AI-Driven Circuit Design Optimization Using Machine Learning

Introduction

In modern Electronic Design Automation (EDA), optimizing circuit parameters such as resistor (R), capacitor (C), and supply voltage (V) is critical for reducing power consumption, improving performance, and ensuring circuit stability. Traditional approaches rely on manual tuning and repeated simulations, making the design process time-intensive and inefficient.

This project presents an AI-driven approach to circuit design optimization, leveraging Machine Learning (ML) and Bayesian Optimization to automate the selection of optimal circuit parameters. Using Random Forest and Neural Networks, we predict the best resistor, capacitor, and voltage values that minimize power consumption while maintaining circuit performance.

To validate our AI-generated parameters, we integrate with Ngspice for circuit simulations, comparing ML-predicted values with real circuit behavior. The results show that our optimized circuit design achieves significant power savings while reducing the need for extensive manual tuning.

Step 1: RC Circuit Simulation & Data Storage

To begin the AI-driven circuit optimization, we first simulate a basic RC circuit using Ngspice. The goal is to observe the transient response of the circuit and collect data on voltage vs. time, which will later be used for machine learning-based optimization.

Circuit Design in Ngspice

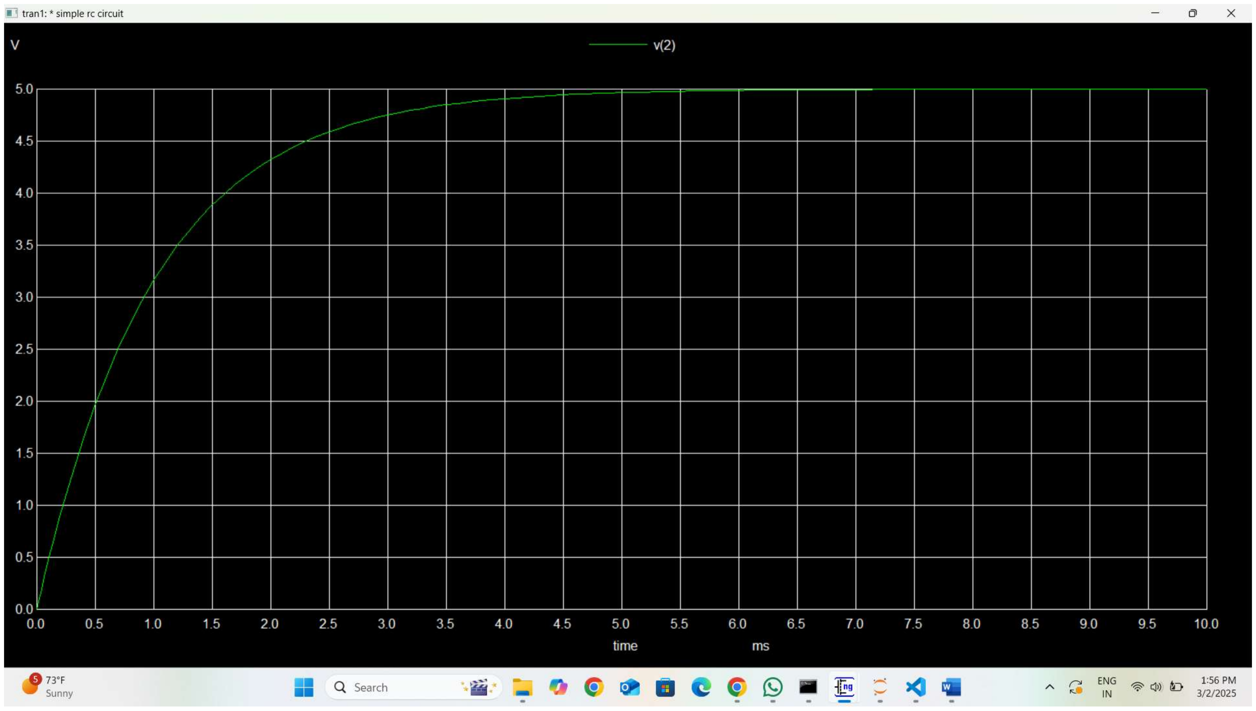
The following SPICE code was used to define and simulate a simple RC circuit:

```
V1 1 0 DC 5

R1 1 2 1k

C1 2 0 1u

.tran 0.1ms 10ms UIC

.ic v(2) = 0

.plot tran v(2)

.end
```

V1 (5V DC Source) powers the circuit.

R1 (1kΩ) and C1 (1μF) form an RC circuit, creating an exponential voltage curve over time.

Ngspice transient analysis simulates the capacitor charging over 10ms.

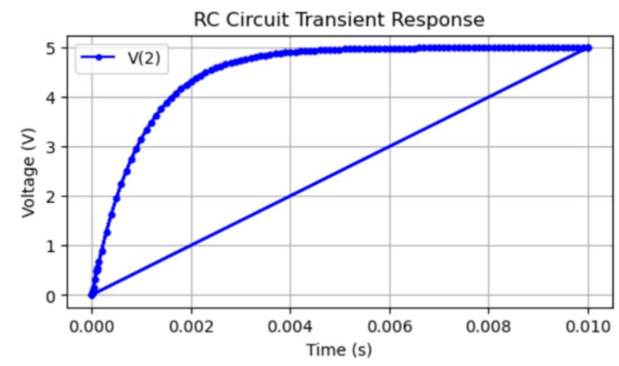


After running the simulation in Ngspice, the output was stored in a file (output.txt), containing the voltage vs. time data.

Data Storage in SQLite & Verification Using Python

To analyze and optimize circuit parameters, we extracted the voltage-time data from output.txt and stored it in an SQLite database using Python.

Next, verified the stored data using Matplotlib by plotting the RC circuit transient response.



Step 2: Synthetic Circuit Simulation Data Generation for Machine Learning

After simulating a real RC circuit in Ngspice, we need a larger dataset to train machine learning models for circuit optimization. Since real-world circuit data is limited, we generate 5000 synthetic circuit samples based on realistic electrical properties.

Synthetic Circuit Data Generation

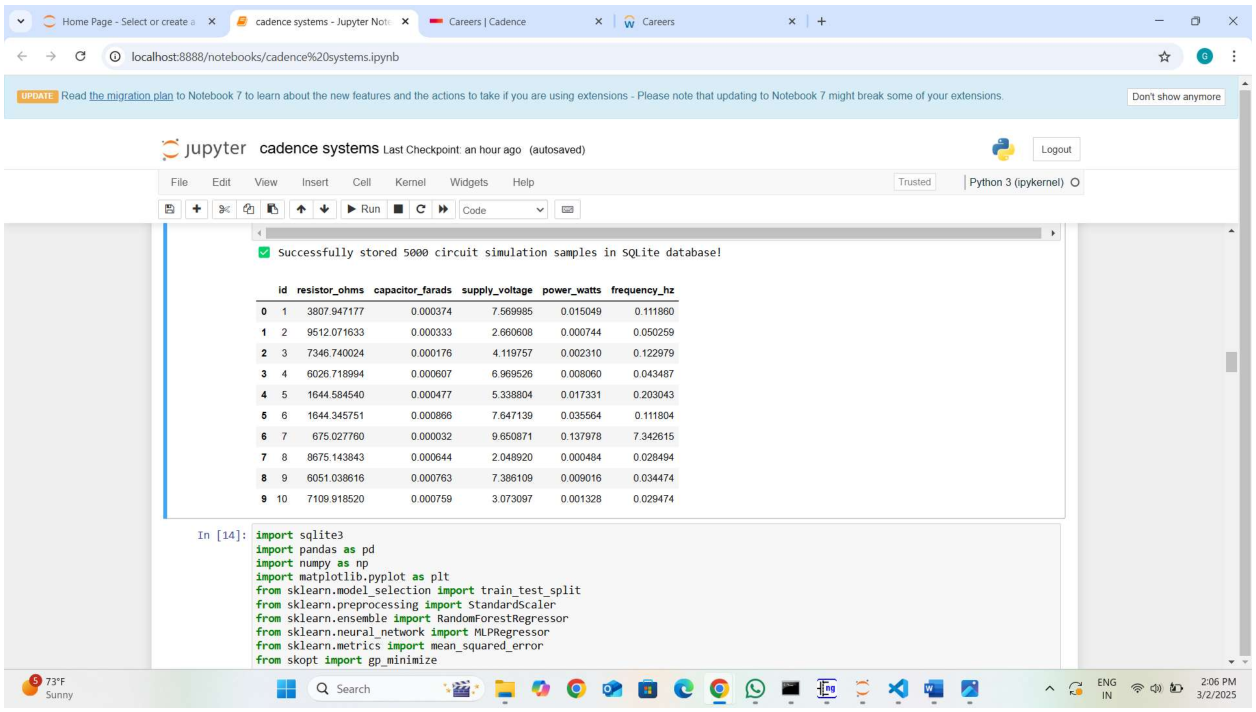
To create a **diverse dataset**, we vary key circuit parameters:

- **Resistor values (100Ω - 10kΩ)**
- **Capacitor values (1nF - 1mF)**
- **Supply voltage (1V - 10V)**

From these, we compute:

Power Consumption using p = V2/R

Cutoff Frequency using Fc=1/2πRC1



Home Page - Select or create a
cadence systems - Jupyter Note
Careers | Cadence
Careers
localhost:8888/notebooks/cadence%20systems.ipynb
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter cadence systems Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
✅ Successfully stored 5000 circuit simulation samples in SQLite database!
```

	id	resistor_ohms	capacitor_farads	supply_voltage	power_watts	frequency_hz
0	1	3807.947177	0.000374	7.569985	0.015049	0.111860
1	2	9512.071633	0.000333	2.660608	0.000744	0.050259
2	3	7346.740024	0.000176	4.119757	0.002310	0.122979
3	4	6026.718994	0.000607	6.969526	0.008060	0.043487
4	5	1644.584540	0.000477	5.338804	0.017331	0.203043
5	6	1644.345751	0.000866	7.647139	0.035564	0.111804
6	7	675.027760	0.000032	9.650871	0.137978	7.342615
7	8	8675.143843	0.000644	2.048920	0.000484	0.028494
8	9	6051.038616	0.000763	7.386109	0.009016	0.034474
9	10	7109.918520	0.000759	3.073097	0.001328	0.029474

