Today: Outline

- **Training Strategies**

- Announcements:

  Pre-lecture Material, due: Oct 1

  Problem Set 1, due: Oct 12 by midnight

Neural Networks IV

Pre-lecture Material

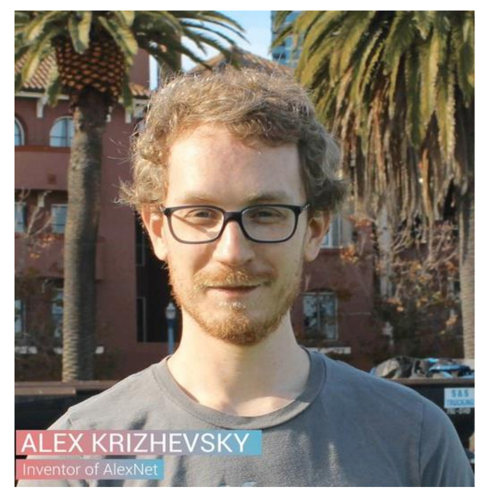
Alex Krizhevsky

Alex Krizhevsky
Dessa
Verified email at dessa.com
Machine Learning

FOLLOW

TITLE	CITED BY	YEAR
Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton Advances in neural information processing systems, 1097-1105	70500	2012
Dropout: a simple way to prevent neural networks from overfitting N Srivastava, G Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov The journal of machine learning research 15 (1), 1929-1958	22808	2014

Hence the name ***AlexNet***



ACM Turing Award (2019)

- Three ‘Godfathers of Deep Learning’ Selected for Turing Award
- ***Geoff Hinton***, an emeritus professor at the University of Toronto and a senior researcher at Alphabet Inc.’s Google Brain
- ***Yann LeCun***, a professor at New York University and the chief AI scientist at Facebook Inc.
- ***Yoshua Bengio***, a professor at the University of Montreal as well as co-founder of AI company Element AI Inc.

Geoffrey E Hinton

Yann LeCun

Yoshua Bengio

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Pre-lecture Material Quiz

AlexNet

Which of the following is a regularization technique that can be implemented in deep neural networks?

- [ ] Minimizing sum of the weight parameters
- [ ] Dropout
- [ ] Data Augmentation
- [ ] Multi-GPU Computation

Neural Networks IV

Training Strategies

Architecture Design and Training Issues

- How many layers? How many hidden units per layer? How to connect layers together? How to optimize?
  - Cost functions
  - *L2/L1 regularization*
  - *Data Set Augmentation*
  - *Early Stopping*
  - *Dropout*
  - Minibatch Training
  - Momentum
  - Initialization
  - Batch Normalization
  - Activation Functions
  - Architectures

Avoid Overfitting

Neural Networks IV

Cost Functions and Regularization

Cost Functions

- For regression problems, quadratic error is typical
- For classification, quadratic loss is not as effective
  - Instead one typically uses softmax outputs with cross-entropy error function
  - Discussed earlier, won't review in depth

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Regularization

- In machine learning, we care about *generalization performance*, not just training error
- With many parameters, models are prone to *overfitting*
- How to regularize?
  - Restrictions on parameter values or function classes
  - Adding terms to the objective function
  - Examples: L2 or L1 regularization

Neural Networks IV

Data Augmentation

Data Augmentation

- Another technique that prevents overfitting.

- How?
  By artificially enlarging the dataset using label-preserving transformations.

- Examples:
  - generating image translations and horizontal reflections
  - altering the intensities of the RGB channels in training images: add perturbations to each RGB image pixel $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]$

[Krizhevsky *et al.*]

Data Augmentation

- Could be computed “on the fly,” and do not necessarily need to be stored on disk.

- How?
  The transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images.

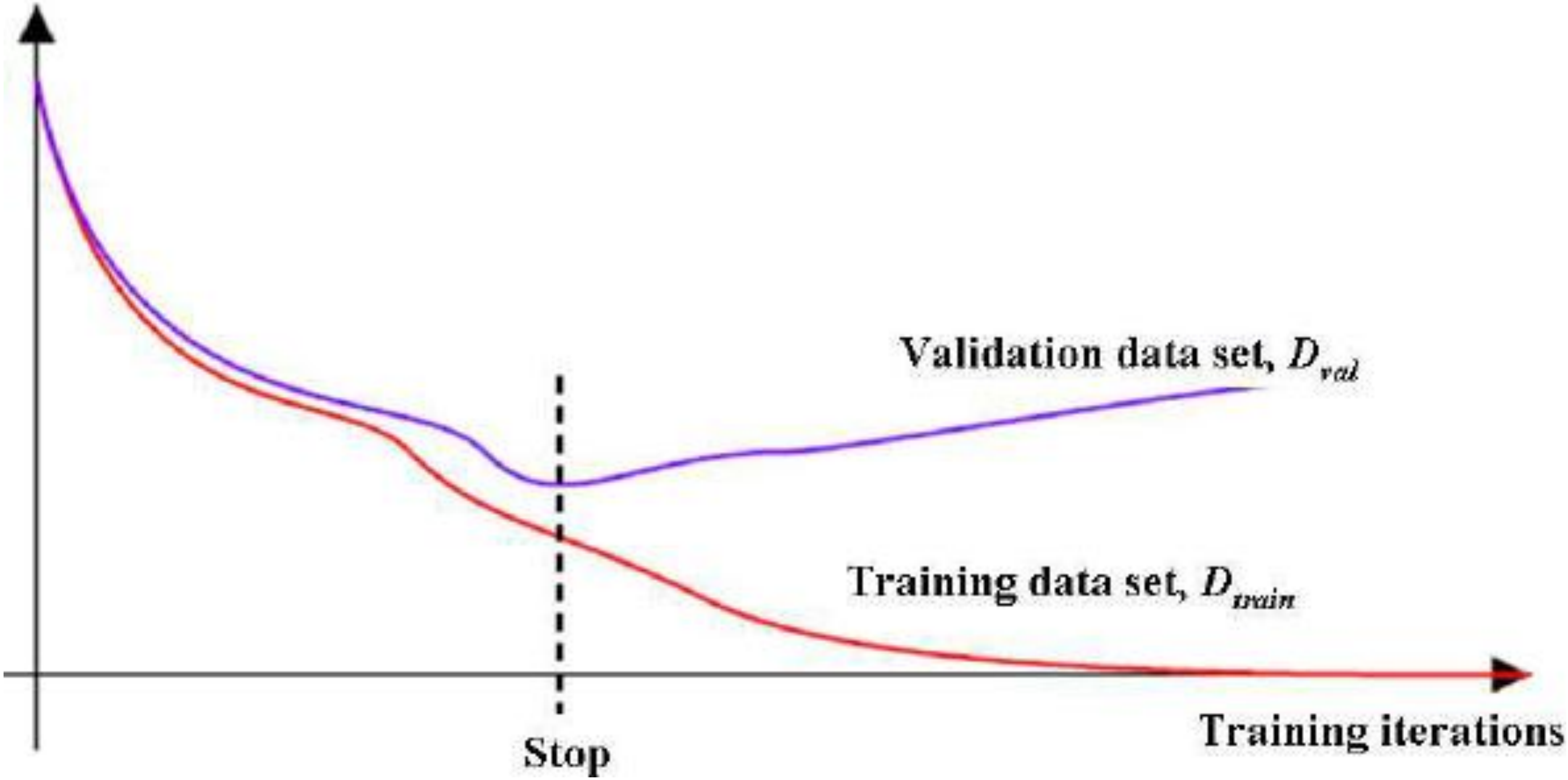
- So these data augmentation schemes can be, in effect, computationally free.

[Krizhevsky *et al.*]

Neural Networks IV

Early Stopping

Early Stopping



Neural Networks IV

Dropout

Dropout: A Classical Regularization Technique

- Combining the predictions of many different models is a very successful way to reduce test errors.

- But it appears to be too expensive for big neural networks that already take several days to train.

- There is, however, a very efficient version of model combination that only costs about a factor of two during training: ***Dropout***

[Krizhevsky *et al.*]

Dropout: A Classical Regularization Technique

- Setting to zero the output of each hidden neuron with a specific dropout probability, *e.g.* 0.5.

