Optimizing Atomic Layer Deposition using a Hybrid of Machine Learning Methods

Anthony DiBenedetto
Department of Engineering, Computing, and Mathematical Sciences
Lewis University
Romeoville, IL, USA
anthonypdibenedett@lewisu.edu

Angel Yanguas-Gil
Applied Materials Division
Argonne National Laboratory
Lemont, IL, USA
ayg@anl.gov

Jeffrey W. Elam
Applied Materials Division
Argonne National Laboratory
Lemont, IL, USA
jelam@anl.gov

Osama "Sam" Abuomar
Department of Engineering, Computing, and Mathematical Sciences
Lewis Univeristy
Romeoville, IL, USA
oabuomar@lewisu.ed

Abstract— **Atomic layer deposition (ALD) is a crucial technique in semiconductor miniaturization and high-precision applications. The quality of ALD processes directly affects the properties of the resulting thin films, leading to extensive evaluations for new ALD procedures. This study uses machine learning to quickly assess ALD process quality, with a focus on predicting the standard deviations of film thickness as an indicator of quality. Using a synthetic dataset simulating non-ideal ALD processes, we evaluated the performance of Random Forest Classifier (RFC), Support Vector Machines (SVMs), and K-Nearest Neighbor (KNN). We also introduced artificial neural network (ANN) and convolutional neural network (CNN) models for predicting standard deviations of film thickness from ALD trials. Our ANN and CNN models showed promising results, positioning them as reliable tools for predicting ALD process quality.**

Keywords—artificial neural network, atomic layer deposition, ensemble learning, classification, regression, machine learning

I. INTRODUCTION

Atomic layer deposition (ALD) is a thin film growth technique used extensively in semiconductor manufacturing and with emerging applications ranging from energy storage to pharmaceuticals [1], [2]. ALD relies on self-limiting chemical reactions between chemical precursor vapors and a solid surface to grow materials in an atomic layer-by-layer fashion [1]. The ALD precursors only react on specific chemical sites on the surface, so the ALD surface reactions naturally terminate when all the available surface sites are consumed. This self-limiting chemistry provides atomic-level control over thickness and composition and allows complex surfaces such as nanoporous membranes and powders to be coated with excellent uniformity and conformality [1], [2]. These attributes make ALD an ideal technology for manufacturing advanced microelectronic devices at the nanometer scale.

The rapid growth of ALD in microelectronics and other applications is necessitates the development of new ALD processes that expand the palette of available thin film materials to include new metals, semiconductors, etc. ALD process development involves extensive experimentation to identify and evaluate chemical precursors to determine the set of conditions (temperature, pressure, precursor exposure times, etc.) that provide self-limiting chemistry and ideal ALD growth. However, many candidate precursors do not provide ideal, self-limiting growth under any conditions and therefore do not provide the precise control over thickness and composition and the uniformity needed for microelectronics [3]. This trial-and-error approach for ALD process development is time consuming and expensive. The ability to classify a particular ALD process as ideal or non-ideal using a minimal set of experiments would greatly accelerate the discovery of ALD processes for new materials. We believe that machine learning (ML) can be beneficial to solve this problem.

In our previous study applying ML to ALD process development, we demonstrated that deep neural networks could effectively predict the precursor exposure times necessary to achieve uniform growth based on a single set of measured thickness values [3]. A key assumption in this previous work was that the ALD chemistry was self-limiting. In the present investigation, we evaluate ML methods for determining if a particular ALD process is self-limiting. As in the previous study, we created a synthetic dataset of film thickness values that simulates various ALD scenarios but in the present work we include both self-limiting and non-self-limiting chemistries. We use this dataset to train and test several machine learning models, including Random Forest Classification (RFC), Support Vector Machine (SVM), and K-Nearest Neighbor (KNN). Each model classifies a sample as either an ideal, self-limited or non-ideal, non-self-limited ALD process [2]. To help us understand our dataset, we perform feature selection, allowing us to understand how many features (dimensions) are needed to make a reliable prediction. We also added Gaussian noise to the dataset, which mimics the randomness inevitable in real-world measurements.

This helps us test how well our models can handle these real-world conditions.

In addition, we develop an artificial neural network (ANN) and a convolutional neural network (CNN) to predict the variations in film thickness resulting from each ALD trial [2], [6]. This ability to predict variation is crucial for evaluating how well the ALD process works [2], [6].

II. Model Generation

A. Dataset Preparation

Our dataset consisted of simulated growth profiles comprising sets of 120 thickness values measured at discrete locations along the axis of a hypothetical ALD viscous flow reactor [4]. The dataset was generated using a reactive transport model of ALD in a cylindrical low-pressure reactor under typical conditions [5]. The growth profiles were simulated using different precursor exposure times to create different degrees of thickness saturation. Each entry consisted of a set of growth profiles and a label categorizing the ALD process as ideal self-limited, soft saturating, or non-self-limited. For binary classification, we grouped the soft saturating and non-self-limited processes into one category and compared them against the ideal self-limited processes.

We developed a balanced dataset incorporating contributions from these three processes, providing two types of labels; the first indicating if the process is fully self-limited, and the second showing whether the standard deviation of the asymptotic z(x) is below 5%. Z(x) represents a series of thickness values where the values are measured at predetermined positions in the reactor, categorized as a vector x. We considered three growth profiles at 20%, 40%, and 60% of the total coverage of the ALD reactor, and each sample included a three-channel 1D input with two separate labels. These points, located 2 cm apart in the ALD reactor, were configured to reflect the experimental ALD coating systems in our lab. All our data was normalized for this study.

Our training dataset consisted of 10,000 samples. The testing data had the same structure as the training dataset and consisted of 1,000 samples.

B. Ensemble Learning

RFC, KNN, and SVMs are popular machine learning models commonly used in classification problems [4]. These models are known as ensemble learning methods as they combine the predictions of several base estimators to improve generalizability and robustness [7]. In this study, we use these models to categorize our dataset into 'ideal' and 'non-ideal' ALD processes, which is known as binary classification.

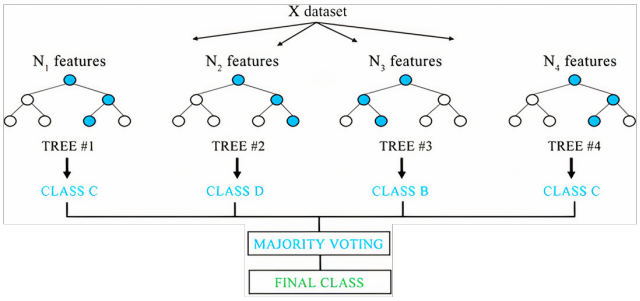

Fig. 1. The overview of the random forest algorithm [7]. Random Forest classifiers use multiple trees with randomized training. Leaf nodes are labeled with class distributions and tests at internal nodes split the data. Randomness is introduced through sub-sampling and node test selection during training.

RFC operates by constructing a multitude of decision trees during training. The model outputs the class that is the mode of the classes of the individual trees for classification [7]. For our study, the RFC model was initialized with 250 estimators, a maximum depth of 25, and the class weight balanced in case different classes have a significantly different number of samples [7]. Displayed in Fig. 1 is a basic visual overview of a random forest classifier.

Another excellent algorithm used for classification is KNN. It works simply by considering the "k" nearest data point in the space [7]. It then takes the majority vote of the data points to choose its prediction. The primary concept behind KNN is to leverage the similarity between data points in the feature space [7]. The assumption is that similar data points are likely to belong to the same class or have similar target values. There are some key hyperparameters we adjusted to this classifier for its predictions. First, we specified "*k*" as three, which as explained

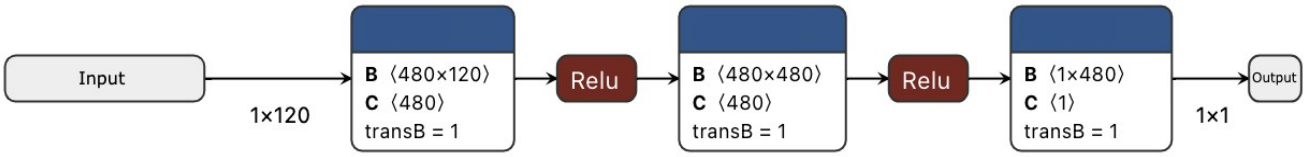

Fig. 2. Our ANN architecture. The input to this model is all 120 columns of data, then the hidden layer expands to 480 nodes. From there the model returns the output which contains one node as we are solving for the standard deviation of the film thickness resulting from each ALD trial.

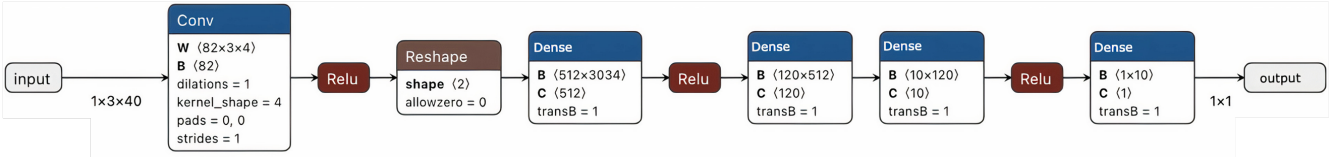

Fig. 3. Our CNN architecture. This model has a convolutional layer, and four fully connected or dense layers. The output of this model returns only one node whose value is the standard deviation of the film thickness resulting from each ALD trial.

previously is the total number of neighbors considered for making a prediction. We also set the leaf size to 30 which specifies how many data points to have on each node before the majority vote occurs.

SVM is a powerful, flexible, yet computationally efficient approach to machine learning that is well suited for classification problems [7]. SVMs are particularly well suited to complex but small or medium-sized datasets. For our SVM, we used the Radial Basis Function (RBF) as our kernel. The RBF kernel, sometimes also referred to as the gaussian kernel, is used to transform data into a high dimensional space [7]. This makes it easier when we have data with many dimensions to find their optimal hyperplane. We also specified the "C" parameter which is also known as the regularization parameter [7]. This parameter shows the tradeoff between low training error and low testing error [7]. The smaller C value results in wider margin which can lead to some misclassification, but it can also be better at predicting unseen data since the decision boundary created with low C value is simpler [7]. We had the best results when we made the regularization parameter as 128.

C. ANN Model Architecture

ANN mimics the functioning of the human brain to process data [10]. They comprise layers of interconnected nodes or "neurons" that can interpret sensory data through a kind of machine perception, labeling or clustering raw input. We utilized a simple three-layer feed-forward neural network (FNN), specifically a Multilayer Perceptron (MLP) [7].

Our ANN consists of an input layer, one hidden layer, and an output layer. The input layer contains 120 nodes (neurons) as we have 120 input features from the three-channel input. The hidden layer expands to 480 nodes, enhancing the model's ability to capture complex features [7]. Both input and hidden layers use ReLU (Rectified Linear Unit) as an activation function. ReLU functions work through a set of conditions. If the input value is greater than or equal to zero, the function returns the input value itself; if the input value is less than zero, the function returns zero [8]. The output layer of our network contains one node as we are solving for the standard deviation of the film thickness resulting from each ALD trial. Displayed in Fig. 2. is the architecture used in this model.

ANNs use loss functions to provide a metric for the data expected versus the data produced. The goal of training ANN model is to have a loss function that decreases and becomes very small [9]. For the model, we used the L1 loss function which is also known as a mean absolute error (MAE). It is defined as the average of the absolute differences between the predicted and true target values.

An optimizer is also used with artificial neural networks to minimize the loss by adjusting the model's parameters [10]. It can learn what adjustments must be made over time to make the loss smaller. For the model we created, we used an optimizer called Stochastic Gradient Descent (SGD). This optimizer updates each step based on the gradients of the loss function [10]. In the context of machine learning, gradient is a vector which represents a partial derivative of a function. This can provide information about the direction and the magnitude of the changing parameters [10]. This is especially useful as it helps to guide the optimization process to produce better results. There are many different optimizers, but there isn't one that is the best. It just depends on your model's architecture and the data we are working with.

