N. KAUSIK
CS21M037

CS6910 End Sem

1. a) Zero Initialisation (All weights same so grads will always be same)

b) L1 Regularisation

c) Instead of training large number of networks for ensemble, we can use DROPOUT which is efficient as we dont need to train all the different networks from scratch. Instead, at each training instance, with some prob p, drop some neuron in the network. So, we get a diff network everytime but also weights are shared accross them.

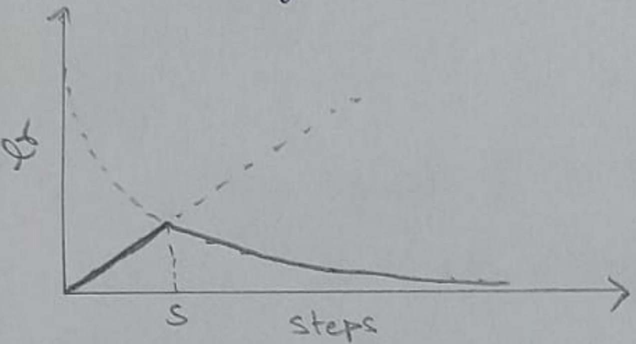
d) The Read, Write and Forget gates

e)
$$W' = \frac{W - F + 2P}{S} + 1$$

$$H' = \frac{H - F + 2P}{S} + 1$$

2. a) $\text{Gelu}(x) = x \underbrace{\phi(x)}_{}$ → Std Gaussian CDF

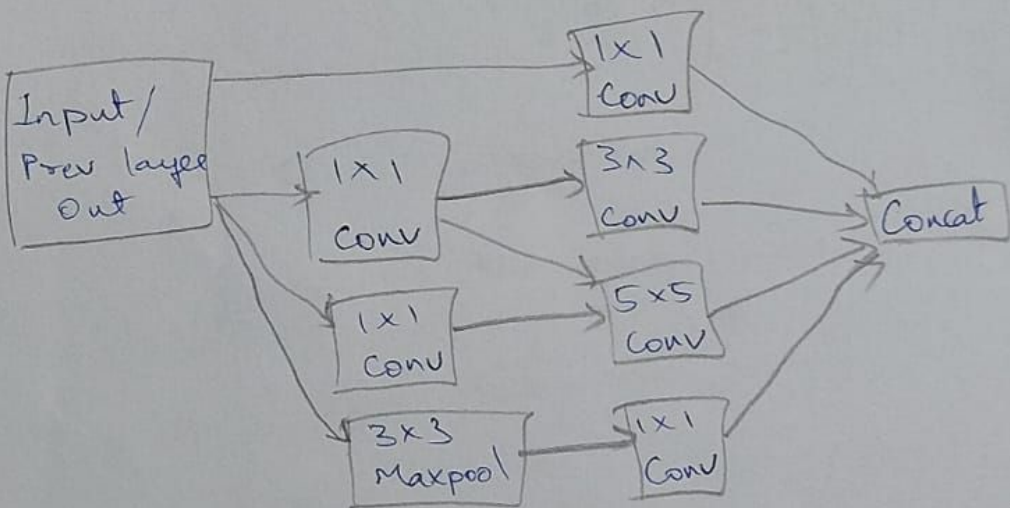
b) Learning Rate with warmup



increase linearly upto step s (warmup) then decrease

eg. $lr = \begin{cases} t \cdot k & \text{if } t < s \\ t^{-1/2} & \text{if } t > s \end{cases}$ (k → const)

c)



d) GRU ⇒ Gates ⇒ $o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$

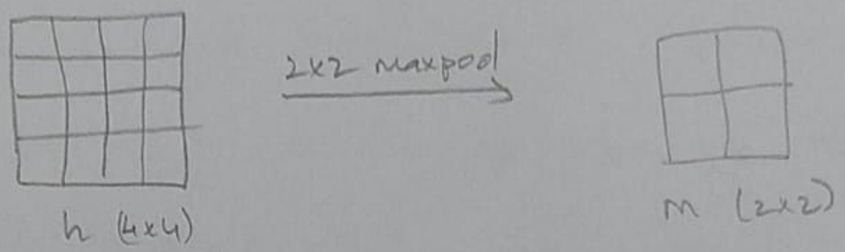
$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$

States ⇒ $\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$

$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$

e) $f_{att}(h_i, c_j) = V_{att}^T \tanh(W_{att}[h_i ; c_j])$

4.



$\frac{\partial L}{\partial m}$ is known. In back prop, due to max pool layer, the gradient propogates to only the 'max' cells in h. All other cells get gradient 0.

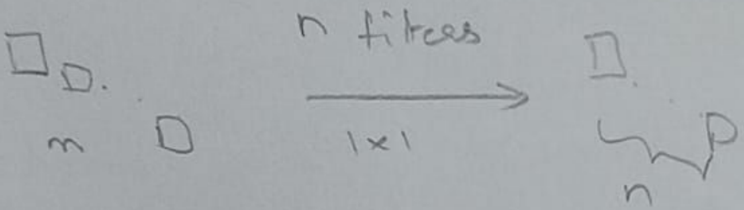
$$\frac{\partial L}{\partial h_{ij}} = \begin{cases} \frac{\partial L}{\partial m} & \text{if } h_{ij} = m_{\lfloor \frac{i-1}{2} \rfloor + 1 \lfloor \frac{j-1}{2} \rfloor + 1} \\ 0 & \text{otherwise.} \end{cases}$$

$m_{\lfloor \frac{i-1}{2} \rfloor + 1 \, \lfloor \frac{j-1}{2} \rfloor + 1}$ is the max value of the cells around h_{ij} when doing max pooling.

5. The m-dim input can be represented as $m \times (1 \times 1)$.

We can get n-dim output by applying 'n' filters that reduce the $m \times (1\times1)$ to 1×1.

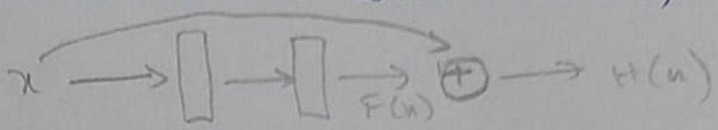
We use filter of size (1×1) and do 1×1 convolution to get 1×1 from the $m\times(1\times1)$ and we do using n such filters.



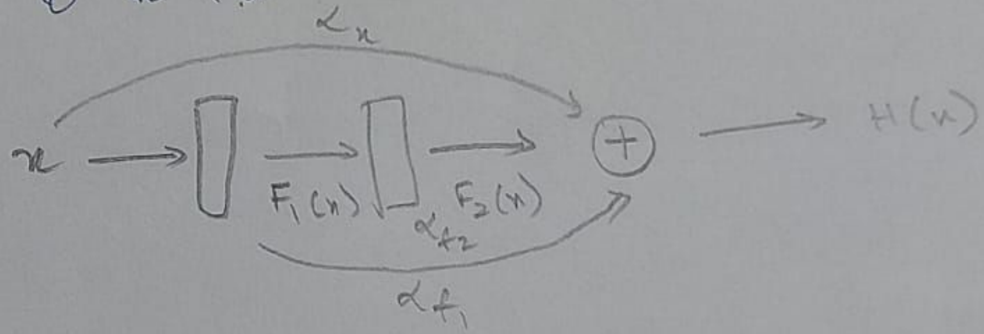
6. Output size after 7×7 filter conv will be smaller than for 3×3 assuming $S=1$, $P=0$.
Max pool layer during backprop only propogates gradients to the cells which were maximum in their neighbourhood.
For 7×7, we get 1 cell where grad is propogated for every 49 neighbourhood whereas for 3×3 it is 1 in every 9. 4.
Hence gradients are propogated in more cells for 3×3 than for 7×7 and hence it can get trained faster.
So, 3×3 will be better.

7\. In Residual Network, we hardcode the connection as,

$H(x) = F(x) + x$



To prevent this hardcoding we can make it adaptive by simply making $H(x)$ as a linear combination of each layers outputs. Weights are fixed to be between 0 to 1 and are learnt.



$$H(x) = \alpha_x \odot x + \alpha_{f_1} \odot F_1(x) + \alpha_{f_2} \odot F_2(x)$$

Hence by learning the αs, we can control the flow of information from all layers to $H(x)$ and not just x and $F(x)$ like before.

If some layer output does not contribute, it's α will goto 0 while training.

This is more adaptive.

9. $E\left((y-\hat{f}(x))^2\right) = E\left((f(x)+\epsilon-\hat{f}(x))^2\right)$

$= E\left((f(x)-E(\hat{f}(x)) + \epsilon + E(\hat{f}(x)) - \hat{f}(x))^2\right)$

$= E\left((f(x)-E(\hat{f}(x)))^2\right) + E(\epsilon^2) + E\left((E(\hat{f}(x))-\hat{f}(x))^2\right)$

$+ 2E\left(\epsilon(f(x)-E(\hat{f}(x)))\right) + 2E\left(\epsilon(E(\hat{f}(x))-\hat{f}(x))\right)$

$+ 2E\left((f(x)-E(\hat{f}(x)))(E(\hat{f}(x))-\hat{f}(x))\right)$

The 2E terms goto 0 as ϵ, $\hat{f}(x)$ are all independent. Covariance = 0. Also, $E(\epsilon^2) = Var(\epsilon)$, $E\left((E(\hat{f}(x))-\hat{f}(x))^2\right) = Var(\hat{f}(x))$

$\Rightarrow (f(x)-E(\hat{f}(x)))^2 + Var(\epsilon) + Var(\hat{f}(x))$

First term is bias of $\hat{f}(x)$, $Var(\epsilon)$ is irreducible error

$\Rightarrow (Bias)^2 + \sigma^2 (\text{irreducible}) + \text{Variance}$ //

10. Feed forward layer is used in encoder in transformer

- Add more model complexity
- Convert attention layer output to size as expected by next layer / decoder
- Add Non-linearity

If we dont use feed forward layer, we will have to feed the output of attention layer to decoder. This can model only linear functions as attention is linear and hence the model cannot perform well for real life data and tasks with lot of non-linearity.

11. To ensure that only attentions of words inside the window are aggregated for a time step 'p' in decoder, we can use a mask vector specific to time step p which is point wise multiplied with the attention outputs from encoder and then aggregated.

$mask_p$ is a vector of size M and has all 0s except at the indices p-k to p+k.

$$mask_p = \begin{array}{l} \left[\begin{array}{ccccccccc} 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots 0 \end{array} \right] \\ \begin{array}{ccccccccc} \text{Index } 1 & 2 & \cdots & p\text{-}k\text{-}1 & p\text{-}k & & p+k & p+k+1 & M \end{array} \end{array}$$

$$\therefore \; c_p = \cancel{mask_p \odot \alpha_p} \; mask_p \, \alpha_p$$

12. Identifying human-animal interactions on college campuses. To detect if someone is feeding the animals, etc.

Input : Image from CCTV

Output : Detect event.

Neural Net : CNN arch.