https://doi.org/10.1016/j.actamat.2021.117535

Machine learning assisted design of FeCoNiCrMn high-entropy alloys with ultra-low hydrogen diffusion coefficients

Xiao-Ye Zhou[a], Ji-Hua Zhu[a, *], Yuan Wu[b, *], Xu-Sheng Yang[c], Turab Lookman[d], Hong-Hui Wu[e,f*]

[a] Guangdong Province Key Laboratory of Durability for Marine Civil Engineering, School of Civil Engineering, Shenzhen University, Shenzhen, Guangdong, 518060, PR China

[b] State Key Laboratory for Advanced Metals and Materials, University of Science and Technology Beijing, Beijing, 100083, China

[c] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

[d] AiMaterials Research LLC, Santa Fe, New Mexico 87501, United States

[e] School of Materials Science and Engineering, University of Science and Technology Beijing, Beijing, 100083, China

[f] Department of Chemistry, University of Nebraska, Lincoln, NE, 68588, USA

[*] Corresponding authors: zhujh@szu.edu.cn (J.H. Zhu), wuyuan@ustb.edu.cn (Y. Wu), hhwuaa@connect.ust.hk (H.H. Wu)

Abstract

The broad compositional space of high entropy alloys (HEA) is conducive to the design of HEAs with targeted performance. Herein, a data-driven and machine learning (ML) assisted prediction and optimization strategy is proposed to explore the prototype FeCoNiCrMn HEAs with low hydrogen diffusion coefficients. The model for predicting hydrogen solution energies from local HEA chemical environments was constructed via ML algorithms. Based on the inferred correlation between atomic structures and diffusion coefficients of HEAs built using ML models and kinetic Monte Carlo simulations, we employed the whale optimization algorithm to explore HEA atomic structures with low hydrogen diffusion coefficients. HEAs with low H diffusion coefficients were found to have high Co and Mn content. Finally, a quantitative relationship between the diffusion coefficient and chemical composition is proposed to guide the design of HEAs with low H diffusion coefficients and thus strong resistance to hydrogen embrittlement.

Keywords: machine learning; high entropy alloy; hydrogen embrittlement; material design;

1. Introduction

The demand for hydrogen (H) energy is rapidly growing in the current society[1-3]. However, the wide application of H energy faces great challenges in the storage and transportation of H gas for that the presence of mobile H atoms in metals could render serious deterioration in the mechanical properties of metals, especially ductility and durability, inducing the so-call hydrogen embrittlement (HE). The mobile H atoms are prone to accumulate at crystal defects *i.e.*, grain boundaries [4, 5], interfaces [6-8], and micro voids [9, 10], inducing decohesion or stress concentration, and consequently crack initiation and propagation. One common strategy to increase the HE resistance of metals is to suppress severe H accumulation at defects by slowing down H diffusion through the dispersion of irreversible traps like precipitates[11] or reversible traps like grain boundaries [12] and dislocation walls [13]. However, to trap enough diffusive H, the defects should have sufficient density and proper dispersion in the matrix. Otherwise, H atoms would accumulate at the defects and cause embrittlement. Therefore, novel effective methodologies for trapping diffusive H are in demand.

High entropy alloys (HEAs) have attracted intensive interest due to their unique properties such as good ductility[14], high thermal stability[15, 16], corrosion resistance[17, 18], and excellent low-temperature mechanical properties[19]. Particularly, certain HEAs, such as the prototype Cantor alloy (FeCoNiCrMn) [20-24], have been found to be much less susceptible to HE than commercially used Ni-based alloys and stainless steel, even at a higher level of H charging. The multiple principal components in HEAs lead to a vast compositional space for designing alloys with strong HE resistance. However, an effective way for designing HE resistant HEAs still requires an in-depth understanding of the underlying mechanisms of HE resistance in HEAs. It has been proposed that the highly fluctuated local chemical environment and the distorted lattice of HEAs result in a wide distribution of H diffusion energy barriers. The peaks/valleys in the potential energy landscape create numerous H traps [25] in the HEA lattice. To develop an effective design strategy for HE resistant HEAs, it is necessary to identify how the local chemical environment is correlated with the local H trapping energy and clarify how the chemical composition influences the diffusion coefficient.

Data-driven material design is an emerging field of research that can search for the optimal material with great efficiency by avoiding the traditional trial-error material development process [26-29]. Here we propose a data-driven and machine learning (ML) assisted strategy to design HE resistant non-equimolar FeCoNiCrMn HEAs. Our approach is to calculate the H solution energies using density functional theory (DFT) at critical sites of the HEA lattice, followed by ML algorithms to correlate the local chemical environment with H solution energies. H diffusion coefficients are subsequently calculated from H diffusion barriers by kinetic Monte Carlo (kMC) simulations [10, 30]. Whale optimization algorithms (WOA) [31] then allow us to explore HEA atomic structures with low H diffusion coefficients. Finally, using a polynomial model, we construct a mapping between the H diffusion coefficient and the chemical composition of the HEA. The model suggests that we can regulate the H diffusion coefficient of the HEA by tuning its chemical composition. In particular, HEAs with low H diffusion coefficients are predicted to have high Co and Mn content, while those with high H diffusion coefficients have high Fe and Ni content. The quantitative model linking the H diffusion coefficient with the chemical composition of HEA is expected to guide the design of high-performance HEAs with low H diffusion coefficients and strong resistance to HE.

2. Theoretical Methods

2.1. DFT calculations of the H solution energies

In face-centered cubic (FCC) crystals, H atoms preferentially occupy the octahedral interstitial (OI) sites (Fig. 1a). H atoms at the tetrahedral interstitials (TIs, see Fig. 1a) are at relatively higher energy states. The diffusion pathway of an H atom in a perfect FCC crystal starts from one OI, climbs across a saddle point, passes through a neighboring TI, then diffuses across another saddle point, and finally arrives at its neighboring OI [32], as shown in Figs. 1b and 1c viewed along the <100> and <110> directions, respectively. The saddle point positions and energies were calculated with the climbing image nudged elastic band method (CINEB) [33]. The TIs and OIs of a perfect FCC lattice can be obtained from the geometric characteristics of the FCC lattice. After several CINEB calculations, we find that the saddle points can be easily located at the centers of the triangles formed by the three nearest atoms in the FCC lattice. This allows us to identify the location of saddle points based on geometry rather than performing DFT calculations. The OIs, TIs and saddle points are the critical sites that determine the diffusion path and barriers of a H atom in an FCC crystal. The H solution energies at the critical sites in FCC crystals are calculated using

$$E_{H-solu} = E_{H+bulk} - E_{bulk} - E_H \, , \tag{1}$$

where E_{H+bulk} is the total energy with zero-point energy (ZPE) correction of the relaxed lattice when the H atom is embedded in the interstitial or saddle point, E_{bulk} is the energy of the relaxed bulk lattice, E_H is the energy of an H atom in a vacuum.

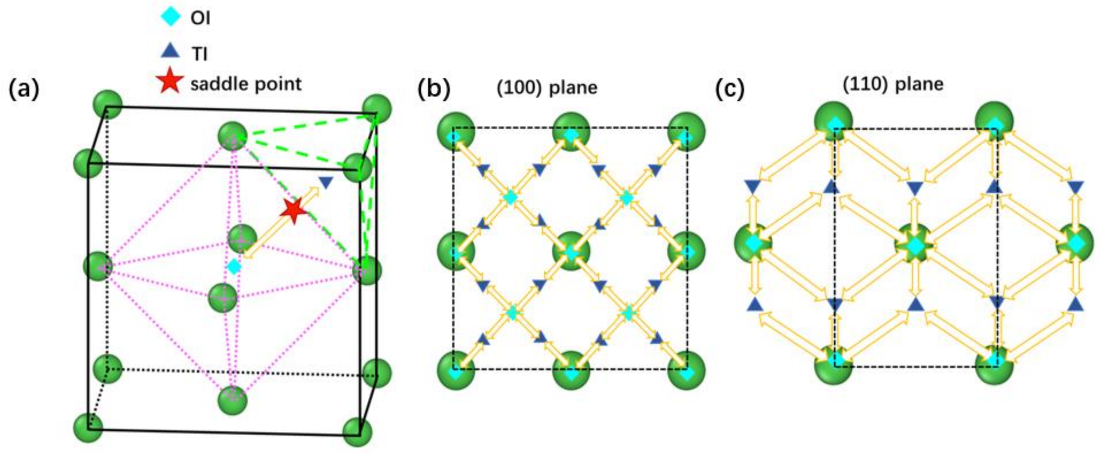

Fig. 1. Critical sites in an FCC crystal (a) The positions of an OI, TI, and saddle point in FCC crystal. (b-c) The trajectory between TIs and OIs viewed along the (b) <100> and (c) <110> directions.

The DFT calculations were performed using the Vienna ab initio Simulation Package (VASP) [34, 35]. Perdew, Burke, and Ernzerhof (PBE) [36] functional of generalized gradient approximation with projector augmented wave (PAW) [37] was adopted to describe the electronic structures of HEAs. The plane-wave basis kinetic energy cut-off was set to 400 eV. All calculations were carried out with spin-polarization switched on. During the geometrical optimizations, all atoms and the simulation cells were allowed to relax, so that the lattice distortion was taken into consideration when calculating the H solution energies. The total energy was converged to less than 10^{-6} eV for the electronic structure optimization. The convergence criterion for geometric optimization for calculating H solution energy was set to 0.1 eV, and the criterion for optimizing the atomic structure of the lattice was set to 0.001 eV. The atomic structure of the lattice was relaxed firstly, and then the H atoms were placed at the TIs, and OIs in the lattice. As the original structure was close to the final structure, the optimization usually finishes within 4 steps.

As one of the limitations of DFT calculations is that the simulation system is finite, it is difficult to directly simulate a real HEA system by DFT. In a real HEA lattice, even though the overall chemical composition is equimolar, the local structure might have elemental segregation that makes the composition deviate from equimolar [38-40]. Therefore, it is inadequate to simulate only five-component equimolar FeCoNiCrMn HEAs within DFT calculations. Hence, we build atomic structures with the number of components ranging from one to five, covering different combinations of the five alloying elements, as shown in Table 1. The H solution energies at all the critical sites in these structures are then calculated by DFT.

Table 1. Structures for DFT calculations of the H solution energies

Pure	Binary	Ternary	Quaternary	Quinary (equimolar FeCoNiCrMn)
Fe	Fe-Co	Fe-Co-Ni	Fe-Co-Ni-Cr	equi-HEA-1
	Fe-Ni	Fe-Ni-Cr		
Co	Fe-Cr	Fe-Ni-Mn	Fe-Ni-Cr-Mn	equi-HEA-2
	Fe-Mn	Co-Ni-Cr		
Ni	Co-Ni	Co-Ni-Mn	Fe-Co-Ni-Mn	equi-HEA-3
	Co-Cr	Ni-Cr-Mn		
Cr	Co-Mn	Fe-Co-Mn	Co-Ni-Cr-Mn	equi-HEA-4
	Ni-Cr	Fe-Cr-Mn		
Mn	Ni-Mn	Fe-Co-Cr	Fe-Co-Cr-Mn	equi-HEA-5
	Cr-Mn	Co-Cr-Mn		

2.2. kMC simulations

The kMC simulations are used for calculating H diffusion coefficients. Here we only consider the low H-concentration scenario. One H atom was firstly placed in a randomly selected OI site. Due to the complicated local chemical environment of HEA, each OI has 8 different neighboring TIs, so there are 8 different barriers for hopping from OI to TI. The hopping rate to the *i*th neighboring TI is calculated by

$$k_i = v_0 \exp\left(-\frac{\Delta E_i}{k_B T}\right) \tag{2}$$

For the H atom at this OI, the total hopping rate is

$$k_{tot} = \sum_i k_i \tag{3}$$

The probability of this H hopping to the *i*th TI is k_i/k_{tot}. The selection of the TI for hopping is then based on the probability of each hopping event. The hopping from TI to OI is determined by the same method.

The time interval for each hopping event is calculated by,

$$\Delta t = -\frac{1}{k_{tot}}\ln(r_2) \tag{4}$$

where r_2 is a random number between 0 and 1. The hopping continues until the preset number of steps has been reached. The H diffusion coefficient is then calculated by

$$D = \frac{1}{6t} MSD \tag{5}$$

where *MSD* is the mean squared displacement of the H atom, *t* is the total time of the kMC simulation.

The attempt frequency v_0 used here is 10^{-14} m^2 s^{-1}, which is calibrated to match the H diffusion coefficient in coarse-grained Ni[41]. The python code for the kMC simulation is presented in the Supplementary Materials.