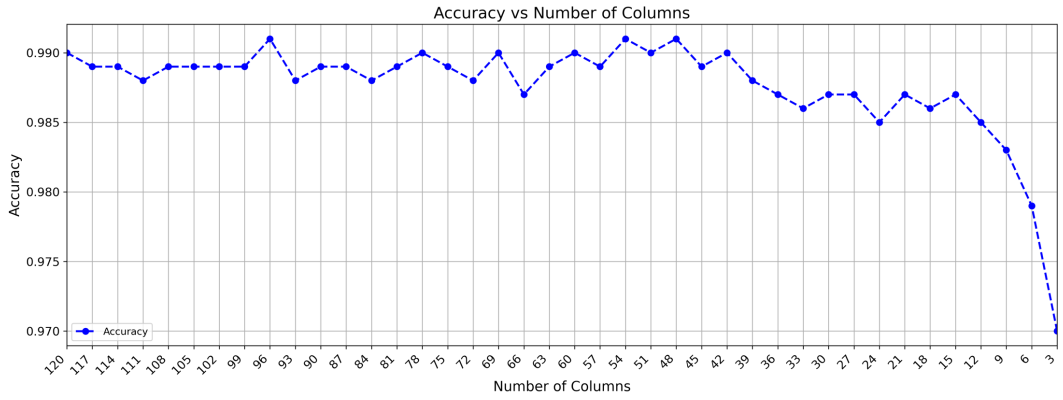

Fig. 4. This graph represents the removal of data points from each channel. Each time the data points are removed the model is retrained on the data. When the model reached 48 columns total or 16 columns per channel the model achieved an accuracy of 99.1%.

Optimizers also have hyperparameters which can help the process along. SGD has its own set of hyper parameters which we specified. We declared its learning rate to be 0.0001. The learning rate determines the step size during the gradient optimization process [10]. The larger the learning rate the higher the steps, which can speed up the process of convergence. We specified the momentum to be 0.9, dampening to be 0.01, and weight decay to be 0.0005. The momentum helps to accelerate the process by accumulating gradient velocity from previous steps [10]. This helps because knowing information from the previous step directs the optimizer to overcome flat regions in the loss function more efficiently [10]. Dampening is a factor that helps to reduce the effect of momentum [10]. This is very useful in our case as we have set the momentum to be 0.9 which is high for a momentum value. Lastly weight decay is a technique that is used to stop over-fitting. Overfitting is when the model is trained well based off training data which makes the loss function looks good, but on new data it performs poorly [10]. Weight decay adds a penalty term to the loss function. The penalty term reduces overfitting by penalizing the model's weights so that the weights take only small values [11].

D. CNN Model Architecture

The CNN architecture begins with a convolutional layer, which differentiates it from the ANN architecture. This layer can recognize local patterns in input data due to the convolution operation [11].

Our initial layer accepts a 1-dimensional input with 3 channels. This layer applies 82 distinct filters, each of a size of 4 and a stride (or step size) of 1. This operation allows the extraction of key features from the input data [11], [12]. Following the convolutional operation, we apply the ReLU activation function.

After the convolutional layer, we must flatten the output tensor into a 2-dimensional tensor to prepare it for the subsequent fully connected layers [11], [12]. These layers serve the purpose of performing the actual regression based on the patterns recognized in the previous layers [12]. The first layer accepts the flattened tensor from the preceding layer and sets the dimension to 512 nodes with a ReLU activation function. The second fully connected layer further reduces the nodes to 120. The third layer applies a ReLU activation function and reduces the number of nodes to 10. Finally, the fourth layer generates a single output node that predicts the standard deviation of film thickness from each ALD trial. Fig. 3 displays the architecture used in this model.

Like the ANN architecture, the CNN model employs the L1 loss function, but uses the Adam optimizer. The learning rate for the optimizer is set to 0.0001.

E. ANN and CNN Model Training

The ANN model was trained for 800 epochs with a batch size of 16. Batch size refers to the number of samples that are processed at once during training [10].

