PROGRAM 11

AIM- Implementation of Autoencoders on Image Dataset.(Use MNIST Dataset)

THEORY-

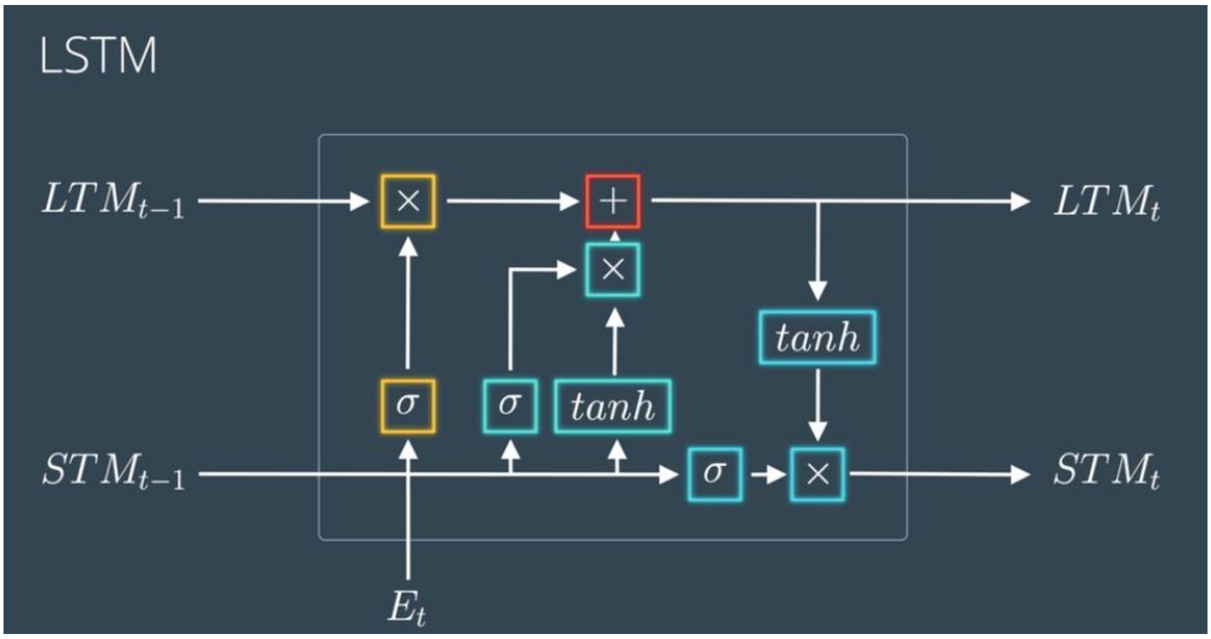
Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to address the vanishing gradient problem that standard RNNs often face when learning from long sequences. LSTMs are particularly useful in sequence-based tasks such as time-series forecasting, natural language processing, and speech recognition.

Here's a breakdown of the LSTM model theory:

1. The Challenge of Long-Term Dependencies in RNNs

Standard RNNs process sequences by passing hidden states from one time step to the next. However, as the sequences become longer, it becomes increasingly difficult for RNNs to retain information from earlier time steps due to the vanishing gradient problem. This results in a loss of long-term dependencies, which LSTMs are explicitly designed to handle.

2. LSTM Architecture



An LSTM cell consists of several components designed to maintain long-term dependencies:

- Cell State (Ct)(C_t)(Ct): This state acts as a "memory" for the LSTM, carrying long-term information through time steps. The cell state is modified by various gates to retain or discard information as needed.
- Hidden State (ht)(h_t)(ht): The hidden state is the short-term memory of the LSTM, containing information that is output at each time step.
- Forget Gate (ft)(f_t)(ft): Determines what proportion of information from the previous cell state (Ct−1)(C_{t-1})(Ct−1) should be kept. The forget gate uses a sigmoid

activation function to produce values between 0 and 1, where 0 means "forget everything" and 1 means "keep everything."

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- Input Gate (it)(i_t)(it): Controls how much of the new information (candidate value) will enter the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

- Candidate Layer (C~t)(\tilde{C}_t)(C~t): Suggests new information that might be added to the cell state.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- Output Gate (ot)(o_t)(ot): Determines how much of the cell state will be exposed to the next hidden state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

3. Updating Cell State

The LSTM cell updates its cell state with the following rule:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

This formula retains some of the old information (scaled by the forget gate) and adds new information (scaled by the input gate and modified by the candidate layer).