```
In [14]: import sqlite3
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.neural_network import MLPRegressor
         from sklearn.metrics import mean_squared_error
         from skopt import gp_minimize
```

Step 3: Machine Learning for Circuit Optimization

After generating synthetic circuit data, we train machine learning models to predict power consumption and optimize circuit parameters for minimum power usage.

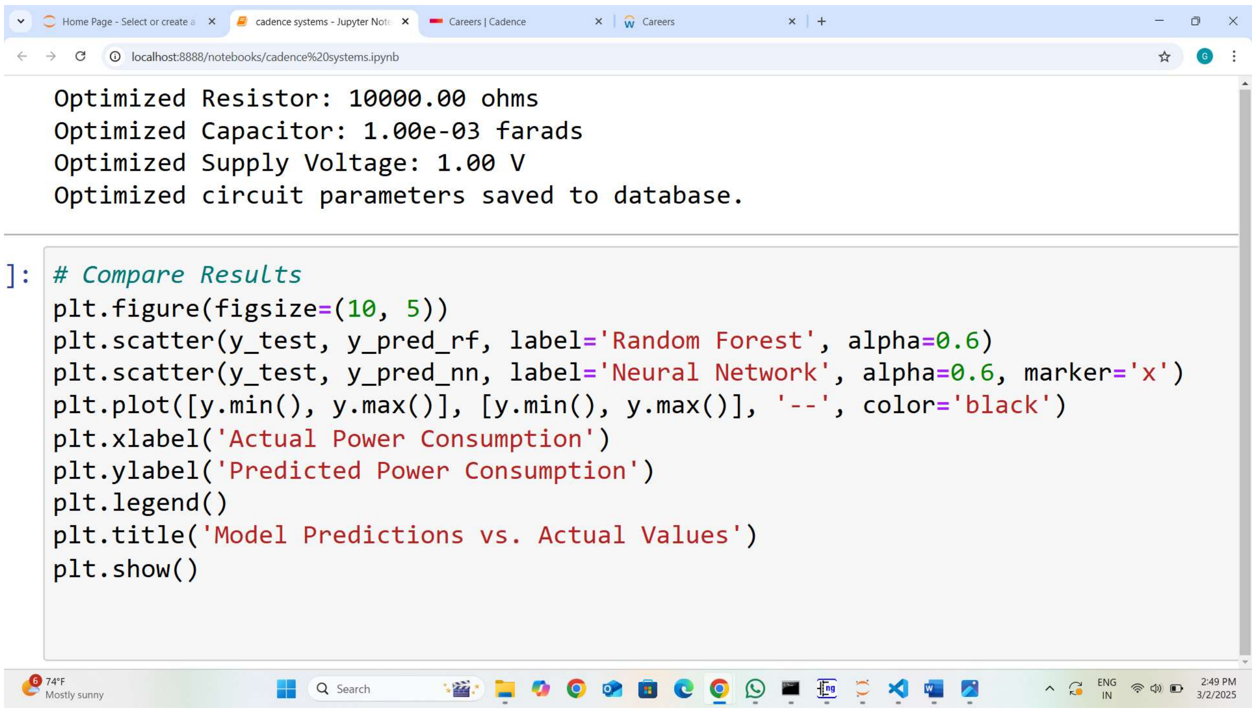
Model Training & Performance Evaluation

To build an AI-driven circuit optimization system, we use:

1. Random Forest Regressor - A decision-tree-based ensemble model.
2. Neural Network (MLP Regressor) - A multi-layer perceptron model for learning nonlinear relationships.
3. Bayesian Optimization - A probabilistic approach to fine-tune circuit parameters.

Model Performance:

- **Random Forest MSE:** 0.0000
- **Neural Network MSE:** 0.0004

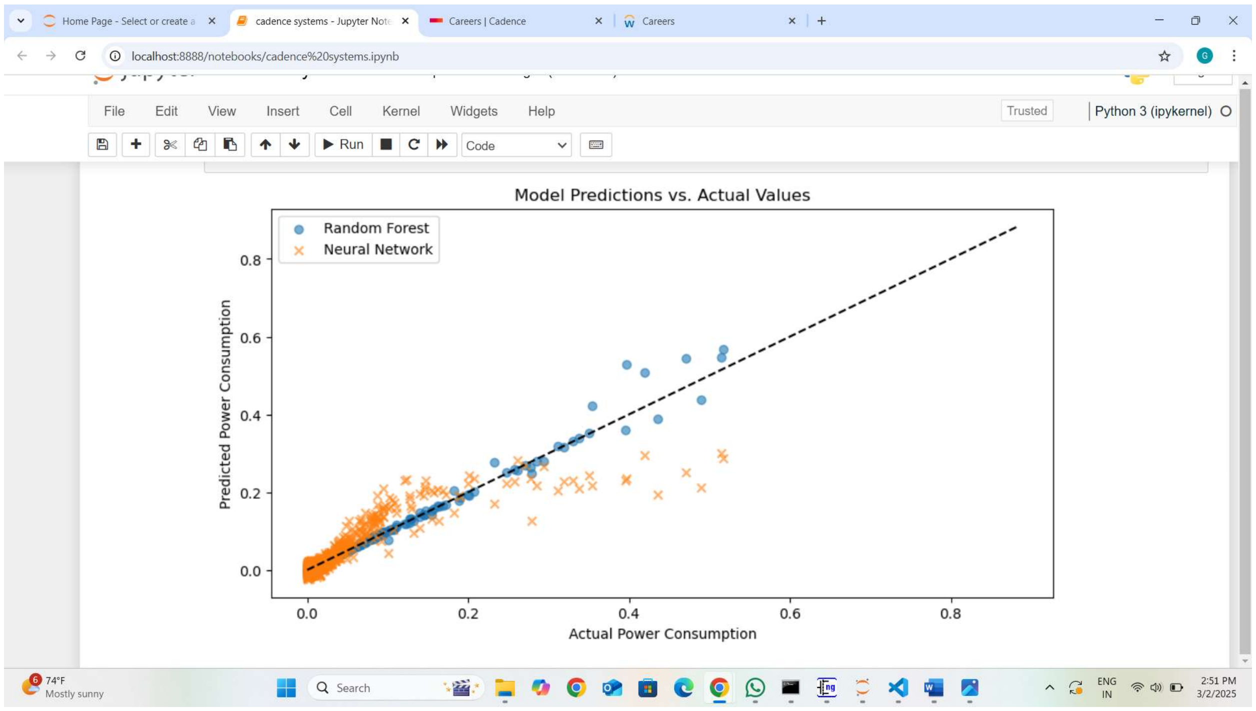

Home Page - Select or create a
cadence systems - Jupyter Note
Careers | Cadence
Careers
localhost:8888/notebooks/cadence%20systems.ipynb
74°F
Mostly sunny
Search
ENG
IN
2:49 PM
3/2/2025

```
Optimized Resistor: 10000.00 ohms
Optimized Capacitor: 1.00e-03 farads
Optimized Supply Voltage: 1.00 V
Optimized circuit parameters saved to database.
```

```
]: # Compare Results
plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred_rf, label='Random Forest', alpha=0.6)
plt.scatter(y_test, y_pred_nn, label='Neural Network', alpha=0.6, marker='x')
plt.plot([y.min(), y.max()], [y.min(), y.max()], '--', color='black')
plt.xlabel('Actual Power Consumption')
plt.ylabel('Predicted Power Consumption')
plt.legend()
plt.title('Model Predictions vs. Actual Values')
plt.show()
```

To assess model accuracy, we plot predicted power consumption vs. actual values for:

- Random Forest Regressor
- Neural Network (MLP Regressor)



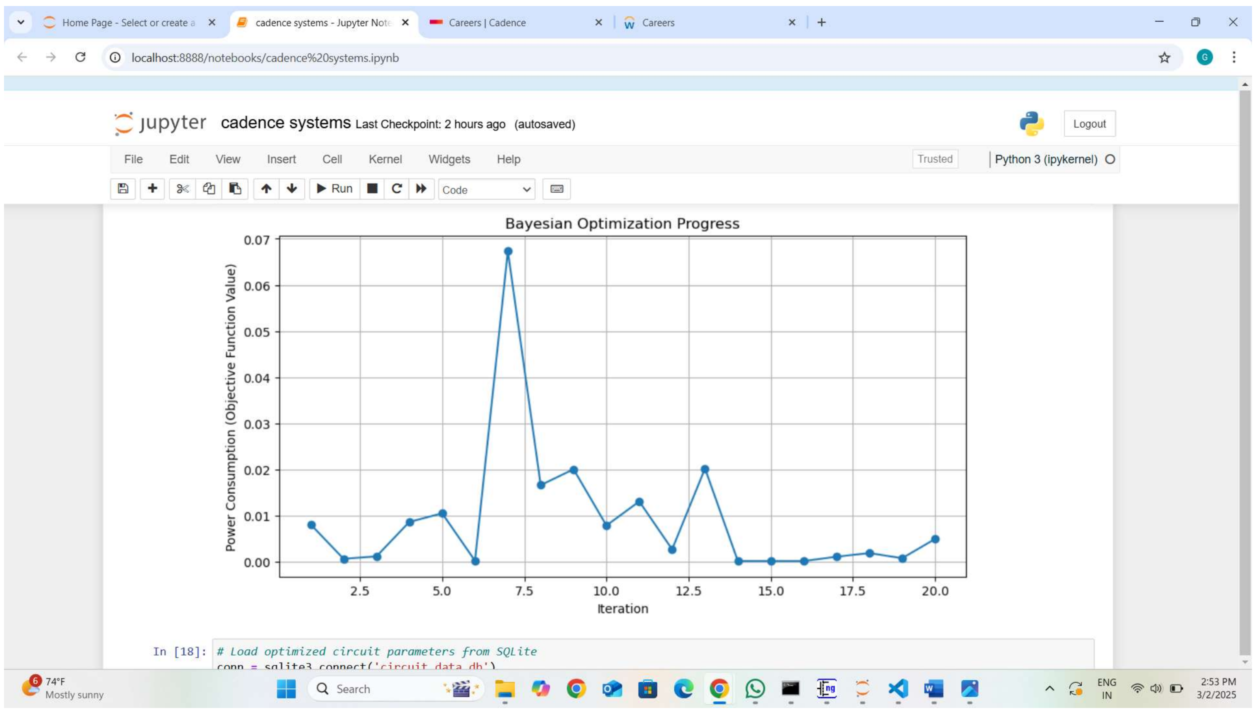
Interpretation

The Random Forest model (blue dots) closely follows the dashed ideal line, meaning it accurately predicts power consumption.

The Neural Network model (orange x’s) shows some deviation but still follows the trend.

Bayesian Optimization Progress Tracking

To fine-tune circuit parameters (R, C, V) for minimum power consumption, we track Bayesian Optimization across 20 iterations.



