MEASURE ENERGY CONSUMPTION

TEAM MEMBER

510521205012 : GOKULAKRISHNAN.T

PHASE 3 : DOCUMENT SUBMISSION

OBJECTIVES :

Certainly, I can help you get started with loading and preprocessing a dataset for measuring energy consumption in a project. The specific steps may vary depending on the dataset format and the programming tools or libraries you are using, but I can provide a general outline of the process.

Data Acquisition:

Identify the source of your dataset. It could be a CSV file, an API, a database, or any other data storage method.

Ensure you have the necessary permissions or credentials to access the data source.

Data Loading:

If your data is in a CSV file, you can use libraries like pandas in Python to load the data into a DataFrame. For other data sources, use appropriate methods or libraries.

Python code:

```
import pandas as pd

df = pd.read_csv('energy_consumption_data.csv')
```

Data Exploration:

Explore the dataset to understand its structure and contents. Check for missing values, anomalies or outliers.

Python code :

```
# Display the first few rows of the dataset
print(df.head())
# Check for missing values
print(df.isnull().sum())
# Basic statistics
print(df.describe())
```

Data Preprocessing:

Handle missing data by imputing or removing them, depending on the situation.

Convert data types if needed (e.g., dates to datetime objects).

Encode categorical variables if necessary.

Normalize or scale features if required.

Python code:

```
# Handle missing data
df = df.dropna()  # Remove rows with missing values
# Convert date column to datetime
df['date'] = pd.to_datetime(df['date'])
# Encode categorical variables (if applicable)
df = pd.get_dummies(df, columns=['category'])
# Normalize numerical features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
df['energy_consumption'] =
scaler.fit_transform(df['energy_consumption'].values.reshape(-1, 1))
```

Data Splitting:

If you're building a predictive model, split your dataset into training and testing subsets.

Python code :

```
from sklearn.model_selection import train_test_split

X = df.drop('energy_consumption', axis=1)

y = df['energy_consumption']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Feature Engineering:

Create new features if needed to enhance your analysis or modeling.

Save Preprocessed Data:

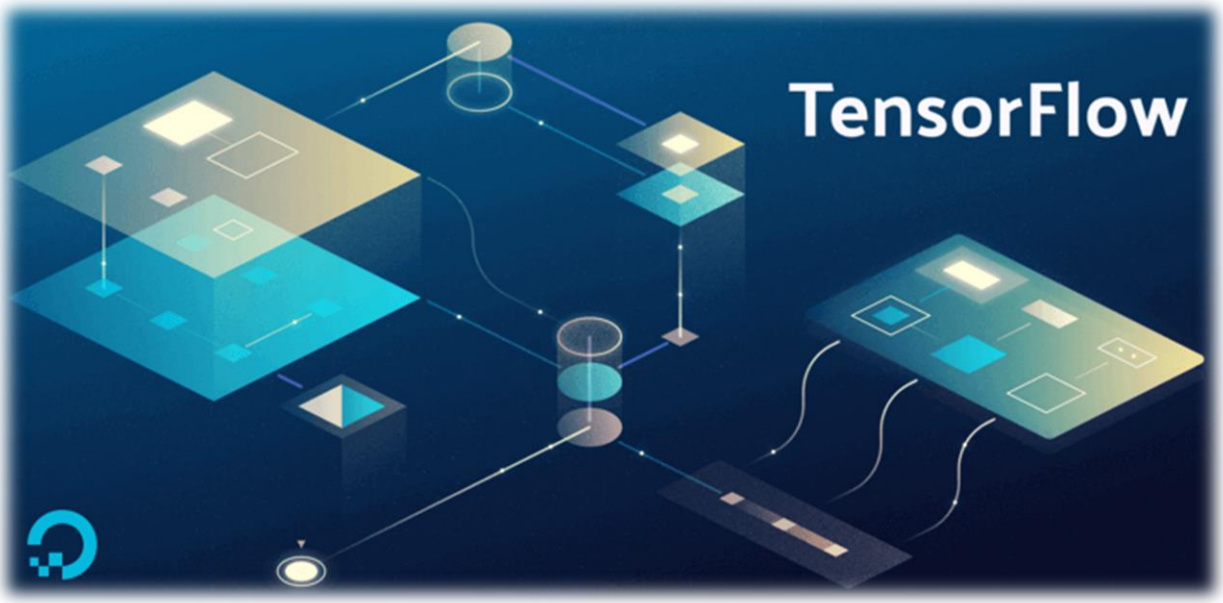
Save the preprocessed data to a new file if you wish to reuse it without going through preprocessing again.

Example :

