorot & Bengio (2010)

Activation Functions

- A new change: modifying the nonlinearity
  - reLU often used in vision tasks

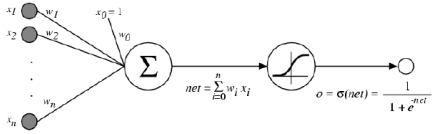


$$\max(0, w \cdot x + b).$$

Alternate 2: rectified linear unit

Linear with a cutoff at zero

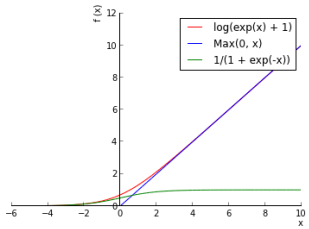
(Implementation: clip the gradient when you pass zero)



Slide from William Cohen

Activation Functions

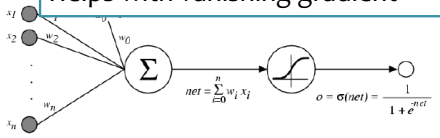
- A new change: modifying the nonlinearity
  - reLU often used in vision tasks



Alternate 2: rectified linear unit

Soft version: log(exp(x)+1)

Doesn't saturate (at one end)
Sparsifies outputs
Helps with vanishing gradient



Slide from William Cohen

Overfitting in Deep Networks

- Very large networks can easily overfit
- Large numbers of parameters
  - More and more with each layer of the network
- What strategies can we use to prevent over fitting?

Data

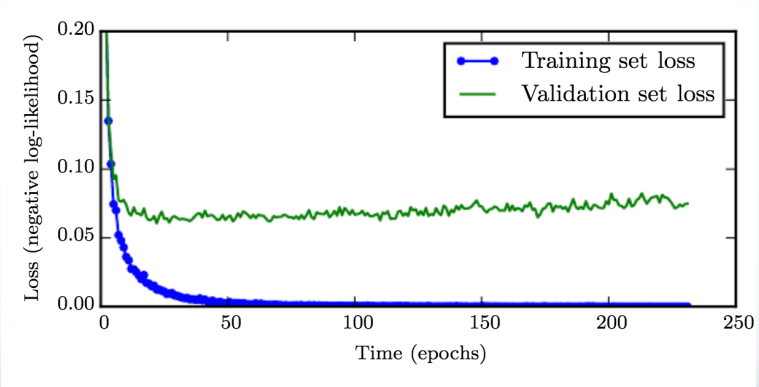
- The more data, the less likely you are to overfit
- A common strategy in the age of big data
- Often times this is the best solution, but won't help when more data isn't available

Regularization

- L1 or L2 regularization on network parameters
  - Straightforward based on previous uses of regularization
- Different parameters call for different regularization
  - We may have more complex network architectures
  - We could apply different types of regularization to different types of parameters

Early Stopping

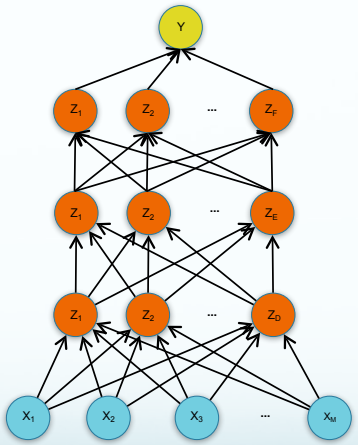
- Traditional training: run until convergence
- Early stopping
  - Measure performance on held out evaluation set
  - Stop when held out accuracy no longer improves
  - This is "early", i.e. before convergence
  - Measuring "no improvement" can be tricky

