Master's thesis
Theoretical Physics

Artificial Neural Networks for Parametrisation of a Kinetic Monte Carlo Model of Surface Diffusion

Jonna Romppainen

November 23, 2020

Supervisors: Prof. Flyura Djurabekova
MSc Jyri Kimari

Examiners: Prof. Flyura Djurabekova
Prof. Kai Nordlund

University of Helsinki
Faculty of Science

PL 64 (Gustaf Hällströmin katu 2a)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Theoretical Physics

Tekijä — Författare — Author

Jonna Romppainen

Työn nimi — Arbetets titel — Title

Artificial Neural Networks for Parametrisation of a Kinetic Monte Carlo Model of Surface Diffusion

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	November 23, 2020	70

Tiivistelmä — Referat — Abstract

Surface diffusion in metals can be simulated with the atomistic kinetic Monte Carlo (KMC) method, where the evolution of a system is modeled by successive atomic jumps. The parametrisation of the method requires calculating the energy barriers of the different jumps that can occur in the system, which poses a limitation to its use. A promising solution to this are machine learning methods, such as artificial neural networks, which can be trained to predict barriers based on a set of pre-calculated ones. In this work, an existing neural network based parametrisation scheme is enhanced by expanding the atomic environment of the jump to include more atoms.

A set of surface diffusion jumps was selected and their barriers were calculated with the nudged elastic band method. Artificial neural networks were then trained on the calculated barriers. Finally, KMC simulations of nanotip flattening were run using barriers which were predicted by the neural networks. The simulations were compared to the KMC results obtained with the existing scheme.

The additional atoms in the jump environment caused significant changes to the barriers, which cannot be described by the existing model. The trained networks also showed a good prediction accuracy. However, the KMC results were in some cases more realistic or as realistic as the previous results, but often worse. The quality of the results also depended strongly on the selection of training barriers. We suggest that, for example, active learning methods can be used in the future to select the training data optimally.

Avainsanat — Nyckelord — Keywords

Surface diffusion, kinetic Monte Carlo, machine learning, artificial neural network, energy barrier

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Matemaattis-luonnontieteellinen tiedekunta	Teoreettinen fysiikka

Tekijä — Författare — Author

Jonna Romppainen

Työn nimi — Arbetets titel — Title

Artificial Neural Networks for Parametrisation of a Kinetic Monte Carlo Model of Surface Diffusion

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Pro gradu	23.11.2020	70

Tiivistelmä — Referat — Abstract

Diffuusiota metallipinnalla voidaan simuloida atomistisella kineettisellä Monte Carlo -menetelmällä (KMC), jossa systeemin aikakehitystä mallinnetaan atomien perättäisillä hypyillä hilassa. Mallin parametrisoimiseksi on laskettava erilaisten mahdollisten hyppyjen energiavallit, mikä rajoittaa menetelmän käyttöä. Lupaava ratkaisu tähän ovat koneoppimista hyödyntävät menetelmät, kuten keinotekoiset neuroverkot. Neuroverkkoja voidaan kouluttaa joukolla tunnettuja energiavalleja, minkä jälkeen ne osaavat arvioida uusien hyppyjen energiavallit. Tässä tutkimuksessa parannetaan aiempaa neuroverkkoihin perustuvaa parametrisointimenetelmää sisällyttämällä malliin useampia atomeja hypyn ympärillä.

Joukko hyppyjä valittiin, ja niiden energiavallit laskettiin nudged elastic band -menetelmällä. Näitä energiavalleja käytettiin neuroverkkojen kouluttamiseen. Lopuksi simuloitiin nanokokoisten pylväiden madaltumista KMC-menetelmällä käyttäen neuroverkoilla laskettuja energiavalleja. Simulaatioita verrattiin aiempaa parametrisointimenetelmää käyttäen saatuihin KMC-tuloksiin.

Malliin sisällytetyt uudet atomit aiheuttivat energiavalleihin merkittäviä eroavaisuuksia, joita ei pystytä kuvaamaan aiemmalla mallilla. Lisäksi koulutetut neuroverkot tuottivat tarkkuudeltaan hyviä arvioita energiavalleista. KMC-tulokset olivat joissain tapauksissa realistisempia tai yhtä realistisia kuin aiemmat tulokset, mutta monissa tapauksissa huonompia. Tulosten laatu riippui vahvasti siitä, mitä energiavalleja neuroverkkojen koulutuksessa oli käytetty. Tulevaisuudessa koulutuksessa käytettävien energiavallien valintaa voidaan parantaa esimerkiksi aktiivisen oppimisen menetelmiä käyttäen.

Avainsanat — Nyckelord — Keywords

pintadiffuusio, kineettinen Monte Carlo, koneoppiminen, keinotekoinen neuroverkko, energiavalli

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

Interest in artificial intelligence applications has grown in recent years in the field of materials research. In particular machine learning methods, which can improve their performance when they encounter new data, have gained popularity in computational materials science [1]. For example, artificial neural networks have been used to predict energy barriers of vacancy migration in metals [2], [3], [4], [5], [6], [7]. The predicted barriers were then used to simulate diffusion with the stochastic simulation method called Kinetic Monte Carlo (KMC) [8], [9]. In these previous studies, the main subject of interest has been bulk diffusion in Fe and Fe based alloys. In this work, neural networks are also used for barrier prediction in KMC, but instead of bulk Fe, the system is a Cu surface.

The choice of Cu surface as the system of interest comes from its importance to the planned Compact Linear Collider (CLIC) particle accelerator at CERN. Electric breakdowns, or sparks, on the surfaces of the Cu accelerating structures hinder the operation of the accelerator by deforming the beam [10]. It is suspected that small protrusions grow on the surfaces during use and initiate the breakdowns [11]. Surface diffusion is expected to contribute to the growth of the protrusions under the electric field [12], and thus having a good computational model of surface diffusion can shed light on the origin of the breakdowns.

To simulate diffusion with KMC, the migration activation energies, also called energy barriers, need to be known for each event that is to be simulated. If the system is complex, there can be far too many possible atomic jumps to calculate all of their energy barriers. With machine learning methods like artificial neural networks, it is possible to calculate only a portion of the barriers, have the network learn from them to predict new barriers, and then use the predictions in KMC. This has been done in the previous works mentioned above, as well as in the work of Kimari *et al.* [13]. They described the local environment of an atomic jump in terms of 26 neighbouring atoms, namely the first and second nearest neighbours of the jumping atom and the vacancy site. The trained networks were then used in several different KMC simulations, including nanotip flattening on {100}, {110}, and {111} Cu surfaces.

A question remains whether the 26 neighbouring atoms describe the atomic environment precisely enough, or whether expanding the jump description to include more neighbouring atoms would have a significant effect on the barriers. Furthermore, if more

atoms are included, the number of possible atomic environments increases. This means that more barriers may need to be calculated for the training of the neural networks, which increases the computational cost of the method.

In this work, we expand the descriptor to include also the third and fourth nearest neighbours of the jump, which is 72 atoms in total. A portion of the barriers is calculated with the nudged elastic band method (NEB) [14], and the effect of changing the configuration of the third and fourth nearest neighbours is studied. Artificial neural networks are also trained with the data, and their prediction accuracy and the computational cost of training are addressed. Finally, KMC simulations of nanotip flattening are carried out using the trained neural networks. The behaviour of the surface atoms is compared to the KMC results by Kimari*et al.* [13]. In the future, the model can be complemented with electric field effects.

Artificial intelligence and its use in materials research is discussed in section 2. In particular, different ways to use machine learning are presented and two computational methods are described. In section 3, it is shown why understanding surface diffusion in metals is important for the development of contemporary technological applications. The section also describes surface diffusion as a physical process and the KMC simulation method, which is often used to simulate diffusion. The rest of the section deals with finding the activation energies for the KMC simulation and previous work in using artificial neural networks for this purpose. Section 4 describes the use of artificial neural networks in this work, the expanded 72D local atomic environment descriptor, and the generation of the training data. The differences in the calculated activation energies due to the expansion of the descriptor are covered in section 5, whereas section 6 deals with the results of training the artificial neural networks. Finally, section 7 shows the results of the KMC simulations, where neural networks were used to obtain the barriers. Conclusions on the results are presented in section 8.

2 Artificial intelligence in materials research applications

Artificial intelligence (AI) is used nowadays for diverse tasks in a wide range of different fields. Modern powerful computers and the wide availability of data have enabled the development and use of new, effective methods to solve problems that have traditionally been troublesome for machines [15]. These problems include tasks like image classification [16] and speech recognition [17], which are simple for a human to perform, but difficult to describe with a computer program. AI methods are also effective when analysing large amounts of data is required, for instance to detect fraud in electronic commerce systems [18]. In many tasks AI can already perform as well as, or even better than humans. When AI was used to identify skin cancer from pictures, it outperformed the average of the human dermatologists in the study [19]. In addition, an AI called AlphaGo has famously won the world champion in the game of Go, which was previously considered too complex a game for computers to master [20], [21].

Although AI is widely used, finding a general definition for it is not so simple. Instead, good definitions seem to be dependent on the context. There are different ways to define what artificial intelligence is, probably because the question "What is artificial intelligence?" immediately brings us to philosophical questions like "What is intelligence?". One practical approach to defining AI is the famous Turing test: if a human poses questions to a computer and cannot tell whether the responses come from another human or a computer, the computer should be considered intelligent [22]. This definition seems suited for a more general AI, which is designed to interact with humans in a human-like manner, but in materials science the definition is not that useful. An AI in materials science would be expected to take in data and do something useful with it, but whether it seems like a human while doing so ought to be unimportant. AI can also be defined in terms of reasoning or behaviour of the system, and in terms of rationality or human-likeness [23]. In materials science the reasoning or thought processes of the intelligent system are not as important as the results that the system produces. Moreover, although AI can be used to do what has traditionally been done by humans, the scientist would probably prefer a system that gives perfectly rational results to a more human-like but less rational system. In materials science AI should therefore be defined in terms of showing rational behaviour.

Many early AIs used a collection of rules that were programmed into the system to make rational decisions. To succeed in more complex tasks, the AI would naturally need a large number of rules. Unfortunately, programming all the rules into the system was found to be unfeasible. Later, however, an idea developed that the system could be programmed to gather the necessary information on its own, rather than program all the information into the system. This is how machine learning eventually became mainstream in the AI community [24]. Nowadays, machine learning is considered a subfield of artificial intelligence. It is characteristic for machine learning methods that they can improve their performance through training [25]. Large amounts of of data that could be used for training already exist in the field of materials research, which makes it a promising field for the application of machine learning [1]. The use of machine learning in materials research is treated in section 2.1.

2.1 Machine learning approaches for solving materials research problems

Machine learning methods often require or benefit from a large amount of data to train on. Fortunately, large volumes of computational and experimental data on the properties of materials already exist in different databases in the field of materials science [1]. Just a few examples of these are the Crystallography Open Database [26] and MatNavi [27], which contain experimental data on crystal structures and material properties respectively, AFLOW [28], that holds *ab initio* calculated data on the structures and properties of inorganic materials, and the NREL Materials Database [29], which hosts computational structures and properties of materials, with a focus on materials used in renewable energy applications. It is suggested that this data could be harnessed with machine learning methods to speed up heavy calculations in computational materials science [1]. For example, density functional theory (DFT) [30], [31] is a popular but computationally rather expensive method for *ab initio* electronic structure calculations. However, it is possible to skip the heaviest calculations in the method with a model that directly learns mappings between the electron density and the total energy or nuclear potential of the system [32]. Alternatively, the results of DFT calculations can be used to train a machine learning potential energy function for molecular dynamics (MD) [33] simulations. In MD, the

function is then used to describe interatomic interactions in the simulated system. This way dynamic simulations can be done much faster than by calculating the interactions with pure DFT, while retaining an accuracy comparable to DFT [34]. It should be noted, however, that in both of these cases a specific training dataset of DFT calculations had to be built instead of using the data in the databases. Therefore it seems that there is still work to be done before *ab initio* data from existing databases can be widely used to train machine learning models.

The work of Bartók *et al.* is a recent example of utilizing machine learning to find a potential energy function for MD simulations [35]. They first calculated the potential energy surfaces of different configurations of Si atoms with DFT, and trained a Gaussian approximation potential (GAP) on the data to interpolate the energy surfaces. GAPs are machine-learning potentials that use Gaussian process regression to make predictions of new data points based on training data. In the study, the material properties reproduced by the GAP were tested against DFT calculations, both for structures included in the dataset and for some completely different configurations. The researchers showed that compared to DFT calculations, the potential reproduced many of the properties of Si in the crystal and liquid as well as in the amorphous phase. The theory of Gaussian processes is described in more detail in section 2.1.2.

Another way to use machine learning in materials science is to support experimental research by predicting the properties of new materials, like the glass transition temperature [36]. Alternatively, it is also possible to first decide on a property and then search for materials that have that property. Machine learning has been used for example to find materials that have a certain crystal structure [37], magnetic properties or optoelectronic properties [38]. A more general application that would rely heavily on machine learning would be so called accelerated discovery of materials, which utilizes an autonomous laboratory to plan and carry out experiments without human interaction [39].

In the line of materials design, Cassar *et al.* used machine learning to predict the glass transition temperature of multicomponent oxide glasses [36]. Their purpose was to produce estimates of the transition temperatures that could serve as a guide to picking new compositions to try out in experiments. They collected the training data, which consisted of chemical compositions and corresponding glass transition temperatures, from an existing glass property database and used it to train an artificial neural network. The network

managed to predict the transition temperatures with a maximum error of 9 % for 90 % of the data, which is of the same order as the scattering of the transition temperatures in the training data. What is more, the error of the predictions was below 9 % also for 15 tested glasses that were not in the original dataset, which is a promising result. Artificial neural networks are described in more detail in section 2.1.1.

2.1.1 Artificial neural networks

Artificial neural networks (ANN) are algorithms that loosely mimic the functioning of a brain to find solutions to complex problems. Originally the networks were aimed to model neurons in the brain [40], but nowadays it is considered more important that the network just finds good solutions. Using training algorithms, the network can learn to find patterns in the input data without having been explicitly programmed to search for a certain kind of pattern. The patterns are encoded to the nodes of the network, and can then be used to approximate continuous functions or to classify the input data in different categories [41].

An artificial neural network consists of layers of nodes: the input layer, output layer, and a number of hidden layers between them. The nodes are connected to each other so that the nodes of the previous layer contribute to the value of the nodes in the next layer with some weight. The output of node l can be expressed as a simple formula

$$y = f\left(\sum_{j=0}^{J} w_{lj} z_j\right) \tag{1}$$

where z_j are the outputs of previous J nodes and $w_l j$ are the weights associated with each previous node. The node thus calculates the weighted sum of the outputs of the previous nodes. The sum is then passed through the activation function f and sent to the nodes in the next layer. The output of the network can be adjusted by changing the weights [42].

The choice of the activation function affects the operation of the network. First of all, the function should be non-linear, because if a linear function is used, the output will be just a linear combination of the inputs. This way, the network would not be able to learn more complex patterns in the data, and the same result could also be computed by a network without any hidden nodes. In addition, the activation function has an effect on the behaviour of the network during training, since the derivative of the activation

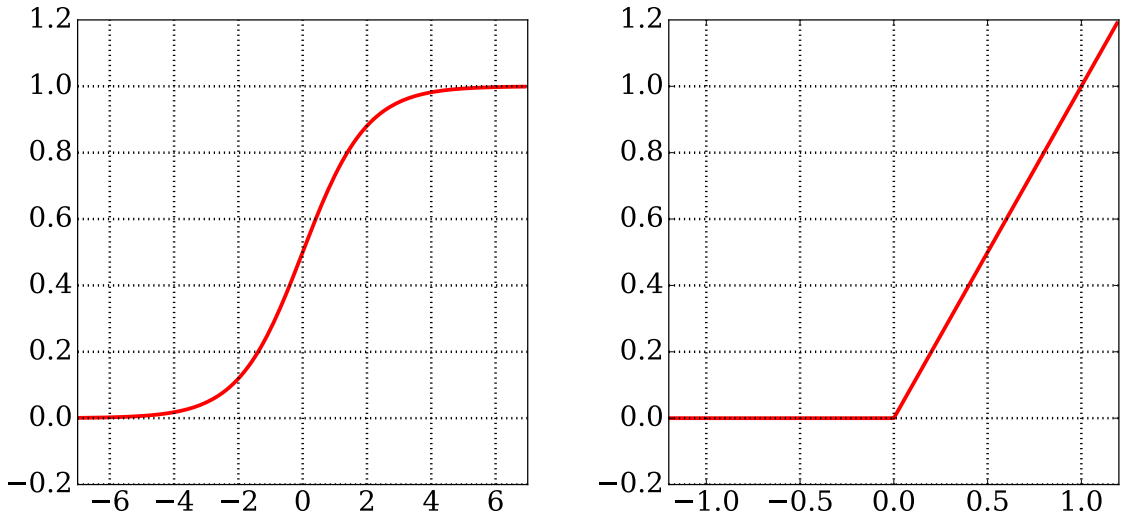
function is needed for the training algorithm. Traditionally, sigmoid activation functions in the shape of

$$f(x) = (1 + \exp(-x))^{-1} \tag{2}$$

have been widely used [43]. A plot of the function can be seen in figure 1a. The derivative of the function is small near the end points and larger in the middle, so that weights that are further away from the middle are adjusted less. As a result, the weights can quickly achieve large values during training. Since the derivative is small for large weights, it thus becomes slow to adjust them further, even if they turn out to be non-optimal [41]. This kind of saturation behaviour is especially visible in neural networks with several hidden layers [44]. For this reason, new activation functions have been developed, and for example the rectified linear unit (ReLU) activation function is popular nowadays [45]. The ReLU [46] is expressed as

$$f(x) = \max(0, x) \tag{3}$$

A plot of the function can be seen in figure 1b. It is a piecewise function, so it fulfills the nonlinearity criterion. Since the derivative does not go to zero in the higher end point, even large weights can be adjusted quickly. Training the network and the different training algorithms are treated in more detail below.



(a) Sigmoid

(b) Rectified linear unit

Figure 1: Figures of the sigmoid activation function a) and the rectified linear unit activation function (ReLU) b)

There are many ways in which the neural network can learn. In unsupervised learning, the network is given some input data, and it then attempts to optimize a given criterion function. In reinforcement learning, the network gets reinforcement signals from its environment. The reinforcement signal tells the network whether its output was favourable or not, and the network then attempts to maximize the positive reinforcement signal. In supervised learning, which is used in this work, the network is given input data and corresponding known output data in pairs. The network is then trained so that its outputs match the known, correct outputs [41].

