Anil Poonai

Github link for files and code: https://github.com/DevonARP/DeepLearning_A3

Problem 1: Part A

CNN's are better at identifying images as each layer in it's architecture helps extract prominent features to figure out what the object in the image is.

They are also really good at reducing the dimensions of the data, this is also caused by the layers filtering the image but it significantly reduces the computations and time needed to classify an image.

Problem 1: Part B

RNN's are typically better at dealing with temporal or sequential data, in this regard if we're dealing with a video, which is just sequential images at different points of time, an RNN would be preferred.

Another time an RNN would be preferred is if we want to understand what's happening in the image, in this regard something like grabbing sentiment from the image for a caption or description would make an RNN ideal.

Problem 2

Hidden state: $H_t = x_t - h_{t-1}$

Number of inputs: n

So lets start 2n inputs, this keeps the amount of the inputs even:

$h_{2n} = x_{2n} - h_{2n-1}$

$h_{2n} = x_{2n} - x_{2n-1} + h_{2n-2}$

$h_{2n} = x_{2n} - x_{2n-1} + x_{2n-2} - h_{2n-3}$

Following the alternating sign pattern:

$h_{2n} = x_{2n} - x_{2n-1} + x_{2n-2} \ldots - x_3 + x_2 - x_1 + h_0$

I'm attaching the work I manually did below this question. I used an example with 4 inputs of consecutive numbers. I also did the example above as well.

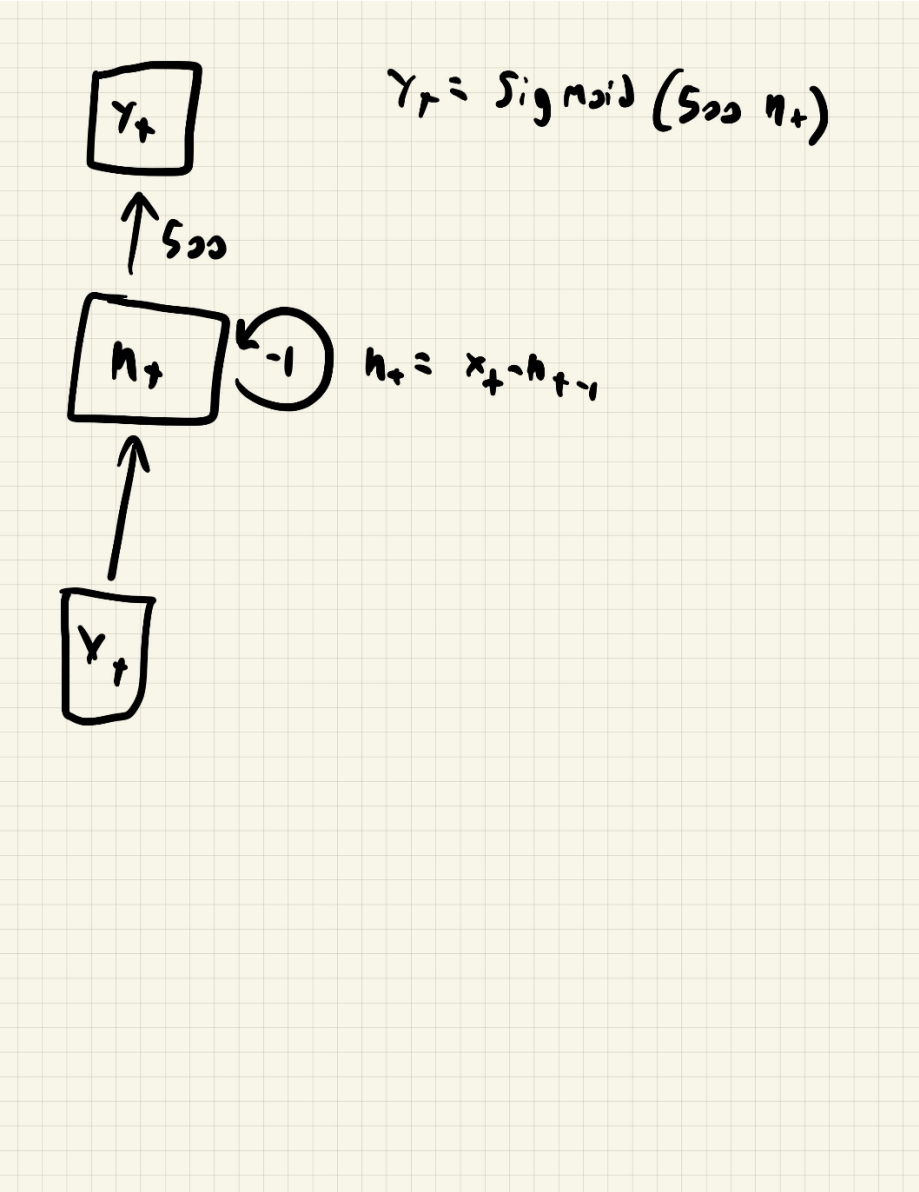
Pattern noticed:

- If the input order number is odd, the input value will end up being a negative value in the equation
- If the input order number is even, the input value will end up being a positive value in the equation

- If the values of the inputs are consecutive and the total inputs is still an even length, then the final value would be $n/2 + h_0$

Output: $y_t = \text{sigmoid}(500*h_{2n})$

$y_t = \text{sigmoid}(500*(x_{2n} - x_{2n-1} + x_{2n-2} \ldots - x_3 + x_2 - x_1 + h_0))$

