Ramaiah Institute of Technology

(Autonomous Institute Affiliated to VTU)

Department of Information Science and Engineering

DEEP LEARNING - ASSIGNMENT II

Course Code:ISE741

Credits: 3:0:0

Name of the Facilitator: **Dr. Vijaya Kumar .**

Department of ISE

Submitted By:

Om Anish.T -1MS19IS084

Sunkara Ayodhya Rami Reddy - 1MS19IS122

Build a deep learning model that would learn from Environmental Sensor Telemetry Data and predict the ppm of smoke based on the data recorded

Introduction

Telemetry, or wireless communication, is a useful tool for real-time environmental monitoring. Common telemetry options are cellular and radio, though satellite telemetry can be used in more remote locations. The deciding factor when determining the most cost-effective telemetry option should be the local site conditions and proximity to a project computer. All three of these options permit real-time updates regarding water quality during a dredging operation.

EXECUTION OF THE PROGRAM :

Step 1: Exploring and Processing the Data

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn import preprocessing
    from keras.models import Sequential
    from keras.layers import Dense
    from sklearn.model_selection import train_test_split
```

-> Importing dataset.

```
[ ] df = pd.read_csv("iot_telemetry_data.csv")
```

-> Data preprocessing

1. Dropping unwanted columns.

```
[ ] df.drop("device",inplace=True,axis=1)
```

```
[ ] df
```

	ts	co	humidity	light	lpg	motion	smoke	temp
0	1.594512e+09	0.004956	51.000000	False	0.007651	False	0.020411	22.700000
1	1.594512e+09	0.002840	76.000000	False	0.005114	False	0.013275	19.700001
2	1.594512e+09	0.004976	50.900000	False	0.007673	False	0.020475	22.600000
3	1.594512e+09	0.004403	76.800003	True	0.007023	False	0.018628	27.000000
4	1.594512e+09	0.004967	50.900000	False	0.007664	False	0.020448	22.600000
...	...	...	...	...	...	...	...	...
405179	1.595203e+09	0.003745	75.300003	False	0.006247	False	0.016437	19.200001
405180	1.595203e+09	0.005882	48.500000	False	0.008660	False	0.023301	22.200000
405181	1.595203e+09	0.004540	75.699997	True	0.007181	False	0.019076	26.600000
405182	1.595203e+09	0.003745	75.300003	False	0.006247	False	0.016437	19.200001
405183	1.595203e+09	0.005914	48.400000	False	0.008695	False	0.023400	22.200000

405184 rows × 8 columns

2. Label encoding certain columns.

```
[ ] from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    df['light'] = le.fit_transform(df['light'])
    df['motion'] = le.fit_transform(df['motion'])
```

3. Converting data into arrays for the machine to process.

```
[ ] #converting dataframe into array for simpler calculations
```

```
[ ] data = df.values
```

```
[ ] data

    array([[1.59451209e+09, 4.95593865e-03, 5.10000000e+01, ...,
            0.00000000e+00, 2.04112701e-02, 2.27000000e+01],
           [1.59451209e+09, 2.84008861e-03, 7.60000000e+01, ...,
            0.00000000e+00, 1.32748367e-02, 1.97000008e+01],
           [1.59451210e+09, 4.97601234e-03, 5.09000000e+01, ...,
            0.00000000e+00, 2.04751256e-02, 2.26000000e+01],
           ...,
           [1.59520342e+09, 4.54046175e-03, 7.56999969e+01, ...,
            0.00000000e+00, 1.90759590e-02, 2.66000004e+01],
           [1.59520342e+09, 3.74462805e-03, 7.53000031e+01, ...,
            0.00000000e+00, 1.64367444e-02, 1.92000008e+01],
           [1.59520342e+09, 5.91448198e-03, 4.84000000e+01, ...,
            0.00000000e+00, 2.33995965e-02, 2.22000000e+01]])
```

4. Using min-max scaler to scale the dataset so that the input features lie between 0 and 1 inclusive.

```
[ ] min_max_scaler = preprocessing.MinMaxScaler()
    X_scale = min_max_scaler.fit_transform(X)
    y_scale = min_max_scaler.fit_transform(y)
```

Step-2: Building and training the model

1. Working the Keras sequential model.

```
[ ] model = Sequential([
        Dense(32, activation='relu',kernel_initializer='normal', input_shape=(7,)),
        Dense(32, activation='relu'),
         Dense(32, activation='relu'),

        Dense(1, activation='sigmoid'),
    ])
```

We have stored our model in the variable ‘model’, and we’ll describe it sequentially (layer by layer) in between the square brackets.

We have our first layer as a dense layer with 32 neurons, ReLU activation and the input shape is 7 since we have 7 input features.

The second layer is also a dense layer with 32 neurons, ReLU activation. Here we do not have to describe the input shape since Keras can infer from the output of our first layer.

```
Dense(32, activation='relu'),
```

The third layer is a dense layer with 1 neuron, sigmoid activation.

```
Dense(1, activation='sigmoid'),
```

2. Configuring the model with these settings requires us to call the function model.compile

```
[ ]
    model.compile(loss='mean_squared_error', optimizer='adam')
```

3. Training the model.

```
[ ]  hist = model.fit(X_scale, y_scale,
              batch_size=32, epochs=30)
```

4. We can evaluate it on the test set. To find the accuracy on our test set.

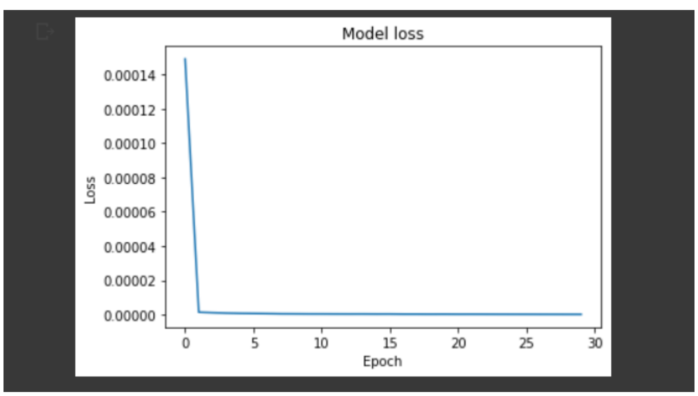
```
[ ] model.evaluate(X,y)

12662/12662 [==============================] - 18s 1ms/step
0.9618597030639648
```

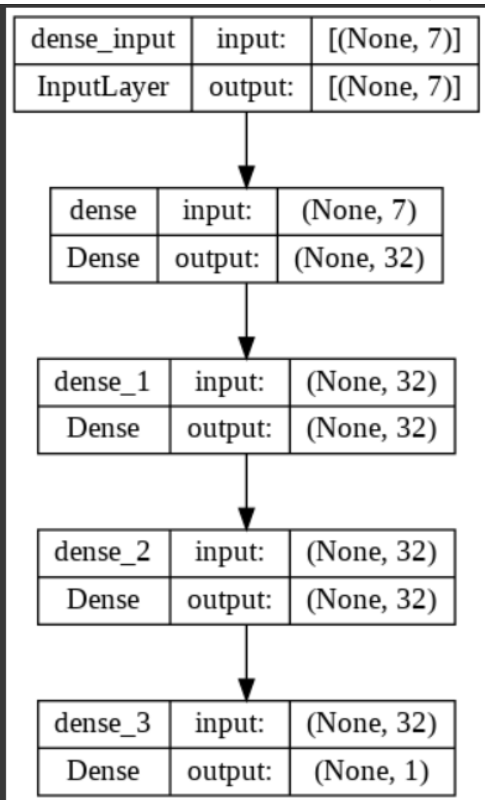
Step-3: Visualizing Loss and Accuracy

1. visualizing the training loss and the validation loss.

```
[ ] plt.plot(hist.history['loss'])
    #plt.plot(hist.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.show()
```



Step 4 - Model Summary




```
[ ] print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                256

 dense_1 (Dense)             (None, 32)                1056

 dense_2 (Dense)             (None, 32)                1056

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 2,401
Trainable params: 2,401
Non-trainable params: 0
_________________________________________________________________
None
```

Conclusion:

Smoke prediction using the environmental telemetry data plays a vital part at the time of distress. Increasing smoke in the environment is an alarm for an increase in the temperature and pollution. The availability of the related data (Humidity, temperature etc) helps the prediction of smoke and prevent further increase in smoke