In [18]:

```
from tensorflow.keras import Sequential #this tensorflow GPU based
from tensorflow.keras.layers import Dense
import pandas as pd
import numpy as np
from sklearnex import patch_sklearn #for speed up sklearn
from daal4py.oneapi import sycl_context #for speed up GPU
patch_sklearn()
sycl_context("gpu")
from sklearn.preprocessing import StandardScaler,OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/sciki
t-learn-intelex)
```

In [19]:

```
data=pd.read_csv("train.csv")
data.head()
```

Out[19]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

In [20]:

```
x=data.iloc[:,:20].values
y=data.iloc[:,20:21].values
#print(y)
```

In [21]:

```
sc= StandardScaler()
x=sc.fit_transform(x)
```

In [22]:

```
print(x)
```

```
[[-0.90259726 -0.9900495   0.83077942 ... -1.78686097 -1.00601811
   0.98609664]
 [-0.49513857  1.0100505  -1.2530642  ...  0.55964063  0.99401789
  -1.01409939]
 [-1.5376865   1.0100505  -1.2530642  ...  0.55964063  0.99401789
  -1.01409939]
 ...
 [ 1.53077336 -0.9900495  -0.76274805 ...  0.55964063  0.99401789
  -1.01409939]
 [ 0.62252745 -0.9900495  -0.76274805 ...  0.55964063  0.99401789
   0.98609664]
 [-1.65833069  1.0100505   0.58562134 ...  0.55964063  0.99401789
   0.98609664]]
```

In [23]:

```
ohe=OneHotEncoder()
y=ohe.fit_transform(y).toarray()
```

In [24]:

```
y
```

Out[24]:

```
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       ...,
       [0., 0., 0., 1.],
       [1., 0., 0., 0.],
       [0., 0., 0., 1.]])
```

In [25]:

```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.1)
```

In [26]:

```
#nn
nn=Sequential()
nn.add(Dense(16, input_dim=20, activation="relu"))
nn.add(Dense(12, activation="relu"))
nn.add(Dense(4,activation="softmax"))
```

In [27]:

```
nn.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [28]:

```
tnn=nn.fit(x_train,y_train,epochs=100,batch_size=64)
```

```
racy: 0.9772
Epoch 60/100
29/29 [==============================] - 0s 2ms/step - loss: 0.0888 - accu
racy: 0.9761
Epoch 61/100
29/29 [==============================] - 0s 2ms/step - loss: 0.0849 - accu
racy: 0.9772
Epoch 62/100
29/29 [==============================] - 0s 2ms/step - loss: 0.0839 - accu
racy: 0.9767
Epoch 63/100
29/29 [==============================] - 0s 2ms/step - loss: 0.0813 - accu
racy: 0.9783
Epoch 64/100
29/29 [==============================] - 0s 2ms/step - loss: 0.0812 - accu
racy: 0.9800
Epoch 65/100
29/29 [==============================] - 0s 2ms/step - loss: 0.0780 - accu
racy: 0.9817
```

In [29]:

```
y_pred=nn.predict(x_test)
```

In [30]:

```
pred=list()
for i in range(len(y_pred)):
    pred.append(np.argmax(y_pred[i]))
test=list()
for i in range(len(y_test)):
    test.append(np.argmax(y_test[i]))
```

In [31]:

```
accuracy_score(pred,test)
```

Out[31]:

```
0.935
```

In [ ]: