In [1]:
```
# importing required libraries
import numpy as np
import pandas as pd
import pickle # saving and loading trained model
from os import path

# importing required libraries for normalizing data
from sklearn import preprocessing
from sklearn.preprocessing import (StandardScaler, OrdinalEncoder,LabelEncoder, Mir
from sklearn.preprocessing import Normalizer, MaxAbsScaler , RobustScaler, PowerTra

# importing library for plotting
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import accuracy_score # for calculating accuracy of model
from sklearn.model_selection import train_test_split # for splitting the dataset fo
from sklearn.metrics import classification_report # for generating a classificatior

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

import tensorflow as tf
from tensorflow.keras.utils import to_categorical

from keras.layers import Dense, Conv1D, MaxPool1D, Flatten, Dropout # importing der
from keras.models import Sequential #importing Sequential layer
from keras.layers import Input
from keras.models import Model
# representation of model layers
from keras.utils.vis_utils import plot_model
```

In [2]:
```
features=["duration","protocol_type","service","flag","src_bytes","dst_bytes","lanc
          "num_failed_logins","logged_in","num_compromised","root_shell","su_attemp
          "num_access_files","num_outbound_cmds","is_host_login","is_guest_login","
          "rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff
          "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port
          "dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_ra
```

In [3]:
```
train='nsl-kdd/KDDTrain+.txt'
test='nsl-kdd/KDDTest+.txt'
test21='nsl-kdd/KDDTest-21.txt'
train_data=pd.read_csv(train,names=features)
train_data
```

Out[3]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urge
0	0	tcp	ftp_data	SF	491	0	0	0	
1	0	udp	other	SF	146	0	0	0	
2	0	tcp	private	S0	0	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	
4	0	tcp	http	SF	199	420	0	0	
...	...	...	...	...	...	...	...	...	
125968	0	tcp	private	S0	0	0	0	0	
125969	8	udp	private	SF	105	145	0	0	
125970	0	tcp	smtp	SF	2231	384	0	0	
125971	0	tcp	klogin	S0	0	0	0	0	
125972	0	tcp	ftp_data	SF	151	0	0	0	

125973 rows × 43 columns

In [4]:
```
train_data.drop(['difficulty'],axis=1,inplace=True)
```

In [5]:
```
test_data = pd.read_csv(test,names=features)
test_data.drop(['difficulty'],axis=1,inplace=True)
```

In [6]:
```
train_data['label'].value_counts()
```

Out[6]:
```
label
normal             67343
neptune            41214
satan               3633
ipsweep             3599
portsweep           2931
smurf               2646
nmap                1493
back                 956
teardrop             892
warezclient          890
pod                  201
guess_passwd          53
buffer_overflow       30
warezmaster           20
land                  18
imap                  11
rootkit               10
loadmodule             9
ftp_write              8
multihop               7
phf                    4
perl                   3
spy                    2
Name: count, dtype: int64
```

In [7]:
```
def change_label(df):
  df.label.replace(['apache2','back','land','neptune','mailbomb','pod','processtabl
  df.label.replace(['ftp_write','guess_passwd','httptunnel','imap','multihop','name
```

```
  df.label.replace(['ipsweep','mscan','nmap','portsweep','saint','satan'],'Probe',i
  df.label.replace(['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack
```

In [8]:
```
change_label(train_data)
```

In [9]:
```
change_label(test_data)
```

In [10]:
```
test_data['label'].value_counts()
```

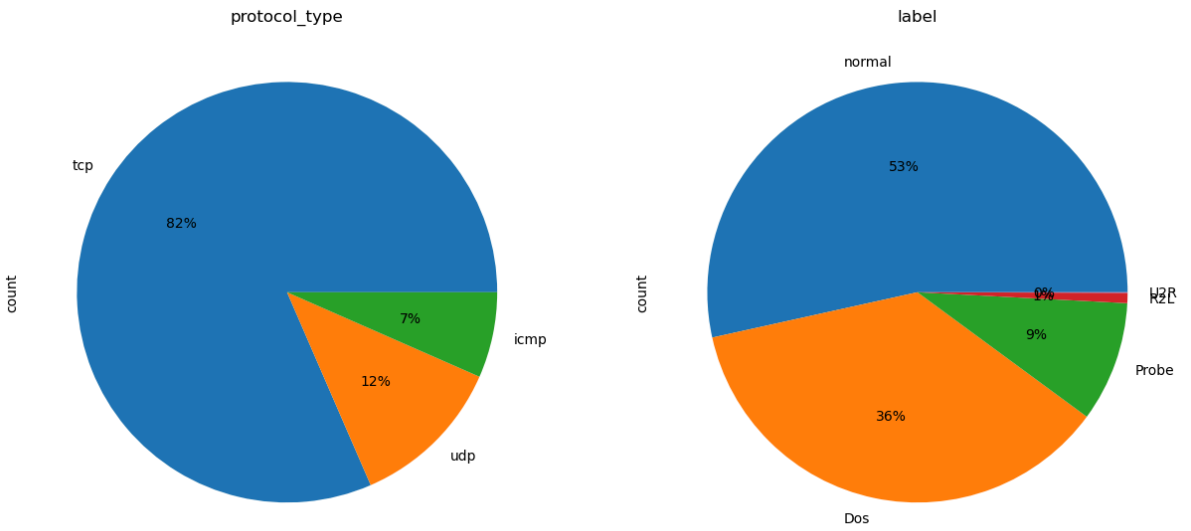
Out[10]:
```
label
normal    9711
Dos       7460
R2L       2885
Probe     2421
U2R         67
Name: count, dtype: int64
```