2.3. WOA

The WOA method, which is a meta-heuristic optimization algorithm [31], was applied to explore the HEA structure with low H diffusion coefficients. WOA mimics the hunting strategy of humpback whales, which starts from encircling the prey with multiple whales, followed by the bubble-net attacking phase, and then the prey searching phase. WOA is an efficient algorithm for structural optimization and displays an excellent balance between exploration and exploitation, enabling searching across a broad space for the optimal solution. However, the WOA method needs modification to search the structural and compositional space of the HEA structure. To be consistent with the structures for the DFT calculation of the H solution energies in quinary HEA, we use the 2×2×5 FCC supercell with a total of 80 atoms as the optimization object. To search the structural and compositional space simultaneously, each atom of the supercell is allowed to choose from the five elements, Fe, Co, Ni, Cr, and Mn. Hence, the atomic structure of the HEA supercell has 5^{80} permutations. By using integers ranging from 1 to 5 to represent the element type and an 80-component vector to represent the permutation of the atoms, we can then iteratively optimize the atomic structure of the HEA supercell to gradually lower the H diffusion coefficient using WOA. We use a vector, **X**, to represent the atomic structures of HEA. **X** has two key characteristics: all the elements are integer, and the elemental index should be in the range of 1-5. In addition, we expect the optimized structure to contain all the five elements of HEA, so the structures with less than five elements are excluded. During the optimization, the components of **X** can easily exceed the range of 1-5. The **X** will be initiated with random integers in the range of 1-5 if any components of **X** are lower than 1 or higher than 5. We used 5 whales for hunting within one WOA iteration and performed 300 iterations. The python codes for WOA are presented in the Supplementary Materials.

3. Results

The H solution energy distributions in pure Ni and HEA structures are plotted in Fig. 2a and Fig. 2b, respectively. It is clear that E_{H-solu} in HEAs have wider distributions than that in Ni, and the E_{H-solu} in HEAs are highly inhomogeneous. The H diffusion barriers in certain areas are significantly higher than others, implying the H diffusion pathway may be more rugged and the H atoms are highly dragged in HEAs. These results are consistent with the findings of Ren *et al*. [42] and Zhou *et al*. [25] that the lattices of HEAs can effectively trap H atoms. However, how the chemical composition of the HEA and the local chemical environment affect the H diffusion behavior, and to what extent the H diffusion coefficient can be reduced by the inhomogeneous distribution of E_{H-solu} in HEA, remain to be resolved.

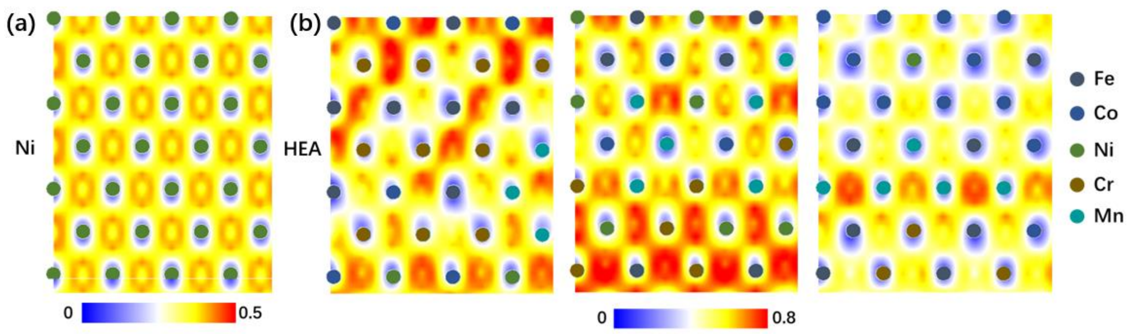

Fig. 2. The H solution energy distributions in pure (a) Ni and (b) HEA structures.

The overall flow chart for designing FeCoNiCrMn HEA with low H diffusion coefficients is presented in Fig. 3. After obtaining the H solution energy through DFT calculations, we adopt the Smooth Overlap of Atomic Positions (SOAP) descriptor

[43-46] to represent the chemical environment of a given H atom embedded in a metal lattice and then explore an ML model to correlate its SOAP descriptor to its solution energy. Using the ML model, H solution energies in new HEA structures that are not in the dataset can be evaluated. The prediction of H solution energies in new HEA structures is important because we need to find the optimal HEA structures from a broad compositional and structural space. The H diffusion coefficient of the new structure can be consequently calculated by kMC simulations. It is WOA that allows us to search the compositional and structural space of HEAs for structures with low diffusion coefficients. These structures generated by WOA serve as input to the ML model mentioned earlier to predict the H solution energy distribution. The H diffusion coefficient is then calculated from the H solution energy distribution. Using the large dataset containing HEA structures and H diffusion coefficients, the relationship between the H diffusion coefficient with the chemical composition of the HEA can then be accurately described by a 2-degree polynomial model. The details of each part in Fig. 3 are elaborated hereafter.

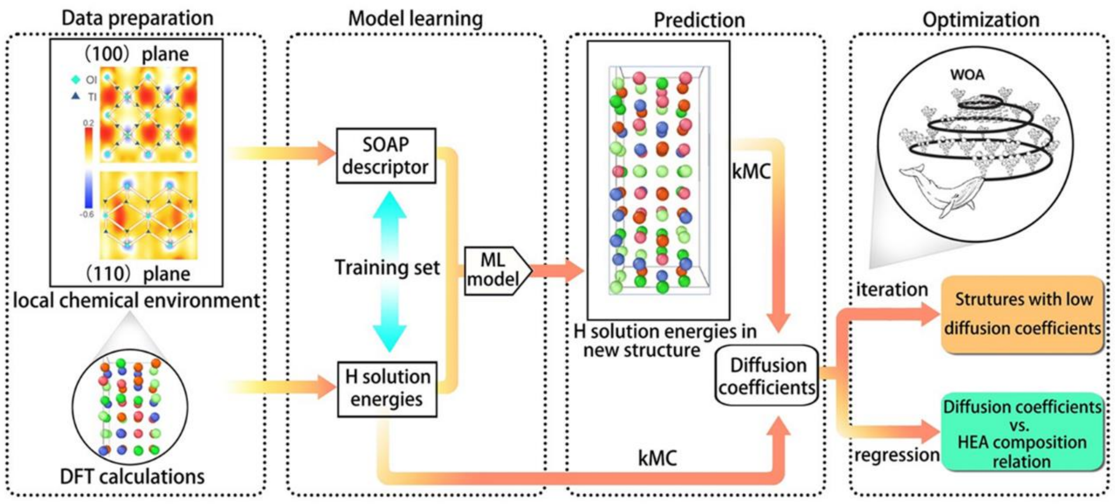

Fig. 3. Graphic representation of the design strategy. Data preparation, model training, prediction, and optimization process for designing FeCoNiCrMn HEA with low H diffusion coefficients.

3.1. Data preparation

In total, we built 35 atomic structures (Table1) for DFT calculations and calculated the H solution energies at the critical sites in the structures. The corresponding atomic structure files can be found in the Supplementary Data. The number of critical sites in each calculated structure ranges from 392 (in binary alloys) to 880 (in quinary alloys). The complete size of the dataset, which contains the atomic positions of critical sites and their corresponding H solution energies in the 35 structures, is 24566. The atomic positions of the critical sites and the corresponding H solutions of each atomic structure can also be found in the Supplementary Data. The calculated H solution energy distributions in several representative binary, ternary, quaternary, and quinary alloys are shown in Fig. 4. The rest of the H solution energy distributions are shown in Fig. S1-4 in the Supplementary Materials. We see that in certain structures, such as FeMn and FeCoMn, the H solution energy distributions are 'narrower', and the energy distributions at saddle and TI sites overlap less. In structures such as CoCr and FeCoNiCrMn, the H solution energy distributions are 'wider' with partially overlapping distributions at saddle and TI points. In addition, the average solution energies at the critical sites of different structures are quite different. For example, the energies at OI sites of the FeNiCrCo structure are centered around -3.1 eV, whereas the energies at OI sites of the FeCoMn structure are centered around -2.95 eV, indicating that the OIs in FeNiCrCo are deeper H traps than those in FeCoMn. The differences in H solution energy distributions will cause large deviations in the lattice-trapping ability, hence we further investigated the influence of HEA structure and composition on the H diffusion coefficient.

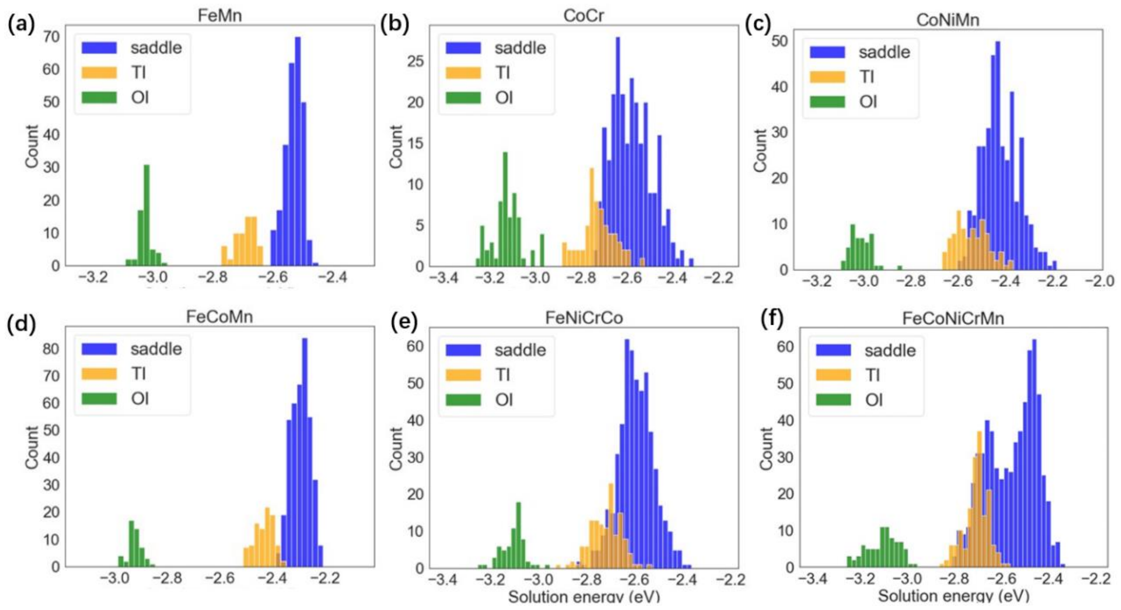

Fig. 4. H solution energy distributions at the critical sites, i.e., the saddle points, TIs and OIs in typical binary (a) FeMn alloy, (b) CoCr alloy, ternary (c) CoNiMn alloy, (d) FeCoMn alloy, (e) quaternary FeNiCrCo alloy, and quinary (f) FeCoNiCrMn HEA.

3.2. Model training

To define features for each H atom embedded in the 35 structures, we adopt SOAP descriptors [43] as they accurately describe the local chemical environment surrounding the target H atom within a certain cutoff. There are three important hyperparameters to be optimized when constructing a SOAP descriptor, namely, the radial cutoff (r_{cut}), the maximum number of radial basis functions (n_{max}), and angular degree of the spherical harmonics (l_{max}). To evaluate the influence of these hyperparameters on the model accuracy, we use the mean square error (MSE) as a performance measure and use linear regression as the ML model. The whole data set was split into a training set (85 %), validation set (5 %), and test set (10 %). We used the validation set for selecting hyperparameters of the SOAP descriptors, while the test set was used for evaluating the performances of the ML models. The whole data set can be found in Supplementary Data (see the 'x_arr.npy' and 'y_arr.npy' files for the SOAP descriptors and the H solution energies)

The variation of MSE with increasing r_{cut}, n_{max} and l_{max} is depicted in Figs. 5a-c, which indicates that for $r_{cut} = 7$, $n_{max} = 4$ and $l_{max} = 4$, the MSE is near-optimal for predicting H solution energies in the validation set. The generated SOAP descriptor is a vector with 1500 components. However, many components of the SOAP descriptor are close to zero, contributing little to the learning target. Therefore, we used principal component analysis (PCA) to reduce the dimension of the features. As shown in Fig. 5d, MSE reaches a steady-state value provided the feature number is no less than 600. Therefore, we use 600 as the feature number of the original SOAP descriptors for further analysis.

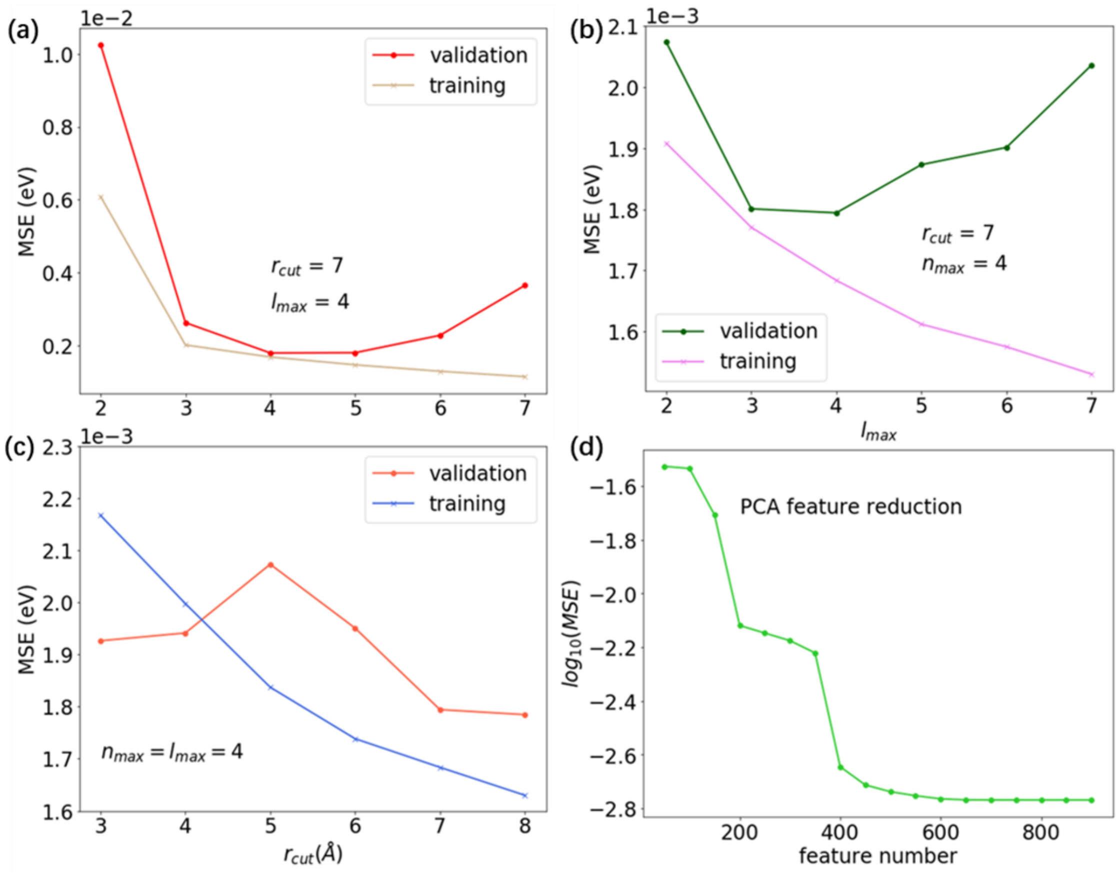

Fig. 5. Hyperparameter analysis for SOAP descriptors. Variation of MSE with increasing (a) r_{cut}, (b) n_{max} and (c) l_{max}, and the (d) number of features after PCA.

After constructing the SOAP descriptors for the critical sites and calculating the corresponding H solution energies, we built the ML model to correlate the local chemical environment of an H atom at a critical site with its solution energy. As presented in Fig. 6, an ML model as simple as ridge regression with built-in cross-validation[47] (referred to as RidgeCV hereafter) can give satisfactory results. RidgeCV outperformed random forest but was slightly inferior to the neural network. Based on the performance of the three ML models, the neural network was selected as the ML model for further prediction of the H solution energy distribution based on the atomic structure of a given HEA.

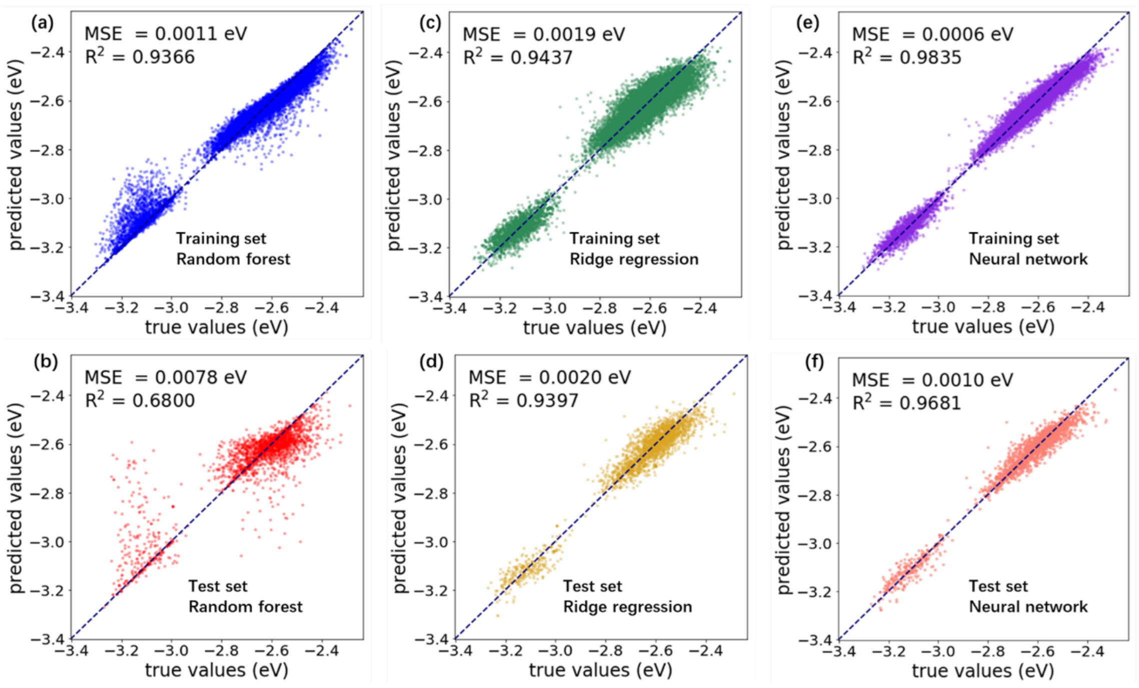

Fig. 6. Performance of the ML models on the training set and the test set, respectively. (a, b) random forest model, (c, d) ridge regression model, (e, f) neural network model.

3.3. Prediction and Optimization

With the trained ML model, we can now predict the complete H solution energies at the critical sites of the HEA atomic structures generated during WOA. The H diffusion coefficient of the structure is calculated through kMC. The H diffusion coefficients at 300 K of the 35 structures are listed in Table S1 in Supplementary

Material. At this point, we have established the correlation between the atomic structure of the alloy and its H diffusion coefficient.

The WOA optimizations are then performed to explore HEA structures with low H diffusion coefficients. Despite the broad search space of the WOA optimization, the H diffusion coefficient can be effectively reduced by 2 orders of magnitude after merely 300 WOA steps with five whales for hunting, wherein the optimized structures are non-equimolar. Because the local chemical environments and the corresponding H solution energies in the training set are all calculated from either pure-metal structures or equimolar-alloy structures, the ML model displays relatively low accuracy in the prediction of H diffusion coefficients of non-equimolar HEA. To address this issue, we performed additional DFT calculations on the optimized non-equimolar structures to obtain H solution energies at the critical sites, and then iteratively added these DFT calculation results along with their corresponding SOAP descriptors to the training set to enhance the accuracy of the ML model predictions (see Fig. 7a). The iteration stops when the convergence criterion between the predicted and DFT calculated H diffusion coefficients is reached.

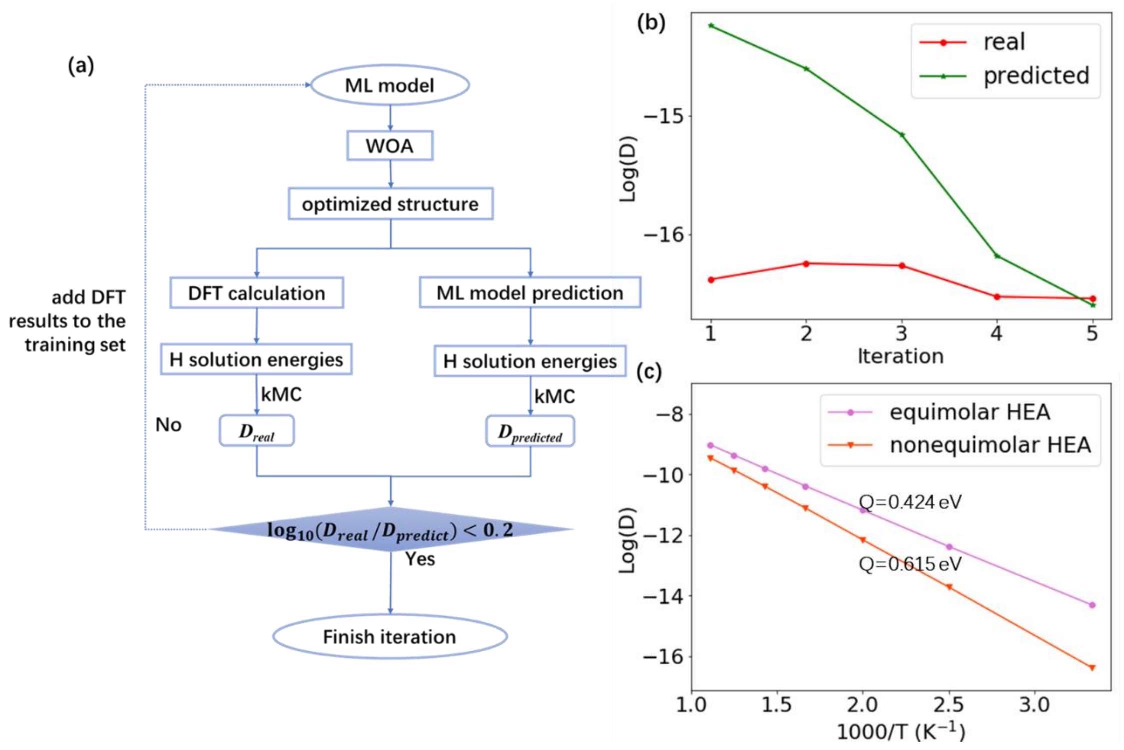

Fig. 7. The iteration process and the results of the iteration loops. (a) The iteration

process to expand the training set and increase the accuracy of ML model prediction on non-equimolar HEA structures. (b) The evolution of the diffusion coefficient (300 K) during the iteration process. (c) The Arrhenius relations of the diffusion coefficient of the equimolar HEA and the optimized non-equimolar structures. The diffusion activation energies, Q, are calculated through the slopes of the curves.

The iteration process in Fig. 7a was performed five times to guarantee the accuracy of the ML model. Fig. 7b shows the evolution of the logarithm of the diffusion coefficient during the iteration process. After five iteration loops, the predicted and the DFT calculated H diffusion coefficients reached the convergence criterion. Five optimized structures generated in the iteration process are labeled non-equi-HEA-1 to non-equi-HEA-5. (See Fig. S5 in Supplementary Materials for the atomic structures.) The H diffusion coefficient of the non-equimolar HEAs that were evaluated by kMC using the DFT calculated H solution energies are listed in Table S1 in Supplementary Materials.

The optimized non-equimolar HEA structures have H diffusion coefficients in the order of 10^{-16}, which is two orders of magnitude smaller than those of the equimolar HEAs. To evaluate the ability of the non-equimolar HEA structures to trap H atoms, we calculated the activation energy of H diffusion in a typical equimolar HEA and the optimized non-equimolar HEA. The Arrhenius equation of the diffusion coefficient as a function of temperature is given by Eq. 2.

$$D(T) = D^0 \exp\left(\frac{Q}{RT}\right) \tag{6}$$

A linear fitting of the logarithm of the diffusion coefficient as a function of 1/T can be described as in Eq. 3,

$$\ln[D(T)] = \ln D^0 + \frac{Q}{R}\frac{1}{T} \tag{7}$$

from which the activation energy of H diffusion in the structures can be obtained. Fig. 7c shows the Arrhenius plots of the diffusion coefficient of an equimolar HEA and an optimized non-equimolar HEA at 300 K. The activation energy of H diffusion in the optimized non-equimolar HEA is 0.191 eV higher than that in the equimolar HEA. The increase of H diffusion activation energy by structural optimization indicates that the optimized non-equimolar HEA has higher trapping ability and may serve as reversible H trapping sites with comparable trapping energies as typical reversible H traps, such as dislocations (0.11 eV for screw dislocation and 0.18 for edge dislocation in aluminum, as calculated by DFT[48]) and low-angle grain boundaries which are often considered to be composed of multiple edge dislocations[49].

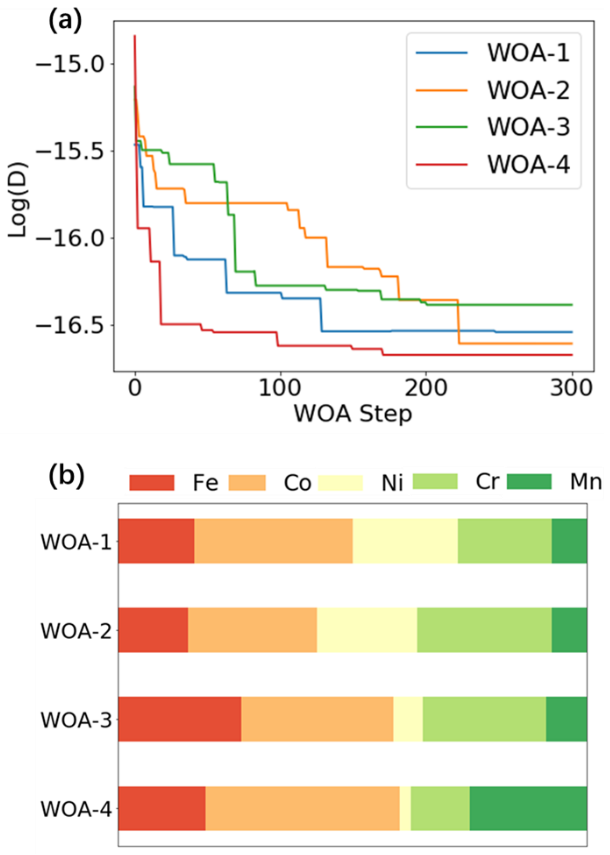

Fig. 8. (a) The evolution of the diffusion coefficient during WOA. (b) Chemical compositions of the optimized structures after WOA.

