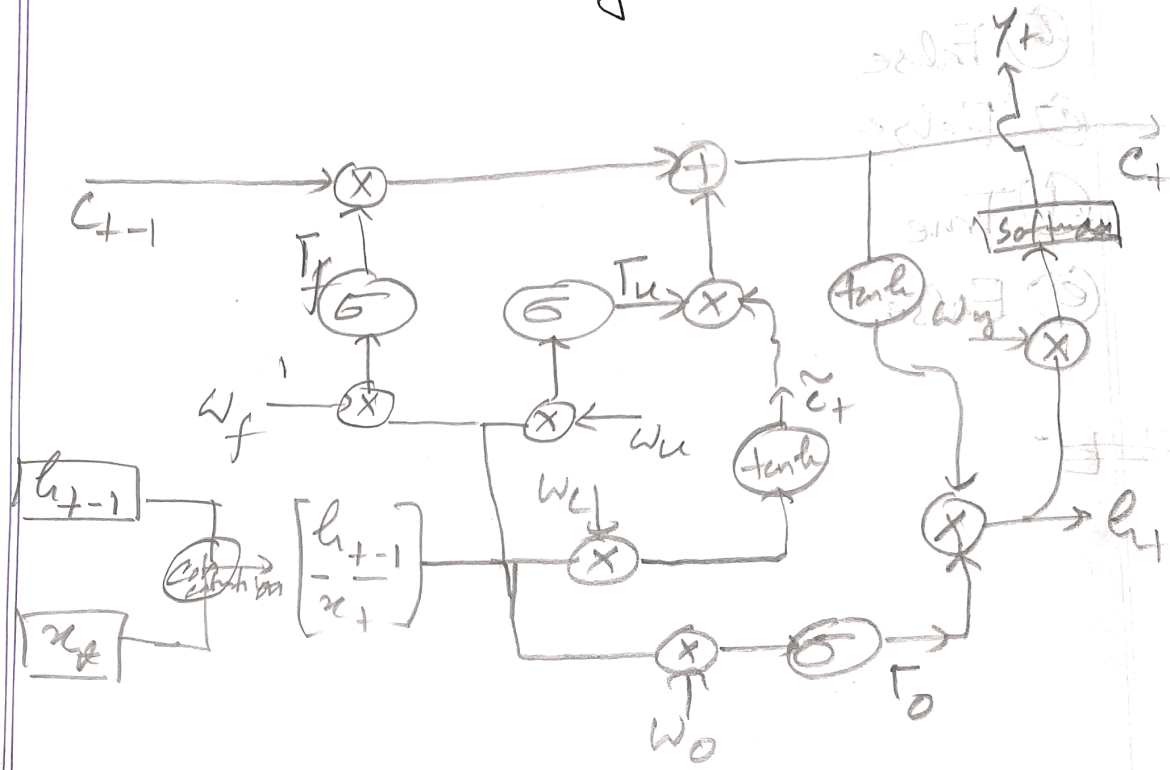
Final - CS 687 Prithul Sarker

1. (a) True
(b) False
(c) False
(d) True
(e) False

5. LSTM Network Diagram




In the above diagram $c_{t-1}$, $c_t$ are cell states. ~~$H_{t-1}$ & $H_t$~~ $h_{t-1}$ & $h_t$ are hidden states. $(t-1)$ & $t$ are previous and next timesteps. $x_t$ is the input.

For easier multiplication, $h_{t-1}$ & $x_t$ are concatenated together as $\begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$.

for updating and removing cell state, two other parameters are used.

$$\Gamma_f = \sigma\left(W_f \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)$$

$$\Gamma_u = \sigma\left(W_u \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)$$

These values are used to update with current and forget previous cell states.