For the training the weights must first be initialized, usually with small random numbers [41]. In supervised training the network then goes through the given dataset of known input-output pairs and updates the weights using an error backpropagation algorithm. The algorithm minimizes the error between the known output and the network output by gradient descent [43]. The original backpropagation algorithm was discovered independently by different groups, but it was made popular by Rumelhart *et al.* [47]. Variations of the algorithm, such as RPROP [48] and iRPROP [49] have been developed, and iRPROP is used for training in this work.

It is easiest to first show how the procedure of updating the weights using the traditional backpropagation algorithm is done on the output nodes, and then modify it a little for use on the hidden nodes. This presentation of the algorithm follows the work of Hassoun in reference [41]. The error can be measured for example with the sum of squared errors function. For a network that has L nodes in the output layer, the function is

$$E(\mathbf{w}) = \frac{1}{2}\sum_{l=1}^{L}(d_l - y_l)^2 \tag{4}$$

where $\mathbf{w}$ is the collection of all the weights in the network, d_l and y_l are respectively the desired output value and the computed output value of a single output node, and the sum is taken over all the output nodes. The change of one weight in the output node can be expressed as a partial derivative of the squared error:

$$\Delta w_{lj} = -\rho\frac{\partial E_l}{\partial w_{lj}} = -\rho\frac{\partial}{\partial w_{lj}}\frac{1}{2}(d_l - y_l)^2 \tag{5}$$

where E_l is the squared error in the output node l and w_{lj} is a weight associated with hidden node j in output node l. The constant ρ is called the learning rate, and it is a hyperparameter of the algorithm. It controls the convergence speed of the network. A

large ρ gives fast convergence but results in oscillations near the minimum error, whereas a small rate avoids oscillations but results in slow converge. Now, to calculate the derivative, we first reformulate equation 1 and then insert it into equation 5. The new form of equation 1 is

$$y = f\left(\sum_{j=0}^{J} w_{lj} z_j\right) = f\left(\text{net}_l\right) \tag{6}$$

where the weighted sum is written simply as net_l. Inserting this into equation 5 gives

$$\Delta w_{lj} = -\rho \frac{\partial}{\partial w_{lj}} \frac{1}{2} \left[d_l - f\left(\text{net}_l\right)\right]^2 \tag{7}$$

We can now see that Δw_{lj} is a function of f, which is in turn a function of net_l. The derivative can be calculated by using the chain rule, which gives the result

$$\Delta w_{lj} = \rho \left(d_l - y_l\right) f'\left(\text{net}_l\right) z_j \tag{8}$$

where $f'\left(\text{net}_l\right)$ is the derivative of the activation function with respect to net_l. This gives us the formula for updating the weights in the output layer. The formula also shows that the derivative of the activation function is needed for the calculation. This explains why the choice of activation function affects the learning process, as stated above.

For the hidden nodes the formula is quite similar, except that there are no desired outputs defined for the hidden nodes. Therefore, the weights must be changed so that the error in the output nodes is minimized. The error in the output layer thus propagates backwards in the network, hence the name of the algorithm. The change of the weight in hidden node j is written as

$$\Delta w_{ji} = -\rho \frac{\partial E}{\partial w_{ji}} = -\rho \frac{\partial}{\partial w_{ji}} \frac{1}{2} \sum_{l=1}^{L} \left[d_l - f\left(\text{net}_l\right)\right]^2 \tag{9}$$

The difference to equation 7 is that the derivative is now calculated with respect to the weight in the hidden node, w_{ji}. The formula for changing the weights in the hidden nodes can be derived by using the chain rule of the derivative. A straightforward, although somewhat lengthy differentiation gives us

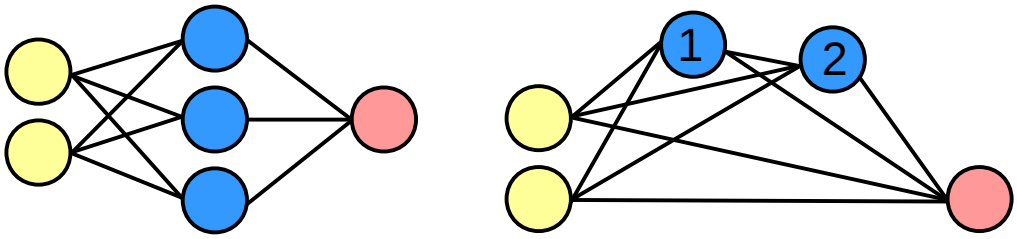
$$\Delta w_{ji} = \rho \left[\sum_{l=1}^{L} (d_l - y_l) f'(\text{net}_l) w_{lj}\right] f'\left(\text{net}_j\right) x_i \tag{10}$$

Here $f'\left(\text{net}_j\right)$ is the derivative of the activation function with respect to net_j, and net_j is the weighted sum computed in hidden node j:

$$\text{net}_\text{j} = \sum_{i=0}^{n} w_{ji} x_i \tag{11}$$

Similarly to equation 6, x_i are the inputs passed to the hidden node j from the input node i. Naturally, the formula can also be generalized to more than one hidden layer. Updating the weights repeatedly with this formula drives the error towards zero. The training can be stopped when the error is low enough or when some other condition is reached, for example a sufficient number of weight updates have been carried out [41].

The network can have a fixed number of layers with a fixed number of nodes, in which case it is called a multilayer perceptron [43]. Another option is to use a cascade correlation neural network, where the number of nodes is determined during training. At first, all the input nodes are connected directly to all the output nodes. This very simple network is then trained until the error does not decrease anymore. A pool of new nodes is then trained, and the node which gives the best result is added to the network. The new node is connected to the input nodes and the output nodes. More hidden nodes can be added in a similar fashion so that the new node is connected to all the previous nodes [50]. Schematic pictures of the multilayer perceptron and the cascade correlation network can be seen in figure 2.



(a) Multilayer perceptron (b) Cascade correlation network

Figure 2: Simple schematics of the architectures of two types of artificial neural networks. A multilayer perceptron is shown in a) and a cascade correlation network in b). The input nodes are shown in yellow, the output nodes in red, and the hidden nodes in blue. The lines show the connections between nodes. The number of hidden nodes in the multilayer perceptron is fixed before the training starts. In a cascade network, hidden nodes are added during the training and connected to all previously existing nodes. In the figure, the hidden nodes of the cascade network are numbered in the order in which they were added.

It has been proven mathematically that neural networks with a single hidden layer and an activation function that fulfills some simple criteria are universal approxima-

tors [51], [52] and classifiers [51]. This means that any continuous multivariate function can be approximated with this kind of neural network to an arbitrary accuracy, using a finite number of hidden nodes. A universal classifier, then, can similarly divide the data to an arbitrary number of disjoint classes. In turn, cascade correlation networks have been shown to be able to approximate any finite single-hidden-layer multilayer perceptron [53], so they too are universal approximators and classifiers. However, it is a different question whether a network can be trained so that the desired accuracy of approximation or classification is reached. The algorithm that minimizes the error of the network may fall to a local minimum, the amount of input data may be insufficient, or the required number of nodes may be so large that the training becomes computationally infeasible. Nevertheless, the proofs mentioned here give a solid mathematical foundation for using artificial neural networks as approximators and classifiers.

2.1.2 Gaussian processes

Gaussian processes are a Bayesian machine learning method that can be used for function approximation and classification problems. The method is exact, so that it will always give the global minimum solution within the training data. Furthermore, the predictions of Gaussian processes always come with uncertainty information, since the prediction is itself a distribution. Gaussian processes are also a non-parametric model. As opposed to parametric models like linear regression, which use a chosen number of parameters to describe the dependencies in the data, non-parametric models are not restricted to a set of parameters. For every Gaussian process, there is a corresponding parametric representation in terms of basis vectors, but in many cases the basis vectors would be infinite dimensional. This gives Gaussian processes an advantage, since using an infinite dimensional model in computations is inefficient [54]. Finally, because known input-output pairs are required to make predictions, Gaussian processes are counted among supervised learning methods [55].

According to Rasmussen and Williams [55], a Gaussian process is defined as "a collection of random variables, any finite number of which have a joint Gaussian distribution". It describes a distribution over functions. Whereas a Gaussian distribution is completely specified by its mean and variance, a Gaussian process is specified by its mean function

$m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, which is also called a kernel. They can be defined as

$$m(\mathbf{x}) = E[f(\mathbf{x})] \tag{12}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = E[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))] \tag{13}$$

where $\mathbf{x}$ is an input vector, $f(\mathbf{x})$ is a real Gaussian process, and $E[f(\mathbf{x})]$ is the expectation value of f with input $\mathbf{x}$ [55]. In practice, it is usually possible to process the data before applying the model in such a way that the trends in the mean are removed, and the mean function can be considered to be zero. Thus, the model depends only on the choice of the covariance function. The covariance function defines a measure of similarity between points, and because of this, it also determines what kind of functions will be included in the distribution of functions. If the inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to each other, then it is expected that the corresponding outputs $\mathbf{y}_i$ and $\mathbf{y}_j$ are also close, in other words they are expected to be correlated. The higher the correlation between nearby outputs is, the smoother are the functions that the kernel generates. In addition, the correlation is generally expected to decrease for points that are farther away from each other, but for periodic functions the correlation of far away points can also be high. As a result, some kernels will give a better fit to the data than others [54]. If something about the data is known beforehand, it should thus be possible to use the information to pick a good covariance function for the problem.

The kernel is used to define a covariance matrix $\mathbf{K}$, whose elements are computed from the training inputs as follows:

$$[\mathbf{K}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) \tag{14}$$

This then gives us a Gaussian distribution of functions

$$p(\mathbf{y}|X) = \mathcal{N}(\mathbf{y}|\mathbf{m}, \mathbf{K}) \tag{15}$$

Here $p(\mathbf{y}|X)$ denotes the probability of the outputs $\mathbf{y}$ when the inputs X are known. X contains all the known input vectors $\mathbf{x}$. $\mathcal{N}(\mathbf{y}|\mathbf{m}, \mathbf{K})$ means that the outputs $\mathbf{y}$ are normally distributed, and the distribution is described by the means $\mathbf{m}$ at different input values $\mathbf{x}$ and the covariance matrix $\mathbf{K}$. In Bayesian terms, this distribution of functions is called a prior [54]. However, although training data was used to construct the prior, the model has not been trained yet. In fact, the training happens only, when the model is used to make predictions based on the training data, as we will see below.

To make predictions, the prior must be trained or conditioned on the known training data. This is done by computing the joint probability of the predictions on condition that the training outputs have been observed. If the training data is noiseless, the joint probability is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(X,X) & \mathbf{K}(X,X_*) \\ \mathbf{K}(X_*,X) & \mathbf{K}(X_*,X_*) \end{bmatrix}\right) \tag{16}$$

where X are the training inputs with known outputs $\mathbf{y}$ and X_* are the inputs for which we want to predict the outputs $\mathbf{y}^*$. If noise is present, a noise term must be added to the $\mathbf{K}(X,X)$ term, but for the sake of simplicity this case is not treated in more detail in this work. For simplicity it is also assumed that the mean trends have been eliminated from the data, and so the mean is set to zero. The mathematical process of conditioning the joint probability is described in reference [55]. The conditioning effectively removes the functions that do not agree with the training data from the distribution. The remaining distribution can then be written as

$$p(\mathbf{y}^*|X_*,X,\mathbf{y}) \sim \mathcal{N}(\mathbf{y}\mathbf{K}(X_*,X)\mathbf{K}(X,X)^{-1}, \mathbf{K}(X_*,X_*) - \mathbf{K}(X_*,X)\mathbf{K}(X,X)^{-1}\mathbf{K}(X,X_*)) \tag{17}$$

This distribution is called the posterior distribution. Its mean is $\mathbf{y}\mathbf{K}(X_*,X)\mathbf{K}(X,X)^{-1}$ and its covariance $\mathbf{K}(X_*,X_*) - \mathbf{K}(X_*,X)\mathbf{K}(X,X)^{-1}\mathbf{K}(X,X_*)$, both of which can be evaluated. The predicted outputs can then be sampled by generating Gaussian samples from this distribution [55]. A posterior distribution of an example Gaussian process trained with four data points can be seen in figure 3. In the figure, we see the predictions of the model as a gray area. For each input $\mathbf{x}$, the prediction is a Gaussian distribution with a mean and standard deviation. The red curve shows the mean and the shaded area shows where the outputs are expected to lie with a 95 % confidence. It can be seen that the uncertainty of the prediction is smaller near the data points. We can also see that the prediction does not follow the true function when $x >> 1$, which is outside the range of the training data.

The advantages of Gaussian process regression are its exactness and the built-in uncertainty information in the prediction. However, the method is computationally expensive. Because of the matrix inversion in equation 17, the cost of making predictions scales as $O(N^3)$ with the number of the training data points [54]

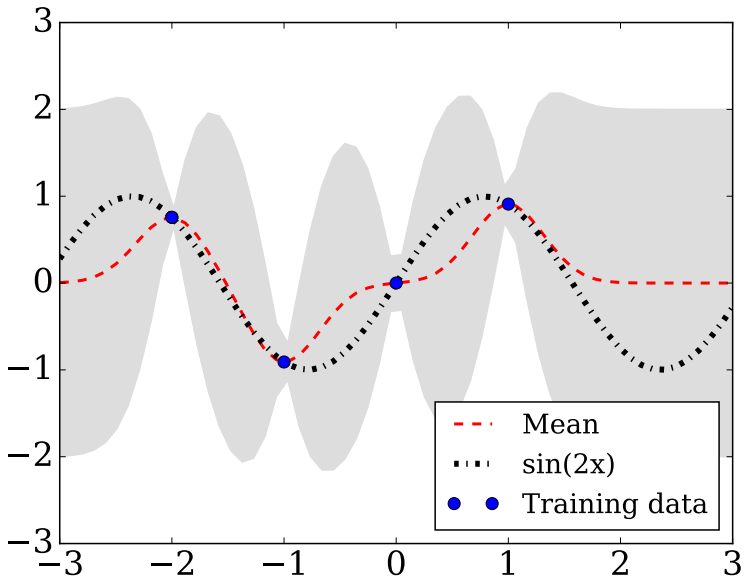


Figure 3: A posterior distribution of a Gaussian process. The blue squares mark the noise-free training data points, and the shaded area shows the 95 % confidence region for the predicted outputs. The red curve is the mean of the conditioned distribution in equation 17 and the black curve is the true function $\sin(2x)$.

3 Modification of surface topology in high electric field applications

In section 2 we saw how artificial intelligence methods have opened possibilities for new technologies and applications in diverse fields. In this work, we aim to contribute to the technologies where surface diffusion plays a significant role by applying machine learning to computer simulations of surface diffusion.

One of the applications, where the understanding of surface diffusion processes is important, is the planned particle accelerator called Compact Linear Collider (CLIC) at CERN. The new accelerator would be used to complement the results obtained at the Large Hadron Collider (LHC) and also to give completely new insights to particle physics [56]. However, the operation of CLIC is limited by vacuum breakdowns happening in the accelerating structures. The cause of the breakdowns is not fully known, but measurements of field emission current show that the electric field is locally enhanced on the surface of the electrode [11]. A surface asperity, such as a sharp nanosized tip, is a possible source of the field enhancement, and it is expected that surface diffusion contributes to the formation of the tip [12]. It is important to be able to model the formation of such a tip to better understand the onset of the vacuum breakdowns.

Sharp nanosized tips can also be manufactured on purpose to use them as field emitters, that emit electrons under a high electric field. Field emitters are used for example in electron microscopes [57], and they can have applications in emerging flat display technologies [58] and novel high frequency electronics [59]. The properties of the emitted electron beam depend on the shape of the tip [60]. The shape of the tip, on the other hand, can change through the diffusion of the surface atoms under electric field [61]. This was also seen in an experiment, where a sharp tip was created using a static electric field combined with laser pulsing [62]. Understanding surface diffusion in the presence of an external field can help to control the shape of the manufactured tip and tailor properties of the emitted electron beam. CLIC and field emitters are treated in more detail in sections 3.1 and 3.2 respectively.

Surface diffusion in metals is a stochastic process, where the atoms jump to nearby vacant lattice sites at random. For this reason, it is natural to study the diffusion processes with stochastic simulation methods like Kinetic Monte Carlo (KMC) [8], [9]. However, a

large number of parameters from either experiments or some other simulations are needed as inputs for a KMC simulation. To obtain the parameters more efficiently, machine learning can be used to predict them according to a previously learned model. The theory of surface diffusion will be presented in section 3.3. The KMC simulation method is described in section 3.4 and finding the parameters in section 3.5.

3.1 Vacuum arcing in CLIC accelerating structures

The Compact Linear Collider (CLIC) is a planned new particle accelerator for accelerating electrons and positrons at CERN. It would be built in three stages, so that in the first stage it would reach 380 GeV collision energies, 1.5 TeV in the second, and 3.0 TeV in the final stage [56]. This enables new research on the Higgs boson [63] and top quarks, and possibly on physics beyond the Standard Model [64]. CLIC utilizes a two-beam acceleration technique, where a drive beam is decelerated to accelerate the main beam. As the drive beam decelerates, its energy is extracted to an electromagnetic field that oscillates at radio frequency (RF) [10]. CLIC is planned to reach an accelerating gradient of 100 MV/m [56].

Due to the strong electric fields, the CLIC accelerating structures suffer from vacuum arcing during the RF pulses. This means that during the pulse an electric breakdown, or in other words a spark, can occur in the accelerating structure. The breakdown can deform the particle beam pulse, and so the desired collision energy may not be reached. To make sure that the accelerator works properly, breakdowns should not occur more often that for 1 % of the beam pulses [10].

