Computational Materials Science 150 (2018) 230–235

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Machine learning for phase selection in multi-principal element alloys

Nusrat Islam[a], Wenjiang Huang[b], Houlong L. Zhuang[a,*]

[a] School for Engineering of Matter Transport and Energy, Arizona State University, Tempe, AZ 85287, USA
[b] School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, AZ 85287, USA

ARTICLE INFO

Keywords:
Multiprincipal element alloys
High entropy alloys
Machine learning
Phase selection

ABSTRACT

Multi-principal element alloys (MPEAs) especially high entropy alloys have attracted significant attention and resulted in a novel concept of designing metal alloys via exploring the wide composition space. Abundant experimental data of MPEAs are available to show connections between elemental properties and the resulting phases such as single-phase solid solution, amorphous, intermetallic compounds. To gain insights of designing MPEAs, here we employ neural network (NN) in the machine learning framework to recognize the underlying data pattern using an experimental dataset to classify the corresponding phase selection in MPEAs. For the full dataset, our trained NN model reaches an accuracy of over 99%, meaning that more than 99% of the phases in the MPEAs are correctly labeled. Furthermore, the trained NN parameters suggest that the valence electron concentration plays the most dominant role in determining the ensuing phases. For the cross-validation training and testing datasets, we obtain an average generalization accuracy of higher than 80%. Our trained NN model can be extended to classify different phases in numerous other MPEAs.

1. Introduction

Metal alloys are critical for structural applications in many industries such as aircraft engines [1]. Conventional metal alloys like various series of aluminum alloys consist of one principal element (*i. e*., aluminum) and other elements with much lower atomic concentrations controlled to obtain desirable properties [2]. Unlike conventional metal alloys, multi-principal element alloys (MPEAs), as its name implies, have (nearly) equal amount of individual principal elements.

MPEA is often interchangeably used with high entropy alloy (HEA) in the literature [3], which is a term coined by Yeh et al. [4], whose experimental work was performed indepently from the earlier pioneering experimental work by Cantor et al. [5] However, the definition of HEA limits the number of species in an HEA to more than four [4]. On the other hand, an MPEA can contain only two species with the same atomic concentration (*e. g*., amorphous $Ni^{50}Nb^{50}$ [6]). Due to the broader definition, we therefore use MPEA throughout this paper.

MPEAs especially HEAs can exhibit superior mechanical properties compared to those of conventional metal alloys [7], which suffer from a tradeoff between strength and toughness. The existence of different phases makes it possible to target one of them in order to obtain the excellent mechanical properties of an MPEA and the common phases in MPEAs include single-phase solid solution (SS), amorphous (AM), intermetallic compounds (IM), and combined SS and IM phases [8]. Predicting the phase selection is therefore important to guide the design of a new MPEA for different purposes. However, the mechanisms of forming different phases in MPEAs remain unclear, making it challenging to predict the phase selection of an MPEA.

It has been generally accepted that phase selection is correlated with basic atomic properties of individual species along with thermodynamic properties (*e. g*., enthalpy) of pair-wise binary alloys [9]. Along this line, a number of previous studies adopt parametric approaches [10–13], where different phases are labeled in a two-dimensional plot with two parameters shown in the abscissa and the ordinate, respectively, and different phases are color-coded in the plot. Empirical rules are then summarized by observation. For instance, mixing enthalpy (ΔH_{mix}) and atomic radius difference (δ) need to lie in certain ranges ($-15 < \Delta H_{mix} < 5\,\text{kJ mol}^{-1}$; $1\% < \delta < 5\%$) to form a SS phase [8]. These parametric approaches can actually trace back to the Hume-Rothery rules discovered more than eight decades ago showing that forming a binary alloy with SS is affected by electrochemical and size factors [14]. Similarly, Inoue et al. proposed three empirical rules for the formation of bulk metallic glasses [15]. The existing parametric approaches provide valuable quantitative criteria, but also face three problems that limit their predictive ability. First, since the phase selection depends on more than three parameters. Such a dependence in a high-dimensional space makes the visualization not possible. Second, the applicability of the quantitative criteria to other new MPEAs that are outside the dataset used for summarizing the criteria is unknown. Third, there is no information about the relative importance of the

* Corresponding author.
E-mail address: zhuanghl@asu.edu (H.L. Zhuang).

https://doi.org/10.1016/j.commatsci.2018.04.003
Received 15 March 2018; Accepted 3 April 2018
Available online 20 April 2018

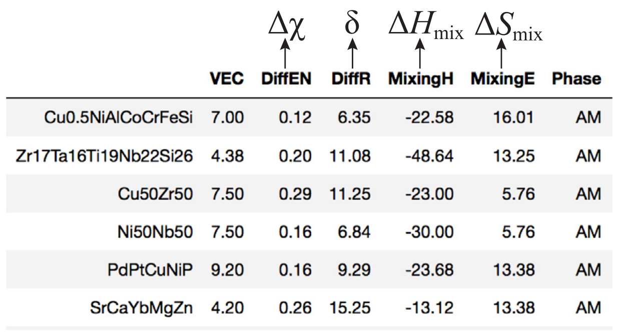
parameters in the parametric approaches.

Through the intensive studies of MPEAs over the past decades, there are a significant amount of data await exploration in order to determine the key factors that lead to a specific phase selection in an MPEA [16,3]. Machine learning (ML) is one state-of-the-art computational technique to recognize data patterns and obtain insights from the data without explicit programming [17]. ML is commonly used in image and speech recognitions and it has recently also gained popularity in materials science [18–23]. Although ML comprises of a variety of algorithms such as decision tree learning and Naïve Bayesian network [24], neural network (NN) inspired from the functioning of human brain is probably the most widely used algorithm. There have been many studies of applying NN to metal alloys, most of which are regression problems, *i. e.*, aiming to extract numerical values from the network. For example, Curtin *etal*. applied NN to obtain Mg-Al-Si interatomic potentials [25]. Zhang et al. used NN for revisiting Hume-Rothery and predicting the solubility of binary alloys [26]. NN was also adopted to predict the mechanical properties of several families of Cu alloys [27]. Similar to the classic ML problem of recognizing handwritten digits [28], phase selection in MPEAs belongs to the category of classification problems. Namely, although the outputs are still numeric, we encode them with different classes.

In this work, we employ an NN architecture to deal with the above-mentioned three problems of parametric approaches for the phase selection in MPEAs. We use the dataset from Ref. [29] involving three phases (AM, SS, and IM) in the MPEAs. Our goal is to obtain a set of hyperparameters that are able to predict the phase selection of a new MPEA with maximum predictive accuracy and generality. Meanwhile, we transform the hyperparameters into singular values as in the singular value decomposition to compare the relative importance of the physical parameters that lead to the phase selection.

2. Computational methods

Prior to training the network, we have adopted standard tools of data science for analyzing and pre-processing the data of MPEAs. We use the Pandas library [30] to import the dataset. Pandas imports data in the form of an *N*-dimensional array which is very useful as it allows linear algebra operations. The data used in this work consists of 118 rows and 6 columns. Fig. 1 shows a snapshot of the first 6 rows of the data in the Pandas DataFrame format. The first column identifies an MPEA and the remaining six columns describe the corresponding properties. In the literature of machine learning, each property of the data represents a feature and each row of the data is called an instance [31]. The dataset therefore has 118 instances and 6 features. Among the 6 features, 5 of them are numeric features containing real numbers corresponding to valence electron concentration (VEC), difference in the Pauling negativities $\Delta\chi$, atomic size difference δ, mixing enthalpy ΔH_{mix}, and mixing entropy ΔS_{mix}. The remaining feature is a categorical feature describing the targeting labels denoting different phases (AM, SS, or IM). Among the 118 MPEAs, 33, 64, and 21 of them are categorized into the AM, SS, and IM classes, respectively. Since the target variables are types of alloy phases, a vector is used as identifier for each of the phases. In our model, vectors (1, 0, 0), (0, 1, 0) and (0, 0, 1) correspond to the AM, SS, and IM phases, respectively.



		$\Delta\chi$	δ	ΔH_{mix}	ΔS_{mix}	
	VEC	**DiffEN**	**DiffR**	**MixingH**	**MixingE**	**Phase**
Cu0.5NiAlCoCrFeSi	7.00	0.12	6.35	-22.58	16.01	AM
Zr17Ta16Ti19Nb22Si26	4.38	0.20	11.08	-48.64	13.25	AM
Cu50Zr50	7.50	0.29	11.25	-23.00	5.76	AM
Ni50Nb50	7.50	0.16	6.84	-30.00	5.76	AM
PdPtCuNiP	9.20	0.16	9.29	-23.68	13.38	AM
SrCaYbMgZn	4.20	0.26	15.25	-13.12	13.38	AM

Fig. 1. A Pandas snapshot of the first 6 instances of the data used in this work. The arrows and notations are added to show the corresponding physical parameters of each feature. The units of ΔH_{mix} and ΔS_{mix} are $\mathrm{kJ\,mol^{-1}}$ and $\mathrm{J\,K^{-1}\,mol^{-1}}$, respectively.

Although the values of the numeric features have been directly taken from Ref. [29], the following equations show calculation methodology of the features [12]:

$$\mathrm{VEC} = \sum_{i=1}^{n} c_i \mathrm{VEC}_i, \tag{1}$$

$$\Delta\chi = \sqrt{\sum\nolimits_{i=1}^{n} c_i(\chi_i - \overline{\chi})^2}, \tag{2}$$

$$\delta = 100 \times \sqrt{\sum\nolimits_{i=1}^{n} c_i(1 - r_i/\overline{r})^2}, \tag{3}$$

$$\Delta H_{\mathrm{mix}} = \sum_{i=1, i<j}^{n} 4H_{ij} c_i c_j, \tag{4}$$

$$\Delta S_{\mathrm{mix}} = -R \sum_{i=1}^{n} c_i \ln c_i, \tag{5}$$

where c_i ($0 < c_i < 1$), VEC_i, and r_i refer to the atomic concentration, VEC, and atomic radius of each species in an MPEA, respectively. H_{ij} is the enthalpy of atomic pairs calculated with Miedema's model.[15] R denotes the gas constant. $\overline{\chi}$ and $\overline{r}$ are weighted Pauling electronegativity and atomic radius written as

$$\overline{\chi} = \sum_{i=1}^{n} c_i \chi_i \tag{6}$$

and

$$\overline{r} = \sum_{i=1}^{n} c_i r_i, \tag{7}$$

respectively.