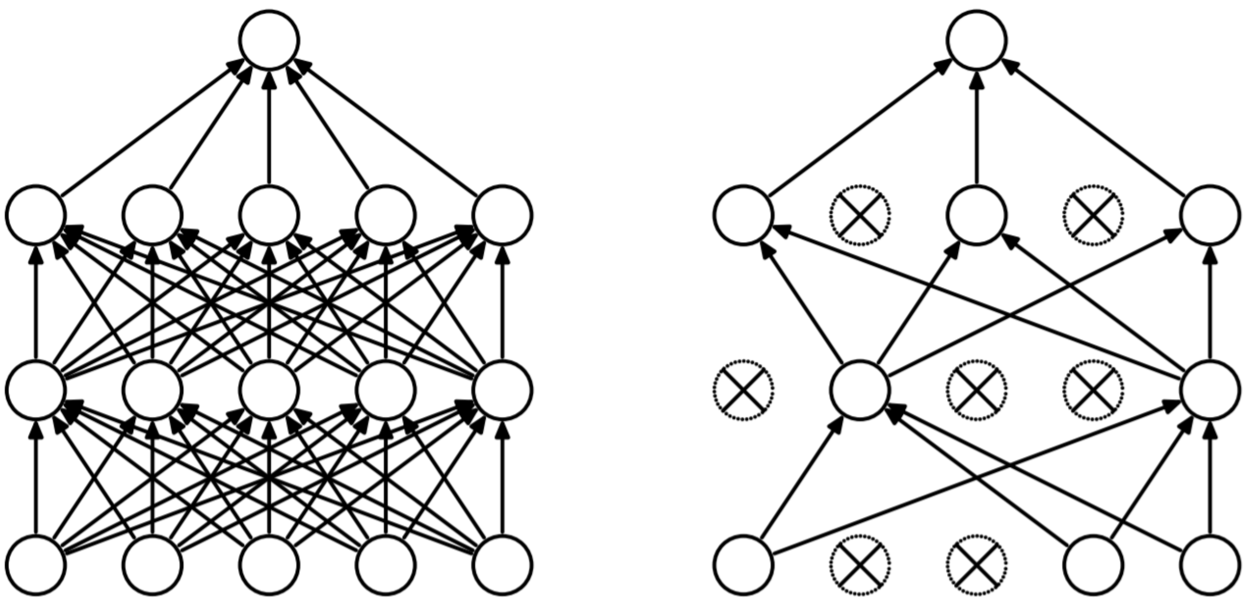
- The neurons which are "dropped out" in this way
  - do not contribute to the forward pass, and
  - do not participate in backpropagation.

- So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights.

[Krizhevsky *et al.*]

Dropout: A Classical Regularization Technique

- Many Deep Models employ dropout at training time to avoid overfitting, allowing for better generalization.



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

[Srivastava *et al.*]

Dropout: A Classical Regularization Technique

- Dropout can be thought of as a model averaging technique.
- Dropout can be applied to fully-connected layers or convolutional layers.
- It has so far been observed to give higher performance gains when applied to fully-connected layers.

Dropout Variants

- Several variants of dropout have been introduced:
  - How much dropout is applied to neurons/weights?
    - Information Dropout
    - DropConnect
    - Curriculum Dropout
  - Which neurons to drop out?
    - Adaptive Dropout
    - DropBlock
    - Excitation Dropout