In [11]:
```
train_data['label'].value_counts()
```

Out[11]:
```
label
normal    67343
Dos       45927
Probe     11656
R2L         995
U2R          52
Name: count, dtype: int64
```

In [12]:
```
def pie_plot(df, cols_list, rows, cols):
    fig, axes = plt.subplots(rows, cols)
    for ax, col in zip(axes.ravel(), cols_list):
        df[col].value_counts().plot(ax=ax, kind='pie', figsize=(15, 15), fontsize=1
        ax.set_title(str(col), fontsize = 12)
    plt.show()
```

In [13]:
```
pie_plot(train_data, ['protocol_type', 'label'], 1, 2)
```



In [14]:
```
multi_data = train_data.copy()
multi_label = pd.DataFrame(multi_data.label)
```

In [15]:
```
multi_data_test = test_data.copy()
multi_label_test = pd.DataFrame(multi_data_test.label)
```

In [16]:
```
multi_label
```

Out[16]:

	label
0	normal
1	normal
2	Dos
3	normal
4	normal
...	...
125968	Dos
125969	normal
125970	normal
125971	Dos
125972	normal

125973 rows × 1 columns

In [17]:
```
std_scaler = StandardScaler()
def standardization(df,col):
  for i in col:
    arr = df[i]
    arr = np.array(arr)
    df[i] = std_scaler.fit_transform(arr.reshape(len(arr),1))
  return df

numeric_col = multi_data.select_dtypes(include='number').columns
data = standardization(multi_data,numeric_col)
```

In [18]:
```
numeric_col_test = multi_data_test.select_dtypes(include='number').columns
data_test = standardization(multi_data_test,numeric_col_test)
```

In [19]:
```
data_test
```

Out[19]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	-0.155534	tcp	private	REJ	-0.021988	-0.096896	-0.017624	-0.059104
1	-0.155534	tcp	private	REJ	-0.021988	-0.096896	-0.017624	-0.059104
2	-0.154113	tcp	ftp_data	SF	0.005473	-0.096896	-0.017624	-0.059104
3	-0.155534	icmp	eco_i	SF	-0.021946	-0.096896	-0.017624	-0.059104
4	-0.154823	tcp	telnet	RSTO	-0.021988	-0.096189	-0.017624	-0.059104
...	...	...	...	...	...	...	...	..
22539	-0.155534	tcp	smtp	SF	-0.020309	-0.081202	-0.017624	-0.059104
22540	-0.155534	tcp	http	SF	-0.021318	-0.052690	-0.017624	-0.059104
22541	-0.155534	tcp	http	SF	0.093373	0.294926	-0.017624	-0.059104
22542	-0.155534	udp	domain_u	SF	-0.021899	-0.094917	-0.017624	-0.059104
22543	-0.155534	tcp	sunrpc	REJ	-0.021988	-0.096896	-0.017624	-0.059104

22544 rows × 42 columns

In [20]:
```
# label encoding (0,1,2,3,4) multi-class labels (Dos,normal,Probe,R2L,U2R)
le2 = preprocessing.LabelEncoder()
enc_label = multi_label.apply(le2.fit_transform)
multi_data['intrusion'] = enc_label
#y_mul = multi_data['intrusion']
multi_data
```

Out[20]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	-0.110249	tcp	ftp_data	SF	-0.007679	-0.004919	-0.014089	-0.089486
1	-0.110249	udp	other	SF	-0.007737	-0.004919	-0.014089	-0.089486
2	-0.110249	tcp	private	S0	-0.007762	-0.004919	-0.014089	-0.089486
3	-0.110249	tcp	http	SF	-0.007723	-0.002891	-0.014089	-0.089486
4	-0.110249	tcp	http	SF	-0.007728	-0.004814	-0.014089	-0.089486
...	...	...	...	...	...	...	...	...
125968	-0.110249	tcp	private	S0	-0.007762	-0.004919	-0.014089	-0.089486
125969	-0.107178	udp	private	SF	-0.007744	-0.004883	-0.014089	-0.089486
125970	-0.110249	tcp	smtp	SF	-0.007382	-0.004823	-0.014089	-0.089486
125971	-0.110249	tcp	klogin	S0	-0.007762	-0.004919	-0.014089	-0.089486
125972	-0.110249	tcp	ftp_data	SF	-0.007737	-0.004919	-0.014089	-0.089486

125973 rows × 43 columns

In [21]:
```
# label encoding (0,1,2,3,4) multi-class labels (Dos,normal,Probe,R2L,U2R)
le2 = preprocessing.LabelEncoder()
enc_label = multi_label_test.apply(le2.fit_transform)
multi_data_test['intrusion'] = enc_label
#y_mul = multi_data['intrusion']
multi_data_test
```

Out[21]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	-0.155534	tcp	private	REJ	-0.021988	-0.096896	-0.017624	-0.059104
1	-0.155534	tcp	private	REJ	-0.021988	-0.096896	-0.017624	-0.059104
2	-0.154113	tcp	ftp_data	SF	0.005473	-0.096896	-0.017624	-0.059104
3	-0.155534	icmp	eco_i	SF	-0.021946	-0.096896	-0.017624	-0.059104
4	-0.154823	tcp	telnet	RSTO	-0.021988	-0.096189	-0.017624	-0.059104
...	...	...	...	...	...	...	...	..
22539	-0.155534	tcp	smtp	SF	-0.020309	-0.081202	-0.017624	-0.059104
22540	-0.155534	tcp	http	SF	-0.021318	-0.052690	-0.017624	-0.059104
22541	-0.155534	tcp	http	SF	0.093373	0.294926	-0.017624	-0.059104
22542	-0.155534	udp	domain_u	SF	-0.021899	-0.094917	-0.017624	-0.059104
22543	-0.155534	tcp	sunrpc	REJ	-0.021988	-0.096896	-0.017624	-0.059104