We also use the Pandas library to normalize the feature values so that they range between 0 and 1 by applying the following relation to each feature:

$$X_{\mathrm{new}} = \frac{X_i - X_{\mathrm{min},i}}{X_{\mathrm{max},i} - X_{\mathrm{min},i}}, \tag{8}$$

where X_{new} refers to a normalized feature and X_i are the original data from one of the 5 features. $X_{\mathrm{max},i}$ and $X_{\mathrm{min},i}$ are the maximum and minimum values of the respective feature. This normalization process leads to dimensionless numeric features. More importantly, the process ensures that each individual feature has the same numeric scale and all of the features are equally treated, which is important for the gradient descent method used for minimizing the cross entropy (see below).

An NN architecture is composed of layers interconnected through neurons. A collection of neurons that performs a common function is known as a layer. These neurons are like small computers that receive information from other neurons and perform simple mathematical operations. The output of each neuron in a hidden layer a_j is given by the following linear equation:

$$a_j = x_i W_{ij} + b_j, \tag{9}$$

where W_{ij} are the weights associated with each input parameters x_i and b_j are the bias terms. These bias terms help to activate the neurons even when all the inputs to the neurons are zero. The value of a_j is then passed through an activation or transfer function which introduces nonlinearity to the network and helps the model to learn complex patterns. All neural networks have an input and output layer to connect to the external environment. There can also be neurons in between these

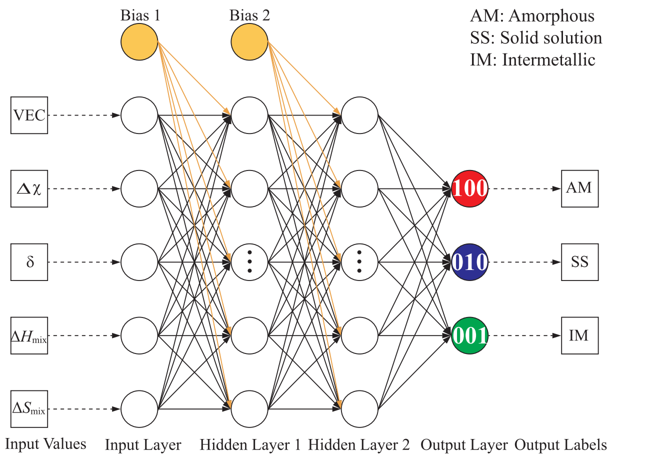


Fig. 2. An NN architecture containing two hidden layers. Only five neurons (denoted by circles) in the hidden layers are shown for clarity. The input features and output values are represented by empty squares. The AM, SS, and IM phases are encoded as vectors (1 0 0), (0 1 0), and (0 0 1), respectively.

layers and are known as hidden layers. The input layer receives information such as data and signals from the outside world. These inputs can be in the form of binary, integers, or real values. In case of strings, the data needs to be encoded in binary variables. The hidden layers help to extract patterns from the input data to analyze the system. Most of the internal processing is carried out by this layer. The output layer produces the result from the preprocessed neurons from the previous layer.

We use the machine learning NN architecture as implemented in Tensorflow developed by Google [32], which is one of the most popular ML libraries. Fig. 2 illustrates the NN architecture used in this work., which is essentially a backpropagation NN model with 2 hidden layers. Each hidden layer consists of 10 neurons associated with weights specific to the connection. Every artificial neuron computes the value of a_j by using Eq. (9). This value could be any real number passed to the activation function. Based on the output of the activation function, the network then decides whether to activate the neuron. The output layer had 3 neurons each representing one of the three phases of the MPEAs.

In the hidden layers, we have used the Rectified linear unit (ReLU) activation function. This is the preferred activation function due to its simple calculation and low computational cost [33]. Fig. 3 shows the graph of ReLU activation function. ReLU has the function form of max $\{0, a_j\}$,where a_j as defined in Eq. (9). The output of the function is the same as its input if the value is greater than zero and zero elsewhere. The sigmoid and hyperbolic tangent functions tanh are two other commonly used activation functions [33]. The output of the sigmoid activation function is between 0 and 1 while that of tanh ranges between -1 and 1. Unlike these two other functions, ReLU does not saturate to -1, 0 or 1. ReLU also has a very small likelihood of vanishing

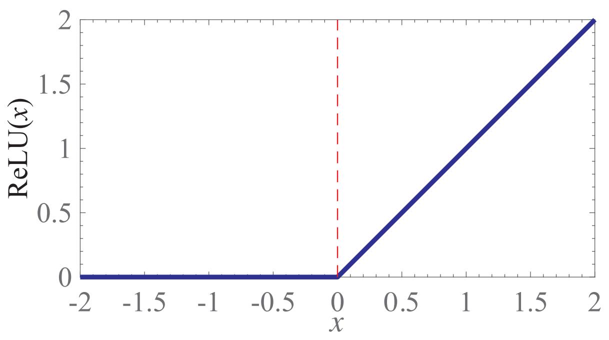


Fig. 3. Rectified linear unit (ReLU) activation function.

gradient. For $x > 0$, there is always a constant gradient and, as such, the learning is much faster. On the contrary, with sigmoid, the gradients decrease as the value of x increases. Additionally, in the region $x < 0$, ReLU produces sparse result while sigmoid always tend to produce non-zero values and as a result the depictions are dense.

The output layer of the model has three nodes each representing one of the alloy phases. These nodes accept input from the second hidden layer and apply activation functions to predict the phase. In classification tasks, the softmax activation function is generally used in the output layer. This is a normalized exponential function that represents the probability that the input falls into each of the classes. The softmax function is defined as,

$$\sigma(y_i') = \frac{e^{y_i'}}{\sum_{i=1}^{n} e^{y_i'}}. \tag{10}$$

where y_i' is the prediction vector and $\sigma(y_i')$ the probability. The error of our network was then estimated by comparing the output of the model with the target labels. The cost (loss) function in this case is cross entropy, which is the negative log likelihood and given by the formula, [34]

$$H_y(y') = -\sum_{i=1}^{n} y\log(\sigma(y_i')), \tag{11}$$

where y is one of the three target vectors. Cross entropy gives a metric of the distance between what the model believes the output distribution should be, and what the original distribution really is. This error is then propagated back to the network using gradient descent algorithm with a learning rate of 0.01. The weights and bias are randomly initialized at the start of the training process and updated at each epoch by minimizing the cost function. The accuracy of the network is defined as the number of times the network was able to correctly identify a target.

3. Results and discussion

We first expend our efforts in understanding the 118 data. This is a crucial step before applying any ML algorithm to learn from the available data. We generate a scatter matrix plot using the Seaborn package and then compute the correlation matrix of the features using Pandas library. These two matrices help to understand the relationship between any two features of the dataset and also provide a qualitative as well as quantitative estimation of their interconnection.

We visualize the data using a 5 × 5 scatter matrix plot as shown in Fig. 4. The diagonal subfigures show the histograms of phase distributions if only one out of the five features is used. As can be seen, all of the histograms in any subfigure cannot be separated from each other, implying that no single feature can be used to fully classify the alloy phases. In other words, there exist correlations between the five features and the phase selection in MPEAs result from these correlations, which are displayed in the off-diagonal subfigures of Fig. 4.

To quantitatively describe the correlations, we compute the Pearson correlation coefficient between features x and y defined as,[31]

$$r_{xy} = \frac{1}{n-1}\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{S_x S_y}, \tag{12}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the two features and S_x and S_y are the corresponding standard deviations. Correlation values range from -1 to 1, implying significantly negative or positive relation, respectively. Table 1 listed the computed correlation matrix elements, which are also displayed in Fig. 4. It is straightforward to see from Eq. (12) that the correlation of a feature with. Focusing on the correlation between two different features, the matrix elements lie between -0.69 to 0.73. Of the ten independent correlation matrix elements, six of them are negative, while the remaining are positive. Consistent with Table 1, the scatter matrix plot of $\Delta\chi$ and δ shows a positive correlation, *i. e.*, $\Delta\chi$ appears larger as δ increases. Both $\Delta\chi$ and δ correlate negatively with

