# UE21CS343BB2
# Topics in Deep Learning

**Dr. Shylaja S S**
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
**shylaja.sharath@pes.edu**

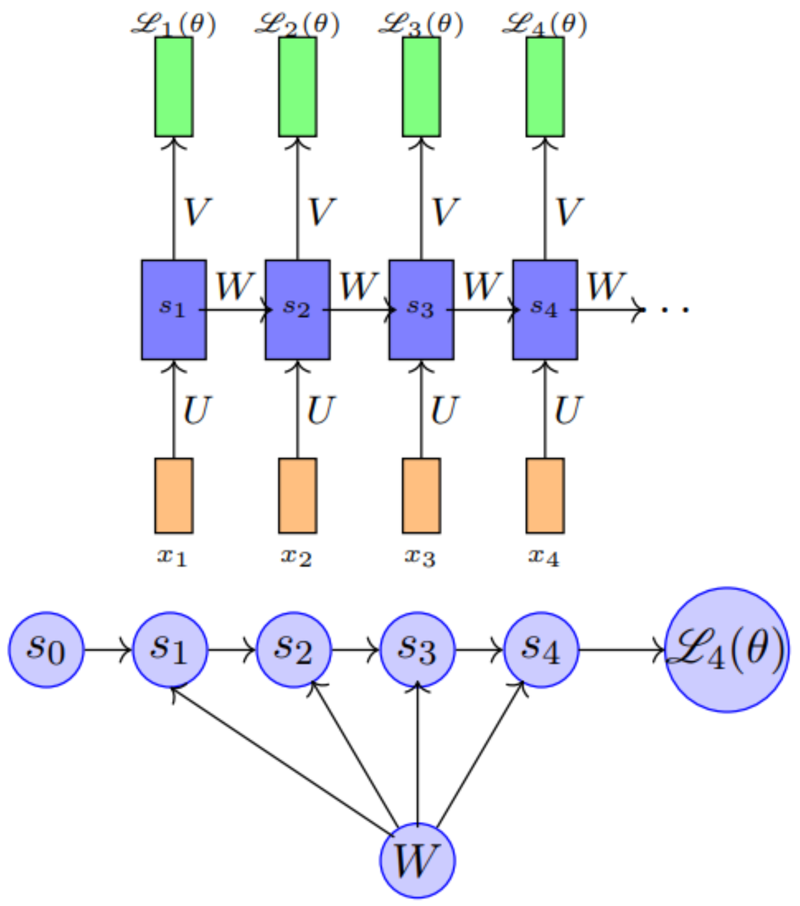
**Ack:Anirudh Chandrasekar, Teaching Assistant**