Dropout

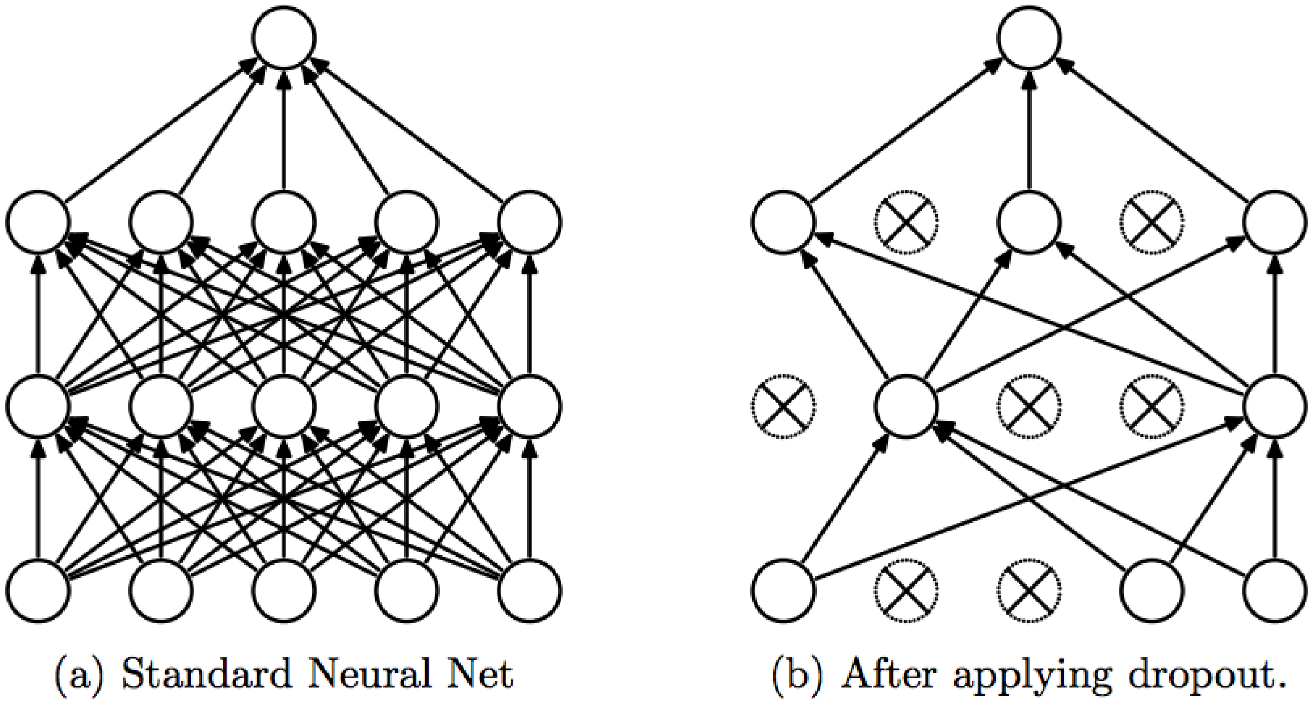


Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Dropout

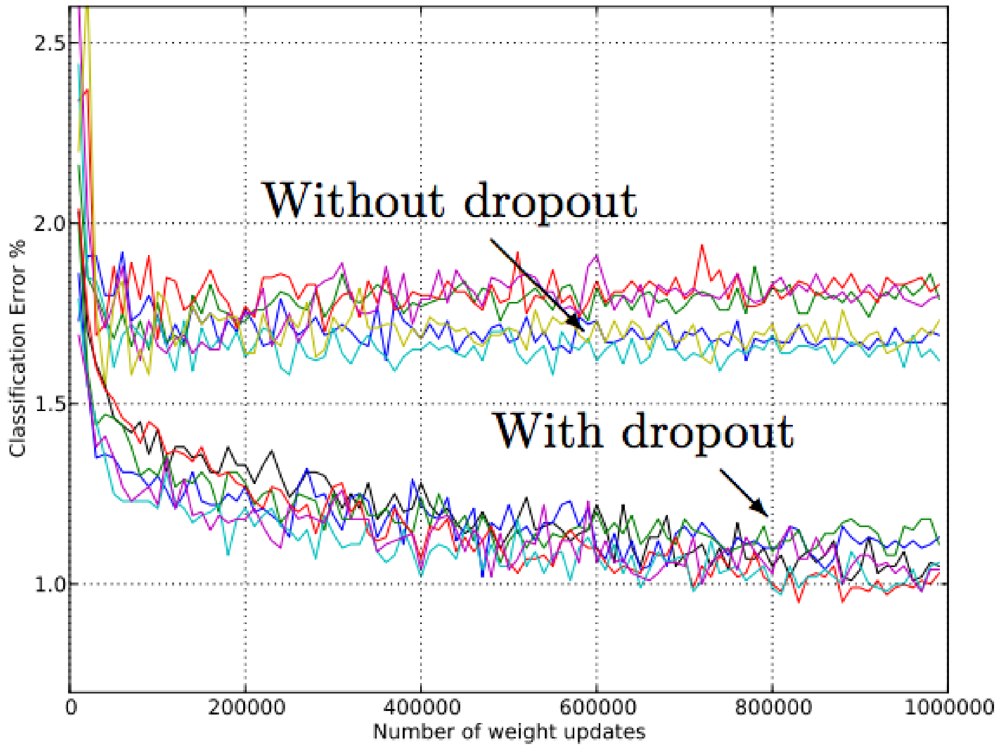


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Architecture Design and Training Issues

- How many layers? How many hidden units per layer? How to connect layers together? How to optimize?
  - Cost functions
  - *L2/L1 regularization*
  - *Data Set Augmentation*
  - *Early Stopping*
  - *Dropout*
  - Minibatch Training
  - Momentum
  - Initialization
  - Batch Normalization
  - Activation Functions
  - Architectures

Avoid Overfitting

Neural Networks IV

Minibatch Training

Mini-batches

- Gradients could be updated using:
  - One data point (too inaccurate)
  - All data points (too expensive)
  - Mini-batch (a good trade-off)

- The size of the mini-batch depends on:
  - How good of an approximation you need
  - How much GPU memory you have per GPU
  - How many GPUs you have

- GPUs can compute gradients of mini-batches in parallel, *i.e. Training on multiple GPUs*: Divide and Conquer.