The breakdown rate depends on the accelerating gradient, the RF pulse length, and on the shape and material of the accelerating structures [10]. To control the breakdown rate, it is important to understand how and why they occur. The first guess for the cause of the breakdowns could be that the surface of the cathode is not clean, or that the manufacturing of the electrodes leaves roughness on the surface. This, however, is not the full picture. The electrodes show conditioning behaviour, which means that they are at first more sensitive to breakdowns, but after some use they become more resistant. This was first thought to be because the breakdowns remove the impurities and initial protrusions on the surface, which act as breakdown sites [11]. Later, however, it was found that the conditioning progresses with the number of RF pulses rather than

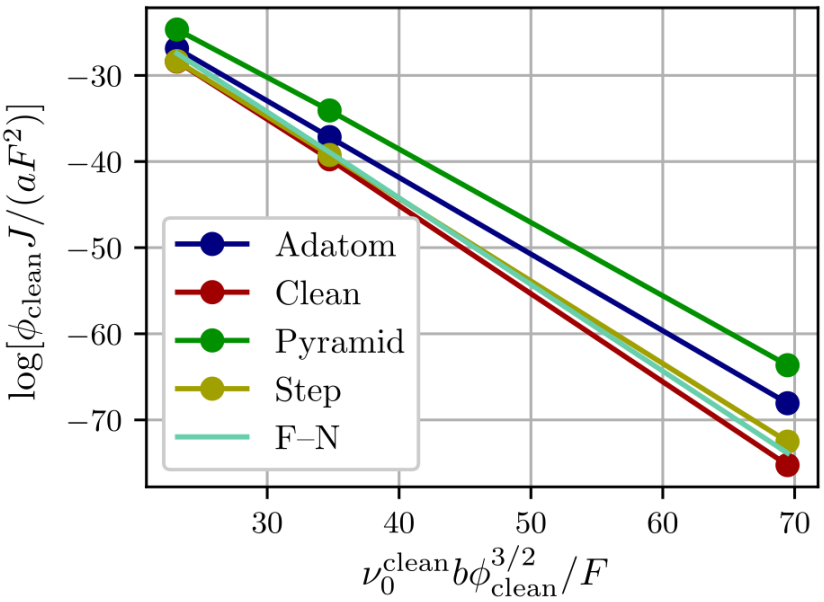


Figure 4: An example of a Fowler-Nordheim plot with computational data. The plot shows the field enhancement factor β as the reciprocal of the slope of the line. Figure[1] by Toijala *et al.* [65].

with the number of breakdowns [66]. Therefore, impurities and initial roughness alone cannot explain the conditioning. The mechanisms that cause the breakdowns and the conditioning behaviour are still actively researched.

The onset of breakdowns has been studied under DC voltage instead of a RF field by Descoeudres *et al.* [11]. They measured the field emission current, that is, the emission of electrons from the cathode under an electric field, as well as the field strength at breakdown and the breakdown rate. They found that the electric field must be locally enhanced on the cathode to produce the measured currents. The local field emission current density J can be calculated using the technically complete Fowler-Nordheim equation [67], [68], [69]

$$J(\phi, E) = \lambda \frac{a\beta^2 E^2}{\phi} \exp\left[-\frac{\nu b \phi^{3/2}}{\beta E}\right] \tag{18}$$

where ϕ is the local work function, E is the applied electric field, a and b are constants and λ is a pre-exponential correction factor. The correction factor ν is a known function [69]. The field enhancement factor β is a parameter that describes the modification of the

[1]Reprinted figure with permission from H. Toijala, K. Eimre, A. Kyritsakis, V. Zadin, and F. Djurabekova, "Ab initio calculation of field emission from metal surfaces with atomic-scale defects", Phys. Rev. B, vol. 100, p. 165421, 2019, https://doi.org/10.1103/PhysRevB.100.165421. Copyright 2019 by the American Physical Society.

applied electric field on a certain site on the emitter [61]:

$$\beta = \frac{E_{\text{local}}}{E} \tag{19}$$

As seen in equation 18, the field emission current depends exponentially on β. The Fowler-Nordheim equation can be linearized, and so finding the value of β experimentally becomes quite simple:

$$\ln\left(\frac{J\phi}{aE^2}\right) = \frac{1}{\beta}\frac{\nu b\phi^{3/2}}{E} + \ln\lambda\beta^2 \tag{20}$$

When $\ln\left(\frac{J\phi}{aE^2}\right)$ is used as the y coordinate and $\frac{\nu b\phi^{3/2}}{E}$ as the x coordinate, β can be found as the reciprocal of the slope of the line using semi-logarithmic axes. An example Fowler-Nordheim plot can be seen in figure 4. The figure shows computational field enhancement factors for different surface defects from another study. Some field enhancement factors measured by Descoeudres *et al.* can be seen in figure 5. The local field can be enhanced compared to a flat surface for example by protrusions and other surface roughness. Field enhancement factors in the range of 30–140 were measured, but protrusions with a β higher than 50 have not been found in microscope images, so the origin of the field enhancement still remains an important question to be answered [11].

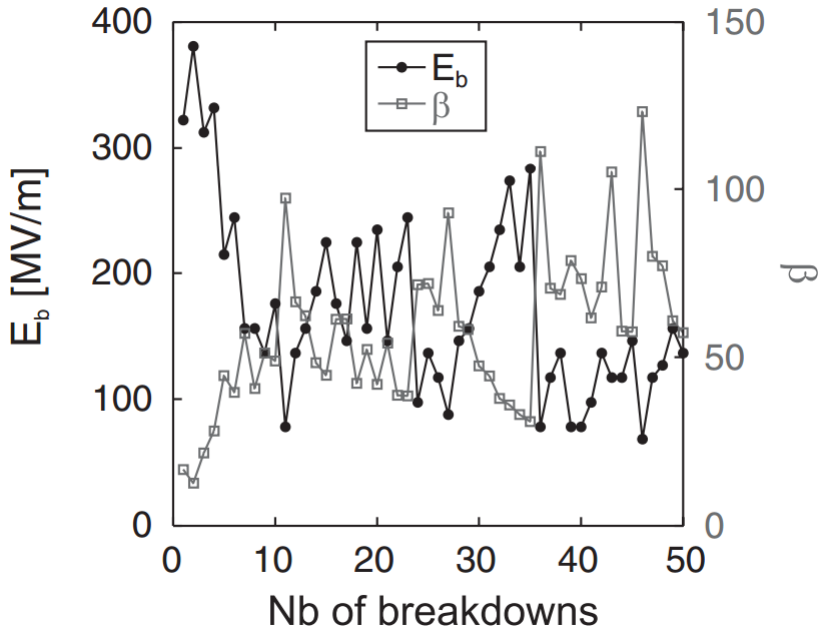


Figure 5: Measured field enhancement factors β and breakdown field strengths during a conditioning experiment on Cu electrodes. Figure[2] by Descoeudres *et al.* [11].

[2]Reprinted figure under Creative Commons Attribution License 3.0 (https://creativecommons.org/licenses/by/3.0/).

Another possible cause for enhanced field emission is a decrease in the local work function ϕ due to surface defects, since this would make it easier for the electrons to detach from the surface. Since the true local work function is not known during experiments, a value for a clean surface is used instead. In this case, the increase in J will be attributed to β. However, it was found in an *ab initio* study at the Helsinki Accelerator Laboratory, that changes in work function do not account for the large βs. Instead, the apparent field enhancement due to the decrease of the work function was found to be only 1.14–1.24 for the studied defects [65].

The protrusions that give rise to the field enhancement are also expected to initiate breakdowns, because the field emission current can heat the protrusion and melt or evaporate it. The released particles can then form plasma, which makes a good path for the vacuum arc [11]. The origin of the protrusions still remains an open question. However, it is suspected that electromigration, or diffusion of atoms under electric field, plays a role in the formation of the protrusions [12]. This motivates us to study surface diffusion, and specifically surface diffusion under electric fields, in order to gain a better understanding of the vacuum breakdowns in CLIC.

3.2 Nanosized field emitters

Nanosized field emitters have many uses in contemporary and emerging technologies. For example, they are used as electron sources in electron microscopes. Compared to the more traditional electron source, a hot filament, a field emitter provides a much brighter beam of electrons. This reduces the time required to make an image, which also makes it easier to focus the instrument and keep it stable during the imaging [57]. Another possible application for nanosized field emitters are flat displays. Liu *et al.* [58], for instance, report that they have manufactured a 16×16 pixel monochrome display using carbon nanotube field emitters. The emitters shoot electrons into phosphorescent anodes, which then emit visible light. They suggest that besides monochrome pixels, RGB pixels could be constructed using different materials on the anodes. Field emitters could also be used in vacuum field emission transistors (VFET). In a regular transistor, the electron current flows through the transistor via a semiconductor channel. In a VFET, the field emitted electrons fly across a vacuum gap instead. Since the theoretical speed of electrons in vacuum is 3×10^{10} cm/s but only on the order of 1×10^{7} cm/s in silicon, for

example, the VFETs are expected to operate at higher frequencies than semiconductor transistors. Furthermore, unlike semiconductors the vacuum would be undisturbed by ionizing radiation or high temperatures, which makes VFETs very suitable for space technologies [59].

The above mentioned applications all require the fabrication of suitable nanotips to acquire desired properties of the electron beam. It has been observed that under a strong electric field, the surface of a nanotip can be modified through surface diffusion. In particular, when the field is strong enough, the tip tends to become sharper, and sometimes nanoprotrusions can grow on the tip. Since the process is thermally activated, this kind of surface modification is enhanced by raising the temperature of the tip [61]. For example, Yanagisawa *et al.* [62] have grown small nanoprotrusions on W tips using a high DC voltage and laser pulse irradiation of the tip. The laser raised the temperature of the tip locally, which resulted in asymmetric changes on the tip surface. They suggest that their method could be used to fabricate very sharp nanotips for technological applications. This shows that understanding surface diffusion under electric field is important for designing and using novel techniques for producing field emitter tips with desired properties.

3.3 Surface diffusion in metals

Diffusion in solids is a process, where the atoms of the material jump to nearby lattice sites. The process is thermally activated, which means that an energy barrier must be overcome for the jump to happen [70]. A schematic example of a one dimensional energy barrier can be seen in figure 6. The atom is at first in a local potential energy minimum on the left. To jump to the other local minimum on the right, it needs to have an energy of E_m, which is the difference between the potential energy of the atom at the initial lattice position and the highest potential energy on the minimum energy path. The figure also shows the energy barrier of jumping back to the minimum on the left. For this jump, the barrier is smaller, because the original potential energy of the atom is higher.

There are several mechanisms by which the jumps occur, but the most important one for the diffusion of the atoms in the lattice is the so called vacancy mechanism. This means that the atoms jump to nearby vacant lattice sites. The vacancy mechanism is so important because it has a low energy barrier compared to other mechanisms, such as direct exchange, where two atoms move at the same time to switch places. Surface

diffusion is faster than bulk diffusion [70]. This is probably because on a surface there are typically plenty of empty lattice sites where an atom can jump. Also, the atoms on the surface have fewer neighbours than the ones in the bulk, so they are more loosely bound to their lattice sites. It thus takes less energy to leave the original site. Furthermore, the jumping atom often needs to squeeze past some neighbouring atoms to get to the new site. Since these atoms are also loosely bound, it is easier to displace them and move past them during the jump.

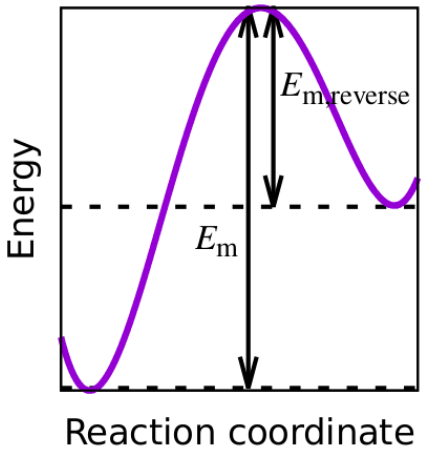


Figure 6: Example energy barriers of diffusion jumps. The purple curve shows the minimum energy path between the two local minima, E_m is the amount of energy needed to jump from the left minimum to the right, and $E_{m,\mathrm{reverse}}$ is the amount of energy needed to jump from the right minimum to the left. Figure[3] by Kimari *et al.* [13].

In this work our main interest is simulating surface diffusion with computational methods. An accurate simulation method for diffusion is Molecular Dynamics (MD) [33]. In MD, the forces between the atoms are calculated from a semi-empirical potential and used to track the movement of the atoms in small timesteps. The force calculations are relatively heavy and the length of a timestep is on the order of femtoseconds, so the length of the simulation is limited. Nowadays, however, the simulations can extend even to the microsecond range [71]. With short timesteps, the random walk of the atoms can be followed very closely: one can even see the vibrations of the atoms on their lattice sites and the repeated attempts to jump to a neighbouring site. The downside is that long simulations with large systems require a large amount of computational resources to reach even nanosecond timescales, which are still quite short to study the effects of diffu-

[3]Reprinted figure with author's permission.

sion. It is possible to speed up the diffusion processes in MD by increasing the simulation temperature, so that the jumps happen more frequently, but often it is better to use a different simulation method.

Another simulation method for diffusion is the the Kinetic Monte Carlo method (KMC), which can reach much longer timescales. In KMC the dynamic interactions of the atoms are omitted, and instead the jumps that take place in the system are picked randomly from a list of possible jumps. It is thus very well suited for simulating random walks and diffusion, and because the calculations are lighter, longer timescales can be reached. The KMC method is described in more detail in section 3.4.

The effects of strong electric fields on surface diffusion have been studied previously at the Helsinki Accelerator Laboratory. In 2017, Jansson *et al.* [72] used KMC simulations to simulate adatom diffusion on a W surface in electric fields. The electric field above the surface was calculated with a field solver and the change of the activation energy of diffusion due to the field was calculated with a formula derived by Tsong and Kellogg [73]. It was found that applying the field enhances diffusion. However, the formula uses the dipole moment and polarisability of the adatom only, whereas from a quantum mechanical point of view, the dipole moment and polarisability of the whole system should be considered. A better formula for migration barriers under electric fields was derived by Kyritsakis *et al.* [74] in 2019. The work also contains a formula for migration barriers under electric field gradient. The formulae were applied to adatom diffusion on the W surface, and the necessary parameters were obtained by DFT calculations. The results were different depending on the direction of the electric field. On the cathode, the barriers decreased as the strength of the electric field increased. Diffusion towards weaker fields was preferred up to a field strength of 11 GV/m, after which the trend reversed. On the anode, the activation energy at first grew with higher fields, but started to decrease after a turning point at 10 GV/m, and diffusion had a preferred direction towards higher fields. However, the authors note that the results can be different for different materials and diffusion processes. The new model was then used in KMC simulations by Jansson *et al.* [75] to study the evolution of W nanotips under electric fields. It was seen that because of the preferential diffusion caused by the electric field, the nanotips under a cathode field flattened, but the ones under anode fields grew in height and became sharper. These studies show how complex a task it is to model diffusion under electric fields.

3.4 Kinetic Monte Carlo approach

The Kinetic Monte Carlo method [8], [9] is a numerical simulation method that uses random sampling of events to find possible trajectories for the time evolution of a system. Because of the random sampling, it is often used to model processes related to diffusion. Examples include the migration of vacancies interstitials [76], motion of dislocations and grain boundaries [77] as well as catalysis on surfaces [78]. Since each KMC run only gives one randomly chosen trajectory of the kinetic evolution of the system, several runs are needed to produce reliable statistics. The results can then be obtained by studying the resulting distribution of the properties of interest.

The simulated system is always in some state, which can be characterized for example by the coordinates of the atoms in the system. The system can then move to a different state by an event, in our case a diffusion jump. The event is chosen randomly from the pool of possible events so that the probability of choosing the event is proportional to the rate with which the event would occur [79]. In the case of thermally activated atomic jumps, the rates of the events can be calculated from the Arrhenius equation [80]:

$$\Gamma = v \exp\left(\frac{-E_m}{k_B T}\right) \tag{21}$$

where v and E_m are the attempt frequency and the energy barrier of the process respectively, k_B is the Boltzmann constant and T is the temperature of the system. To choose the next event, a cumulative function of the rates is calculated at first [9]:

$$R_i = \sum_{j=1}^{i} \Gamma_j \tag{22}$$

Here R_i are the values of the function and Γ_j are the rates of different processes, and i and j are the indices of the events. Then a random number u is chosen from a uniform distribution between 0 and 1, and multiplied by Γ_{tot}, which is the sum of all rates. The index of the next event is the one for which [9]

$$R_{i-1} \leq u\Gamma_{\text{tot}} < R_i \tag{23}$$

A visualization of the choosing process can be seen in figure 7.

Simulation time is not included in the above algorithm. In a real system the events would occur at varying intervals, which makes the simulation time a random number. The time can be estimated using the residence time algorithm. After each event, a new

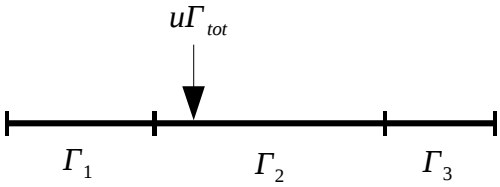


Figure 7: Picking the next event in KMC from three alternatives. The line represents the cumulative function of the rates Γ_i. The value of the random number $u \in (0, 1]$ times the sum of all rates Γ_{tot} can be found somewhere on the line. In this case it is in the segment of Γ_2, so event 2 is chosen as the next one.

uniform random number between 0 and Γ_{tot} is generated and the time is incremented by a timestep Δt:

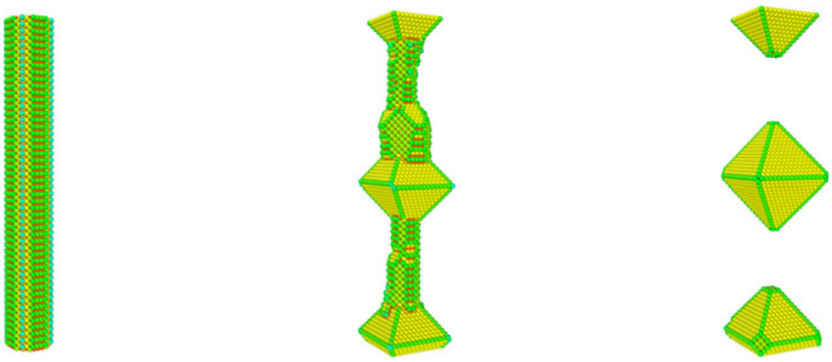
$$\Delta t = -\Gamma_{\text{tot}}^{-1} \ln(u) \tag{24}$$

where u is the random number. This way the simulation time will be approximately proportional to the real time [9]. We see that Δt is proportional to the inverse of the sum of the rates. This means that if the rates are low, the timesteps tend to be longer, whereas higher rates give rise to shorter timesteps. As a result, longer timescales can be reached with the same number of simulation steps when the studied processes are rare. This is one of the appeals of KMC, since it helps save computational resources when studying rare processes.

In principle, the only restriction for the movement of atoms in a solid is that the atoms cannot overlap. In a KMC simulation, however, the list of possible events and the number of necessary barriers can easily become unfeasibly large if no other restrictions are applied. One way to reduce the number of possible jumps is to assume a rigid lattice, where the atoms can only occupy lattice positions [79]. This is usually a reasonable approximation in solid metals, where most of the atoms mainly vibrate around their lattice positions, but it also limits the phenomena that can be simulated with the model. Relaxation effects and surface reconstruction cannot be described in a rigid lattice, and many lattice defects, such as interstitial atoms and dislocations, are also excluded. Taking the limitations into account, rigid lattice KMC is still very useful tool for studying surface diffusion.