# Topics in Deep Learning

## RNN - Truncated Backpropagation Through Time

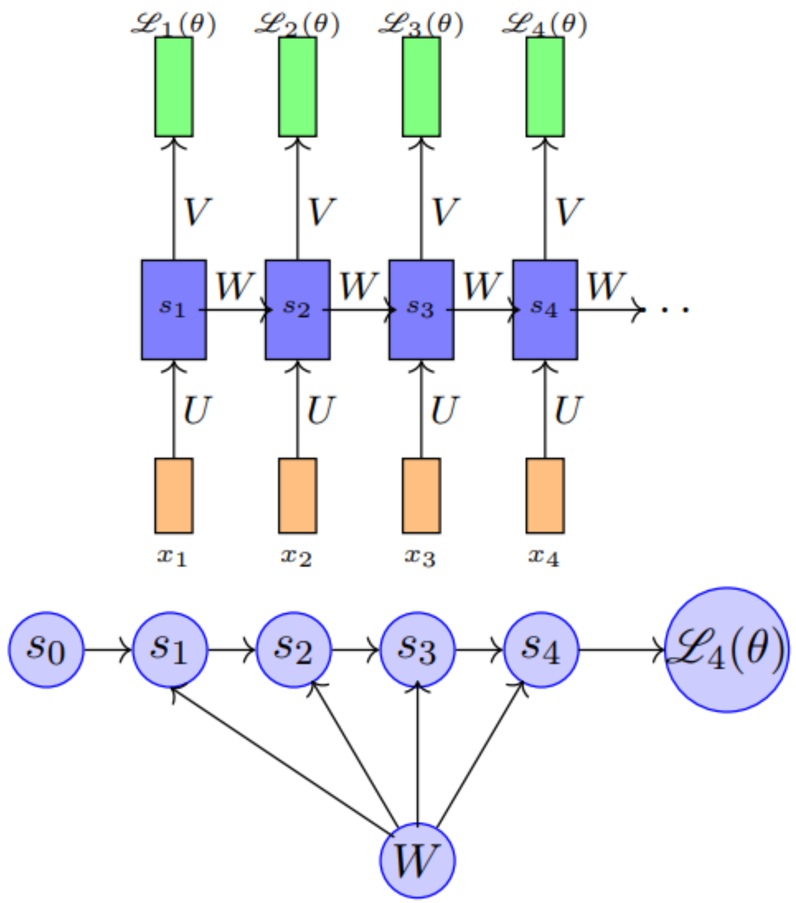
## Backpropagation Through Time

- We have seen how to perform Backpropagation Through Time in our previous lecture.
- We go n timesteps forward in the forward propagation and then perform backpropagation through all the n timesteps.
- But what if the input sequence is very long causing n to be large.



## Truncated Backpropagation Through Time

- But what if the input sequence is very long causing n to be large.
- It becomes expensive to calculate the backpropagation in terms of space since in order to backpropagate we need to save the activations in every timestep.
- This could lead to us running out of physical space.



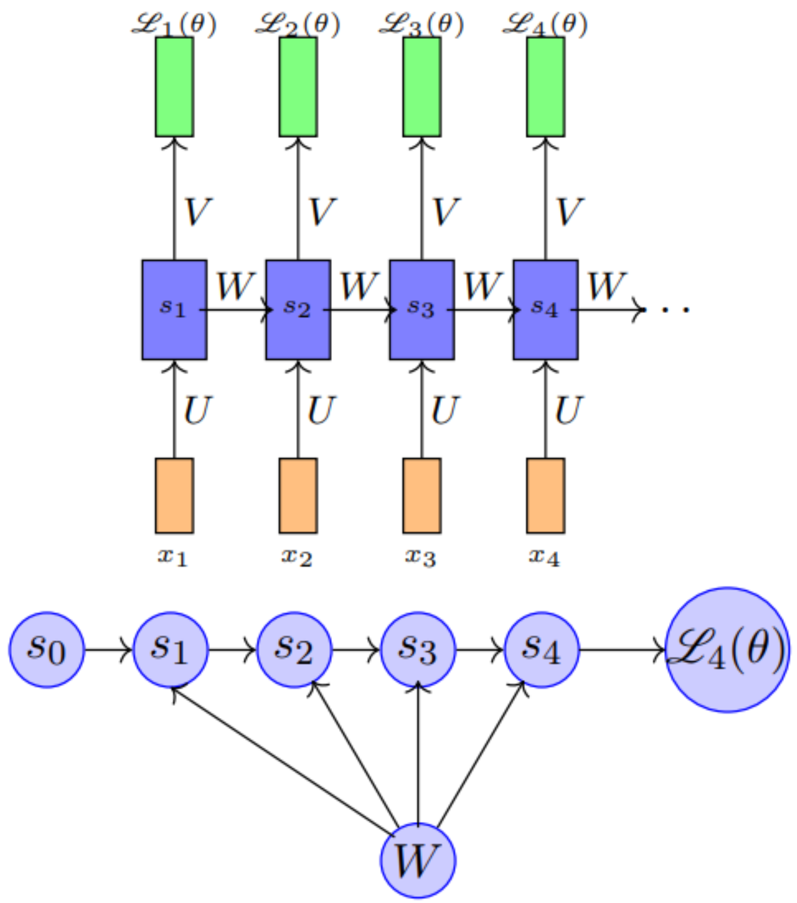
## Truncated Backpropagation Through Time

- This is where Truncated Backpropagation Through Time comes in.

Truncated BPTT defines two constants $k_1$ and $k_2$ such that

$$1 <= k_1 <= k_2 <= n$$

where n is the sequence length.



## Truncated Backpropagation Through Time

- BPTT(n,n) : This is nothing but the traditional BPTT where we move n timesteps forward followed by n timesteps back.
- BPTT(1,n): In this approach we move 1 timestep in the forward pass followed by n timesteps back.
    - Example: At t=1 we move 1 time step forward and then n timesteps back, at t=2 we move another time step forward and n timesteps back(here 2 as we backpropagate all the way to the first time step) and so on.
    - So we move forward through n time steps in total and backpropagate n(n+1) i.e $n^2$ times.

## Truncated Backpropagation Through Time

- BPTT(1,n): In this approach we move 1 timestep in the forward pass followed by n timesteps back.
    - The advantage of this approach is that there is a lot of updating taking place in the early gradients i.e in the early timesteps we are updating the weight matrices and over emphasising on the early timesteps.