With the expanded training set after the iterative process, the ML model can accurately predict the H solution energies in non-equimolar HEAs. We then performed another four WOAs to obtain four optimized HEA structures with the improved ML model. The evolution of the diffusion coefficient during WOA and the chemical compositions of the optimized structures after WOA are shown in Fig. 8a and 8b, respectively. Fig. 8a shows that the H diffusion coefficient can be rapidly reduced, indicating that WOA is efficient in exploring the broad and high-dimensional searching space of the current optimization problem. We can see that owing to the broad composition and structure space of the HEA atomic structure, the WOA exploration arrives at different local optimal points each time, and it is not easy to locate the global minimum. However, all the optimized structures have H diffusion coefficients considerably lower than those of the unoptimized ones. The optimized structures are labeled WOA-1 to WOA-4 and their atomic structures are plotted in Fig. S5 in the Supplementary Materials. The charge density distributions of the four optimized structures are representatively shown in Fig. S6 in the Supplementary Materials. It can be seen that the charge density distribution is highly inhomogeneous, which would result in the highly varied H solution energies, and then the formation of various H traps in the HEA lattices. All of the optimized structures are non-equimolar, and typically have high Co content and low Ni content. This suggests that there might be certain relationships between the chemical composition and the H diffusion coefficient of the structure, which make it possible to achieve a low H diffusion coefficient by tuning the chemical composition.

4. Discussion

We have shown that the optimized structures generated by WOA can act as reversible H trapping sites, with trapping ability comparable to that of dislocation cores and low-angle grain boundaries. As the optimization target, the H diffusion coefficient has multiple local minima, thus this kind of lattice trap can be abundant in HEAs. Unlike H traps such as grain boundaries and dislocations, lattice traps result from the heterogeneity of the local chemical environment of the HEA lattice, which will not cause severe segregation of H atoms and can thus avoid the risk of H-induced interface decohesion, stress concentration, and crack initiation. This quality of lattice traps might explain the high HE resistance observed in many HEAs[17, 20, 22-24, 50, 51].

The WOA not only finds the HEA structures with low H diffusion coefficients but also generates massive data containing the HEA chemical compositions and their corresponding H diffusion coefficients during the optimization process, which makes it possible to further uncover the characteristics of HEA structures with low H diffusion coefficients, and the relationship between the chemical composition of the HEA structure and its H diffusion coefficient.

Fig. 9a and Fig. 9b present the chemical composition distributions of the HEAs, which were generated from WOA, with H diffusion coefficients lower than 10^{-16} m^2 s^{-1} (Fig. 9a), or higher than 10^{-14} m^2 s^{-1} (Fig. 9b). The HEAs with low H diffusion coefficients have high Co and Mn content, while those with high H diffusion coefficients have high Fe and Ni content. The influence of Cr is not as obvious, but still has a positive effect on reducing the H diffusion coefficient.

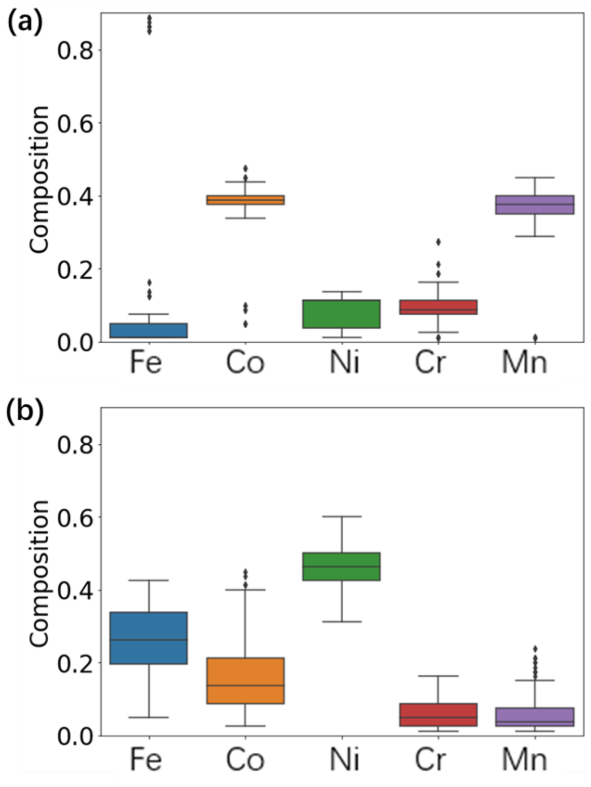

Fig. 9. The chemical composition distributions of the HEAs with (a) low (< 10^{-16} m^2 s^{-1}) or (b) high (> 10^{-14} m^2 s^{-1}) H diffusion coefficients.

We further carried out linear regression with the composition of Fe, Co, Ni, Cr, Mn as features and the logarithmic H diffusion coefficient as the learning target. The dataset was split into the training set and the test set with a split ratio of 4:1. The performance of linear regression is shown in Fig. 10a and Fig. 10b. Linear regression gives an R^2 value of 0.75 on the test set, indicating that although the variance is large, there does exist a certain relationship between the H diffusion coefficient and the chemical composition. The large variance may come from the nonlinear dependency of the H diffusion coefficient on the chemical composition. It also indicates that as the H diffusion coefficient decreases, the linear prediction becomes less accurate, as shown by the lower-left region of Fig. 10a and 10b, suggesting that the lack of model complexity manifests itself when predicting low H diffusion coefficients. The linear model gives an explicit expression of the logarithm of H diffusion coefficient as a function of chemical components, *i.e.*,

$$\log(D) = 0.096Fe - 0.47Co + 3.05Ni - 1.01Cr - 1.66Mn - 15.05 \quad (8)$$

where *Fe, Co, Ni, Cr, Mn* represent the atomic ratio of the elements. Eq. 8 suggests that the increased Co, Cr, and Mn ratio can reduce the H diffusion coefficient whereas an increase of Fe and Ni might result in the opposite effect. Note that the linear model fails to make accurate predictions for structures with H diffusion coefficients lower than 10^{-16} m^2 s^{-1}.

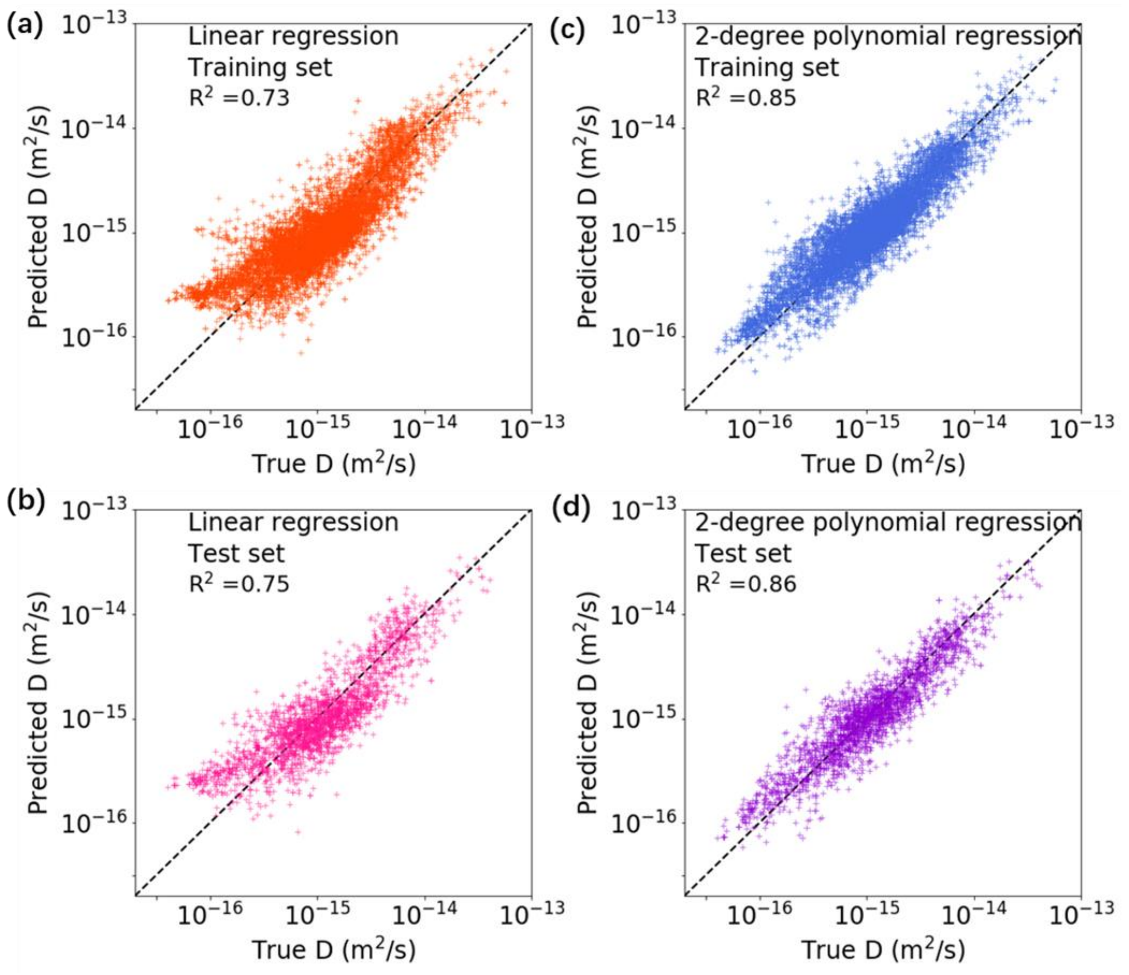

Fig. 10. The performance of the linear model on the training set (a) and the test set (b) and the performance of the 2-degree polynomial model on the training set (c) and the test set (d).

To predict the H diffusion coefficients with higher accuracy, we adopted 2-degree polynomial regression and obtained a higher R^2 value of 0.86 on the test set (Fig. 10d). The coefficients and intercept of the 2-degree polynomial regression are listed in Table 2. The polynomial model can predict H diffusion coefficients lower than 10^{-16} m^2 s^{-1} with improved accuracy, which could be a better choice when quantitative prediction is necessary. Although the linear model is less accurate, it gives a more intuitive understanding of the influence of each element on the H diffusion coefficient, which can serve as a general estimation for designing HEAs with low diffusion coefficients.

Table 2. The coefficients and intercept of the 2-degree polynomial regression

Intercept	***FeFe***	***FeCo***	***FeNi***	***FeCr***	***FeMn***	***CoCo***	***CoNi***
-15.3483	-1.4491	0.2543	2.5556	-0.4160	-0.9448	3.4148	4.3146
CoCr	***CoMn***	***NiNi***	***NiCr***	***NiMn***	***CrCr***	***CrMn***	***MnMn***
0.2795	6.0980	1.6180	-1.1773	-5.8116	-2.0221	-0.3837	2.8789

5. Conclusion

In summary, we have proposed a data-driven and ML-assisted prediction and optimization strategy to assist the design of FeCoNiCrMn HEAs with low H diffusion coefficients. The neural network was used to bridge the relationship between the SOAP descriptors and the H solution energies. With the help of kMC, the H diffusion coefficients based on the H solution energies can be accurately evaluated, and the relationship between the atomic structure of HEA to its H diffusion coefficient can be built. From the data collected from the WOA processes, we found that HEAs with low H diffusion coefficients typically have high Co and Mn content, while those with high H diffusion coefficients have high Fe and Ni content. We then developed linear and polynomial models between the chemical composition and the H diffusion coefficient to guide the design of HEAs with low H diffusion coefficient and high HE resistance.

Acknowledgments

This research was supported by the National Key Research and Development Program of China (2018YFE0124900), the National Natural Science Foundation of China (51778370, 51901013, 52071023, 51921001, 51871016, 51861165204, and 52061135207) and the Fundamental Research Funds for the Central Universities (University of Science and Technology Beijing) (No. 06500135).

Competing interests

The authors declare no competing interests.

References

[1] N. Mahmood, Y. Yao, J.-W. Zhang, L. Pan, X. Zhang, J.-J. Zou, Electrocatalysts for Hydrogen Evolution in Alkaline Electrolytes: Mechanisms, Challenges, and Prospective Solutions, Adv. Sci. 5(2) (2018) 1700464.

[2] S. Zhao, J. Berry-Gair, W. Li, G. Guan, M. Yang, J. Li, F. Lai, F. Corà, K. Holt, D.J.L. Brett, G. He, I.P. Parkin, Hydrogen Evolution: The Role of Phosphate Group in Doped Cobalt Molybdate: Improved Electrocatalytic Hydrogen Evolution Performance (Adv. Sci. 12/2020), Adv. Sci. 7(12) (2020) 2070067.