The CNN model was trained for 400 epochs with a batch size of 20. During each epoch, the training data was also shuffled, and the model weights were updated based on the calculated loss.

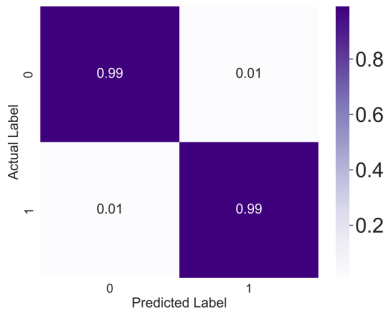

Fig. 5. This is a confusion matrix for the output of the random forest algorithm. The matrix displays the probability of how often the actual label is the predicted label.

F. Reducing Data Requirements for Accurate Results

Features selection in ALD evaluation serves multiple purposes, including the reduction of input features and simplification of the dataset. By reducing the number of features, computational complexity is alleviated, and the model's interpretability is enhanced. However, another practical advantage of reducing data is that it can potentially reduce the amount of data required to achieve accurate results.

This reduction in data requirements not only simplifies the data collection process, but also facilitates the overall workflow in ALD evaluation. With fewer data points to gather, prepare, and process, researchers can streamline their efforts and focus on collecting the most crucial data that contributes to accurate classification results. This convenience in data preparation enables practitioners to obtain reliable outcomes from the ML model while optimizing resource utilization in the lab.

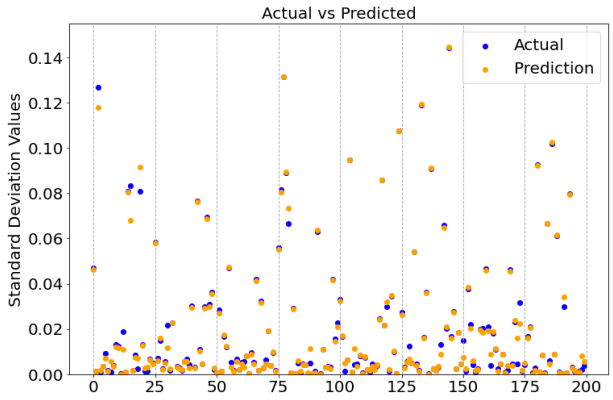

Fig. 6. Shown is the performance of the ANN model. Displayed is the comparison of the actual versus predicted standard deviation values. While there is a discernible correspondence between the data points, some discrepancies illustrate areas of improvement for the model.

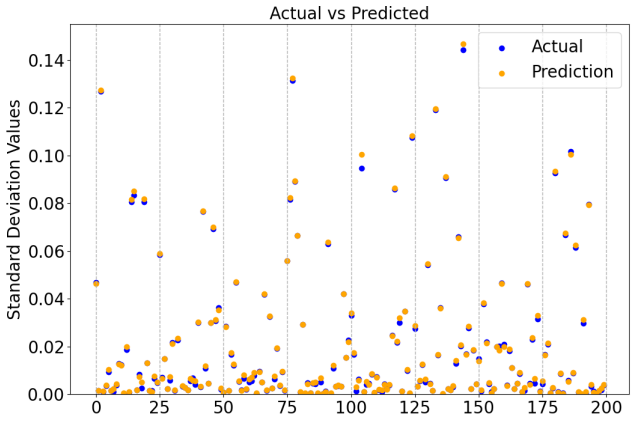

Fig. 7. Displayed is the accuracy of the CNN model. The alignment between the actual and predicted standard deviation values shows how well the model can make predictions, as it is much more difficult to see blue points (actual values) compared to the blue points on the ANN model. Over a span of 200 points, you can view how the model's adeptness can be used accurately in predicting real world outcomes.

G. Methodology

The primary goal was to reduce the number of features while maintaining or enhancing the models' accuracy. Since the data is separated into three channels each being 40 columns, we removed one column at a time from right to left. This corresponds to eliminating samples from the upstream side of the ALD reactor [4]. After removing each column, we retrained the model and ran it on test data. We used the RFC model for this test as it provided us with the best results previously. This approach was chosen because it not only reduced the dataset's complexity, but also it preserved some of the original structure of the data set.