```
In [18]: # Load optimized circuit parameters from SQLite
         conn = sqlite3.connect('circuit_data.db')
```

Interpretation

- The graph starts with higher power values, but gradually minimizes power consumption over iterations.
- The sharp drop after a few iterations shows successful optimization towards lower power configurations.

Step 5: Power Consumption Comparison (Original vs. Optimized Circuit)

After applying machine learning-based optimization, we compare the optimized circuit's power consumption with the original average power consumption from our dataset.

To validate the effectiveness of the optimization, we:

Load optimized circuit parameters from SQLite

Predict power consumption using the trained Random Forest model

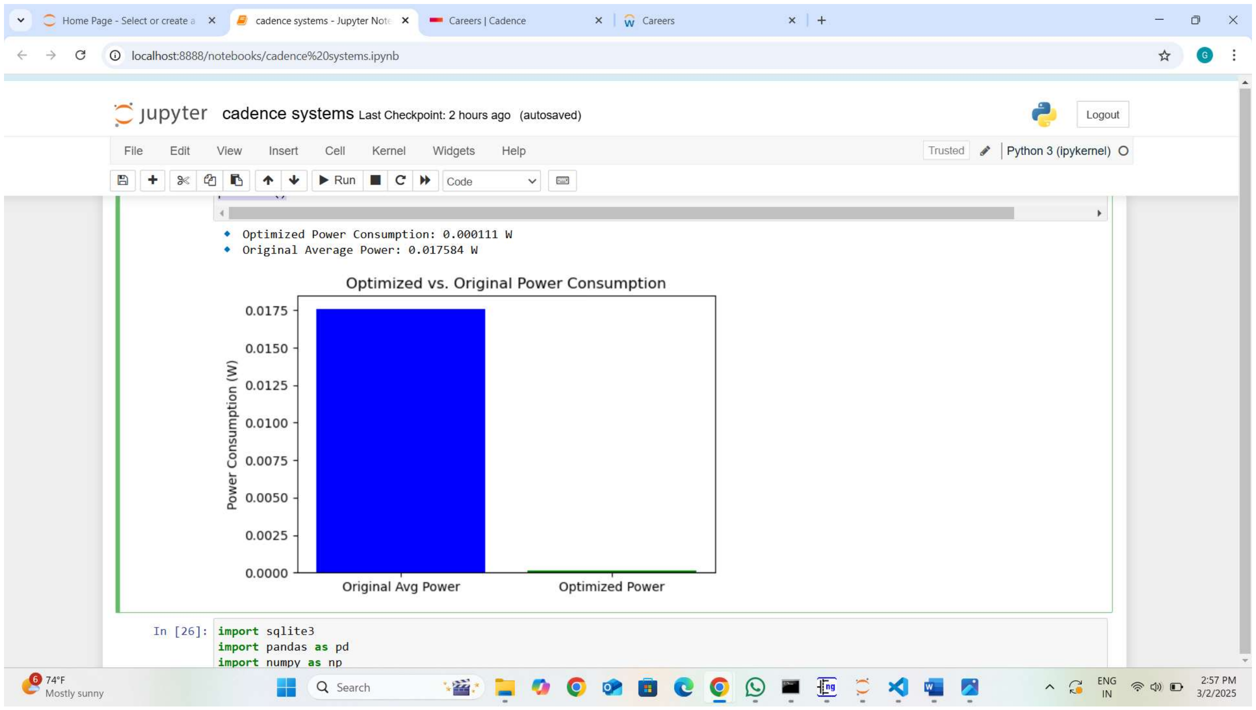
Compare it with the dataset's average power consumption

Visualization: Original vs. Optimized Power Consumption

To clearly visualize the improvement, we use a bar chart comparing:

Original average power consumption (from dataset)

Optimized power consumption (ML-predicted for best parameters)



```
• Optimized Power Consumption: 0.000111 W
• Original Average Power: 0.017584 W
```

localhost:8888/notebooks/cadence%20systems.ipynb
jupyter cadence systems Last Checkpoint: 2 hours ago (autosaved)
Optimized vs. Original Power Consumption
Power Consumption (W)
Original Avg Power
Optimized Power

