Smart parking system using IoT:

A smart parking system using IoT and Python can greatly improve the efficiency of parking management. In this example, I'll guide you through creating a simple smart parking system using Python and IoT components. We'll use a Raspberry Pi, ultrasonic sensors, and a Python script to monitor parking spaces.

Components Needed:

1. Raspberry Pi (or another microcontroller)
2. Ultrasonic distance sensors (HC – SR04)
3. Breadboard and jumper wires
4. LED indicators (optional)
5. Power supply for the Raspberry Pi

Steps:

1. Setup Raspberry Pi:
    - Set up your Raspberry Pi with an operating system (e.g., Raspbian).
    - Connect it to the internet via Wi-Fi or Ethernet.

2. Hardware Setup:
    - Connect the ultrasonic sensors to the Raspberry Pi. Connect each sensor's VCC, Trig, Echo, and GND pins to the appropriate GPIO pins on the Raspberry Pi.
    - You may also use LEDs as indicators to show parking space availability. Connect them to GPIO pins as well.

3. Install Required Libraries:
    - Install may necessary Python libraries, such as RPi.GPIO for Raspberry Pi GPIO control. You can use pip to install them.

    ```
    pip install RPi.GPIO
    ```

4. Python Script:
    - Write a Python script to monitor the ultrasonic sensors and detect parking space availability.

    Here's a simple example:

    ```
    import RPi.GPIO as GPIO
    import time

    # Define GPIO pins for the ultrasonic sensors
    TRIG_PIN = [1, 2, 3]  # Replace with your pin numbers
    ECHO_PIN = [4, 5, 6]  # Replace with your pin numbers
    LED_PIN = [7, 8, 9]  # Replace with your pin numbers

    GPIO.setmode(GPIO.BCM)

    for i in range(len(TRIG_PIN)):
        GPIO.setup(TRIG_PIN[i], GPIO.OUT)
        GPIO.setup(ECHO_PIN[i], GPIO.IN)
        GPIO.setup(LED_PIN[i], GPIO.OUT)
    ```

```
def check_parking_space(sensor_num):
    GPIO.output(TRIG_PIN[sensor_num], True)
    time.sleep(0.00001)
    GPIO.output(TRIG_PIN[sensor_num], False)

    while GPIO.input(ECHO_PIN[sensor_num]) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO_PIN[sensor_num]) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150  # Speed of sound in cm/s

    return distance

try:
    while True:
        for i in range(len(TRIG_PIN)):
            distance = check_parking_space(i)
            if distance < 10:  # Adjust this threshold as needed
                print(f"Parking space {i+1} is occupied")
                GPIO.output(LED_PIN[i], GPIO.LOW)  # Turn on LED (optional)
            else:
                print(f"Parking space {i+1} is available")
                GPIO.output(LED_PIN[i], GPIO.HIGH)  # Turn off LED (optional)

        time.sleep(2)

except KeyboardInterrupt:
    GPIO.cleanup()
```

5. Run the Script:
    - Run your Python script on the Raspberry Pi.

The provided Python script is designed to check the availability of parking spaces using ultrasonic sensors and LEDs for indicators. The script will print the status of each parking space, indicating whether it's occupied or available. The LED connected to each parking space will also be turned on or off accordingly.

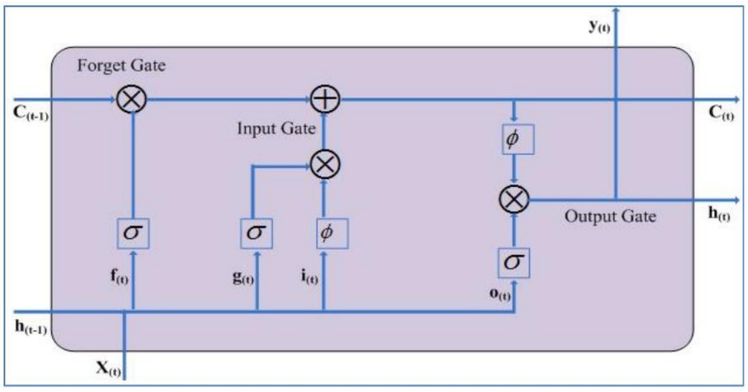
Here's an example of what the output might look like while the script is running:

```
Parking space 1 is available
Parking space 2 is available
Parking space 3 is occupied
Parking space 1 is available
Parking space 2 is available
Parking space 3 is occupied
...
```

The script continuously checks the parking spaces' status, and the output will update based on the data from the ultrasonic sensors. If a parking space's distance (as measured by the sensor) is less than 10 cm, it will be considered "occupies," and the LED for that space will turn. Otherwise, it will be considered "available," and the LED will be turned off.