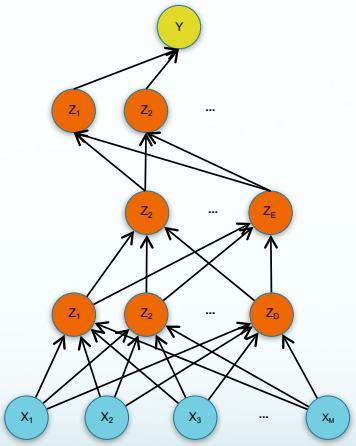


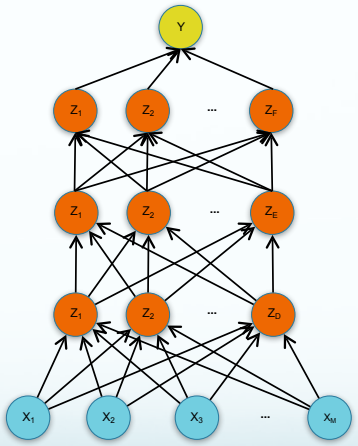
http://www.deeplearningbook.org/contents/regularization.html

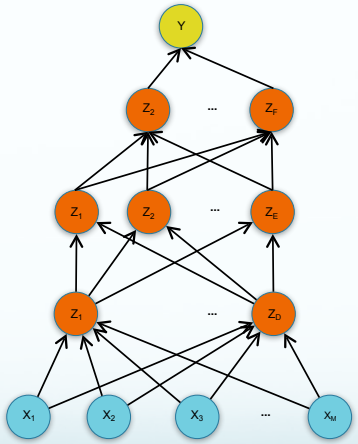
Dropout

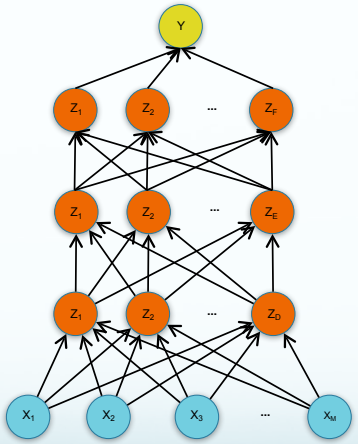
- Instead of modifying objective function (regularization), modify the **network structure**

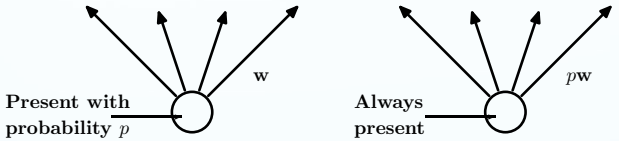












(a) At training time (b) At test time

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*15.1 (2014): 1929-1958.

Dropout

- Randomly "dropout" some neurons with probability p
- The chosen neurons change on each mini-batch
- All other learning remains unchanged

Why?

- It works!

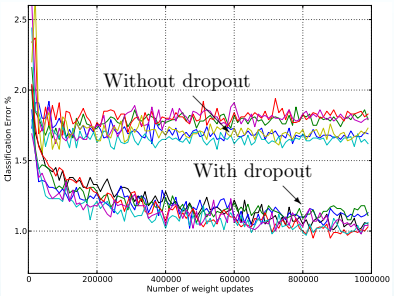


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*15.1 (2014): 1929-1958.

Why does it work?

- Noise robustness
  - Dropout is similar to making random perturbations in the input
  - These small changes shouldn't change the prediction for an input
  - Think "reduces variance"

Why does it work?

- Prevents neuron co-adaptation
  - Neurons, especially near the top of the network, suffer from co-adaptation
  - The neurons take on values that rely on other neurons for correct predictions
  - Multiple points of failure: any single neuron failing can invalidate the others
  - Dropout prevents co-adaptation by randomly removing some of these neurons
  - Network is forced to learn more robust features that are useful with many different random subsets of other neurons

Why does it work?

- Ensemble training
  - Recall from boosting: we can combine many classifiers to produce better output
  - During training, each random permutation is one of 2^n possible neural networks
  - All networks share weights so number of parameters is still $O(n^2)$
  - Dropout training is like training a collection of 2^n "thinned" networks with extensive weight sharing
  - Test time: approximate the average prediction of all networks by using a single network without dropout

Dropout: Advantages

- Works really well for a wide variety of network types and domains
- Very easy to implement with no modifications in training
  - Some minor details (omitted) are important
- Drawbacks
  - Need to select dropout rate
  - This depends on the types of parameters and network structure

Still Unanswered Questions

- Lots of motivations help us to understand why they work
- But they are mostly ad hoc explanations for an empirically useful training procedure

Insights into Dropout

- Dropout vs. regularization
  - Regularization penalty grows linearly in depth of network, dropout grows exponentially
    - Exponential penalty may be more suited to NNs
  - Dropout is insensitive to rescaling of input, or internal neurons

Helmbold, David P., and Philip M. Long. "Fundamental Differences Between Dropout and Weight Decay in Deep Networks." *arXiv preprint arXiv:1602.04484* (2016).

More data?

- 1) Pre-training on large amounts of data
- 2) Randomly permute input data to create a larger data set
- 3) Add additional objective functions to increase supervision

Network Architectures

- Multi-layer Perceptrons
  - Deep, fully connected neural networks
- Other common structures useful for various domains

Softmax

- Binary classification
  - Sigmoid function: probability distribution over two labels
- Multiclass classification
  - Softmax function: probability distribution over multiple discrete labels
    - Generalization of the sigmoid function

Softmax

$$\mathbf{z} = \mathbf{W}^T \mathbf{h} + \mathbf{b}$$

$$z_i = \log P(y = i|\mathbf{x})$$

$$\text{softmax}(\mathbf{z}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

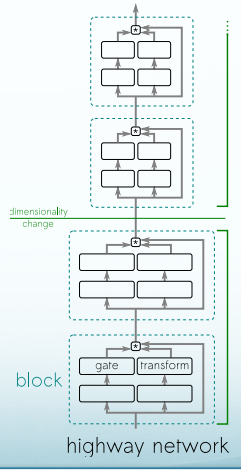
Negative Sampling

- Softmax with 10k outputs!
  - Requires a summation over 10k labels during training
    - Very slow
- Negative sampling
  - During training randomly select incorrect labels
  - Only update those labels
  - Over many updates will eventually update all labels

Very Deep Networks

- How deep can we make MLPs?
  - Hundreds of layers!
  - Poor propagation of activations and gradients
- Solution: add “highways”
  - Direct connections that skip across layers

Highway Network

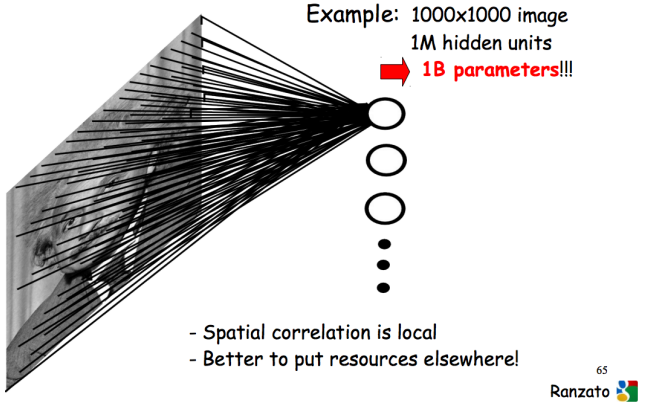


Greff, Klaus, Rupesh K. Srivastava, and Jürgen Schmidhuber. "Highway and residual networks learn unrolled iterative estimation." *arXiv preprint arXiv:1612.07771* (2016).

Convolutional Networks

- ConvNets are everywhere!
  - Image classification
  - Self driving cars
  - Learning word representations
  - Automated captioning

When the input data is an image..



Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

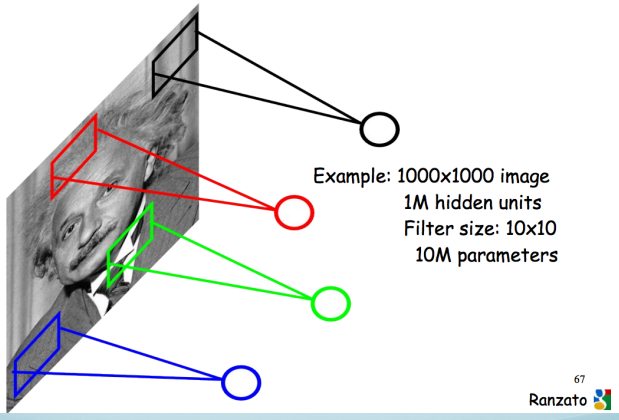
Insight

- Helpful features for images are based on local structures
- Where these occur in the image is not relevant
- Goal: learn representations of local structure
- Apply those representations throughout the image

Example: Face Recognition

- A face should have
  - 2 eyes
  - mouth
  - nose
- We don't care where these features appear in the image, only that they appear
- Idea: find each feature in the image, check that they are all present

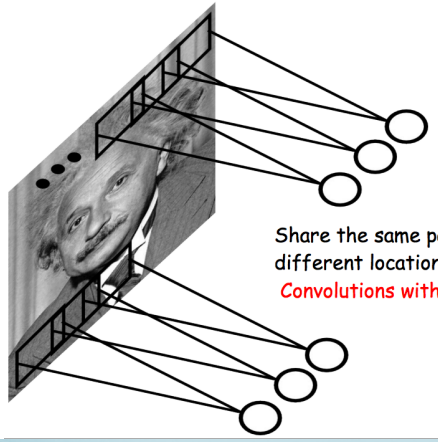
Reduce connection to local regions



67
Ranzato

Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

Reuse the same kernel everywhere



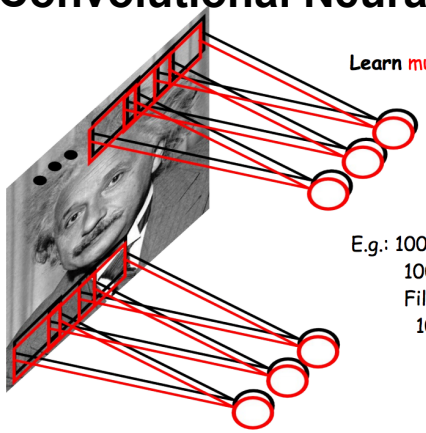
Because interesting features (edges) can happen at anywhere in the image.

Share the same parameters across different locations:
Convolutions with learned kernels

69
Ranzato

Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

Convolutional Neural Nets



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

70
Ranzato

LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

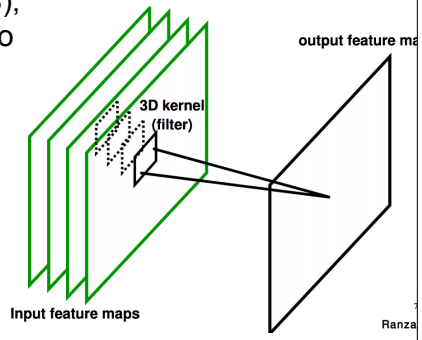
Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

Detail

If the input has 3 channels (R,G,B), 3 separate k by k filter is applied to each channel.

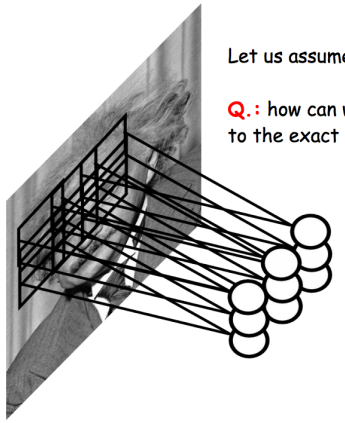
Output of convolving 1 feature is called a *feature map*.

This is just sliding window, ex. the output of one part filter of DPM is a feature map



Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

Building Translation Invariance



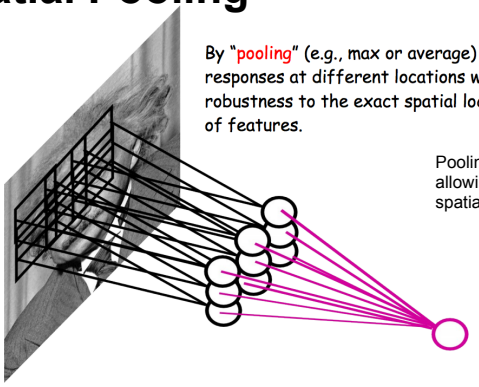
Let us assume filter is an "eye" detector.

Q.: how can we make the detection robust to the exact location of the eye?

72
Ranzato

Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

Building Translation Invariance via Spatial Pooling



By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

Pooling also subsamples the image, allowing the next layer to look at larger spatial regions.

73
Ranzato

Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

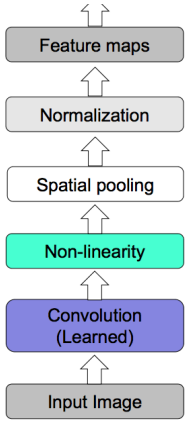
Summary of a typical convolutional layer

Doing all of this consists one layer.

- Pooling and normalization is optional.

Stack them up and train just like multi-layer neural nets.

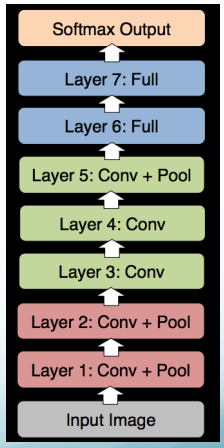
Final layer is usually fully connected neural net with output size == number of classes



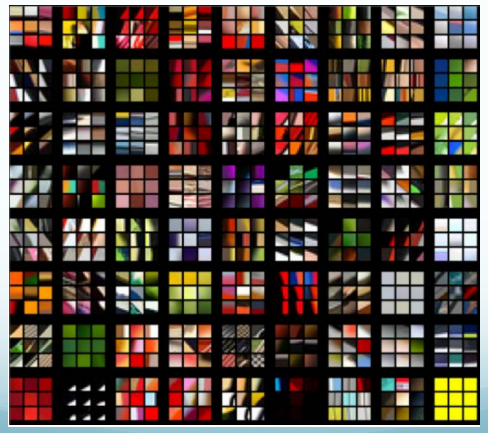
Slide from UMD CMSC 733: http://www.cs.umd.edu/~djacobs/CMSC733/CNN.pdf

AlexNet

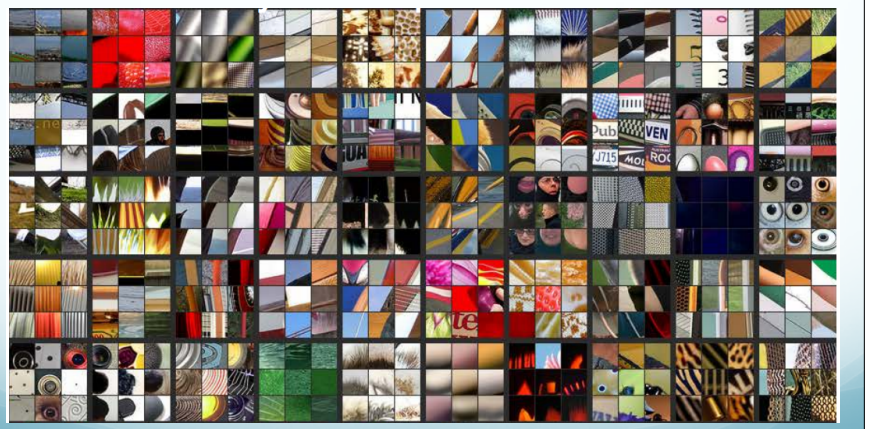
- Commonly used image processing neural network