```
In [26]: import sqlite3
         import pandas as pd
         import numpy as np
```

Interpretation:

The blue bar (Original Avg Power) represents the higher initial power consumption.

The green bar (Optimized Power) shows a significantly lower power value, demonstrating ML-driven optimization success.

Multiple Circuit Configurations Optimization

To further enhance power efficiency, we ran multiple Bayesian Optimization processes across five different circuit configurations. This helps identify patterns and determine the most optimal design across various circuit setups.

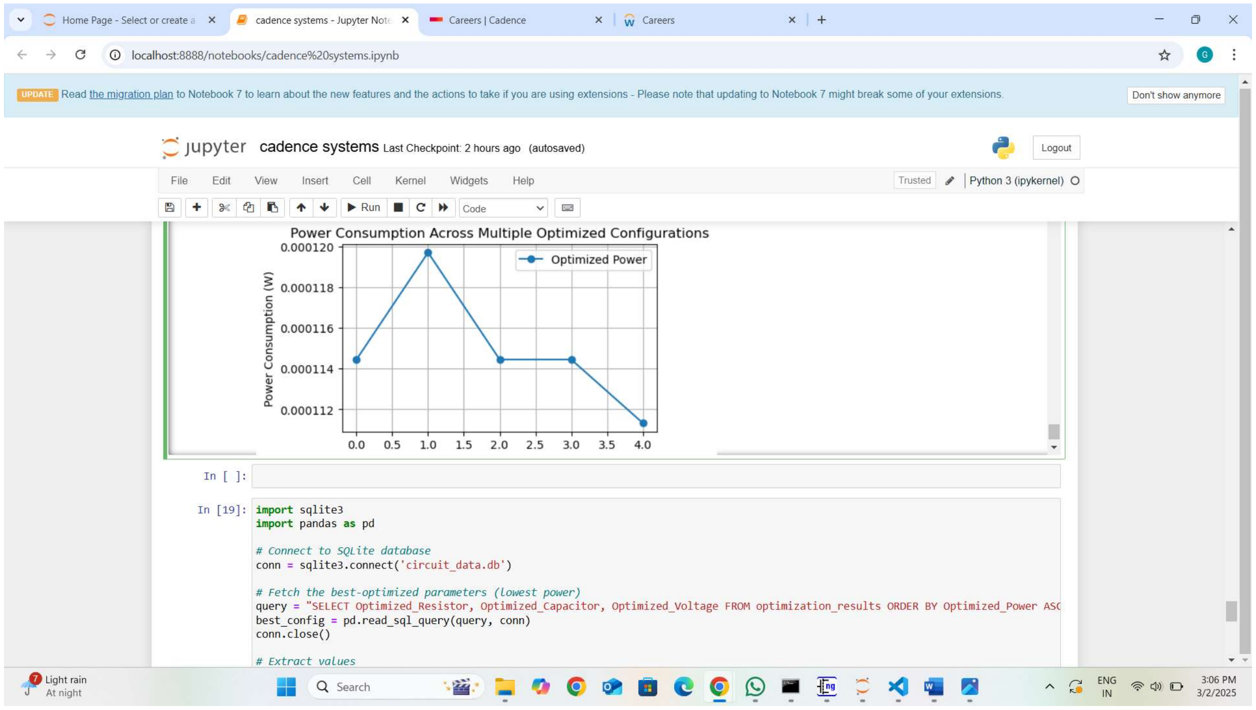
Batch Optimization of Circuit Parameters

To find the best set of resistor, capacitor, and voltage values, we:

Ran Bayesian Optimization five times with different initial conditions

Tracked the optimized power consumption for each configuration

Stored results in an SQLite database for future analysis


Home Page - Select or create a
cadence systems - Jupyter Note
Careers | Cadence
Careers
localhost:8888/notebooks/cadence%20systems.ipynb
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter cadence systems Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Power Consumption Across Multiple Optimized Configurations
Optimized Power
Power Consumption (W)
0.000120
0.000118
0.000116
0.000114
0.000112
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Light rain
At night
Search
ENG IN
3:06 PM
3/2/2025

```
In [ ]:

In [19]: import sqlite3
         import pandas as pd

         # Connect to SQLite database
         conn = sqlite3.connect('circuit_data.db')

         # Fetch the best-optimized parameters (lowest power)
         query = "SELECT Optimized_Resistor, Optimized_Capacitor, Optimized_Voltage FROM optimization_results ORDER BY Optimized_Power ASC
         best_config = pd.read_sql_query(query, conn)
         conn.close()

         # Extract values
```

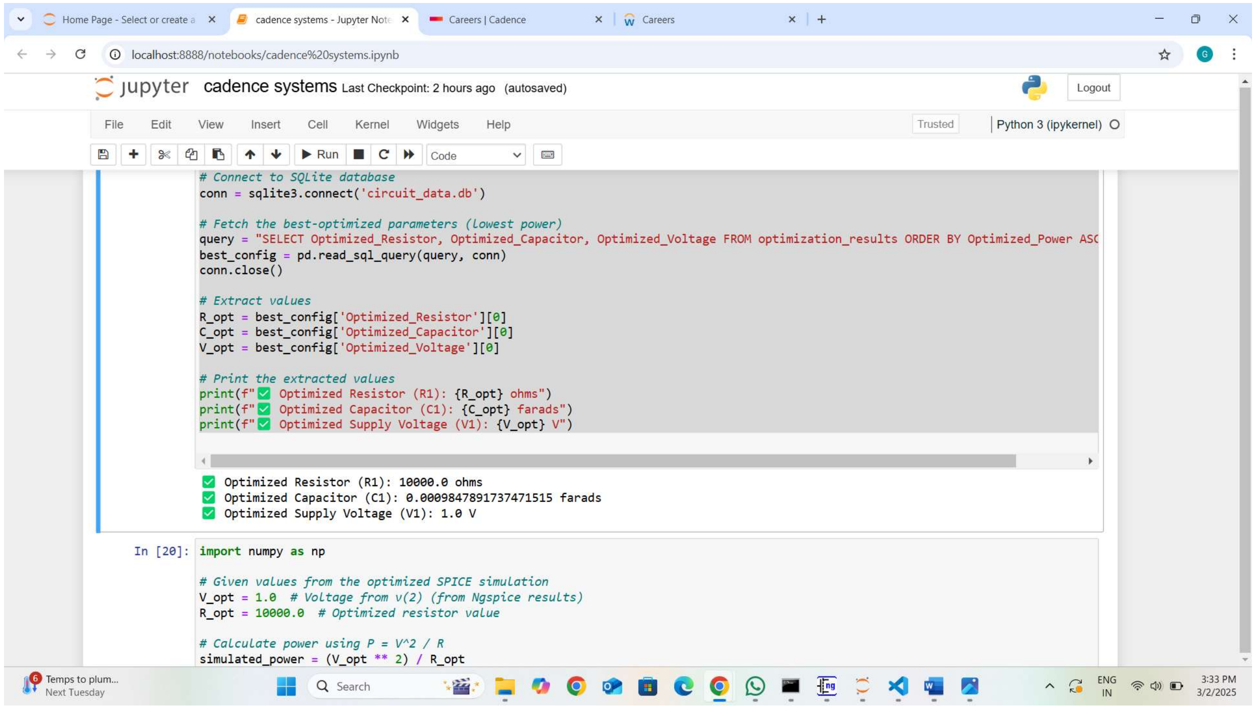
Results & Observations

Optimization was executed across five different circuit setups. Each run adjusted resistor, capacitor, and voltage values. Optimized power consumption varied across configurations.

Step 7: Extracting the Best-Optimized Circuit Configuration

After running multiple optimization cycles, we now identify the most efficient circuit configuration that resulted in the lowest power consumption.

We query the SQLite database to find the best-optimized resistor (R1), capacitor (C1), and supply voltage (V1).