$$\tilde{c}_t = \tanh\left(W_c \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)$$

$$c_t = \Gamma_u \cdot \tilde{c}_t + \Gamma_f \cdot c_{t-1}$$

For new hidden state value,

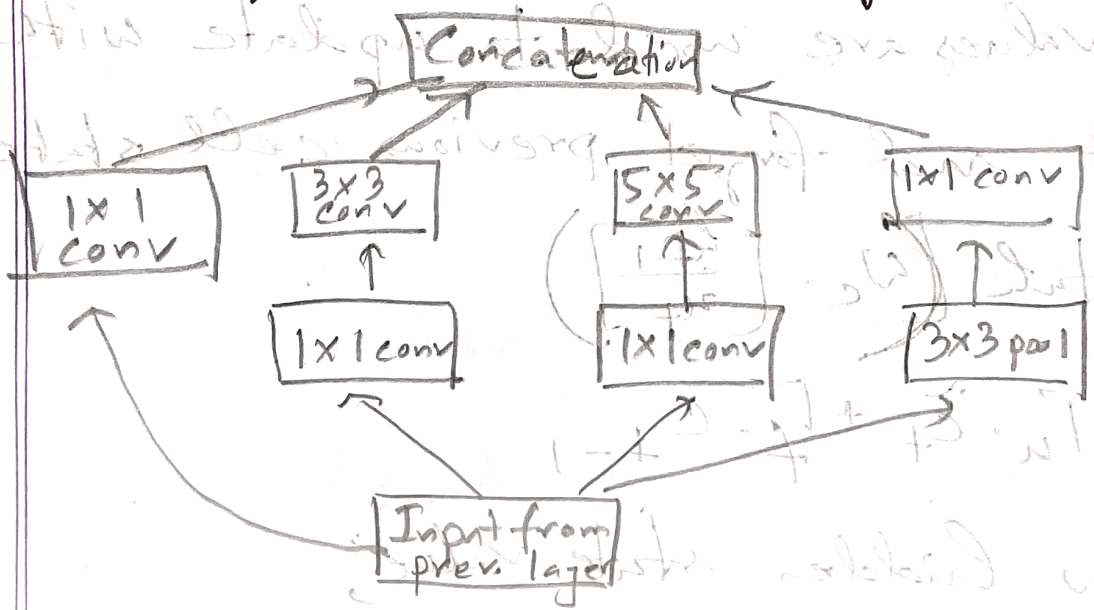
$$h_t = \Gamma_o \tanh(c_t) \quad \text{where } \Gamma_o = \sigma\left(W_o \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)$$

$y = \text{softmax}(W_y \cdot h_t)$

This is the output using new hidden state.

$W_o, W_u, W_f, W_c$ are ~~are~~ two parameters of $h_{t-1}$ and $x_t$ added together for easier matrix multiplication.

6. (a) Inception module in GoogleNet:



(b) GoogleNet uses 1×1 convolutional layer before any other convolution which reduces dimension of the input and eventually number of operations and ~~par~~ overall parameters. Other than this, ~~VGG Net uses~~ GoogLeNet uses parallel convolution maintaining the same output spatial dimension and finally concatenates them. VGG-Net does not perform such parallel convolution or 1×1 convolution. ~~This~~ VGG-Net uses normal convolution. Other than this, VGG-Net ~~does not~~ uses ~~f~~fully connected network before final layer ~~&~~ whereas GoogLeNet does not use FC layers.

(c) Bottleneck layers are those layers which is used to reduce the dimension of the input. This is done using 1*1 convolution before convolving with higher number of filters. This reduces computation numbers, number of parameters and time needed overall.

With the Bottleneck layer, high convolution is performed with less computation.

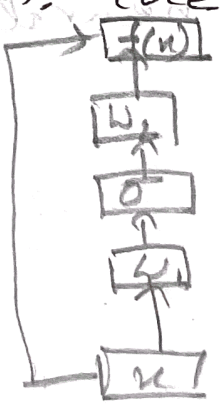
7. (a)

Stacking many layers of VGG-net or GoogLeNet does not improve accuracy. This is because of the degradation problem. In the Res-Net paper, authors mentioned that even though they used identity mappings for the added layers, the accuracy decreased.

(b) Residual Network solves this problem using shortcut connection. The ~~H(x)~~ previously,

$H(x) = W_2 \circ W_1(x)$

but for ResNet,



$H(x) = W_2 \sigma W_1(x) + x$

which improves the gradient at earlier stage and helps the network to approximate values.

