Build RNN for deep learning | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=uplfqlwrfkWn

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More resources
- Featured examples
- + Section

[9] ✓ 0s

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense
from sklearn.model_selection import train_test_split
```

[11] ✓ 0s

```
# Generate sine wave data
x = np.linspace(0, 100, 1000)
y = np.sin(x)

# Normalize
y = (y - y.min()) / (y.max() - y.min())

# Prepare sequences
timesteps = 10
X, Y = [], []

for i in range(len(y) - timesteps):
    X.append(y[i:i+timesteps])
    Y.append(y[i+timesteps])

X = np.array(X)
Y = np.array(Y)

# Reshape for RNN input: (samples, timesteps, features)
X = X.reshape(X.shape[0], X.shape[1], 1)

# Split data
```

Variables | Terminal | ✓ 22:39 | Python 3

Build RNN for deep learning | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=uplfqlwrfkWn

Gmail Maps News Translate Disney+ Hotstar -... YouTube All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More resources
- Featured examples
- + Section

[14] ✓ 0s

```
loss = model.evaluate(X_test, y_test)
print(f"\nTest Loss (MSE): {loss:.4f}")
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 1.7237e-05

Test Loss (MSE): 0.0000
```

[15] ✓ 0s

```
y_pred = model.predict(X_test)
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```

[16] ✓ 2s

```
plt.figure(figsize=(10,5))
plt.plot(y_test, label='Actual')
plt.plot(y_pred, label='Predicted', linestyle='--')
plt.title('RNN Sequence Prediction (Sine Wave)')
plt.xlabel('Time Steps')
plt.ylabel('Normalized Value')
plt.legend()
plt.show()

# Plot training loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Training Performance')
plt.xlabel('Epoch')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.show()
```

Variables Terminal

22:39 Python 3

Build RNN for deep learning | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=upIfqlwrfkWn

Gmail Maps News Translate Disney+ Hotstar -... YouTube All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
  - Featured examples
- + Section

[11] ✓ 0s

[12] ✓ 0s

```
model = Sequential([
    SimpleRNN(50, activation='tanh', input_shape=(timesteps, 1)),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.summary()
```

```
/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
```

[13] ✓ 11s

```
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2, verbose=1)
```

```
Epoch 1/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 0.1528 - val_loss: 0.0235
Epoch 2/50
```

Variables Terminal

✓ 22:39 Python 3

Build RNN for deep learning | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=upIfqlwrfkWn

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

```
Epoch 1/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 0.1528 - val_loss: 0.0235
Epoch 2/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0185 - val_loss: 0.0067
Epoch 3/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0039 - val_loss: 0.0020
Epoch 4/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0020 - val_loss: 0.0015
Epoch 5/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0015 - val_loss: 0.0013
Epoch 6/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0012 - val_loss: 0.0011
Epoch 7/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0010 - val_loss: 8.3107e-04
Epoch 8/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 8.0911e-04 - val_loss: 6.4882e-04
Epoch 9/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 6.8500e-04 - val_loss: 5.7329e-04
Epoch 10/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.2483e-04 - val_loss: 4.1876e-04
Epoch 11/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 3.9495e-04 - val_loss: 3.5078e-04
Epoch 12/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.8169e-04 - val_loss: 2.6994e-04
Epoch 13/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 2.7757e-04 - val_loss: 2.3912e-04
Epoch 14/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.2805e-04 - val_loss: 2.1247e-04
Epoch 15/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.8258e-04 - val_loss: 1.6565e-04
Epoch 16/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.3742e-04 - val_loss: 2.1505e-04
Epoch 17/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.2711e-04 - val_loss: 9.5978e-05
```

Variables | Terminal | 22:39 | Python 3

Build RNN for deep learning | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=upIfqlwrfkWn

