Journal of Physics: Condensed Matter

PAPER

Parameter estimation for pattern formation induced by ion bombardment of solid surfaces using deep learning

To cite this article: Kevin M Loew and R Mark Bradley 2020 *J. Phys.: Condens. Matter* **33** 025901

IOP Publishing Journal of Physics: Condensed Matter
J. Phys.: Condens. Matter **33** (2020) 025901 (12pp) https://doi.org/10.1088/1361-648X/abb996

Parameter estimation for pattern formation induced by ion bombardment of solid surfaces using deep learning

Kevin M Loew[1] **and R Mark Bradley**[2,*]

[1] Department of Physics, Colorado State University, Fort Collins, CO 80523, United States of America
[2] Departments of Physics and Mathematics, Colorado State University, Fort Collins, CO 80523, United States of America

E-mail: mark.bradley@colostate.edu

Received 25 June 2020, revised 1 September 2020
Accepted for publication 17 September 2020
Published 15 October 2020

Abstract

The nanostructures produced by oblique-incidence broad beam ion bombardment of a solid surface are usually modelled by the anisotropic Kuramoto–Sivashinsky equation. This equation has five parameters, each of which depend on the target material and the ion species, energy, and angle of incidence. We have developed a deep learning model that uses a single image of the surface to estimate all five parameters in the equation of motion with root-mean-square errors that are under 3% of the parameter ranges used for training. This provides a tool that will allow experimentalists to quickly ascertain the parameters for a given sputtering experiment. It could also provide an independent check on other methods of estimating parameters such as atomistic simulations combined with the crater function formalism.

Keywords: ion sputtering, nanostructures, pattern formation, deep learning

(Some figures may appear in colour only in the online journal)

1. Introduction

Nominally flat solid surfaces bombarded by obliquely incident, broad ion beams often self-organize into nanoscale ripple structures [1, 2]. These ripples are a result of a surface instability. Consider an elemental target material below its recrystallization temperature that is bombarded with noble gas ions. The incident ions are to have an energy in the range between roughly 0.2 and 20 keV. After transforming to an appropriate moving frame of reference, the equation of motion (EoM) becomes the anisotropic Kuramoto–Sivashinsky (aKS) equation

$$u_t = \kappa_1 u_{xx} + \kappa_2 u_{yy} - B\nabla^2\nabla^2 u + \lambda_1 u_x^2 + \lambda_2 u_y^2, \qquad (1)$$

where $u(x, y, t)$ is the height of the surface above the point (x, y) in the x–y plane at time t, the subscripts x, y, and t on u denote partial derivatives, and $\nabla^2 \equiv \partial_x^2 + \partial_y^2$ [3, 4]. According to the Bradley–Harper theory [5], the second order derivative terms u_{xx} and u_{yy} are produced by the curvature dependence of the sputter yield, although it is now known that ion-induced mass redistribution [6–8], ion-induced viscous flow [2, 9–11] and ion implantation [12, 13] also contribute to these terms. The smoothing term $-B\nabla^2\nabla^2 u$ comes from surface diffusion or ion-induced viscous flow near the surface [2, 14]. The coefficients κ_1, κ_2, λ_1, λ_2 and B are constants which depend on the target material and the ion species, energy and angle of incidence [3–5, 15–18]. The parameter B also depends on the sample temperature, but the remaining parameters do not. The x axis is oriented so that the angle of incidence θ of the ion beam is nonnegative and the azimuthal angle is zero.

If both κ_1 and κ_2 are positive, there is no surface instability and the surface smooths. This occurs in the case of normal incidence and for θ below a threshold value. If, however, $\kappa_1 < 0$ and $\kappa_1 < \kappa_2$, ripples with wave vector $\vec{k}$ parallel to the x axis emerge; these are called parallel-mode ripples. If $\kappa_2 < \kappa_1$ and

* Author to whom any correspondence should be addressed.

$\kappa_2 < 0$, on the other hand, the selected wave vector is along the y direction and so-called perpendicular-mode ripples form.

Spontaneous pattern formation induced by ion bombardment is not only of academic interest: ion bombardment has the potential to provide a method for fast, versatile, and low-cost fabrication of large-area nanostructures with features smaller than optical lithography can produce.

Multiple studies in which the parameters in the EoM are estimated have been conducted using a variety of techniques. Grazing-incidence small angle x-ray scattering (GISAXS) has been employed to estimate the coefficients κ_1 and κ_2 for semiconductor surfaces bombarded with noble gas ions [19–21]. More recently, GISAXS with a coherent x-ray beam rather than an incoherent one has been used to estimate the coefficients of the leading-order linear and nonlinear terms [22]. Estimates of the parameters have also been obtained for normal-incidence bombardment of silicon with iron co-deposition starting from measurements of physical attributes of the pattern like the characteristic lateral length scale and the pattern's amplitude at long times [23, 24].

The coefficients can also be determined by atomistic simulations combined with the so-called crater function formalism (CFF) [15, 17]. Norris *et al* input the results of molecular dynamics simulations into the CFF to estimate the coefficients κ_1 and κ_2 for irradiation of a silicon target with 100 and 250 eV Ar ions [25]. More recently, Hofsäss and Bobes carried out Monte Carlo simulations using SDTrimSP and input the results into the CFF [26]. This yielded estimates for all of the parameters in the AKS equation (1) for a variety of target materials and ion species.

Deep learning algorithms are a class of machine learning (ML) algorithms that use artificial neural networks (ANNs) to extract important features from raw data [27]. This same technology is at the forefront of advancements in self-driving cars [28], facial recognition [29], and natural language processing [30]. In the case of computer vision problems, ANNs learn to recognize and understand visual input [31]. This suggests that deep learning could be used effectively to recognize the parameters associated with patterns produced by ion sputtering and to predict what the parameters are for a particular ion-target combination given a single atomic force microscope (AFM) scan of the surface. The first steps in this direction have recently been made by Reiser [32]. His work was restricted to normal incidence ion bombardment, however, and the surface of an elemental material simply remains flat when it is bombarded with a normally incident noble gas ion beam. Reiser's method also requires two images of the surface that are taken at times separated by a short time interval. This is impractical if the surface scans must be made *ex-situ*, as is invariably the case.

In this paper, we develop an ANN that is able to estimate the five parameters in the EoM (1) from a single AFM scan of a solid surface that has been bombarded with an obliquely incident ion beam. We trained the network for a range of parameters using images generated by numerical integration of the aKS equation (1). The parameter ranges selected are appropriate for sputtering of a silicon surface with a 1 keV argon ion beam, and the ANN was trained for fluences of 5×10^{17} ions/cm^2 and 1×10^{17} ions/cm^2. For the first fluence, our ANN was able to estimate all five parameters in the EoM with root-mean-square errors less than 3% of the parameter ranges used for training. For the second fluence, it estimated the parameters with root-mean-square errors values under 2% of the parameter training ranges. This demonstrates that our network can reliably predict the parameters and can be trained for various ion fluences.

A key benefit of the tool we have developed is that it could be used to provide a validation check on parameter estimates determined by other means, e.g., GISAXS or atomistic simulations combined with the CFF. Just like estimates based on GISAXS, our tool takes experimental results and analyzes them to estimate the parameters in the EoM. A notable difference is that our method utilizes a single AFM image, and AFMs are widely available. GISAXS, on the other hand, requires the experiment to be conducted at a facility with a synchrotron x-ray beamline.

This paper is organized as follows. A brief look at the principles of training a ML algorithm and a general overview of ANNs are presented in section 2. In section 3, we discuss the particular architecture used for our analysis as well as the production of the dataset used for training and evaluation. A simple test of estimating a single parameter in the EoM is conducted in section 4. In section 5, we generalize the ANN to estimate all five parameters and present the estimates. Next, we consider related work and the previous studies of the KS equation using ML techniques in section 6. Potential extensions and considerations of the utility of the network are addressed in section 7. Our conclusions and final thoughts are given in section 8.

2. Machine learning and convolutional neural networks

Machine learning (ML) is a subset of artificial intelligence in which a system learns to identify patterns in data and report the associated meaning [33]. We leverage this ability to recognize and interpret patterns to analyze the pattern formation observed on ion-sputtered solid surfaces. Supervised ML algorithms learn to associate patterns in data with known 'targets'. The network developed in our study uses supervised learning and the targets are the values of κ_1, κ_2, B, λ_1, and λ_2.

Supervised learning can be further subdivided into classification or regression algorithms. The distinction is that the output for classification models are discrete values or 'classes', whereas the output from a regression model is continuous [33]. While the outputs differ, the training methodology is similar. Since our targets κ_1, κ_2, B, λ_1, and λ_2 are continuous variables, regression methods were applied to our problem.

Supervised training of ML models involves providing the specified learning algorithm with well understood training data. This data has a known target which the algorithm is used to identify. By repeatedly presenting the model with training data and telling it what the target is, the parameters of the algorithm are adjusted to best align the model output with the

targets. This is done by maximizing a metric, such as accuracy, or minimizing a 'loss function', such as the root-mean-square error (RMSE). In our case, the target is the parameters in the EoM (1) and the mean-square error (MSE) is the loss function.

Numerous ML algorithms that can analyze data and extract meaning exist [33–35]. In our work, the algorithm we employed was an ANN. An ANN consists of layers of interconnected artificial neurons, which we will refer to simply as neurons for the sake of brevity [35]. Each neuron in the network receives input, processes the input, and calculates an output signal. Once a neuron has computed its output signal, it passes it forward to the connected neurons in the next layer. The neuron's output is calculated via a mathematical activation function. The input to the activation function is a weighted sum of the inputs to the neuron and the output is a real number. A linear combination, step function, and sigmoid are examples of commonly employed activation functions [36]. The connections between neurons have associated weights, which are typically randomly assigned at the start but are modified during training to highlight the aspects of the training data that are the most useful for identifying the target. These weights tell the next neuron how important the input from the previous neuron is. Deep learning is a subset of ML which uses ANNs with numerous layers to extract features and aid in prediction of the targets.

ANNs can be broadly divided into three parts: the input layer, the output layer, and the hidden layers. The input layer takes the raw data and processes it for the rest of the model. The output layer is typically a fully connected layer of neurons in which each neuron is connected to every neuron in the preceding layer. The activation function of neurons in the output layer is selected to best match the target: for example, a linear activation function is often employed for regression, while a tanh function is frequently used for binary classification. In our model, we used a linearly activated output layer of neurons in which the number of neurons matched the number of parameters being estimated. Each neuron in this layer output the estimated value of one of parameters in the EoM. The hidden layers are where the bulk of the calculation is done. They allow the model to identify important features of the training data that enable it to provide a good estimate of the target.

We chose to utilize a particular kind of ANN—a convolutional neural network (CNN)—because of their high levels of performance on computer vision problems. CNNs are adept at identifying spatial relationships and have had significant success in practical applications [35, 37]. These networks achieve this by utilizing a specialized linear operation called a convolution. Convolutions allow the network to extract high-level features such as edges to better identify the target. An in-depth explanation of CNNs can be found in chapter 9 of reference [35].

A key benefit of CNNs is that they can accept high dimensional data like an image as input, whereas many ML algorithms require the user to reduce the number of dimensions and determine useful features manually. CNNs take the raw images as input and determine which features of the image are useful in estimating the parameters during the training process. Other algorithms, such as a support vector machine, can do well in image processing but typically require the user to do some preprocessing to determine which features of the image are of interest and then translate that information into a form that the algorithm can use.

Additional details of ML algorithms, and of deep learning models in particular, are beyond the scope of this paper. Further discussion of the learning methods, training methodology, and architecture can be found in references [33–35].[3]

3. The convolutional neural network and dataset generation

Rather than building and training a network from scratch, we chose to use a dense convolutional network (DenseNet) with 121 layers [38]. This architecture is well understood and has been shown to perform remarkably well for computer vision problems, making it an excellent candidate for the analysis of AFM images[4]. An additional benefit of using this structure for the network is that one can obtain a network that is pre-trained. The DenseNet architecture was trained on the ImageNet dataset, which consists of more than 14 million images and 20 000 classes[5]. This means that the network had already learned to extract information from images, a benefit that we repurposed for our problem using transfer learning. Transfer learning takes a network that has already been trained for a similar task and repurposes it for a new problem [39]. This drastically reduces the training time needed and allows a smaller dataset to be used for the training process. We adapted the network by replacing the output layer with a new layer designed for our problem. The layers of the base model were frozen and the output layer and associated connections were trained to understand what to report for the new problem. Once the model had a good 'understanding' of what the output should be, the base layers were unfrozen and the entire model was fine-tuned to optimize the results.

We used PyTorch for our analysis. PyTorch is a software tool that facilitates in the construction, training, and implementation of deep neural networks. This tool is openly available, is commonly used in deep learning research, and has a dedicated team for the support and advancement of the tool itself[6]. We also made use of the fastai library, which provides additional functionality and assists in the training process.

Our dataset was produced via numerical simulations[7]. Equation (1) was numerically integrated using the fourth-order

[3] Tutorials on implementing these algorithms can be found at https://scikit-learn.org/stable/tutorial/index.html, https://pytorch.org/tutorials/, and https://tensorflow.org/tutorials.

[4] The structure of DenseNet-121 is described in table 1 of reference [38]. The key difference between our model and the one presented in reference [38] lies in the final output layer. In our model, this layer is a fully connected five-neuron layer. The activation function was also changed from softmax to linear.

[5] The ImageNet dataset can be found at http://image-net.org/.

[6] We used the PyTorch framework which is documented at https://pytorch.org/. The fast.ai library was also used to facilitate calculations. This library can be found at https://fast.ai/.

[7] Simulated images have been used elsewhere to train ML models. See, for example, reference [40].

Runge–Kutta exponential time-differencing method described by Cox and Matthews [41]. Periodic boundary conditions were applied. The linear terms were calculated exactly in Fourier space and the nonlinear terms were approximated in real space by a central difference finite differencing scheme accurate to fourth order in the grid spacing. For example, the partial derivative of u with respect to x was approximated by

$$\frac{\partial u}{\partial x}(x,y) \approx \frac{u(x-2\Delta x,y) - 8u(x-\Delta x,y) + 8u(x+\Delta x,y) - u(x+2\Delta x,y)}{12\Delta x}, \tag{2}$$

where Δx is the spatial separation of grid points in the x direction [42].[8] Each simulation was started from a low-amplitude white noise initial condition and the parameters κ_1, κ_2, λ_1, λ_2 and B were randomly selected from parameter ranges that will be specified later.