```
df.to_csv('preprocessed_energy_data.csv', index=False)
```

These are the general steps to load and preprocess a dataset for measuring energy consumption. Depending on your specific project requirements and dataset characteristics, you might need to perform additional data cleaning and preprocessing steps.

TENSORFLOW & KERAS-ANN



TensorFlow and Keras can be used to build artificial neural networks (ANNs) for various tasks, including measuring energy consumption. The basic idea is to use historical data to train an ANN to predict energy consumption based on input features such as temperature, time of day, and other relevant factors.

Data Collection:

Gather historical data on energy consumption. This data should include the target variable (energy consumption) and relevant features (e.g., temperature, time, day of the week, and any other factors that may impact energy consumption).

Data Preprocessing:

Clean and preprocess the data. This includes handling missing values, scaling numerical features, encoding categorical features, and splitting the data into training and testing sets.

Building the Artificial Neural Network:

Import TensorFlow and Keras.

Define the architecture of your neural network. The architecture depends on the complexity of your problem and the nature of your data. You can start with a simple feedforward neural network. Consider the number of layers, neurons in each layer, activation functions, and output layer design.

Compile the model, specifying the loss function and optimizer. Since this is a regression problem (predicting a continuous value), you can use Mean Squared Error (MSE) as the loss function.

Python code :

```
import tensorflow as tf
from tensorflow import keras
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(input_dim,)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)  # Output layer for regression
])
model.compile(loss='mean_squared_error', optimizer='adam')
```

Model Training :

Train the model using your training data. Monitor the training process and adjust hyperparameters as needed.

Python code :

```
model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_test, y_test))
```

Model Evaluation:

Evaluate the model on your test data to assess its performance.

Python code :

```
loss = model.evaluate(X_test, y_test)
```

Predict Energy Consumption:

Use the trained model to make predictions on new or unseen data. Input the relevant features (e.g., temperature, time) to get the predicted energy consumption.

