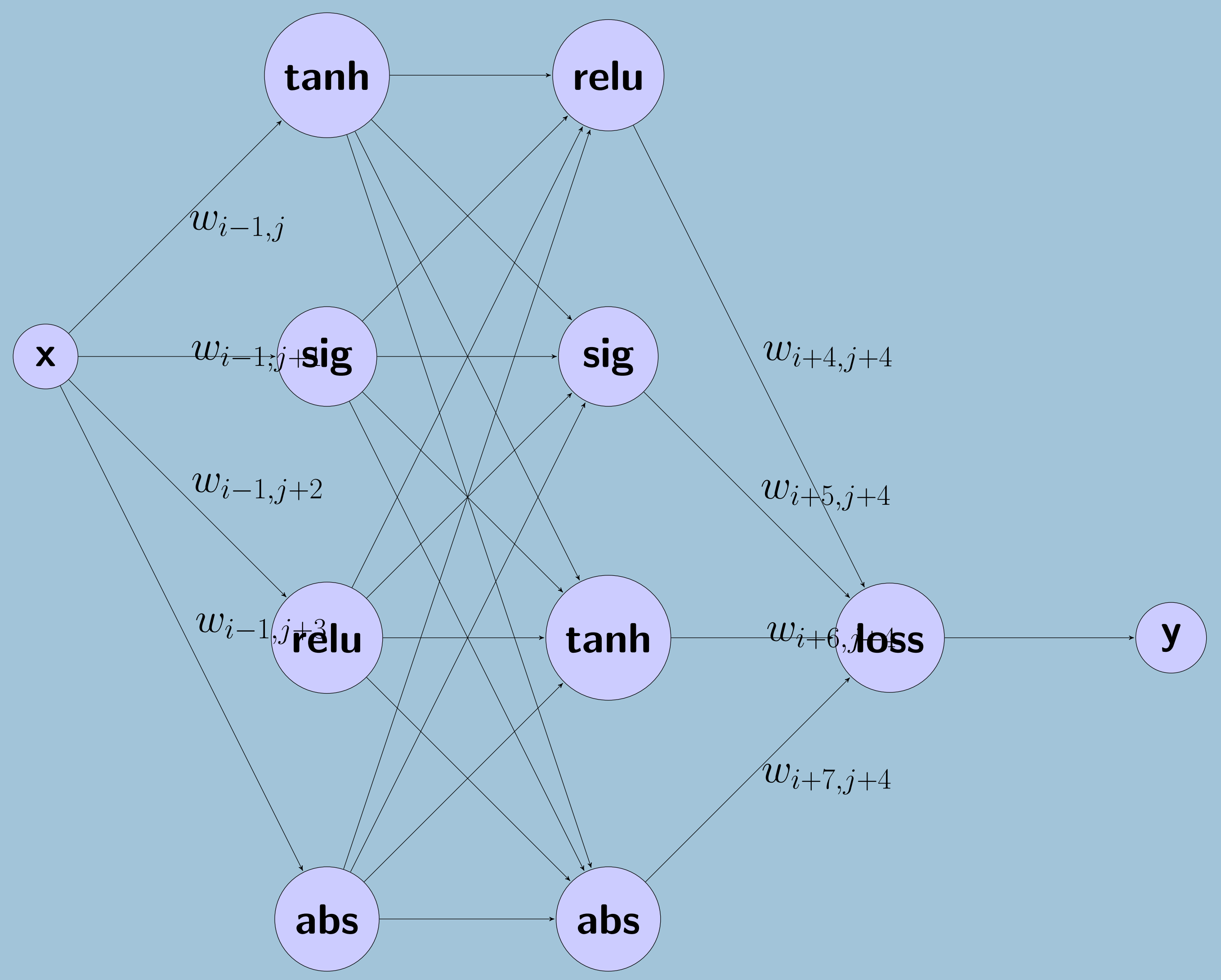
Scene Recognition with Convolutional Neural Networks

Alexander Seewald
Computer Vision Lab at Indiana University Bloomington

Neural Networks



A fully connected neural network with a single hidden layer.