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Run all cells in notebook

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.5021e-05 - val_loss: 7.9243e-06
Epoch 35/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 8.7559e-06 - val_loss: 7.7689e-06
Epoch 36/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 7.9123e-06 - val_loss: 7.4062e-06
Epoch 37/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 7.4976e-06 - val_loss: 8.7286e-06
Epoch 38/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 9.7619e-06 - val_loss: 8.2764e-06
Epoch 39/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.0049e-05 - val_loss: 1.7283e-05
Epoch 40/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 8.5267e-06 - val_loss: 6.3656e-06
Epoch 41/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.2620e-06 - val_loss: 9.2408e-06
Epoch 42/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 7.3299e-06 - val_loss: 5.7646e-06
Epoch 43/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 7.2817e-06 - val_loss: 6.6464e-06
Epoch 44/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 5.2611e-06 - val_loss: 5.2855e-06
Epoch 45/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 5.4060e-06 - val_loss: 5.2684e-06
Epoch 46/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 1.5078e-05 - val_loss: 4.5510e-06
Epoch 47/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 1.1490e-05 - val_loss: 1.5243e-05
Epoch 48/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.9822e-05 - val_loss: 1.7374e-05
Epoch 49/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.7315e-05 - val_loss: 5.1728e-06
Epoch 50/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.4227e-06 - val_loss: 1.9773e-05
```

Variables | Terminal | 22:39 | Python 3

ex:-9 Build a Recurrent Neural Network for sequence prediction using a Recurrent Neural Network.

Aim:- To design and implement a Recurrent Neural Network that can learn temporal (time-based) patterns from sequential data and make accurate predictions based on those patterns.

objectives:-

1. To understant the architecture and working of RNN
2. To preprocess sequential data for RNN training.
3. To implement an RNN model using a deep learning framework.
4. to Train the RNN model on sequence data.
5. To evaluate the performance of the model and visualize results

Pseodocode:-

1. Import libraries.

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense
```

2. Load and preprocess data
   - Generate or load sequential data (eg Sine wave).
   - Normalize data
   - create input sequences and labels.

3. Define model

```
model = Sequential()
model.add(SimpleRNN(unit=50; activation='tanh', input_sh
        = (timesteps; features)))
model.add(Dense())
```

4. Train model.

```
history = model.fit(x_train; y_train, epochs=50, batch_s
        =32, validation_data = (x_val, y_val))
```

5. Evaluate model

```
test_loss = model.evaluate(x_test, x_test)
```

6. predict

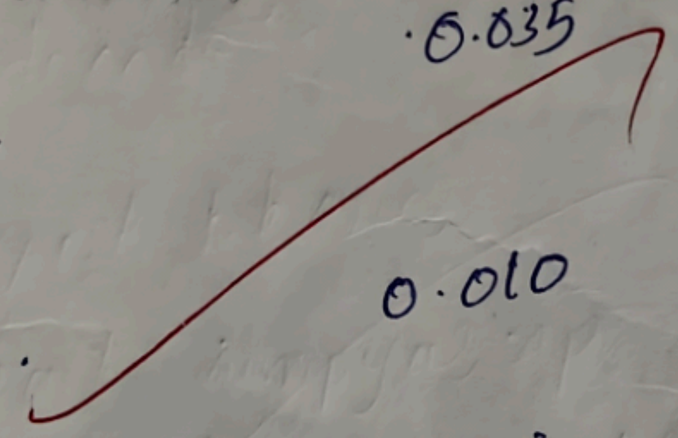
```
y_pred = model.predict(x_test)
```

7. visualize results

plot y_test vs y_pred to observe prediction performance.

observation :-

Epoch	Training loss	Validation loss	observation
1	0.035	0.030	model starting to learn sequence pattern.
10	0.010	0.009	convergence improved
50	0.003	0.004	model stabilized minimal overfitting



- The RNN captures temporal dependencies
- Slight overfitting might occur if training continues too long.
- Adding dropout or switching to LSTM can improve long term memory handling.

Result :-

(Ayy) history = model.fit(X_train, epochs=50, batch_size=32,
validation_split=0.2, verbose=1).

Epoch over Training & validation.