The surface images produced by our simulations are single channel images in which the intensity at each point on the spatial grid is determined by the value of u. The ImageNet dataset is made up of three channel images, however. Therefore, in order to use transfer learning with the ImageNet dataset, we had to convert our single channel simulated images to three channel images. This was done using the default color mapping of Python's matplotlib library[9]. The resulting three channel pseudo-color images were saved as 224×224 pixel images using the .png file type. They were also normalized so that the pixel values for each channel had the same mean and standard deviation as the corresponding channel of the images in the ImageNet dataset. Finally, the normalized three channel images were used to train our network. Conversion of greyscale to pseudo-color images is a practice that has also been employed in ML processing of medical images like chest x-rays [43].

We introduced a 5% probability that a given image was rotated azimuthally before it was input into the CNN. If an image was selected to be rotated, the rotation angle was randomly chosen from the range $[-5°, 5°]$. This rotation was done because normally the AFM image would be produced *ex situ* and the sample could be slightly misaligned when it is positioned in the AFM sample holder.

The produced images were segmented into a training set, a validation set, and a test set. The training set was segmented into batches of 64 images. A single batch was given to the model, and the weights between neurons were updated to increase performance. Another batch was then given to the model and the weights were updated again. This process was repeated until the model had seen all of the images in the training set. The model was then asked to predict the parameter values for all the images in the validation set. By checking the performance on the validation set, we could monitor how much the model had learned and determine how useful the network was for our problem. A complete pass through every image in the training set is called an epoch. After each epoch, the model was validated by comparing the predicted and true values for the validation set. This process of segmenting the training set into batches, showing images to the model, updating weights, and validating was reiterated for 60 epochs to train the model and evaluate its performance during the training process. The model's performance was gauged by determining the MSE between the predicted values and the actual values of the parameters. The network weights were adjusted during training in accordance with the Adam optimization algorithm [44]. Rather than using a fixed learning rate to update the network, we utilized a '1 cycle policy' [45]. This method uses a variable learning rate to help prevent overfitting.

Once training was completed, the final performance of the model was evaluated. The test set had been withheld from the model until this point, which made it suitable for the final test. The model predicted the targets for every image within the test set. These predictions were then compared to the true values and the RMSE was calculated. The RMSE was selected for the final evaluation because of its familiarity to physicists, whereas the MSE was used during training because it stores the same information but is simpler computationally.

4. Estimation of a single parameter

We began by setting all of the parameters in equation (1) to fixed values and only allowing λ_1 to vary. We then trained our ANN to predict the value of λ_1. The coefficients in equation (1) were chosen to be $\kappa_1 = \kappa_2 = -1$, $B = 1$, and $\lambda_2 = 1$, while λ_1 could vary between 0.1 and 5.1. The anisotropy present for $\lambda_1 \neq 1$ gives a tendency for the pattern to be elongated along either the x or y direction. Three examples of simulated surfaces are shown in figure 1. As λ_1 increases, so does the elongation of protrusions and depressions along the x direction. Large values of λ_1 cause significant elongation in the x direction whereas smaller values do not.

Each of the images used for training, validation and testing was taken from a simulation in which λ_1 was randomly selected from the range $0.1 \leqslant \lambda_1 \leqslant 5.1$. All of the images were taken at time $t = 10$; this time was chosen to match previous work [46]. The network was trained on 4000 images and the validation set consisted of 1000 images. After training was

[8] Higher order finite difference schemes can be determined via the tool found at https://web.media.mit.edu/crtaylor/calculator.html.

[9] The default color map in version 2.0 of the matplotlib library is 'viridis'. Detailed information about the coloring mapping can be found at https://matplotlib.org/3.3.0/users/dflt_style_changes.html.

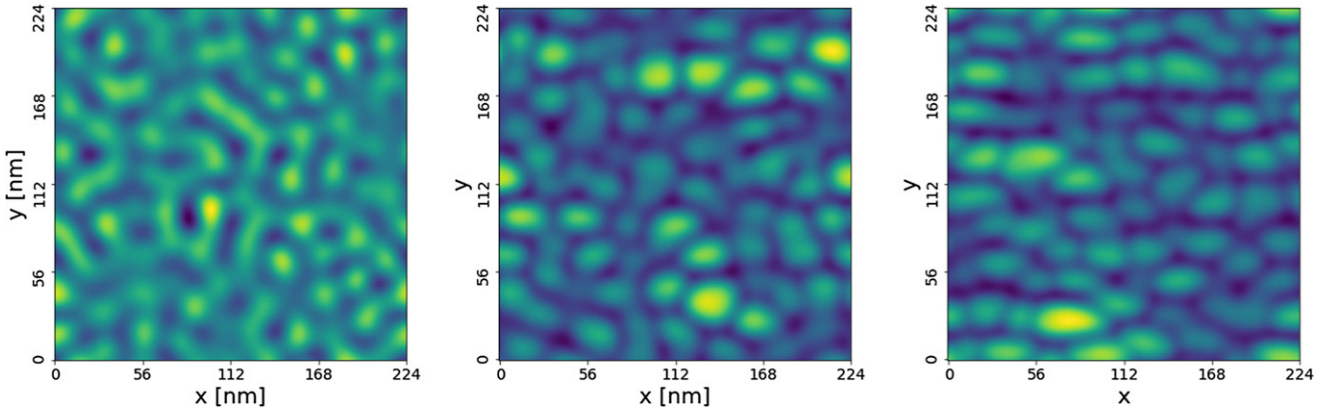

Figure 1. u vs x and y at $t = 10$ for $\lambda_1 = 0.10$ (left), $\lambda_1 = 2.55$ (middle), and $\lambda_1 = 4.79$ (right). The domain size is 224×224.

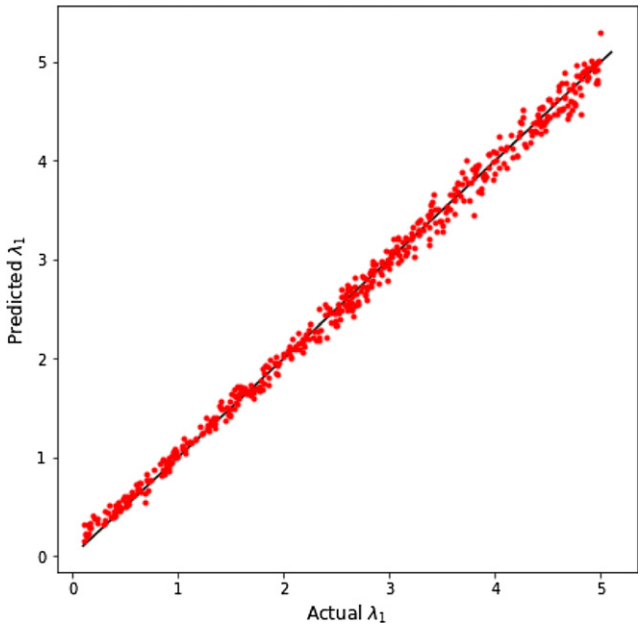

Figure 2. Predicted values of λ_1 vs the actual values. The red points show the actual points and the solid black line shows what perfect agreement would be. The RMSE was 0.107.

completed, the final network was evaluated on a test data set consisting of 500 images.