4. Generating Output and Hidden State

The hidden state hth_tht at each time step is updated based on the cell state and output gate:

$$h_t = o_t * \tanh(C_t)$$

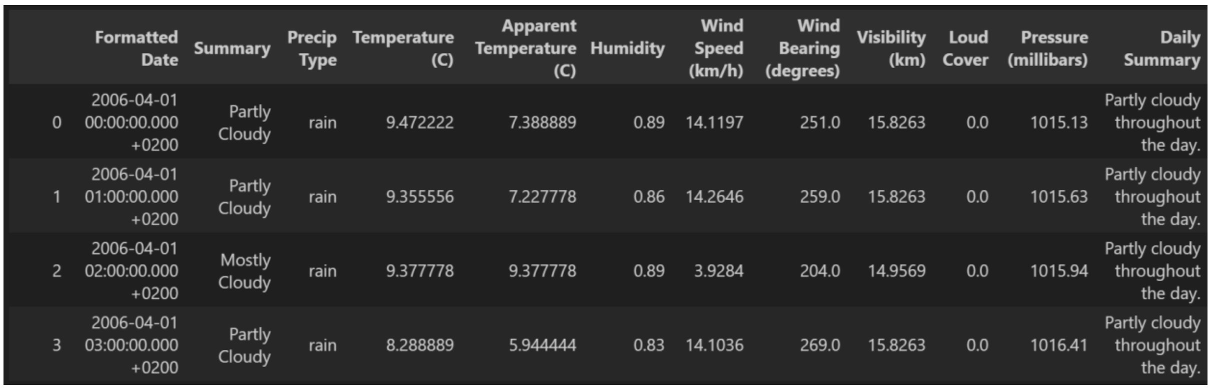
This hidden state can then be used for prediction or fed into another LSTM layer if the model is stacked.

5. Advantages of LSTM Networks

- **Long-Term Dependency**: LSTMs are adept at capturing long-term dependencies, making them useful for time-series analysis and NLP tasks.
- **Better Gradient Flow**: LSTMs address the vanishing gradient issue by controlling information flow through the gating mechanism.
- **Flexible Memory Mechanism**: By tuning gates, LSTMs dynamically decide what information to retain or forget, making them adaptable to various sequence lengths.

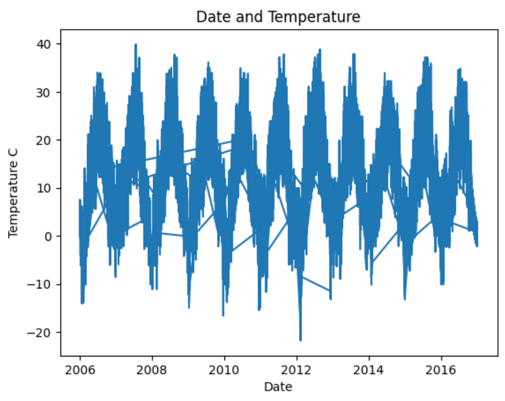
CODE AND OUTPUT-

```
import pandas as pd

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow import keras

df= pd.read_csv("weatherHistory.csv")

df
```

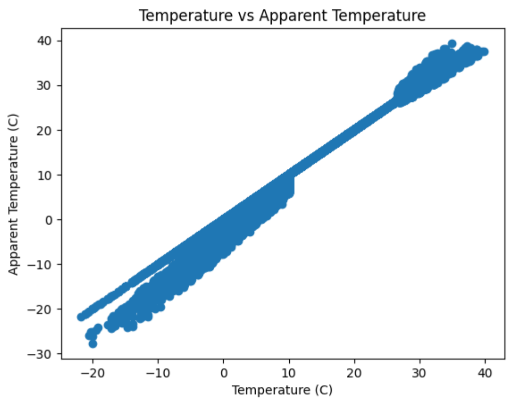


	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.