22544 rows × 43 columns

In [22]:
```
multi_data.drop(labels= [ 'label'], axis=1, inplace=True)
multi_data
```

Out[22]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	-0.110249	tcp	ftp_data	SF	-0.007679	-0.004919	-0.014089	-0.089486
1	-0.110249	udp	other	SF	-0.007737	-0.004919	-0.014089	-0.089486
2	-0.110249	tcp	private	S0	-0.007762	-0.004919	-0.014089	-0.089486
3	-0.110249	tcp	http	SF	-0.007723	-0.002891	-0.014089	-0.089486
4	-0.110249	tcp	http	SF	-0.007728	-0.004814	-0.014089	-0.089486
...	...	...	...	...	...	...	...	...
125968	-0.110249	tcp	private	S0	-0.007762	-0.004919	-0.014089	-0.089486
125969	-0.107178	udp	private	SF	-0.007744	-0.004883	-0.014089	-0.089486
125970	-0.110249	tcp	smtp	SF	-0.007382	-0.004823	-0.014089	-0.089486
125971	-0.110249	tcp	klogin	S0	-0.007762	-0.004919	-0.014089	-0.089486
125972	-0.110249	tcp	ftp_data	SF	-0.007737	-0.004919	-0.014089	-0.089486

125973 rows × 42 columns

In [23]:
```
multi_data_test.drop(labels= [ 'label'], axis=1, inplace=True)
```

In [24]:
```
np.array(test_data["label"])
```

Out[24]:
```
array(['Dos', 'Dos', 'normal', ..., 'Dos', 'normal', 'Probe'],
      dtype=object)
```

In [1]:
```
test_data["label"].unique
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
d:\NIDS\AI-Based-Network-IDS_ML-DL\ClassicalMulticlass.ipynb Cell 25 line 1
----> <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#Y120sZmlsZQ%3D%3D?line=0'>1</a> test_data["label"].unique

NameError: name 'test_data' is not defined
```

In [25]:
```
# one-hot-encoding attack label
multi_data = pd.get_dummies(multi_data,columns=['protocol_type','service','flag'],p
multi_data
```

Out[25]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_fai
0	-0.110249	-0.007679	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
1	-0.110249	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
2	-0.110249	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
3	-0.110249	-0.007723	-0.002891	-0.014089	-0.089486	-0.007736	-0.095076	
4	-0.110249	-0.007728	-0.004814	-0.014089	-0.089486	-0.007736	-0.095076	
...	...	...	...	...	...	...	...	
125968	-0.110249	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
125969	-0.107178	-0.007744	-0.004883	-0.014089	-0.089486	-0.007736	-0.095076	
125970	-0.110249	-0.007382	-0.004823	-0.014089	-0.089486	-0.007736	-0.095076	
125971	-0.110249	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
125972	-0.110249	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	

125973 rows × 123 columns

In [26]:
```
# one-hot-encoding attack label
multi_data_test = pd.get_dummies(multi_data_test,columns=['protocol_type','service'
multi_data_test
```

Out[26]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_fail
0	-0.155534	-0.021988	-0.096896	-0.017624	-0.059104	-0.019459	-0.113521	
1	-0.155534	-0.021988	-0.096896	-0.017624	-0.059104	-0.019459	-0.113521	
2	-0.154113	0.005473	-0.096896	-0.017624	-0.059104	-0.019459	-0.113521	
3	-0.155534	-0.021946	-0.096896	-0.017624	-0.059104	-0.019459	-0.113521	
4	-0.154823	-0.021988	-0.096189	-0.017624	-0.059104	-0.019459	-0.113521	
...	...	...	...	...	...	...	...	
22539	-0.155534	-0.020309	-0.081202	-0.017624	-0.059104	-0.019459	-0.113521	
22540	-0.155534	-0.021318	-0.052690	-0.017624	-0.059104	-0.019459	-0.113521	
22541	-0.155534	0.093373	0.294926	-0.017624	-0.059104	-0.019459	2.040705	
22542	-0.155534	-0.021899	-0.094917	-0.017624	-0.059104	-0.019459	-0.113521	
22543	-0.155534	-0.021988	-0.096896	-0.017624	-0.059104	-0.019459	-0.113521	

22544 rows × 117 columns

In [27]:

```
unique_columns_multi_data = set(multi_data.columns) - set(multi_data_test.columns)
unique_columns_multi_data_test = set(multi_data_test.columns) - set(multi_data.colu


print("Columns in multi_data but not in multi_data_test:", unique_columns_multi_dat
print("Columns in multi_data_test but not in multi_data:", unique_columns_multi_dat


multi_data = multi_data.drop(columns=unique_columns_multi_data)
multi_data
```

```
Columns in multi_data but not in multi_data_test: {'http_8001', 'harvest', 'http_2
784', 'aol', 'red_i', 'urh_i'}
Columns in multi_data_test but not in multi_data: set()
```