Training the output layer resulted in a RMSE of 0.193, which is less than 4% of the range of λ_1 values. After unfreezing the hidden layers and fine-tuning the network, the RMSE on the validation set was 0.101, i.e. 2.12% of the range. Lastly, the CNN was given the test set and asked to predict the value of λ_1 for each image. Figure 2 shows that the values predicted by the network agree well with the true values. The final RMSE on the test set was 0.107, which is 2.14% of the range. These results demonstrate that using a single image, the CNN can very effectively estimate the parameter λ_1 in equation (1) for fixed values of κ_1, κ_2, λ_2 and B.

5. Prediction of all of the parameters

As we have seen, the model does quite well at estimating a single parameter. However, in an experiment, all of the parameters in equation (1) have unknown values. As such, it is necessary to train the model to estimate not just one parameter, but five. To achieve this, a training set of 32 000 images, a validation set of 8000 images, and a test set of 5000 images were generated.

For each simulation, the values of the parameters κ_1, κ_2, B, λ_1 and λ_2 were randomly selected from the ranges given in table 1. These ranges were chosen to be approximately twice as wide as the parameter ranges determined in reference [26] for a 1 keV Ar beam incident on Si with an angle of incidence θ between 55° and 80°. (This is the range in which there is a linear instability [26].) To ensure that a surface instability always existed, a simulation was run only if one or both of the randomly generated κ_1 and κ_2 values was negative. If both of these values was positive, a new pair of κ_1 and κ_2 values was randomly chosen.

The images were taken at a time t that corresponds to a fluence of 5×10^{17} ions/cm^2. Most sputtering experiments are run until a fluence in excess of this value is reached. The domain size of the simulations was 500 nm × 500 nm and each surface image was saved as a 224×224 pixel image. It took approximately a week to produce these images on a workstation with an ASUS X299-A motherboard and 78GB of RAM. Any experimental image that is input into our CNN should be a 224×224 pixel image of a 500 nm × 500 nm surface region that has been irradiated to a fluence of 5×10^{17} ions/cm^2.

The CNN used was the same as in section 4 with the exception of the output layer; there were five neurons in the output layer instead of a single one, which allowed the network to return values for the five parameters κ_1, κ_2, λ_1, λ_2 and B. Training the CNN took about a day on a GPU.

Figure 3 shows three of the images used for training. Depending on the values of the parameters, the surface could be similar to the surfaces shown in figure 1, or could display parallel or perpendicular-mode ripples. When only a single parameter is varied, the effect on the simulated surface is fairly easy to recognize, but with five free parameters, each one's impact on the surface dynamics and the observed patterns is significantly harder to identify.

Figure 4 shows the predicted parameter values vs the actual values for the 5000 images in the test set after preliminary training and fine-tuning. Since 5000 data points on a single

Table 1. The ranges of the five parameters in the EoM (1) that were used for training and the RMSE and R^2 evaluation metrics for these parameters for a fluence of 5×10^{17} ions/cm^2.

Parameter	Minimum value	Maximum value	RMSE	R^2
κ_1	-0.12 nm^2 s^{-1}	0.22 nm^2 s^{-1}	0.000 37 nm^2 s^{-1}	0.959
κ_2	-0.12 nm^2 s^{-1}	0.22 nm^2 s^{-1}	0.000 35 nm^2 s^{-1}	0.960
B	0.010 nm^4 s^{-1}	3.96 nm^4 s^{-1}	0.11 nm^4 s^{-1}	0.902
λ_1	-1.47 nm s^{-1}	0.72 nm s^{-1}	0.045 nm s^{-1}	0.889
λ_2	-1.47 nm s^{-1}	0.72 nm s^{-1}	0.039 nm s^{-1}	0.903

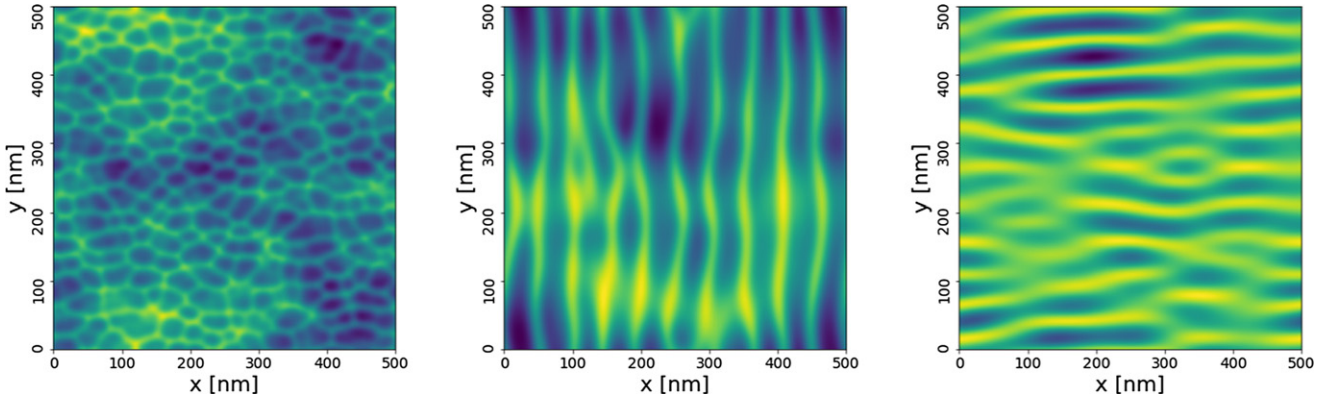

Figure 3. u vs x and y at a fluence of 5×10^{17} ions/cm^2 for $\kappa_1 = -0.098$ nm^2 s^{-1}, $\kappa_2 = -0.094$ nm^2 s^{-1}, $B = 1.169$ nm^4 s^{-1}, $\lambda_1 = -1.075$ nm s^{-1}, and $\lambda_2 = -0.629$ nm s^{-1} (left); $\kappa_1 = -0.107$ nm^2 s^{-1}, $\kappa_2 = 0.128$ nm^2 s^{-1}, $B = 2.893$ nm^4 s^{-1}, $\lambda_1 = -0.152$ nm s^{-1}, and $\lambda_2 = 0.175$ nm s^{-1} (middle); and $\kappa_1 = 0.078$ nm^2 s^{-1}, $\kappa_2 = -0.010$ nm^2 s^{-1}, $B = 2.860$ nm^4 s^{-1}, $\lambda_1 = 0.187$ nm s^{-1}, and $\lambda_2 = -0.035$ nm s^{-1} (right). The domain size is 500 nm $\times$ 500 nm.

plot would be difficult to interpret, we opted to visualize the data via binning. The parameter space was divided into hexagonal bins that were colored according to the number of points within each bin. For each of the five parameters, we observe that the data aligns well with the cyan 1-to-1 agreement line[10]. The best agreement between the values predicted by the CNN and the actual values is seen in the coefficients of the second order linear terms: κ_1 has a RMSE of 0.000 37 nm^2 s^{-1} while κ_2 has a RMSE of 0.000 35 nm^2 s^{-1}; these errors are 0.090% and 0.097% of the range used for training, respectively. The fourth order linear term's coefficient B has a RMSE of 0.11 nm^4 s^{-1}, which is 3.0% of the range. The coefficients of the nonlinear terms λ_1 and λ_2 have RMSEs of 0.045 nm s^{-1} and 0.039 nm s^{-1}, respectively. These RMSE values are 2.1% and 1.8% of the parameter ranges. We conclude that the performance of the CNN was excellent: it learned to predict the five parameters in the EoM (1) to within 3% of the ranges used for training.

