Blind Spot problem in PixelCNN

Neha Muthiyan Follow
Jun 14, 2017 · 3 min read

In the last post on summary of PixelRNN research paper, we learnt about PixelCNN.

Overview of PixelCNN-

Row and Diagonal LSTM layers cover long range dependencies among the pixels in the images. Learning of long range dependencies comes at a computational cost due to complex nature of LSTM layers. Standard convolutional layers can capture a bounded receptive field and compute features for all pixel positions at once. PixelCNN reduces the computational cost required in Row LSTM and diagonal BLSTM but suffers from the problem of blind spot.

Blind spot problem is basically, not including all the previous pixels in the context/history used to compute the hidden state of a pixel. According to the PixelRNN research paper, value of a pixel depends on all the previously predicted pixels. Thus, blind spot can be termed as loss of information in a broad way.

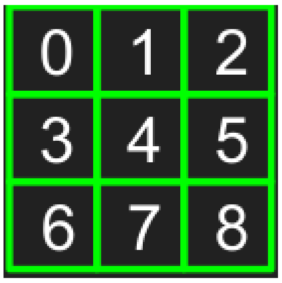

Figure 1: 3x3 filter (each element is a weight)

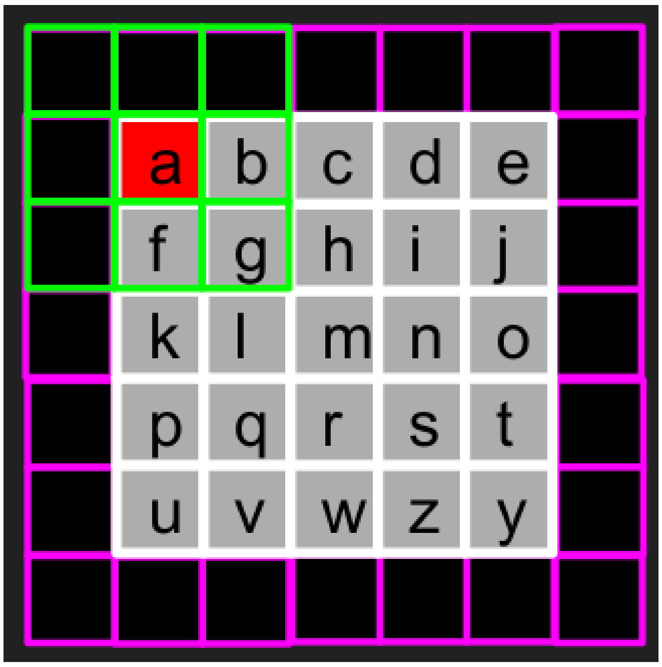

Figure 2: 5x5 zero padded image(pink portion is the zero padding)

In figure 2, red box denotes the the value/pixel to be predicted. The pixel to be predicted is always at the centre of the filter.

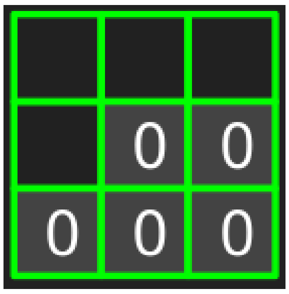

Figure 3: filter to predict the value 'a' in image

Note: The elements of the filter which are not zero are part of zero padding. (Dark gray coloured boxes in the filter (green) denote the 'zeroed' future values.

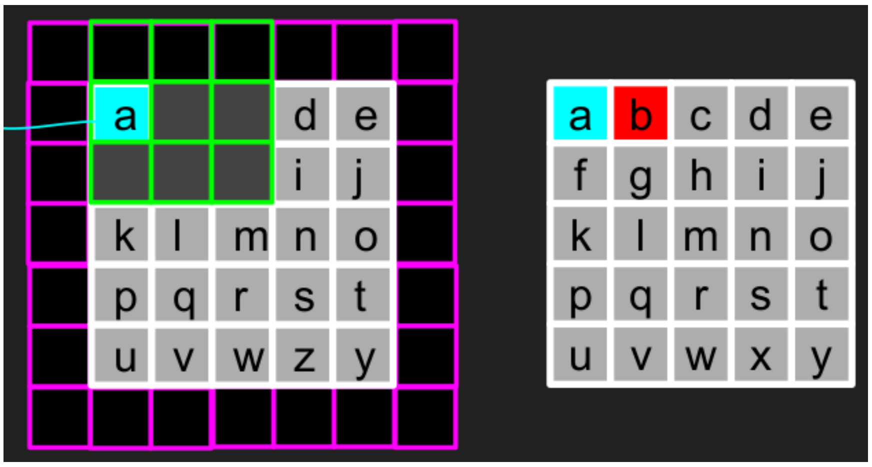

Figure 4: Filter(in green) to predict the value 'b'

Value ‘b’ (in red) denotes the value to be predicted and value in cyan colour denotes previously predicted values. Here, ‘a’ has been predicted and the filter (left image in figure 4) includes previously predicted pixel value ‘a’ and doesn’t include future pixels. Thus, the value of pixel in ‘red’ depends on previous pixels.

