Building Convolution Layer

* Convolution Layer
  + Input Channels
  + Number of filters per channel
  + Filter size, step, and input padding
    - FFT implementation of filter
  + Dimension of input and convolution/correlation. 1D, 2D, 3D, greater
  + Convolution or Correlation   
    (Literature suggests inversing Kernel in some way makes a convolution filter into a correlation filter. Not sure most aren’t overlooking some things)
* Activation Layer
* Pooling Layer
* Flattening Layer

How to abstract the convo layer from the FC layer?

The convo layer is a different animal. Perhaps leave it separate. A model can consist of a convo layer feeding into a FC layer.

This is the fully connected layer Eval function:

ColVector Eval(const ColVector& \_x) {

X.topRows(InputSize) = \_x;

X(InputSize) = 1; // This accounts for the bias weight.

Temp = W \* X;

return pActive->Eval( Temp );

}

Could ColVectors be used for the Convo layer?

For 1D Convo it works!

How about 2D Convo?

Matrix M 🡪 an image

We

Specify the Input dimension, Kernel,