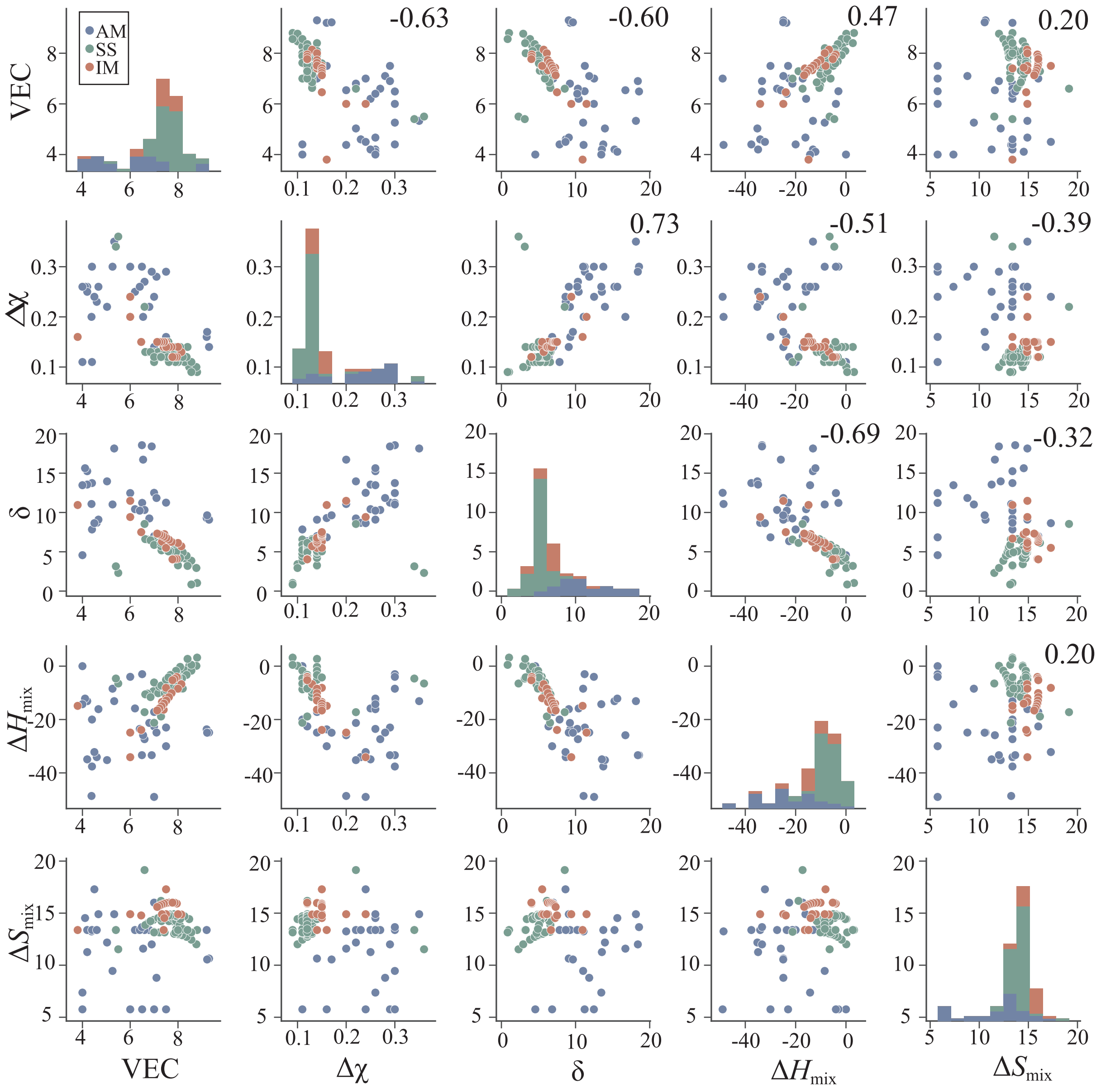


Fig. 4. Scatterplot matrices showing the correlation between 5 features that are link with the phase selection in 118 multi-principal element alloys. The circles are color-coded by different phases. Blue is for amorphous (AM), green for solid solution (SS) and red refers to intermetallic (IM). The axes are the input features. The diagonal histograms determine whether the classifications are separable by individual features. The numbers shown in the upper off-diagonal subfigures refer to the Pearson correlation defined in Eq. (12). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Correlations between the features that are linked to different resulting alloy phases in 118 multi-principal element alloys.

	VEC	$\Delta\chi$	δ	ΔH_{mix}	ΔS_{mix}
VEC	1	−0.63	−0.60	0.47	0.20
$\Delta\chi$	−0.63	1	0.73	−0.51	−0.39
δ	−0.60	0.73	1	−0.69	−0.32
ΔH_{mix}	0.47	−0.51	−0.69	1	0.20
ΔS_{mix}	0.20	−0.39	−0.32	0.20	1

VEC and ΔH_{mix}. Overall, the correlation matrix elements are all of moderate size so that as the 5 features can be used simultaneously as the input features for our NN architecture.

We next apply the NN architecture to train the entire dataset. Fig. 5(a) shows the training loss as a function of the number of epochs. We observe that, at each epoch, the optimization algorithm reduces the loss by updating the weights of the hidden layer neurons thereby optimizing the learning process. In contrast to the training loss, the corresponding accuracy improves as can be seen from Fig. 5(b). The NN model is able to classify the target labels rapidly and an accuracy of about 75% is reached after 10^3 epochs. The learning process further improves the accuracy and saturates at the value of 99.2% after 7.0×10^4 epochs. This accuracy level indicates that only one MPEA, whose phase is mislabeled, in the 118 instances.

In the learning process of updating the weights, the numeric values associated with the neurons in the hidden layers exhibit no physical meanings due to applying the somewhat arbitrary activation functions. This is possibly one main drawback of applying NN to problems in material science. Nevertheless, to gain some relevant physical insights on the relative importance of the features that result in correctly labeled phases, we extract the weights from the NN. Fig. 6 shows the (converged) weight matrix connecting the input layer and the first hidden layer at the last epoch. This matrix is the first one that transforms the