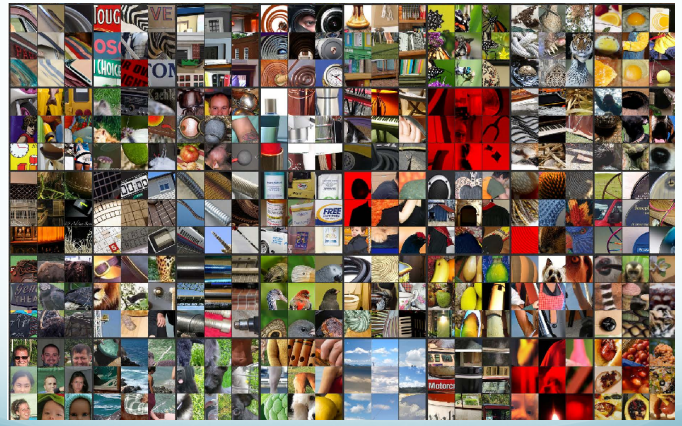
Layer 1



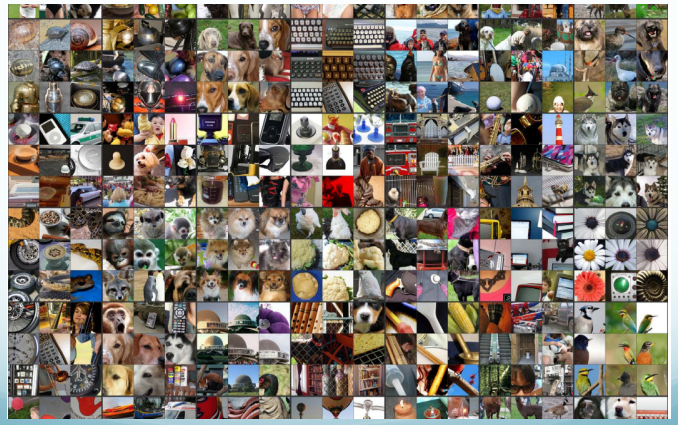
Layer 2



Layer 3



Layer 5



Network Outputs

- Most examples have focused on classification or regression
- Neural networks can output much more than that!
- Numerous outputs can be used to represent complex objects

Image Generation

- Take an image and a painting, render the image in the style of the painting

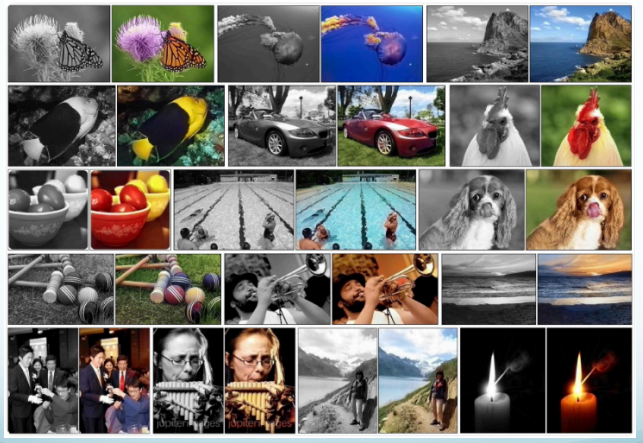
https://deepdreamgenerator.com/

GANs: Image Generation from Captions

Figure 3. Example results by our proposed StackGAN, GAWWN [20], and GAN-INT-CLS [22] conditioned on text descriptions from CUB test set. GAWWN and GAN-INT-CLS generate 16 images for each text description, respectively. We select the best one for each of them to compare with our StackGAN.

https://medium.com/@Moscow25/gans-will-change-the-world-7ed6ae8515ca

Image Colorization



https://machinelearningmastery.com/inspirational-applications-deep-learning/

Caption Generation

More great example applications:
https://machinelearningmastery.com/inspirational-applications-deep-learning/

Other Network Architectures

- Deep belief networks
- Generative Models
- Boltzmann machines
- Recurrent Neural Networks
  - We'll see these later in the semester

Summary

- Deep networks
  - Strong foundations based on traditional neural networks
    - Learned representations using backpropogation
  - Many avenues to obtain deep networks
    - Strong empirical evidence that deep networks perform well on many tasks
  - Active research exploring: architectures, training, regularization