Out[27]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_fai
0	-0.110249	-0.007679	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
1	-0.110249	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
2	-0.110249	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
3	-0.110249	-0.007723	-0.002891	-0.014089	-0.089486	-0.007736	-0.095076	
4	-0.110249	-0.007728	-0.004814	-0.014089	-0.089486	-0.007736	-0.095076	
...	...	...	...	...	...	...	...	
125968	-0.110249	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
125969	-0.107178	-0.007744	-0.004883	-0.014089	-0.089486	-0.007736	-0.095076	
125970	-0.110249	-0.007382	-0.004823	-0.014089	-0.089486	-0.007736	-0.095076	
125971	-0.110249	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	
125972	-0.110249	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	

125973 rows × 117 columns

In [28]:
```
y_train_multi= multi_data[['intrusion']]
X_train_multi= multi_data.drop(labels=['intrusion'], axis=1)

print('X_train has shape:',X_train_multi.shape,'\ny_train has shape:',y_train_multi
```

```
X_train has shape: (125973, 116)
y_train has shape: (125973, 1)
```

In [29]:
```
y_test_multi= multi_data_test[['intrusion']]
X_test_multi= multi_data_test.drop(labels=['intrusion'], axis=1)

print('X_train has shape:',X_test_multi.shape,'\ny_train has shape:',y_test_multi.s
```

```
X_train has shape: (22544, 116)
y_train has shape: (22544, 1)
```

In [30]:
```
from sklearn.preprocessing import LabelBinarizer

y_train_multi = LabelBinarizer().fit_transform(y_train_multi)
y_train_multi
```

Out[30]:
```
array([[0, 0, 0, 0, 1],
       [0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0],
       ...,
       [0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1]])
```

In [31]:
```
from sklearn.preprocessing import LabelBinarizer

y_test_multi = LabelBinarizer().fit_transform(y_test_multi)
y_test_multi
```

Out[31]:
```
array([[1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1],
       ...,
       [1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1],
       [0, 1, 0, 0, 0]])
```

In [ ]:
```

```

In [32]:
```
X_train = np.array(X_train_multi)
y_train = np.array(y_train_multi)
```

In [33]:
```
X_test=np.array(X_test_multi)
y_test=np.array(y_test_multi)
```

In [34]:
```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape
```

Out[34]:
```
(125973, 116, 1)
```

In [35]:
```
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
X_test.shape
```

Out[35]:
```
(22544, 116, 1)
```

In [36]:
```
model = Sequential() # initializing model
# input layer and first layer with 50 neurons
```

```
model.add(Conv1D(32, 3, padding="same",input_shape = (X_train.shape[1], 1), activat
model.add(MaxPool1D(pool_size=(4)))
model.add(Dropout(0.2))
model.add(Conv1D(32, 3, padding="same", activation='relu'))
model.add(MaxPool1D(pool_size=(4)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units=50))
# output layer with softmax activation
model.add(Dense(units=5,activation='softmax'))
```

In [37]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
```

In [38]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 116, 32)           128

 max_pooling1d (MaxPooling1D  (None, 29, 32)           0
 )

 dropout (Dropout)           (None, 29, 32)            0

 conv1d_1 (Conv1D)           (None, 29, 32)            3104

 max_pooling1d_1 (MaxPooling  (None, 7, 32)            0
 1D)

 dropout_1 (Dropout)         (None, 7, 32)             0

 flatten (Flatten)           (None, 224)               0

 dense (Dense)               (None, 50)                11250

 dense_1 (Dense)             (None, 5)                 255

=================================================================
Total params: 14,737
Trainable params: 14,737
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
with tf.device('/GPU:0'):
    # Your Keras code here

    # test with GPU
    history = model.fit(X_train, y_train, epochs=100, batch_size=5000,validation_sp
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
d:\NIDS\AI-Based-Network-IDS_ML-DL\ClassicalMulticlass.ipynb Cell 40 line 5
      <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#X54sZmlsZQ%3D%3D?line=0'>1</a> with tf.device('/GPU:0'):
      <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#X54sZmlsZQ%3D%3D?line=1'>2</a>     # Your Keras code here
      <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#X54sZmlsZQ%3D%3D?line=2'>3</a>
      <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#X54sZmlsZQ%3D%3D?line=3'>4</a>     # test with GPU
----> <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#X54sZmlsZQ%3D%3D?line=4'>5</a>     history = model.fit(X_train, y_train, epochs=100, batch_size=5000,validation_split=0.2)

File c:\Users\Cheddad\anaconda3\envs\tf-gpu\lib\site-packages\keras\utils\traceback_utils.py:70, in filter_traceback.<locals>.error_handler(*args, **kwargs)
     67     filtered_tb = _process_traceback_frames(e.__traceback__)
     68     # To get the full stack trace, call:
     69     # `tf.debugging.disable_traceback_filtering()`
---> 70     raise e.with_traceback(filtered_tb) from None
     71 finally:
     72     del filtered_tb

File c:\Users\Cheddad\anaconda3\envs\tf-gpu\lib\site-packages\tensorflow\python\framework\constant_op.py:102, in convert_to_eager_tensor(value, ctx, dtype)
    100     dtype = dtypes.as_dtype(dtype).as_datatype_enum
    101 ctx.ensure_initialized()
--> 102 return ops.EagerTensor(value, ctx.device_name, dtype)

ValueError: Failed to convert a NumPy array to a Tensor (Unsupported object type float).
```

