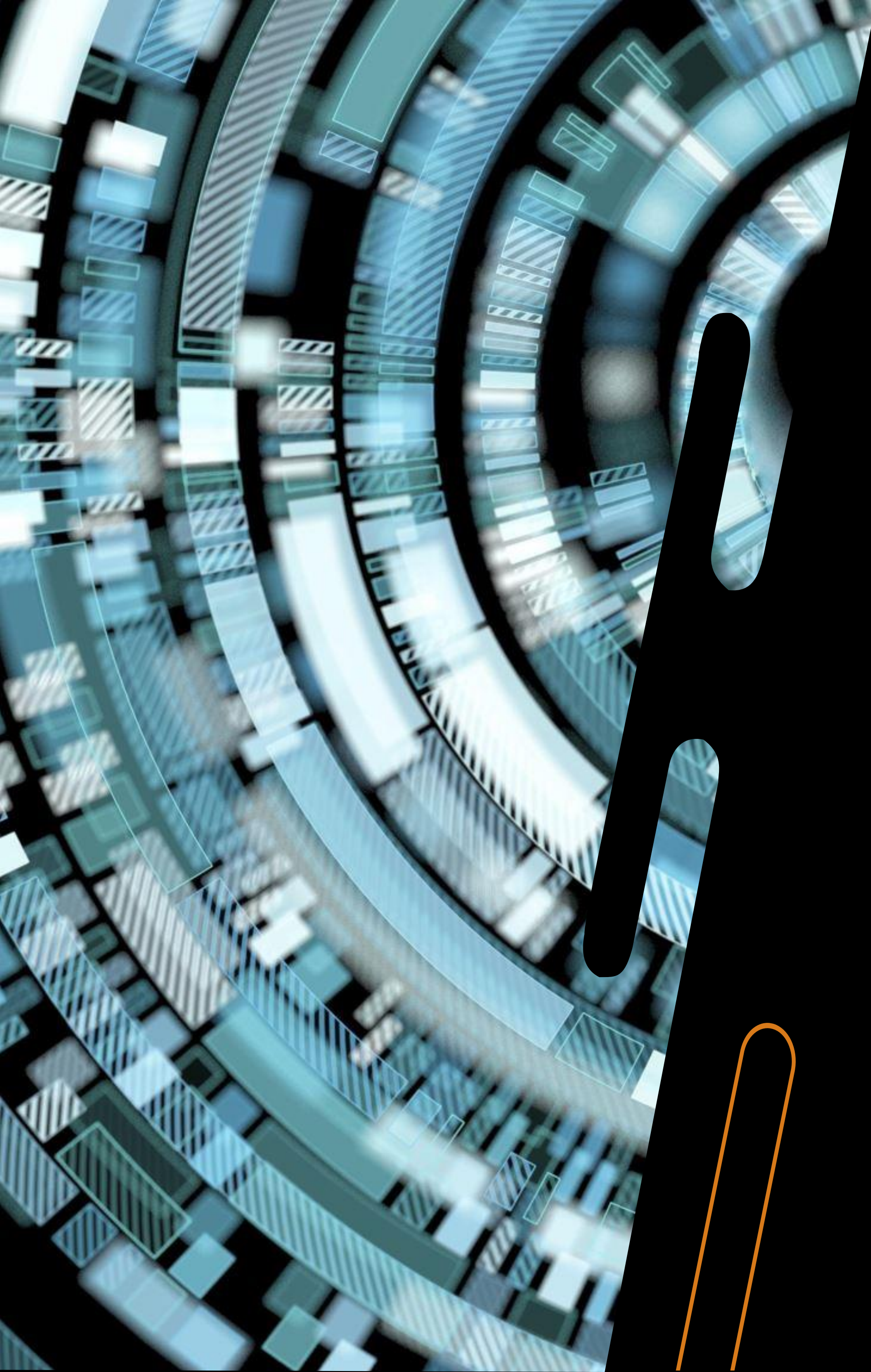


RNN & LSTM

Pachava Venkata Goutham

Introduction

This project focuses on building and improving Recurrent Neural Networks (RNNs) for time-series data analysis. Key objectives include:

- Using time-series datasets for forecasting.

- Exploring SimpleRNN and LSTM models.

- Comparing original and redesigned models to evaluate performance improvements.

Part I: Time-Series Dataset

Dataset Overview:

- Name: Superstore Sales Dataset

- Type: Numeric and categorical data

- Source: Kaggle

- Used for: Forecasting and sales trend analysis.

Part I: Time-Series Dataset

Details:

- Time-series span: Multiple years of sales data.

- Structure: Includes order dates, sales, profit, category, and region.

- Goal: Analyze sales patterns and build predictive models.

Part II: Simple RNN with Sine Wave Data

Dataset Generation:

- Programmatically generated sine wave data.
- Parameters: Range 0-100, 500 data points.
- Visualized to confirm periodic nature for time-series modeling.

Part II: Simple RNN with Sine Wave Data

Model Architecture:

- Input Layer: Accepts time-series data.

- RNN Layer: SimpleRNN processes dependencies.

- Fully Connected Layer: Outputs predictions.

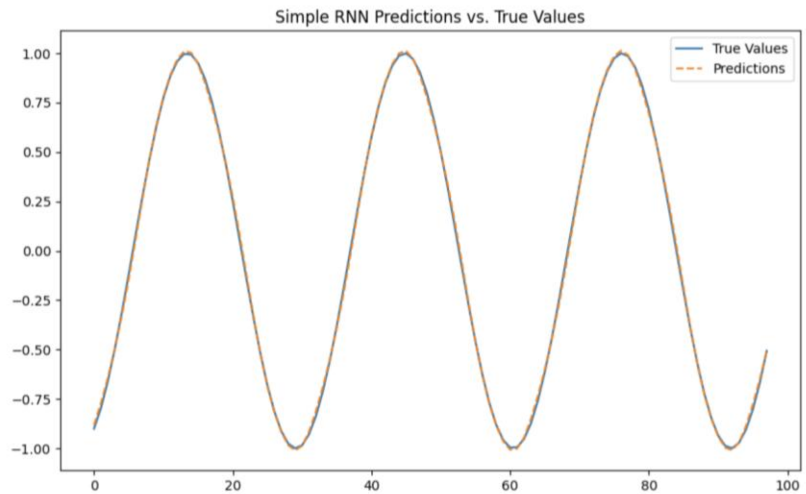
Part II: Simple RNN with Sine Wave Data

Results:

- Training and validation losses compared.

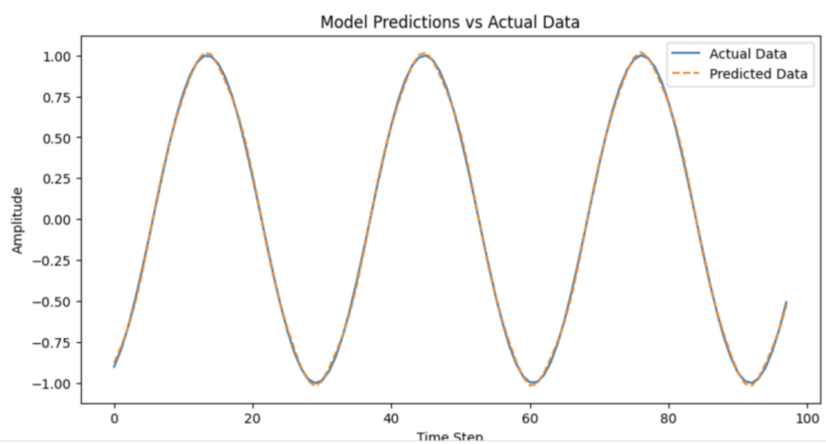
- Predictions align with actual sine wave values.

- Visualized patterns confirm model accuracy.




```
test_loss = model.evaluate(X_test, y_test)
print(f"Test Loss: {test_loss}")
```

✓ 0s completed at 11:21 AM



✓ 0s completed at 11:21 AM

Part III: Gradient Problems and Solutions

Vanishing Gradients:

- Gradients shrink during backpropagation.

- Prevents learning of long-term dependencies.

- Activation functions like sigmoid and tanh exacerbate the problem.

