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."

ft=σ(Wf⋅[ht−1,xt]+bf)f\_t = \sigma(W\_f \cdot [h\_{t-1}, x\_t] + b\_f)ft​=σ(Wf​⋅[ht−1​,xt​]+bf​)

* Input Gate (it)(i\_t)(it​): Controls how much of the new information (candidate value) will enter the cell state.

it=σ(Wi⋅[ht−1,xt]+bi)i\_t = \sigma(W\_i \cdot [h\_{t-1}, x\_t] + b\_i)it​=σ(Wi​⋅[ht−1​,xt​]+bi​)

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

C~t=tanh⁡(WC⋅[ht−1,xt]+bC)\tilde{C}\_t = \tanh(W\_C \cdot [h\_{t-1}, x\_t] + b\_C)C~t​=tanh(WC​⋅[ht−1​,xt​]+bC​)

* Output Gate (ot)(o\_t)(ot​): Determines how much of the cell state will be exposed to the next hidden state.

ot=σ(Wo⋅[ht−1,xt]+bo)o\_t = \sigma(W\_o \cdot [h\_{t-1}, x\_t] + b\_o)ot​=σ(Wo​⋅[ht−1​,xt​]+bo​)

3. Updating Cell State

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

Ct=ft∗Ct−1+it∗C~tC\_t = f\_t \ast C\_{t-1} + i\_t \ast \tilde{C}\_tCt​=ft​∗Ct−1​+it​∗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).

CODE AND OUTPUT-

import pandas as pd

import numpy as np

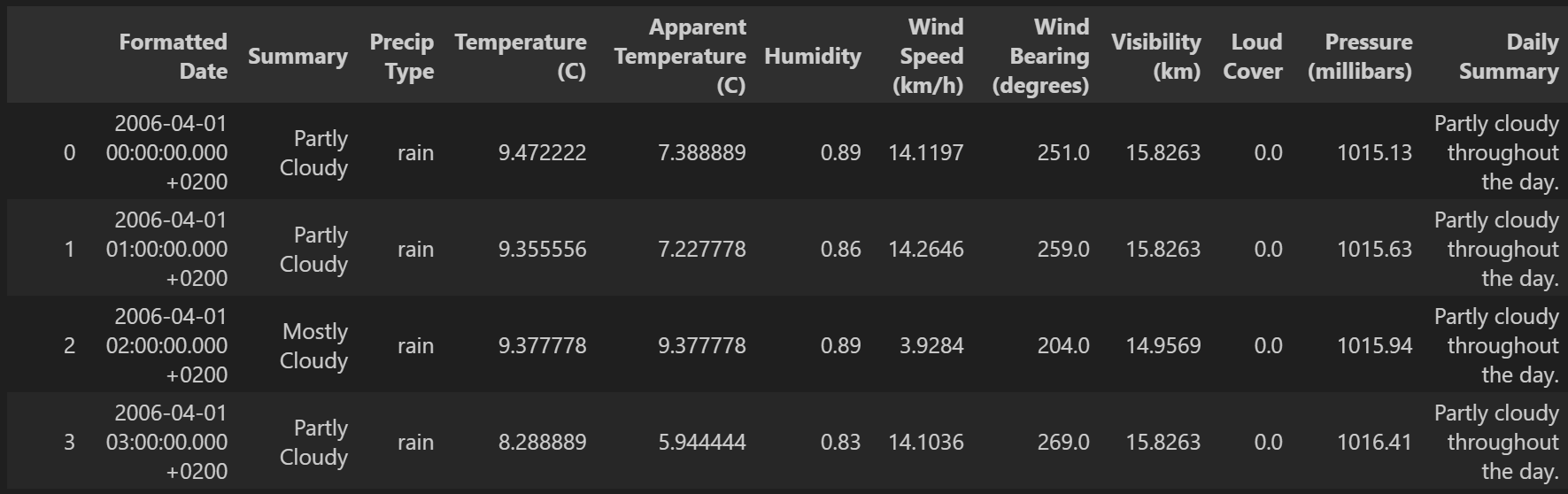
import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow import keras

df= pd.read\_csv("weatherHistory.csv")

df



df['Formatted Date']=pd.to\_datetime(df['Formatted Date'],utc=True)

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

from sklearn.preprocessing import LabelEncoder

from sklearn.compose import ColumnTransformer

columns\_to\_encode=['Formatted Date', 'Summary', 'Precip Type', 'Daily Summary']

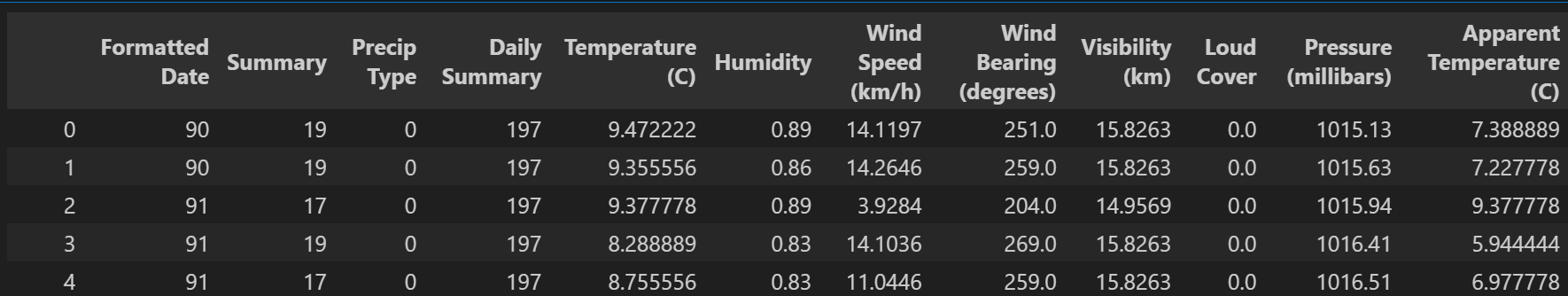
label\_encoders={}

for i in columns\_to\_encode:

    le=LabelEncoder()

    df[i]=le.fit\_transform(df[i])

    label\_encoders[i]=le



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)

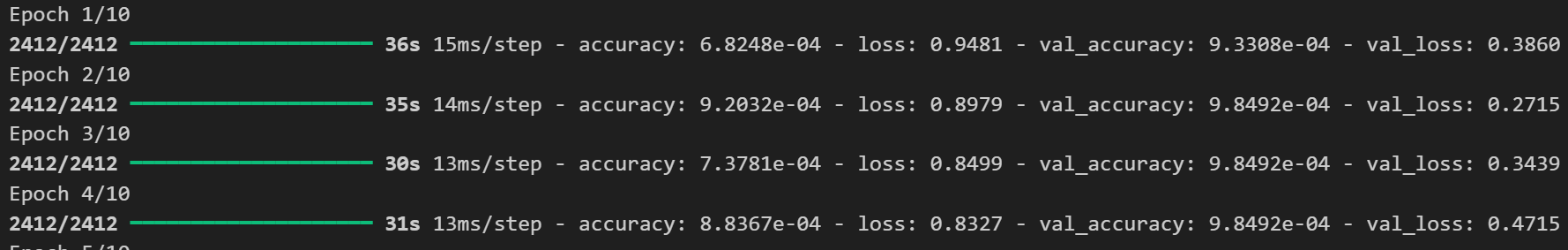
x=Dropout(0.2)(x)

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)



model.evaluate(tf.convert\_to\_tensor(x\_test,np.float32),tf.convert\_to\_tensor(y\_test,np.float32))



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'])

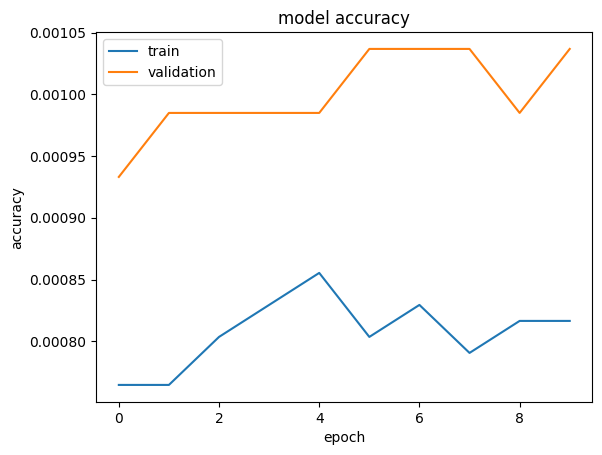
    plt.title('model loss')

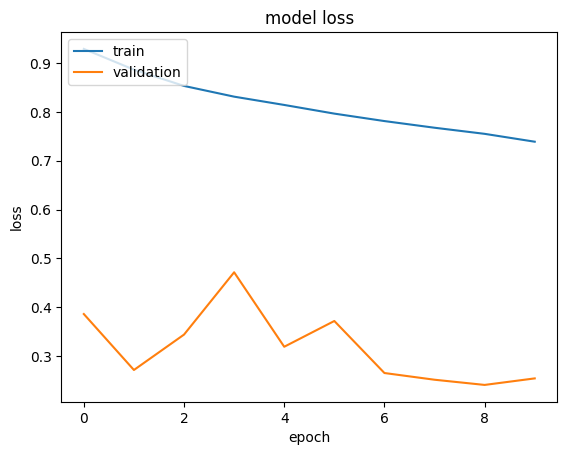
    plt.ylabel('loss')

    plt.xlabel('epoch')

    plt.legend(['train', 'validation'], loc='upper left')

    plt.show()

plot\_curves(history)



VIVA VOICE-

1. What are the main limitations of LSTMs, and in what situations might you prefer other models?
   * Tests knowledge of LSTM limitations, such as computational cost, and when alternatives like GRUs or Transformers may be more efficient.
2. How does an LSTM update the cell state using the candidate layer, and why is a tanh activation used here?
   * Examines understanding of the cell state update equation and the importance of the tanh activation function to regulate values within a certain range.
3. What is backpropagation through time (BPTT), and how is it applied in LSTM networks?
   * Tests understanding of how gradients are propagated in sequence models and the specific challenges in training LSTMs over long sequences.
4. How do you handle overfitting in LSTMs, especially when working with smaller datasets?
   * Focuses on practical techniques like dropout, regularization, and data augmentation to prevent overfitting in LSTMs.
5. Can you explain an application where LSTMs are commonly used and how the architecture benefits this application?

* Tests knowledge of practical applications (e.g., sentiment analysis, time-series prediction) and the importance of LSTMs' ability to capture temporal patterns in data.