Project Phase Report - 2

INTP22-ML-5: Power Line Fault Detection

Sshubam Verma

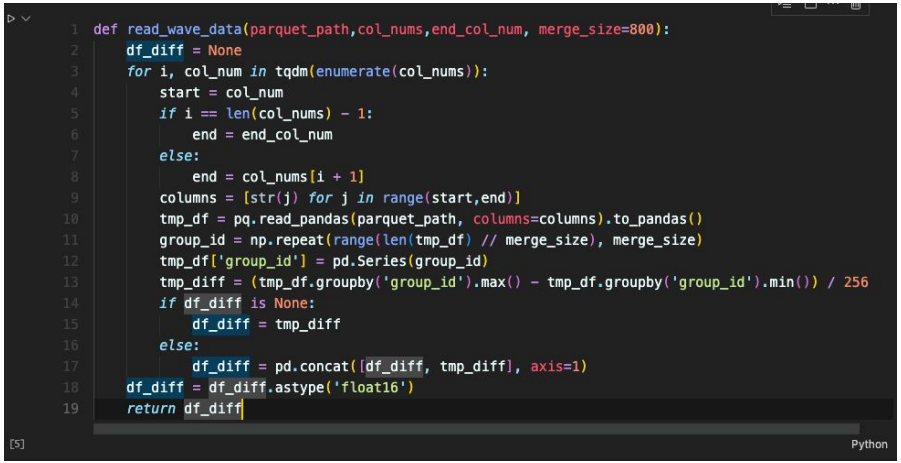
PROJECT OBJECTIVES FOR THE PHASE:

- Perform preprocessing on signals to extract features from the Data
- Build a baseline model for the signal data
- Creating the data pipeline from the signal parquet and metadata
- Fit the Model on the data pipeline optimally
- Fix code issues

PHASE - 2 PROGRESS

To load the Data, this function will convert all the train/test signals containing 800000 values to a dataframe iteratively for each column, for optimal ram utilization, then it will merge 800 signal values consecutively as a group and perform Standard Scaling on each group id and each output value is divided by 256 for normalizing the data. As the main loop runs, it concatenates every output iteratively as float values which are then ready to be fitted to the model.

This technique helps us reduce the computation power and time, also extracts the most significant features out of every signal, therefore reducing noise and potentially giving better training results.



```python
def read_wave_data(parquet_path,col_nums,end_col_num, merge_size=800):
    df_diff = None
    for i, col_num in tqdm(enumerate(col_nums)):
        start = col_num
        if i == len(col_nums) - 1:
            end = end_col_num
        else:
            end = col_nums[i + 1]
        columns = [str(j) for j in range(start,end)]
        tmp_df = pq.read_pandas(parquet_path, columns=columns).to_pandas()
        group_id = np.repeat(range(len(tmp_df) // merge_size), merge_size)
        tmp_df['group_id'] = pd.Series(group_id)
        tmp_diff = (tmp_df.groupby('group_id').max() - tmp_df.groupby('group_id').min()) / 256
        if df_diff is None:
            df_diff = tmp_diff
        else:
            df_diff = pd.concat([df_diff, tmp_diff], axis=1)
    df_diff = df_diff.astype('float16')
    return df_diff
```

[5] Python

PHASE - 2 PROGRESS

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 2000, 32)	288
max_pooling1d	(None, 1000, 32)	0
conv1d_1 (Conv1D)	(None, 1000, 64)	16448
max_pooling1d_1	(None, 500, 64)	0
conv1d_2 (Conv1D)	(None, 500, 128)	65664
max_pooling1d_2	(None, 250, 128)	0
conv1d_3 (Conv1D)	(None, 250, 256)	262400
lstm (LSTM)	(None, 64)	82176
dense (Dense)	(None, 1)	65

Total params: 427,041
Trainable params: 427,041
Non-trainable params: 0

After researching about time-series modelling approaches, Built a baseline model for the time-series signal data, The CNN-LSTM architecture has proven highest performance for this data after I learnt about various architectures like multiple folds LSTM, PCA for time-series, Attention, etc. Journal link: https://www.hindawi.com/journals/complexity/2020/6622927/

The model consists of 3 consecutive 1 dimensional convolutions followed by a max-pooling layer, then our LSTM layer and finally our classification ayer which gives the probability of possible fault in the power line. Also took some time to resolve issues with layer dimension errors.

PHASE - 2 PROGRESS

```python
def train_data_gen(metadata_train, train_diff, batch_size=128, is_reverse=False):
    np.random.seed(1)
    while True:
        x_train = []
        y_train = []
        true_sample = metadata_train[metadata_train['target']==1].sample(batch_size // 2)
        neg_sample = metadata_train[metadata_train['target']==0].sample(batch_size // 2)

        sample_signal_id = np.concatenate([true_sample['signal_id'].values,neg_sample['signal_id'
        np.random.shuffle(sample_signal_id)
        for signal_id in sample_signal_id:
            diffs = train_diff[str(signal_id)].values.T

            if is_reverse:
                diffs = diffs[::-1]

            data = diffs[:, np.newaxis]
            x_train.append(data)
            y_train.append(metadata_train[metadata_train['signal_id']==signal_id]['target'].value

        x_train = np.array(x_train)
        y_train = np.array(y_train)
        yield x_train, y_train
```

Python

Implemented a function to link the metadata with its respective Signal values, shuffled both the signal fault types (0 or 1) while generating the training while training to ensure better results, this function also is memory optimized since working iteratively for every batch size and performs at the spot, training data generation, so as to avoid ram issues. After utilizing the read_wave_data function, this will help in training generation.
Learned a lot about how to develop memory optimized functions.

Gantt Chart

[Gantt Chart LINK]

PROJECT TRACKING

PROJECT TITLE	Power Line Fault Detection	COMPANY NAME	IAFSM
PROJECT COORDINATOR	DEVESH TARASIA	DATE	01/06/2022

PROJECT DETAILS								DELIVERABLES	
STATUS	PRIORITY	START DATE	END DATE	DURATION	TASK NAME	ASSIGNEE	DESCRIPTION	DELIVERABLE	% DONE
Project Initiation, briefing and planning									**54%**
In Progress	Medium	01/06/2022	06/06/2022	5	Analysis	Sshubam Verma	Problem statement analysis		100%
In Progress	High	07/06/2022	10/06/2022	3	Research	Sshubam Verma	Read and analyse related research papers		100%
In Progress	High	11/06/2022	15/06/2022	4	Data Cleaning	Sshubam Verma	Understanding data and Exploratory Data Analysis		100%
In Progress	High	15/06/2022	24/06/2022	9	Modelling	Sshubam Verma	Model Building and Training		80%
Not Yet Started	Medium	25/06/2022	28/06/2022	3	Tuning	Sshubam Verma	Model Hyperparameter tuning		0%
Not Yet Started	High	29/06/2022	10/07/2022	11	Testing	Sshubam Verma	Model comparison and testing		0%
Not Yet Started	High	11/07/2022	25/07/2022	14	Deployment	Sshubam Verma	Model Deployment		0%
Project Submission and Presentation									**50%**
In Progress	High	05/06/2022	10/06/2022	5	Task	Sshubam Verma	Phase Report - 1		100%
In Progress	High	20/06/2022	30/06/2022	10	Task	Sshubam Verma	Phase Report - 2		100%
Not Yet Started	High	05/07/2022	10/07/2022	5	Task	Sshubam Verma	Phase Report - 3		0%
Not Yet Started	High	15/07/2022	25/07/2022	10	Task	Sshubam Verma	Phase Report - 4		0%