Machine learning augmented NEGF simulation engine

By **Ritesh Das**

Under supervision of **Dr. Aniket Singha**

Department of Electronics and Electrical Communications Engineering

Indian Institute of Technology, Kharagpur

Introduction

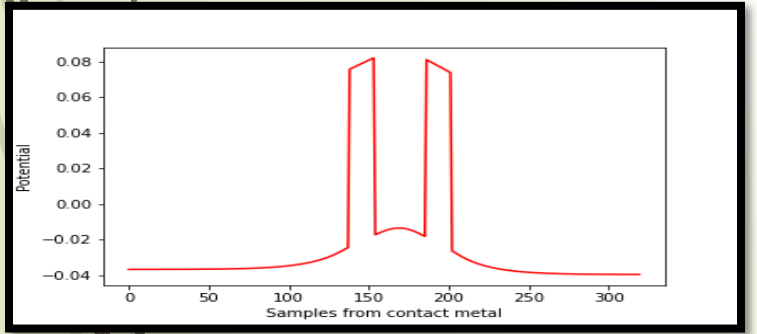
- The **Non-Equilibrium Green's Function (NEGF)** formalism provides a powerful conceptual and computational framework for treating quantum transport in nanodevices.

- Simulations to be done are of multi-junction semiconductor devices of the nanoscale range.

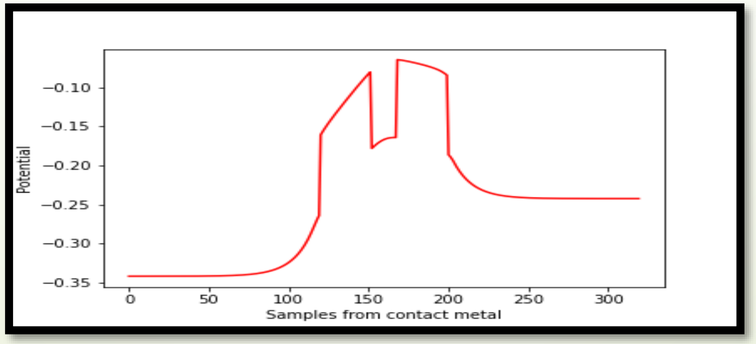
- As input to the simulator, we provide **i) Applied bias voltage across the device**, **ii) Doping concentration profile from 0 to 80nm at steps of 0.25nm** and **iii) The flat band potential profile from 0 to 80nm at steps of 0.25 nm**.

- The simulator gives us the **Potential profile of the device from 0 to 80nm at steps of 0.25nm** as the output.

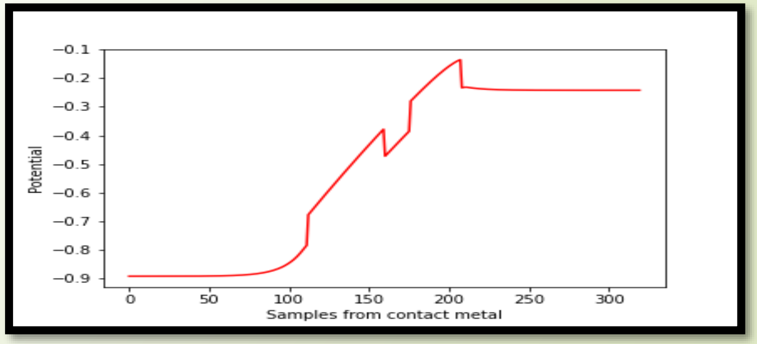
- A fully-quantum-mechanical simulator which does repeated matrix calculations with several energy integration solvers to get carrier density, using Poisson's equation. Process of getting the final profile is iterative in nature and requires the simulator to reach a convergence point/condition.



