Deep RNN

Deep Recurrent Neural Networks (RNNs) are neural network architectures with multiple hidden layers that are designed to handle sequential data by maintaining an internal memory of previous inputs. These networks pass the output from one layer of recurrent units to the next layer, allowing them to capture complex relationships between input and output sequences

Deep RNNs are used in various real-life applications such as speech recognition systems, language translation software, and self-driving cars, particularly where there is a need to process a lot of sequential data like human language understanding

In the context of deep RNNs, the notion of depth refers to stacking multiple layers of RNNs on top of each other to increase the network's ability to model complex relationships in sequential data. This stacking of layers allows the network to capture data at different levels, enabling it to retain high-level information while also recording shorter-term temporal dynamics

The training of deep RNNs involves considerations like learning rate and clipping to ensure proper convergence

Deep RNNs have been successfully applied in various fields such as natural language processing, speech recognition, image captioning, and music generation. These networks have shown significant performance improvements compared to single-layer RNNs or shallow neural networks, making them a powerful tool in handling sequential data and capturing intricate relationships within it.

Here's a simple intuition of how a deep RNN works and the problem it solves:

1. **Sequential Data Processing**: Imagine you're analyzing a sentence, where each word is a sequential input. A deep RNN processes each word in the sequence, one at a time, while maintaining a memory of previous words it has seen. This memory allows it to understand the context of each word within the entire sentence.
2. **Hierarchical Feature Representation**: Each layer in the deep RNN captures different levels of abstraction or features. The lower layers might capture basic features like individual words, while higher layers combine these features to capture more abstract concepts or contexts. This hierarchical representation enables the network to understand complex patterns in the data.
3. **Long-Term Dependencies**: One key problem with basic RNNs is the vanishing or exploding gradient problem, which limits their ability to capture long-term dependencies in sequential data. Deep RNNs help alleviate this problem by introducing multiple layers, allowing the network to learn more complex and abstract representations of the data, which can span longer sequences.
4. **Improved Performance**: By stacking multiple layers of recurrent units, deep RNNs can learn more complex functions and achieve better performance on tasks like language modeling, machine translation, and sentiment analysis. The

additional depth allows the network to learn hierarchical representations of the input data, leading to more accurate predictions.

In essence, a deep RNN tackles the challenge of processing sequential data by building a hierarchical representation of the input, enabling it to capture long-term dependencies and extract meaningful patterns from the data. This makes it a powerful tool for a wide range of tasks in natural language processing, time series analysis, and beyond.

An RNN is deep with respect to time. But what if it's deep with respect to space as well, as in a feed-forward network? This is the fundamental notion that has inspired researchers to explore Deep Recurrent Neural Networks, or Deep RNNs

In a typical deep RNN, the looping operation is expanded to multiple hidden units.

An RNN can also be made deep by introducing depth to a hidden unit.

```
# Define the RNN model
model = Sequential([
    Embedding(10000, 32, input_length=100), # Embedding layer to convert words to vectors
    SimpleRNN(5, return_sequences=True),    # RNN layer with 5 units
    SimpleRNN(5),                           # Another RNN layer with 5 units
    Dense(1, activation='sigmoid')          # Output layer for binary classification
])

model.summary()
```

Esehie Hum

GRU lele then Deep GRU bnnjayega

&

LSTM lele then Deep LSTM bnnjayega

Disadvantages:

1. ***Ovefitting***
2. ***Model Training time increases, Specially on large dataset.***