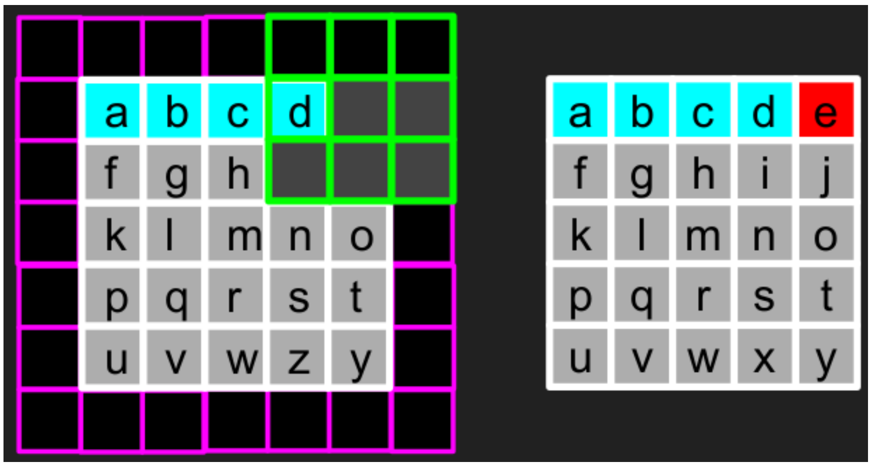

Figure 5: Filter(in green) to predict the value 'e'

In figure 5, the pixel to be predicted (‘e’) includes pixel ‘d’ in its filter. ‘d’ depends on ‘c’; ‘c’ depends on ‘b’ and ‘b’ depends on ‘a’. Thus, directly or indirectly ‘e’ depends on ‘a’, ‘b’, ‘c’, ‘d’.

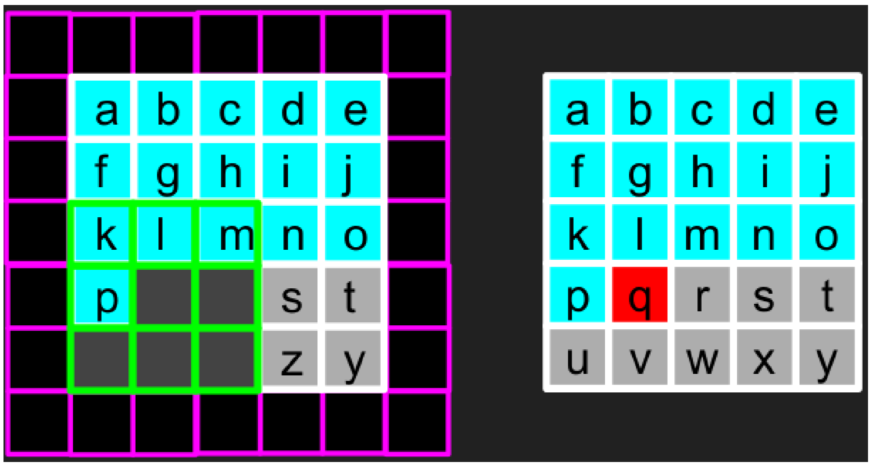

Figure 6: Filter to predict the value 'q'

As shown in figure 6, pixel with value 'q' is at the centre of the filter. Future values/pixels (including 'q') have been zeroed out by making the weights for those pixels as 0. The predicted value of pixel at 'q' depends on 'k', 'l', 'm', 'p'. 'k', 'l', 'm', 'p' in turn depend on f, g, h, i which in turn depend on a,b,c,d,e.

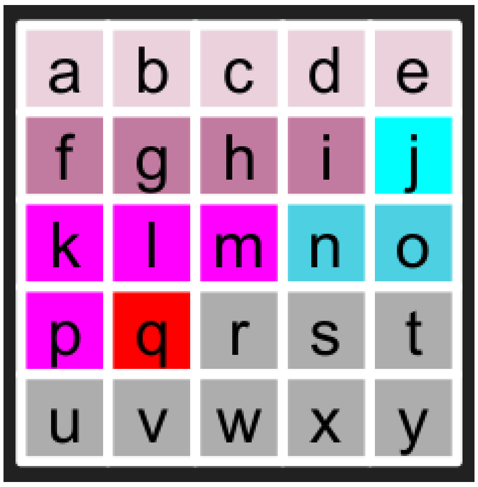

Figure 7: Blind Spot in predicting the pixel 'q'

As in figure 7, pixel at 'q' depends on all previously predicted pixels except 'j', 'n', 'o'. The pixels in cyan colour ain't used in prediction of 'q'.

This is called 'Blind Spot'. Pixels predicted using PixelCNN, aren't dependent on all previous pixels, which is undesirable.

To solve the problem of Blind spot 'Gated PixelCNN' has been introduced which will be covered later.