```
# Name: Alam Ali,
# Net ID: aa8007,
# Course: DL Project,

# Project Code for ACOPF solver,

from pandas import read_csv
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.layers import LeakyReLU
from tensorflow.python.keras.layers import BatchNormalization
from tensorflow.python.keras.wrappers.scikit_learn import KerasRegressor
from keras.constraints import max_norm

# load dataset

system = "14"

dataframeX = read_csv("/content/drive/MyDrive/14result_overall_in.csv", delimiter=',', header=None)
X = dataframeX.values[0:]
dataframeY = read_csv("/content/drive/MyDrive/14result_overall_out.csv", delimiter=',', header=None)
Y = dataframeY.values

# split into input (X) and output (Y) variables
print (len(X))
print (X.shape)

print (len(Y))
print (Y.shape)

nsamples = X.shape[0]
npredictors = X.shape[1]
noutvars = Y.shape[1]
print (npredictors)
print (noutvars)
print (Y.shape[1])
```

```
70677
(70677, 60)
70677
(70677, 38)
60
38
38
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cont
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(npredictors, input_dim=npredictors, kernel_initializer='normal', activation='tanh'))
model.add(Dense(10000, activation='relu'))
model.add(Dense(noutvars, activation='linear'))

model.summary()

opt = tf.keras.optimizers.Adam(learning_rate=0.00005)
model.compile(loss='mse', optimizer=opt, metrics=['mse','mae'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 60)                3660      
                                                                 
 dense_1 (Dense)             (None, 10000)             610000    
                                                                 
 dense_2 (Dense)             (None, 38)                380038    
                                                                 
=================================================================
Total params: 993,698
Trainable params: 993,698
Non-trainable params: 0
_________________________________________________________________
```

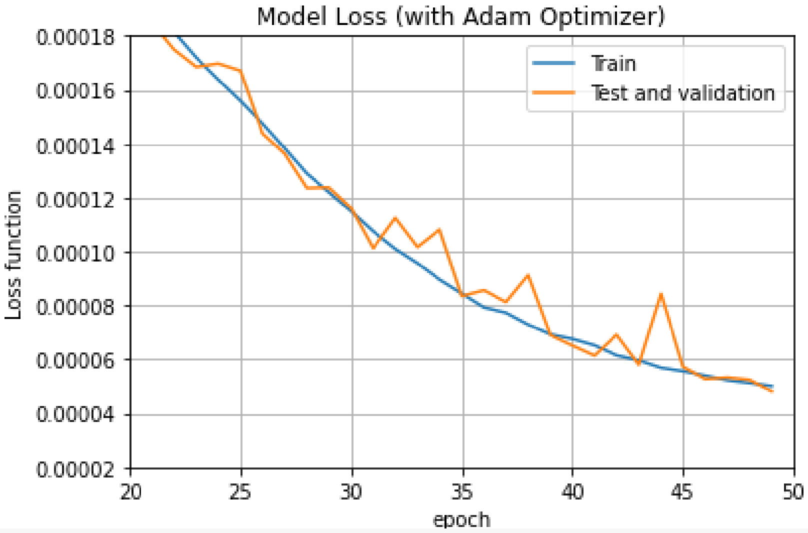
```
history = model.fit(X, Y, epochs=50, batch_size=64,  verbose=1, validation_split=0.01)
```

```
model.save(system+'busNN.h5')
```

```
print(history.history.keys())

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss (with Adam Optimizer)')
plt.xlim([20,50]);
plt.ylim([0.00002,0.00018]);
plt.ylabel('Loss function')
plt.xlabel('epoch')
plt.legend(['Train', 'Test and validation'], loc='upper right')
plt.grid()
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae'])
```



```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential() # do this every time to reset the model!
model.add(Dense(npredictors, input_dim=npredictors, kernel_initializer='normal', activation='tanh'))
model.add(Dense(10000, activation='relu'))
model.add(Dense(noutvars, activation='linear'))#,kernel_constraint=max_norm(3), bias_constraint=max_n

model.summary()
opt = tf.keras.optimizers.RMSprop(learning_rate=0.00005)
model.compile(loss='mse', optimizer=opt, metrics=['mse','mae'])
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_3 (Dense)             (None, 60)                3660      
                                                                 
 dense_4 (Dense)             (None, 10000)             610000    
                                                                 
 dense_5 (Dense)             (None, 38)                380038    
                                                                 
=================================================================
Total params: 993,698
Trainable params: 993,698
Non-trainable params: 0
_________________________________________________________________
```

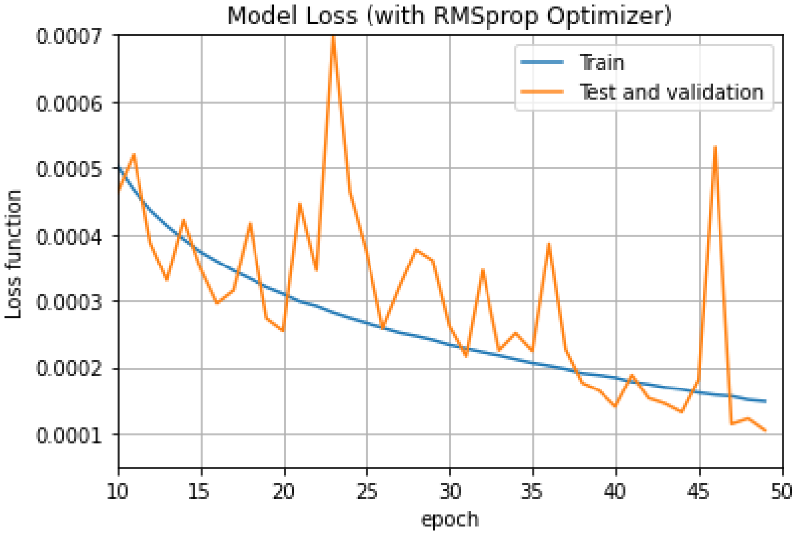
```
history = model.fit(X, Y, epochs=50, batch_size=64,  verbose=1, validation_split=0.01)
```

```
model.save(system+'busNN.h5')
```

```
print(history.history.keys())

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss (with RMSprop Optimizer)')
plt.xlim([10,50]);
plt.ylim([0.00005,0.00070]);
plt.ylabel('Loss function')
plt.xlabel('epoch')
plt.legend(['Train', 'Test and validation'], loc='upper right')
plt.grid()
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae'])
```


Model Loss (with RMSprop Optimizer)

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential() # do this every time to reset the model!
model.add(Dense(npredictors, input_dim=npredictors, kernel_initializer='normal', activation='tanh'))
model.add(Dense(10000, activation='relu'))
model.add(Dense(noutvars, activation='linear'))#,kernel_constraint=max_norm(3), bias_constraint=max_n

model.summary()
opt = tf.keras.optimizers.SGD(learning_rate=0.005)
model.compile(loss='mse', optimizer=opt, metrics=['mse','mae'])
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_39 (Dense)            (None, 60)                3660

 dense_40 (Dense)            (None, 10000)             610000

 dense_41 (Dense)            (None, 38)                380038

=================================================================
Total params: 993,698
Trainable params: 993,698
Non-trainable params: 0
_________________________________________________________________
```

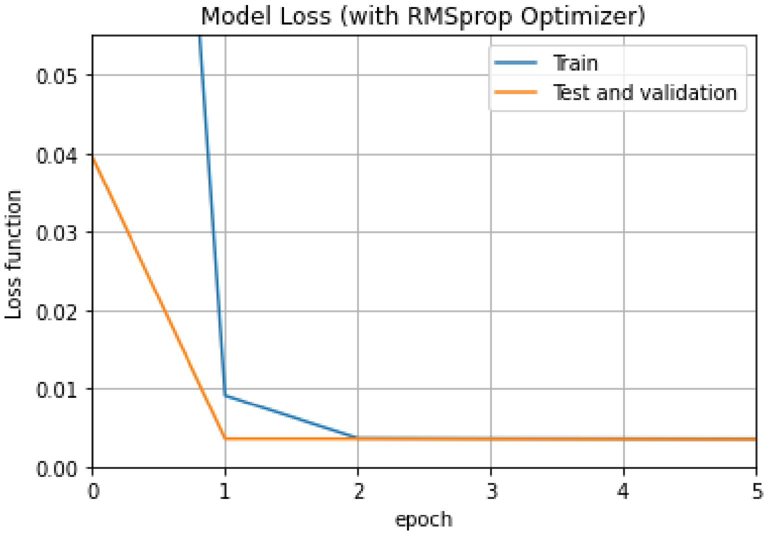
```
history = model.fit(X, Y, epochs=50, batch_size=64,  verbose=1, validation_split=0.01)
```

```
model.save(system+'busNN.h5')
```

```
print(history.history.keys())

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss (with RMSprop Optimizer)')
plt.xlim([0,5]);
plt.ylim([0.00000,0.055]);
plt.ylabel('Loss function')
plt.xlabel('epoch')
plt.legend(['Train', 'Test and validation'], loc='upper right')
plt.grid()
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae'])
```



✓ 0s completed at 3:13 PM