III. Results and Discussion

A. Evaluation of the Ensemble Learning Models

Among the 3 ensemble learning models we used; RFC performed the best with an accuracy of 99%. Displayed in Fig. 5 is a confusion matrix to help visualize these results. This high accuracy shows how the model is very effective at capturing patterns in the data to produce an accurate outcome. KNN performed very well as it achieved an accuracy of 98.5%. This is not as well as the RFC, but still provided very accurate results. The SVM achieved an accuracy of 97.2%. These results are very impressive, but they are not as accurate as the RFC or the KNN.

B. Evaluation of the ANN

The ANN performed well in predicting the standard deviation of the film thickness. Displayed in Table I we produced some metrics to understand how the model ran on the test data. The high R-squared value and low error metrics (MSE and MAE) suggest that the model has successfully captured the relationships in the dataset. Shown is Fig. 6 the ANN model's performance is displayed visually through actual and predicted standard deviation values. There's a clear relationship between the two, but there are noticeable errors in the predictions.

C. Evaluation of the CNN

The CNN performed better than the ANN model overall, with less epochs. Table II displays the metrics produced on test data using this model. This model produced a higher R-squared value and had even lower error metrics (MSE and MAE). This means that this model was able to successfully find patterns in the data more efficiently than the ANN model. You're able to view the model's predictive capabilities shown in Fig. 7 where it compares the actual and predicted standard deviation values.

D. Analysis of Reducing Data

The features selection analysis demonstrated that the optimal configuration of 16 columns per channel, or 48 columns total, produced impressive accuracy of 99.1% with the RFC model. Displayed in Fig. 4 we can see the results and the point where the model becomes more accurate with less data. These results indicate that reducing the dataset's features while preserving the structure can lead to improved model performance.

E. Evaluation using Gaussian Noise Dataset

Adding Gaussian noise to our dataset simulates the random errors inherent to real world thickness measurements of actual samples. This noise can help evaluate the robustness of

Table I. Shown is the performance from the ANN model. The R-squared value of 0.9922, indicating high correlation. With low MSE and MAE values, the model demonstrates good accuracy. The MAE is notably lower than the baseline, showing model improvement.

Metric	Value
Mean of Target	0.0217
Mean of Prediction	0.0219
Median of Target	0.0056
Median of Prediction	0.006
R-Squared	0.9922
MSE	9.0032e-06
MAE	0.0014
Baseline MAE	0.0241

Table II. Displayed is the performance by the CNN model, with an R-squared of 0.99928 indicating nearly perfect fit. Extremely low MSE and MAE values highlight remarkable accuracy, with the MAE notably lower than the baseline, signifying substantial improvement.

Metric	Value
Mean of Target	0.0217
Mean of Prediction	0.0220
Median of Target	0.0056
Median of Prediction	0.0060
R-Squared	0.99928
MSE	8.264124e-07
MAE	0.00056607
Baseline MAE	0.0241

our models to perform accurate classification using real-world, noisy data.

In our case, after adding Gaussian noise to our dataset, we ran the RFC model, which resulted in an impressive accuracy of 99.4% on the test data. This accuracy, even better than with the clean dataset, is a strong indicator of the model's capacity to handle noise and its robustness in its classification tasks. The Random Forest algorithm's inherent property of averaging over several decision trees can help reduce variance and manage noise. It's likely that each individual decision tree in the Random Forest model was able to learn from different noise patterns and thus, when aggregated, these models could make more accurate predictions.

IV. Conclusions and Future Work

In this research, we developed and compared machine learning models for the optimization of atomic layer deposition processes. Our primary focus was on binary classification to predict whether an ALD process was ideal and self-limited, or non-ideal and non-self-limited. We employed Random Forest, K-Nearest Neighbors, and Support Vector Machine algorithms, with the Random Forest model yielding the highest accuracy of 99%.

The performance of the Random Forest Classifier model on the dataset with added Gaussian noise was excellent, reaching an accuracy of 99.4%. This result suggests the model's robustness against noise and its suitability for real-world data, where noise and variability are inherent.

In addition to binary classification, we developed two types of neural networks to predict the standard deviation of film thickness. The convolutional neural network demonstrated strong performance, with an R-squared value of 0.99928 and low error metrics, indicating that it effectively captured the dataset's structure and the relationships between its features and samples. Furthermore, we conducted feature selection to explore the impact of the features on prediction accuracy. The optimal configuration was found to be 16 columns per channel, resulting in an improved accuracy of 99.1% with the Random Forest model.

The findings of this research highlight the potential of machine learning techniques, particularly the Random Forest model, to accurately evaluate ALD processes. Applied in a real-world setting, this model could dramatically accelerate ALD process development by reducing experimental time and cost. Although this study used simulated thickness measurements from a simple, 1D model of a research-scale ALD reactor, these methods could easily be applied to actual thickness values measured on a production-scale (i.e., 300 mm Si wafer) commercial ALD tool. Future work in this area could explore other machine learning techniques, as well as incorporating additional features, target variables, and data sources to further enhance the exploratory and predictive capabilities of the models. Additional future work could explore the lower limits for the size of the data set required for accurate classification.

V. Acknowledgment

This research is based in part on the work supported by the Laboratory Directed Research and Development (LDRD) funding from the Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. DOE under Contract No. DE-AC02-06CH11357. We would like to thank Dr. James Girard summer undergraduate research (SURE) program at Lewis University for its support and funding.

VI. References

[1] Vos and A.J.M. Mackus, "Atomic Layer Deposition Process Development – 10 steps to successfully develop, optimize and characterize ALD recipes – Atomic Limits." https://www.atomiclimits.com/2019/02/12/atomic-layer-deposition-process-development-10-steps-to-successfully-develop-optimize-and-characterize-ald-recipes/ (accessed Oct. 16, 2021).

[2] S. M. George, "Atomic Layer Deposition: An Overview," Chemical Reviews, vol. 110, no. 1, pp. 111–131, Jan. 2009, doi: 10.1021/CR900056B.

[3] H. H. Sønsteby, A. Yanguas-Gil, and J. W. Elam, "Consistency and reproducibility in atomic layer deposition," J. Vac. Sci. Technol. A, vol. 38, p. 020804, 2020, doi: 10.1116/1.5140603.

[4] G. Rimal, A. R. Mazza, M. Brahlek, and S. Oh, "Diffusion-assisted molecular beam epitaxy of CuCrO2 thin films," J. Vac. Sci. Technol. A, vol. 40, p. 062408, 2022, doi: 10.1116/6.0001973.

[5] A. Yanguas-Gil, J. A. Libera, and J. W. Elam, "Reactor scale simulations of ALD and ALE: Ideal and nonideal self-limited processes in a cylindrical and a 300 mm wafer cross-flow reactor," J. Vac. Sci. Technol. A, vol. 39, p. 062404, 2021, doi: 10.1116/6.0001212.

[6] Ramprasad, R., Batra, R., Pilania, G., Mannodi-Kanakkithodi, A., & Kim, C. (2017). Machine learning in materials informatics: recent applications and prospects. npj Computational Materials, 3(1). https://doi.org/10.1038/s41524-017-0056-5

[7] Boateng, E. Y., Otoo, J., & Abaye, D. A. (2020). Basic Tenets of Classification Algorithms K-Nearest-Neighbor, Support Vector Machine, Random Forest and Neural Network: A Review. Journal of Data Analysis and Information Processing, 8(4), November 2020. DOI: 10.4236/jdaip.2020.84020.

[8] J. Brownlee, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," Machine Learning Mastery, January 9, 2019. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[9] J. Brownlee, "Loss and Loss Functions for Training Deep Learning Neural Networks," Machine Learning Mastery, January 28, 2019. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

[10] J. Price, A. Wong, T. Yuan, J. Mathews, T. Olorunniwo, "Stochastic gradient descent," Systems Engineering at Cornell University, 2020. https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent

[11] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," Insights into Imaging 2018 9:4, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/S13244-018-0639-9.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature 2015 521:7553, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.