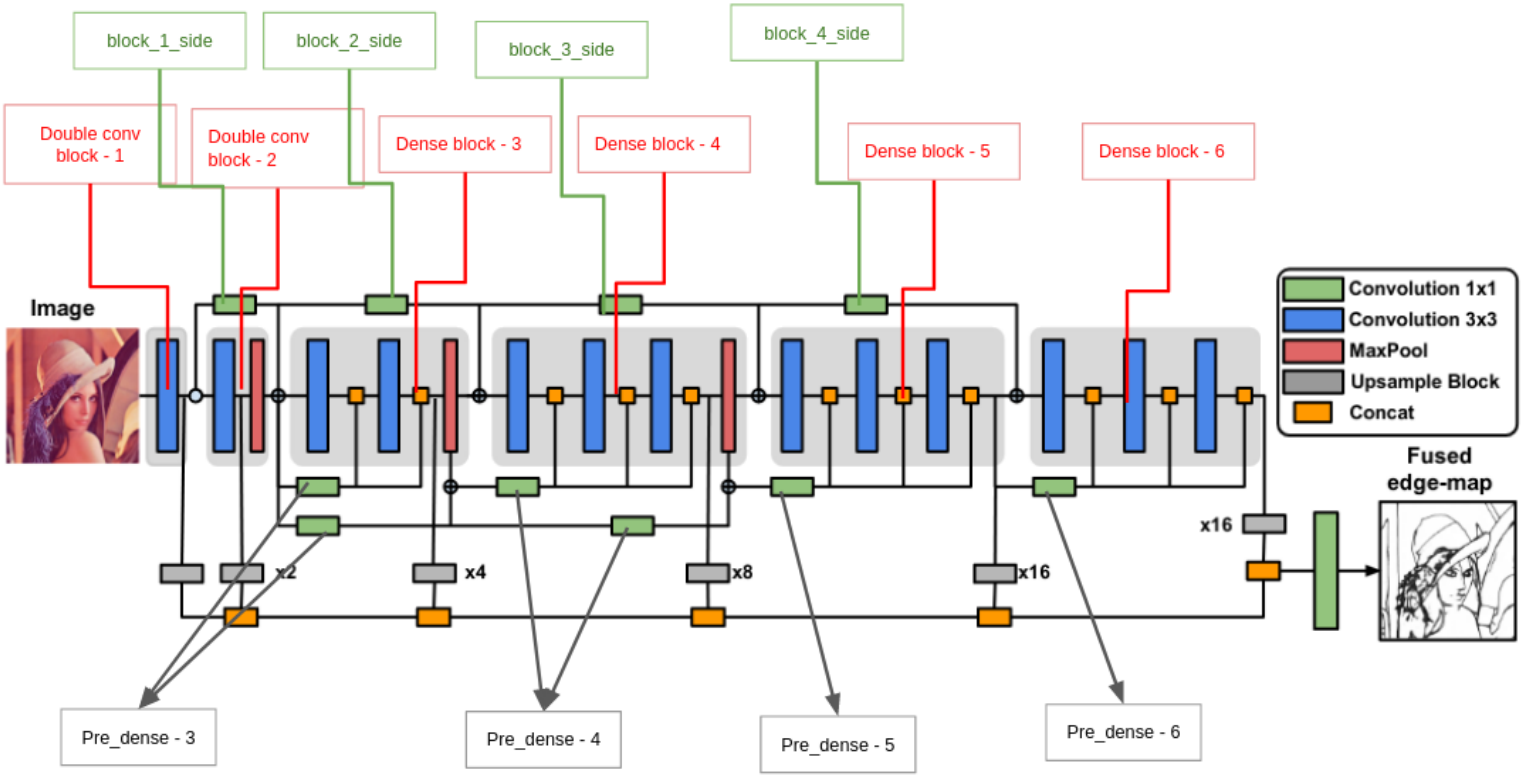
3rd_to_7th_May_Update

1. DexiNed Model Architecture



The various layers can be split into the following parts to describe the way batch normalization is spread in these blocks:
A. Side blocks = 1 batchnorm layer
B. Pre_dense - 3 = 1 batchnorm layer
C. Double_conv block = 2 batchnorm layers
D. Dense block = 4 / 6 batchnorm layers

