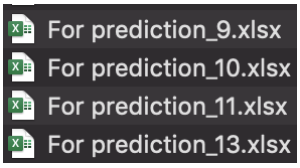
A Machine Learning Guided Appraisal of Phase Design for High Entropy Alloys

1. **Datasets:**
   We have four datasets, which contain the same alloys with different number of features.

   These are datasets with 13, 9, 11, 10 features respectively.



   These are datasets of unknown test alloys the phases of which are to be predicted.

2. **Data Preprocessing:**
   Each dataset containing 601 entries is converted to .csv format (comma-separated values) and then split into three parts, 70% for training, 15% for validating and 15% for testing. The distribution of binary, ternary and quaternary alloys remain constant along the split sets.

   Run the Jupyter Notebook *data_preprocessing.ipynb* to process the data before feeding to networks:

   Parameters:

```
In [1]:
1  import numpy as np
2  data_file = 'data.csv'
3  data_pred = 'for_pred.csv'
4  num_of_feature = 13
```

   - *data_file*: the .csv file to be processed;
   - *data_pred*: the dataset with unknown test alloys the phases of which are to be predicted from the trained model;
   - *num_of_feature*: the number of features recorded in the dataset to be trained.

   After running the notebook, 14 .npy files to be fed to networks are generated under the *data* directory. *x_train, x_val, x_test* contain the features of the alloys, which are used as training, validating and testing sets respectively. *y_train_AM* contains the label of all the alloys used to train the AM phase class; if the label is 1, then AM is the phase of the corresponding alloy. Remaining files are similar.

3. **Train CNN model:**
   Run the Jupyter Notebook *cnn.ipynb* to train the convolutional neural network model.

   Parameters:

In [1]:
```
import torch
import numpy as np
import matplotlib.pyplot as plt

class Flatten(torch.nn.Module):
    def forward(self, input):
        return input.view(input.size(0), -1)

num_of_feature = 11
```

- *num_of_feature*: the number of features of the dataset to be trained.

In [3]:
```
def train(batch, epoch, dataset):
    x_train = np.load('data/x_train_11.npy') #420
    x_val = np.load('data/x_val_11.npy')     #91
    x_test = np.load('data/x_test_11.npy')   #90
    y_train = np.load('data/y_train_' + dataset + '_11.npy')
    y_val = np.load('data/y_val_' + dataset + '_11.npy')
    y_test = np.load('data/y_test_' + dataset + '_11.npy')
    x_train = np.expand_dims(x_train, axis=1)
    x_val = np.expand_dims(x_val, axis=1)
    x_test = np.expand_dims(x_test, axis=1)
    data_pred = np.load('data/for_pred_11.npy')
```

- *x_train_**.npy*, *y_train_**_**.npy*: the data files to be loaded, the digits denote the number of features of the datasets to be trained.

In [4]:
```
data_pred_labels = train(20, 200, 'SS')
```

- *train(20, 200, 'SS')*: 20 denotes the batch size, 200 denotes the number of epochs of training, 'SS' denotes the phase class on which the model is trained.

In [5]:
```
print(data_pred_labels[70])
```

- *data_pred_labels[70]*: return the predicted labels of the unknown test alloys at a given epoch.

4. **SVM:**

Run the Jupyter Notebook *svm.ipynb* to train the SVM model:

Parameters:

In [9]:
```
data_pred_label = train('SS', 10)
```

- *train('SS', 10)*: 'SS' denotes the phase class on which the model is trained, 10 denotes the value of C, the regularization parameter.

```
print(data_pred_label)
```

- *data_pred_label*: return the predicted labels of the unknown test alloys.