Epoch 1/50 - step-loss : 0.1528
val_loss. 0.0235

Epoch 5/50 step_loss :. 0.0015
Val - loss : 0.0013

Epoch 10/50 step_loss : 5.2483e-04
val-loss : 4.1976e-04

Epoch 15/50 step-loss : 1.8258e-04.
val_loss : 1.6565e-04

Epoch 20/50 step-loss : 5.6608e-05
val : loss : 4.1410e-05

Epoch 30/50 step-loss : 1.9077e-05
val-loss : 1.2687e-05

Epoch 40/50 step : loss : 8.5267e-06
val-loss : 6.3656e-06

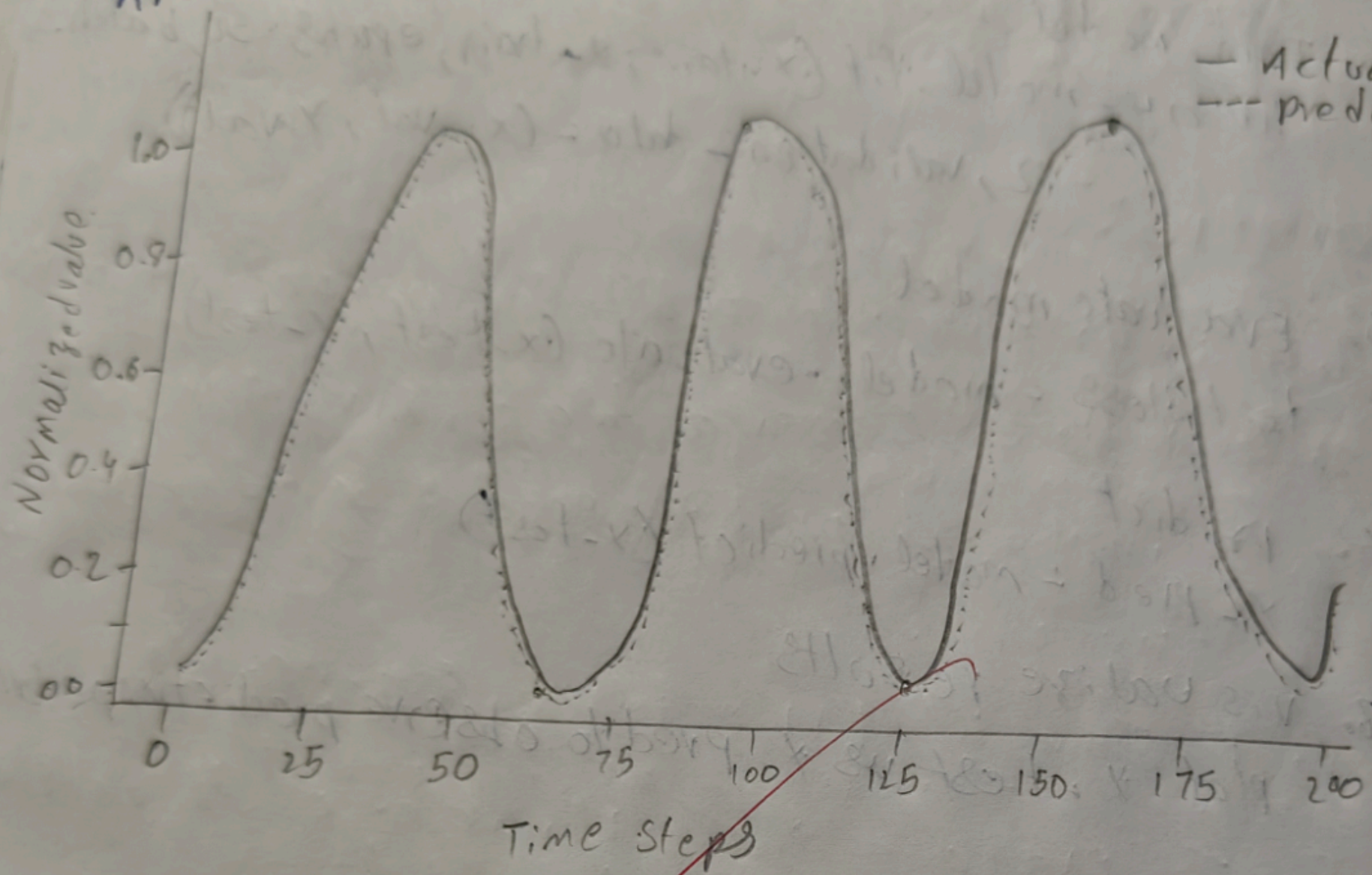
Epoch 50/50 step_loss : 5.4227e-06
val_loss : 1.9773e-05.

Output :-

graphical output

- A plot showing predicted vs. actual sequence values
- Loss curve showing convergence during training

RNN sequence Rediction (sine wave).



Model Training performance.