"How Do Recurrent Neural Networks Remember? A Journey Through Time"

Abstract:

A Recurrent Neural Network (RNN) is a class of neural networks designed for sequential data, where each output depends on both current input and past information. It captures temporal dependencies through internal memory, making it ideal for tasks like time series forecasting, speech recognition, and natural language processing.

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data by maintaining an internal *hidden state* that captures information about previous inputs. This hidden state acts as the network's *memory*, allowing it to learn temporal dependencies through time steps.

Concept:

Recurrent Neural Networks (RNNs) differ from regular neural networks in how they process information. While standard neural networks pass information in one direction i.e. from input to output, RNNs feed information back into the network at each step.
Unlike traditional feedforward networks, RNNs loop information within their architecture, allowing previous outputs to influence future predictions. This feedback mechanism helps the network remember context over time, though it can struggle with long sequences due to vanishing or exploding gradients.

Imagine reading a sentence and you try to predict the next word, you don't rely only on the current word but also remember the words that came before. RNNs work similarly by "remembering" past information and passing the output from one step as input to the next i.e it considers all the earlier words to choose the most likely next word. This memory of previous steps helps the network understand context and make better predictions.

Types Of Recurrent Neural Networks

There are four types of RNNs based on the number of inputs and outputs in the network:

1. One-to-One RNN

This is the simplest type of neural network architecture where there is a single input and a single output. It is used for straightforward classification tasks such as binary classification where no sequential data is involved.

2. One-to-Many RNN

In a One-to-Many RNN the network processes a single input to produce multiple outputs over time. This is useful in tasks where one input triggers a sequence of predictions (outputs). For example in image captioning a single image can be used as input to generate a sequence of words as a caption.

3. Many-to-One RNN

The Many-to-One RNN receives a sequence of inputs and generates a single output. This type is useful when the overall context of the input sequence is needed to make one prediction. In sentiment analysis the model receives a sequence of words (like a sentence) and produces a single output like positive, negative or neutral.

4. Many-to-Many RNN

The Many-to-Many RNN type processes a sequence of inputs and generates a sequence of outputs. In language translation task a sequence of words in one language is given as input and a corresponding sequence in another language is generated as output

Python Implementation using Tensorflow:

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(seq_len, 1)),
    tf.keras.layers.SimpleRNN(64, activation='tanh'),
    tf.keras.layers.Dense(1)
])
```

Applications

RNNs are used in various applications where data is sequential or time-based:

- **Time-Series Prediction**: RNNs excel in forecasting tasks, such as stock market predictions and weather forecasting.
- **Natural Language Processing (NLP)**: RNNs are fundamental in NLP tasks like language modeling, sentiment analysis and machine translation.
- **Speech Recognition**: RNNs capture temporal patterns in speech data, aiding in speech-to-text and other audio-related applications.
- **Image and Video Processing**: When combined with convolutional layers, RNNs help analyze video sequences, facial expressions and gesture recognition.

References:

GeeksforGeeks. (2025, October 07). *Introduction to Recurrent Neural Networks*. https://www.geeksforgeeks.org/machine-learning/introduction-to-recurrent-neural-network/