In the paper, the authors showed even with 1202 layers, the training error reduced. Before this paper, 22-layer was the max number of layers used for deep neural networks.

Q1.

$$\Gamma_u = \sigma\left(W_u \cdot \left[\frac{c_{t-1}}{x_t}\right]\right)$$

$$\tilde{c}_t = \tanh\left(W_c \cdot \left[\frac{c_{t-1}}{x_t}\right]\right)$$

$$c_t = \Gamma_u \cdot \tilde{c}_t + (1-\Gamma_u) \cdot c_{t-1}$$

$$y_t = \text{softmax}(W_y \cdot c_t)$$

$$\frac{\partial c_t}{\partial \Gamma_u} = \tilde{c}_t - c_{t-1}$$

$$\Gamma_u = \sigma(m) \quad \left[\text{Let } m = W_u \cdot \left[\frac{c_{t-1}}{x_t}\right]\right]$$

$$\frac{\partial \Gamma_u}{\partial m} = \sigma(m)\left[1-\sigma(m)\right]$$

$$\frac{\partial c_t}{\partial m} = \frac{\partial c_t}{\partial \Gamma_u} \cdot \frac{\partial \Gamma_u}{\partial m}$$

$$= (\tilde{c}_t - c_{t-1})\left[\sigma(m)\left[1-\sigma(m)\right]\right]$$

$$\frac{\partial m}{\partial w_u} = \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}$$

$$\frac{\partial c_t}{\partial w_u} = \frac{\partial c_t}{\partial m} \cdot \frac{\partial m}{\partial w_u}$$

$$= (\tilde{c}_t - c_{t-1}) \cdot \left[ \sigma\left(w_u \cdot \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}\right) \left[1 - \sigma\left(w_u \cdot \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}\right)\right] \right] \times \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}$$

[Answer]

$$\frac{\partial c_t}{\partial \tilde{c}_t} = \Gamma_u \qquad \tilde{c}_t = \tanh(n)$$

[Let $n = w_c \cdot \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}$]

$$\frac{\partial \tilde{c}_t}{\partial n} = 1 - \tanh^2(n)$$

$$\frac{\partial n}{\partial w_c} = \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}$$

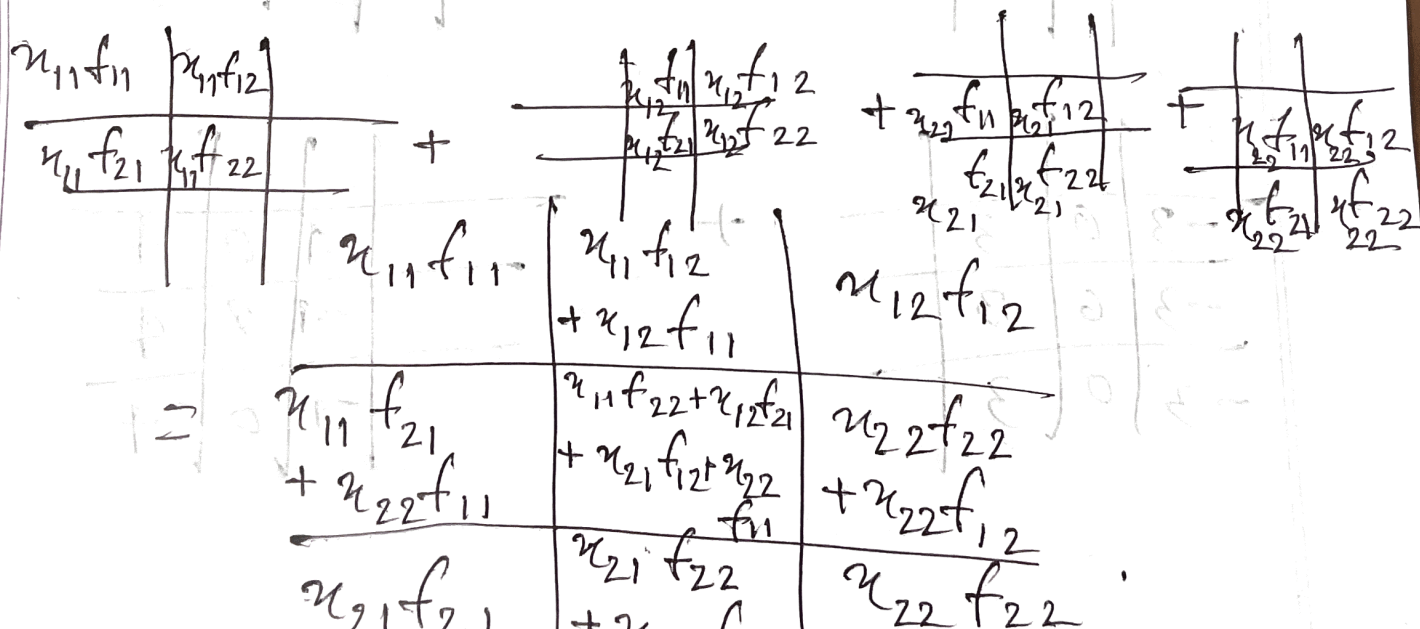
$$\frac{\partial c_t}{\partial w_c} = \frac{\partial c_t}{\partial \tilde{c}_t} \cdot \frac{\partial \tilde{c}_t}{\partial n} \cdot \frac{\partial n}{\partial w_c}$$