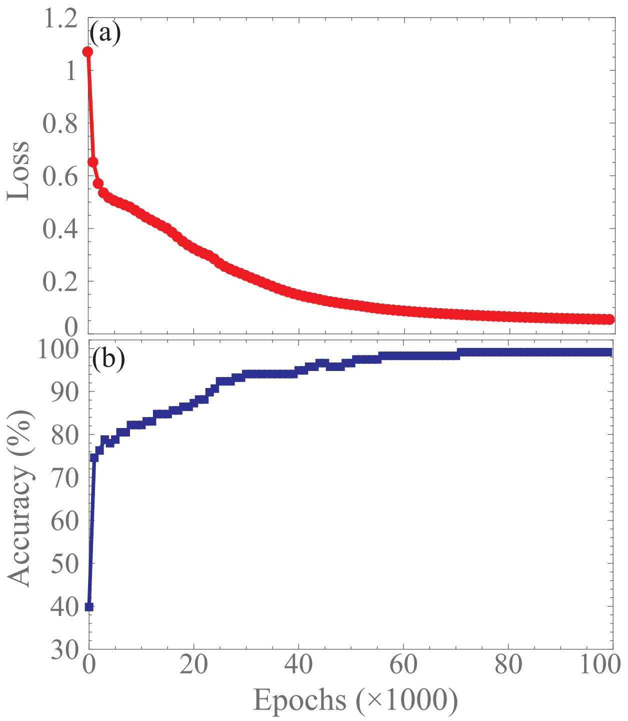


Fig. 5. (a) Training loss and (b) accuracy of the full data set of 118 multi-principal element alloys.

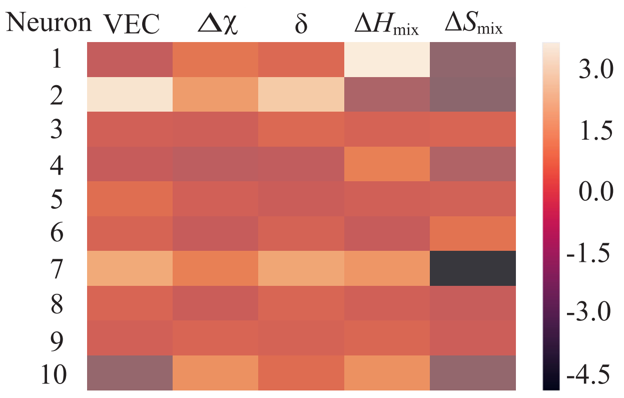


Fig. 6. Weight matrix elements that fully connect the input layer with the neurons in the first hidden layer. The color bar displays the intensity of variance of the weight matrix elements.

physical properties to numerical values, which are further converted to other numeric values via the applied activation function (ReLU in this case). We therefore suggest the first weight matrix is crucial for understanding the physical system that determine the resulting alloy phases.

Singular values from singular value decomposition (SVD) on a general (square or rectangular) matrix provides information about the importance of the features.[35] We perform SVD on the rectangular (10 × 5) weight matrix and obtain singular values of 7.63, 5.58, 1.80, 2.03, and 0.60 for the five features. The relative importance of the five features is thus in the order: VEC>$\Delta\chi > \Delta H_{mix} > \delta > \Delta S_{mix}$, implying that the VEC plays the most important role in determining the phase selection in MPEAs. By contrast, ΔS_{mix} shows the least influence. Interestingly, the importance of VEC has been emphasized by Hume-Rothery, who found that similar crystal structures occur if the VEC of two IM compounds (*e. g.*, Cu_5Sn and Cu_3Al) are the same.[36] In addition, the calculation of ΔS_{mix} is based on ideal SS rather than on real metallic solutions.[3] This deficiency may explain why ΔS_{mix} plays the least important role in the phase selection.

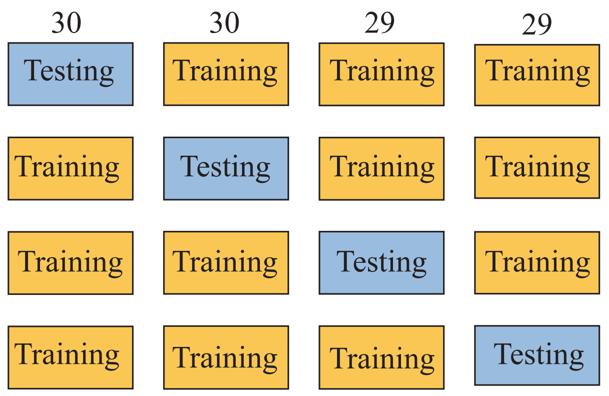


Fig. 7. Illustration of the cross-validation method. The numbers denote the amount of instances in each subset of the 118 data.

Although the accuracy from learning the full data set is higher than 99%, a natural question to ask is how well the trained NN model can be generalized to the data that the model has never seen. To evaluate the generalization accuracy, we use the cross-validation method, which is a common strategy used when the size of a dataset is limited. The idea of the cross-validation process is to split the available dataset into subsets which play the roles of both testing and training dataset. Depending on the number of subsets *N*, the method is called the *N*-fold cross-validation process. In the process, we repeatedly train the NN for *N* times and each time a different subset is used as the testing set and the remaining subsets are treated as the training set. The accuracy is then determined as the average value of the accuracy results from *N* training processes. Here we use 4 folds in the cross-validation method by dividing our dataset into 30, 30, 29, and 29 instances as illustrated in Fig. 7. Before the cross-validation process, we shuffle the full dataset with a random number to ensure that there are a balanced number of the three different phases in each subset.

Fig. 8(a)–(h) shows the training loss and accuracy of the network with 4-fold cross-validation. Each of the sub-figures corresponds to the loss and accuracy of one of the 4 cross-validation datasets. Similar to Fig. 5, the training loss and accuracy increases and decreases, respectively, as the learning progresses. The results of generalization accuracy (see Fig. 8(i)–(l)) obtained from this four cross-validation datasets are 86.7%, 83.3%, 86.2%, and 75.9%, respectively, and the average accuracy is thus 83.0%. This accuracy is expectedly lower than that obtained from training the full dataset, indicating that overfitting occurs in the process of learning from the full dataset. This overfitting is caused by the limited dataset used in this work. Although the prediction accuracy of 83% may still be an acceptable level to predict the phase selection in MPEAs, we suggest that an improved generalization accuracy could be obtained by using a larger dataset.

4. Conclusions

To summarize, we analyzed the correlations between five features that lead to the phase selection in a dataset with 118 data of MPEAs. We also trained a NN model to classify the resulting phases based on the input features. We reached a high accuracy (> 99%) in the learning of the full dataset. The converged weight matrix in the NN model reveals that the most important factor that correlates with the phase selection is the valence electron concentration and the least one is the mixing entropy. We also obtained an average predictive accuracy of higher than 80% in the cross-validation training and testing datasets. In the future work, we will collect more data in order to improve the predictability of phase selection in MPEAs. Alternatively, more hidden layers and neurons to make a deeper NN may also be helpful to enhance the predictive accuracy. Other ML algorithms for classification tasks such as support

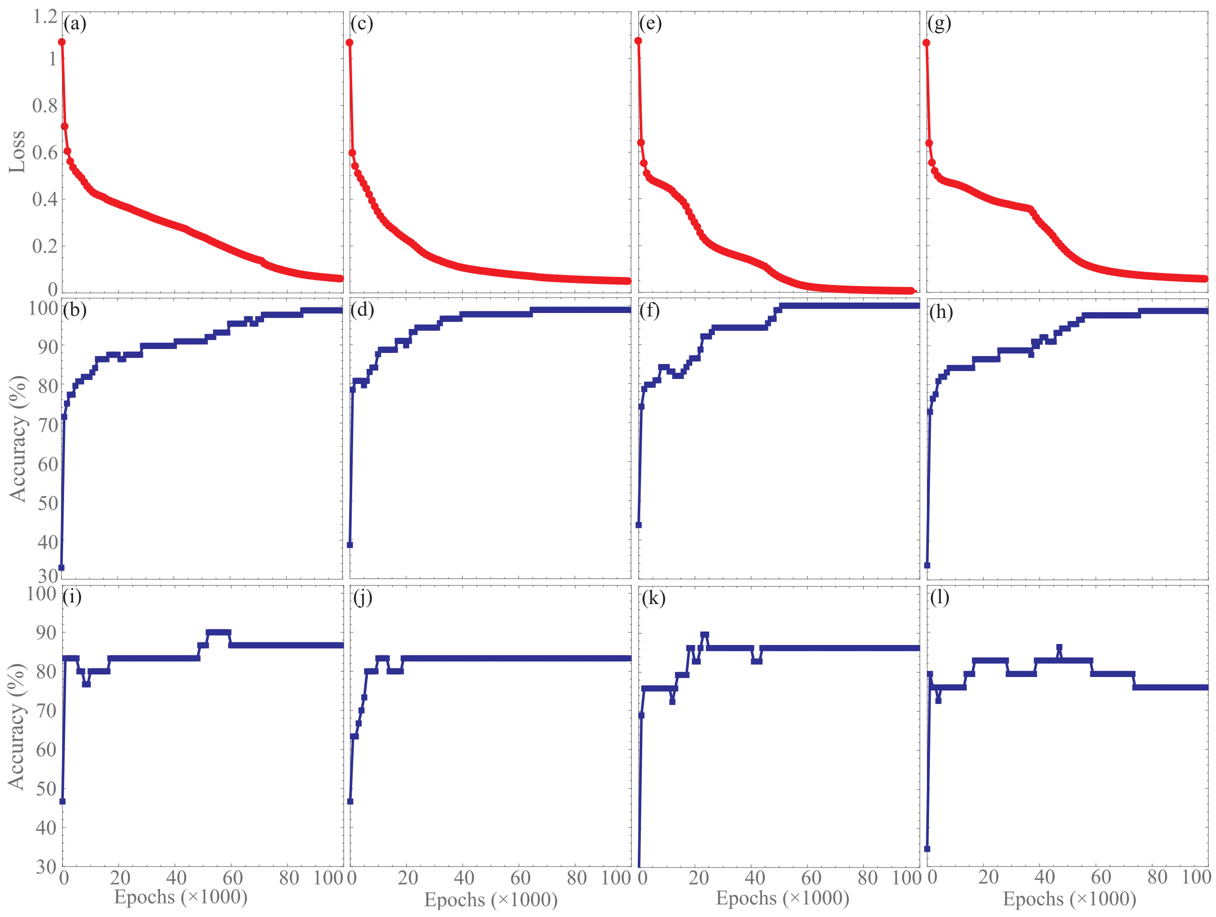


Fig. 8. (a–h) Training loss and accuracy and (i–l) generalization accuracy of the cross-validation data sets.

vector machine [37] may also be helpful to determine the phase selection in MPEAs.