Another evaluation metric commonly used in ML is R^2. We also used this metric to gauge the performance of our model. The R^2 values are 0.959, 0.960, 0.902, 0.889, and 0.903 for κ_1, κ_2, B, λ_1, and λ_2, respectively. This further confirms that our model performs well on the test set and accurately predicts the parameters in the EoM. For convenience, the RMSE and R^2 values for the five parameters in the EoM (1) are compiled in table 1.

There is an uncertainty in the value of the fluence in an experiment. Thus, although an experimental image input into our network could have a putative fluence of 5×10^{17} ions/cm^2, the actual value of the fluence could be different. We therefore tested the performance of our model when there could be an error in the reported fluence. To accomplish this, we generated a test set of 500 simulations. The parameters κ_1, κ_2, B, λ_1, and λ_2 for each simulation were randomly selected from the previously specified ranges in the manner we described above. Images were produced every 10^{16} ions/cm^2 from a fluence of 4.5×10^{17} to a fluence of 5.5×10^{17} ions/cm^2 for each of the 500 simulations. The resulting 5500 images were input into our CNN and the parameters values were estimated for each. The RMSE values of the predicted parameter values were 0.000 35 nm^2 s^{-1} for κ_1, 0.000 41 nm^2 s^{-1} for κ_2, 0.12 nm^4 s^{-1} for B, 0.047 nm s^{-1} for λ_1, and 0.040 nm s^{-1} for λ_2. These are only marginally different than the RSME errors we previously obtained, and so a 10% error in the reported fluence has little effect on the accuracy of our model's predictions.

The next logical question is whether the accuracy of the CNN's predictions could be further improved. This could potentially be achieved by more training on the dataset, by adding additional images to the dataset, or by increasing the number of layers within the network. Let us first consider whether more training would improve the model's

[10] One feature that is worth mentioning is the 'hot spots' present for the negative values of κ_1 and κ_2. These appear because the number of simulations per nm^2 s^{-1} was more than twice as large for negative κ_i as for positive κ_i for $i = 1$ and 2.

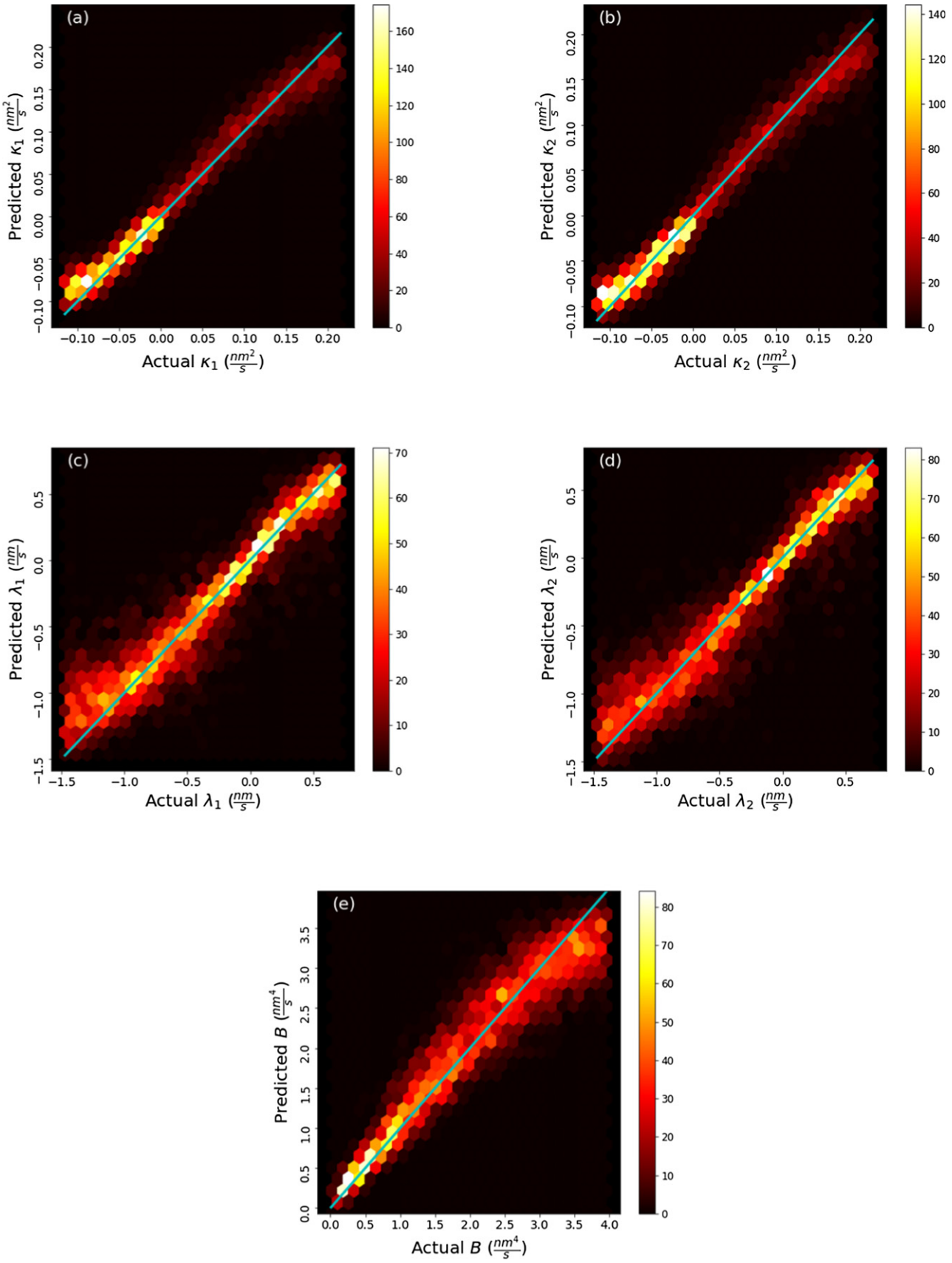

Figure 4. Predicted values vs the true values of the parameters κ_1 (a), κ_2 (b), λ_1 (c), λ_2 (d), and B (e) for a fluence of 5×10^{17} ions/cm^2. The color of a hexagonal region indicates the number of data points within that bin. The solid cyan lines represent perfect agreement between the predicted and actual values. The respective RMSEs are 0.000 37 nm^2 s^{-1} (a), 0.000 35 nm^2 s^{-1} (b), 0.045 nm s^{-1} (c), and 0.039 nm s^{-1} (d), and 0.11 nm^4 s^{-1} (e).