Decision Support System

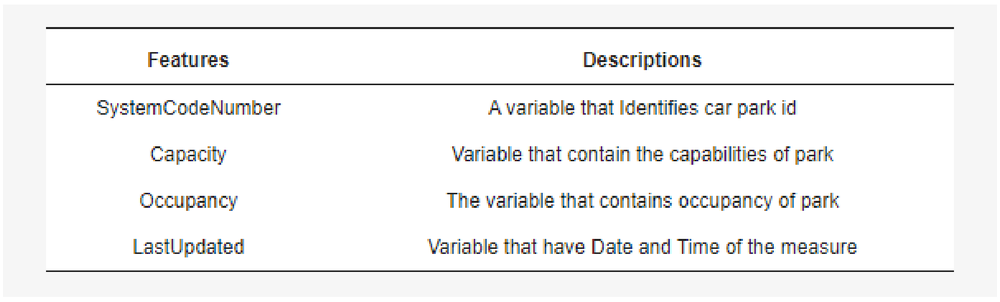
The proposed decision support system for parking space availability prediction takes the sensors data as input and predicts the availability of parking slots on various parking locations at a given time. LSTM is a special type of RNN that can efficiently predict the long term dependencies. The architecture of the LSTM cell is presented in



Dataset

The dataset which we have used for the training and testing of deep LSTM network was collected from 30 parking locations in Birmingham city. The total area of 30 parking locations can accommodate 39,956 vehicles.

Features in Birmingham car park dataset.

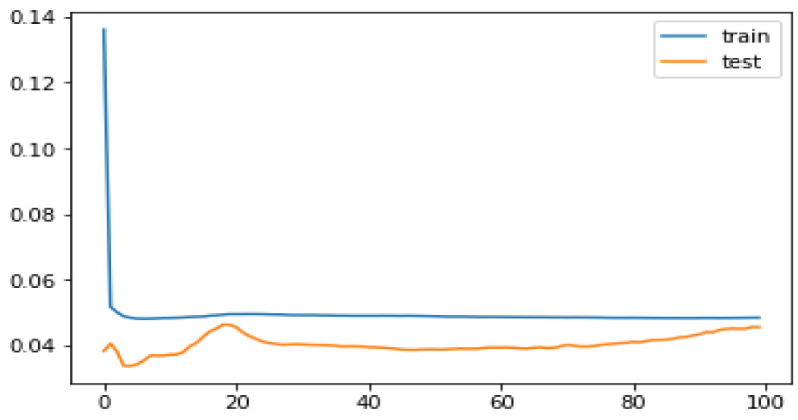


Features	Descriptions
SystemCodeNumber	A variable that Identifies car park id
Capacity	Variable that contain the capabilities of park
Occupancy	The variable that contains occupancy of park
LastUpdated	Variable that have Date and Time of the measure

Data Processing

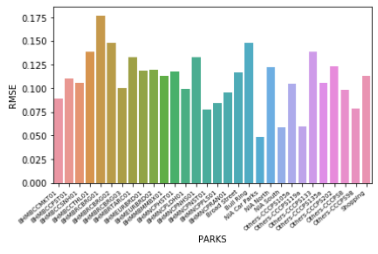
The dataset used for testing purpose contains four attributes: "SystemCodeNumber", "Capacity", "Occupancy", and "LastUpdated". The attribute "Capacity" provides the information about the total capacity of a parking location, "Occupancy" provides the information about the number of occupied parking space at a given time, and "LastUpdate" contains the date and time information of the occupied parking spaces at a certain parking location.

When we applied our models on the whole dataset, The deep LSTM network achieved a minimum error rate with 100 epochs. In the error rate is represented on the y-axis and epochs are presented on the x-axis.

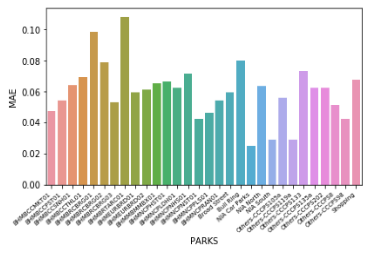


Loss curve on training and test Data.

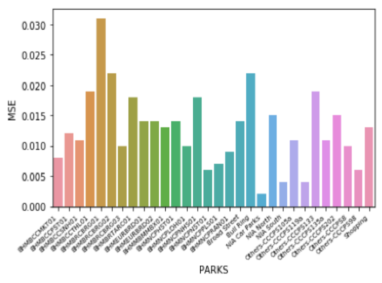
Similarly, it gives a clear idea about the predication variation on each parking location using five loss functions. In the representation of each loss function, the y-axis represents the error rate and the x-axis represents the parking locations. The RMSE, MAE, MSE, MdAE, and MSLE for each park lots are shown as (a) to (e), respectively.