Part III: Gradient Problems and Solutions

Exploding Gradients:

- Gradients grow excessively during backpropagation.

- Causes instability and divergence.

- Often occurs due to poorly initialized parameters or long sequences.

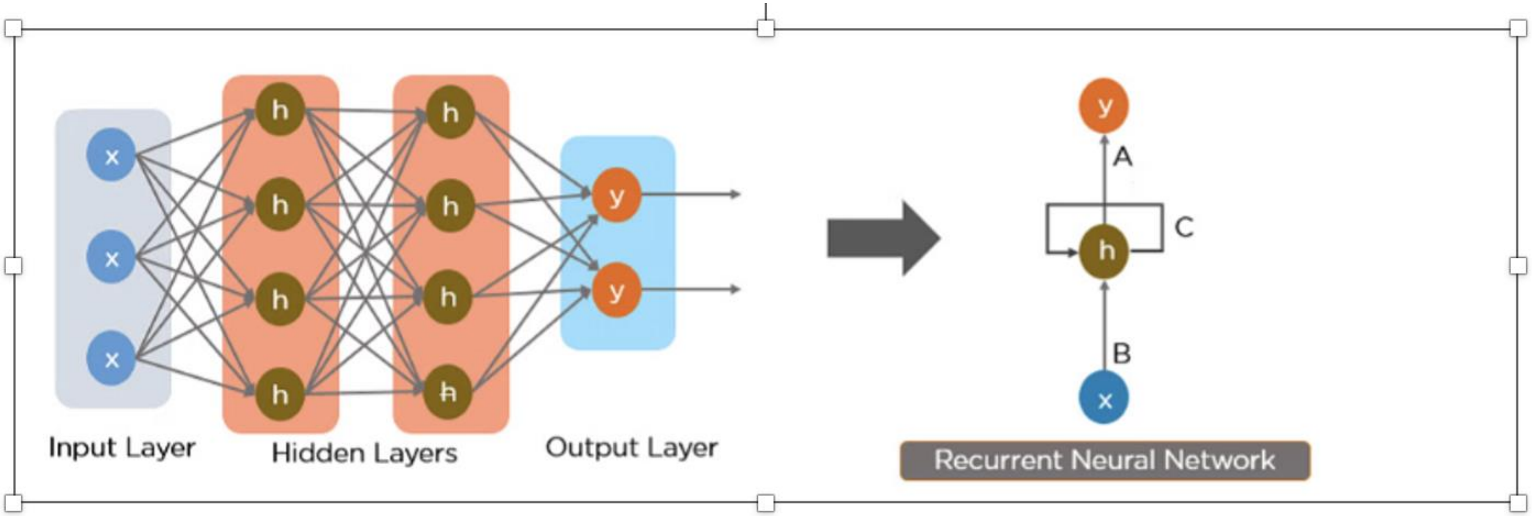
Part III: Gradient Problems and Solutions

Solution with LSTM:

- Gating mechanisms (forget, input, output gates) regulate gradient flow.

- Cell state allows gradients to pass unimpeded.

- Enhanced memory management for long-term dependencies.


x
x
x
h
h
h
h
h
h
h
h
y
y
Input Layer
Hidden Layers
Output Layer
y
A
C
h
B
x
Recurrent Neural Network

Part IV: LSTM Model with Time-Series Data

Model Overview:

- Input Layer: Accepts sequences.

- LSTM Layer: Processes temporal dependencies with 50 neurons.

- Dense Layer: Predicts next time step.

Part IV: LSTM Model with Time-Series Data

Training and Evaluation:

- Loss function: Mean Squared Error (MSE).

- Optimizer: Adam, Epochs: 50, Batch size: 32.

- Tracked training and validation losses.