In [69]:
```
print(X_train.dtype)
print(X_test.dtype)
```

```
object
object
```

In [42]:
```
X_train = X_train.astype('float64')
X_test = X_test.astype('float64')
```

In [43]:
```
X_train.shape
```

Out[43]:
```
(125973, 116, 1)
```

In [1]:
```
import tensorflow as tf

print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU
if tf.test.is_gpu_available():
    print("TensorFlow is using the GPU")
else:
    print("TensorFlow is not using the GPU")
```

```
Num GPUs Available:  1
WARNING:tensorflow:From C:\Users\Cheddad\AppData\Local\Temp\ipykernel_23848\2513694533.py:4: is_gpu_available (from tensorflow.python.framework.test_util) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
TensorFlow is using the GPU
```

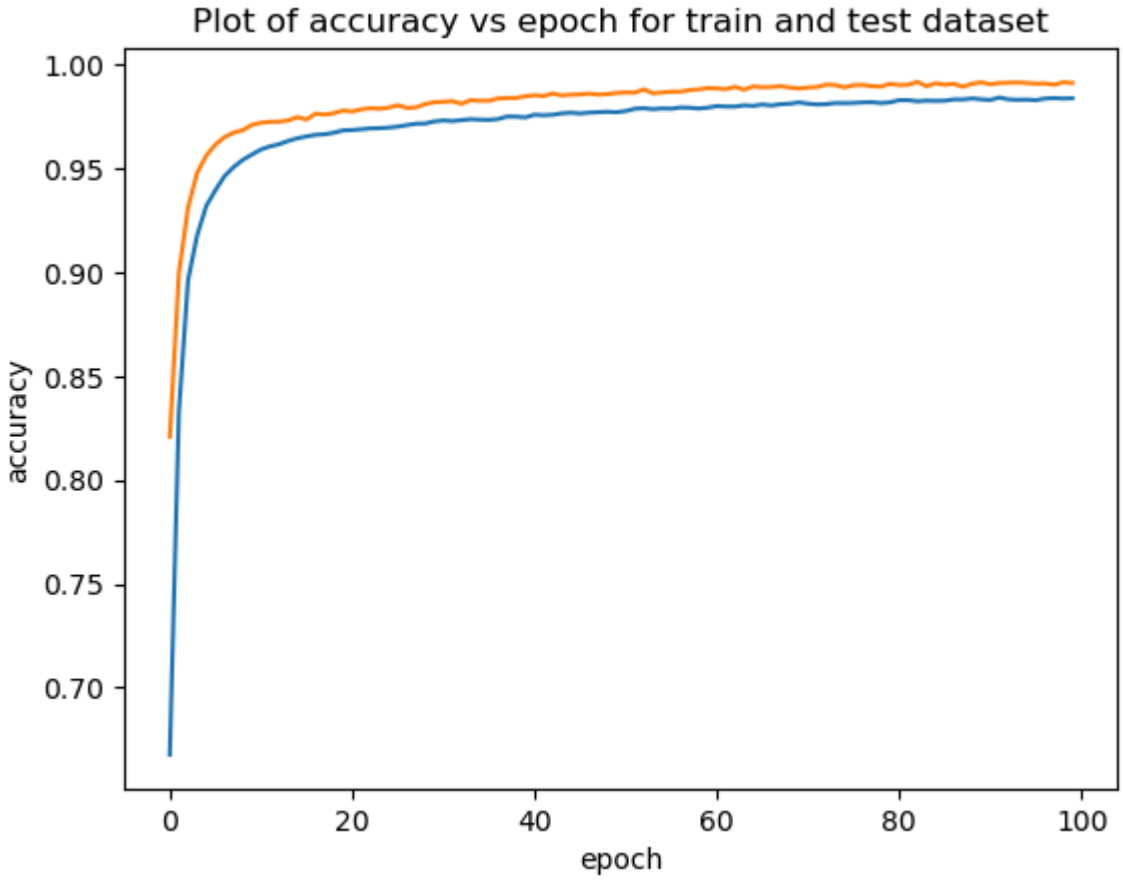
In [2]:
```
tf.config.list_physical_devices('GPU')
```

Out[2]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

In [45]:
```
# predicting target attribute on testing dataset
test_results = model.evaluate(X_test, y_test, verbose=1)
print(f'Test results - Loss: {test_results[0]} - Accuracy: {test_results[1]*100}%')
```

```
705/705 [==============================] - 2s 3ms/step - loss: 1.6345 - accuracy:
0.7441
Test results - Loss: 1.6345133781433105 - Accuracy: 74.41004514694214%
```

In [46]:
```
# Plot of accuracy vs epoch for train and test dataset
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("Plot of accuracy vs epoch for train and test dataset")
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [47]:
```
y_pred_test = model.predict(X_test, batch_size=500)

# For Plot curve
y_pred_evaluation_test = to_categorical(np.argmax(y_pred_test, axis=1), 5)

# For evaluation
y_pred_argmax_test=(np.argmax(y_pred_test, axis=1))

y_test_argmax=(np.argmax(y_test, axis=1))
```

```
46/46 [==============================] - 0s 4ms/step
```

In [48]:
```
# Calculating Area under the curve
def AUC(actual_class, pred_class, average = "micro"):

    #Making a set of all the unique classes
    unique_class = set(actual_class)
```

```
    roc_auc_dict = {}
    for per_class in unique_class:

        #Making a list of all the classes except the current class
        other_class = [x for x in unique_class if x != per_class]