[3] D.A. Cullen, K.C. Neyerlin, R.K. Ahluwalia, R. Mukundan, K.L. More, R.L. Borup, A.Z. Weber, D.J. Myers, A. Kusoglu, New roads and challenges for fuel cells in heavy-duty transportation, Nat. Energy 6(5) (2021) 462-474.

[4] S. Huang, D. Chen, J. Song, D.L. McDowell, T. Zhu, Hydrogen embrittlement of grain boundaries in nickel: an atomistic study, NPJ Comput. Mater. 3(1) (2017) 28.

[5] X. Zhou, D. Marchand, D.L. McDowell, T. Zhu, J. Song, Chemomechanical Origin of Hydrogen Trapping at Grain Boundaries in fcc Metals, Phys. Rev. Lett. 116(7) (2016) 075502.

[6] Z. Zhang, K.L. Moore, G. McMahon, R. Morana, M. Preuss, On the role of precipitates in hydrogen trapping and hydrogen embrittlement of a nickel-based superalloy, Corros. Sci. 146 (2019) 58-69.

[7] B. Malard, B. Remy, C. Scott, A. Deschamps, J. Chêne, T. Dieudonné, M.H. Mathon, Hydrogen trapping by VC precipitates and structural defects in a high strength Fe–Mn–C steel studied by small-angle neutron scattering, Mater. Sci. Eng. A 536 (2012) 110-116.

[8] S. Zhang, J. Wan, Q. Zhao, J. Liu, F. Huang, Y. Huang, X. Li, Dual role of nanosized NbC precipitates in hydrogen embrittlement susceptibility of lath martensitic steel, Corros. Sci. (2019) 108345.

[9] G. Lv, M. Zhang, H. Zhang, Y. Su, Hydrogen diffusion and vacancy clusterization in iron, Int. J. Hydrogen Energy 43(32) (2018) 15378-15385.

[10] J. Hou, X.-S. Kong, X. Wu, J. Song, C.S. Liu, Predictive model of hydrogen trapping and bubbling in nanovoids in bcc metals, Nature Materials 18(8) (2019) 833-839.

[11] R. Shi, Y. Ma, Z. Wang, L. Gao, X.-S. Yang, L. Qiao, X. Pang, Atomic-scale investigation of deep hydrogen trapping in NbC/α-Fe semi-coherent interfaces, Acta Mater. 200 (2020) 686-698.

[12] Y. Bai, Y. Momotani, M.C. Chen, A. Shibata, N. Tsuji, Effect of grain refinement on hydrogen embrittlement behaviors of high-Mn TWIP steel, Mater. Sci. Eng. A 651 (2016) 935-944.

[13] L. Chen, X. Xiong, X. Tao, Y. Su, L. Qiao, Effect of dislocation cell walls on hydrogen adsorption, hydrogen trapping and hydrogen embrittlement resistance, Corros. Sci. 166 (2020) 108428.

[14] Z. Li, K.G. Pradeep, Y. Deng, D. Raabe, C.C. Tasan, Metastable high-entropy dual-phase alloys overcome the strength–ductility trade-off, Nature 534(7606) (2016) 227-230.

[15] J. Wang, S. Wu, S. Fu, S. Liu, M. Yan, Q. Lai, S. Lan, H. Hahn, T. Feng, Ultrahigh hardness with exceptional thermal stability of a nanocrystalline CoCrFeNiMn high-entropy alloy prepared by inert gas condensation, Scripta Mater. 187 (2020) 335-339.

[16] P. Sathiyamoorthi, J. Basu, S. Kashyap, K.G. Pradeep, R.S. Kottada, Thermal stability and grain boundary strengthening in ultrafine-grained CoCrFeNi high entropy alloy composite, Mater. Des. 134 (2017) 426-433.

[17] H. Luo, S.S. Sohn, W. Lu, L. Li, X. Li, C.K. Soundararajan, W. Krieger, Z. Li, D. Raabe, A strong and ductile medium-entropy alloy resists hydrogen embrittlement and corrosion, Nat. Commun. 11(1) (2020) 3081.

[18] J.R. Scully, S.B. Inman, A.Y. Gerard, C.D. Taylor, W. Windl, D.K. Schreiber, P. Lu, J.E. Saal, G.S. Frankel, Controlling the corrosion resistance of multi-principal element alloys, Scripta Mater. 188 (2020) 96-101.

[19] Gludovatz, Bernd, Hohenwarter, Anton, Catoor, Dhiraj, Chang, Edwin, H., George, A fracture-resistant high-entropy alloy for cryogenic applications, Science (2014).

[20] Y. Zhao, D.-H. Lee, W.-J. Kim, M.-Y. Seok, J.-Y. Kim, H.N. Han, J.-Y. Suh, U. Ramamurty, J.-i. Jang, Influence of pre-strain on the gaseous hydrogen embrittlement resistance of a high-entropy alloy, Mater. Sci. Eng. A 718 (2018) 43-47.

[21] H. Luo, Z. Li, D. Raabe, Hydrogen enhances strength and ductility of an equiatomic high-entropy alloy, Sci. Rep. 7(1) (2017) 9892.

[22] Y. Zhao, D.-H. Lee, M.-Y. Seok, J.-A. Lee, M.P. Phaniraj, J.-Y. Suh, H.-Y. Ha, J.-Y. Kim, U. Ramamurty, J.-i. Jang, Resistance of CoCrFeMnNi high-entropy alloy to gaseous hydrogen embrittlement, Scripta Mater. 135 (2017) 54-58.

[23] H. Luo, W. Lu, X. Fang, D. Ponge, Z. Li, D. Raabe, Beating hydrogen with its own weapon: Nano-twin gradients enhance embrittlement resistance of a high-entropy alloy, Mater. Today 21(10) (2018) 1003-1009.

[24] Z. Pu, Y. Chen, L.H. Dai, Strong resistance to hydrogen embrittlement of high-entropy alloy, Mater. Sci. Eng. A 736 (2018) 156-166.

[25] X. Zhou, W.A. Curtin, First principles study of the effect of hydrogen in austenitic stainless steels and high entropy alloys, Acta Mater. 200 (2020) 932-942.

[26] J.M. Rickman, H.M. Chan, M.P. Harmer, J.A. Smeltzer, C.J. Marvel, A. Roy, G. Balasubramanian, Materials informatics for the screening of multi-principal elements and high-entropy alloys, Nat. Commun. 10(1) (2019) 2618.

[27] J. Rickman, G. Balasubramanian, C. Marvel, H. Chan, M.T. Burton, Machine learning strategies for high-entropy alloys, J. Appl. Phys. 128 (2020) 221101.

[28] D. Xue, P.V. Balachandran, J. Hogden, J. Theiler, D. Xue, T. Lookman, Accelerated search for materials with targeted properties by adaptive design, Nat. Commun. 7(1) (2016) 11241.

[29] J. Rickman, T. Lookman, S.V. Kalinin, Materials Informatics: From the Atomic-Level to the Continuum, Acta Mater. 168 (2019).

[30] Y.A. Du, J. Rogal, R. Drautz, Diffusion of hydrogen within idealized grains of bcc Fe: A kinetic Monte Carlo study, Phys. Rev. B: Condens. Matter 77(13) (2008) 134305

[31] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Adv. Eng. Softw. 95 (2016) 51-67.

[32] E. Wimmer, W. Wolf, J. Sticht, P. Saxe, C. Geller, R. Najafabadi, G. Young, Temperature-Dependent Diffusion Coefficients from ab initio Computations: Hydrogen in Nickel, Phys. Rev. B: Condens. Matter 2008, 77 (13), 134305.

[33] G. Henkelman, B.P. Uberuaga, H. Jónsson, A climbing image nudged elastic band method for finding saddle points and minimum energy paths, J. Chem. Phys. 113(22) (2000) 9901-9904.

[34] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Comput. Mater. Sci. 6(1) (1996) 15-50.

[35] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B: Condens. Matter 54(16) (1996) 11169-11186.

[36] J.P. Perdew, K. Burke, M. Ernzerhof, Generalized Gradient Approximation Made Simple, Phys. Rev. Lett. 77(18) (1996) 3865-3868.

[37] G. Kresse, D. Joubert, From ultrasoft pseudopotentials to the projector augmented-wave method, Phys. Rev. B: Condens. Matter 59(3) (1999) 1758-1775.

[38] J.Y. Zhang, Q.F. He, J. Li, Y. Yang, Chemical fluctuation enabling strength-plasticity synergy in metastable single-phase high entropy alloy film with

gigapascal yield strength, Int. J. Plast. 139 (2021) 102951.
[39] Y. Wu, F. Zhang, X. Yuan, H. Huang, X. Wen, Y. Wang, M. Zhang, H. Wu, X. Liu, H. Wang, S. Jiang, Z. Lu, Short-range ordering and its effects on mechanical properties of high-entropy alloys, J. Mater. Sci. Technol. 62 (2021) 214-220.
[40] Q.-J. Li, H. Sheng, E. Ma, Strengthening in multi-principal element alloys with local-chemical-order roughened dislocation pathways, Nat. Commun. 10(1) (2019) 3563.
[41] A. Oudriss, J. Creus, J. Bouhattate, E. Conforto, C. Berziou, C. Savall, X. Feaugas, Grain size and grain-boundary effects on diffusion and trapping of hydrogen in pure nickel, Acta Mater. 60(19) (2012) 6814-6828.
[42] X.L. Ren, P.H. Shi, W.W. Zhang, X.Y. Wu, Q. Xu, Y.X. Wang, Swamps of hydrogen in equiatomic FeCuCrMnMo alloys: First-principles calculations, Acta Mater. 180 (2019) 189-198.
[43] L. Himanen, M.O.J. Jäger, E.V. Morooka, F. Federici Canova, Y.S. Ranawat, D.Z. Gao, P. Rinke, A.S. Foster, DScribe: Library of descriptors for machine learning in materials science, Comput. Phys. Commun. 247 (2020) 106949.
[44] M.O.J. Jäger, E.V. Morooka, F. Federici Canova, L. Himanen, A.S. Foster, Machine learning hydrogen adsorption on nanoclusters through structural descriptors, NPJ Comput. Mater. 4(1) (2018) 37.
[45] M. Wagih, P.M. Larsen, C.A. Schuh, Learning grain boundary segregation energy spectra in polycrystals, Nat. Commun. 11(1) (2020) 6376.
[46] S. Fujii, T. Yokoi, C. Fisher, H. Moriwake, M. Yoshiya, Quantitative prediction of grain boundary thermal conductivities from local atomic environments, Nat. Commun. 11 (2020).
[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[48] M. Yamaguchi, M. Itakura, T. Tsuru, K.-i. Ebihara, Hydrogen-Trapping Energy in Screw and Edge Dislocations in Aluminum: First-Principles Calculations, Materials Transactions 62(5) (2021) 582-589.
[49] Q. Zhu, Q. Huang, C. Guang, X. An, S.X. Mao, W. Yang, Z. Zhang, H. Gao, H. Zhou, J. Wang, Metallic nanocrystals with low angle grain boundary for controllable plastic reversibility, Nat. Commun. 11(1) (2020) 3100.
[50] H. Luo, Z. Li, W. Lu, D. Ponge, D. Raabe, Hydrogen embrittlement of an interstitial equimolar high-entropy alloy, Corros. Sci. 136 (2018) 403-408.
[51] C.K. Soundararajan, H. Luo, D. Raabe, Z. Li, Hydrogen resistance of a 1 GPa strong equiatomic CoCrNi medium entropy alloy, Corros. Sci. 167（2020) 108510.