The KMC code Kimocs [81], that is used for KMC simulations in this work is an example of a KMC code that uses the rigid lattice approximation. It has been used to

simulate for example the flattening of Cu nanotips on a surface and the necking of Cu nanowires. The evolution of a nanowire during the simulation is shown in figure 8. The system is at first a nanowire with ⟨100⟩ direction along the wire, as seen in figure 8a. Periodic boundary conditions are used to simulate an infinitely long wire. Via diffusion jumps to first nearest neighbour position, larger clusters start to form along the wire while the parts between the clusters become thinner. This is the necking phenomenon, and it can be seen in figure 8b. Finally, the clusters detach from each other, as shown by figure 8c. The ratio of the average distance between the clusters to the initial wire radius and the ratio of the average cluster diameter to the initial wire radius are in agreement with theoretical predictions [81].



(a) 0 μs (b) 3 μs (c) 6 μs

Figure 8: Example use case of Kimocs KMC code: necking of Cu nanowire. Pictures a)-c) show the evolution of the nanowire during the simulation. Periodic boundary conditions were used. Figure[4] by Jansson *et al.* [81].

3.5 Parametrisation of KMC models for surface diffusion

The migration energy barriers and attempt frequencies are not predicted by the KMC method. Instead, they need to be given to the model as input parameters. The barriers must be correct to describe the processes well, and the attempt frequencies need to be right to obtain the correct time scale for the processes. Finding the attempt frequencies is the simpler task. The attempt frequency is the frequency of atomic vibrations at the

[4]Source: V. Jansson, E. Baibuz, and F. Djurabekova, "Long-term stability of Cu surface nanotips, " Nanotechnology, vol. 27, p. 265708, 20 May 2016, https://doi.org/10.1088%2F0957-4484%2F27%2F26%2F265708. © IOP Publishing. Reproduced with permission. All rights reserved.

lattice site in the direction of the jump, and it is usually simply assumed to be of the order of the Debye frequency of the material [70]. It can also be assumed that the jumps made by the same atomic species have the same attempt frequency, since the atoms will have the same vibration frequency, too. Often these assumptions yield good enough results, but there are also other methods for determining the attempt frequency. For example, the value can be chosen so that it reproduces the experimental diffusion coefficient of the material [82]. It is also possible to fit the attempt frequency to reproduce the timescale of the simulated processes in MD [81].

Finding the energy barriers for the different events is also simple in principle, but requires more effort because so many barriers are often needed. Often it is not possible to find experimental values for all the required barriers, and thus the values need to be calculated using MD or DFT, for example. The Nudged Elastic Band method (NEB) is a practical computational method for this task [14]. The method uses several replicas of the jumping atom, which are connected by a spring force. This chain of replicas forms the elastic band. In addition to the spring force, the replicas also feel forces exerted by the real atoms of the system. However, only the component of the interaction that is orthogonal to the spring force is taken into account, which produces the nudge. The forces can be calculated from an MD potential or from the electronic structures using DFT. When the energy of the structure is then minimized, the chain of replicas relaxes along the minimum energy path of the jump. However, it may be that in the end none of the replicas lie exactly at the saddle point. This can be corrected with the climbing image method [83]. After the energy minimization, the replica with the highest energy is chosen and its energy is maximized instead. As a result, it moves along the minimum energy path towards the saddle point. The energy barrier can then easily be computed as the energy difference between the initial state and the replica with the highest energy.

Before the barriers can be calculated, the relevant jumps need to be identified and labeled. Then the corresponding structures can be generated and NEB or other methods can be used to find the barriers associated with the jumps. It is also necessary to recognize the possible jumps in the simulated system during a KMC run so that the correct barriers can be used. Since the barriers depend on the local environment of the jumping atom, one way to characterize the jumps is to parametrise the environment by using local atomic environment descriptors.

3.5.1 Local atomic environment descriptors

One method that has been used at the Helsinki Accelerator Laboratory to parametrise the local atomic environment (LAE) of diffusion jumps is to use a four parameter description, or 4D model in short. The model was described by Jansson *et al.* in 2016 [81]. In this approach the parameters are the number of first and second nearest neighbours (1nn and 2nn) of the initial site of the jumping atom, and the number of first nearest and second nearest neighbours of the landing site. This model can be used to describe jumps where the atom jumps to a vacant first nearest neighbour site in a single element system. The barrier values can be calculated with the NEB method. The number of different barriers in this model is relatively low, and the necessary barriers can be calculated and used in KMC quite easily. However, the 4D model is limited. In the 4D model various configurations of neighbours are associated with the same barrier, but in reality the different configurations have different barriers. In reference [81] the standard deviation for a set of energy barriers was 0.13 eV, corresponding to 14.8 %. The precision is decent, but still leaves room for improvement. For this reason a 26D model that also describes the locations of the neighbours was developed.

In the 26D model developed by Kimari *et al.* [13], each of the 1nns and 2nns of the initial site and the landing site has its own parameter. The parameter is either 1 or 0, and it denotes whether the particular neighbour is present or not. The model has only been used for pure FCC Cu, but it can be straightforwardly extended to multicomponent systems by using different numbers to denote different elements. The 26D model gives a more detailed representation of the local potential energy landscape, and thus of the associated barriers, than the 4D model. This can be seen in figure 9, that shows the comparison between barriers in the 4D and 26D models. Although the general agreement between the datasets is decent, there is a region where the average 26D barriers are clearly lower than the 4D. Furthermore, the 26D barriers that correspond to one 4D descriptor often have a range of over one eV. Theoretically, $2^{26} \approx 67$ million different jumps can be described with the 26D descriptor. Because of the large number of barriers, Kimari *et al.* used artificial neural networks to predict barrier values based on a set of calculated barriers. To ensure the stability of the systems during NEB calculations, the calculated set consisted of systems where each atom had at least three first nearest neighbours. There were about 11.7 million such barriers in total.

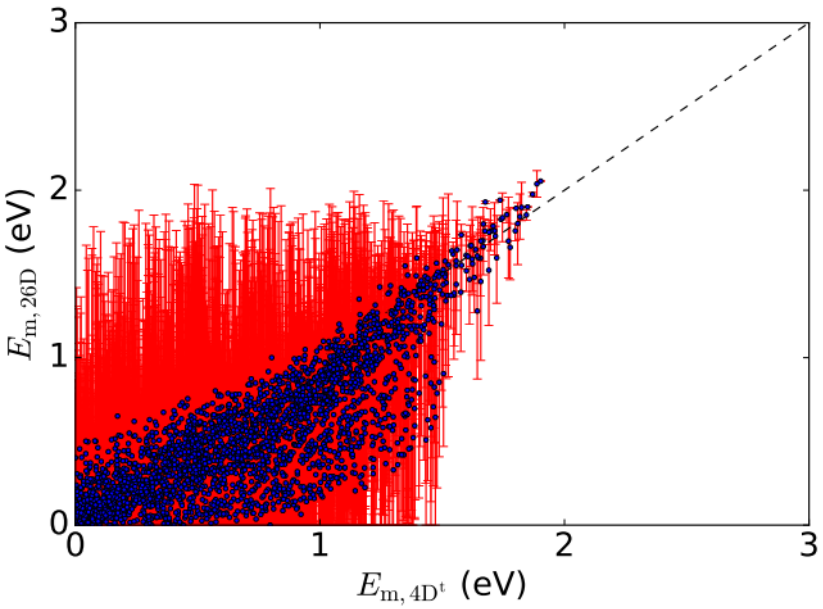


Figure 9: Comparison of the 4D and 26D barrier datasets by Kimari *et al.* The 4D barriers are on the horizontal axis and the 26D barriers are on the vertical axis. The red bars show the minimum and maximum value of the 26D barriers, and the dots show the mean values. Figure[5] by Kimari [13].

It is reasonable to expect that mainly the first and second nearest neighbours affect the barriers, because the electrical interactions between atoms fade quite fast as the distance between the atoms grows. In fact, the potentials used in MD often apply a cutoff radius, so that atoms that are far away from each other do not interact at all. If the barriers are calculated with a potential like this, one might expect that atoms outside of the cutoff radius do not contribute significantly to the result, because they do not directly interact with the jumping atom. According to an article by Castin and Malerba from 2010 [4], however, the atoms can interact indirectly through strain fields and this interaction has an effect on the barriers. They studied the average error of the barrier value as a function of the number of atoms included in the LAE descriptor and found that at least 200 atoms should be included to achieve optimal accuracy. When 200 atoms were included, the average error compared to a descriptor with 615 atoms was reduced to 1 %. Furthermore, when they used the calculated barriers to train an artificial neural network, the prediction error converged when about 100 atoms were used as input. This is more than the 77 atoms within their MD potential's cutoff radius from the initial site and the final site. In both

[5]Reprinted figure with author's permission.

cases the number of parameters is larger than 26, so it is probable that expanding the 26D model to include more neighbours will further improve the description of the barriers, as well as the prediction accuracy.

3.6 Artificial neural network for KMC parametrisation

Neural networks have been used to predict migration energy barriers for rigid lattice KMC simulations by at least Djurabekova *et al.* [2], Castin and *et al.* [3], [4], [5], Pasquet *et al.* [6], Messina *et al.* [7], and Kimari *et al.* [13]. In references [2], [3], [4], [5], [6], and [7] the main interest was bulk diffusion in Fe alloys, usually Fe-Cu. Djurabekova *et al.* used a multilayer perceptron with one hidden layer and LAE descriptors containing 14 or 21 atoms, which correspond to atoms within the 1nn or 2nn distance. Castin *et al.*, Pasquet *et al.*, and Messina *et al.* also used multilayer perceptrons with a single hidden layer, but they applied a different training algorithm than the traditional backpropagation used by Djurabekova *et al.* Instead, they used a constructive algorithm that allows connecting the inputs gradually to the network and creating new hidden nodes during the training [4]. They also used larger LAE descriptors that include more atoms. In particular, Castin and Malerba tested LAE sizes up to 615 atoms in reference [4], and reported that the barrier values obtained from NEB calculations converged when at least 200 atoms were included, and the ANN prediction accuracy converged when 100 or more atoms were included. However, Messina *et al.* found that in their case the neural network achieved good prediction accuracy with 76 atoms included in the descriptor. Overall, the reported average errors in the predictions compared to the NEB calculations were between 1.1 % and 8.9 % in references [2], [3], [4], and [6].

In the work by Kimari *et al.* [13], neural networks were used to predict the migration energy barriers for surface diffusion in pure Cu. They used the 26D descriptor described above. The data from jumps on different surfaces — {100}, {110}, and {111} — was separated to different sets and separate predictor networks were trained for each surface. Because the networks were used to pass the barriers to a KMC simulation, it was necessary to identify which process belongs to which surface in order to use the correct network. This was done by training a classifier network, which takes a process descriptor as an input and gives the surface where it belongs to as an output. In addition, five multilayer

perceptrons were trained on different subsets of the data, and combined to an ensemble. The average of their predictions was taken to be the barrier value.

Combining the three network ensembles and the classifier, Kimari *et al.* [13] achieved a RMS error of 0.086 eV in the dataset containing all 11.7 million barriers. When the network combination was used to simulate Cu nanotips on a surface, the tips on {100} and {111} surfaces flattened as was expected, which indicates that the machine learning method is working. On the other hand, on the {110} surface the situation is not entirely clear: in high temperatures a ridge often formed on the surface and the surface was eventually covered with facets of {100} and {111} orientation. Roughening of the {110} surface at high temperatures is a known phenomenon, but it is not certain whether the observed behaviour matches this roughening.

4 Methods

The primary computational tools used in this work are artificial neural networks, which are used to predict migration energy barriers, and the 72D LAE descriptor, which describes the environment of the jump up to fourth nearest neighbours. The architecture and training of the neural networks are covered in section 4.1, and the 72D LAE descriptor is handled in section 4.2. It is also important to describe how the structures for training and testing the network were selected and how the NEB calculations were carried out to find out the corresponding barriers. This is done in section 4.3.

4.1 Artificial neural networks

Artificial neural networks were used to predict migration energy barriers on the Cu surface. Both cascade networks and fixed-size multilayer perceptrons with a single hidden layer were tested with different numbers of hidden nodes to see which network architecture gives the best predictions. The neural networks were trained by a supervised learning scheme, where the network is given example input and output data pairs. The root-mean-squared (RMS) error of the network predictions compared to the desired output was followed during the training to see whether the network is still learning. The fixed networks were trained using the iRPROP algorithm [49] and the cascade networks using the Cascade2 algorithm [84]. The algorithms are implemented in the open source Fast Artificial Neural Network (FANN) library, which was utilized for training and using the networks. The library is described thoroughly in reference [85], but many updates have been made to the library since the publication of the report. For example, cascade networks and new learning algorithms have been added. The most up to date description is the reference manual on the FANN website [86].

Both multilayer perceptrons and cascade correlation networks were trained with the NEB calculated barrier data. All networks had 72 input nodes, one for every digit in the binary local atomic environment descriptor, and one output node for the barrier value, but the number of hidden nodes varied. When the 26D model was used in the work of Kimari *et al.*, the best prediction accuracy was achieved when the size of the network was 35 nodes for a multilayer perceptron with a single hidden layer and about 30–70 nodes for a cascade network [13]. At first it was expected that more nodes would be

needed with the 72D model, since the input vector was longer, and a smaller network might not be able to learn all the patterns in a more complex model. For this reason it was decided to first train fixed networks with 30, 70, and 100 nodes and a single hidden layer, as well as cascade networks with 40 and 60 nodes. Separate networks were trained with the {100}, {110}, and {111} surface diffusion data, because in Kimari's work this reduced the error in the predictions compared to training a single network. In addition, networks were trained to predict the barriers of so called surface breaking jumps, where the jump direction points outward from the surface. The training datasets are described in more detail in section 4.3.

During the training of cascade networks it turned out that the mean squared error of the network compared to the training set of barriers did not decrease anymore after adding about 20 nodes. Therefore 20 nodes was deemed a sufficient size for a cascade network. It also soon became evident that the multilayer perceptrons were much slower to converge to a point where the error no longer decreases compared to the cascade networks. In addition, even with the longer training their prediction performance was worse, so they were abandoned. More information on finding the optimal size and structure for the network can be found in section 6.

4.2 72D Local atomic environment descriptor

The energy barrier of atomic diffusion in metals is affected by the local atomic environment of the jumping atom. The effect of the first and second nearest neighbours of the jumping atom and the landing site has been studied in previous work by Kimari *et al.* [13]. In this work, the local atomic environment is expanded to include the third and fourth nearest neighbours, so that the environment comprises 72 atoms. The environment can be described by a binary 72 digit integer, where each digit corresponds to a certain site in the local environment. If that digit is zero, there is a vacancy in that site, and if that digit is one, that site holds an atom. The LAE cluster where all the neighbours are present can be seen in figure 10. The figure also shows how the atoms in the cluster correspond to digits in the descriptor. This method of describing the local environment can be straightforwardly expanded in the future to include more neighbours, if necessary, by using more digits. It is also possible to modify this method for more than one atom type. One could for example use the number 2 to mark atoms of another type, or use

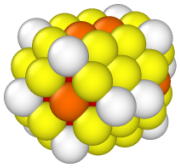
separate binary integers for all atom types. Each integer could then hold information of whether an atom of that type is present or not.

Because the constructed local atomic environment consists of 72 atoms, a total of $2^{72} \approx 4.7 \times 10^{21} = 4.7$ sextillion environments can be described with the present model. However, not all of these environments are unique. The environment has three reflectional symmetries, so for one environment there may be up to seven physically equivalent environments. The lower bound of the number of different environments thus comes to $2^{64} \approx 1.8 \times 10^{19}$, which is still an extremely large number. Many of these environments are unstable, and in reality they would quickly collapse to a different structure. Due to the vast number of different environments, finding all the stable configurations and then calculating the barriers for them separately is not feasible. Instead, it is possible to compute some subset of the barriers and use them to train an artificial neural network, which then in turn predicts the rest of the barriers more or less correctly.

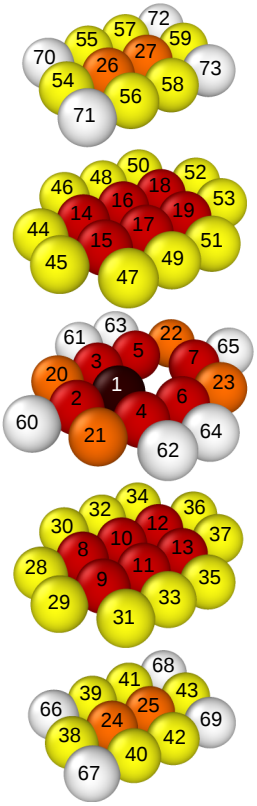
4.3 Simulation setup

For the barrier calculations, the 72D LAE cluster containing the jumping atom and the vacancy site along with their up to 4th nearest neighbours is embedded into a slab with a smooth surface. The surface can have either {100}, {110}, or {111} orientation, and the LAE cluster is rotated to match the surface. The size of the slab is approximately 4.5 nm × 4.5 nm × 2.0 nm, but the exact size depends on the surface orientation. Two kinds of configurations were created: surface processes and surface breaking processes. In the surface processes the cluster is positioned so that the jumping atom is in an adatom layer immediately above the rest of the flat surface and the jump direction is parallel to the surface. In the surface breaking processes, the initial position of the jumping atom is in the surface layer and the jump direction is diagonally upwards from the surface. The final position of the jumping atom is in the adatom layer.

When 3rd and 4th nearest neighbours are included the number of different configurations is much larger than in 26D, so the choice of training data becomes crucial. The data points should represent the complete set of barriers well, which means that there should be enough data points from diverse configurations. On the other hand, however, calculating millions of barriers and training networks with them also requires a lot of computational resources. In this work the main interest is surface diffusion, so jumps on



(a) LAE cluster



(b) Labeled neighbours in the cluster

Figure 10: Figure a) shows the 72D LAE cluster with the {100} surface facing up. The cluster contains the jumping atom, the vacancy site, and their neighbouring atoms up to 4nn. Figure b) shows the same cluster in layers. The atoms are labeled in the same order as they appear in the descriptor, and the colours show which neighbour the atom is. The brown atom marked with a 1 is the jumping atom, the red ones are 1nn, the orange ones 2nn, the yellow ones 3nn and the white ones 4nn.

a rough surface were included in the training configurations. A rough surface jump is defined here as follows:

1. There are no atoms present above the jumping atom
2. Atoms from the layer where the jumping atom is and from the layer directly under it may be present or not present
3. The rest of the atoms are always present

Example structures can be found in figure 11. Separate datasets were generated for {100}, {110}, and {111} surfaces. In addition, a number of surface breaking jumps were created

on the three surfaces. In a surface breaking jump, the jumping atom moves from the surface layer to the adatom layer.

To keep the number of generated structures feasible in the surface jump datasets, some atoms in the jump layer and the layer below it were selected to be present in all structures on a given surface. This way the number of different possible structures was brought down to 2^{20}, or 2^{22} in the case of {111} surface. Then all of these structures were generated. In the surface breaking datasets, the configurations were created randomly. Half of these systems were fully random configurations, where all neighbours were given a certain probability to be present. Probabilities of 30 %, 50 %, and 70 % were used to make sure that there would be sufficiently many LAE's with different portions of vacancies. The other half were partly random configurations, where the probabilities were assigned in an otherwise similar manner, but for the neighbours below the jump layers the probability was 100 % and for the ones above the jump layers the probability was 0 %. This way, the presence of the neighbours varied only in the jump layers. The datasets were generated separately on the three different surfaces. As a summary, the datasets for {100}, {110}, and {111} surfaces contained the following kinds of configurations:

- Rough surface jumps, where the jump happens in the surface layer. The present neighbours vary in the jump layer and the layer immediately below it, whereas all neighbours below those layers are present, and the neighbours above those layers are absent.

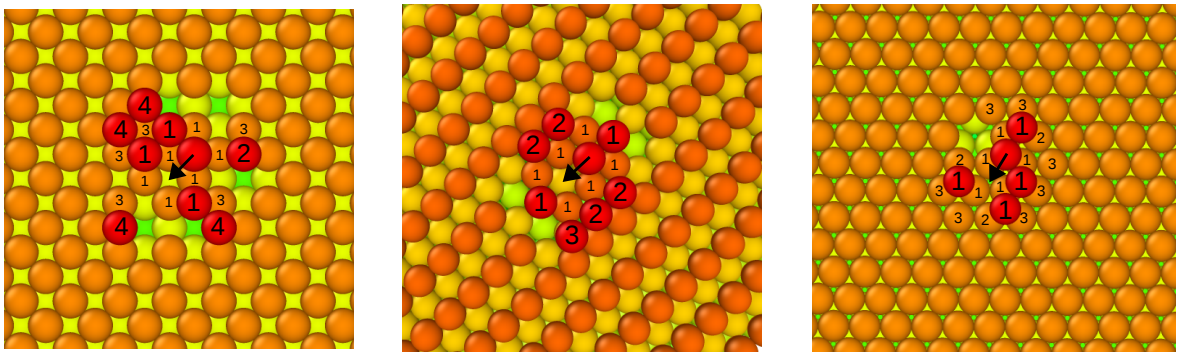
- Surface breaking jumps, where the atom jumps from the surface layer to the adatom layer. These can be divided to two classes as follows:

  - Fully random configurations, where each neighbour has the same probability to be present. There were three sets with a different probability. The used probabilities were 30 %, 50 %, and 70 %.

  - Partly random configurations, where the neighbours below the jump layers are always present and the neighbours above the jump layers are never present. The neighbours in the jump layers all had the same probability to be present. There were three sets with a different probability. The used probabilities were 30 %, 50 %, and 70 %.

As the resulting structures would be used in the NEB calculations, systems where the LAE was symmetric to a previously generated system were discarded. This way the energy barrier would be calculated only once for each unique system. Unstable structures that are likely to collapse to a different configuration during the NEB calculation were also filtered out by accepting only structures where each atom had at least a minimum number of first nearest neighbours. The minimum was three for surface jumps and two for surface breaking jumps. If the surface configuration changes during the NEB run the calculated barrier will be useless, since it does not correspond to the jump described by the associated descriptor, and the used CPU resources would thus be wasted.

After generating the structure and checking for symmetries and stability the initial and final state of the jump were printed into files, which were labeled with the process descriptor. In the surface jump datasets, there were about 850000 unique processes on the {100} surface, 250 000 on the {110} surface and 3.3 million on the {111} surface. In the surface breaking datasets, there were about 24 000 processes on each surface.



(a) {100} (b) {110} (c) {111}

Figure 11: Examples of structures picked for NEB calculation on three different surfaces. The atoms are coloured by their z coordinate and the atomic jump is marked with an arrow. Different neighbours in the jump layer and the one below it are labeled with numbers so that a 1 marks first nearest neighbours and so on.

The NEB calculations were run with the LAMMPS molecular dynamics code [87]. The interatomic forces were computed with a corrected effective medium potential by Stave *et al.* [88]. When the NEB calculations are run, the replicas interact with the other atoms of the system and may also displace them from their original lattice sites. If this happens, the obtained barrier cannot be used, since the final state would differ from the descriptor.

To prevent this, the other atoms were tethered to their original positions with a force of $2.0\,\mathrm{eV/\AA^2}$. This value is the same as in previous work by Kimari *et al.* [13]. Using the same value makes it easier to compare the results, because the tethering also changes the barriers slightly. The results of the NEB runs were collected by pairing each descriptor to the corresponding barrier.

It must be noted that the barrier data was calculated without an electric field. The networks trained with this data therefore cannot take the electric field effect into account when predicting barriers. Instead, the purpose of these networks is to improve the barrier predictions by using a more detailed description of the atomic interactions in the vicinity of the jump event. This will allow for a precise model of the diffusion without the electric field, which can in the future be enhanced with a description of the electric field effects. This can be done, for example, by calculating the barrier change due to the electric field using an analytical formula, as described in section 3.3.

5 Effect of the 3nn and 4nn environment on the surface migration barriers

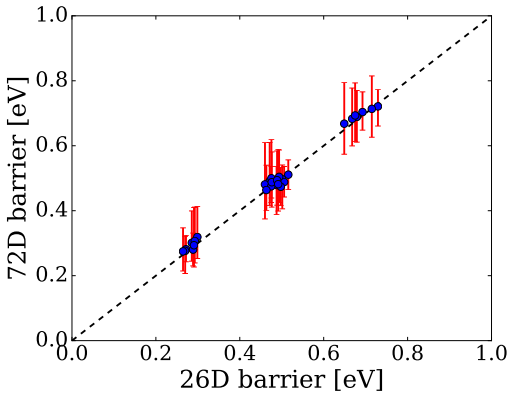
A single 26D descriptor corresponds to several 72D descriptors, which can all have different barriers. If a 26D descriptor is used instead of a 72D one, all the different barriers are replaced with a single value corresponding to the 26D descriptor, which introduces error into the training data set. The effect of 3rd and 4th nearest neighbours on the migration energy barriers was studied in a subset of about 50 000 NEB calculated surface jump barriers on each surface. This was done by first finding the processes that have the same 26D descriptor and different 72D descriptors. Then a reference case was chosen from among these processes such that beyond 2nn the system has a smooth surface. This means that in the reference case, all 3nn and 4nn below the jumping atom are present and all 3nn and 4nn in the same layer as the jumping atom are absent. The maximum, minimum, and average barriers of the jumps that had the same 26D descriptor were computed and plotted against the barrier of the reference case. The results for different surfaces can be seen in figure 12. We see that the average of the 72D barriers is quite close to the corresponding reference 26D barrier, but there is significant spread in the individual 72D barriers.

The above analysis shows that there is variation in the barriers due to changes in the 3nn and 4nn range. The distribution of these variations was studied by calculating the relative difference between the barrier of a process and its 26D reference process. The distribution of the differences was then plotted as a histogram, and the mean and the root-mean-square differences were calculated. The histograms can be found in figure 13. A negative change means that changing the 3nn and 4nn compared to the reference case lowered the barrier. On the {100} surface, the mean relative change is +1.4 % and the relative RMS difference is 7.1 %. This is reflected in the histogram in figure 13a so that there are about as much negative and positive barrier changes and that the changes are mostly quite small. In fact, most relative changes are smaller than ± 10 %. On the {110} surface the situation is quite different: the mean relative difference is −14.9 % and the mean RMS difference is 22.9 %. As can also be seen in figure 13b, most of the changes are negative, and differences of even −40 % are quite common. On the {111} surface the mean difference is +3.2 % and the RMS difference 25.9 %. The ratio of positive and

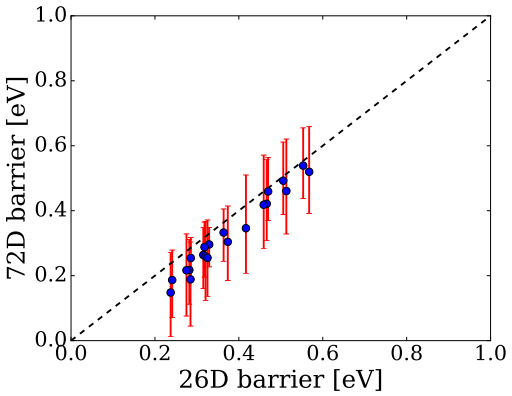
negative changes is thus balanced like in the case of {100} surface, but the changes can be larger. Figure 13c shows that the portion of changes over ±50 % is larger on the {111} surface than on {100}. These results suggest that using the 72D descriptor instead of the 26D one improves the precision of the data especially on the {110} and {111} surfaces.

The asymmetric shape of the histogram in figure 13b ({110} surface) needs an explanation. One possibility is that the presence or absence of some neighbours affects the barrier more than others. To study this, the average relative barrier change was calculated for every structure where a certain atom was absent, and the change was associated to that specific atom. The calculation was done for all 3nn and 4nn atoms in the descriptor. The average relative barrier changes related to the absence of each of these atoms can be seen in figure 14a. We see that there are some atoms, for which the barrier change is 0. These are atoms that were present in all studied structures. For atoms that were absent in all studied structures, the associated barrier change is exactly the same as the average barrier change. For the rest of the atoms, the absence of most atoms can be associated with a barrier reduction of about 15 %, which is also very close to the average. By contrast, the atoms 41 and 55 have an especially large effect on the barriers, as their absence is related to a barrier reduction of over 20 % on average. These two atoms belong to the third nearest neighbours, and they lie under the jump path. Because of symmetry, the absence of atoms 39 and 57 should also result in a large barrier reduction, but in the studied cases they are always present, so the effect could not be observed. The large effect that these third nearest neighbours can have is completely neglected by the 26D descriptor.

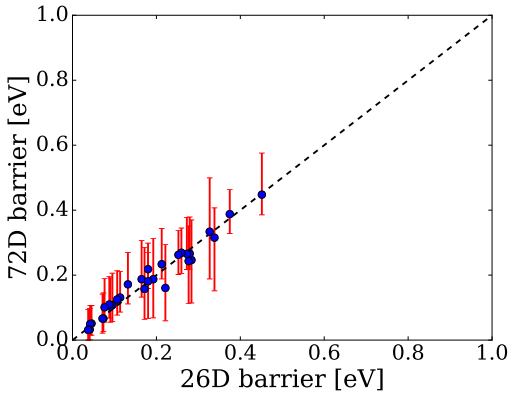
The difference between the 26D model and the 72D model is most significant on the {110} and {111} surfaces. On the {111} surface, taking into account the 3nn and 4nn has a large relative effect, because the barriers on that surface are in general quite low. On the {110} surface the barriers values are in general quite similar to those on the {100} surface, but the 26D descriptor clearly underestimates the effect that the 3nn and 4nn can have on the barriers, especially certain 3nn atoms under the jump path. Using the 72D descriptor therefore improves the quality of the data in the training set especially for the {110} and {111} surfaces.



(a) {100} surface

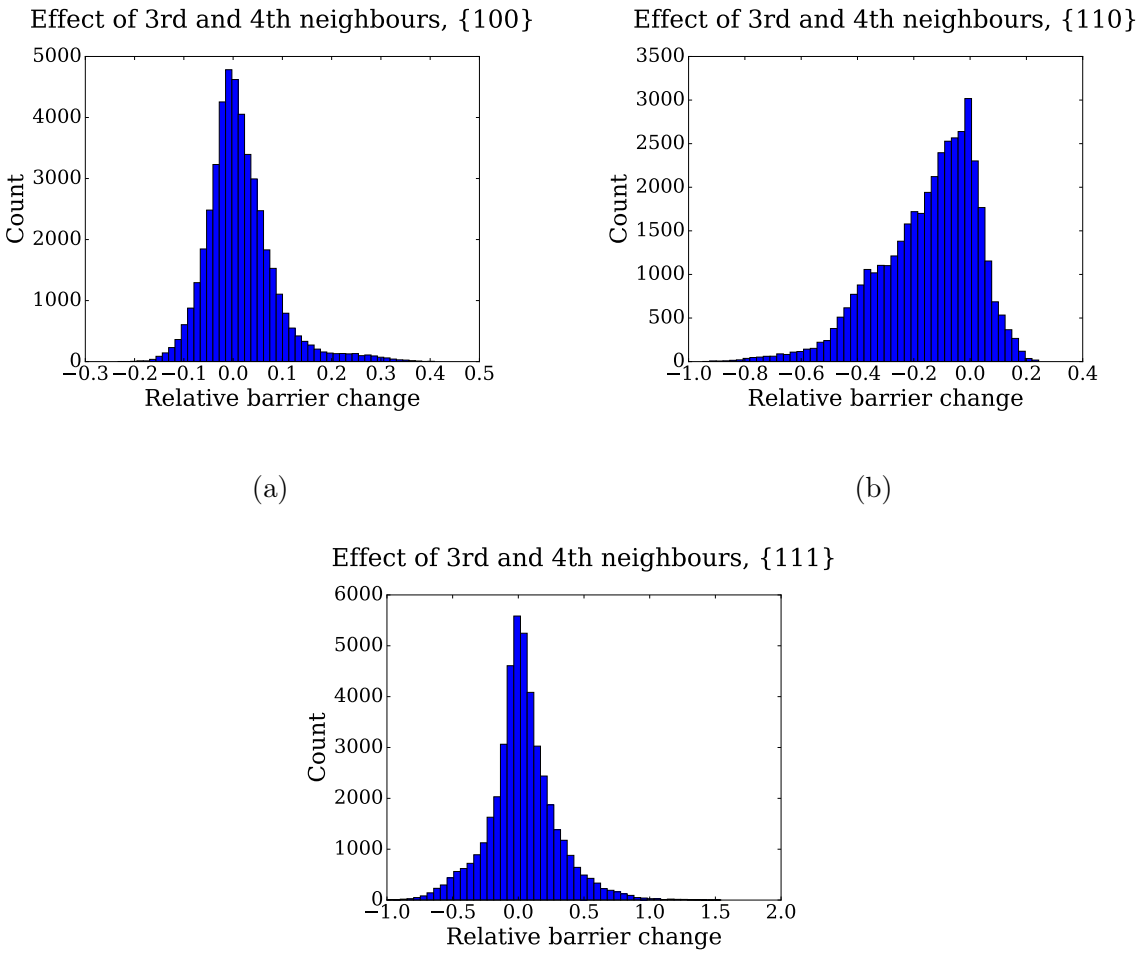


(b) {110} surface



(c) {111} surface

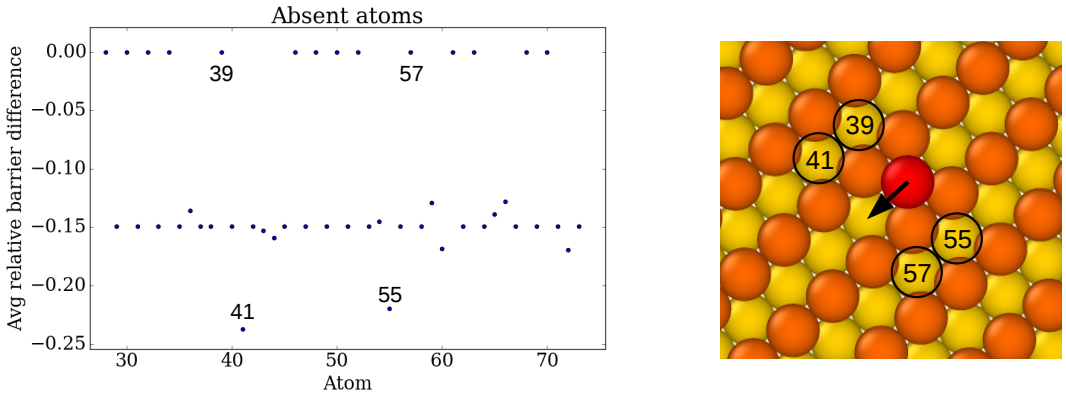
Figure 12: The range of 72D barriers that correspond to a single 26D descriptor. The 26D reference barrier is on the horizontal axis and the 72D barriers on the vertical axis. The red error bars show the minimum and maximum 72D barrier and the blue dots show the average 72D barrier. The black dashed line is the identity line $y = x$.



(a) (b)

(c)

Figure 13: Relative change of barrier value caused by changing the 3nn and 4nn on three different surfaces. On the horizontal axis is the barrier difference and on the vertical axis the number of times this difference appeared in the studied dataset. A negative change means that the barrier is reduced when the 3nn and 4nn are changed compared to the reference system.



(a) (b)

Figure 14: The average relative barrier change related to the absence of different 3nn and 4nn atoms on the {110} surface is shown in a). The numbers in the horizontal axis refer to the position of the atom in the 72D descriptor, as in figure 10b. The atoms 41 and 55 are marked in the plot because their absence is associated with a larger average change. The atoms 39 and 57 are marked similarly, because they should have a similar effect on the barriers due to symmetry. In this case however, atoms 39 and 57 were present in all studied structures, and so their associated barrier difference is zero. Figure b) shows the location of the atoms 39, 41, 55, and 57 on the {110} surface in relation to the jumping atom. The atoms are coloured according to their z coordinate and the jump is marked with an arrow. The atoms of interest are highlighted with a black circle and labeled with numbers.

6 Barrier prediction accuracy of artificial neural networks

The learning and prediction performance of the network depends on the network architecture, i.e. the structure and size of the network, as well as the amount of training data. Training larger networks is computationally more expensive than training smaller ones, so different sizes were tested to find a network with good prediction accuracy and low CPU cost. To validate the predictions, the set of NEB calculated barriers was divided to a training set and a test set. The network was trained using the training set, and after training the network was used to predict the barriers of the processes in the test set. The performance of the neural network in the test set shows whether the network is capable of predicting barriers for processes that it has not encountered before. In addition, training networks with a larger amount of data requires more computational resources, so it is also of interest, how much data is needed to achieve good performance. This was estimated with data efficiency tests, in which the sizes of the training set and the test set are varied. Furthermore, it is often possible to improve the prediction accuracy by using an ensemble of networks instead of just one, but it also takes more time. The final test was thus to see if an ensemble gives so much better predictions that it is worth the increased computational cost. Different neural network architectures and sizes were tested by using the surface jump data for training, and the data efficiency tests were also carried out using the same data. The results of these tests were then used to guide the choice of the networks which were trained with the surface breaking data.