        #Making the current class with label 1 and all other classes as a label 0
        new_actual_class = [0 if x in other_class else 1 for x in actual_class]
        new_pred_class = [0 if x in other_class else 1 for x in pred_class]

        # Calculating the roc_auc_score
        roc_auc = roc_auc_score(new_actual_class, new_pred_class, average = average
        roc_auc_dict[per_class] = roc_auc

    return roc_auc_dict
```

In [49]:

```
def plot_roc_curve(y_test,y_pred, classes):
    n_classes=len(classes)
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])
    fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_pred.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
    lw=2
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
    plt.figure(figsize=(10, 10))
    plt.plot(fpr["micro"],
    tpr["micro"],
    label="micro-average ROC curve (area = {0:0.2f})".format(roc_auc["micro"]),
    color="deeppink",
    linestyle=":",
    linewidth=4,)

    plt.plot(fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (area = {0:0.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)

    colors = cycle(["aqua", "darkorange", "cornflowerblue"])
    for i, color in zip(range(n_classes), colors):
        plt.plot(
           fpr[i],
           tpr[i],
           color=color,
           lw=lw,
        label="ROC curve of {0} (area = {1:0.2f})".format(classes[i], roc_auc[i]),
```

```
    )

    plt.plot([0, 1], [0, 1], "k--", lw=lw)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("AUC")
    plt.legend(loc="lower right")
    plt.show()
```

In [50]:
```
import itertools
from itertools import cycle
```

In [51]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    Prints and plots the confusion matrix.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [52]:
```
# predicting target attribute on testing dataset
test_results = model.evaluate(X_test, y_test, verbose=1)
print(f'Test results - Loss: {test_results[0]} - Accuracy: {test_results[1]*100}%')
```

```
705/705 [==============================] - 2s 3ms/step - loss: 1.6345 - accuracy:
0.7441
Test results - Loss: 1.6345133781433105 - Accuracy: 74.41004514694214%
```

In [53]:
```
print('AUC Score is on Test : ' , AUC(y_test_argmax, y_pred_argmax_test))
```

```
AUC Score is on Test :  {0: 0.8778409450419917, 1: 0.7669850042232011, 2: 0.546488
5571084037, 3: 0.6044331220172661, 4: 0.7881146975697078}
```

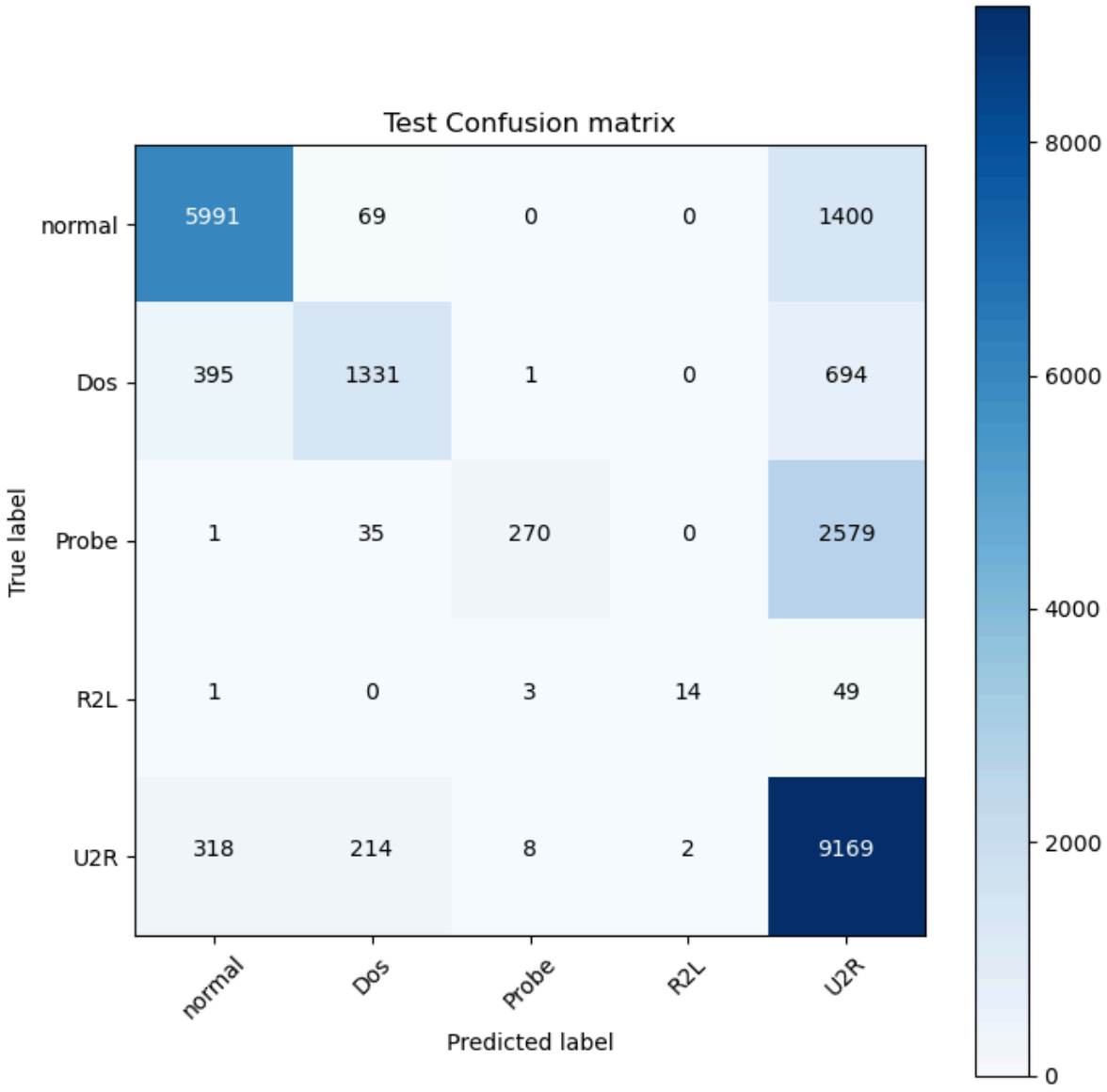
In [1]:
```
from sklearn.metrics import classification_report
classes=['normal', 'Dos','Probe', 'R2L','U2R']