$$= \Gamma_u \cdot \left[1 - \tanh^2\left(w_c \cdot \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}\right)\right] \cdot \begin{bmatrix} c_{t-1} \\ x_t \end{bmatrix}$$

[Answer]

2. (a) Let $X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$ is the input and $f = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}$ is the filter. (for 3×3 output)

Transpose conv. of X and f =




$$= \begin{array}{|c|c|c|} \hline x_{11}f_{11} & x_{11}f_{12} + x_{12}f_{11} & x_{12}f_{12} \\ \hline x_{11}f_{21} + x_{21}f_{11} & x_{11}f_{22} + x_{12}f_{21} + x_{21}f_{12} + x_{22}f_{11} & x_{22}f_{12} + x_{12}f_{22} \\ \hline x_{21}f_{21} & x_{21}f_{22} + x_{22}f_{21} & x_{22}f_{22} \\ \hline \end{array}$$

each input is multiplied / convolved with filter / kernel and finally all are added together based on stride.

(b) $X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $M = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 2 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

Upsampling =

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 2 | 1 |
| -1 | 0 | 1 |

+

| | | |
|---|---|---|
| -2 | 0 | 2 |
| -2 | 4 | 2 |
| -2 | 0 | 2 |

+

| | | |
|---|---|---|
| -3 | 0 | 3 |
| -3 | 6 | 3 |
| -3 | 0 | 3 |

+

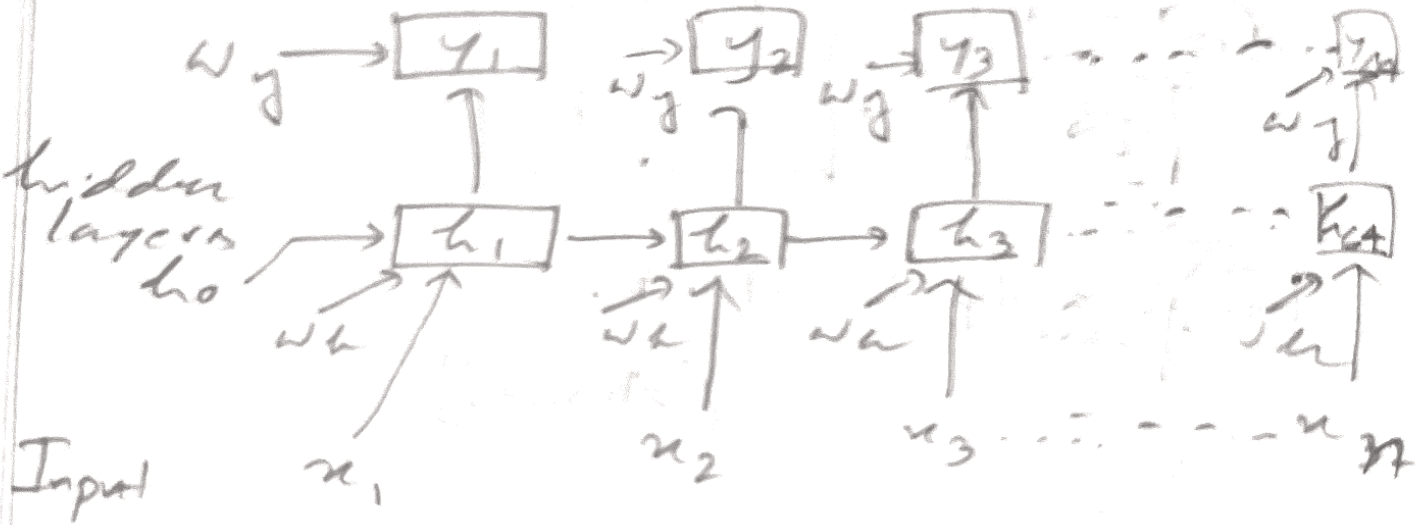
| | | |
|---|---|---|
| -4 | 0 | 4 |
| -4 | 8 | 4 |
| -4 | 0 | 4 |