Mathematical Techniques

$$W = w_{i,j}, b \qquad y_0 = (\vec{x}, t) \qquad y_i = \sum_j w_{i,j} y_i - 1, j \tag{1}$$

$$sig(x) = \frac{1}{1 - e^{-x}} \quad tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad relu(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2}$$

$$E(y,t) = \frac{1}{2}(t-y)^2 \qquad G = \frac{\partial E}{\partial w_{i,j}} \qquad \Delta w_{i,j} = -\mu \frac{\partial E}{\partial w_{i,j}} \tag{3}$$

A Basic Explanation of Forward Feed and Back Propagation (Training)

Initially, all weights are small random values. When input is run all the way through the neural network, the ground truth label t, the weights $w_{i,j}$, and the prediction y_f are known. The error E can be computed and, then, so can the partial derivative of the error with respect to each of the weights. The weights are then changed such that the error declines most steeply (opposite the direction of the gradient). Such a naive approach would obviously get into trouble with local minima; more advanced methods are discussed in the procedure section.

Convolutional Neural Networks (CNN)

CNNs are a generalization of the simple neural networks described above. They have the following features...

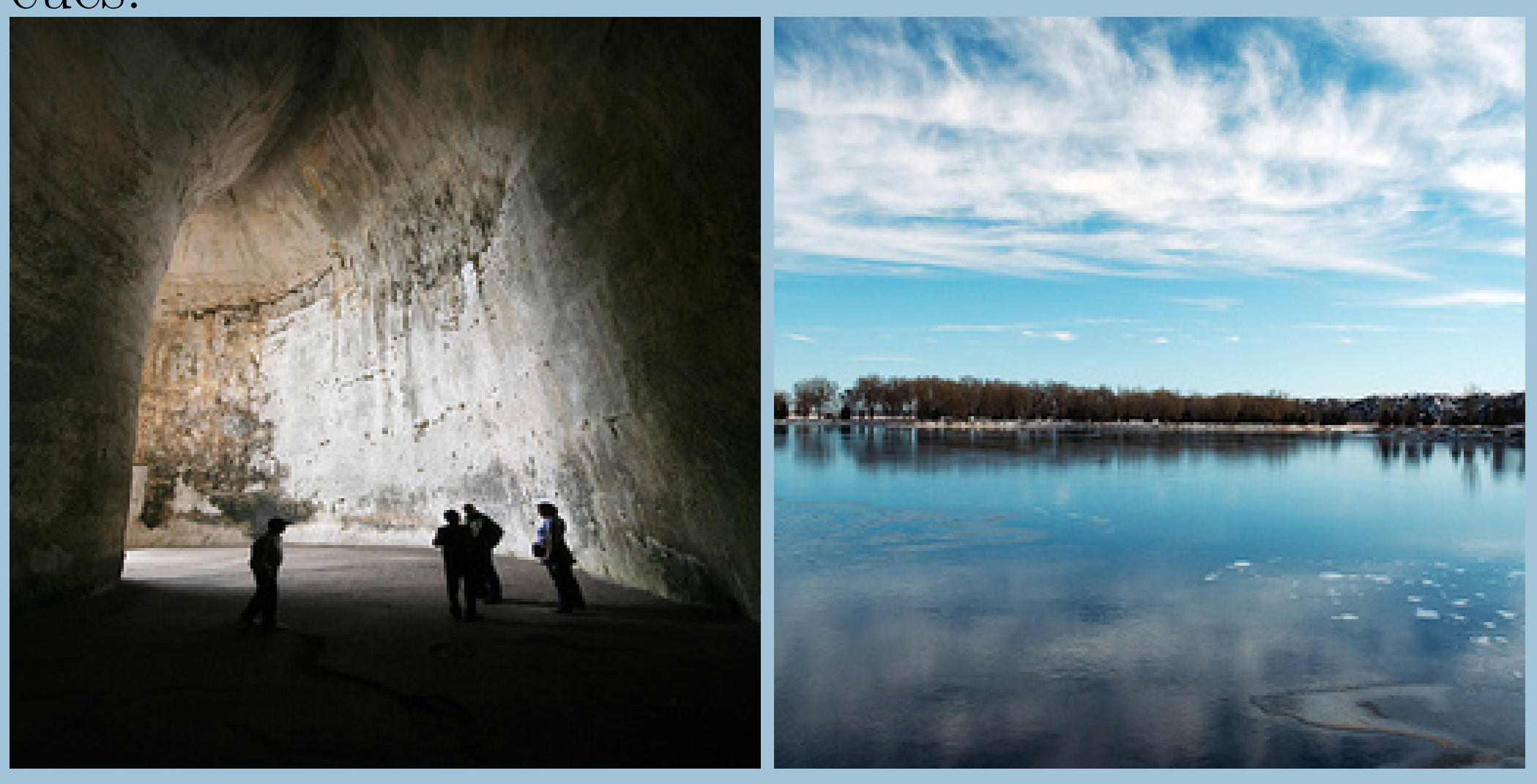
- Inner product layers, also known as fully connected layers, do what most people understand neural networks to do.
- Convolution layers perform a convolution operation with kernels such as... $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$ This has the effect of sharpening edges.
- Pooling layers homogenize a neighborhood to the value of a certain pixel, perhaps the pixel of maximum intensity and perhaps that of average intensity. Pooling reduces the computational complexity of the problem.
- Dropout layers randomly set an activation function's value to 0 every so often.