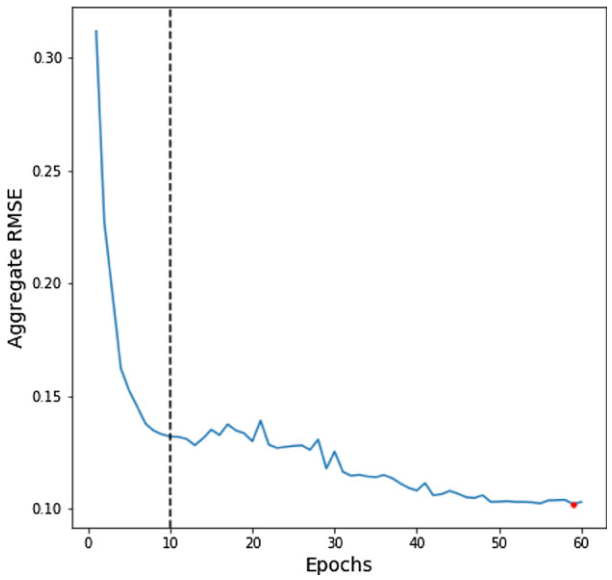

Figure 5. Aggregate fractional RMSE vs epochs for the validation data. The dashed black line indicates when the model was unfrozen. The red dot indicates the minimum.

performance. The sum of the fractional RMSEs of the five parameters is plotted versus the number of epochs in figure 5. The blue line shows the aggregate fractional RMSE and the dashed black line shows the point when the model was 'unfrozen' and fine-tuned. The figure shows that at the start of training, the aggregate RMSE decreases quite dramatically, but later it levels off. We stopped the training at 60 epochs because we had entered a regime of diminishing returns. More training did not greatly improve the aggregate RMSE, and eventually it caused the aggregate RMSE for the validation set to increase (not shown).

Next we consider increasing the size of the dataset used for training the CNN. More data would certainly help to reduce the RMSE on the validation set: as with any analysis, more data is always desirable. However, the network was able to estimate the parameters to within 3% of the parameter ranges used for training. We consider this to be excellent performance, but if greater degree of accuracy were required, the dataset could be augmented by carrying out more simulations.

Lastly, we consider whether adding more layers of neurons to the network (i.e., increasing its 'depth') would be beneficial. Before delving into this question, we first need to discuss underfitting and overfitting of networks. (A comprehensive general discussion of this topic can be found in reference [33].) In general, an ANN does not fully understand the patterns in the training data in the early stages of training and is 'underfit'. Continued training allows the network to learn more about the patterns and the RMSE on both the training and validation sets is reduced. However, eventually the model starts to recognize peculiarities of the training set that are not present in the validation or test sets. Once the model starts to make these connections, it is considered to be 'overfit'. Overfitting is particularly problematic when it causes a decrease of the RMSE on the training set, but increases the RMSE on the validation set.

Additional layers do give the network an increased ability to identify patterns and features within the data. This ability is what has made deep learning so successful for a wide variety of problems. However, deeper networks are not always better. At a certain point, adding more layers makes the model more likely to overfit and perform worse on test data [47]. When we compared results from DenseNet 121 (which has 121 layers of neurons) to results from DenseNet 201 (which has 201), we found that there was more overfitting in the case of the larger network. We therefore opted to use DenseNet 121.

6. Related work

Considering the significant progress achieved with ML and deep learning in particular, it is unsurprising that these algorithms were eventually applied to the study of partial differential equations (PDEs). Numerous studies have been conducted in which these techniques were used both to carry out numerical integration of PDEs as well to infer the values of the parameters in a PDE used to model some aspect of the real world.

Multiple studies of the Kuramoto–Sivashinsky (KS) equation in particular have been conducted using ML algorithms and methods [32, 46, 48–52]. Most of these studies focused on the 1D KS equation, whereas, in our study, we considered the more general case in which the field u depends on both the transverse and longitudinal coordinates x and y. Raissi and Karniadakis used Gaussian processes to estimate the parameters in the one-dimensional KS equation, but required two snapshots of the field u separated by a small time interval Δt and the performance of their model decreases as Δt becomes larger [51]. In another study, Adams *et al* implemented a support vector machine to analyze the two-dimensional aKS equation [46]. However, their study only considered variation of the single parameter λ_1. In addition, rather than looking at a range of values, the parameter was chosen to belong to one of five classes. That reduced the problem from a regression to a classification problem.

The study most closely related to ours was conducted by Reiser [32]. In his study, Reiser used linear regression to estimate the coefficients in the isotropic two-dimensional KS equation and specifically discussed applying his work to ion sputtering. His results demonstrated that ML techniques can predict parameters to a significant degree of accuracy. This was the first paper to use ML techniques to estimate the parameters in the EoM for an ion-bombarded surface and it provided a proof of concept. However, there would be serious difficulties in the application of Reiser's model to real experimental results. Firstly, Reiser's work was limited to normal incidence; this simplifies the problem because it makes the EoM isotropic and reduces the number of parameters. However, the linear terms in the EoM are stabilizing at normal incidence. This means that the surface simply flattens as it is irradiated and no nanostructures emerge. The parameter estimates obtained

