POWER QUALITY CLASSIFICATION USING ML

Problem Statement:

Given is a time series data corresponding to different power quality conditions. The objective is to determine a deep learning model that can classify them effectively. The input is a voltage signal and output is the condition 1 to 5.

Data Pre-Processing:

On looking at the input dataset, we can see that there are null values present.

```
In [5]: data1.tail()
Out[5]:
```

	0	1	2	3	4	5	6	7	8	9	...	118	119	120	121	122	123	124	125	126	127
11895	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11896	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11897	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11898	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11899	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 128 columns

```
In [8]: data1.isnull().sum()
Out[8]: 0      5900
        1      5900
        2      5900
        3      5900
        4      5900
               ...
        123    5900
        124    5900
        125    5900
        126    5900
        127    5900
        Length: 128, dtype: int64
```

Let us drop those null values first.

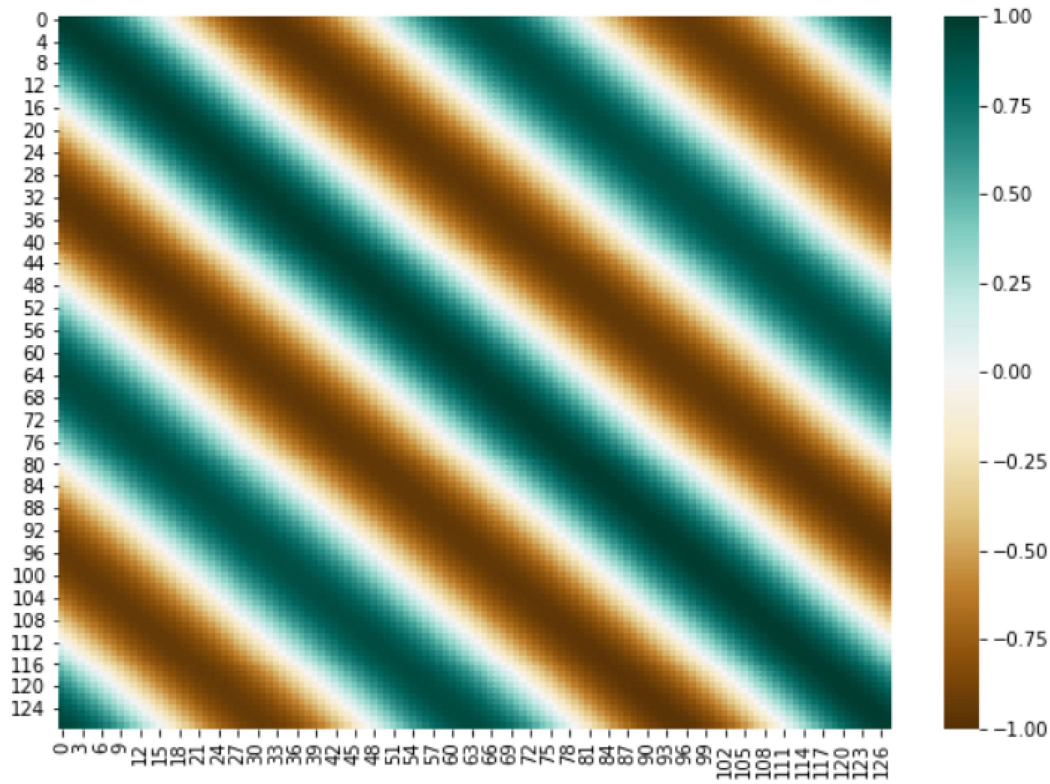
```
In [9]: data1=data1.dropna()
        data1.shape
Out[9]: (6000, 128)
```

```
In [10]: data1.isnull().sum()
Out[10]: 0      0
         1      0
         2      0
         3      0
         4      0
               ..
         123    0
         124    0
         125    0
         126    0
         127    0
         Length: 128, dtype: int64
```

Now, looking at the correlation plot, we can observe that the data features are highly correlated to each other.

```
In [13]: plt.figure(figsize=(10,7))
         sns.heatmap(data1.corr(),vmin=-1, vmax=1, cmap='BrBG')
```

```
Out[13]: <AxesSubplot:>
```

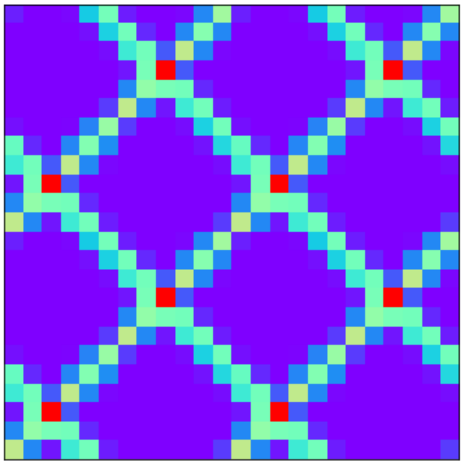


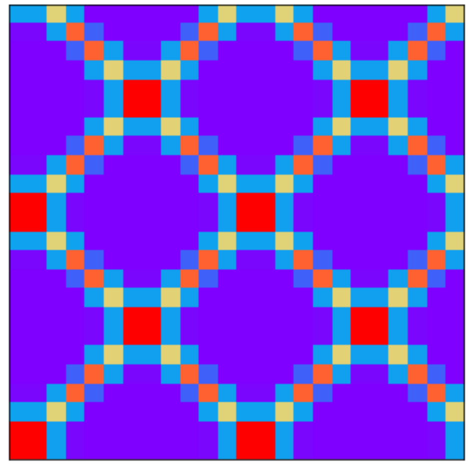
Markov Transition Fields

A Markov Transition Field is an image obtained from a time series, representing a field of transition probabilities for a discretized time series.

We had generated 12,000 MTF images representing each of the data points.

However, we had to drop the model, as the computation time and memory was significantly high.





```
In [9]: mtf = MarkovTransitionField(image_size=24)
        X_mtf = mtf.fit_transform(data)
```

```
In [ ]: for i in range(12000):
            fig,ax = plt.subplots(1)
            fig=plt.figure(figsize=(6, 6))
            plt.gca().axes.get_yaxis().set_visible(False)
            plt.gca().axes.get_xaxis().set_visible(False)
            #ax.set_yticklabels([])
            #ax.set_xticklabels([])

            plt.imshow(X_mtf[i], cmap='rainbow', origin='lower')
            save_name =  'Markov_Transition_Field_Images/'+"MTF_Image_"+str(i) + '.png'
            plt.savefig(save_name)
            plt.close(fig)
```

Building the Keras model:

Let us first split the train and test data.

```
In [23]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(data,out_data,test_size=0.33, random_state=21)
```

```
In [24]: print(x_train.shape,y_train.shape)
         print(x_test.shape,y_test.shape)

         (8040, 128) (8040, 1)
         (3960, 128) (3960, 1)
```

To build the model, we add a few **sequential layers** to the neural network and activate it using the **ReLU and Softmax** activation functions.

In [25]:
```
from keras.models import Sequential
from keras.layers import Dense,Dropout,Activation
model = Sequential()
model.add(Dense(64,activation='relu', input_dim=128))
model.add(Dense(32,activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(16,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(6, activation='softmax'))
```

In [26]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 102
=================================================================
Total params: 10,966
Trainable params: 10,966
Non-trainable params: 0
_________________________________________________________________
```

The accuracy after testing the model is shown below.

In [ ]:
```
metrics=model.evaluate(scaler.transform(x_test),np_utils.to_categorical(np.array(y_test),6),verbose=1)
print()
print("%s: %.2f%%" % (model.metrics_names[1], metrics[1]*100))
predictions = model.predict(x_test)
```

```
124/124 [==============================] - 0s 811us/step - loss: 0.3403 - accuracy: 0.8250

accuracy: 82.50%
```

To achieve a better accuracy, we fine **tuned the hyperparameters** of the neural network.

```
In [ ]: new_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_4 (Dense)              (None, 72)                9288      
_________________________________________________________________
dense_5 (Dense)              (None, 36)                2628      
_________________________________________________________________
dropout_2 (Dropout)          (None, 36)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 16)                592       
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 102       
=================================================================
Total params: 12,610
Trainable params: 12,610
Non-trainable params: 0
_________________________________________________________________
```

```
In [ ]: model2 = Sequential()
        model2.add(Dense(72,activation='relu',input_dim=128))
        model2.add(Dense(48,activation='relu'))
        model2.add(Dense(32,activation='relu'))
        model2.add(Dropout(0.3))
        model2.add(Dense(16,activation='relu'))
        model2.add(Dropout(0.2))
        model2.add(Dense(6, activation='softmax'))
```

```
In [ ]: model2.compile(optimizer="RMSprop", loss="categorical_crossentropy", metrics=["accuracy"])
        model2.fit(scaler.transform(x_train),y_train1,batch_size=32,epochs=50,verbose=1)
```

```
In [ ]: model2.save('seventh_9.h5')
```

```
In [ ]: x_train,x_test,y_train,y_test=train_test_split(data,out_data,test_size=0.33, random_state=50)
```

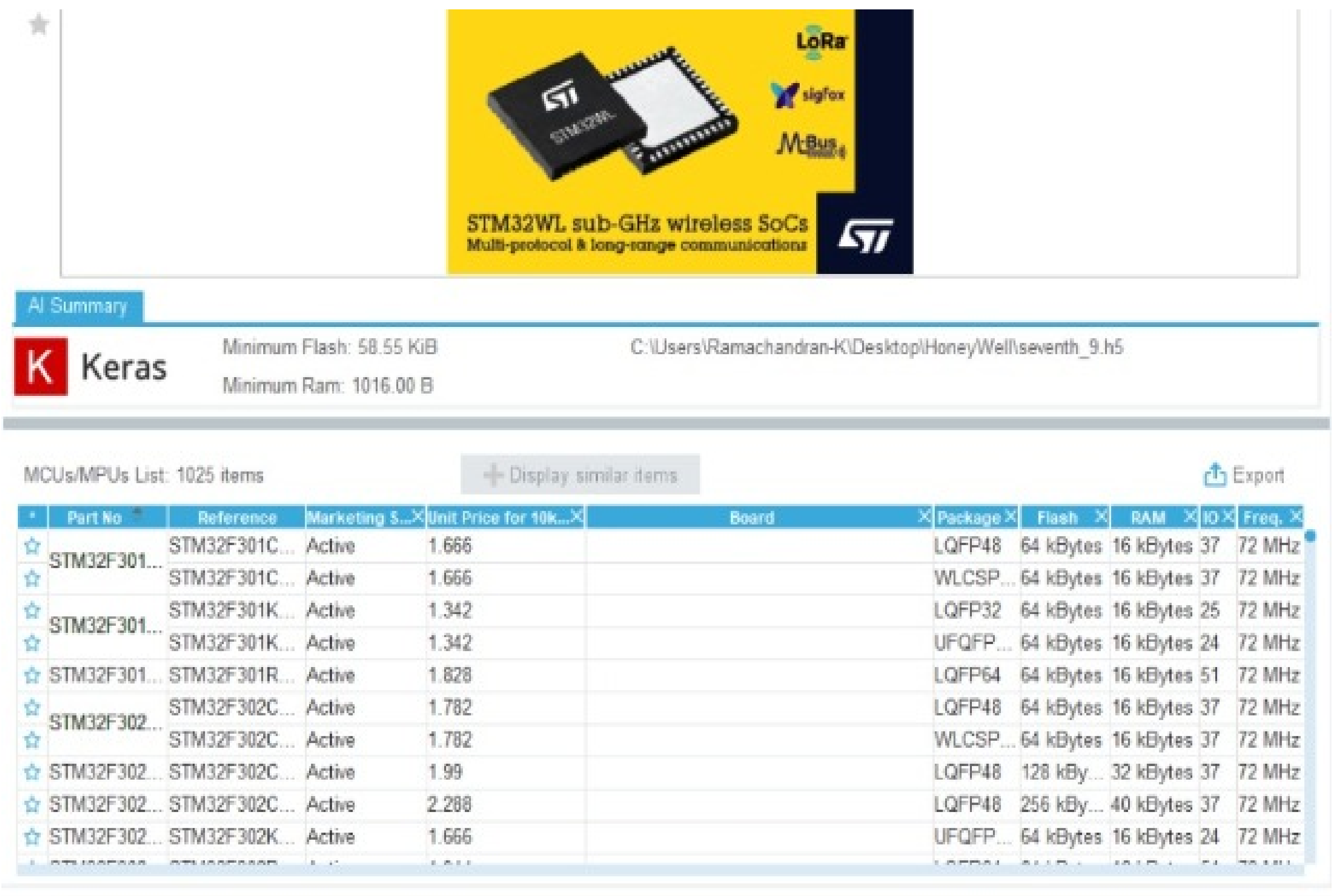
```
In [ ]: metrics=model2.evaluate(scaler.transform(x_test),np_utils.to_categorical(np.array(y_test),6),verbose=1)
        print()
        print("%s: %.2f%%" % (model2.metrics_names[1], metrics[1]*100))
        predictions = model2.predict(x_test)

124/124 [==============================] - 0s 868us/step - loss: 0.0145 - accuracy: 0.9980

accuracy: 99.80%
```

By adding a sequential layer and fine tuning the hyperparameters to the neural network we have increased the **accuracy to a staggering 99.80%, while keeping the parameters less.**

We tested the memory requirements for our Keras model using the STM32Cube software.



Part No	Reference	Marketing S...	Unit Price for 10k...	Board	Package	Flash	RAM	IO	Freq.
STM32F301...	STM32F301C...	Active	1.666		LQFP48	64 kBytes	16 kBytes	37	72 MHz
	STM32F301C...	Active	1.666		WLCSP...	64 kBytes	16 kBytes	37	72 MHz
STM32F301...	STM32F301K...	Active	1.342		LQFP32	64 kBytes	16 kBytes	25	72 MHz
	STM32F301K...	Active	1.342		UFQFP...	64 kBytes	16 kBytes	24	72 MHz
STM32F301...	STM32F301R...	Active	1.828		LQFP64	64 kBytes	16 kBytes	51	72 MHz
STM32F302...	STM32F302C...	Active	1.782		LQFP48	64 kBytes	16 kBytes	37	72 MHz
	STM32F302C...	Active	1.782		WLCSP...	64 kBytes	16 kBytes	37	72 MHz
STM32F302...	STM32F302C...	Active	1.99		LQFP48	128 kBy...	32 kBytes	37	72 MHz
STM32F302...	STM32F302C...	Active	2.288		LQFP48	256 kBy...	40 kBytes	37	72 MHz
STM32F302...	STM32F302K...	Active	1.666		UFQFP...	64 kBytes	16 kBytes	24	72 MHz

As for the Dataset 2 which has increased sample rate:

```
________________________________________________________________
Layer (type)                 Output Shape              Param #
================================================================
dense_239 (Dense)            (None, 128)               32896
________________________________________________________________
dense_240 (Dense)            (None, 64)                8256
________________________________________________________________
dense_241 (Dense)            (None, 32)                2080
________________________________________________________________
dropout_89 (Dropout)         (None, 32)                0
________________________________________________________________
dense_242 (Dense)            (None, 24)                792
________________________________________________________________
dropout_90 (Dropout)         (None, 24)                0
________________________________________________________________
dense_243 (Dense)            (None, 7)                 175
================================================================
Total params: 44,199
Trainable params: 44,199
Non-trainable params: 0
```

```
model2 = Sequential()
model2.add(Dense(128,activation='relu',input_dim=256))
model2.add(Dense(64,activation='relu'))
model2.add(Dense(32,activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(24,activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(7, activation='softmax'))
```

```
[196] model2.compile(optimizer="RMSprop", loss="categorical_crossentropy", metrics=["accuracy"])
```

```
history=model2.fit(train_sc,y_train1,batch_size=32,epochs=60,verbose=1)
```

```
metrics=new_model.evaluate(train_sc,np_utils.to_categorical(np.array(y_train),7),verbose=1)
print()
print("%s: %.2f%%" % (new_model.metrics_names[1], metrics[1]*100))
```

```
188/188 [==============================] - 0s 1ms/step - loss: 0.0450 - accuracy: 0.9830

accuracy: 98.30%
```

```
metrics=new_model.evaluate(test_sc,np_utils.to_categorical(np.array(y_test),7),verbose=
print()
print("%s: %.2f%%" % (new_model.metrics_names[1], metrics[1]*100))
```

```
113/113 [==============================] - 0s 1ms/step - loss: 0.0856 - accuracy: 0.9661

accuracy: 96.61%
```

Accuracy for the train data: 98.3%.
Accuracy for the test data: 96.61%.

The **memory requirements** for our Keras model which used the new dataset is,

AI Summary

Keras | Minimum Flash: 91.15 KiB | C:\Users\Ramachandran-K\Desktop\HoneyWell\data2_fifth_9661.h5
Minimum Ram: 1.53 KiB

MCUs/MPUs List: 969 items | Display similar items | Export

*	Part No	Reference	Marketing S...	Unit Price for 10k...	Board	Package	Flash	RAM	IO	Freq.
☆	STM32F302...	STM32F302C...	Active	1.99		LQFP48	128 kBy...	32 kBytes	37	72 MHz
☆	STM32F302...	STM32F302C...	Active	2.288		LQFP48	256 kBy...	40 kBytes	37	72 MHz
☆	STM32F302...	STM32F302R...	Active	2.152		LQFP64	128 kBy...	32 kBytes	52	72 MHz
☆	STM32F302...	STM32F302R...	Active	2.499		LQFP64	256 kBy...	40 kBytes	52	72 MHz
☆	STM32F302...	STM32F302R...	Active	2.846		LQFP64	384 kBy...	64 kBytes	51	72 MHz
☆	STM32F302...	STM32F302R...	Active	3.24		LQFP64	512 kBy...	64 kBytes	51	72 MHz
☆	STM32F302...	STM32F302V...	Active	2.522		LQFP100	128 kBy...	32 kBytes	87	72 MHz
☆	STM32F302...	STM32F302V...	Active	2.869		LQFP100	256 kBy...	40 kBytes	87	72 MHz
☆		STM32F302V...	Active	2.869		WLCSP...	256 kBy...	40 kBytes	78	72 MHz
☆	STM32F302	STM32F302V...	Active	3.216		UFBGA...	384 kBy...	64 kBytes	86	72 MHz

Final Statistics:

For Dataset 1:

Train accuracy : 99.8%

Test Accuracy : 99.8%

Minimum Flash : 58.55 KiB

Minimum Ram : 1016 B

For Dataset 2:

Train Accuracy : 98.3%

Test Accuracy : 96.61%

Minimum Flash : 91.15 KiB

Minimum Ram : 1.53 KiB