Support Vector Machines

Like neural networks, SVMs store knowledge and define a loss function which training attempts to minimize. The difference is that SVMs' knowledge is a hypersurface that separates clustered data (which correspond to categories). Because SVMs are purely general purpose, they can be connected to neural networks as part of a prediction ensemble.

Why Snow?

Snow's presence in digital images is physically similar to other phenomena, such as glare and overexposure. Where humans see a wintery scene that may trigger memories of cold and play, a naive computer program sees a region of highly saturated pixels. Humans' ease in classifying snow is based on a vast amount of knowledge about the world and the ability to pick up on context cues.



The reflected sunlight in the image on the left may look like snow to a machine, but is clearly a stone surface to a human can examine the whole scene. The image on the right could easily be interpreted as an upside down lake with snow on top and trees in the distance. However, humans can observe the symmetry between the trees and their reflection, identify the true lake line, and infer that the white fluff must be clouds. In order for computers to compete, an understanding of the scene must be had. Convolutional neural networks have been shown to have special promise in scene recognition [1].

Methods

I trained many neural networks with a variety of initial learning rates, dynamic learning rate function types, and parameters to those dynamic functions. The idea is to characterize the best way to learn the binary snow classification problem.

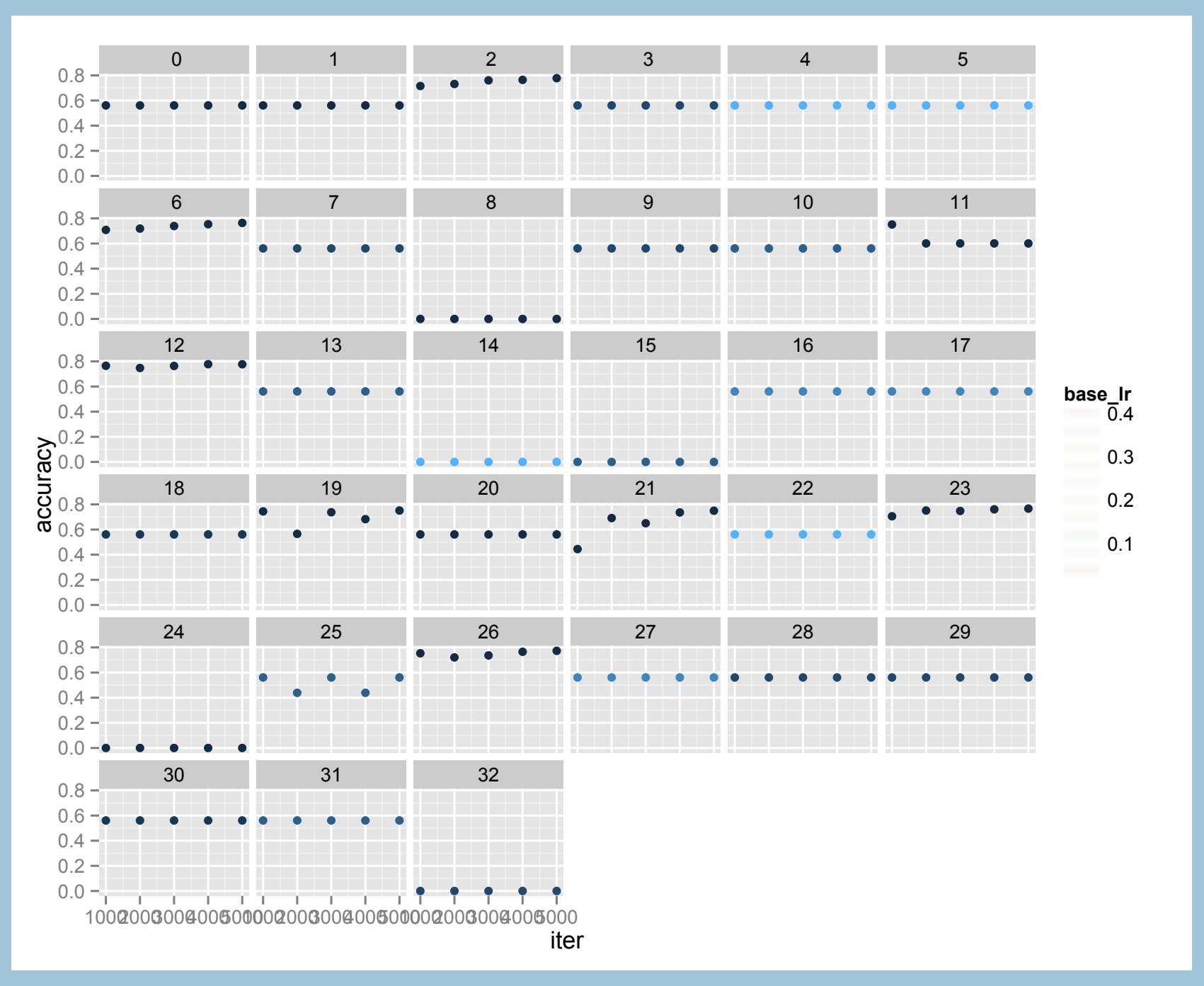
There are a number of design decisions to make. When doing back propogation weights may be updated as soon as their change in value due to a single image is computed, or the changes may be stored in an accumulator variable until 'N' images passing through (at which point all the changes will be applied at once). The latter approach is called batch learning. Because the first images have less of an undue influence, machine learning done in batches tends to generalize better. In addition to making intuitive sense, this is a well-documented result in the literature [2].

There exists some software called caffe that implements highly-flexible CNN forward feed and back propagation solvers [3]. One simply specifies a network topology in a Google protocol buffer (see below for a sample) and can start experimenting.