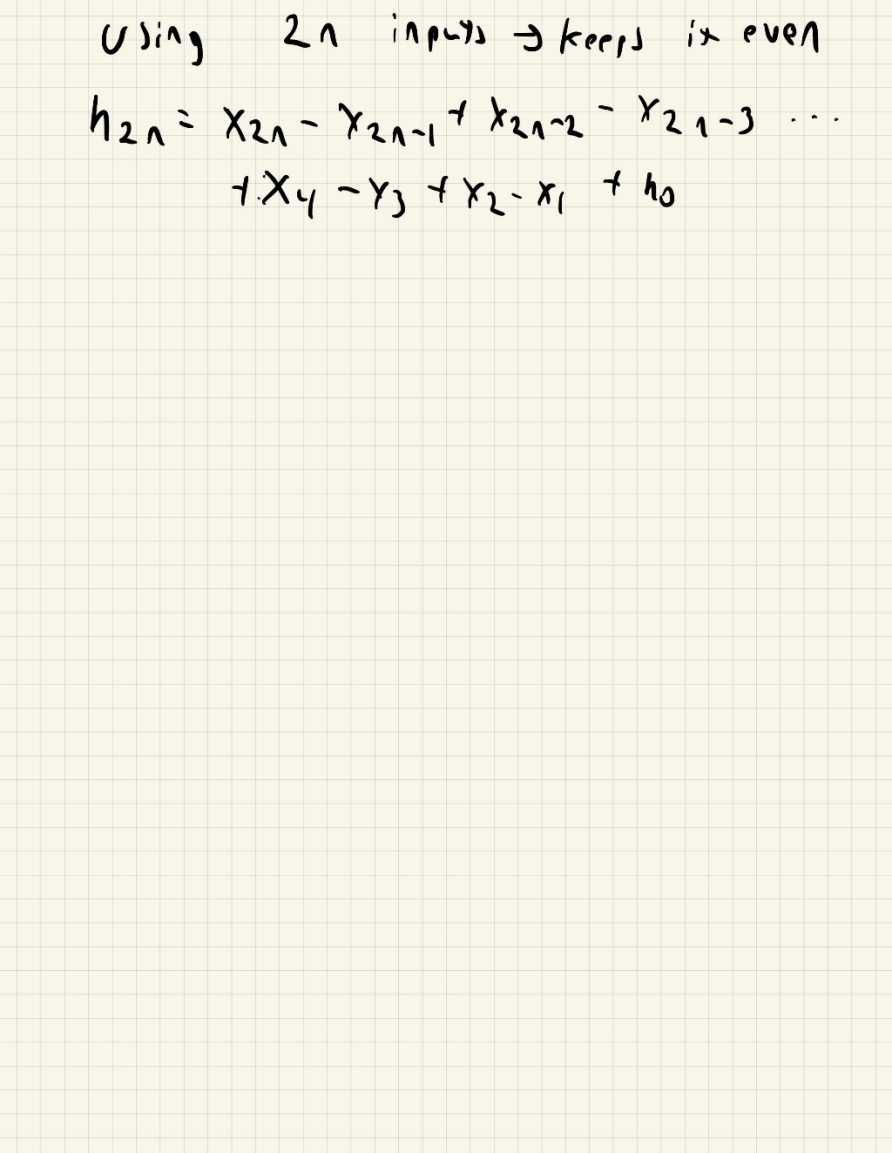


Example with 4 inputs [1,2,3,4]

$$h_4 = x_4 - h_3$$
$$= x_4 - (x_3 - h_2)$$
$$= x_4 + h_2 - x_3$$
$$= x_4 + (x_2 - h_1) - x_3$$
$$= x_4 + x_2 - (x_1 - h_0) - x_3$$
$$= x_4 + x_2 + h_0 - x_1 - x_3$$
$$= 4 + 2 + h_0 - 1 - 3$$
$$= 2 + h_0$$

- if inputs are consecutive $h_1 = \frac{1}{2} + h_0$
- all odd input orders will be subtracted & all even ones will be added

$$y_t = \text{sigmoid}\left(f_{20}\left(x_4 + x_2 + h_0 - x_1 - x_3\right)\right)$$



Problem 3

We can use convolutions attention model to reduce the time complexity to O(T), it would actually be O(T*k) where k is the kernel size, that dimension reduction plus the stride change would make this immensely faster. The down side to this is that it becomes more of a local receptive field that the network grabs information from which is more of a convolutional way of going about this. It doesn't allow context to be taken into account, which is what attention is meant to involve, so it misses out on the points it should be paying attention to in the data.

We can also do something similar with recurrent networks, by combining both of these. It would be faster as it would become O(T) it would also lose the attention being given to the important parts as all of the hidden states wouldn't pass through.

References:

https://arxiv.org/pdf/2111.14556.pdf

https://www.researchgate.net/publication/347999026_A_Transformer_Self-Attention_Model_for_Time_Series_Forecasting

https://stackoverflow.com/questions/65703260/computational-complexity-of-self-attention-in-the-transformer-model

https://www.youtube.com/watch?v=fjJOgb-E41w

Problem 4 Part A

Code is attached

Problem 4 Part B

Code is attached

Problem 4 Part C

I did 4 different examples, 2 are on the Fashion dataset and 2 are on MNIST. They each have one Conditional GAN and one GAN being used on the data. The key difference I found is the randomness of what's generated, the Conditional GAN would be consistent and specify what type of object or number it would generate while the regular GAN would not.