## Truncated Backpropagation Through Time

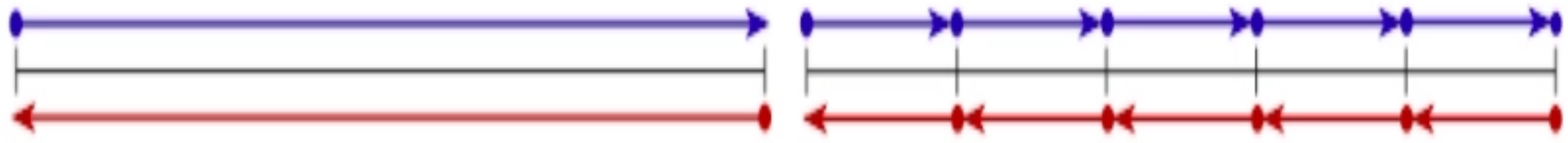
- BPTT(k,k): In this approach we move k timestep in the forward pass followed by k timesteps back.
  - Example: At t=1 we move k time step forward and then k timesteps back, at t=2 we move another k timesteps forward and k timesteps back and till the end of the input sequence.
  - The last timestep need not be (k,k) and this can be used when n is not a multiple of k as well.
  - The advantage of this approach is that we need to save only k activations as compared to all the activations in the previous two approaches.

## Truncated Backpropagation Through Time

- BPTT(k,2k): In this approach we move k timestep in the forward pass followed by 2k timesteps back.
  - In BPTT(k,k) we are not backpropagating over certain timesteps at all.
  - To avoid such a scenario BPTT(k,2k) can be used to ensure all the timesteps are backpropagated over.

Note: PyTorch implements only BPTT(n,n) and BPTT(k,k). Any other combination of $k_1$ and $k_2$ can be implemented manually.

## Truncated Backpropagation Through Time



(a) BPTT (b) Truncated BPTT

## Preparing Sequence Data for Truncated BPTT

- Use Data As-Is

  If the number of timesteps in each sequence is small, such as tens or a few hundred, you can utilize the input sequences as-is. TBPTT has been suggested to have practical limitations of 200 to 400 timesteps. You can reshape the sequence observations as timesteps for the input data if your sequence data is smaller than or equal to this range.

  For example, if you had a collection of 100 univariate sequences with 25 timesteps, you could reshape it into 100 samples, 25 timesteps, and 1 feature, or [100, 25, 1].

## Preparing Sequence Data for Truncated BPTT

- Naive Data Split

  If your input sequences are large, such as hundreds of timesteps, you may need to divide them up into many contiguous subsequences.

  This will need the implementation of a stateful LSTM in Keras, with the internal state being retained across sub-sequence input and only being reset at the conclusion of a true fuller input sequence.

## Preparing Sequence Data for Truncated BPTT

- Naive Data Split

  If you had 100 input sequences with 50,000 timesteps, for example, each one might be broken into 100 subsequences with 500 timesteps. One input sequence would provide 100 samples, resulting in a total of 10,000 original samples. Keras' input would have a dimensionality of 10,000 samples, 500 timesteps, and 1 feature, or [10000, 500, 1]. It would be necessary to take care to preserve the state throughout every 100 subsequences and to explicitly or implicitly reset the internal state after every 100 samples.

## Preparing Sequence Data for Truncated BPTT

- Domain-Specific Data Split

Knowing the proper number of timesteps to generate a useful estimate of the error gradient can be difficult.

We can generate a model rapidly using the naïve technique, but the model may not be optimal. Alternatively, while learning the issue, we may utilize domain-specific knowledge to predict the number of timesteps that will be important to the model.

## Preparing Sequence Data for Truncated BPTT

- Systematic Data Split

  You can systematically examine a suite of possible subsequence lengths for your sequence prediction challenge rather than guessing at a reasonable number of timesteps.

  You might do a grid search over each sub-sequence length and pick the arrangement that produces the best overall model.

## Acknowledgements & References

- https://deeplearning.ai
- https://youtu.be/CrMgI3OiNnw?si=pQrpZJUFWMhH-pSK
- https://www.codingninjas.com/studio/library/truncated-bptt

# UE21CS343BB2
# Topics in Deep Learning

**Dr. Shylaja S S**
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
**shylaja.sharath@pes.edu**

Ack:Anirudh Chandrasekar,
Teaching Assistant