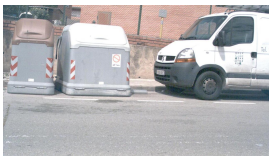
2. **Working of model**:
A. The input image is first fed into the Double conv block 1 (which contains 2 conv operations, 2 batchnorm operations, Relu activation), and the flow of features in the model is as shown in the diagram.
B. The output of each block produces an intermediate edge map (for example at the end of *Doube_conv_block-1*, *Double_conv_block-2*, …, *Dense_block-6,* edge maps are produced) which are concated and fused/ averaged to produce the final edge map prediction.
C. There are in total 6 main blocks (shaded gray in the img) which give out intermediate edge maps which are averaged/ fused to give out the final edge

map prediction.
D. The ***block_side*** and ***pre_dense*** layers are used as means of carrying forward the important edge information that is lost in the deeper layers due to multiple convolution operations being performed.

3. The model gives 2 sets of outputs namely - averaged and fused (as discussed in the paper).
   Avg = the average of all the 6 intermediate edge maps from the respective upsampling blocks (better quantitatively - gives better error metric values - as mentioned in the paper)
   Fused = the 6 intermediate edge maps are fused (better qualitatively)

4. The inference runs for the following layers were given for 5 images from the BIPED dataset and the predicted edge maps are shown below.

Input images

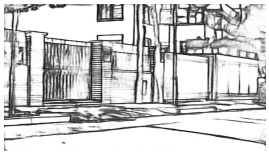
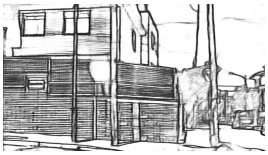
Img1	Img2	Img3	Img4	Img5
				

1) Approximation on *block1.bn1*

a) Avg

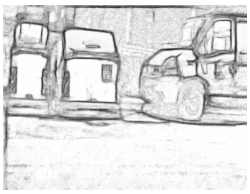
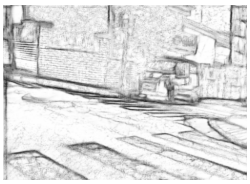
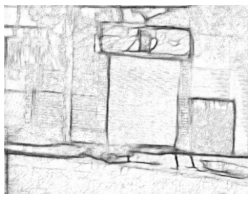
Edge map 1	Edge map 2	Edge map 3	Edge map 4	Edge map 5
				

b) Fused

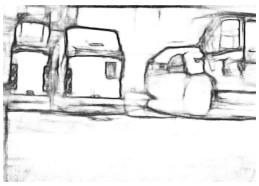
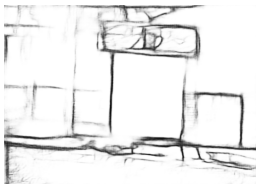
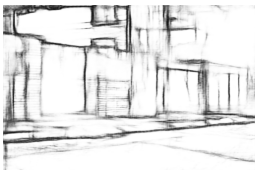
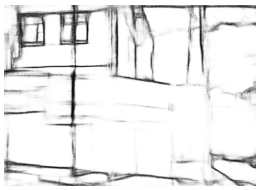
Edge map 1	Edge map 2	Edge map 3	Edge map 4	Edge map 5
				

2) Approximation on *block1.bn2*

a) Avg

Edge map 1	Edge map 2	Edge map 3	Edge map 4	Edge map 5
				

b) Fused

Edge map 1	Edge map 2	Edge map 3	Edge map 4	Edge map 5
				

The error metrics values for these runs will be provided along with the next update in Dropbox.

5. The codes for the following have been added in the github repo for your reference:
   - matlab evaluation for error metrics - ODS, OIS and AP
   - pytorch implementation of model and custom batchnorm function
   - excel sheet updated with the model architecture and the number of Batchnorm2d layers

6. BatchNorm2d implementation details:
The division in batchnorm for each layer is done with different divisors. As you can see, the definition of batchnorm2d is given as:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

Here
E(x) - is the mean value of the input done on a per channel basis.
Var(x) - is the variance of input done on per channel basis
epsilon - is a constant val as defined in the documentation defaulted to the value 1e-5
gamma and beta - are the learnable parameters (weight and bias respectively), which are vectors of dimensions 1xC (where C is the number of channels)