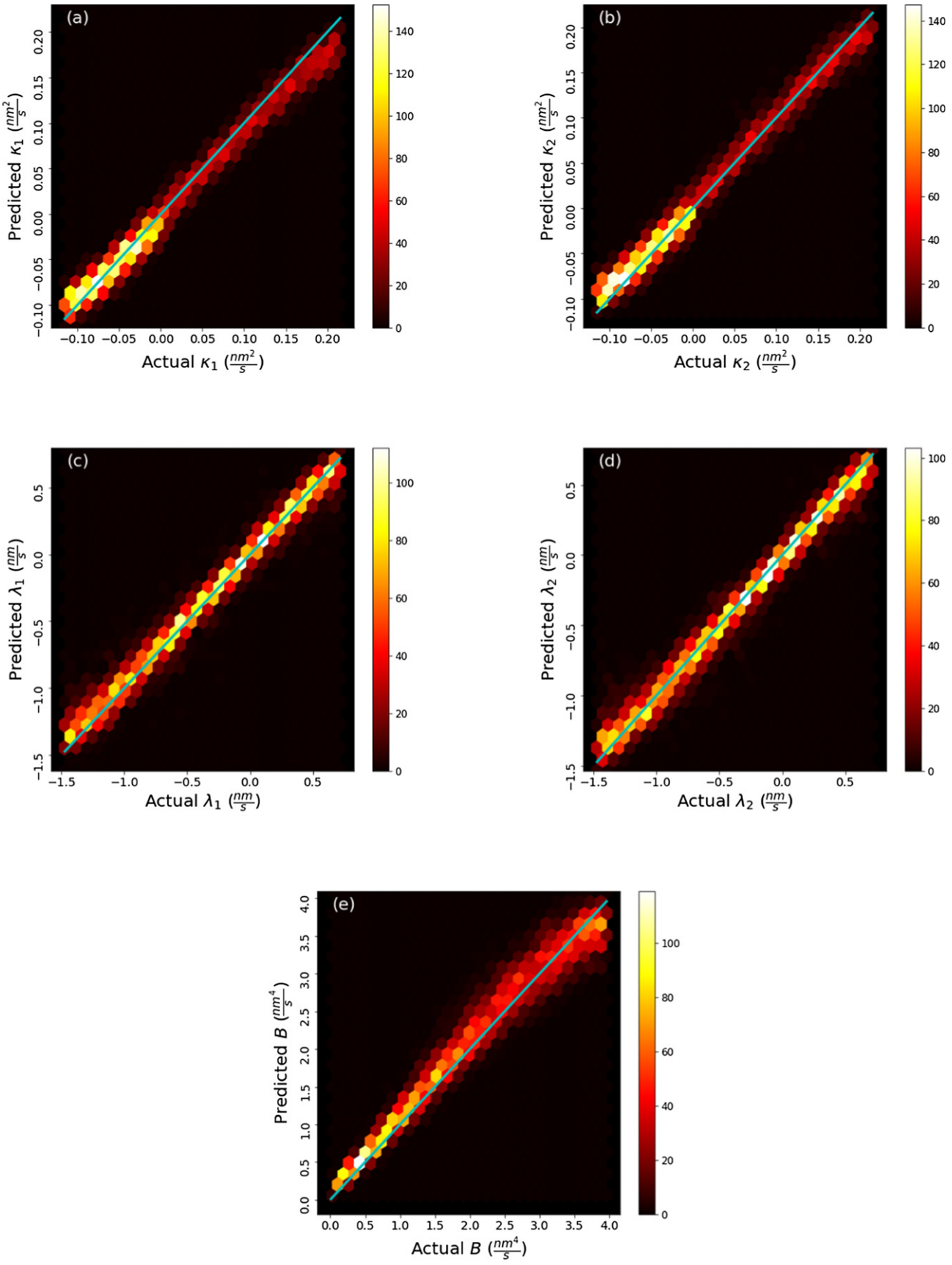

Figure 6. Predicted values vs the true values of the parameters κ_1 (a), κ_2 (b), λ_1 (c), λ_2 (d), and B (e) at a fluence of 10^{17} ions/cm^2. The color of a hexagonal region indicates the number of data points within that bin. The solid cyan lines represent perfect agreement between the predicted and actual values. The respective RMSEs are 0.000 26 nm^2 s^{-1} (b), 0.000 25 nm^2 s^{-1} (b), 0.010 nm s^{-1} (c), 0.011 nm s^{-1} (d), and 0.058 nm^4 s^{-1} (e).

from real experimental surface images would be very inaccurate as a consequence. In any event, the case of normal incidence bombardment of an elemental target with a noble gas ion beam is of little interest. Reiser's method also requires two consecutive snapshots of the surface separated by a short time interval Δt; this would be particularly problematic for the implementation of his method. If a sample were removed from the vacuum chamber to take the first AFM scan, it would be exposed to atmosphere, allowing deposition and/or oxidation to occur. That would change the material present on the surface and its topography and, therefore, the dynamics when the sample was returned to the chamber and irradiation was resumed. Additionally, there would be no guarantee that the second AFM scan would be taken at exactly the same location on the sample as the first. This would introduce an additional source of error. In contrast, our model requires a single AFM image and can be used when the ion beam is obliquely incident.

7. Discussion

In this paper, we demonstrated that a CNN can be trained to predict the parameter values in the EoM (1) from a single AFM scan of an ion-sputtered surface. Our network therefore provides a concrete tool to analyze the nanostructures produced by ion sputtering.

The parameter ranges we chose for training our CNN are for 1 keV Ar^+ bombardment of silicon. Silicon is the most frequently studied target material, argon is the most widely used ion species, and 1 keV is a typical ion energy used in studies of ion-induced pattern formation. However, using a different ion species or target material or changing the ion energy to another energy in the range of say 500–1500 eV would not radically change the dynamics of the bombarded surface—ripple patterns would still be observed and the wavelengths would still be of the same order of magnitude. As a consequence, the parameters in the EoM (1) would not change radically either. Thus, although the parameter ranges used for the training of our CNN were selected for a particular choice of target material, ion species and ion energy, it could be used to estimate the parameter values for many choices of target material, ion species and ion energy.

We trained our network for an ion fluence of 5×10^{17} ions/cm^2. Most experiments are continued until at least this fluence is reached, and it is ordinarily large enough for discernible patterns to emerge. If the parameters are to be inferred for a particular choice of ion beam and target material, an experiment with that ion beam and target material and our selected fluence could be carried out, an AFM scan could be done, and the image could be input into our CNN. If it were for some reason necessary that the parameters be estimated using a preexisting AFM scan with a different ion fluence, our CNN could be retrained for that fluence. To show that this can be done, we retrained our CNN for one fifth of the original fluence, i.e., 1×10^{17} ions/cm^2. We again generated a training set of 32 000 images, a validation set of 8000 images, and a test set of 5000 images. The results are shown in figure 6. Once again, we observed that the predicted parameters cluster about the 1-to-1 agreement line. The respective RMSEs of the parameters κ_1, κ_2, B, λ_1, and λ_2 were 0.000 26 nm^2 s^{-1}, 0.000 25 nm^2 s^{-1}, 0.058 nm^4 s^{-1}, 0.010 nm s^{-1}, and 0.011 nm s^{-1} respectively. To put these values into perspective, the RMSEs for the κ_1 and κ_2 estimates were less than 0.08% of the range used for training, the RMSE of the B predictions was under 1.5%, and the RMSEs of the λ_1 and λ_2 estimates were both under 0.5%. This demonstrates that the network could be retrained for other fluences and reliably estimate the parameters. In fact, the network performed better for the lower fluence, but since we wanted to train our CNN for typical experimental conditions, we initially selected a fluence of 5×10^{17} ions/cm^2.

For high ion fluences, additional terms that are not included in the aKS equation (1) may begin to play a non-negligible role in the dynamics [18, 53–57]. The CKS nonlinearity $\partial_x^2 u_x^2$, for example, is likely responsible for the ripple coarsening that is often observed in experiments [55–57]. As a consequence, if only the parameters in the aKS equation (1) are to be estimated, it would be preferable to use an AFM scan in the low fluence regime in which it is safe to neglect the effect of the additional terms. If, on the other hand, the coefficients of the additional terms are to be estimated, an AFM scan in the high fluence regime would be needed. Simulations would then have to be carried out with the additional terms in the EoM. These would be used to train a CNN that had a number of neurons in the output layer equal to the number of parameters in the modified EoM. This CNN could then be used to estimate all of the parameters in the modified EoM using the AFM scan taken in the high fluence regime.

For the equation of motion (1) to apply, the target material must be elemental and the incident ions must be noble gas ions. Our CNN can therefore only be used to estimate the parameters in the EoM for experiments of this type. If the target material is binary or the incident ions are not noble gas ions, the surface morphology is influenced by the composition of a surface layer and coupled equations of motion are needed [58–62]. Development of a CNN that is capable of estimating the parameters that appear in these coupled equations would be a worthwhile direction for future work.

8. Conclusions

The nanostructures produced by bombardment of the surface of an elemental material with an obliquely incident noble gas ion beam are usually modelled by the aKS equation. This equation has five parameters, each of which depend on the target material and the ion species, energy, and angle of incidence. In this paper, we developed a CNN that uses a single image of the surface to estimate all five parameters in the EoM with root-mean-square errors that are under 3% of the parameter ranges used for training. The network was trained and tested using thousands of images produced by numerically integrating the EoM, but it was developed to enable experimentalists to quickly ascertain the parameters for a given ion-sputtering experiment from a single AFM scan of the solid surface. In future work, our tool will be used to provide a check on parameter estimates determined by other means, e.g.,

GISAXS or atomistic simulations combined with the crater function formalism.

Acknowledgments

The authors are grateful to Matt Harrison and Patrick Shipman for useful discussions. This work was supported by NSF Grant DMS-1814941.

ORCID iDs

R Mark Bradley https://orcid.org/0000-0001-6233-9016

References

[1] Navez M, Sella C and Chaperot D 1962 *C. R. Acad. Sci.* **254** 240
[2] Muñoz-García J, Vázquez L, Castro M, Gago R, Redondo-Cubero A, Moreno-Barrado A and Cuerno R 2014 *Mater. Sci. Eng. R: Rep.* **86** 1–44
[3] Cuerno R and Barabási A-L 1995 *Phys. Rev. Lett.* **74** 4746–9
[4] Makeev M A, Cuerno R and Barabási A-L 2002 *Nucl. Instrum. Methods Phys. Res.* B **197** 185–227
[5] Bradley R M and Harper J M E 1988 *J. Vac. Sci. Technol.* A **6** 2390–5
[6] Carter G and Vishnyakov V 1996 *Phys. Rev.* B **54** 17647–53
[7] Moseler M, Gumbsch P, Casiraghi C, Ferrari A C and Robertson J 2005 *Science* **309** 1545–8
[8] Davidovitch B, Aziz M J and Brenner M P 2007 *Phys. Rev.* B **76** 205420
[9] Castro M and Cuerno R 2012 *Appl. Surf. Sci.* **258** 4171–8
[10] Castro M, Gago R, Vázquez L, Muñoz-García J and Cuerno R 2012 *Phys. Rev.* B **86** 214107
[11] Muñoz-García J, Cuerno R and Castro M 2019 *Phys. Rev.* B **100** 205421
[12] Bradley R M and Hofsäss H 2016 *J. Appl. Phys.* **120** 074302
[13] Hofsäss H, Zhang K and Bobes O 2016 *J. Appl. Phys.* **120** 135308
[14] Umbach C C, Headrick R L and Chang K-C 2001 *Phys. Rev. Lett.* **87** 246104
[15] Norris S A, Brenner M P and Aziz M J 2009 *J. Phys.: Condens. Matter* **21** 224017
[16] Bradley R M 2011 *Phys. Rev.* B **84** 075413
[17] Harrison M P and Bradley R M 2014 *Phys. Rev.* B **89** 245401
[18] Pearson D A and Bradley R M 2015 *J. Phys.: Condens. Matter* **27** 015010
[19] Norris S A, Perkinson J C, Mokhtarzadeh M, Anzenberg E, Aziz M J and Ludwig K F 2017 *Sci. Rep.* **7** 2016
[20] Madi C S, Anzenberg E, Ludwig K F and Aziz M J 2011 *Phys. Rev. Lett.* **106** 066101
[21] Anzenberg E, Perkinson J C, Madi C S, Aziz M J and Ludwig K F 2012 *Phys. Rev.* B **86** 245412
[22] Mokhtarzadeh M, Ulbrandt J G, Myint P, Narayanan S, Headrick R L and Ludwig K F 2019 *Phys. Rev.* B **99** 165429
[23] Muñoz-García J, Gago R, Vázquez L, Sánchez-Garcia J A and Cuerno R 2010 *Phys. Rev. Lett.* **104** 026101
[24] Muñoz-García J, Gago R, Cuerno R, Sánchez-García J A, Redondo-Cubero A, Castro M and Vázquez L 2012 *J. Phys.: Condens. Matter* **24** 375302
[25] Norris S A, Samela J, Bukonte L, Backman M, Djurabekova F, Nordlund K, Madi C S, Brenner M P and Aziz M J 2011 *Nat. Commun.* **2** 276
[26] Hofsäss H and Bobes O 2019 *Appl. Phys. Rev.* **6** 021307
[27] Deng L and Yu D 2014 *FNT Signal Process.* **7** 197–387
[28] Maqueda A I, Loquercio A, Gallego G, Garcia N and Scaramuzza D 2018 *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 5419–27
[29] Wen Y, Zhang K, Li Z and Qiao Y 2016 A discriminative feature learning approach for deep face recognition *Notes Comput. Sci.* **499** 499–515
[30] Young T, Hazarika D, Poria S and Cambria E 2018 *IEEE Comput. Intell. Mag.* **13** 55–75
[31] Voulodimos A, Doulamis N, Doulamis A and Protopapadakis E 2018 *Comput. Intell. Neurosci.* **2018** 1–13
[32] Reiser D 2019 *Phys. Rev.* E **100** 033312
[33] Gérón A 2019 *Hands-on Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems* (Sebastopol: O'Reilly)
[34] James G, Witten D, Hastie T and Tibshirani R 2017 *An Introduction to Statistical Learning: with Applications in R* (Berlin: Springer)
[35] Goodfellow I, Bengio Y and Courville A 2017 *Deep Learning* (Cambridge, MA: MIT Press)
[36] Nwankpa C, Ijomah W, Gachagan A and Marshall S 2018 arXiv:1811.03378
[37] Dhillon A and Verma G K 2020 *Prog. Artif. Intell.* **9** 85–112
[38] Huang G, Liu Z, Maaten L V D and Weinberger K Q 2017 *2017 IEEE Conf. Comp. Vis. Patt. Recog. (CVPR)* pp 2261–9
[39] Shin H-C, Roth H R, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D and Summers R M 2016 *IEEE Trans. Med. Imaging* **35** 1285–98
[40] Hajilounezhad T, Oraibi Z A, Surya R, Bunyak F, Maschmann M R, Calyam P and Palaniappan K 2020 https://doi.org/10.31224/osf.io/7tqam
[41] Cox S M and Matthews P C 2002 *J. Comput. Phys.* **176** 430–55
[42] Constantinides A 1988 Finite difference methods *Applied Numerical Methods with Personal Computers* (New York: McGraw-Hill) p 299
[43] Rajpurkar P *et al* 2017 arXiv:1711.05225
[44] Kingma D P and Ba J 2014 *ICLR Proc. of the 3rd Int. Conf. on Learn. Rep.* vol 112
[45] Smith L N 2018 arXiv:1803.09820
[46] Adams H *et al* 2017 *J. Mach. Learn. Res.* **18** 218–52
[47] Bengio Y 2012 *Lect. Notes Comput. Sci.* **7700** 437
[48] Wang M, Li H-X, Chen X and Chen Y 2016 *IEEE Trans. Syst. Man Cybern.* **46** 1664–74
[49] Pathak J, Lu Z, Hunt B R, Girvan M and Ott E 2017 *Chaos* **27** 121102
[50] Pathak J, Wikner A, Fussell R, Chandra S, Hunt B R, Girvan M and Ott E 2018 *Chaos* **28** 041101
[51] Raissi M and Karniadakis G E 2018 *J. Comput. Phys.* **357** 125–41
[52] Pathak J, Hunt B, Girvan M, Lu Z and Ott E 2018 *Phys. Rev. Lett.* **120** 024102
[53] Loew K M and Bradley R M 2019 *Phys. Rev.* E **100** 012801
[54] Harrison M P, Pearson D A and Bradley R M 2017 *Phys. Rev.* E **96** 032804
[55] Castro M, Cuerno R, Vazquez L and Gago R 2005 *Phys. Rev. Lett.* **94** 016102
[56] Muñoz-García J, Castro M and Cuerno R 2006 *Phys. Rev. Lett.* **96** 086101
[57] Muñoz-García J, Cuerno R and Castro M 2008 *Phys. Rev.* B **78** 205408

[58] Bradley R M and Shipman P D 2010 *Phys. Rev. Lett.* **105** 145501
[59] Shipman P D and Bradley R M 2011 *Phys. Rev.* B **84** 085420
[60] Bradley R M and Shipman P D 2012 *Appl. Surf. Sci.* **258** 4161
[61] Motta F C, Shipman P D and Bradley R M 2012 *J. Phys. D: Appl. Phys.* **45** 122001
[62] Mollick S A, Ghose D, Shipman P D and Mark Bradley R 2014 *Appl. Phys. Lett.* **104** 043103