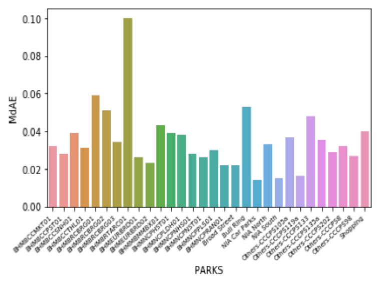
(a)



(b)



(c)

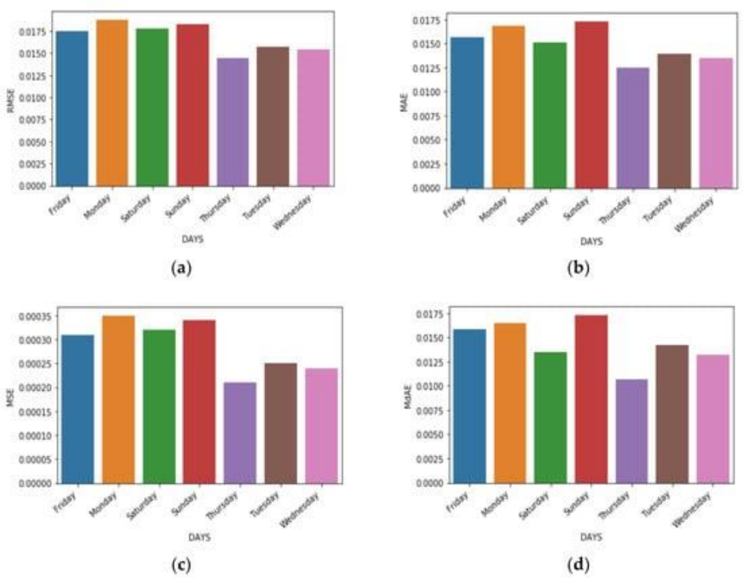


(d)

Parking availability prediction error rate against each parking location using loss functions:
(**a**) root mean square error (RMSE),
(**b**) mean absolute error (MAE),
(**c**) mean squared error (MSE),
(**d**) MdSE.

Day-Wise Parking Space Occupancy

The day-wise prediction was made to facilitate the drivers to check the availability of parking on specific day. First, we divide our dataset intoday-wise and then perform prediction day-wise through proposed model. The results produced by Deep LSTM on day-wise dataset are presented.

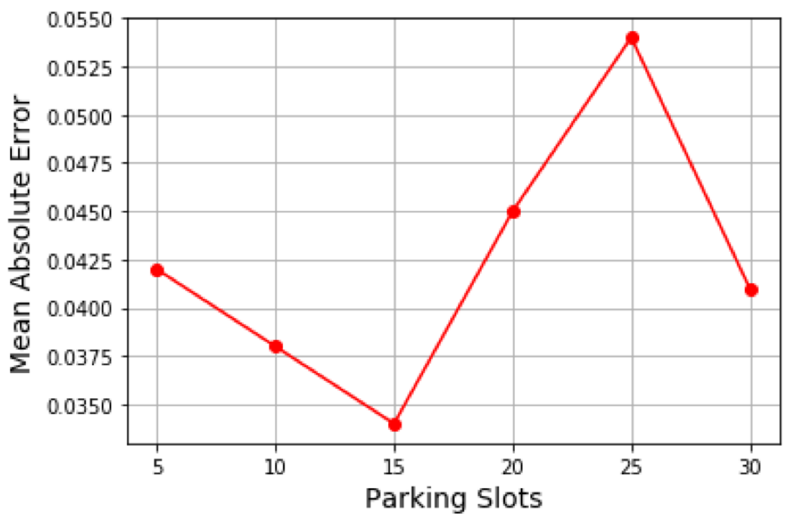


Parking availability prediction error rate with respect to day-wise using loss functions:
(**a**) RMSE,
(**b**) MAE,
(**c**) MSE,
(**d**) MdSE.

Discussion :

During the system verification, the performance of the decision support system was extensively analyzed in three different ways: location-wise, day-wise, and hour-wise. While comparing the performance of the proposed technique with existing techniques to predict the availability of parking space.

the MAE with different number of parking lots varies from 0.0335 to 0.0540. It also demonstrates that the MAE with 5, 10, 15, 20, 25, and 30 parking lots is 0.0420, 0.0380, 0.0335, 0.0450, 0.0450, and 0.410, respectively.



Performance evaluation of proposed approach by varying the parking lots count.

Conclusions

The development of IoT based smart parking information systems is one of the most demanded research problems for the growth of sustainable smart cities. It can help the drives to find a free car parking space near to their destination (market, office, or home). It will also save time and energy consumption by efficiently and accurately predicting the available car parking space.

In the future, this work will be extended to implement all the services (parking location, parking information, parking supervision, vehicle tracking, vehicle registration, and identification) described in the adopted smart parking framework. Future work will also focus on the development of various techniques to predict the real-time availability of parking lots using image and video data captured through various sensors.