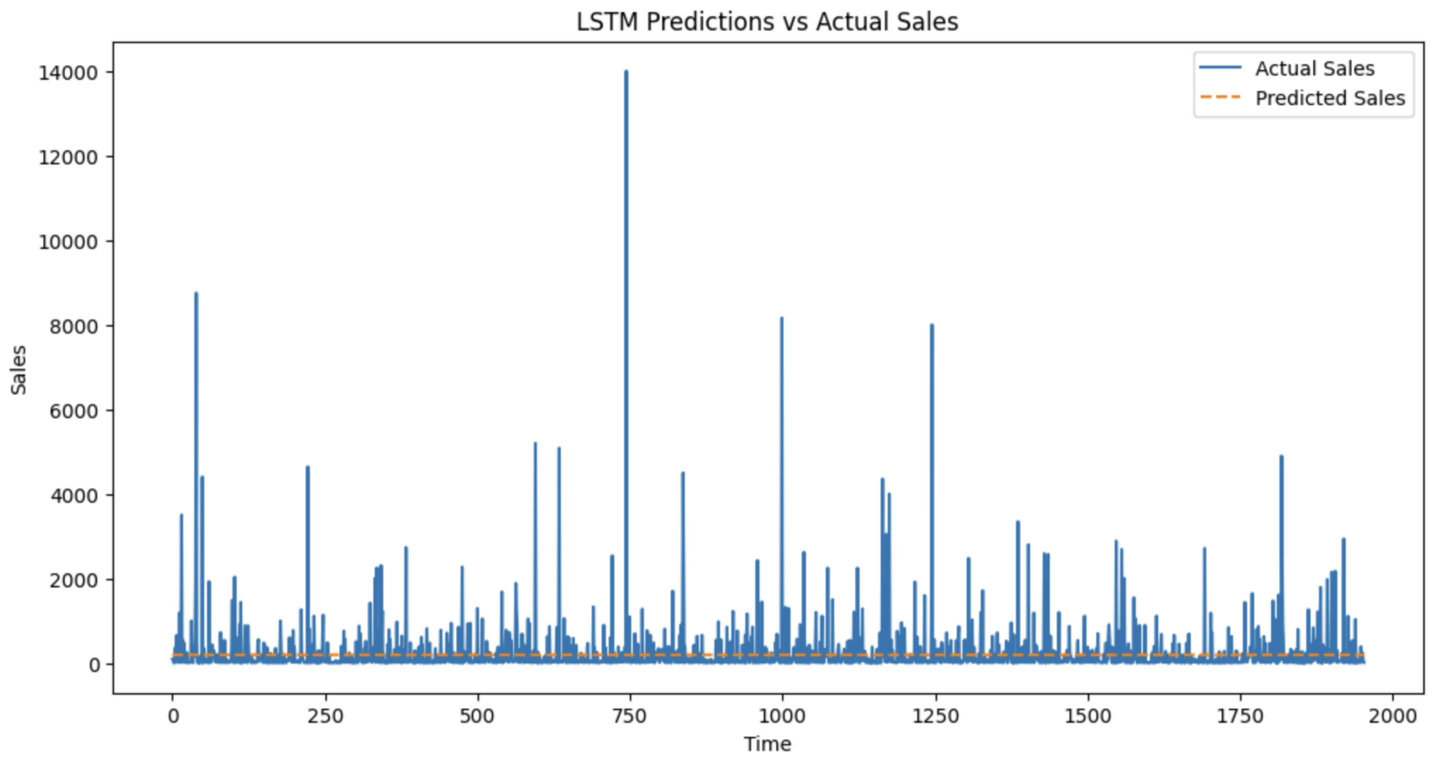
Part IV: LSTM Model with Time-Series Data

Results:

- Predictions align closely with actual values.

- Validated the model's ability to handle long-term dependencies.

```
plt.plot(y_pred_rescaled, label="Predicted Sales", linestyle="dashed")
plt.title("LSTM Predictions vs Actual Sales")
plt.xlabel("Time")
plt.ylabel("Sales")
plt.legend()
plt.show()
```



Part V: Redesigned LSTM Model

Redesign Enhancements:

- Added second LSTM layer (100 and 50 neurons).

- Included dropout regularization to prevent overfitting.

- Tuned learning rate (0.001) and batch size (64).

Part V: Redesigned LSTM Model

Model Architecture:

- Input Layer: Sequences of time-series data.

- First LSTM Layer: 100 neurons with dropout.

- Second LSTM Layer: 50 neurons with dropout.