Multilayer perceptrons with 30, 70 and 100 hidden nodes in a single layer and cascade correlation networks with 40 and 60 nodes were tested at first to find a good network architecture. The sizes were chosen based on the work of Kimari *et al.* [13], assuming that larger networks would be needed when a larger 72D descriptor is used instead of a 26D one. The networks were trained separately for the {100} and {110} surfaces, using 80 % of the calculated surface jump barriers for the training and keeping the rest for the test set. Because the data set on the {111} surface was larger than on the other two surfaces and the training would have been expensive, only {100} and {110} were used at first to find a sufficient network size. The multilayer perceptrons were trained for 14 CPU days to obtain an estimate of the error levels that they can reach. The trained network was

then used to predict the barriers for the processes in the test set. A comparison of the predicted and NEB calculated values on the {100} surface can be seen in figure 15. The RMS errors in the predictions are 0.037 eV for the 30 node network, 0.036 eV for the 70 node network, and 0.049 eV for the 100 node network. The network with 30 nodes has a peculiar feature, where the network predicts too high barriers for certain processes, which shows as a peak in the graph. In the 70 node network there is no such peak, and the overall prediction performance is similar to the 30 node network. The 100 node network has the worst overall prediction performance measured by the RMS error. On the {110} surface the RMS errors were 0.021 eV, 0.034 eV, and 0.029 eV for the 30, 70, and 100 node networks respectively. The errors are in general lower than on the {100} surface.

The cascade correlation networks were trained until the network had 40 or 60 nodes. Again, this was only done on the {100} and {110} surfaces, and the training set contained 80 % of the calculated barriers. The error of the network during the training was recorded and plotted as a function of added nodes. The results can be seen in figure 16a. In this figure we see that both networks reach RMS error levels of about 0.025 eV, which is lower than the error levels of the tested multilayer perceptrons. Furthermore, the error no longer decreases after about 20 nodes, so in the rest of this work there is no need to train the cascade networks to larger sizes than 20 nodes. The network that was trained until 60 nodes were added shows a sudden increase in RMS error at 22 nodes, but it is just a random feature that can sometimes appear during the training. Figure 16b shows the prediction performance of the fully trained 40 node network in the {100} test set. Overall the network has good prediction accuracy and the RMS error is only 0.026 eV, but there are some outlying points, for which the predicted barrier is far away from the NEB value. For example, where the NEB barriers are about 0.5 eV, the network predicts some of them to be 0 eV and some 0.9 eV. The outliers probably appeared as a result of overtraining the network even after the error level had converged at 20 nodes. On the {110} surface the RMS errors were 0.013 eV for the 40 node cascade network and 0.015 eV for the 60 node one. Similarly to prediction errors in multilayer perceptrons, the errors in the {110} surface data are lower than in the {100} surface data.

Adding 40 nodes to the network took about 6 days of CPU time for the {100} surface and 2 days on the {110} surface, whereas adding 60 nodes took 11 days on {100} and 3 days on {110}. The CPU time required for training is thus significantly shorter for cascade

networks than the 14 days used for multilayer perceptrons, and the prediction accuracy is better. Moreover, the required training time of cascade networks will be further reduced, since the same error level can be reached using only 20 nodes. Stopping the training at 20 nodes should also prevent the appearance of outlying points seen in figure 16b. Since the cascade networks clearly outperformed the MLPs, only cascade networks are used in the rest of this work.

Data efficiency tests were performed to estimate how much data is needed to train a network to make reliable predictions. To do this, sets of five cascade networks with 20 nodes were trained using 80 %, 60 %, 40 % or 20 % of the calculated barriers as the training data. The rest of the data was used as the test set. The average RMS errors of the network sets were calculated to compare the prediction capabilities of networks that are trained with different percentages of the full dataset. Training networks for the {111} surface with a large percentage of data is computationally expensive, so the tests were carried out on the two other surfaces. The results for the {100} and {110} surfaces can be seen in table 1. In general, the errors are lower on the {110} than on the {100} surface, which is probably due to the smaller data set size on {110}. The training set size, however, has little effect on the prediction accuracy, as the difference between the RMS errors due to the training set size are on the order of 0.1 meV. Based on these data efficiency tests, using a 20 % training set size should yield just as good results as a 80 % training set, so the networks for {111} surface can be trained with only 20 % of the calculated data.

		Training set %			
		20	40	60	80
Surface	{100}	0.0247	0.0248	0.0252	0.0241
	{110}	0.0128	0.0126	0.0126	0.0121

Table 1: Average RMS errors of artificial neural networks calculated in the test set. Each average is calculated from a set of five networks that were trained on either {100} or {110} surface data. The training set size was 20 %, 40 %, 60 % or 80 %, and the rest of the calculated data was used as the test set.

A set of five cascade networks was trained on the {111} surface, too, using 20 % of the calculated barriers as the training set. The sets of five cascade networks for all three surfaces with 20 % training set were then studied further to see how well the networks in

a set perform compared to each other and to the performance of the whole set used as an ensemble. A neural network ensemble consists of several networks, which all give a prediction for a certain input, and the average of the outputs is used as the result. Since there is an element of randomness in the network training, different networks may learn different features of the data better than others, and taking the average minimizes this random error. The RMS errors of the most accurate network and the ensemble of five networks calculated in the training set are tabulated in table 2. The results show that the prediction performance of the ensemble is on all surfaces slightly better than that of the single network, as expected. Reaching this slight improvement, however, multiplies the CPU time needed to make the predictions by a factor of five. Since the difference is so small, the best single networks were chosen to be used in the KMC simulations instead of the ensemble. The comparison of the NEB calculated barriers and the barriers predicted by these networks is shown in figure 17. The RMS errors of the networks were 0.024 eV on the {100} surface, 0.012 eV on the {110} surface and 0.045 eV on the {111} surface. The errors are all significantly lower than 0.086 eV, which was reached by Kimari with the 26D descriptor [13]. It should be noted that Kimari measured the prediction accuracy in a dataset with barriers from all three surfaces, and used a classifier network to find out which network should be used for which process. In this case the measurement was done for the surfaces separately, but the comparison is still valid as long as a good classifier network can be trained.

		Best single	Ensemble
Surface	100	0.0240	0.0234
	110	0.0121	0.0111
	111	0.0446	0.0435

Table 2: The RMS prediction error of the ensemble of five networks and the best single one of those networks. The networks were cascade networks trained separately for the three different surface orientations using 20 % of the calculated barriers as the training dataset. The RMS errors were calculated in the training sets.

It was also tested how the selected networks predict the energy change of the system ΔE due to the jump. The network does not predict the ΔE directly, but it can be calculated as the difference of the energy barriers of a forward jump and a reverse jump.

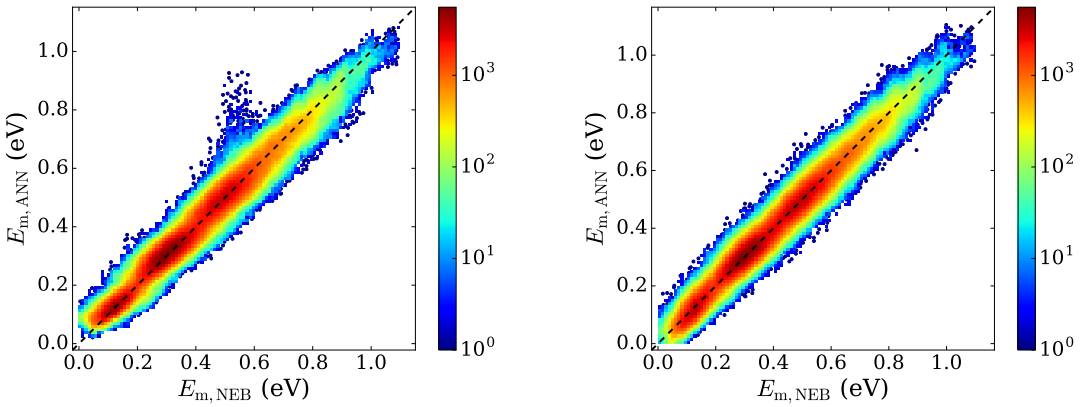
ΔE_{NEB} values of the jumps were extracted from the results of the NEB calculations, and the ΔE_{ANN} values were calculated from the barrier predictions. The calculations were done separately for different surfaces. The correlation of these values can be seen in figure 18. It can be seen that for the most part, the ANN predictions are in line with the NEB results. Most importantly, the predicted ΔE_{ANN} is mainly positive where ΔE_{NEB} is positive, and mainly negative where ΔE_{NEB} is negative. The exceptions are few, and they occur when ΔE_{NEB} is small in the first place. This means that the network should reproduce the thermodynamics of the system quite reasonably.

Based on the test results on surface jump networks, cascade networks with 20 nodes and a training set size of 20 % were used also for training the networks for surface breaking jumps. Separate networks were trained with the {100}, {110}, and {111} surface breaking jump data. In addition, a single network was trained with the data from all three surfaces for comparison. The correlation of the predictions and the barrier values calculated with NEB can be seen in figure 19. The RMS errors are now 0.18 eV on the {100} surface, 0.15 eV on the {110} surface, 0.18 eV on the {111} surface, and 0.17 eV for the network for all three surfaces. The errors are much larger than in the networks for surface jumps, which is probably because the surface breaking dataset is more heterogeneous, and includes LAEs that are very different from each other. They are also larger than the 0.086 eV in Kimari's work [13]. However, since surface jumps are expected to be the most common diffusion processes in the KMC simulations, some inaccuracies in the predicted surface breaking barriers should not affect the results too much.

Sets of five networks were also trained on the three surfaces as well as on the combined data from all surfaces to compare the prediction performance of a single network to an ensemble of five networks. The RMS errors of the best single networks and the ensembles are tabulated in table 3. Rather surprisingly, the prediction accuracy is lower for the ensemble than the single network. For this reason, single networks were used in the KMC simulations instead of ensembles for surface breaking processes, too.

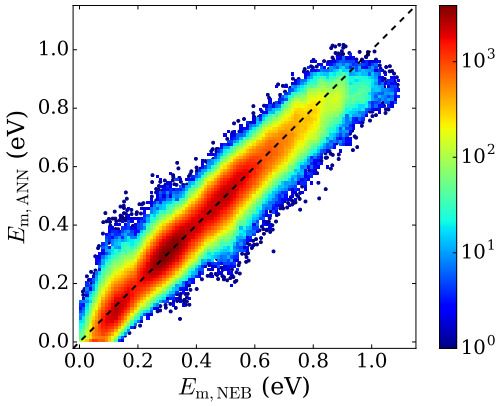
		Best single	Ensemble
Surface	100	0.1801	0.2013
	110	0.1462	0.1662
	111	0.1804	0.1922
	All surfaces	0.1673	0.1832

Table 3: The RMS prediction error of the ensemble of five networks and the best single one of those networks. The networks were cascade networks trained separately for the three different surface orientations, and for all surfaces combined, using 20 % of the NEB calculated surface breaking barriers as the training dataset. The RMS errors were calculated in the test sets.



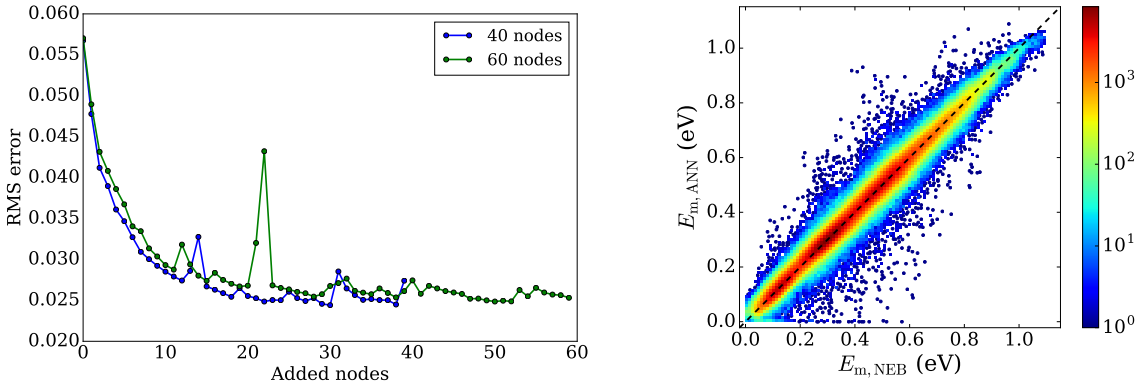
(a) 30 nodes

(b) 70 nodes



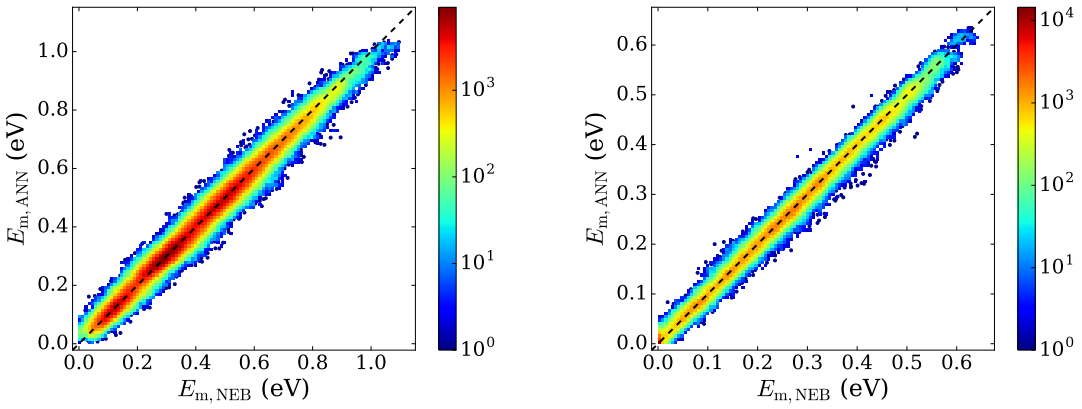
(c) 100 nodes

Figure 15: Prediction accuracy on the {100} surface in the test set using networks with a different number of hidden nodes. The value on the horizontal axis shows the NEB calculated barrier value, and the value on the vertical axis shows the barrier value predicted by the neural network. The colours show the density of points in logarithmic scale. The black dashed line marks the identity line $x = y$. If the predictions are close to the NEB calculated barriers, the pattern is centered closely around this line. The RMS errors were 0.037 eV for 30 nodes, 0.036 eV for 70 nodes and 0.049 eV for 100 nodes.



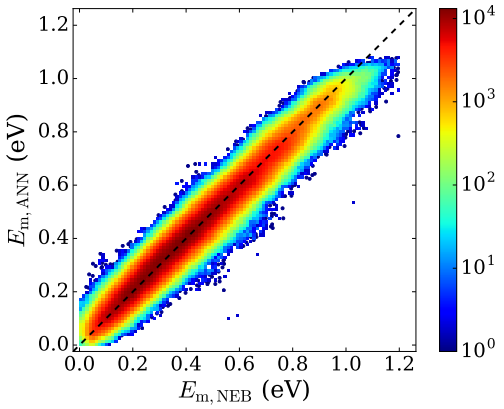
(a) (b)

Figure 16: a): Evolution of RMS error in two cascade networks during the training as a function of added nodes. The blue dots show the RMS error in the 40 node network and the green dots in the 60 node network. The lines are guides for the eye. Both networks were trained with 80 % of the calculated {100} data. b) Prediction accuracy of the 40 node network in the test set. The value on the horizontal axis shows the NEB calculated barrier value, and the value on the vertical axis shows the barrier value predicted by the neural network. The colours show the density of points in logarithmic scale. The black dashed line marks the identity line $x = y$. The RMS error was 0.026 eV



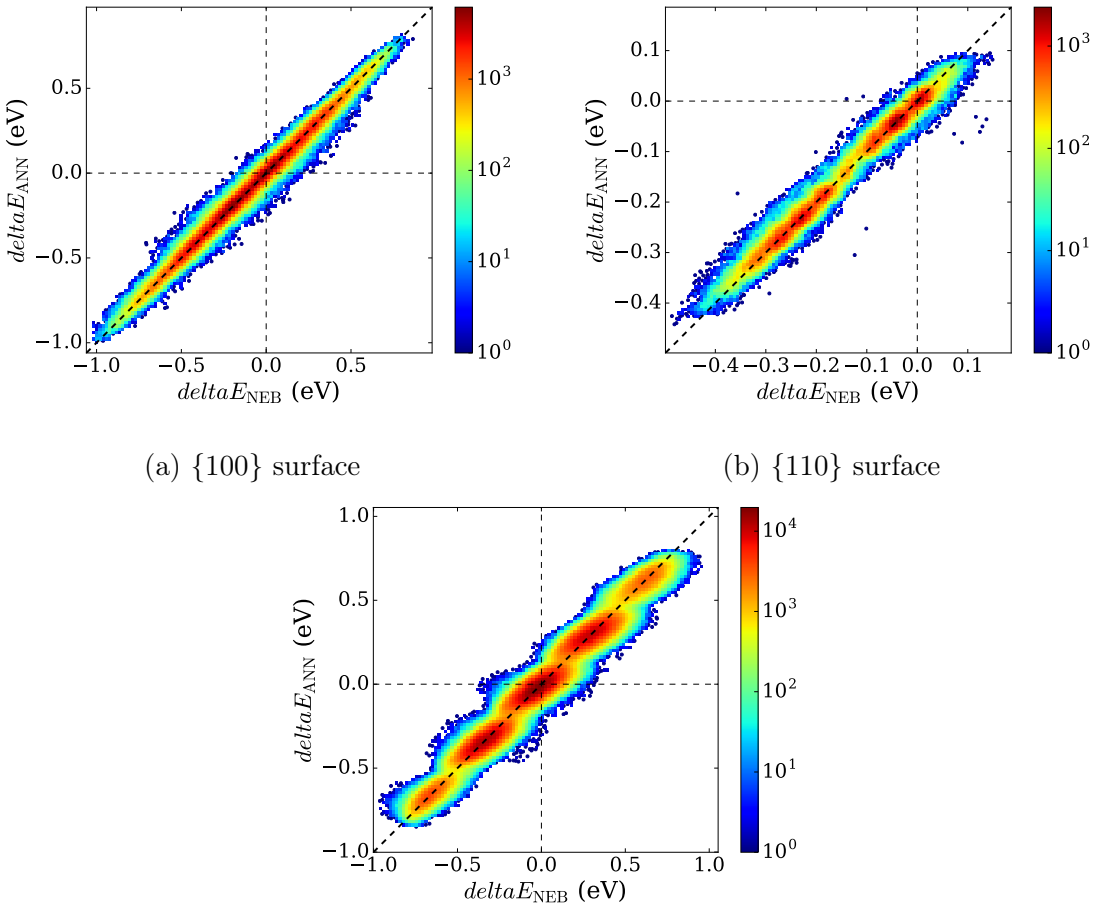
(a) {100} surface

(b) {110} surface



(c) {111} surface

Figure 17: Correlation of the NEB calculated barriers in the training set and the barriers predicted by the surface jump networks that were used in the KMC simulations. The value on the horizontal axis shows the NEB calculated barrier value, and the value on the vertical axis shows the barrier value predicted by the neural network. The colours show the density of points in logarithmic scale. Note that there are differences in the colourmaps, because the datasets on different surfaces contain a different number of jumps. The black dashed line marks the identity line $x = y$. The RMS errors of the networks are 0.024 eV on the {100} surface, 0.012 eV on the {110} surface and 0.045 eV on the {111} surface.

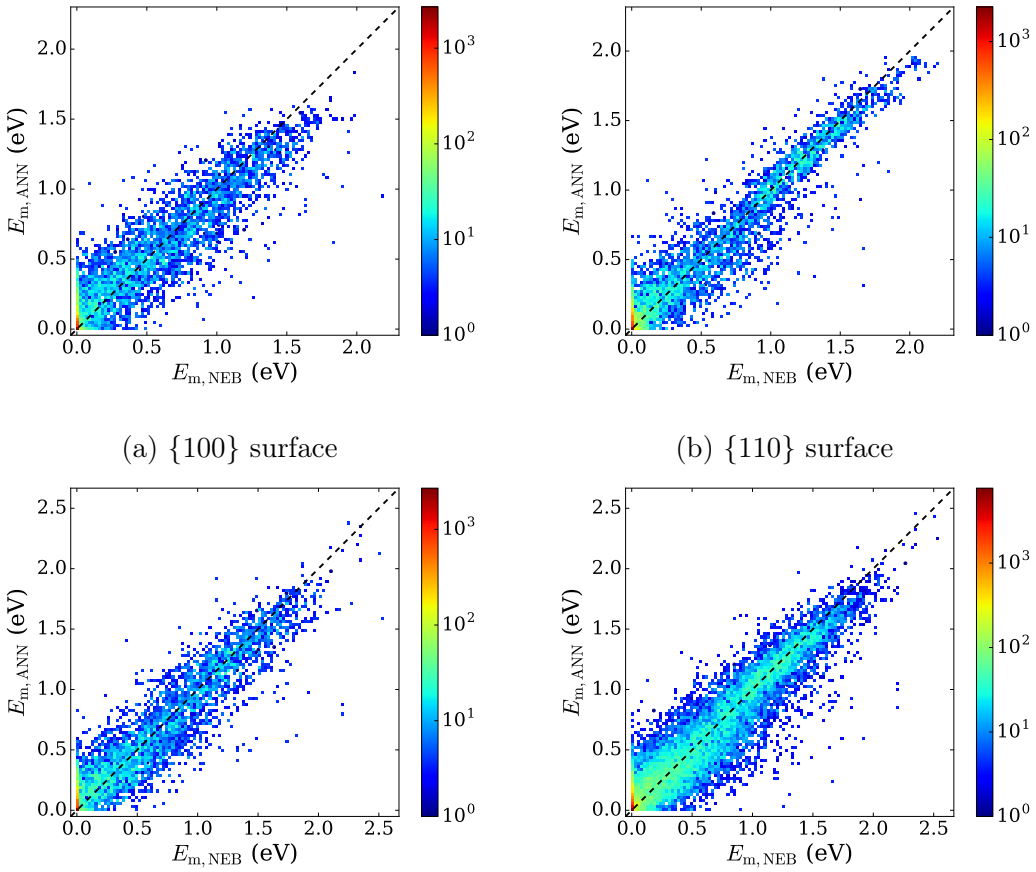


(a) {100} surface

(b) {110} surface

(c) {111} surface

Figure 18: Correlation of the energy change of the jump (ΔE) as calculated from the NEB results and ANN predictions. The same neural networks that were used in the KMC simulations for prediction of surface jump barriers. The NEB calculated ΔE value is on the horizontal axis, and the ANN value is on the vertical axis. The colours show the density of points in logarithmic scale. Note that there are differences in the colourmaps, because the datasets on different surfaces contain a different number of jumps. The black dashed lines mark the identity line $x = y$ and the zeros $x = 0$ and $y = 0$.



(a) {100} surface

(b) {110} surface

(c) {111} surface

(d) All surfaces

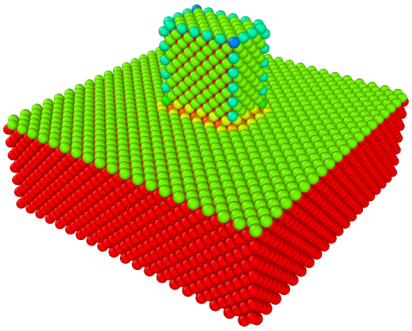
Figure 19: Correlation of the NEB calculated barriers in the test set and the barriers predicted by the surface breaking networks. These networks were chosen to be used in KMC simulations. The value on the horizontal axis shows the NEB calculated barrier value, and the value on the vertical axis shows the barrier value predicted by the neural network. The colours show the density of points in logarithmic scale. Note that there are differences in the colourmaps, because the datasets on different surfaces contain a different number of jumps. The black dashed line marks the identity line $x = y$. The RMS errors of the networks are 0.18 eV on the {100} surface, 0.15 eV on the {110} surface, 0.18 eV on the {111} surface, and 0.17 eV for the network for all three surfaces.

7 KMC simulations using the 72D descriptor and artificial neural networks

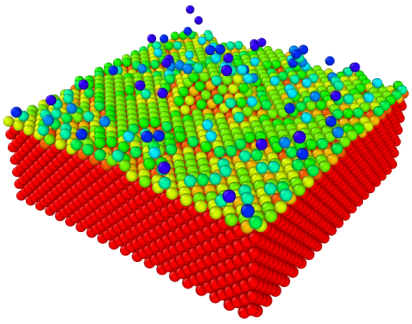
The trained cascade networks were used to simulate the flattening of cuboid Cu nanotips in 1000 K on {100}, {110}, and {111} Cu surfaces. It should be noted, that the aim of the simulations was to compare the behaviour of surface atoms to the KMC results by Kimari *et al.*, not to study the flattening process itself. In the first runs, only the networks trained with surface jump barriers were used. In addition to the three regressor networks, a classifier network was trained. The classifier network was used for classifying the possible jumps to different surfaces, so that the associated barrier could be requested from the correct network. The simulations were carried out using the KMC code Kimocs [81]. Periodic boundary conditions were used on the sides of the simulated system. The height of the nanotip was 12 atomic monolayers. If individual atoms or clusters of atoms became detached from the bulk during the simulation, they were considered to be evaporated and they were removed from the simulation. The simulation was stopped when the tip had flattened to half of its original height. Ten cases were run on all three surfaces to collect statistics. The initial states and example final states of the systems can be seen in figure 20.

During the simulation, the tips flattened as was expected, but not as a result of diffusion transporting the atoms down from the tip. Instead, the flattening was due to evaporation of atoms, which was caused by an artefact in the simulation. Direct evaporation, where an atom jumps into a lattice position without any first nearest neighbours was not allowed to happen during the runs, but evaporation in groups was still possible. For example, atom A could first jump up from the surface to an adatom site. Then atom B could jump from the surface to a site on top of atom A. The resulting cluster of two atoms would then be detached from the bulk and removed from the simulation. This is an obviously unphysical process, which probably happens because the networks which were trained with only surface jumps underestimate the energy barriers of surface breaking jumps.

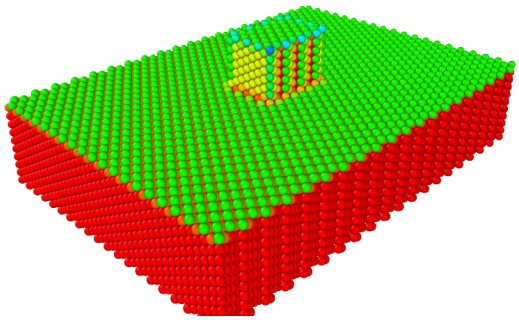
The flattening simulations were repeated using a different set of prediction networks. The same surface jump networks were still used, and a surface breaking network was added to the set. A classifier of four classes was trained to identify surface jumps on the



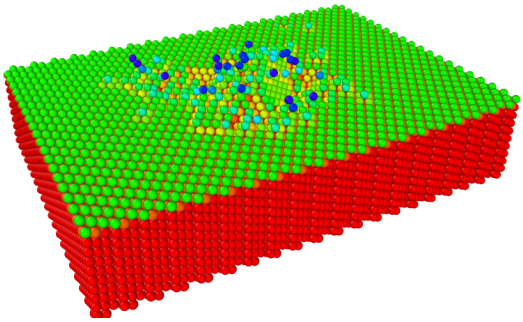
(a) {100} initial



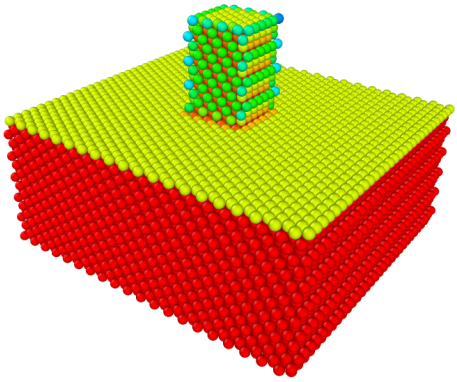
(b) {100} final



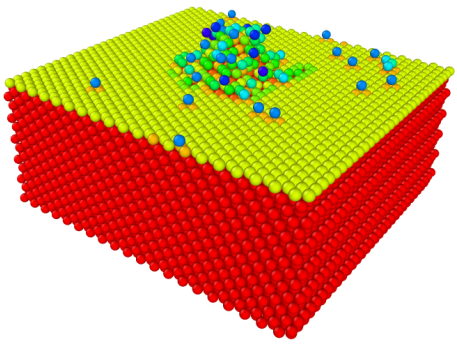
(c) {110} initial



(d) {110} final

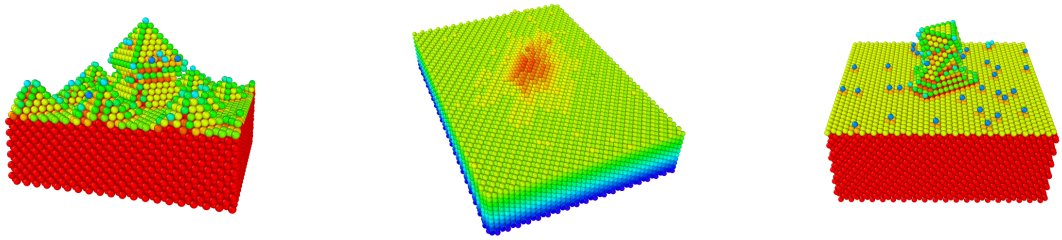


(e) {111} initial



(f) {111} final

Figure 20: Initial and final states of nanotip flattening simulations in KMC. The atoms are coloured according to the number of first nearest neighbours that they have. In bulk the atoms have all 12 first nearest neighbours, and they are coloured red. On the {100} and {110} surfaces, the surface atoms have 8 neighbours and they are green. On {111} the surface atoms are yellow, since they have 9 first nearest neighbours. Only the surface diffusion networks were used to run the simulations, whereas the surface breaking networks were left out.



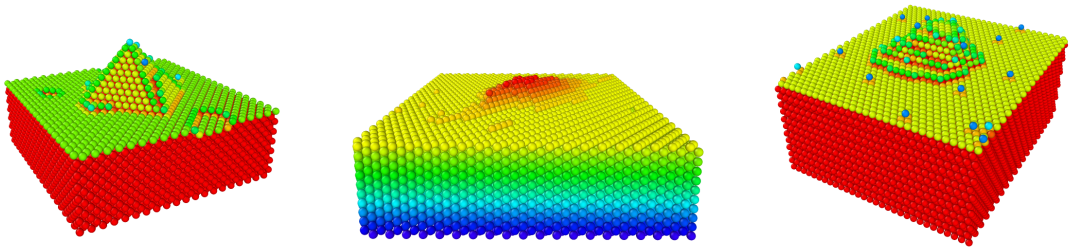
(a) {100} final (b) {110} final (c) {111} final

Figure 21: Final states of nanotip flattening simulations in KMC. The atoms in the {100} and {111} systems are coloured according to the number of first nearest neighbours that they have. In bulk the atoms have all 12 first nearest neighbours, and they are coloured red. On the {100} surface, the surface atoms have 8 neighbours and they are green. On {111} the surface atoms are yellow, since they have 9 first nearest neighbours. The atoms in the {110} system are coloured according to their z coordinate to better show the shape of the flattened tip. Three surface diffusion networks and one network for all surface breaking processes were used in these simulations.

{100}, {110}, and {111} surfaces as well as surface breaking jumps. The final states of the simulated systems can be seen in figure 21. This time we see that the tip on the {110} surface has succesfully flattened to half of its original height. Closer inspection of the simulation revealed that there was no artificial evaporation, and the flattening was due to diffusion as it is supposed to be. This is an improvement compared to our previous simulations, and also different from Kimari's work, where the {110} surface often became unstable in the flattening simulations [13]. Since the only thing that was changed from our previous simulations was the addition of a surface breaking network, we can see that the underestimation of the barriers of surface breaking processes was the cause of the artificial evaporation. However, now the other two surfaces show problematic behaviour. On the {100} and {111} surfaces the tips did not start flattening, but instead formed {111} facets. In addition, small peaks start growing from the {100} surface. A closer examination revealed that the peaks form because the atoms tend to jump on top of each other. The process is related to the evaporation problem in the previous KMC runs, but less pronounced, since now the atoms did not evaporate.

The final KMC simulations were run using six prediction networks: three networks for surface jumps on the {100}, {110}, and {111} surfaces, and another three networks

for surface breaking jumps on these three surfaces. A classifier was trained and used to divide the jumps to these six classes. The results of the KMC simulations can be seen in figure 22. Now the tips on both {110} and {111} surfaces have flattened as they are expected to. Moreover, small peaks are no longer forming on the {100} surface, which is an improvement. Unfortunately, the tip still did not flatten, and instead changed its shape to a pyramid with {111} facets on the sides. All in all, this combination of networks seems to still overestimate the stability of the {111} surface at the expense of the {100} surface. The pyramid also forms quite quickly during the simulation and then keeps its shape, whereas in reality the atoms on the edges should be quite likely to jump away from their unstable positions. Thus the barriers related to these jumps appear to be overestimated. This may well be the case, since jumps on the edges of two surfaces did not appear in the surface jump training sets, and it is unlikely that there would be such jumps in the randomly generated surface breaking training sets either. Since the networks have not seen such jumps during the training, it is plausible that their barriers are wrongly predicted.



(a) {100} final (b) {110} final (c) {111} final

Figure 22: Final states of nanotip flattening simulations in KMC. The atoms in the {100} and {111} systems are coloured according to the number of first nearest neighbours that they have. In bulk the atoms have all 12 first nearest neighbours, and they are coloured red. On the {100} surface, the surface atoms have 8 neighbours and they are green. On {111} the surface atoms are yellow, since they have 9 first nearest neighbours. The atoms in the {110} system are coloured according to their z coordinate to better show the shape of the flattened tip. Six neural networks in total were used in these simulations: three for surface diffusion processes on the {100}, {110}, and {111} surfaces, and three for surface breaking processes on the three surfaces.

8 Conclusions

The use of artificial neural networks for barrier prediction in KMC was studied. In particular, the existing model for the parametrisation of atomic jumps, where the local atomic environment includes the 1st and 2nd nearest neighbours of the jump, was expanded to include atoms up to the 4th nearest neighbours. The number of atoms in the descriptor thus increased from 26 to 72. Barrier values were calculated and compared to see if changing the 3rd and 4th nearest neighbours has a significant effect on the barriers. This was found to be the case especially for jumps on the {110} and {111} surfaces. This means that with the expanded 72D local atomic environment descriptor, more detailed data can be given the neural networks for training. As a result, the prediction accuracy of the networks can improve.

The calculated barriers were then used to train artificial neural networks. The barrier prediction accuracy of the networks was assessed, as well as the computational cost of training them. The accuracy depended strongly on the network architecture, and cascade correlation networks were found to be more accurate than multilayer perceptrons. The cascade networks were also faster to train. The computational cost depended mostly on the amount of training data, because calculating the barriers was relatively expensive computationally. However, it was found that using a training set of only 20 % of the calculated barriers instead of 80 % did not worsen the predictions. In future training tasks with the 72D descriptor it is thus possible to try even smaller training sets, which helps keep the computational cost of training feasible. Finally, the prediction accuracy of the networks that were trained with barriers of surface jumps was better than in earlier work, where the 26D descriptor was used.

Two neural networks were trained for each used surface orientation: one with barriers of surface jumps, where the jump happens along the surface in the adatom layer, and one with barriers of surface breaking jumps, where the atom jumps from the surface layer to the adatom layer. In addition, one network was trained with barriers of surface breaking jumps on all surfaces. The used surface orientations were {100}, {110}, and {111}. The neural networks were then used in KMC simulations of nanotip flattening in different combinations. Using only the three surface networks for barrier prediction resulted in artificial, unphysical evaporation of atoms. Adding a fourth network for all surface breaking jumps removed the evaporations, and allowed the tip on the {110} surface

to flatten in the expected manner. However, the tips on the {100} and {111} surfaces did not flatten. When three surface networks and three surface breaking networks were used, the tips on the {110} and {111} surfaces flattened, but the tip on the {100} surface only changed its shape to a pyramid.

Considering that the analysis of the calculated barriers showed good potential for improvement and that the prediction accuracy of the networks was promising, it is rather disappointing that the KMC results were partly unphysical. It seems that the surface jump datasets were too homogeneous after all, and as a result the trained networks gave inaccurate predictions for jumps that were very different from what they had encountered in the training set. This is supported by the fact that adding separate networks for surface breaking processes alleviated the problem, because that made it possible to identify some jumps where the surface networks could not give good predictions. However, there are still more challenges, as the pyramid on the {100} surface shows. In reality, the atoms on the edges are in energetically unfavourable positions, and should be prone to jumping away from there, but this did not happen in the simulations. These kinds of jumps were not included in either the surface or surface breaking dataset, and so their barriers were probably overestimated.

If the 72D descriptor is used in future works, good care should be taken to make sure that all kinds of jumps are well represented in the training set. If some kinds of processes are left out because they are considered improbable, the network may severely underestimate their barriers, which in turn makes them probable in the KMC simulations. In this work, separate networks were used for surface jumps and surface breaking jumps, but it was not tested whether it is necessary. If more jumps of different classes are added to the training data, and different networks are used for each class, it may lead to an impractical number of predictor networks. Therefore one should rather check if barriers of many kinds of jumps could be predicted well with a single network instead. Furthermore, the selection of the training data could be optimized with active learning methods. In active learning, the learning algorithm can ask for new training data where its predictions are uncertain. This way fewer data points are needed for learning [89]. In our case, uncertain barriers could be calculated with NEB during the KMC run as they are encountered, and the neural network could be retrained before it makes the next prediction.

The training barriers for the networks were calculated without an electric field, and so the trained networks cannot predict how the field would affect the barriers. To be able to simulate the metal tip under an electric field, the effect of the electric field must be accounted for in some other manner. An attractive way to do this is to use an analytical formula like Jansson *et al.* [72] and Kyritsakis *et al.* [74] did, as described in section 3.3. In future works, the barrier under zero field could be predicted using neural networks, and the barrier change due to local electric field could be calculated using the formula. Another option would be to include barriers calculated under an electric field to the training set or train a separate network to predict the barrier change, but these methods are unnecessarily complicated compared to fitting and using an analytical formula.

All in all, the expanded 72D descriptor can describe the different diffusion jumps in more detail than the 26D descriptor. The neural networks also seem capable of learning the barriers from the more detailed training data. With proper selection of the jumps in the training set, using the 72D descriptor with artificial neural networks has the potential to improve KMC simulations of diffusion.

References

[1] Butler Keith T., Davies Daniel W., Cartwright Hugh, Isayev Olexandr, and Walsh Aron, "Machine learning for molecular and materials science," *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.

[2] F. Djurabekova, R. Domingos, G. Cerchiara, N. Castin, E. Vincent, and L. Malerba, "Artificial intelligence applied to atomistic kinetic Monte Carlo simulations in Fe–Cu alloys," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 255, no. 1, pp. 8 – 12, 2007. Computer Simulation of Radiation Effects in Solids.

[3] N. Castin and L. Malerba, "Prediction of point-defect migration energy barriers in alloys using artificial intelligence for atomistic kinetic Monte Carlo applications," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 267, no. 18, pp. 3148 – 3151, 2009. Proceedings of the Ninth International Conference on Computer Simulation of Radiation Effects in Solids.

[4] N. Castin and L. Malerba, "Calculation of proper energy barriers for atomistic kinetic Monte Carlo simulations on rigid lattice with chemical and strain field long-range effects using artificial neural networks," *The Journal of Chemical Physics*, vol. 132, no. 7, p. 074507, 2010.

[5] N. Castin, L. Messina, C. Domain, R. C. Pasianot, and P. Olsson, "Improved atomistic Monte Carlo models based on ab-initio-trained neural networks: Application to FeCu and FeCr alloys," *Phys. Rev. B*, vol. 95, p. 214117, Jun 2017.

[6] M. Pascuet, N. Castin, C. Becquart, and L. Malerba, "Stability and mobility of Cu–vacancy clusters in Fe–Cu alloys: A computational study based on the use of artificial neural networks for energy barrier calculations," *Journal of Nuclear Materials*, vol. 412, no. 1, pp. 106 – 115, 2011.

[7] L. Messina, N. Castin, C. Domain, and P. Olsson, "Introducing ab initio based neural networks for transition-rate prediction in kinetic Monte Carlo simulations," *Phys. Rev. B*, vol. 95, p. 064112, Feb 2017.

[8] W. M. Young and E. W. Elcock, "Monte Carlo studies of vacancy migration in binary ordered alloys: I," *Proceedings of the Physical Society*, vol. 89, pp. 735–746, nov 1966.

[9] A. Bortz, M. Kalos, and J. Lebowitz, "A new algorithm for Monte Carlo simulation of Ising spin systems," *Journal of Computational Physics*, vol. 17, no. 1, pp. 10 – 18, 1975.

[10] P. Lebrun, L. Linssen, A. Lucaci-Timoce, D. Schulte, F. Simon, S. Stapnes, N. Toge, H. Weerts, and J. Wells, *The CLIC programme: Towards a staged e^+e^- linear collider exploring the terascale: CLIC conceptual design report.* CERN Yellow Reports: Monographs, Geneva: CERN, 2012. Comments: 84 pages, published as CERN Yellow Report https://cdsweb.cern.ch/record/1475225.

[11] A. Descoeudres, Y. Levinsen, S. Calatroni, M. Taborelli, and W. Wuensch, "Investigation of the dc vacuum breakdown mechanism," *Phys. Rev. ST Accel. Beams*, vol. 12, p. 092001, Sep 2009. https://doi.org/10.1103/PhysRevSTAB.12.092001.

[12] C. Antoine, F. Peauger, and F. L. Pimpec, "Electromigration occurences and its effects on metallic surfaces submitted to high electromagnetic field: A novel approach to breakdown in accelerators," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 665, pp. 54 – 69, 2011.

[13] J. Kimari, V. Jansson, S. Vigonski, E. Baibuz, D. R., V. Zadin, and F. Djurabekova, "Application of artificial neural networks for rigid lattice kinetic Monte Carlo studies of Cu surface diffusion." https://arxiv.org/abs/1806.02976, April 2020.

[14] H. Jonsson, G. Mills, and K. W. Jacobsen, *Nudged elastic band method for finding minimum energy paths of transitions*, pp. 385–404. World Scientific Publishing Co. Pte. Ltd., 06 1998.

[15] N. J. Nilsson, *The Quest for Artificial Intelligence.* Cambridge University Press, 2009.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*

25 (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[17] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, Nov 2012.

[18] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *Journal of Network and Computer Applications*, vol. 68, pp. 90 – 113, 2016.

[19] Esteva Andre, Kuprel Brett, Novoa Roberto A., Ko Justin, Swetter Susan M., Blau Helen M., and Thrun Sebastian, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[20] Silver David, Huang Aja, Maddison Chris J., Guez Arthur, Sifre Laurent, van den Driessche George, Schrittwieser Julian, Antonoglou Ioannis, Panneershelvam Veda, Lanctot Marc, Dieleman Sander, Grewe Dominik, Nham John, Kalchbrenner Nal, Sutskever Ilya, Lillicrap Timothy, Leach Madeleine, Kavukcuoglu Koray, Graepel Thore, and Hassabis Demis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] Silver David, Schrittwieser Julian, Simonyan Karen, Antonoglou Ioannis, Huang Aja, Guez Arthur, Hubert Thomas, Baker Lucas, Lai Matthew, Bolton Adrian, Chen Yutian, Lillicrap Timothy, , Hui Fan, S. van den Driessche George, Graepel Thore, and Hassabis Demis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[22] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, pp. 433–460, 10 1950.

[23] S. Russell and P. Norvig, *Artificial Intelligence: a Modern Approach*. Prentice Hall, 3 ed., 2010.

[24] M. Kubat, *An Introduction to Machine Learning*. Springer International Publishing AG, 2017.

[25] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[26] S. Gražulis, A. Daškevič, A. Merkys, D. Chateigner, L. Lutterotti, M. Quirós, N. R. Serebryanaya, P. Moeck, R. T. Downs, and A. Le Bail, "Crystallography open database (COD): an open-access collection of crystal structures and platform for world-wide collaboration," *Nucleic Acids Research*, vol. 40, no. D1, pp. D420–D427, 2012. http://crystallography.net/cod/ (visited 14th January 2020).

[27] "MatNavi." Website. https://mits.nims.go.jp/index_en.html (Accessed 14th January 2020).

[28] S. Curtarolo, W. Setyawan, G. L. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. J. Mehl, H. T. Stokes, D. O. Demchenko, and D. Morgan, "AFLOW: An automatic framework for high-throughput materials discovery," *Computational Materials Science*, vol. 58, pp. 218 – 226, 2012. http://aflowlib.org/ (visited 15th January 2020).

[29] "NREL materials database." Website. https://materials.nrel.gov/ (Accessed 15th January 2020).

[30] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Phys. Rev.*, vol. 136, pp. B864–B871, Nov 1964.

[31] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Phys. Rev.*, vol. 140, pp. A1133–A1138, Nov 1965.

[32] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, "Bypassing the Kohn-Sham equations with machine learning," *Nature Communications*, vol. 8, pp. 1–10, 10 2017. Copyright - © 2017. This work is published under http://creativecommons.org/licenses/by/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2019-09-27.

[33] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. Oxford University Press, 1987.

[34] J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost," *Chem. Sci.*, vol. 8, pp. 3192–3203, 2017.

[35] A. P. Bartók, J. Kermode, N. Bernstein, and G. Csányi, "Machine learning a general-purpose interatomic potential for silicon," *Phys. Rev. X*, vol. 8, p. 041048, Dec 2018.

[36] D. R. Cassar, A. C. de Carvalho, and E. D. Zanotto, "Predicting glass transition temperatures using neural networks," *Acta Materialia*, vol. 159, pp. 249 – 256, 2018.

[37] F. A. Faber, A. Lindmaa, O. A. von Lilienfeld, and R. Armiento, "Machine learning energies of 2 million elpasolite (ABC_2D_6) crystals," *Phys. Rev. Lett.*, vol. 117, p. 135502, Sep 2016.

[38] N. Kiselyova, V. Gladun, and N. Vashchenko, "Computational materials design using artificial intelligence methods," *Journal of Alloys and Compounds*, vol. 279, no. 1, pp. 8 – 13, 1998.

[39] D. P. Tabor, L. M. Roch, S. K. Saikin, C. Kreisbeck, D. Sheberla, J. H. Montoya, S. Dwaraknath, M. Aykol, C. Ortiz, H. Tribukait, C. Amador-Bedolla, C. J. Brabec, B. Maruyama, K. A. Persson, and A. Aspuru-Guzik, "Accelerating the discovery of materials for clean energy in the era of smart automation," *Nature Reviews.Materials*, vol. 3, pp. 5–20, 05 2018. Copyright - Copyright Nature Publishing Group May 2018; Last updated - 2019-10-04.

[40] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, Dec 1943.

[41] M. H. Hassoun, *Fundamentals of artificial neural networks*. MIT Press, 1995.

[42] R. Rohwer, M. Wynne-Jones, and F. Wysotzki, "Neural networks," in *Machine Learning, neural and statistical classification* (D. Michie, D. J. Spiegelhalter, and C. C. Taylor, eds.), pp. 84–124, Ellis Horwood Limited, 1994.

[43] S. Haykin, *Neural networks : a comprehensive foundation*. Prentice Hall, 2nd ed., 1999.

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[45] LeCun Yann, Bengio Yoshua, and Hinton Geoffrey, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[46] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[48] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, March 1993.

[49] C. Igel and M. Hüsken, "Improving the Rprop learning algorithm," in *Proceedings of the Second International Symposium on Neural Computation, NC'2000*, pp. 115–121, ICSC Academic Press, 2000.

[50] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, ed.), pp. 524–532, Morgan-Kaufmann, 1990.

[51] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, Dec 1989.

[52] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359 – 366, 1989.

[53] G. P. Drago and S. Ridella, "Convergence properties of cascade correlation in function approximation," *Neural Computing & Applications*, vol. 2, pp. 142–147, Sep 1994.

[54] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[55] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[56] P. Burrows, N. C. Lasheras, L. Linssen, M. Petric, A. Robson, D. Schulte, E. Sicking, S. Stapne, *et al.*, *The Compact Linear Collider (CLIC) – 2018 Summary Report*. CERN Yellow Reports: Monographs, Geneva: CERN, 2018.

[57] A. V. Crewe, "Scanning electron microscopes: Is high resolution possible?," *Science*, vol. 154, no. 3750, pp. 729–738, 1966.

[58] P. Liu, Y. Wei, K. Liu, L. Liu, K. Jiang, and S. Fan, "New-type planar field emission display with superaligned carbon nanotube yarn emitter," *Nano Letters*, vol. 12, no. 5, pp. 2391–2396, 2012. PMID: 22494219.

[59] J. Han, J. S. Oh, and M. Meyyappan, "Cofabrication of vacuum field emission transistor (VFET) and MOSFET," *IEEE Transactions on Nanotechnology*, vol. 13, pp. 464–468, May 2014.

[60] D. M. Trucchi and N. A. Melosh, "Electron-emission materials: Advances, applications, and models," *MRS Bulletin*, vol. 42, pp. 488–492, 07 2017. Copyright - Copyright © Materials Research Society 2017; Last updated - 2017-10-06.

[61] M. K. Miller and R. G. Forbes, *Atom-Probe Tomography: The Local Electrode Atom Probe*. Springer US, 2014.

[62] H. Yanagisawa, V. Zadin, K. Kunze, C. Hafner, A. Aabloo, D. E. Kim, M. F. Kling, F. Djurabekova, J. Osterwalder, and W. Wuensch, "Laser-induced asymmetric faceting and growth of a nano-protrusion on a tungsten tip," *APL Photonics*, vol. 1, no. 9, p. 091305, 2016.

[63] H. Abramowicz *et al.*, "Higgs physics at the CLIC electron–positron linear collider," *The European Physical Journal C*, vol. 77, p. 475, Jul 2017.

[64] The CLICdp collaboration, H. Abramowicz, *et al.*, "Top-quark physics at the CLIC electron-positron linear collider," *Journal of High Energy Physics*, vol. 2019, p. 3, Nov 2019.

[65] H. Toijala, K. Eimre, A. Kyritsakis, V. Zadin, and F. Djurabekova, "Ab initio calculation of field emission from metal surfaces with atomic-scale defects," *Phys. Rev. B*, vol. 100, p. 165421, Oct 2019. https://doi.org/10.1103/PhysRevB.100.165421.

[66] A. Degiovanni, W. Wuensch, and J. Giner Navarro, "Comparison of the conditioning of high gradient accelerating structures," *Phys. Rev. Accel. Beams*, vol. 19, p. 032001, Mar 2016.

[67] R. H. Fowler and L. Nordheim, "Electron emission in intense electric fields," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 119, no. 781, pp. 173–181, 1928.

[68] E. L. Murphy and R. H. Good, "Thermionic emission, field emission, and the transition region," *Phys. Rev.*, vol. 102, pp. 1464–1473, Jun 1956.

[69] R. G. Forbes, "Extraction of emission parameters for large-area field emitters, using a technically complete Fowler–Nordheim-type equation," *Nanotechnology*, vol. 23, p. 095706, feb 2012.

[70] H. Mehrer, *Diffusion in Solids: Fundamentals, Methods, Materials, Diffusion-Controlled Processes*. Springer-Verlag Berlin Heidelberg, 2007.

[71] M. J. Harvey and G. D. Fabritiis, "High-throughput molecular dynamics: the powerful new tool for drug discovery," *Drug Discovery Today*, vol. 17, no. 19, pp. 1059 – 1062, 2012.

[72] V. Jansson, E. Baibuz, A. Kyritsakis, and F. Djurabekova, "Adatom diffusion in high electric fields," in *2017 30th International Vacuum Nanoelectronics Conference (IVNC)*, pp. 40–41, July 2017.

[73] T. T. Tsong and G. Kellogg, "Direct observation of the directional walk of single adatoms and the adatom polarizability," *Phys. Rev. B*, vol. 12, pp. 1343–1353, Aug 1975.

[74] A. Kyritsakis, E. Baibuz, V. Jansson, and F. Djurabekova, "Atomistic behavior of metal surfaces under high electric fields," *Phys. Rev. B*, vol. 99, p. 205418, May 2019.

[75] V. Jansson, E. Baibuz, A. Kyritsakis, S. Vigonski, V. Zadin, S. Parviainen, A. Aabloo, and F. Djurabekova, "Growth mechanism for nanotips in high electric fields." https://arxiv.org/abs/1909.05825, 9 2019.

[76] Fu Chu-Chun, Torre Jacques Dalla, Willaime François, Bocquet Jean-Louis, and Barbu Alain, "Multiscale modelling of defect kinetics in irradiated iron," *Nature Materials*, vol. 4, no. 1, pp. 68–74, 2005.

[77] M. Prieto-Depedro, I. Martin-Bragado, and J. Segurado, "An atomistically informed kinetic Monte Carlo model of grain boundary motion coupled to shear deformation," *International Journal of Plasticity*, vol. 68, pp. 98 – 110, 2015.

[78] K. Reuter, D. Frenkel, and M. Scheffler, "The steady state of heterogeneous catalysis, studied by first-principles statistical mechanics," *Phys. Rev. Lett.*, vol. 93, p. 116105, Sep 2004.

[79] M. Andersen, C. Panosetti, and K. Reuter, "A practical guide to surface kinetic Monte Carlo simulations," *Frontiers in Chemistry*, vol. 7, p. 202, 2019.

[80] K. J. Laidler, "A glossary of terms used in chemical kinetics, including reaction dynamics (iupac recommendations 1996)," *Pure and Applied chemistry*, vol. 68, pp. 149–192, 1996.

[81] V. Jansson, E. Baibuz, and F. Djurabekova, "Long-term stability of Cu surface nanotips," *Nanotechnology*, vol. 27, p. 265708, may 2016.

[82] F. Soisson and C.-C. Fu, "Cu-precipitation kinetics in $\alpha-$Fe from atomistic simulations: Vacancy-trapping effects and Cu-cluster mobility," *Phys. Rev. B*, vol. 76, p. 214102, Dec 2007.

[83] G. Henkelman, B. P. Uberuaga, and H. Jónsson, "A climbing image nudged elastic band method for finding saddle points and minimum energy paths," *The Journal of Chemical Physics*, vol. 113, no. 22, pp. 9901–9904, 2000.

[84] L. Prechelt, "Investigation of the CasCor family of learning algorithms," *Neural Networks*, vol. 10, no. 5, pp. 885 – 896, 1997.

[85] S. Nissen, "Implementation of a fast artificial neural network library (fann)," tech. rep., Department of Computer Science University of Copenhagen (DIKU), 1993. http://fann.sf.net (Accessed 4th July 2019).

[86] S. Nissen, "Fann reference manual." http://leenissen.dk/fann/html/files/fann-h.html (Accessed 9th January 2020).

[87] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1 – 19, 1995.

[88] M. S. Stave, D. E. Sanders, T. J. Raeker, and A. E. DePristo, "Corrected effective medium method. v. simplifications for molecular dynamics and Monte Carlo simulations," *The Journal of Chemical Physics*, vol. 93, no. 6, pp. 4413–4426, 1990.

[89] B. Settles, "Active learning literature survey," tech. rep., University of Wisconsin-Madison, Department of Computer Sciences, 2009.