```
predicted_energy_consumption = model.predict(new_data)
```

Deployment:

If the model performs well, you can deploy it to make real-time predictions. You can build a web application or API to serve these predictions to users or integrate them into energy management systems.

Continuous Monitoring and Maintenance :

Keep the model up-to-date with new data to ensure its accuracy remains relevant over time.

Note that this is a simplified overview, and building an energy consumption prediction model can be a complex task. The quality and quantity of your data, feature engineering, and model architecture play crucial roles in achieving accurate predictions. It's also important to consider potential ethical and privacy issues when dealing with energy consumption data.

CONVOLUTIONAL NEURAL NETWORKS

Using convolutional neural networks (CNNs) to measure energy consumption is an innovative application that can provide valuable insights and help optimize energy usage. Although I don't have information about specific models or methodologies beyond my last training data in September 2021, I can provide a general guideline on how CNNs can be used in this context.

Data Collection:

Collect data on energy consumption over a period of time. This data should include various parameters like power usage, temperature, time of day, and other relevant factors.

Data Preprocessing:

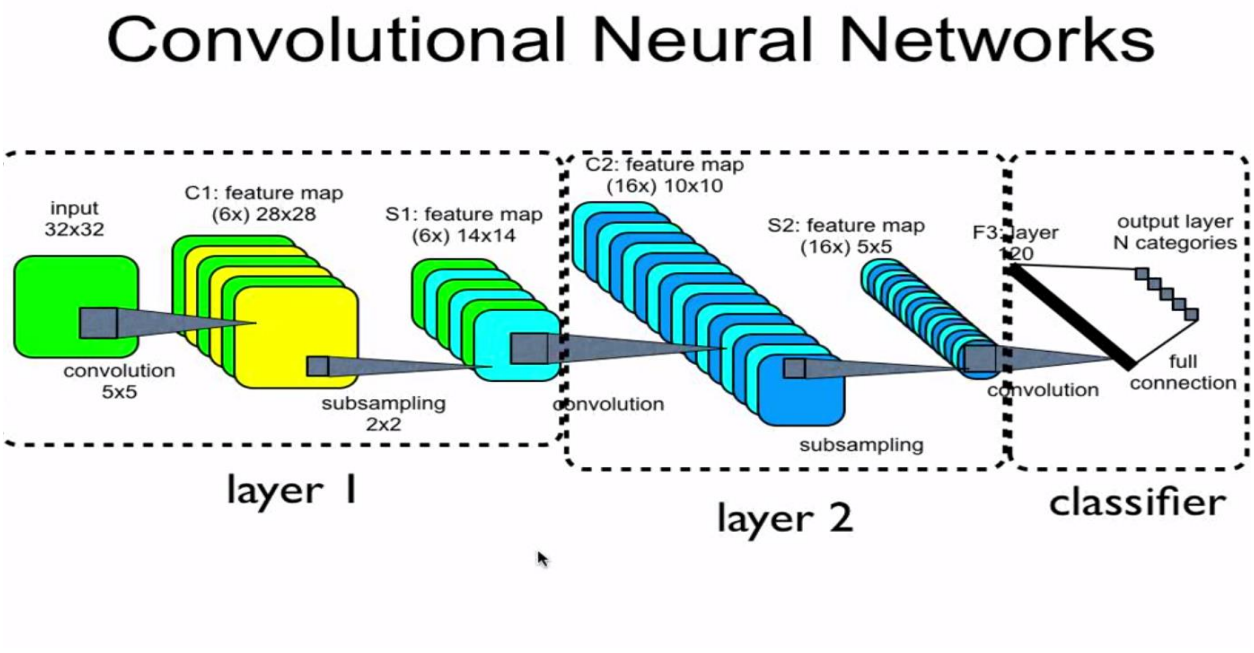
Prepare the data for training by cleaning and formatting it appropriately. This step may involve normalization, scaling, and dealing with missing values if any.

Feature Extraction:

Extract meaningful features from the collected data. CNNs can automatically learn these features, but pre-processing might involve extracting specific features that are relevant to the energy consumption patterns.

Model Architecture:

Design a CNN architecture suitable for the energy consumption prediction task. This architecture might include several convolutional layers followed by pooling layers, fully connected layers, and an output layer for prediction.



Training:

Train the CNN model using the prepared dataset. This step involves feeding the input data into the network, computing the loss, and updating the network parameters using optimization algorithms like stochastic gradient descent or its variants.

Validation and Testing:

Evaluate the model's performance on a validation dataset to ensure that it generalizes well to unseen data. This step helps in tuning the hyperparameters of the model and avoiding overfitting.

Prediction and Analysis:

Use the trained model to predict energy consumption based on new input data. Analyze the predictions and compare them with actual values to assess the model's accuracy and effectiveness.

Optimization and Deployment:

Use the insights from the model to optimize energy usage patterns. Deploy the model in real-time to monitor and predict energy consumption in various contexts, such as industrial settings or smart home environments.

It's crucial to ensure that the dataset is diverse and representative of various energy consumption scenarios to enable the CNN model to learn robust patterns and make accurate predictions. Additionally, regularly updating the model with new data and fine-tuning it can help improve its accuracy and applicability over time.

OPEN CV

OpenCV is a popular library for computer vision tasks, and it doesn't provide direct functionality to measure energy consumption. However, you can use it in conjunction with other tools and techniques to indirectly measure energy consumption.

One way to measure the energy consumption of a computer vision task using OpenCV is to utilize system-level tools or third-party libraries that can monitor the energy consumption of your system. For example, you can use tools like the Intel Power Gadget or the Linux powerstat command to measure the power consumption of your system during the execution of the OpenCV code.

Here's an example of how you can use the psutil library in Python to monitor the system's power consumption during the execution of an OpenCV script.

pytho code :

```
import psutil
import time
import cv2
# Function to measure system power consumption
def measure_power_consumption(duration):
    start_time = time.time()
    end_time = start_time + duration
    power_measurements = []
    while time.time() < end_time:
      power_measurements.append(psutil.sensors_battery().power_plugged)
        time.sleep(1)
    return power_measurements

# Example OpenCV code
def example_opencv_code():
    # Your OpenCV code here
    # For example, reading an image
    img = cv2.imread('example.jpg')

    # Perform some image processing tasks
    # For example, convert the image to grayscale
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Measure energy consumption during OpenCV execution
```

```
power_measurements = measure_power_consumption(10)
 # Measure for 10 seconds
print("Power consumption measurements: ", power_measurements)
```

CONCLUSION :

This example demonstrates how to monitor the system's power consumption using the psutil library while an OpenCV script is running. You can customize the duration of the power consumption measurement according to your requirements.

Remember that the energy consumption might not be precisely due to OpenCV alone, but the overall system load. Therefore, it's essential to consider the other processes running on the system during the measurement.