GPUs

- NVIDIA TITAN V GPU

Minibatch Training

- Larger batches provide a more accurate estimate of the gradient. If all examples in the batch are processed in parallel, amount of memory scales with batch size

- Small batches can offer a regularizing effect due to the noise added during the learning process

- When using GPUs it is common for power of 2 batch sizes to offer better runtime; typical sizes range from 32 to 256, with 16 being common for large models

- Minibatches should be selected randomly!

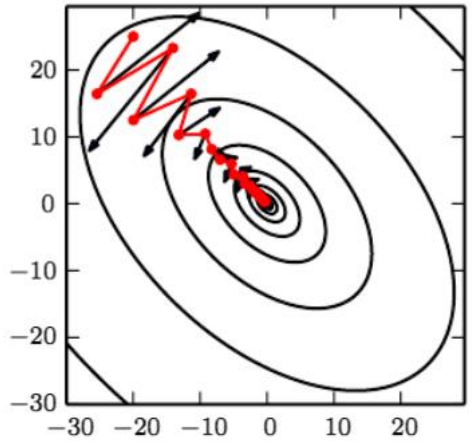
Neural Networks IV

Momentum, Batch Normalization, and Data Independence

Momentum

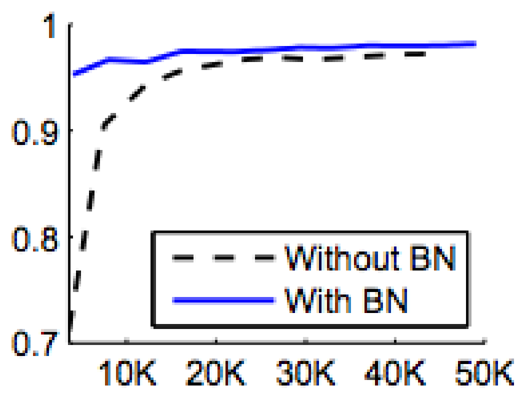
- Accumulates an exponentially decaying moving average of past gradients and continues to move in their direction

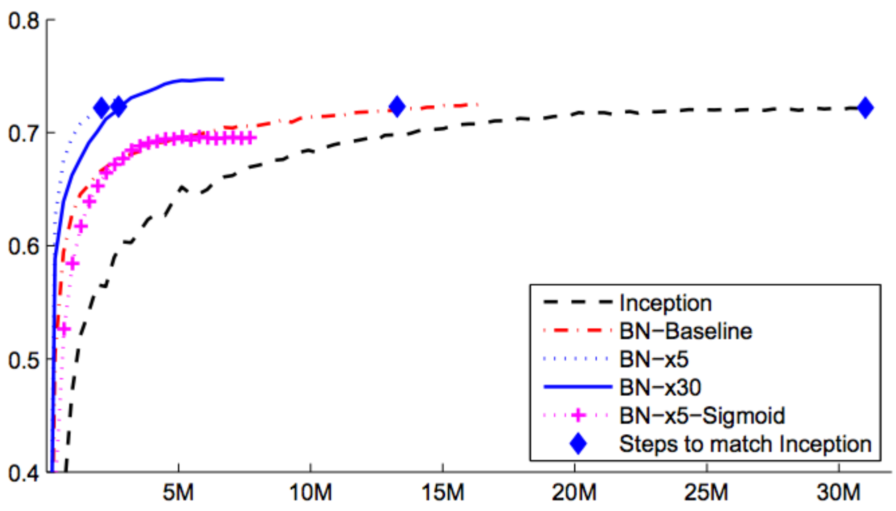
- exponentially weighed averages can provide us a better estimate which is closer to the actual derivate than our noisy batch calculations



Batch Normalization

- High-level idea: whitening the data at each layer makes training faster
- Left: MNIST, Right: ImageNet





Data Independence

- NN models converging to the correct solution depends on the iid assumption
- i.e. that our data are independent and identically distributed
- Important to randomly shuffle examples! Otherwise net can fail to converge

Neural Networks IV

Activation Functions and Architectures

Activation Functions

- ReLU: $g(x) = \max(0, x)$
- Leaky ReLU: $g(x) = \max(0, x) + \alpha \min(0, x) \quad (\alpha \approx .01)$
- Tanh: $g(x) = 2\sigma(2x) - 1$
- Radial Basis Functions: $g(x) = \exp(-(w - x)^2/\sigma^2)$
- Softplus: $g(x) = \log(1 + e^x)$
- Hard Tanh: $g(x) = \max(-1, \min(1, x))$
- Maxout: $g(x) = \max_{j \in \mathbb{G}} x_j$
- ….

Architectures

- Some commonly referred to architectures:
  - AlexNet
  - VGG16/19
  - GoogleNet
  - ResNet
  - WideResNet
  - Inception
  - …