Supplementary Materials for

Machine learning assisted design of FeCoNiCrMn high entropy alloys with ultra-low hydrogen diffusion coefficients

Xiao-Ye Zhou[a], Ji-Hua Zhu[a, *], Yuan Wu[b, *], Xu-Sheng Yang[c], Turab Lookman[d], Hong-Hui Wu[e,f*]

[a] Guangdong Province Key Laboratory of Durability for Marine Civil Engineering, School of Civil Engineering, Shenzhen University, Shenzhen, Guangdong, 518060, PR China

[b] State Key Laboratory for Advanced Metals and Materials, University of Science and Technology Beijing, Beijing, 100083, China

[c] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

[d] AiMaterials Research LLC, Santa Fe, New Mexico 87501, United States

[e] School of Materials Science and Engineering, University of Science and Technology Beijing, Beijing, 100083, China

[f] Department of Chemistry, University of Nebraska, Lincoln, NE, 68588, USA

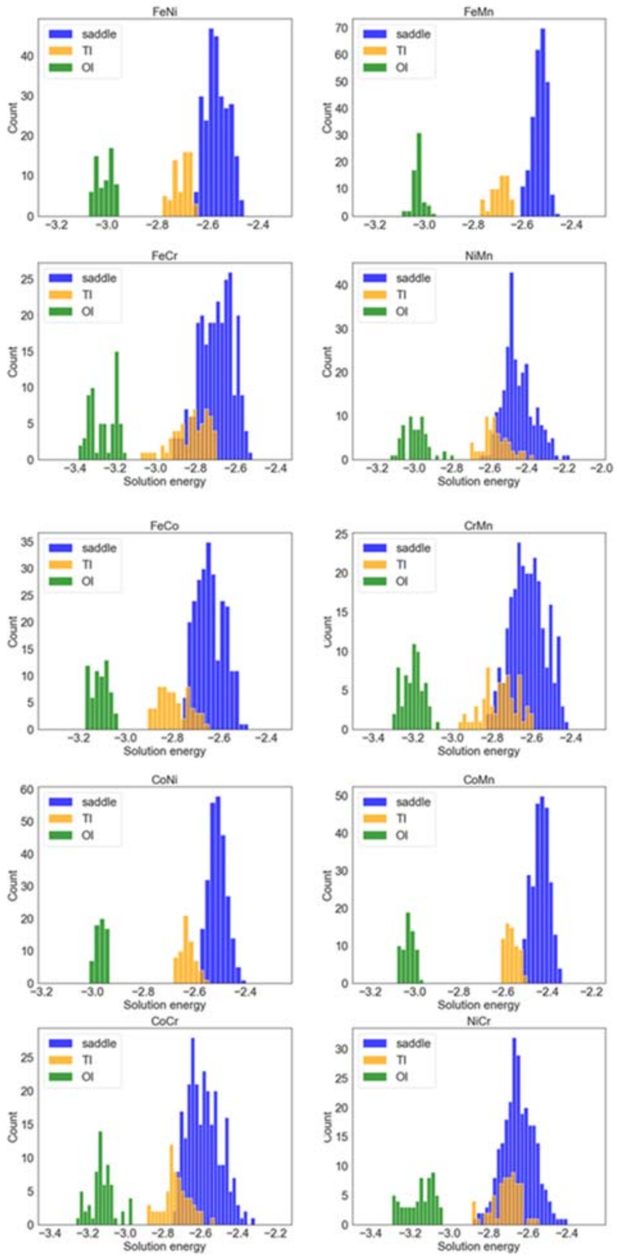

Fig. S1. H solution energy distributions at saddle points, TIs, and OIs in binary alloys.

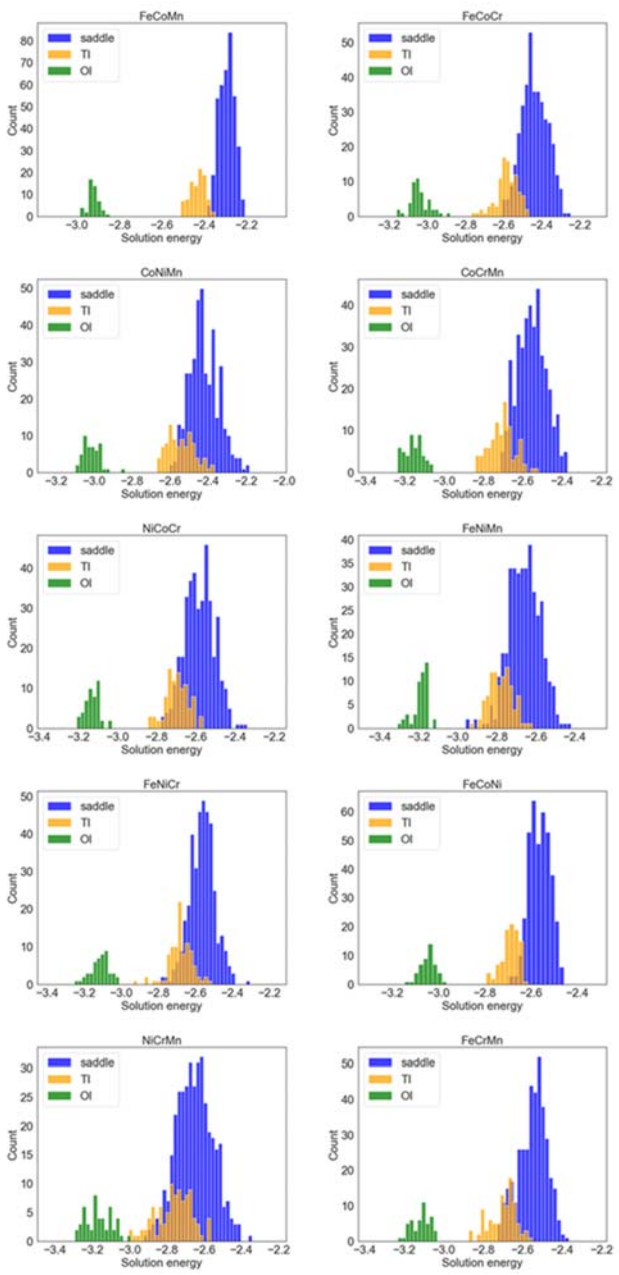

Fig. S2. H solution energy distributions at saddle points, TIs, and OIs in ternary alloys.

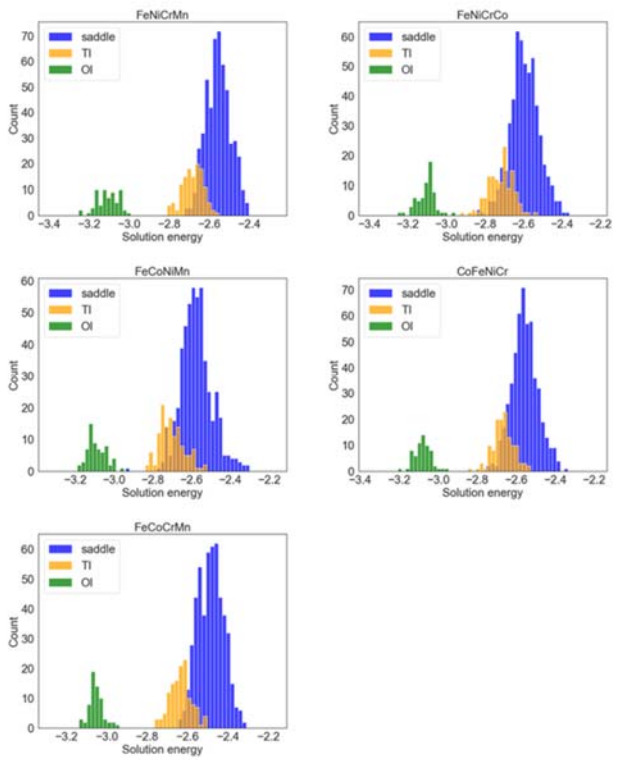

Fig. S3. H solution energy distributions at saddle points, TIs, and OIs in quaternary alloys.

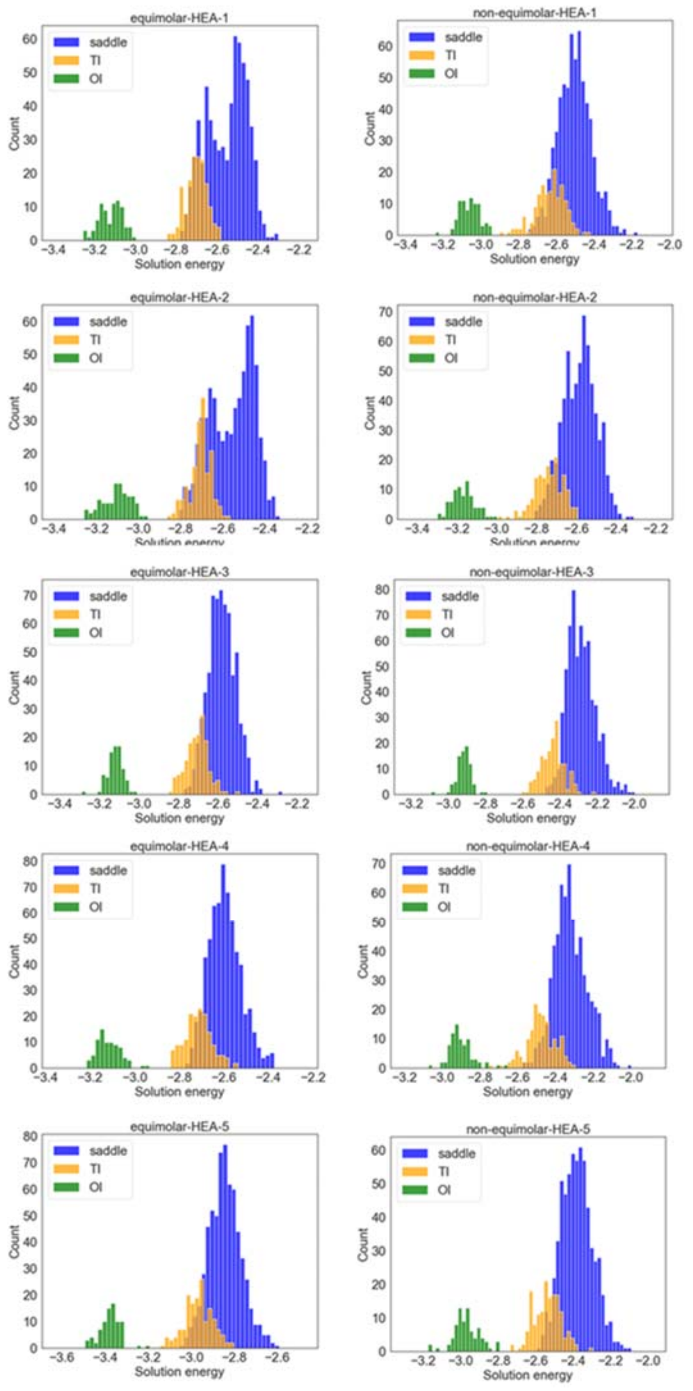

Fig. S4. H solution energy distributions at saddle points, TIs, and OIs in quinary alloys.

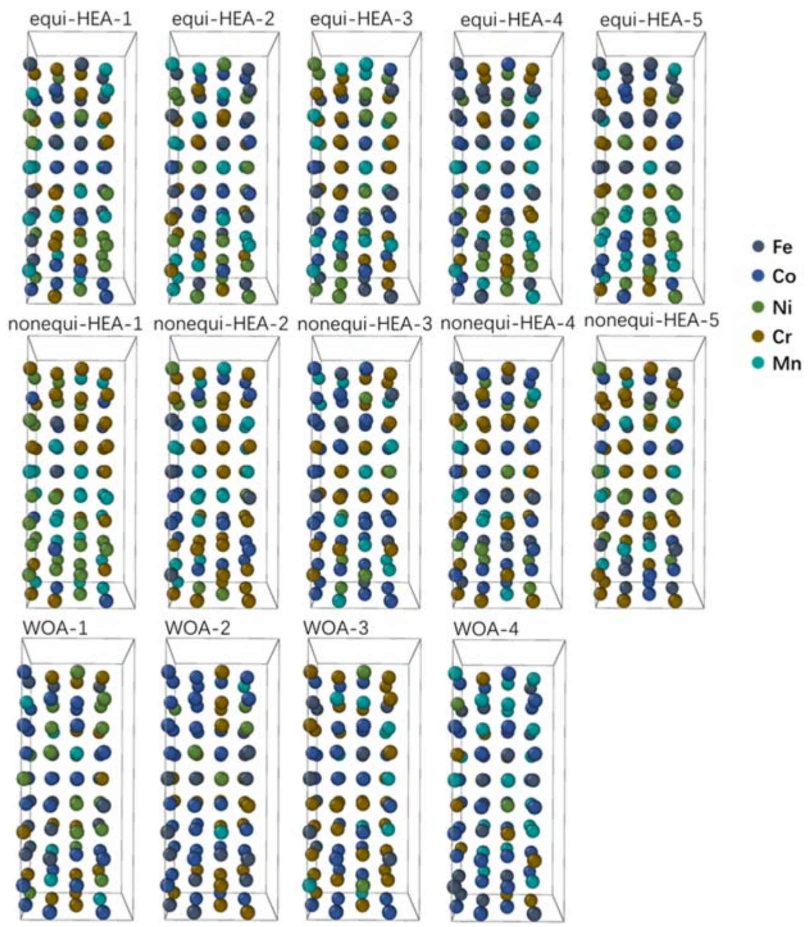

Fig. S5. Atomic arrangements of the equimolar HEA1-5, the optimized non-equimolar HEA1-5, and the structures generated by four WOA processes, WOA1-4.