$$= \begin{array}{|c|c|c|c|} \hline -1 & -2 & 1 & 2 \\ \hline -4 & -4 & 8 & 6 \\ \hline -4 & 0 & 12 & 6 \\ \hline -3 & -4 & 3 & 4 \\ \hline \end{array}$$

Final output 4×4.

[Answer]

1. (a) (b)




Since there are 64 nodes, ~~and~~ there $h_{64}$. in the hidden layers

The inputs and outputs are $x_1, x_2, x_3 \ldots x_n$ and $y_1, y_2 \ldots y_n$ respectively.

(c) Hidden layer ~~and~~ at t timestep:

$$h_t = \tanh\left(W_h \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)$$

$$y_t = \text{softmax}(W_y \cdot h_t)$$

(c) Input dimension $32 \times 1$

Hidden state dimension $64 \times 1$

output dimension $32 \times 1$

(d) Here, 2 parameters are available $\rightarrow W_h$ & $W_y$.

dimension of $W_{hh} \rightarrow 64 \times 64$

$W_{xh} \rightarrow 64 \times 32$

So dimension of $W_h \rightarrow 64 \times 96$

Dimension of $W_y \rightarrow 32 \times 64$.

9. (a) Deep convolutional GAN uses ReLU activation for all layers in generator except for the output which uses tanh and LeakyReLU for discriminator. The original GAN used maxout activation function. Also, DCGAN removed FC layers which is used in GAN. DCGAN also uses batchnorms for both Generator and discriminator.

(a) DCGAN should fail when number of layers are more for degradation problem. And since (b) with lower layers, they only learn from the receptive fields, the outputs should be somewhat similar.

| Layer | Input Size | Kernel Size | Output Size | Pad | No. of Params (Neglecting bias) | No. of Ops |
|---|---|---|---|---|---|---|
| L-1-Conv-1 | 64x64x3 | 7x7x3 | 64x64x32 | 3x3 | 4704 | 19267584 |
| L-1-Conv-2 | 64x64x32 | 3x3x32 | 64x64x64 | 1x1 | 18432 | 75497472 |
| L-1-Pool | 64x64x64 | 3x3 | 32x32x64 | 0.5x0.5(s = 2) So 1 padding in either side | 0 | 2359296 |
| L2-Conv-1 | 32x32x64 | 3x3x64 | 32x32x128 | 1x1 | 73728 | 75497472 |
| L2-Conv-2 | 32x32x128 | 3x3x128 | 32x32x64 | 1x1 | 73728 | 75497472 |
| L2-Conv-3 | 32x32x64 | 3x3x64 | 32x32x16 | 1x1 | 9216 | 9437184 |
| L-2-Pool | 32x32x16 | 3x3 | 16x16x16 | 0.5x0.5 (s =2) 1 padding either side | 0 | 147456 |
| L3-FC-1 | 4096 | NA | 1000 | N/A | 4096000 | 4096000 |
| L3-FC-2 | 1000 | NA | 1000 | N/A | 1000000 | 1000000 |
| L3-Softmax | 1000 | NA | 10 | N/A | 10000 | 10000 |

Prithul Sarker