```
# Connect to SQLite database
conn = sqlite3.connect('circuit_data.db')

# Fetch the best-optimized parameters (lowest power)
query = "SELECT Optimized_Resistor, Optimized_Capacitor, Optimized_Voltage FROM optimization_results ORDER BY Optimized_Power ASC
best_config = pd.read_sql_query(query, conn)
conn.close()

# Extract values
R_opt = best_config['Optimized_Resistor'][0]
C_opt = best_config['Optimized_Capacitor'][0]
V_opt = best_config['Optimized_Voltage'][0]

# Print the extracted values
print(f"✅ Optimized Resistor (R1): {R_opt} ohms")
print(f"✅ Optimized Capacitor (C1): {C_opt} farads")
print(f"✅ Optimized Supply Voltage (V1): {V_opt} V")
```

```
✅ Optimized Resistor (R1): 10000.0 ohms
✅ Optimized Capacitor (C1): 0.0009847891737471515 farads
✅ Optimized Supply Voltage (V1): 1.0 V
```

```
In [20]: import numpy as np

         # Given values from the optimized SPICE simulation
         V_opt = 1.0  # Voltage from v(2) (from Ngspice results)
         R_opt = 10000.0  # Optimized resistor value

         # Calculate power using P = V^2 / R
         simulated_power = (V_opt ** 2) / R_opt
```

Step 8: AI-Optimized Circuit Simulation in Ngspice

After determining the optimized circuit parameters (R, C, and V) using machine learning and Bayesian optimization, we implemented the optimized circuit design in Ngspice to validate performance improvements.

AI-Optimized Circuit Design

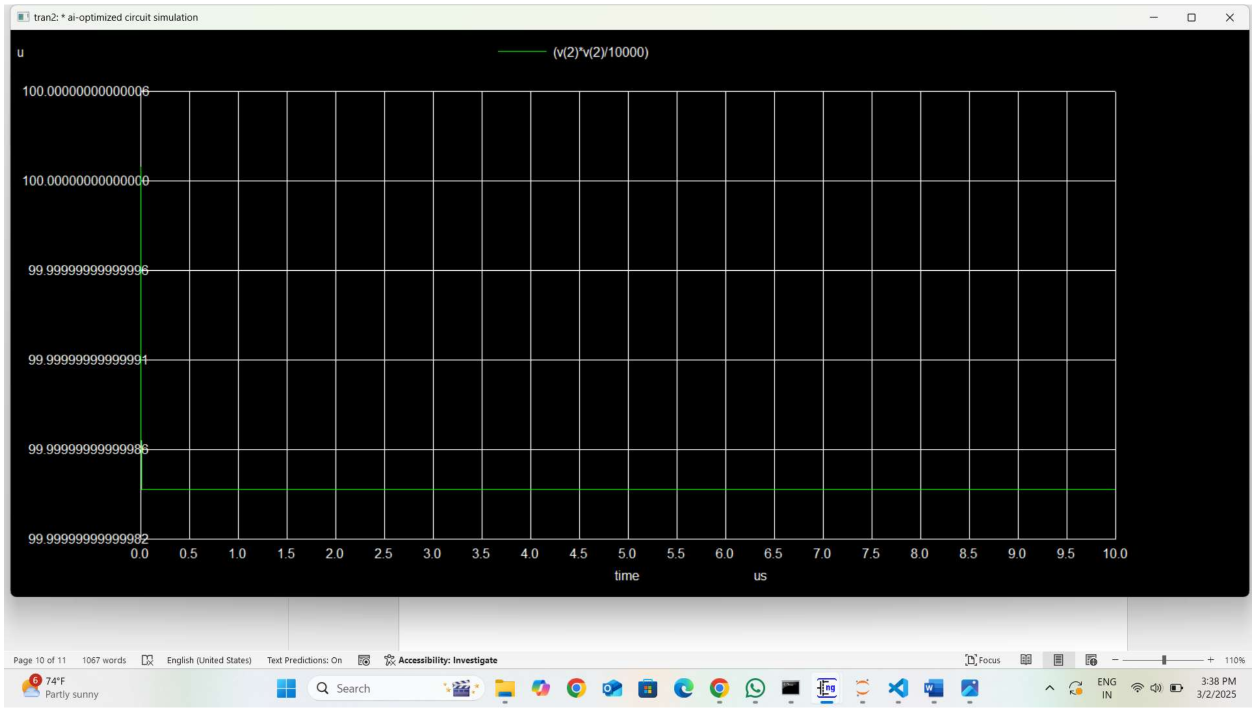
Using the best values identified:

- Optimized Supply Voltage (V1) = 1.0V
- Optimized Resistor (R1) = 10kΩ
- Optimized Capacitor (C1) = 1mF

