1.1.1

Let there be n number of layers in the network, then $f(x)$ would have n nested Sigmoid functions. By applying chain rule for backpropagation, we know $\frac{df(x)}{dx}$ would be a product of n first derivative of sigmoid functions. Therefore, we can first determine the max of the derivative of sigmoid function.

$$\frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \left(\frac{1}{1+e^{-x}}\right)\left(1-\frac{1}{1+e^{-x}}\right) = \sigma(x)\big(1-\sigma(x)\big) = \sigma(x) - \sigma(x)^2$$

We can then take the second derivative with respect to sigmoid function and set it to zero to find the global maximum of sigmoid function:

$$\frac{d}{d\sigma(x)}\sigma(x) - \sigma(x)^2 = 1 - 2\sigma(x) = 0$$

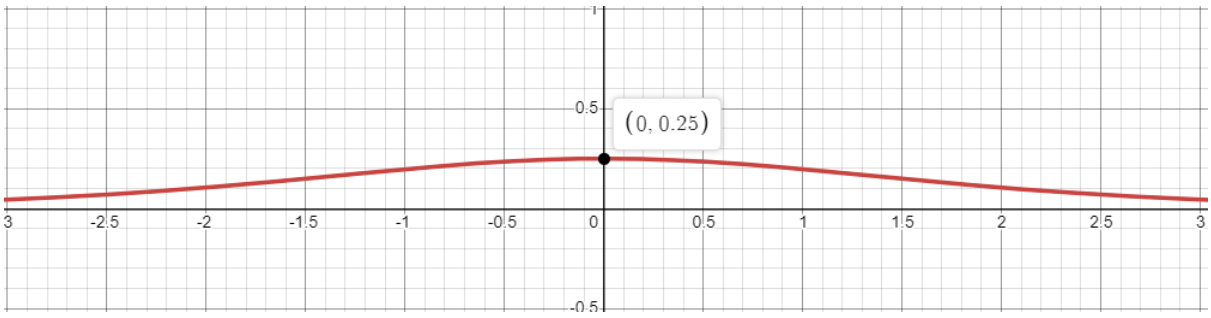
$$\sigma(x) = \frac{1}{2}$$

Thus, we know we can obtain maximum value of the first derivative of sigmoid function when sigma is equal to ½.

$$\sigma(x) - \sigma(x)^2 = \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$$

Therefore, we now know the maximum value of the first derivative of sigmoid is ¼.

To make a sanity check, by plotting the first derivative, we can confirm that the global max is ¼.



Since we know $0 \leq \sigma(x) \leq 1$, the minimum value that the first derivative of sigmoid function must also be at least 0.

Therefore, given a n layered network, the backpropagation will give a product of n first derivative of sigmoid function, of which the maximum of each first derivative is ¼ and minimum is 0.

We can conclude that:

$$0 \leq \left|\frac{\partial f(x)}{\partial x}\right| \leq \left(\frac{1}{4}\right)^n$$

Since the gradient is upper bounded by a very small number, the gradients are necessarily to have to vanish during backpropagation.

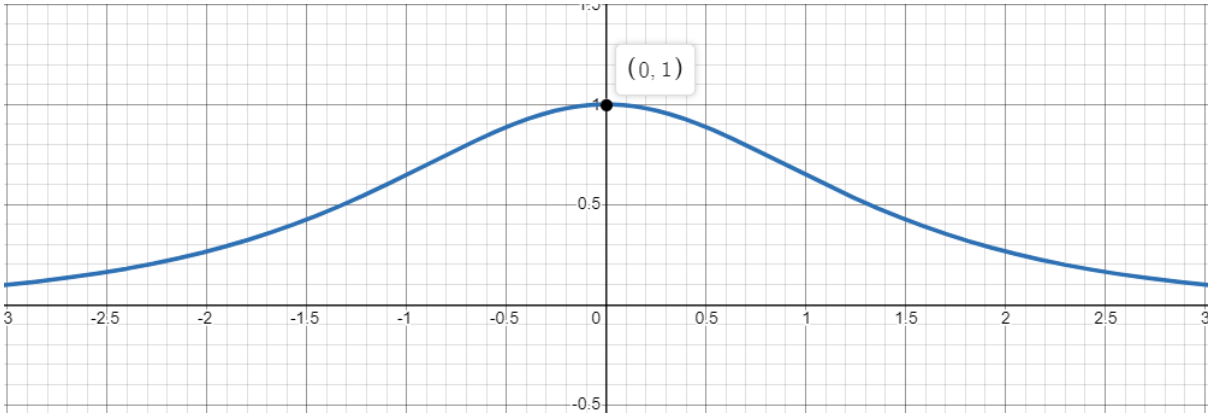
1.1.2

Similarly, we must find the maximum and minimum of the first derivative of the tanh function:

$$\frac{d}{dx}\tanh x = 1 - \tanh^2 x = \operatorname{sech}^2 x$$

We know that $\operatorname{sech} x$ is bounded between 0 and 1, thus we know $0 \leq \operatorname{sech}^2 x \leq 1$.

To perform sanity check, we make a plot:



Therefore, given a n layered network, the backpropagation will give a product of n first derivative of tanh functions, of which the maximum of each first derivative is 1 and minimum is 0.

Thus, we can conclude that:

$$0 \leq \left|\frac{\partial f(x)}{\partial x}\right| \leq 1^n$$

Since the gradients are always smaller or equal to 1, and it can be as small as 0, it is still necessarily to have to vanish as they are backpropagated because the value of the gradients will almost always decrease.

1.2.1

To find the maximum singular value of the input-output Jacobian, we need to first find the maximum singular value of the Jacobian during one time step. So, first find the derivative of the following:

$$\frac{d}{dx_t}x_{t+1} = \frac{d}{dx_t}\tanh Wx_t = W * \operatorname{sech}^2 Wx_t$$

Since the maximum value of the weight matrix is $\sigma_{max}(W) = \frac{1}{2}$ and the maximum of sech function is 1 no matter what the input is. Therefore, the maximum value of the Jacobian during one time step is as follows:

$$\sigma_{max}\left(\frac{d}{dx_t}x_{t+1}\right) = \sigma_{max}(W * \operatorname{sech}^2 Wx_t) \leq \sigma_{max}(W)\sigma_{max}(\operatorname{sech}^2 Wx_t) = \frac{1}{2} * 1$$

Since the input-output Jacobian will repeat n time steps, we know the Jacobian will be a product of n derivative we just calculated. Thus, we can conclude that the singular value of the input-output Jacobian will be bounded $0 \leq \sigma_{max}\left(\frac{\partial x_n}{\partial x_1}\right) \leq \left(\frac{1}{2}\right)^n$.

1.2.3

From 1.2.2, we know the following statements are true:

$$\sigma_{min}\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \geq 1 - \sigma_{small} \;\; and \;\; \sigma_{max}\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \geq \sigma_{big} - 1$$

Then we know that the minimum value of the Jacobian is a value that is very close to 1 since $\sigma_{small} \ll 1$. Therefore, the residual connections layers can solve the vanishing gradients since the gradients will not vanish as quickly compare to sigmoid and tanh functions.

However, since the maximum value of the Jacobian is larger than $\sigma_{big} - 1$ and $\sigma_{big} \gg 2$, we know the maximum value can be very big. Therefore, the residual connections layers cannot solve the exploding gradients problem.

1.3.2

The architecture on the right is easier to learn in terms of exploding / vanishing gradient because it performs a batch normalization after the residual connections is added. Therefore, this batch normalization will ensure that the gradients will not vanish or blow up. Thus, this architecture will be easier to learn overall.

On the other hand, the architecture on the left adds the residual connection after the batch normalization and ReLU layer, therefore the residual can contribute to exploding / vanishing gradient since the residuals are not normalized. Thus, this architecture is harder to learn.

2.2.1

For each convolutional layer, the number of connections is as follows:

$$= WH(3)^2k^2$$

Then for a d-layered PixelCNN model, the total number of connections is then equal to:

$$= dWH(3)^2k^2 = O(dWHk^2)$$

2.2.2

Suppose that we can compute as many matrix-vector multiplications in parallel as we can, the minimum number of sequential operations is $O(d)$. This is because there are d number of layers, and for each layer, since we can perform the raster scan order by altering the mask, then we can scan all pixels in parallel. Then, for each pixel, we can also perform the matrix multiplication with the kernel. Therefore, if we can maximize the number of parallel matrix-vector multiplication. Thus, resulting in $O(d)$.

2.3.1

For each layer, there are WH hidden units. Then, for each unit, there are 3 connections, one from previous layer, 1 from north neighbor, and 1 from east neighbor. Then for each connection, there are k input and output channels. Therefore, the total number of connections is as follow:

$$= O(dWHk^2)$$

2.3.3

Since MDRNN cannot be parallelized because it would require the north and east neighbors be calculated first, thus this sequential operation cannot be parallelized. On the other hand, as shown from 2.2.2, PixelCNN can have as little as $O(d)$ sequential operations given maximum parallelization.

Therefore, PixelCNN has better parallelization potential than MDRNN.

In addition, since PixelCNN can have far less sequential operations, it would also require less memory space to store intermediate values. Therefore, PixelCNN has lower memory complexity than MDRNN. This also applies to computational complexity because our current hardware allows much faster parallel calculation from libraries such as numpy. Therefore, PixelCNN also has lower computational complexity than MDRNN.

Lastly, since the context window of MDRNN is 3, while the context window of PixelCNN depends on the number of zeros in the mask, i.e. in our case $= 3^2 - \#\ of\ 0$. Therefore, as we increase the number of layers, we would cover much more pixels thus the context window will be larger than 3 with a very high probability. Thus, we can say that PixelCNN has a larger context window than MDRNN.

In conclusion, PixelCNN is almost always better than MDRNN, since it is faster, consumes less space, and perhaps learn more from the input.