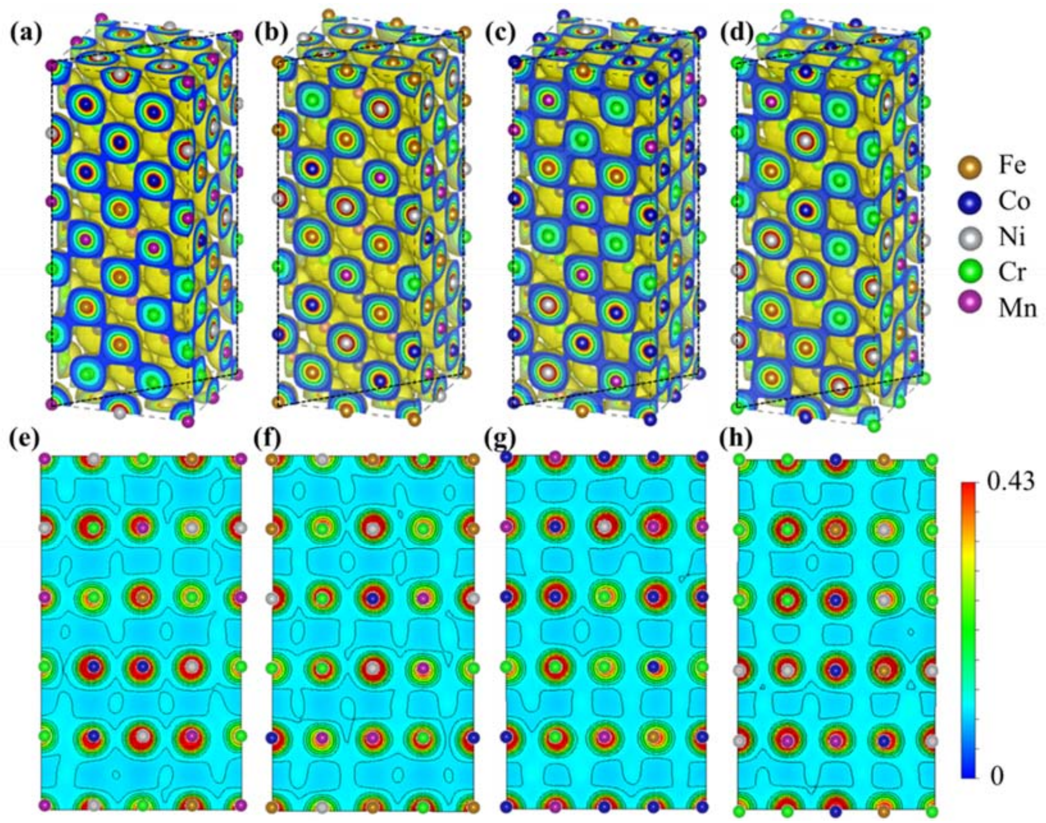

Fig. S6. (a-d) Charge density distributions in the four optimized HEA structures. (e-h) Cross-section views of the charge density distributions.

Table S1. The H diffusion coefficients D ($m^2\ s^{-1}$) calculated by kMC in various atomic systems.

Pure		**Binary**		**Ternary**		**Quaternary**		**Quinary**	
Fe	1.26E-16	CoCr	8.41E-16	FeCoNi	8.29E-15	CoFeNiCr	4.49E-15	equi-HEA-1	2.13E-15
		CoNi	1.52E-14	FeNiCr	2.00E-15			equi-HEA-2	5.44E-15
Co	1.18E-13	FeCo	2.47E-14	FeNiMn	7.94E-14	FeCoNiMn	5.38E-13	equi-HEA-3	1.30E-15
		FeCr	3.17E-15	NiCoCr	4.35E-15			equi-HEA-4	4.97E-15
Ni	2.36E-14	FeNi	2.46E-14	NiCrMn	3.08E-14	FeNiCrCo	1.88E-14	equi-HEA-5	3.84E-15
		FeMn	3.37E-15	CoCrMn	1.40E-16			non-equi-HEA-1	5.72E-15
Cr	4.00E-16	NiCr	6.64E-15	CoNiMn	4.76E-16	FeNiCrMn	3.46E-16	non-equi-HEA-2	2.50E-15
		CoMn	1.02E-16	FeCoCr	1.43E-16			non-equi-HEA-3	6.97E-16
Mn	8.21E-14	CrMn	1.25E-15	FeCoMn	2.77E-16	FeCoCrMn	7.99E-16	non-equi-HEA-4	6.66E-17
		NiMn	1.21E-15	FeCrMn	9.48E-16			non-equi-HEA-5	2.57E-17

Description of the Supplementary Data

x_arr.py: The scaled SOAP descriptors constructed using the optimized parameters (rcut = 7, nmax = l max = 4, after reduced to 600 components after PCA)

y_arr.py: The H solution energies

Note that the x_arr and y_arr represent the whole data set.

compositions.npy: The chemical compositions for the linear and the polynomial model

diffusion_coefficients.npy: The corresponding diffusion coefficients

The **position_energy_converted_X** file contains the positions of the H atoms (1-3 columns) in the X model and their corresponding H solution energies (the 4th column). The last column represents the interstitial type, with 3 representing the saddle point, 2 representing TI and 1 representing OI.

The **POSCAR_X** files contain the POSCAR of the X models.

The python codes for establishing ML models, kMC simulations and WOA optimizations

The SOAP descriptors were constructed based on the ideal FCC lattices, meaning that the lattice distortion induced by different elements is not considered. The positions of H atoms were also determined from the ideal lattice. We use the ideal lattice instead of the relaxed lattice and the relaxed H positions to ensure the predictability of the ML model in predicting the H solution energies in completely new HEA structures. Using the ideal lattice for constructing SOAP descriptors indicates that only the chemical environments of the H atoms are considered, excluding the influence of lattice distortion induced by multiple components and H embedding, which we cannot predict in new HEA structures. This is important because little variations in the SOAP descriptor can cause a large difference in the predicted H solution energies.

The construction of SOAP descriptors is achieved through the following Python codes.

```
import numpy as np
import ase.io
from dscribe.descriptors import SOAP

# Read in the POSCAR file for the metal matrix
fname = ‘POSCAR’
fhand = open(fname)
header = []
atoms = []
for n, line in enumerate(fhand):
    if n <=7:
        header.append(line)
    if n>7:
        line = line.rstrip().split()
        if len(line) ==3:
            atom = [float(line[0]), float(line[1]), float(line[2])]
            atoms.append(atom)
header[5] = 'H '+ header[5]
header[6] = str(1)+ ' '+header[6]
```

# Read in the H positions, embed the H atom into the metal matrix, and write it to a CONTCAR file

```
for i in range(len(H_positions)):
    fname = 'CONTCAR'
    with open(fname, "w") as fwrite:
        for line in header:
            fwrite.writelines(line)
        atom = H_positions[i]
        line = []
        line.append(str(atom[0]) + '\t')
        line.append(str(atom[1]) + '\t')
        line.append(str(atom[2]) + '    '+ '\n')
        fwrite.writelines(line)
        for atom in atoms:
            line = []
            line.append(str(atom[0]) + '\t')
            line.append(str(atom[1]) + '\t')
            line.append(str(atom[2]) + ' '+'\n')
            fwrite.writelines(line)
```

# Use ase module to read in the CONTCAR file

```
    model=ase.io.read('CONTCAR',format='vasp')
    model.set_pbc([1,1,1])
```

Before constructing the SOAP descriptor, there are several parameters to be tuned before constructing the SOAP descriptors, namely rcut, nmax and lmax. The whole data set has 24566 data points. Here we randomly select 1230 (about 5 % of the whole data set) data points for validation, 20876 (85 %) data points as the training set and 2461 data points as the test set. The training set and the test set were used for model selection. The rcut, nmax and lmax were then tuned for better performance on the validation set.

```
# Parameter setting for the SOAP descriptors
H_soap_desc = []
periodic_desc = SOAP(species=['H','Fe','Co','Ni','Cr','Mn'],rcut=rcut, \
                              average = 'off', nmax=nmax,lmax=lmax,periodic=True,sparse=False)

# Create SOAP descriptor for the H atoms
H_soap = periodic_desc.create(model,positions =[0],n_jobs=-1)
H_soap_desc.append(H_soap[0])
```

After constructing the descriptors for the H atoms, we then **build the ML models to learn the relation between the SOAP descriptors and the H solution energies**. To further improve the prediction accuracy of the ML models, the calculated H solution energies of different structures were shifted by their average values so the H solution energies of each structure are centered around 0 eV. The shift of the H solution energies of a certain structure as a whole will not affect the H diffusion barriers and thus will not influence the prediction on the H diffusion coefficient of the structure.

```
# First, we scale the features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(H_soap_desc)
H_soap_desc = scaler.transform(H_soap_desc)

y_train = solution_energies
x_train = H_soap_desc

# Train the model using RidgeCV
from sklearn.linear_model import RidgeCV
# Train model
nn = RidgeCV()
nn.fit(x_train, y_train)

# Train the model use Random Forest
from sklearn.ensemble import RandomForestRegressor
nn = RandomForestRegressor(max_depth=25, random_state=0)
nn.fit(x_train, y_train)
```

# Train the model use neural network

# The hyperparameters (node_num and reg_para) for the neural network have already been optimized

```
reg_para=0.005
node_num = 100

nn=tf.keras.Sequential()
nn.add(tf.keras.layers.Dense(2*node_num,input_dim=600,kernel_initializer='normal',\

activation='relu',kernel_regularizer=regularizers.l2(reg_para)))
nn.add(tf.keras.layers.Dense(node_num,kernel_initializer='normal',\

activation='relu',kernel_regularizer=regularizers.l2(reg_para)))
nn.add(tf.keras.layers.Dense(1,kernel_initializer='normal'))
optimizer=tf.optimizers.Adam(learning_rate=0.00001)
model.compile(loss='mse',optimizer= optimizer)

nn.fit(x_train, y_train,verbose=0,epochs=100)
```

The H diffusion coefficients are calculated by the kMC simulation. We define two functions to complete the simulation. The first one is for identifying the neighbor list for H jumping, the second one is for the calculation of the diffusion coefficient. The codes of which are as follows:

```
def neighborlist(system):
# Read in atomic positions of the metal matrix (perfect crystal)
    fname = 'POSCAR'
    fhand = open(fname)
    header = []
    atoms = []
    for n, line in enumerate(fhand):
        if n <=7:
            header.append(line)
        if n>7:
            line = line.rstrip().split()
            if len(line) ==3:
                atom = [float(line[0]), float(line[1]), float(line[2])]
                atoms.append(atom)
    lattice_const = float(header[1].rstrip().split()[0])
    xhi=float(header[2].rstrip().split()[0])
    yhi=float(header[3].rstrip().split()[1])
    zhi=float(header[4].rstrip().split()[2])
    for atom in atoms:
        if atom[0]<0.3:
            atoms.append([atom[0]+1,atom[1],atom[2]])
        if atom[1]<0.3:
            atoms.append([atom[0],atom[1]+1,atom[2]])
        if atom[2]<0.3:
            atoms.append([atom[0],atom[1],atom[2]+1])

    for atom in atoms:
        if atom[0]>0.7:
            atoms.append([atom[0]-1,atom[1],atom[2]])
        if atom[1]>0.7:
            atoms.append([atom[0],atom[1]-1,atom[2]])
        if atom[2]>0.7:
            atoms.append([atom[0],atom[1],atom[2]-1])
    atoms = np.array(atoms)
    for atom in atoms:
        atom[0]=round(atom[0]*xhi*lattice_const,3)
        atom[1]=round(atom[1]*yhi*lattice_const,3)
        atom[2]=round(atom[2]*zhi*lattice_const,3)
    atoms = np.unique(atoms,axis=0)

    fname = 'H_positions.dat' #POSCAR file, Direct coordinate, lattice constant on the second
row
    fhand = open(fname)
    H_atoms = []
    for n,line in enumerate(fhand):
        line = line.rstrip().split()
        if len(line) ==3:
            atom = [float(line[0]), float(line[1]), float(line[2]),n]
            H_atoms.append(atom)

    for atom in H_atoms:
        if atom[0]<0.3:
            H_atoms.append([atom[0]+1,atom[1],atom[2],atom[3]])
        if atom[1]<0.3:
            H_atoms.append([atom[0],atom[1]+1,atom[2],atom[3]])
        if atom[2]<0.3:
            H_atoms.append([atom[0],atom[1],atom[2]+1,atom[3]])

    for atom in H_atoms:
        if atom[0]>0.7:
            H_atoms.append([atom[0]-1,atom[1],atom[2],atom[3]])
        if atom[1]>0.7:
            H_atoms.append([atom[0],atom[1]-1,atom[2],atom[3]])
        if atom[2]>0.7:
            H_atoms.append([atom[0],atom[1],atom[2]-1,atom[3]])
    H_atoms = np.array(H_atoms)
    for atom in H_atoms:
        atom[0]=round(atom[0]*xhi*lattice_const,3)
        atom[1]=round(atom[1]*yhi*lattice_const,3)
        atom[2]=round(atom[2]*zhi*lattice_const,3)
    H_atoms = np.unique(H_atoms,axis=0)

# classify the H positions into saddle points, TIs and OIs based on their numbers of neighboring metal atoms
    saddles = []
    for H_atom in H_atoms:
        neighbors = []
        for atom in atoms:
            if cal_distance(H_atom,atom)<2:
                neighbors.append(atom)
        if len(neighbors)==3: #if the H atom has 3 nearest neighbors, then it should be at a saddle point
```

```
            saddles.append(H_atom)


    TIs=[]
    for H_atom in H_atoms:
        neighbors = []
        for atom in atoms:
            if cal_distance(H_atom,atom)<2:
                neighbors.append(atom)
        if len(neighbors)==4: #if the H atom has 4 nearest neighbors, then it should be at a
TI
            TIs.append(H_atom)


    OIs=[]
    for H_atom in H_atoms:
        neighbors = []
        for atom in atoms:
            if cal_distance(H_atom,atom)<2:
                neighbors.append(atom)
        if len(neighbors)==6: #if the H atom has 6 nearest neighbors, then it should be at a
OI
            OIs.append(H_atom)
    H_num=0
    H_atom_inbox=[]
    for H_atom in H_atoms:
        if In_Boundary(H_atom,xhi*lattice_const,yhi*lattice_const,zhi*lattice_const): #
exclude atoms outside the simulation box
            H_num=H_num+1
            H_atom_inbox.append(H_atom)
```