Acknowledgements

We thank the start-up funds from Arizona State University. This work used computational resources of the Texas Advanced Computing Center under Contract No. TG-DMR170070.

References

[1] T.M. Pollock, Nat. Mater. 15 (2016) 809.
[2] W. Gale, T. Totemeier, Smithells Metals Reference Book, Elsevier Science, 2003.
[3] D. Miracle, O. Senkov, Acta Mater. 122 (2017) 448.
[4] J.-W. Yeh, S.-K. Chen, S.-J. Lin, J.-Y. Gan, T.-S. Chin, T.-T. Shun, C.-H. Tsau, S.-Y. Chang, Adv. Eng. Mater. 6 (2004) 299.
[5] B. Cantor, I. Chang, P. Knight, A. Vincent, Mater. Sci. Eng.: A 375–377 (2004) 213.
[6] E. Reineke, O. Inal, Mater. Sci. Eng. 57 (1983) 223.
[7] B. Gludovatz, A. Hohenwarter, K.V. Thurston, H. Bei, Z. Wu, E.P. George, R.O. Ritchie, Nat. Commun. 7 (2016) 10602.
[8] Y. Zhang, T.T. Zuo, Z. Tang, M.C. Gao, K.A. Dahmen, P.K. Liaw, Z.P. Lu, Prog. Mater. Sci. 61 (2014) 1.
[9] B. Murty, J. Yeh, S. Ranganathan, in: B. Murty, J. Yeh, S. Ranganathan (Eds.), High Entropy Alloys, Butterworth-Heinemann, Boston, 2014, pp. 37–56.
[10] X. Yang, Y. Zhang, Mater. Chem. Phys. 132 (2012) 233.
[11] S. Guo, Mater. Sci. Technol. 31 (2015) 1223.
[12] Y. Zhang, Y. Zhou, J. Lin, G. Chen, P. Liaw, Adv. Eng. Mater. 10 (2008) 534.
[13] B. Ren, Z. Liu, D. Li, L. Shi, B. Cai, M. Wang, J. Alloys Comp. 493 (2010) 148.
[14] W. Hume-Rothery, R. Smallman, C. Haworth, The Structure of Metals and Alloys, Monograph and Report Series, Metals & Metallurgy Trust, 1969.
[15] A. Takeuchi, A. Inoue, Mater. Trans. 46 (2005) 2817.
[16] Y. Ye, Q. Wang, J. Lu, C. Liu, Y. Yang, Mater. Today 19 (2016) 349.
[17] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc.,, Secaucus, NJ, USA, 2006.
[18] L. Ward, A. Agrawal, A. Choudhary, C. Wolverton, NPJ Comput. Mater. 2 (2016) 16028.
[19] L. Ward and C. Wolverton, Current Opinion in Solid State and Materials Science 21, 167 (2017), materials Informatics: Insights, Infrastructure, and Methods.
[20] T. Mueller, A.G. Kusne, R. Ramprasad, Machine learning in materials science, Reviews in Computational Chemistry, John Wiley and Sons, Inc, 2016, pp. 186–273.
[21] M. De Jong, W. Chen, R. Notestine, K. Persson, G. Ceder, A. Jain, M. Asta, A. Gamst, Scient. Rep. 6 (2016) 34256.
[22] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, C. Kim, NPJ Comput. Mater. 3 (2017) 54.
[23] B. Medasani, A. Gamst, H. Ding, W. Chen, K.A. Persson, M. Asta, A. Canning, M. Haranczyk, NPJ Comput. Mater. 2 (2016) 1.
[24] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[25] R. Kobayashi, D. Giofré, T. Junge, M. Ceriotti, W.A. Curtin, Phys. Rev. Mater. 1 (2017) 053604.
[26] Y. Zhang, S. Yang, J. Evans, Acta Mater. 56 (2008) 1094.
[27] M.S. Ozerdem, S. Kolukisa, Mater. Des. 30 (2009) 764.
[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Proc. IEEE 86 (1998) 2278.
[29] S. Guo, C. Liu, Prog. Nat. Sci.: Mater. Int. 21 (2011) 433.
[30] W. McKinney, Python for data analysis: data wrangling with Pandas, NumPy, and IPython, second ed., O'Reilly, Beijing u.a., 2017.
[31] J.D. Kelleher, B.M. Namee, A. D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies, The MIT Press, 2015.
[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015, Software Available from < http://tensorflow.org > .
[33] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2017.
[34] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Applications of Mathematics, vol. 31, corrected second ed., Springer, 1997 (missing).
[35] H. Brink, J. Richards, M. Fetherolf, Real-World Machine Learning, Manning, 2016.
[36] U. Mizutani, Hume-Rothery Rules for Structurally Complex Alloy Phases, CRC Press, 2016.
[37] L. Hamel, Knowledge Discovery with Support Vector Machines, Wiley Series on Methods and Applications in Data Mining, Wiley, 2009.