print("Classification Report on Data Test \n" , classification_report(y_test_argmax
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
d:\NIDS\AI-Based-Network-IDS_ML-DL\ClassicalMulticlass.ipynb Cell 55 line 4
      <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#Y105sZmlsZQ%3D%3D?line=0'>1</a> from sklearn.metrics import classification_report
      <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#Y105sZmlsZQ%3D%3D?line=1'>2</a> classes=['normal', 'Dos','Probe', 'R2L','U2R']
----> <a href='vscode-notebook-cell:/d%3A/NIDS/AI-Based-Network-IDS_ML-DL/ClassicalMulticlass.ipynb#Y105sZmlsZQ%3D%3D?line=3'>4</a> print("Classification Report on Data Test \n" , classification_report(y_test_argmax, y_pred_argmax_test, target_names=classes))

NameError: name 'y_test_argmax' is not defined
```

In [ ]:

```python
# Compute confusion matrix
from sklearn.metrics import confusion_matrix

cnf_matrix = confusion_matrix(y_test_argmax, y_pred_argmax_test)

#DL confusion matrix

# Plot non-normalized confusion matri
#
# x
plt.figure(figsize=(7, 7))
plot_confusion_matrix(cnf_matrix, classes=classes,
                      title='Test Confusion matrix')
plt.show()
```

```
Confusion matrix, without normalization
```



In [62]:
```
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

rf = RandomForestClassifier().fit(X_train_multi, y_train_multi)
```

```
C:\Users\Cheddad\AppData\Local\Temp\ipykernel_23848\3689068072.py:3: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
  rf = RandomForestClassifier().fit(X_train_multi, y_train_multi)
```

In [65]:
```
y_pred_test = rf.predict(X_test_multi)