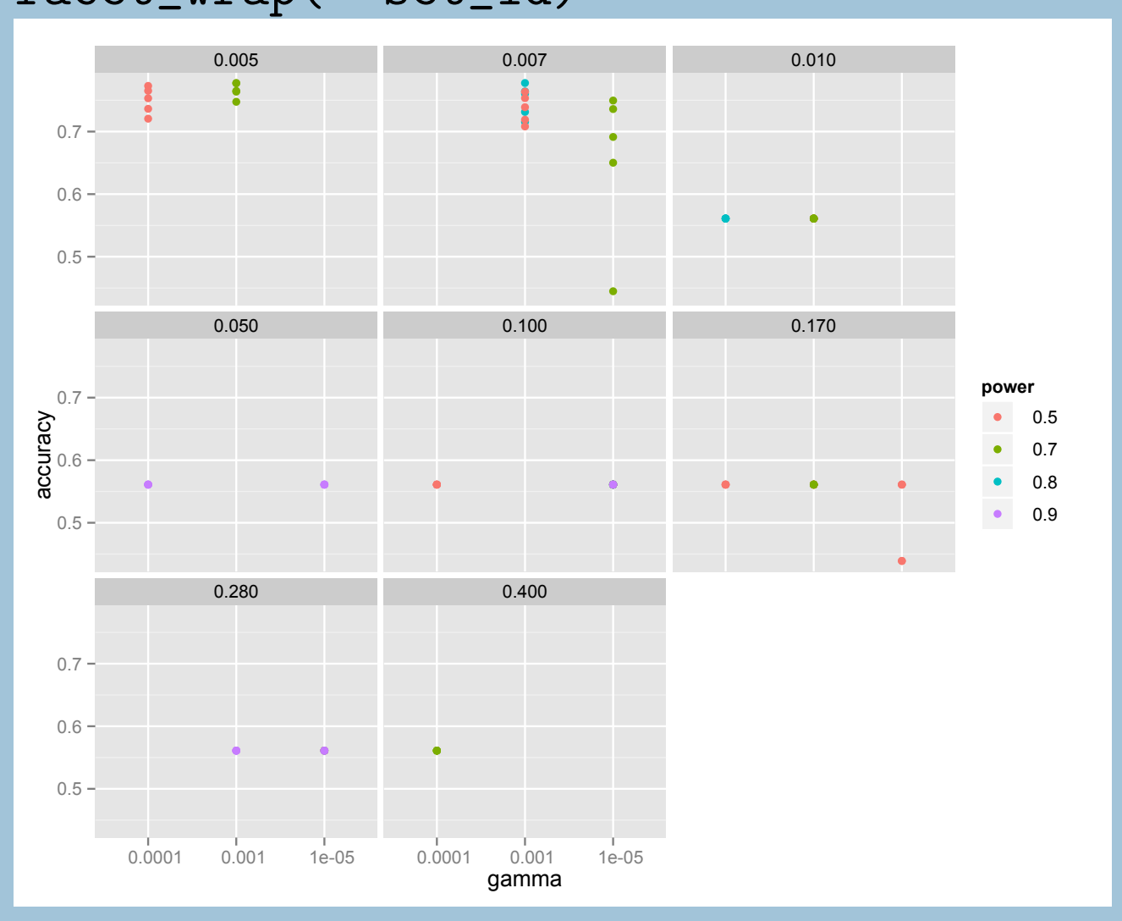
```
      layers {
    name:  "conv2"
  type:  CONVOLUTION
   bottom:  "norm1"
    top:  "conv2"
     blobs_lr:  1
     blobs_lr:  2
   weight_decay:  1
   weight_decay:  0
  convolution_param {
          num_output:  256
               pad:  2
           kernel_size:  5
              group:  2
    weight_filler {
              type:  "gaussian"
                  std:  0.01
           }
     bias_filler {
              type:  "constant"
                 value:  1
           }
           }
           }
```

Another method I used to recognize snow is to extract image features at various points in the network and pass the data off to an SVM for classification. Such an approach has been found to be successful [4]. Results in these experiments are preliminary.