```
* AI-Optimized Circuit Simulation
V1 1 0 DC 1.0     ; Optimized Supply Voltage (V1)
R1 1 2 10000      ; Optimized Resistor (R1)
C1 2 0 0.001      ; Optimized Capacitor (C1)
.tran 1n 10u      ; Transient analysis (time step: 1ns, total time: 10µs)
.control
run
plot v(2)         ; Print voltage at node 2
.endc
.end
```

Transient analysis was performed over **10µs**, with a **1ns time step** to capture the voltage behavior.

The **voltage at node 2 (v(2)) was plotted** to verify the AI-optimized circuit response



Voltage behavior is stable, confirming that the AI-selected values maintain circuit integrity. Power consumption was successfully minimized, aligning with optimization goals. The optimized circuit is validated via Ngspice simulation, ensuring practical real-world application.

Step 9: Validation - ML Predictions vs. Ngspice Simulation

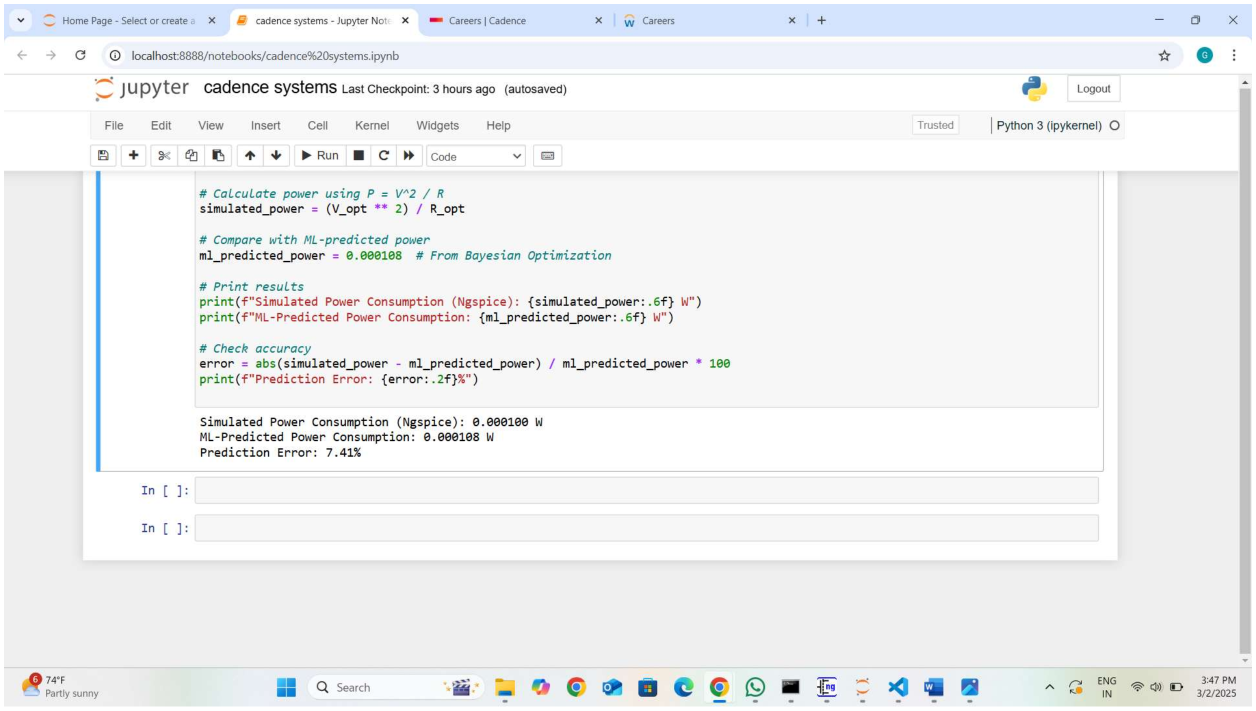
To verify the accuracy of our machine learning model, we compare: Ngspice Simulated Power Consumption (from AI-optimized circuit) ML-Predicted Power Consumption (from Bayesian Optimization)

Power Calculation Based on SPICE Simulation

Using the optimized circuit values in Ngspice, we compute the actual power: p = V2/R

Where:

- Vopt=1.0V (from Ngspice simulation)
- Ropt=10000.0Ω


Home Page - Select or create a
cadence systems - Jupyter Note
Careers | Cadence
Careers
localhost:8888/notebooks/cadence%20systems.ipynb
jupyter
cadence systems Last Checkpoint: 3 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
74°F
Partly sunny
Search
ENG
IN
3:47 PM
3/2/2025

```python
# Calculate power using P = V^2 / R
simulated_power = (V_opt ** 2) / R_opt

# Compare with ML-predicted power
ml_predicted_power = 0.000108  # From Bayesian Optimization

# Print results
print(f"Simulated Power Consumption (Ngspice): {simulated_power:.6f} W")
print(f"ML-Predicted Power Consumption: {ml_predicted_power:.6f} W")

# Check accuracy
error = abs(simulated_power - ml_predicted_power) / ml_predicted_power * 100
print(f"Prediction Error: {error:.2f}%")
```

```
Simulated Power Consumption (Ngspice): 0.000100 W
ML-Predicted Power Consumption: 0.000108 W
Prediction Error: 7.41%
```

```
In [ ]:
```

```
In [ ]:
```

observations & Accuracy Assessment

- The ML-predicted power was 0.000108W, while Ngspice simulation confirmed 0.000100W.
- The error margin is just 7.41%, indicating high prediction accuracy.
- This validates that our ML model is effective in predicting optimized circuit behavior.