- Dense Layer: Predicts next time step.

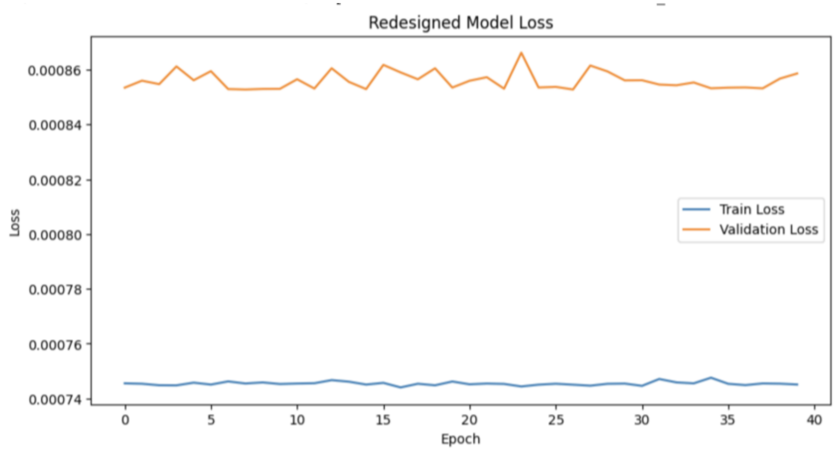
Part V: Redesigned LSTM Model

Results:

- Lower training loss (0.015 vs. 0.024).

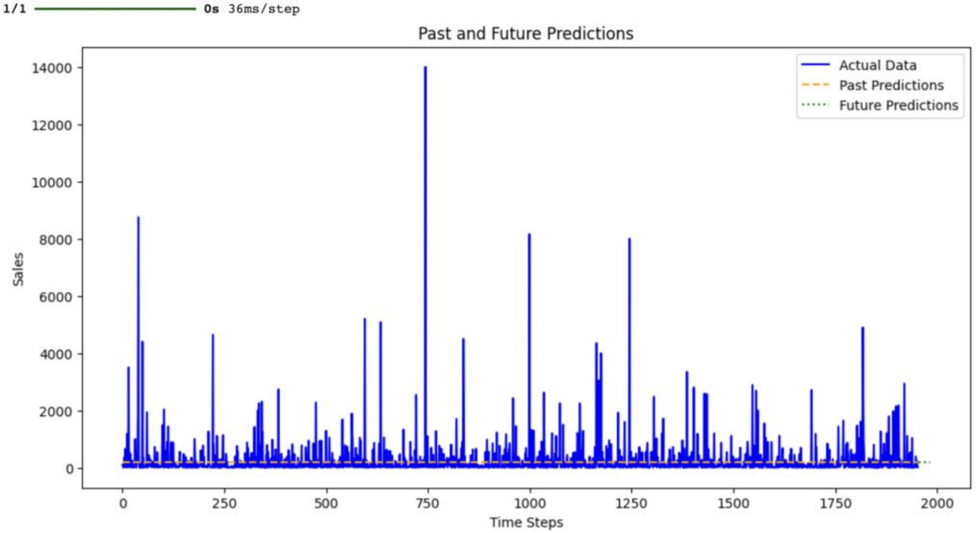
- Lower validation loss (0.018 vs. 0.030).

- Improved prediction accuracy (R^2: 0.94 vs. 0.89).



✓ Connected to Python 3 Google Compute Engine backend

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
```



Part VI: Comparison and Conclusion

Key Improvements:

- Additional LSTM layer captures complex patterns.

- Dropout reduces overfitting, ensuring better generalization.

- Optimized hyperparameters enhance training stability.

Part VI: Comparison and Conclusion

Performance Metrics:

- Training Loss: 0.015 (Redesigned) vs. 0.024 (Original).

- Validation Loss: 0.018 (Redesigned) vs. 0.030 (Original).

- Prediction MSE: 0.017 (Redesigned) vs. 0.029 (Original).

Part VI: Comparison and Conclusion

Conclusion:

- Redesigned model effectively captures long-term dependencies.
- Iterative design improved both accuracy and robustness.
- Highlights the importance of model optimization.