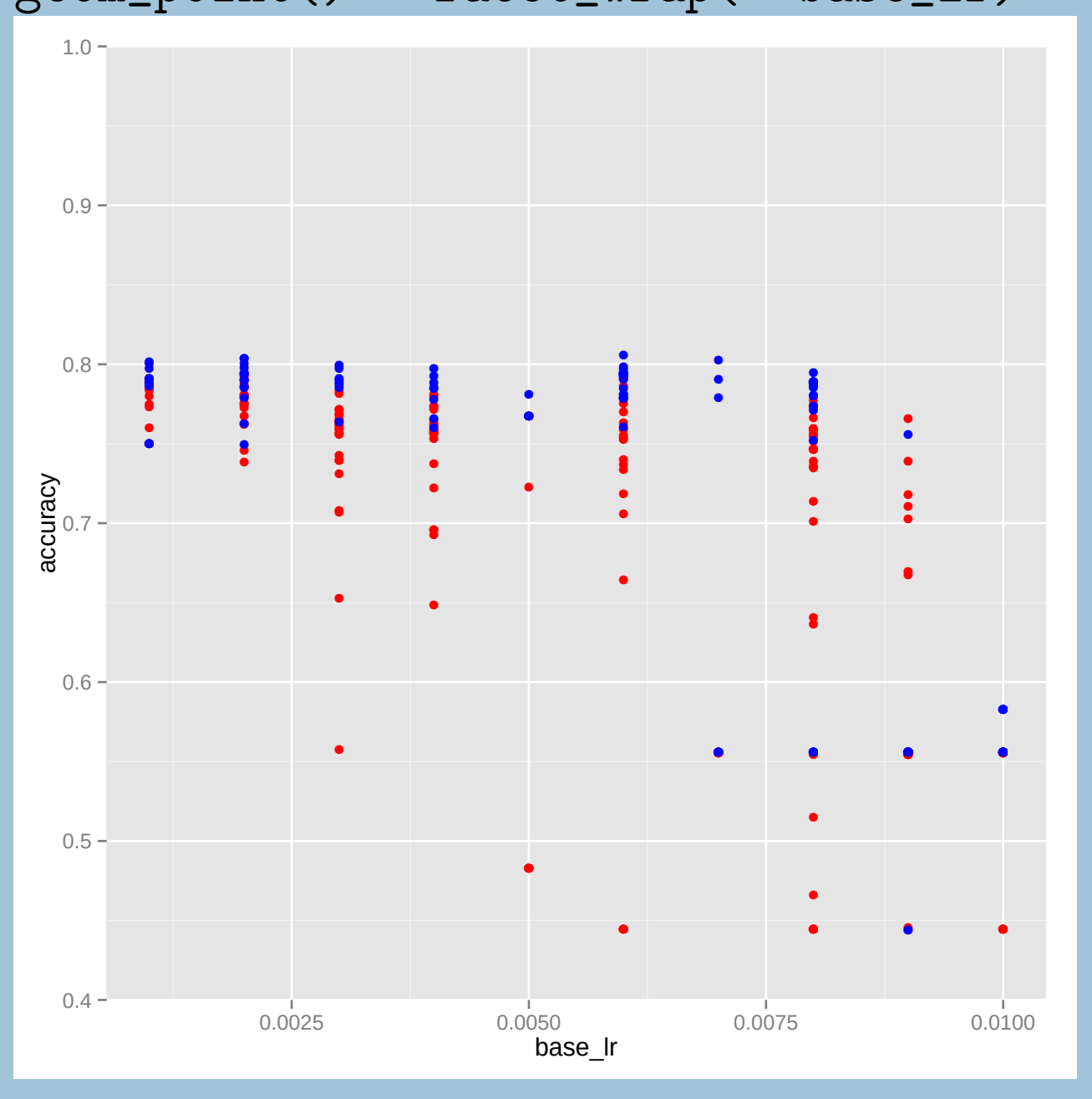
Results



```
ggplot( data=the_data, aes(x = iter, y = ac-
curacy, colour = base_lr)) + geom_point() +
facet_wrap(  set_id)
```



```
ggplot(data=the_data[the_data$base_lr == lrs[i]],
aes(x = gamma, y = accuracy, colour = power)) +
geom_point() + facet_wrap(  base_lr)
```



```
ggplot() + geom_point(data=no_mean, aes(x
= base_lr, y = accuracy), colour='red') +
geom_point(data=yes_mean, aes(x = base_lr, y =
accuracy), colour='blue') + coord_cartesian(ylim
= c(0.4,1.0))
```

Analysis

1. One starkly evident property of the data is that many of the models have the same exact predictive accuracy. These common values are 56%, 63%, and 80%. This is most likely because there are a few local minima of the loss function somewhere in the parameter space that are hard to tunnel out of. Because the value of 56% accompanies high initial learning rates, μ_0, changing the weights too rapidly seems to have the effect of entirely skipping over the favorable values.
2. Performance tends to be slightly better when the mean pixel intensity value for a given image is subtracted from every pixel value. This does not come as a surprise; it is a commonly used technique whose benefit is shown here.
3. Classification accuracy can vary significantly over time. Good engineering practice is, then, to generate models at many points in the training and use the best one for discovery.

References

[1] Aditya Khosla and Byoungkwon An and Joseph J. Lim and Antonio Torralba, Looking Beyond the Visible Scene. 2014

[2] Ofer Dekel, From Online to Batch Learning with Cutoff-Averaging. D. Koller and D. Schuurmans and Y. Bengio and L. Bottou, 2008

[3] Yangqing Jia, Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding. http://caffe.berkeleyvision.org/ 2013

[4] Oquab, Bottou, Laptev, Sivic, Learning and Transferring Mid-Level Image Represenations using Convolutional Neural Networks. 2013

Acknowledgements

This work was supported in part by the National Science Foundation through CAREER award IIS-1253549 and through the FutureGrid project (ACI-0910812).