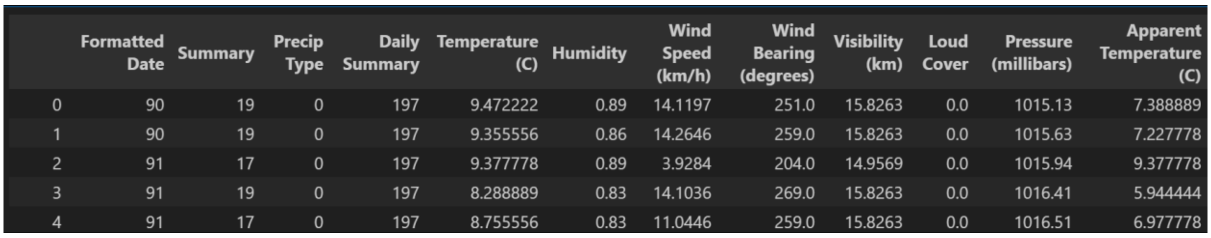
```
df['Formatted Date']=pd.to_datetime(df['Formatted Date'],utc=True)

df['Formatted Date'] = df['Formatted Date'].dt.strftime('%Y-%m-%d')

plt.plot(df["Formatted Date"],df['Temperature (C)'])

plt.title('Date and Temperature')

plt.xlabel('Date')

plt.ylabel("Temperature C")

plt.show()
```



```
plt.scatter(df['Temperature (C)'], df['Apparent Temperature (C)'] )

plt.title('Temperature vs Apparent Temperature')

plt.xlabel('Temperature (C)')

plt.ylabel('Apparent Temperature (C)')

plt.show()
```

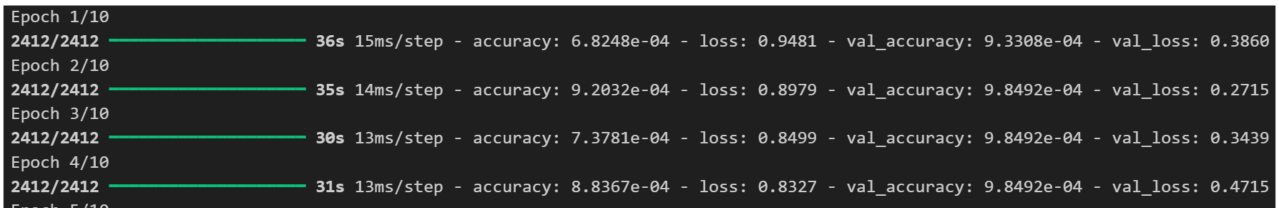


```
from sklearn.preprocessing import LabelEncoder

from sklearn.compose import ColumnTransformer

columns_to_encode=['Formatted Date', 'Summary', 'Precip Type', 'Daily Summary']

label_encoders={}

for i in columns_to_encode:

   le=LabelEncoder()

   df[i]=le.fit_transform(df[i])

   label_encoders[i]=le
```



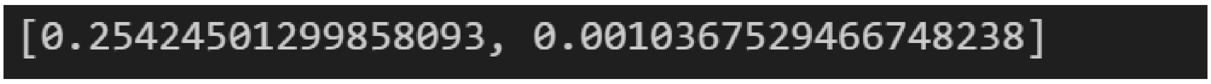
	Formatted Date	Summary	Precip Type	Daily Summary	Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Apparent Temperature (C)
0	90	19	0	197	9.472222	0.89	14.1197	251.0	15.8263	0.0	1015.13	7.388889
1	90	19	0	197	9.355556	0.86	14.2646	259.0	15.8263	0.0	1015.63	7.227778
2	91	17	0	197	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	9.377778
3	91	19	0	197	8.288889	0.83	14.1036	269.0	15.8263	0.0	1016.41	5.944444
4	91	17	0	197	8.755556	0.83	11.0446	259.0	15.8263	0.0	1016.51	6.977778

```
x=df.iloc[:,:-1].values

y=df.iloc[:,-1].values

x = x.reshape(x.shape[0], x.shape[1], 1)

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)

x_train.shape,x_test.shape,y_train.shape,y_test.shape

from tensorflow.keras.layers import *

from tensorflow.keras import *

import tensorflow

input_layer = Input(shape=(x_train.shape[1], 1))

x=LSTM(50, return_sequences=True)(input_layer)

x=Dropout(0.2)(x)

x=LSTM(50, return_sequences=False)(x)

x=Dropout(0.2)(x)

output=Dense(1)(x)

model = tensorflow.keras.Model(inputs=input_layer, outputs=output)

model.compile(metrics=["accuracy"],loss="mean_absolute_error",optimizer='adam')

history=model.fit(x_train,y_train,validation_data=[x_test,y_test],epochs=10,batch_size=32)
```



```
Epoch 1/10
2412/2412 ━━━━━━━━━━━━━━━━━━━━ 36s 15ms/step - accuracy: 6.8248e-04 - loss: 0.9481 - val_accuracy: 9.3308e-04 - val_loss: 0.3860
Epoch 2/10
2412/2412 ━━━━━━━━━━━━━━━━━━━━ 35s 14ms/step - accuracy: 9.2032e-04 - loss: 0.8979 - val_accuracy: 9.8492e-04 - val_loss: 0.2715
Epoch 3/10
2412/2412 ━━━━━━━━━━━━━━━━━━━━ 30s 13ms/step - accuracy: 7.3781e-04 - loss: 0.8499 - val_accuracy: 9.8492e-04 - val_loss: 0.3439
Epoch 4/10
2412/2412 ━━━━━━━━━━━━━━━━━━━━ 31s 13ms/step - accuracy: 8.8367e-04 - loss: 0.8327 - val_accuracy: 9.8492e-04 - val_loss: 0.4715
```

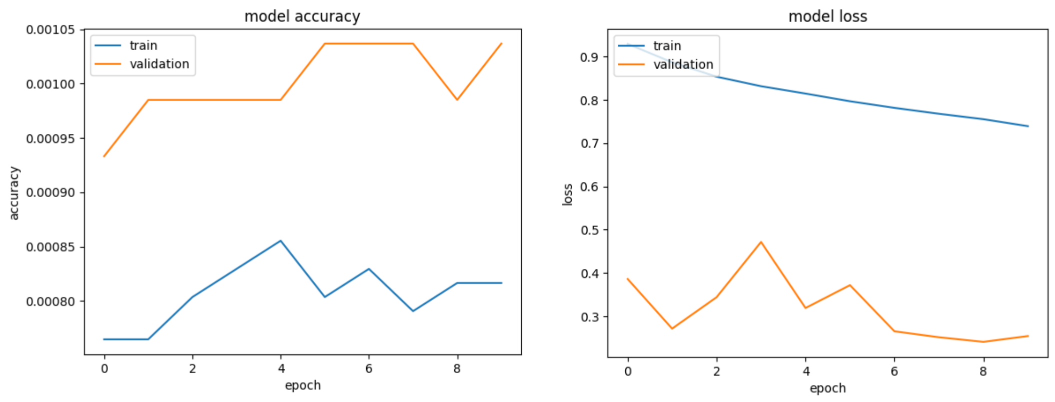
```
model.evaluate(tf.convert_to_tensor(x_test,np.float32),tf.convert_to_tensor(y_test,np.float32))
```



```
[0.25424501299858093, 0.0010367529466748238]
```

```
def plot_curves(history):

   plt.plot(history.history['accuracy'])
```

```
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
    # "Loss"
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
plot_curves(history)
```



VIVA VOICE-

1. **What problem does the LSTM architecture aim to solve in sequential models, and how does it differ from a standard RNN?**

   Explores understanding of the limitations of RNNs and why LSTMs were developed to address issues like the vanishing gradient problem.

2. **Describe the role of the cell state in an LSTM. How is it maintained across time steps?**

   Tests understanding of how the cell state functions as a memory system, storing long-term dependencies through the sequence.

3. **What are the functions of the forget, input, and output gates in an LSTM cell?**

   Ensures a detailed understanding of each gate and how it manages the flow of information within the LSTM cell.

4. **How is the hidden state computed in an LSTM, and how does it relate to the cell state?**

   Examines the mathematical relationship between the hidden state and cell state, clarifying how they work together to retain and pass information.

5. **Explain the vanishing gradient problem in the context of RNNs. How do LSTMs help mitigate this issue?**

   Focuses on understanding the vanishing gradient problem and how LSTMs' architecture allows them to learn long-term dependencies more effectively.