# build the neighbor list based on the connectivity of the TIs, OIs and saddle points

```
    OI_TI_saddle = []
    for OI in OIs:
        if In_Boundary(OI,xhi*lattice_const,yhi*lattice_const,zhi*lattice_const):
            neighbor_OI=[]
            neighbor_saddle=[]
            TI_saddle = []
            for TI in TIs:
                if cal_distance(OI,TI)<1.8:
                    neighbor_OI.append(TI)
            for TI in neighbor_OI:
                nearest_saddle = []
                for saddle in saddles:
                    if cal_distance(saddle,TI)<0.8:
                        nearest_saddle.append(saddle)
                for saddle in nearest_saddle:
                    if cal_distance(OI,saddle)<cal_distance(OI,TI):
                        neighbor_saddle.append(saddle)
                        TI_saddle.append([OI,TI,saddle])
            OI_TI_saddle.append(TI_saddle)

    TI_OI_saddle = []
    for TI in TIs:
        if In_Boundary(TI,xhi*lattice_const,yhi*lattice_const,zhi*lattice_const):
            neighbor_TI=[]
            neighbor_saddle=[]
            OI_saddle = []
            for saddle in saddles:
                if cal_distance(saddle,TI)<0.8:
```

```
                neighbor_saddle.append(saddle)
        for OI in OIs:
            if cal_distance(OI,TI)<1.8:
                neighbor_TI.append(OI)
        for OI in neighbor_TI:
            for saddle in neighbor_saddle:
                if cal_distance(OI,saddle)<cal_distance(OI,TI):
                    OI_saddle.append([TI,OI,saddle])
        TI_OI_saddle.append(OI_saddle)

neighborlist=[OI_TI_saddle,TI_OI_saddle]
return (neighborlist)

def kMC(neighborlist,sol_energy): #input the neighbor list and the H solution energies
    OI_TI_saddle=neighborlist[0]
    TI_OI_saddle=neighborlist[1]
    #kMC steps start
    kBT = 0.025852
    t = []
    distance = []
    OI_id = random.randint(0,len(OI_TI_saddle)-1)
    positions=[]
    for step in range(100000):
        rates=[]
    ##### OI jump to TI################################################
        for TI_saddle in OI_TI_saddle[OI_id]:
            OI=TI_saddle[0]
            TI=TI_saddle[1]
            saddle=TI_saddle[2]
            barrier = sol_energy[int(saddle[-1])]-sol_energy[int(OI[-1])]
            rate = 10**13*math.exp(-barrier/kBT)
            rates.append(rate)
        sum_rates=sum(rates)
        x=[0]
        accum_rate=0
        for rate in rates:
            accum_rate=accum_rate+rate
            x.append(accum_rate/sum_rates)

        gamma=random.random()
        for i in range(0,len(x)):
            if gamma>x[i] and gamma<x[i+1]:
                next_TI_index=i
        next_TI=OI_TI_saddle[OI_id][next_TI_index][1]
        positions.append(next_TI)
        rho=random.random()
        t.append(-math.log(rho)/sum_rates)
        distance.append(cal_distance(next_TI, OI))

    ##### TI jump to OI################################################
        for i, OI_saddle in enumerate(TI_OI_saddle):
            TI=OI_saddle[0][0]
            if TI[-1]==next_TI[-1]:
                next_TI_ID=i
        rates=[]
        for OI_saddle in TI_OI_saddle[next_TI_ID]:
            TI=OI_saddle[0]
            OI=OI_saddle[1]
            saddle=OI_saddle[2]

            barrier = sol_energy[int(saddle[-1])]-sol_energy[int(TI[-1])]

            rate =10**13*math.exp(-barrier/kBT)

            rates.append(rate)

        sum_rates=sum(rates)

        x=[0]

        accum_rate=0

        for rate in rates:

            accum_rate=accum_rate+rate

            x.append(accum_rate/sum_rates)


        gamma=random.random()

        for i in range(0,len(x)):

            if gamma>x[i] and gamma<x[i+1]:

                next_OI_index=i

        next_OI=TI_OI_saddle[next_TI_ID][next_OI_index][1]

        positions.append(next_OI)

        rho=random.random()

        t.append(-math.log(rho)/sum_rates)

        distance.append(cal_distance(next_OI, TI_OI_saddle[next_TI_ID][0][0]))


        for i, TI_saddle in enumerate(OI_TI_saddle):

            OI=TI_saddle[0][0]

            if OI[-1]==next_OI[-1]:

                next_OI_ID=i

        OI_id = next_OI_ID

positions=np.array(positions)


D=0
```

```
        t_add=0
        add_D=[]
        add_t=[]
        for step in range(len(t)):
            delta_D = distance[step]**2
            D=D+delta_D
            t_add=t_add+t[step]
            add_D.append(D)
            add_t.append(t_add)
        d_D=add_D[-1]-add_D[0]
        d_t=add_t[-1]-add_t[0]
        D_coeff=d_D/(6*d_t)*10e-20      #convert unit to m^2/s
        plt.plot(add_t,add_D)
        return D_coeff
```

Using the two functions, one can calculate the H diffusion coefficient in any HEA system once the H solution energies are known. With the functions for kMC simulations, we can then **perform the WOA to find the optimized HEA structures with low H diffusion coefficients**.

```
    dim=80
    b=1
    whale_num=6
    max_iter=500


    #initialize the locations of whales
    X = []
    for whale in range(whale_num):
        atom_types=[]
        for i in range(dim):
                atom_types.append(random.randint(1,5))
            X.append(atom_types)
        X = np.array(X)


        gBest_coeff = 1
        gBest_X = np.zeros(dim)
        gBest_curve = np.zeros(max_iter)
        nei_list = neighborlist('equiHEA1')
        gBest_solu_ener = []
        t = 0
        random.seed(19)
        while t < max_iter:
            #update best whale and best coefficient
            for i in range(whale_num):
                for ele in range(dim):
                    if X[i,ele]> 5 or X[i,ele] <1:
                        X[i,ele]=random.randint(1,5)
                x=[0,0,0,0,0]
                for atom_type in X[i,:]:
                    x[int(atom_type)-1]+=1
                if x[0]*x[1]*x[2]*x[3]*x[4]==0:
                    atom_types = []
                    for j in range(dim):
                        atom_types.append(random.randint(1,5))
                    X[i,:]=np.array(atom_types)
                solu_ener = solution_energy(X[i,:])        #use the trained ML model to predict H
solution energies


                fitness = kMC(nei_list,solu_ener)
                if round(np.log(fitness),2) <= round(np.log(gBest_coeff),2): #to avoid falling into local minimum
                    gBest_coeff = fitness
                    gBest_X = X[i,:].copy()
                    gBest_solu_ener = solu_ener

            a = 2*(max_iter - t)/max_iter
            #update the whales
            for i in range(whale_num):
                p = np.random.uniform()
                R1 = np.random.uniform()
                R2 = np.random.uniform()
                A = 2*a*R1-a
                C = 2*R2
                l = 2*np.random.uniform()-1

                if p >= 0.5:
                    D = abs(gBest_X - X[i, :])
                    X[i, :] = D*np.exp(b*l)*np.cos(2*np.pi*l)+gBest_X
                else:
                    if abs(A) < 1:
                        D = abs(C*gBest_X - X[i, :])
                        X[i, :] = gBest_X - A*D
                    else:
                        rand_index = np.random.randint(low=0, high=whale_num)
                        X_rand = X[rand_index, :]
                        D = abs(C*X_rand - X[i, :])
                        X[i, :] = X_rand - A*D
```

The chemical compositions of the HEA structures generated during the WOA process and their corresponding H diffusion coefficients were saved as numpy arrays. We then use them to perform regressions to reveal the dependence of H diffusion coefficient on chemical composition. The python codes are as follows:

```
x_arr=np.load(‘compositions.npy') #the atomic ratios of each element in the HEA
y_arr=np.load('diffusion_coefficients.npy') #the corresponding H diffusion coefficients
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_arr, y_arr, test_size=0.2, random_state=42)
from sklearn.linear_model import RidgeCV
from sklearn.metrics import r2_score
nn = RidgeCV()

nn.fit(x_train,y_train)
w=nn.coef_

import matplotlib.pyplot as plt
low    = min(y_arr)-0.1
high = max(y_arr)+0.1
lims = [-16.9,-13]

text = 'Linear regression'
plt. figure(2,figsize=(8,8))
plt.plot(lims,lims,'--',linewidth=2,color='black')
coeff_predict = nn.predict(x_train)
plt.plot(y_train, coeff_predict, '+', color = 'orangered',alpha=0.5)
plt.xlim(lims)
plt.ylim(lims)
plt.text(-16.8,-13.2, text,fontsize=22)
plt.text(-16.8,-13.5,'Training set',fontsize=22)
plt.text(-16.8,-13.8,'$\mathregular{R^2}$                                ='+str(round(r2_score(y_train,
coeff_predict),2)),fontsize=22)
plt.ylabel('Predicted value', fontsize=20)
plt.xlabel('True value', fontsize=20)
a=[-16.5,-16,-15.5,-15,-14.5,-14,-13.5,-13]
plt.xticks(a,fontsize = 18)
plt.yticks(a,fontsize = 18)

plt.figure(3,figsize=(8,8))
plt.plot(lims,lims,'--',linewidth=2,color='black')
coeff_predict = nn.predict(x_test)
plt.plot(y_test, coeff_predict, '+', color = 'deeppink',alpha=0.5)
plt.xlim(lims)
plt.ylim(lims)

plt.text(-16.8,-13.2, text,fontsize=22)
plt.text(-16.8,-13.5,'Test set',fontsize=22)
plt.text(-16.8,-13.8,'$\mathregular{R^2}$                                 ='+str(round(r2_score(y_test,
coeff_predict),2)),fontsize=22)
plt.ylabel('Predicted value', fontsize=20)
plt.xlabel('True value', fontsize=20)
a=[-16.5,-16,-15.5,-15,-14.5,-14,-13.5,-13]
plt.xticks(a,fontsize = 18)
plt.yticks(a,fontsize = 18)

# 2-degree polynomial regression
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(interaction_only=True,degree=2)
poly.fit(x_arr)
x_arr = poly.transform(x_arr)

x_train, x_test, y_train, y_test = train_test_split(x_arr, y_arr, test_size=0.2, random_state=42)
nn = RidgeCV()
nn.fit(x_train,y_train)
w=nn.coef_

import matplotlib.pyplot as plt
low    = min(y_arr)-0.1
high = max(y_arr)+0.1
lims = [-16.9,-13]

text = 'Degree-2 polynomial regression'
plt.figure(2,figsize=(8,8))
plt.plot(lims,lims,'--',linewidth=2,color='black')
coeff_predict = nn.predict(x_train)
plt.plot(y_train, coeff_predict, '+', color = 'royalblue',alpha=0.5)
plt.xlim(lims)
plt.ylim(lims)

plt.text(-16.8,-13.2, text,fontsize=22)
plt.text(-16.8,-13.5,'Training set',fontsize=22)
plt.text(-16.8,-13.8,'$\mathregular{R^2}$                              ='+str(round(r2_score(y_train,
coeff_predict),2)),fontsize=22)
plt.ylabel('Predicted value', fontsize=20)
plt.xlabel('True value', fontsize=20)
a=[-16.5,-16,-15.5,-15,-14.5,-14,-13.5,-13]
```

```
plt.xticks(a,fontsize = 18)

plt.yticks(a,fontsize = 18)


plt. figure(3, figsize=(8,8))

plt.plot(lims,lims,'--',linewidth=2,color='black')

coeff_predict = nn.predict(x_test)

plt.plot(y_test, coeff_predict, '+', color = 'darkviolet',alpha=0.5)

plt.xlim(lims)

plt.ylim(lims)


plt.text(-16.8,-13.2, text,fontsize=22)

plt.text(-16.8,-13.5,'Test set',fontsize=22)

plt.text(-16.8,-13.8,'$\mathregular{R^2}$='+str(round(r2_score(y_test,

coeff_predict),2)),fontsize=22)

plt.ylabel('Predicted value', fontsize=20)

plt.xlabel('True value', fontsize=20)

a=[-16.5,-16,-15.5,-15,-14.5,-14,-13.5,-13]

plt.xticks(a,fontsize = 18)

plt.yticks(a,fontsize = 18)
```