y_pred_test
```

Out[65]:
```
array([1, 1, 4, ..., 4, 4, 1])
```

In [66]:
```
y_pred_labels = le2.inverse_transform(y_pred_test)
y_pred_labels
```

Out[66]:
```
array(['Probe', 'Probe', 'normal', ..., 'normal', 'normal', 'Probe'],
      dtype=object)
```

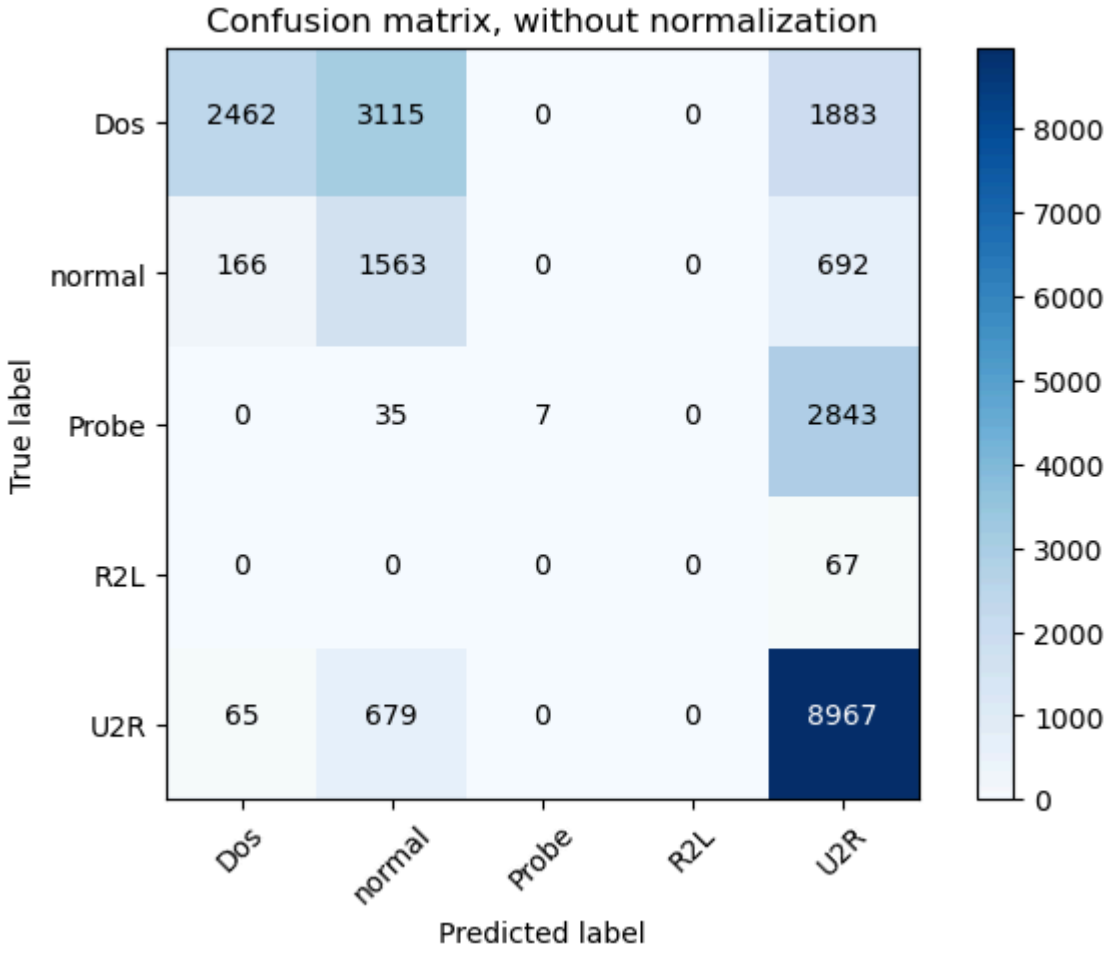
In [71]:
```
np.array(test_data["label"])
```

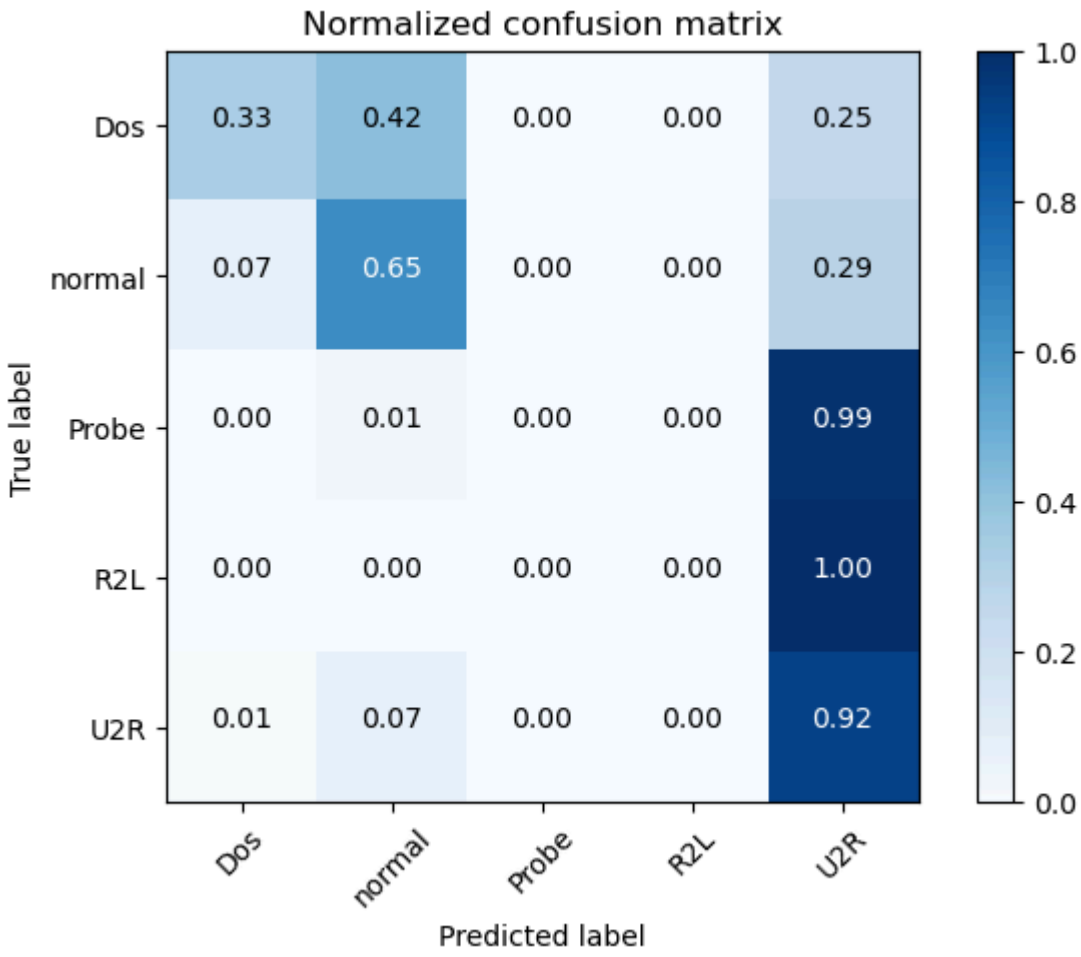
Out[71]:
```
array(['Dos', 'Dos', 'normal', ..., 'Dos', 'normal', 'Probe'],
      dtype=object)
```

In [72]:
```
# Compute confusion matrix
cm = confusion_matrix(np.array(test_data["label"]), y_pred_labels)
```

In [73]:
```
# List of classes
classes = ["Dos", "normal", "Probe", "R2L", "U2R"]

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cm, classes=classes, title='Confusion matrix, without normali

# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cm, classes=classes, normalize=True, title='Normalized confus

plt.show()
```

```
Confusion matrix, without normalization
Normalized confusion matrix
```




Normalized confusion matrix
Dos
normal
Probe
R2L
U2R
True label
0.33 0.42 0.00 0.00 0.25
0.07 0.65 0.00 0.00 0.29
0.00 0.01 0.00 0.00 0.99
0.00 0.00 0.00 0.00 1.00
0.01 0.07 0.00 0.00 0.92
Dos
normal
Probe
R2L
U2R
Predicted label
1.0
0.8
0.6
0.4
0.2
0.0

In [ ]: