A Deep Learning Approach to Maximizing Electrostatic Sieve Efficiency in Regolith Beneficiation

Kalpit M. Vadnerkar*
Dept. of Electrical and Computer Engr.
Clemson University
Clemson, SC, USA
kvadner@clemson.edu

Emmanuela Amen Eze*
Dept. of Mechanical and Aerospace Engr.
Missouri University of Science and Technology, Rolla, MO, USA
eeze@mst.edu

Rinoj Gautam
Dept. of Computer Science and Engr.
University of North Texas
Denton, TX, USA
rinojgautam@my.unt.edu

Daoru Han
Dept. of Mechanical and Aerospace Engr.
Missouri University of Science and Technology, Rolla, MO, USA
handao@mst.edu

Xin Liang
Department of Computer Science
University of Kentucky
Lexington, KY, USA
xliang@uky.edu

Tong Shu†
Dept. of Computer Science and Engineering
University of North Texas
Denton, TX, USA
tong.shu@unt.edu

***Abstract*—This study investigates the optimization of an electrostatic sieve designed for lunar regolith beneficiation. Two parameters of the electrostatic sieve, 1) the voltage amplitude and 2) angle of inclination, were chosen as variables in the optimization process. Numerical simulations revealed that increasing voltage amplitude significantly enhances sieve performance over the sieve angle. However, optimal separation required careful voltage adjustment for specific sieve angles. A comprehensive dataset incorporating additional parameters was then created to train Machine Learning (ML) and Deep Learning (DL) models for further optimization. The ML/DL models were trained on a small subset of the original dataset to predict the yield. We showcase the benefits of leveraging DL techniques to improve the electrostatic sieve for regolith beneficiation via tailored evaluations. Our model, trained on lower-yield examples, accurately (92%) identifies parameter combinations that increase yields above 30%. It leads to a near-optimal yield with $10\times$ reduction on runtime when compared with exhaustive simulations. This not only reduces the reliance on resource-intensive numerical simulations but also offers a rapid, validated approach to optimizing equipment for lunar mining operations.**

***Index Terms*—Electrostatic Sieve, Simulation, Sampling, Regression, High Performance Computing, Machine Learning, Deep Learning Model**

I. Introduction

As NASA prepares to return to the Moon, renewed focus on *In-Situ* Resource Utilization (ISRU) is critical. Efficiently utilizing lunar resources like water, oxygen, and metals is key to reducing dependence on Earth-based supplies [1]. Lunar regolith contains valuable metals, but extracting them requires pre-processing to separate the desired mineral-rich portions from less valuable material. This beneficiation step improves the efficiency of subsequent chemical extraction processes. The absence of standard ore grades on the Moon adds another layer of complexity to ISRU efforts. Terrestrial particle separation methods, such as cyclones and air classifiers, are often power-hungry, bulky, and complex, making them unsuitable for lunar applications [1]–[3]. Simpler, more adaptable solutions are needed. Electrostatic separation has emerged as a promising approach due to its potential for compact, low-power operation [4], [5]. Prior work by Kawamoto and Adachi explored electrostatic sieves and filing separators for particle classification, but achieved limited yields of fine particles (less than 10μm) [2]. This paper builds on previous research in particle dynamics within electric fields [6]–[9] and leverages machine learning (ML) and deep learning (DL) techniques to enhance the performance of electrostatic sieves for lunar beneficiation.

A. Kinetic Modeling of an Electrostatic Sieve

The sieve modeling is similar to the work done by Ingram et al. in [9]. Two parameters, the voltage amplitude and the angle of inclination of the sieve were manipulated to run four cases and observe the corresponding effect on the performance of the sieve on Earth. The kinetic modeling of the sieve was carried out using a FORTRAN code package designed to solve for the electric potential and field in a particular domain of interest and using the resulting electrostatic force from the field to track the motion of lunar dust grains. Due to the expensive simulation procedure, optimizing the performance of the sieve is highly desired. This necessitates the use of ML and DL models for fast and accurate prediction using the different combinations of input parameters.

* These authors contributed equally to this work.
† Tong Shu's ORCID iD: 0000-0001-8617-1772

B. Leveraging Deep Learning for Optimization

Deep learning models offer a transformative approach to electrostatic sieve optimization by leveraging existing data to predict optimal operational parameters (e.g., voltage magnitude, frequency, sieve angle, electrode pitch). Unlike traditional methods that rely heavily on scarce mathematical models, deep neural networks (DNNs) are model-agnostic and can be fine-tuned for superior performance, as highlighted by Shlezinger et al. [10]. Yang et al. [11] show how deep learning has been successfully applied to several application problems. This data-driven approach has demonstrated remarkable success in various fields, including the classification of handwritten digits [12], image recognition [13]–[15], natural language understanding [16], and so on. In our study, we harness the power of deep learning to develop a neural network that predicts particle yield based on historical experimental data. Due to its high inference speed (compared with simulations), the neural network functions as a surrogate model, efficiently optimizing operational parameters and significantly reducing the time required for researchers to identify optimal input combinations. Our methodology includes comprehensive data processing and feature engineering techniques, which will be detailed later. Additionally, we benchmark the performance of our deep learning model against conventional machine learning approaches to showcase its effectiveness in this context.

C. Technical Contributions

1) A feedforward neural network architecture for accurate yield prediction on unseen data.
2) Comparison of machine learning and deep learning models for inference speed and accuracy.
3) Integration of the model into the optimization approach, enabling configuration recommendations and reduced runtime.

The remainder of this paper is structured as follows: Section II introduces the kinetic modeling approach, which serves as our data generator and a baseline for identifying the best-fit configuration. Section III describes the design and architecture of the proposed neural network, along with the evaluation results for (1) comparison with traditional machine learning approaches and (2) efficiency in finding the best-fit configurations when compared with exhaustive simulations. Section IV concludes the paper with a vision for future works.

II. Kinetic Modeling

A. Generating Particle Radii

Lunar regolith radii follows the Logarithmic-Normal Distribution according to data acquired from the Apollo missions [17]. Since, obtaining the inverse CDF of the Logarithmic-Normal Distribution is not feasible, we use the Accept-Reject method to generate particle sizes that follow the Logarithmic-Normal Distribution using the Uniform Distribution which can be easily sampled from [18].

B. Solving the Electric Field

Changing electric fields with respect to time are needed to propel the lunar soil particles. These electric fields constitute a multi-phase electrostatic traveling wave. In order to move the particles, we have to solve for the electric potential, electric field, and subsequently the electrostatic force caused by the electrode configuration for each phase. For this work, the Immersed Finite Element (IFE) - Poisson Solver was employed to solve for the electric potential throughout the domain of interest. The IFE - Poisson Solver has been utilized in solving other problems related to particle dynamics in electrostatic fields [19]–[29]. The electric potential and electric field are obtained using Eq. (1) and Eq. (2), respectively, where Φ is the electric potential and $\mathbf{E}$ is the electric field.

$$\nabla^2 \Phi = 0 \tag{1}$$

$$\mathbf{E} = -\nabla \Phi \tag{2}$$

C. Dust Particle Dynamics in Electric Fields

The motion of a charged dust particle influenced by both electric and gravitational fields is described by Newton's second law, as expressed in Equation (3):

$$\begin{gathered}\mathbf{F} = m_{\mathrm{d}} \frac{d\mathbf{v}}{dt} = Q_{\mathrm{d}} \mathbf{E} - m_{\mathrm{d}} g \hat{\mathbf{k}}, \\ \text{where } m_{\mathrm{d}} = \rho \frac{4}{3} \pi r^3\end{gathered} \tag{3}$$

In this equation, m_d represents the mass of the dust particle (assumed to be spherical), Q_{d} is its charge, g is the gravitational acceleration, and $\mathbf{v}$ is the dust particle's velocity vector. The instantaneous local electric field, $\mathbf{E}$, varies depending on the phase of the electrode bias, as each phase results in a distinct electric field distribution due to the changing potentials on the electrodes. We assume a lunar regolith density, ρ, of 2.65 g/cm^3 [30].

$$Q_{\mathrm{d}} = C\Phi_{\mathrm{d}} = 4\pi\epsilon_0 r_{\mathrm{d}} \Phi_{\mathbf{d}} \tag{4}$$

In Equation (4), Φ_{d} denotes the potential of the dust particle. This potential is interpolated from the electric field data generated by the biased electrodes at a specific phase. The interpolation is based on the dust particle's position within the computational domain.

D. Cases

We conducted four case studies, summarized in Table I, to analyze how electric potential and inclination angle impact sieve performance in lunar conditions. These findings inform the parameters used for our machine learning models, where we explore variations in both declining and inclining sieve configurations (Fig. 1) to evaluate their effects.

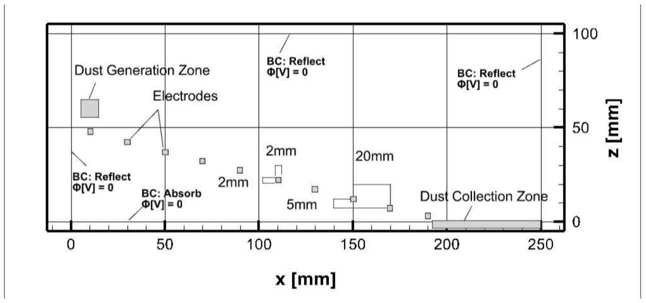
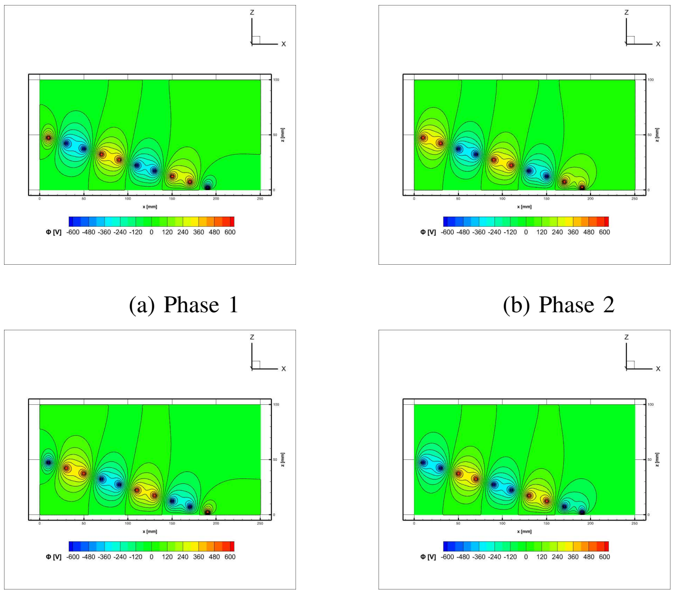


Fig. 1: Setup and boundary conditions of the simulation domain for an example case

TABLE I: Electrostatic Sieve Parameters (All dimensions are in millimeters, except voltage amplitude in volts and sieve angle in degrees)

Sieve Parameter	Cases			
	Case 0	**Case 1**	**Case 2**	**Case 3**
Voltage Amplitude	600	1800	600	1800
Horizontal Pitch	20	20	20	20
Vertical Pitch	5	5	18	18
Sieve Angle	14	14	42	42
X-dimension	250	250	250	250
Z-dimension	100	100	250	250
Dust Gen. Zone - Zmin	55	55	190	190
Dust Gen. Zone - Zmax	65	65	200	200



(a) Phase 1

(b) Phase 2

(c) Phase 3

(d) Phase 4

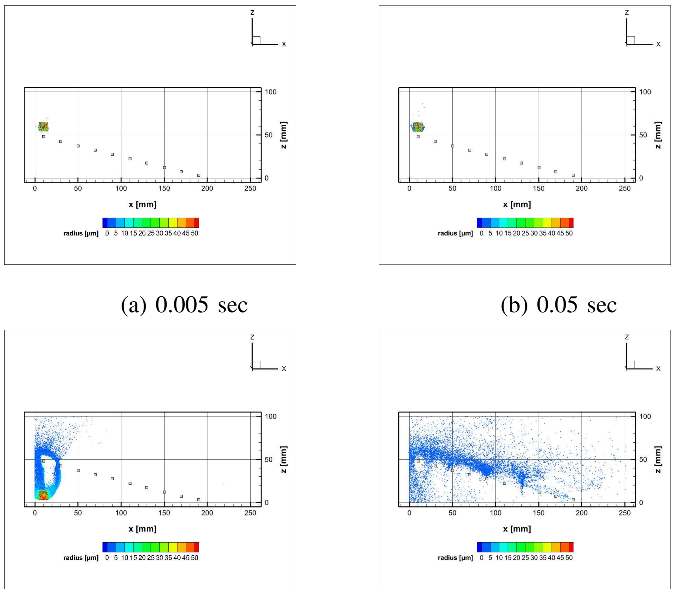
Fig. 2: Potential contours of each phase of the four-phase electrostatic traveling wave.

The varied parameters include sieve inclination angle, electrode number and dimensions, electric potential, wave frequency and phase, lunar soil properties, computational domain mesh size, gravity, phase shift, time step, particle density and size, and electrode distance. The absolute angle of inclination is used to connect the declining and inclining sieve setups.

1) Electrostatic Sieve Parameters: Some of the parameters for the cases run are shown in Table I. The acceleration due to gravity used for all four cases is that of the lunar surface (1.62 m/s^2). The voltage amplitude is the amplitude of the electrostatic traveling wave. The horizontal or x pitch is the horizontal distance between the center-lines of two consecutive electrodes. The sieve angle is the angle of inclination of the electrostatic sieve. The dust generation zone - Zmin and Zmax are the minimum and maximum positions of the dust generations on the z-axis.

2) Computational Domain: The computational domain consists of the array of electrodes, the dust generation zone, dust collection zone, and boundary conditions. In Fig. 1, the simulation and boundary conditions are depicted.

3) Potential Contours: For each case, the magnitude of the electric potential on all the electrodes is the same but the polarity is switched in order to the generate an electrostatic traveling wave. The frequency of the wave is 10 Hz. Each electrode embedded in the domain is a 2mm cube. The array of electrodes for the four example cases is declining. The electric potential contour for each phase of the wave for an example case is shown in Fig. 2. Switching from one phase to the next phase transports the particles in the desired direction.



(a) 0.005 sec

(b) 0.05 sec

(c) 0.10 sec

(d) 0.30 sec

Fig. 3: Dust particle trajectory at selected time steps. Dust grains are colored by size. Electrodes are shown as light gray boxes.

4) Motion of Charged Dust: 1 million dust particles are loaded at the initial time step into the sieve through the dust generation zone. For the four example cases, the dust generation zone is located from 5 to 15 mm along the x-axis. The minimum and maximum radii used for sampling with Accept-Reject method are 0 and 2000 μm. In order to closely model the FSJ-1 simulant and generate a large range of lunar regolith particle sizes, the mean and standard deviation of the particle sizes used are 36 and 1200 μm [31].

The lunar dust particles are injected into the computational domain, only once, at the beginning of the simulation, through the dust generation zone. Figure 3 shows the trajectory of the dust grains at different time steps. After injection, the dust grains start to move. Initially, they are above the first electrode. Afterwards, they start to loft due to the the electrostatic force created by the electrodes. The heavier particles due to the action of the gravitational force fall through the computational domain while the lighter ones travel in the direction of the electrodes and get to the dust collection zone. The particles that get to the dust collection zone are called 'classified'.

E. Results and Inference

Figure 4 shows the comparison of the particle size distribution of the collected dust grains for all four cases.

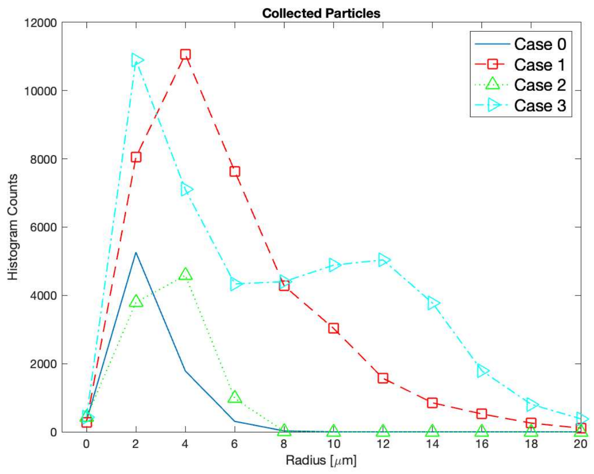


Fig. 4: The comparison of histograms of collected dust grains for four cases.

Case 0 exhibits the lowest quantity and range of classified particles. In contrast, Cases 1 and 3 show the highest quantity and range of classified particle sizes, with Case 3 demonstrating a wider range of collected particles compared to Case 1. The sieve's performance is gauged by its percentage yield, calculated as the percentage ratio of the weight of collected particles smaller than 10 μm ($W_{\text{collected},<10\mu\text{m}}$) to weight of fed particles with size less than 10 μm ($W_{\text{fed},<10\mu\text{m}}$), as can be seen in Eq. (5) [1].

$$\text{yield}(\%) = \frac{W_{\text{collected},<10\mu\text{m}}}{W_{\text{fed},<10\mu\text{m}}} \times 100\% \tag{5}$$

Table II presents the yield, average, and maximum lunar regolith radii classified. The table reveals that tripling the applied voltage amplitude results in a doubling of the average radius classified. On the other hand, tripling the electrostatic sieve angle only increases the average radius classified by a factor of approximately 1.3. This indicates that increasing the applied voltage has a significantly more pronounced effect on the yield and average radius of collected particles than increasing the sieve angle. Furthermore, a comparison of Cases 1 and 3 demonstrates that increasing the voltage amplitude beyond a certain threshold for a given sieve angle leads to a decrease in the sieve's yield. Examining the collected particle radii, it's evident that to collect particle sizes within a specific range, the voltage must be limited to a certain value. This observation aligns with the results obtained from modeling the electrostatic sieve as if it were operating on Earth [9].

TABLE II: Percentage Yield, Average and Maximum Classified Radii

	Case 0	*Case 1*	*Case 2*	*Case 3*
Yield (%)	1.062	29.938	2.641	29.031
Avg. Classified Radius (μm)	2.489	5.770	3.266	7.415
Max. Classified Radius (μm)	8.24	24.817	7.603	22.599

III. Machine Learning and Deep Learning Models

A. Approach

Our ML/DL models aim to predict the yield of an electrostatic sieve based on a variety of input parameters, including the sieve's physical configuration, operational settings, the properties of the lunar soil particles, etc. To create a comprehensive dataset, a FORTRAN-based simulator was employed to model the kinetic behavior of the sieve under varying conditions. The dataset contains over 1500 data points (referred to as configurations in the later text), each mapping a specific combination of input parameters to the resulting yield. The simulator takes five minutes to generate one data point. The yield, representing the fraction of particles below a certain size successfully collected, was normalized for consistency. Due to the diverse scales of the input parameters, preprocessing steps were taken to ensure effective model training. This includes scaling numerical features and encoding categorical variables. The dataset is then partitioned into an 80% training set and a 20% testing set, facilitating evaluation of the model's generalization performance.

B. Model Architecture and Training Strategy

We evaluate four traditional machine learning models: linear, ridge, LASSO, and polynomial regression. To this end, we propose to leverage a feedforward neural network [32] to further improve the efficiency. The neural network architecture was carefully designed and optimized through a series of experiments. These experiments explored the impact of different activation functions, network depth (number of layers), and network width (hidden layer size) on both training and test loss. The neural network architecture we use comprises several key components:

1) **Input Layer:** The input layer is dimensioned to accommodate the number of relevant input features, which captures the initial state of the electrostatic sieve configuration.

2) **Network Depth:** As shown in Figure 5, the number of layers in the network has a significant impact on model performance. A two-layer architecture appears to strike an optimal balance between model complexity and generalization ability. While a single layer shows higher loss, indicating underfitting, three layers begin to show signs of overfitting with increased test loss despite lower training loss. This guided our decision to use two hidden layers in the final architecture.
3) **Hidden Layer Size:** Figure 6 illustrates the effect of hidden layer size on model performance. The graph shows that increasing the number of neurons in the hidden layers generally improves performance up to a point, after which returns diminish and the risk of overfitting increases. Based on this, we choose a hidden layer size of 64 neurons, which provides a good trade-off between model capacity and generalization.
4) **Activation Function:** Figure 7 compares the performance of different activation functions. The results show that LeakyReLU slightly outperforms ReLU, with both achieving lower loss compared to using no activation function or the hyperbolic tangent (tanh) function. This suggests that the use of rectified linear units helps the model learn non-linear relationships in the data more effectively. As such, we selected ReLU as the activation function in our implementation.
5) **Output Layer:** A single-neuron output layer generates the final yield prediction. This neuron's output is transformed by a sigmoid function shown in equation (6) to ensure the predicted yield falls within the valid range of [0, 1].

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{6}$$

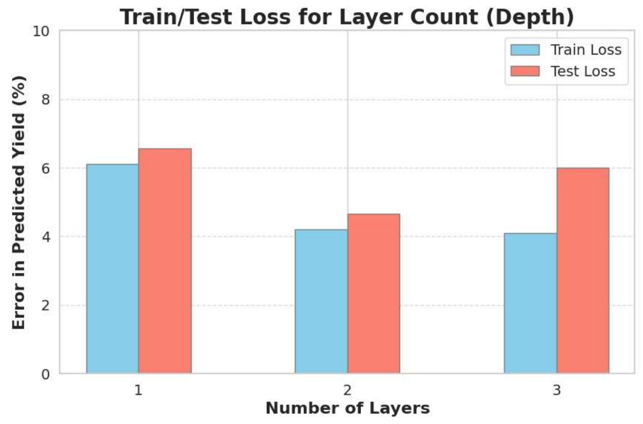


Fig. 5: Impact of Layer Count (Width of the NN) on the Neural Network Performance

C. Loss Function

To train the model effectively, we leverage the Root Mean Squared Error (RMSE) as our loss function. RMSE quantifies the average discrepancy between the model's predicted yield values and the corresponding ground-truth yield values, providing a clear metric for evaluating model performance. Equation (7) shows the mathematical formulation of our loss function.

$$L = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{7}$$

where n is the number of samples, y_i is the true yield value for the i-th sample, and $\hat{y}_i$ is the predicted yield value for the i-th sample.

D. Optimization Strategy

Optimization of the model's parameters is achieved through the Adaptive Moment Estimation (Adam) algorithm introduced by Kingma in [33]. Adam is a widely adopted optimization technique renowned for its efficiency and adaptability in training deep neural networks [34]. It intelligently adjusts learning rates for individual parameters, accelerating convergence and enhancing model performance. It combines the benefits of two other optimization methods:

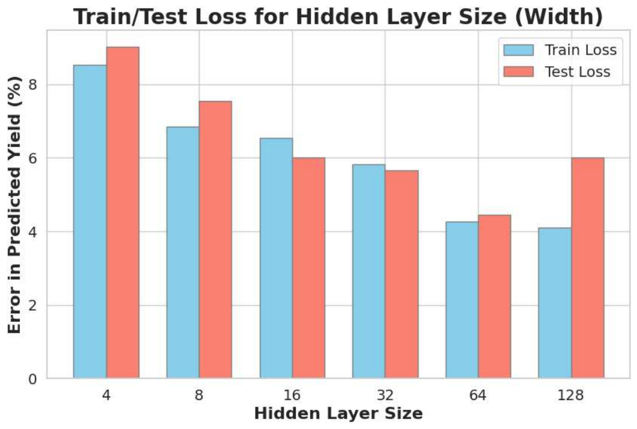


Fig. 6: Impact of Layer Size (Height of the NN) on the Neural Network Performance

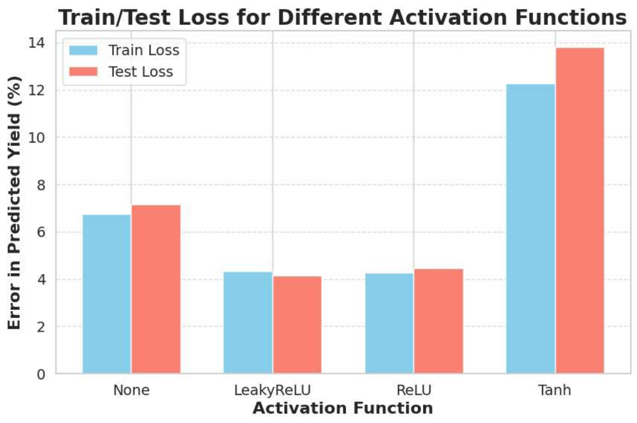


Fig. 7: Impact of Different Activation Functions on the Neural Network Performance

- **Momentum:** Accelerates optimization by considering past gradients.
- **RMSprop:** Adapts learning rates based on the magnitudes of recent gradients.

E. Results and Comparison

We showcase how our methods accelerate the electrostatic sieve for lunar regolith beneficiation using the experimental setting below. As a baseline, our vanilla approach evaluates 15,000 configurations of input parameters using the Kinetic Modeling Simulator and picks the optimal one that produces the highest yield. We further leverage an optimized approach that trains machine learning and deep learning models using only the 1,500 training data (10% of the total number of configurations), predicts the other 90% with the trained models, and then selects the configuration with the highest predicted yield. In the following, we show the trade-off in performance (in terms of execution time) and efficiency (in terms of the value of yield).

Performance: We present the performance of the Kinetic Modeling Simulator and ML/DL models in Figure 8. While the baseline approach is able to find the best configuration, it has a high runtime overhead that cannot be overlooked ($1500 * 300 = 450000$ seconds). In contrast, the inference time of the used ML/DL models is orders of magnitude lower and thus negligible compared with the simulation time. This leads to a $10\times$ reduction in the time for finding an acceptable configuration using the optimized approach with any of the models.

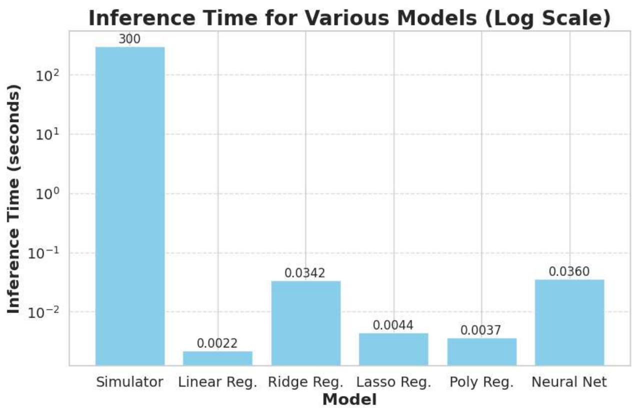


Fig. 8: Comparison of Inference Time of ML/DL models with the Kinetic Modeling Simulator

Efficiency: We then present the accuracy of different models in the optimized approach and compare their efficiency. The training and test losses are shown in Figure 9. According to the figure, it is observed that the neural network produced the best results. Looking at the magnitudes of the model coefficients we can interpret that the most significant predictors of yield were the absolute angle of inclination, number of electrodes, voltage magnitude, electrode dimension, dist. dust generation zone to first electrode, and gravitational force.

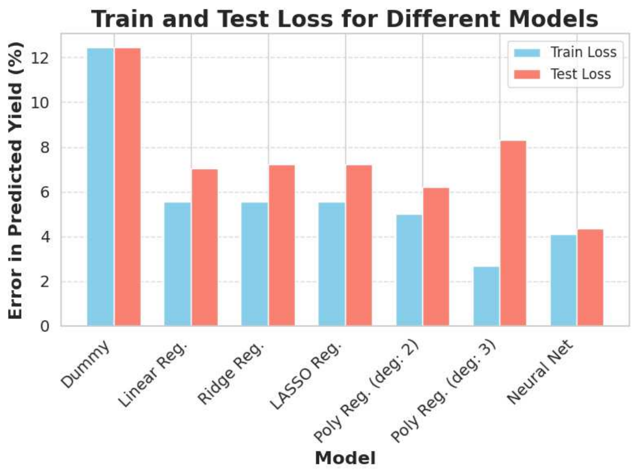


Fig. 9: Comparison of Deep and Machine Learning Models

We then integrate the deep learning model into the optimized approach and compare its efficiency with the baseline. Specifically, we use this model to predict the yield for all 15,000 candidate configurations evaluated in the baseline and identify the highest predicted yield among them. Using this approach, we obtain a predicted yield of 33.79% with the following parameter values:

- absolute angle of inclination = 335°
- number of electrodes = 4
- voltage magnitude = 2000 $volts$
- voltage frequency = 5 Hz
- electrode dimension = 6 mm
- x pitch = 20 mm

This produces an actual yield of 38.54% when running another Kinetic Modeling simulation using these parameters.

As a comparison, exhausting all the 15,000 combinations leads to an optimal yield of 38.79%. This result demonstrates that the proposed approach is able to produce a near-optimal configuration with significantly reduced runtime.

IV. Conclusion

This study demonstrates the potential of neural networks on the electrostatic sieve for lunar regolith beneficiation. They exhibit better quality on yield prediction than traditional machine learning methods such as regression. Their ability to learn complex relationships surpasses linear and nonlinear regression, offering a more accurate representation of the underlying relationship. We intend to further explore physics-informed neural networks, which promise even better performance on unseen data due to their adherence to fundamental laws. Ultimately, these advanced techniques serve as powerful surrogate models, drastically reducing the time and resources required for scientific experimentation. By combining the speed of machine learning and deep learning with the precision of established scientific methods, we pave the way for accelerated optimization and innovation in this field. As validated in our evaluation, the DL model leads to almost the same yield compared with exhaustive simulation while reducing the runtime by $10\times$. This significantly reduces the time and resources for the electrostatic sieve.

Acknowledgment

This research is sponsored by the National Science Foundation under Grant No. OAC-2306184 with the University of North Texas and Grant No. OAC-2330364 with the University of Kentucky and NASA Lunar Surface Technology Research (LuSTR) program. The simulations presented here were carried out with computing resources provided by the Center for High Performance Computing Research at Missouri University of Science and Technology through an NSF grant OAC-1919789.

References

[1] H. Kawamoto, H. Morooka, and H. Nozaki, "Improved electrodynamic particle-size sorting system for lunar regolith," *Journal of Aerospace Engineering*, vol. 35, no. 1, p. 04021115, 2022.

[2] H. Kawamoto and M. Adachi, "Electrostatic particle-size classification of lunar regolith for in-situ resource utilization," in *AIAA SciTech Forum 2014*, no. AIAA 2014-0341, (National Harbor, Maryland), 2014.

[3] H. Kawamoto, H. Morooka, and H. Nozaki, "Vertical transport of lunar regolith and ice particles using electrodynamic traveling wave," *Journal of Aerospace Engineering*, vol. 34, no. 4, p. 04021042, 2021.

[4] R. J. Williams, D. S. McKay, D. Giles, and T. E. Bunch, "Mining and beneficiation of lunar ores," Tech. Rep. V-6, NASA, 1979.

[5] W. N. Agosto, "Electrostatic concentration of lunar soil minerals," in *Lunar Bases and Space Activities of the 21st Century* (W. W. Mendell, ed.), (Houston, TX), p. p.453, 1985.

[6] J. Zhao, X. He, G. Yan, and D. Han, "Kinetic Particle Simulations of Plasma and Dust Environments at Robotic Construction Sites near the Lunar Terminator," *Journal of Aerospace Engineering*, vol. 35, p. 04022095, November 2022.

[7] J. Zhao, D. Lund, and D. Han, "Development of a fully kinetic particle simulation code for coupled plasma-dust transport," in *16th Spacecraft Charging and Technology Conference (SCTC)*, SCTC 2022-075, (Virtual), April 4-8, 2022.

[8] J. Zhao, G. Yan, X. He, and D. Han, "Kinetic Particle Simulations of Plasma Charging and Dust Transport near Uneven Lunar Surface Terrain," in *AIAA SciTech 2022*, AIAA 2022-1988, (San Diego, CA & Virtual), January 3-7, 2022.

[9] E. Ingram, A. Eze, J. Smith, F. Rezaei, D. Bayless, W. Schonberg, and D. Han, "Kinetic modeling of electrostatic sieving for lunar regolith beneficiation: Case studies," in *AIAA SciTech Forum*, AIAA 24-2540, (Orlando, Florida), January 2024.

[10] N. Shlezinger, Y. C. Eldar, and S. P. Boyd, "Model-based deep learning: On the intersection of deep learning and optimization," *IEEE Access*, vol. 10, pp. 115384–115398, 2022.

[11] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An introductory review of deep learning for prediction models with big data," *Frontiers in Artificial Intelligence*, vol. 3, p. 4, 2020.

[12] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 1058–1066, PMLR, 17–19 Jun 2013.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[14] K. Vadnerkar and M. E. Patil, "Design and implementation of intelligent recommendation system for farmers using cnn and rnn," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 7, 2018.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 778–784, 2014.

[17] J. Park, Y. Liu, K. D. Kihm, and L. A. Taylor, "Characterization of lunar dust for toxicological studies i: Particle size distribution," *Journal of Aerospace Engineering*, vol. 21, no. 4, pp. 266–271, 2008.

[18] L. Martino, D. Luengo, and J. Miguez, *Independent Random Sampling Methods*. Springer International Publishing, 2018.

[19] R. Kafafy and J. Wang, "Whole subscale ion optics simulation: Direct ion impingement and electron backstreaming," in *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA 2005-3691, (Tucson, Arizona), July 2005.

[20] R. Kafafy and J. Wang, "Whole ion optics gridlet simulations using a hybrid-grid immersed-finite-element particle-in-cell code," *Journal of Propulsion and Power*, vol. 23, pp. 59–68, January-February 2007.

[21] R. I. Kafafy and J. Wang, "A hybrid grid immersed finite element particle-in-cell algorithm for modeling spacecraft-plasma interactions," *IEEE Transactions on Plasma Science*, vol. 34, pp. 2114–2124, October 2006.

[22] J. Wang, Y. Cao, R. Kafafy, J. Pierru, and V. K. Decyk, "Simulations of ion thruster plume-spacecraft interactions on parallel supercomputer," *IEEE Transactions on Plasma Science*, vol. 34, pp. 2148–2158, October 2006.

[23] D. Han and J. J. Wang, "Simulations of ion thruster plume contamination with a whole grid sputtered mo source model," in *49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA 2013-3888, (San Jose, California), July 2013.

[24] D. Han, *Particle-in-Cell Simulations of Plasma Interactions with Asteroidal and Lunar Surfaces*. PhD thesis, University of Southern California, 2015.

[25] D. Han, P. Wang, X. He, T. Lin, and J. Wang, "A 3D immersed finite element method with non-homogeneous interface flux jump for applications in particle-in-cell simulations of plasma-lunar surface interactions," *Journal of Computational Physics*, vol. 321, pp. 965–980, September 2016.

[26] D. Han, J. Wang, and X. He, "A Nonhomogeneous Immersed-Finite-Element Particle-in-Cell Method for Modeling Dielectric Surface Charging in Plasmas," *IEEE Transactions on Plasma Science*, vol. 44, pp. 1326–1332, August 2016.

[27] D. Han, J. Wang, and X. He, " Immersed Finite Element Particle-in-Cell Simulations of Plasma Charging at the Lunar Terminator," *Journal of Spacecraft and Rockets*, vol. 55, pp. 1490–1497, November-December 2018.

[28] D. Han and J. Wang, "3-D Fully-Kinetic Particle-in-Cell Simulations of Small Asteroid Charging in the Solar Wind," *IEEE Transactions on Plasma Science*, vol. 47, pp. 3682–3688, August 2019.

[29] W. Yu, D. Han, and J. Wang, "Numerical Simulations of Dust Dynamics Around Small Asteroids," *IEEE Transactions on Plasma Science*, vol. 47, pp. 3724–3730, August 2019.

[30] G. H. Heiken, D. T. Vaniman, and B. M. French, *Lunar Sourcebook: A User's Guide to the Moon*. Cambridge, England: Cambridge University Press, 1991.

[31] Shimizu, "FJS-1 / FJS-1g Shimizu Lunar Soil Simulant." https://www.shimz.co.jp/en/, 2022.

[32] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53040–53065, 2019.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] S. Bock and M. Weiß, "A proof of local convergence for the adam optimizer," in *2019 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2019.

ARTIFACT DESCRIPTION/EVALUATION APPENDIX

A. Summary of the Experiments Reported

1) Abstract: We provide this artifact appendix to make our results reproducible. We measure two main results: i) The inference time of models in seconds, and ii) The model performance through Root Mean Squared (RMSE) values.

2) Artifacts: The GitHub Link to the source code: https://github.com/Kalpit-Vadnerkar/DL-GPU-Electrostatic-Sieve

B. Experimental Setup

1) Relevant Hardware Details:

- Tesla V100-PCIE-16GB
- Driver Version: 550.54.15

2) Operating Systems and Versions:

- Operation System: Rocky Linux
- version: 8.8 (Green Obsidian)

3) Compilers and Versions:

- GCC 11.2
- Python 3.11.10

4) Libraries and Versions:

- pandas: 2.2.3
- scikit-learn: 1.5.2
- pytorch: 2.3.0
- pytorch-cuda: 11.8
- numpy: 2.0.0
- matplotlib: 3.9.2

5) Input Datasets and Versions: Over 15000 data points were generated and are stored across multiple .txt files in the ”Dataset” folder within the project repository.

6) Other Installation Software:

- CUDA Version: 12.4

C. Evaluation Experiments

This section details the experiments performed for evaluating electrostatic sieve configurations and developing predictive models for sieve performance. The code for all experiments is available in the accompanying Jupyter notebooks. The $'Optimization.ipynb'$ notebook goes through the entire dataset analyzing and identifying the best electrostatic sieve configurations. Finally it displays the filtered data with configurations providing the maximum yields.

*1) **Experiment 1**:* The $'DataProcessing.ipynb'$ notebook covers the following steps:

- Data Loading and Visualization: The code reads and loads the data from the ”Dataset” folder. Numerical features are visualized using histograms to understand their distributions.
- Data Splitting and Preprocessing: The dataset is shuffled and split into an 80-20 train-test split. A data processing pipeline is constructed to encode the data using sklearn’s RobustScaler for numerical features and OrdinalEncoder for categorical features. This ensures robust handling of outliers and prepares the data for machine learning models.
- Regression Model Evaluation: Multiple regression models are trained and evaluated using the Root Mean Squared Error (RMSE) as the performance metric. This includes both linear and non-linear models.
- Hyperparameter Tuning: Grid search and cross-validation techniques are employed to identify the optimal hyperparameters for the non-linear regression models, ensuring the models are well-tuned to the data.
- Inference Time Measurement: The code measures the inference time for each model to predict a single data point, providing insights into the computational efficiency of different models.

*2) **Experiment 2**:* The $'MPL.ipynb'$ notebook focuses on developing a deep learning model for prediction:

- Data Preparation: A random subset of 10% of the entire dataset is selected to manage computational resources during deep learning training.
- Custom Dataset and Dataloader: A CustomDataset class is implemented to create a dataloader, facilitating efficient batching and shuffling of data for the neural network.
- Model Training: A feed-forward neural network is initialized and trained for 100 epochs with a batch size of 32. The RMSE serves as the loss function for optimization.
- Architecture Search and Cross-Validation: Different neural network architectures are explored, varying the number of layers and neurons (width and depth). Cross-validation is used to evaluate the performance of each architecture and prevent overfitting.
- Model Selection: The notebook provides summaries of two promising architectures that achieved similar performance, highlighting the trade-offs between model complexity and predictive accuracy. The best-performing architecture is presented in the main paper.

Two interesting feed-forward Neural Network architectures that produced similar results are summarized below:

- Model Summary 1:
  - Structure: This model has three layers: An input layer with 9 features (think of these as 9 input variables). A hidden layer with 64 neurons. An output layer with 1 neuron (suggesting this model predicts a single value).
  - Activation Function: ReLU (Rectified Linear Unit) is used to introduce non-linearity. ReLU sets any negative input values to 0 and keeps positive values unchanged. This helps the network learn complex patterns.

```
    Net(
(layers): ModuleList(
  (0): Linear(in_features=9,
              out_features=64,
              bias=True)
  (1): Linear(in_features=64,
              out_features=64,
                bias=True)
      (2): Linear(in_features=64,
                out_features=1,
                bias=True)
    )
    (activation_function):
          ReLU()
  )
```

- Model Summary 2:
  - Structure: This model has four layers: An input layer with 9 features (same as Model 1). Three hidden layers with 32, 16, and 8 neurons respectively. This creates a more gradual decrease in the number of neurons compared to Model 1. An output layer with 1 neuron.
  - Activation Function: LeakyReLU is a variation of ReLU. Instead of setting negative values to 0, LeakyReLU allows a small, non-zero gradient for negative inputs. This can sometimes help prevent issues that can occur with ReLU.

```
          Net(
    (layers): ModuleList(
      (0): Linear(in_features=9,
                out_features=32,
                bias=True)
      (1): Linear(in_features=32,
                out_features=16,
                bias=True)
      (2): Linear(in_features=16,
                out_features=8,
                bias=True)
      (3): Linear(in_features=8,
                out_features=1,
                bias=True)
    )
    (activation_function):
          LeakyReLU(negative_slope=0.01)
  )
```