Simple RNN

In [2]:

```python
import tensorflow as tf
import numpy as np

# Create toy dataset
data = np.array([1.0, 0.5, 0.7, 0.3, 0.2])
data = data.reshape(1, -1, 1)  # (batch_size, sequence_length, input_dimens

# Define the RNN model
model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(units=1, activation='tanh', return_sequences=
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(data, data, epochs=1000)

# Predict using the trained model
predicted = model.predict(data)

print("Original data:", data)
print("Predicted data:", predicted)
```

```
Epoch 997/1000
1/1 [==============================] - 0s 11ms/step - loss: 0.0118
Epoch 998/1000
1/1 [==============================] - 0s 10ms/step - loss: 0.0118
Epoch 999/1000
1/1 [==============================] - 0s 10ms/step - loss: 0.0118
Epoch 1000/1000
1/1 [==============================] - 0s 11ms/step - loss: 0.0118
1/1 [==============================] - 0s 359ms/step
Original data: [[[1. ]
  [0.5]
  [0.7]
  [0.3]
  [0.2]]]
Predicted data: [[[0.950798  ]
  [0.4570682 ]
  [0.8003449 ]
  [0.2640567 ]
  [0.40774655]]]
```

Deep RNN

In [3]:

```
import tensorflow as tf
import numpy as np

# Create toy dataset
data = np.array([1.0, 0.5, 0.7, 0.3, 0.2])
data = data.reshape(1, -1, 1)  # (batch_size, sequence_length, input_dimens

# Define the Deep RNN model
model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(units=1, activation='tanh', return_sequences=
    tf.keras.layers.SimpleRNN(units=1, activation='tanh', return_sequences=
    tf.keras.layers.SimpleRNN(units=1, activation='tanh', return_sequences=
    tf.keras.layers.SimpleRNN(units=1, activation='tanh', return_sequences=
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(data, data, epochs=1000)

# Predict using the trained model
predicted = model.predict(data)

print("Original data:", data)
print("Predicted data:", predicted)
```

```
Epoch 997/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0578
Epoch 998/1000
1/1 [==============================] - 0s 13ms/step - loss: 0.0577
Epoch 999/1000
1/1 [==============================] - 0s 15ms/step - loss: 0.0577
Epoch 1000/1000
1/1 [==============================] - 0s 14ms/step - loss: 0.0577
1/1 [==============================] - 1s 854ms/step
Original data: [[[1. ]
  [0.5]
  [0.7]
  [0.3]
  [0.2]]]
Predicted data: [[[0.67927617]
  [0.37597153]
  [0.6209335 ]
  [0.45615074]
  [0.57362837]]]
```

In [ ]: