

## Chatbot

- ✓ **Video:** Tasks with Long Sequences  
2 min
- ✓ **Reading:** Tasks with Long Sequences  
10 min
- ✓ **Reading:** Optional AI Storytelling  
15 min
- ✓ **Video:** Transformer Complexity  
3 min
- ✓ **Reading:** Transformer Complexity  
10 min
- ✓ **Video:** LSH Attention  
4 min
- ✓ **Reading:** LSH Attention  
10 min
- ✓ **Reading:** Optional KNN & LSH Review  
20 min
- 📅 **Lab:** Ungraded Lab: Reformer LSH  
1h
- ✓ **Video:** Motivation for Reversible Layers: Memory!  
2 min
- ✓ **Reading:** Motivation for Reversible Layers: Memory!  
10 min
- ✓ **Video:** Reversible Residual Layers  
5 min
- 📖 **Reading:** Reversible Residual Layers  
10 min
- 📅 **Lab:** Ungraded Lab: Revnet  
1h
- 🎥 **Video:** Reformer  
2 min
- 📖 **Reading:** Reformer  
10 min
- 📖 **Reading:** Optional Transformers beyond NLP  
20 min
- 📖 **Reading:** Acknowledgments  
10 min

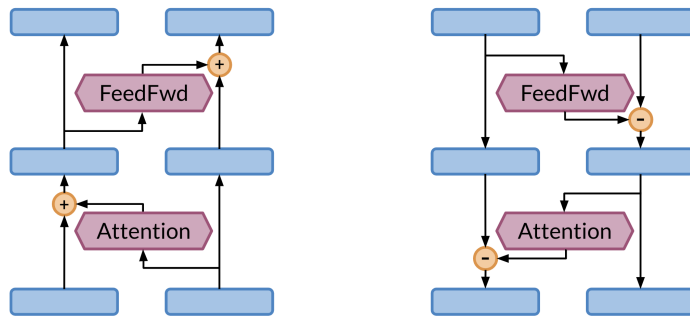
## Heroes of NLP: Quoc Le

### Assignment

### Course Resources

# Reversible Residual Layers

Reversible residual layers allow you to reconstruct the forward layer from the end of the network. Usually you have two similar branches in the network that you use to compute the network.



In the left picture, you have the forward propagation. One side of the network is used as input and the other is used for the attention. In the left side, the same thing is happening but in the opposite direction.

Standard Transformer:

$$y_a = x + \text{Attention}(x)$$

$$y_b = y_a + \text{FeedFwd}(y_a)$$

Reversible:

$$y_1 = x_1 + \text{Attention}(x_2)$$

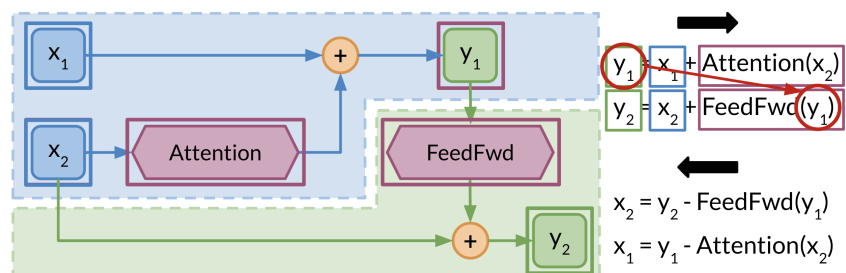
$$y_2 = x_2 + \text{FeedFwd}(y_1)$$

Recompute  $x_1, x_2$  from  $y_1, y_2$ :

$$x_1 = y_1 - \text{Attention}(x_2)$$

$$x_2 = y_2 - \text{FeedFwd}(y_1)$$

Take a few minutes and try to understand the equations above. You basically make use of the two branches of the network. When coming back for the back propagation, you only need the  $y$ 's to compute  $x_2$  and then you can use  $x_2$  along with  $y_1$  to compute  $x_1$ . Pretty neat! Now you don't have to store the weights, because you can just compute them from scratch. This image shows you a visualization of what is happening.



Mark as completed