**Introduction**

Recurrent Neural Networks handle **sequence data** to predict the next event. To understand the need for this model, let’s start with a thought experiment.

**RNN** architecture is a chain of **memory cells** distributed over time steps t.

A diagram of a flowchart

AI-generated content may be incorrect.

**Unfolded RNN architecture**

A diagram of a diagram

AI-generated content may be incorrect.

**Memory cell**

Each of the **memory cells** takes 2 inputs:

* **xt**— input (e.g. word in the sentence)
* **ht-1**— hidden state (contains information about context from the previous cell)

And returns 2 values:

* **ht** — next hidden state
* **yt**— prediction (e.g. prediction of next word in the sentence)

**Mathematical Operations behind RNN:**

**A white paper with writing on it

AI-generated content may be incorrect.**

**Let’s create RNN**

The **hidden unit’s** variable is the number of “neurons” in the hidden state. One time step can be loosely analogous to the layer in a neural network.

**Dataset**

Our dataset is **a sinusoid function** turned into 200 samples each of 25-time steps.

**Forward pass**

Forward pass involves looping over **every step** in the sample to get the final output **yt**and calculate **MSE** loss.

**Backpropagation**

This is where things are getting a bit tricky. In RNN, weights are shared across the time steps, so weight **Wh** depends on each hidden state back in time. Analogous weight **Wx**depends on each input from the previous time step. That’s why we call it backpropagation through time(**BPTT**).

We also limit the gradients to prevent **vanishing** and **exploding** gradient problems.

**Let’s pull it all together**

So we build our code for forward and backward propagation, now we can feed our network with the data and see how it behaves.

**Improvements**

Since it is a vanilla implementation of RNN, there are plenty of upgrades:

* Adding bias to the hidden state
* Truncated back propagation through time (TBPTT) — for longer sequences
* More advanced models like LSTM or GRU

**Conclusions**

Richard Feynman’s famous quote is, “What I cannot create, I do not understand.” That’s why I think it is essential to build these models from scratch to gain a more profound understanding of mathematics inside, problems related to the task, and how they are solved.