Applied Bias Voltage = 0V

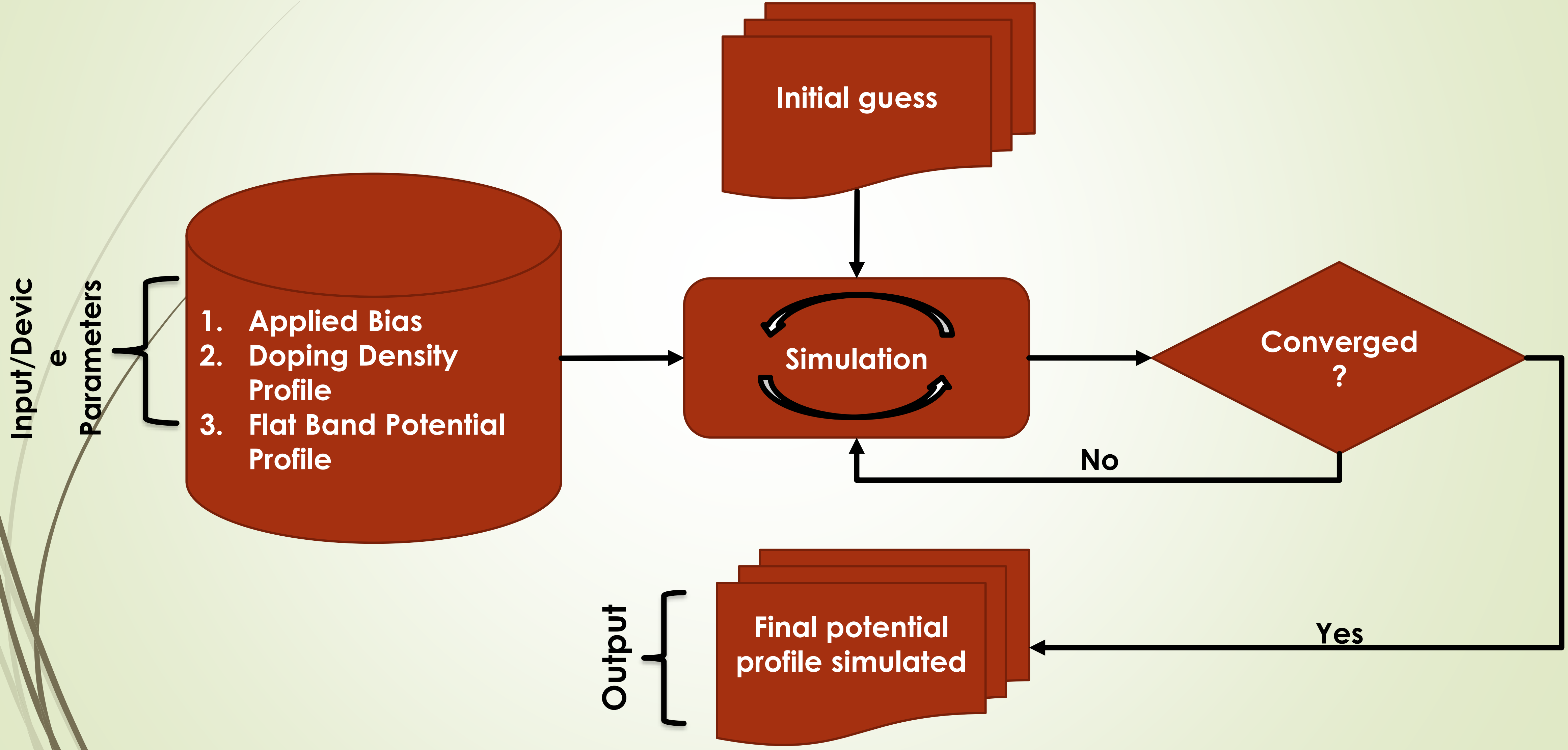


Applied Bias Voltage = 0.1V

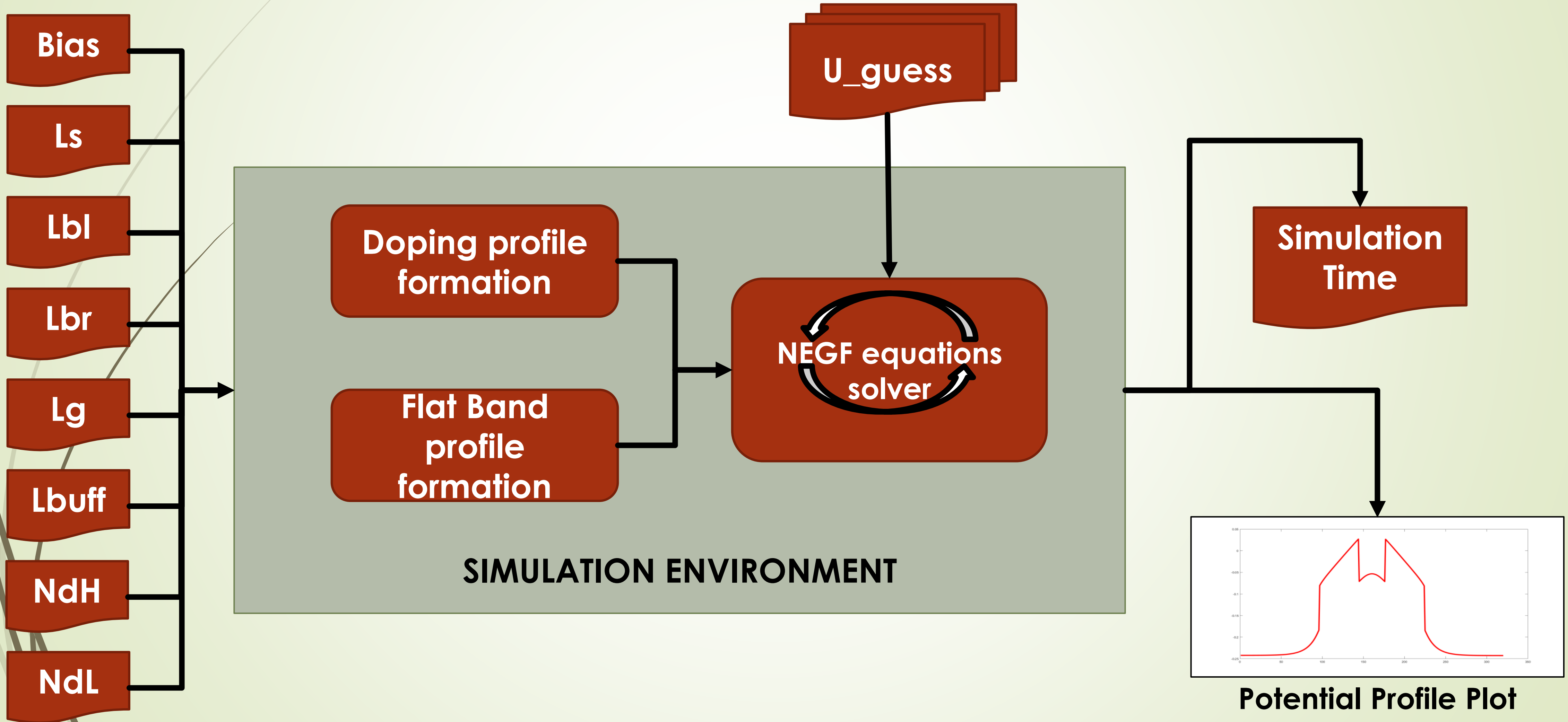


Applied Bias Voltage = 0.65V

CURRENT SCENARIO



THE SIMULATION PROCESS FLOW



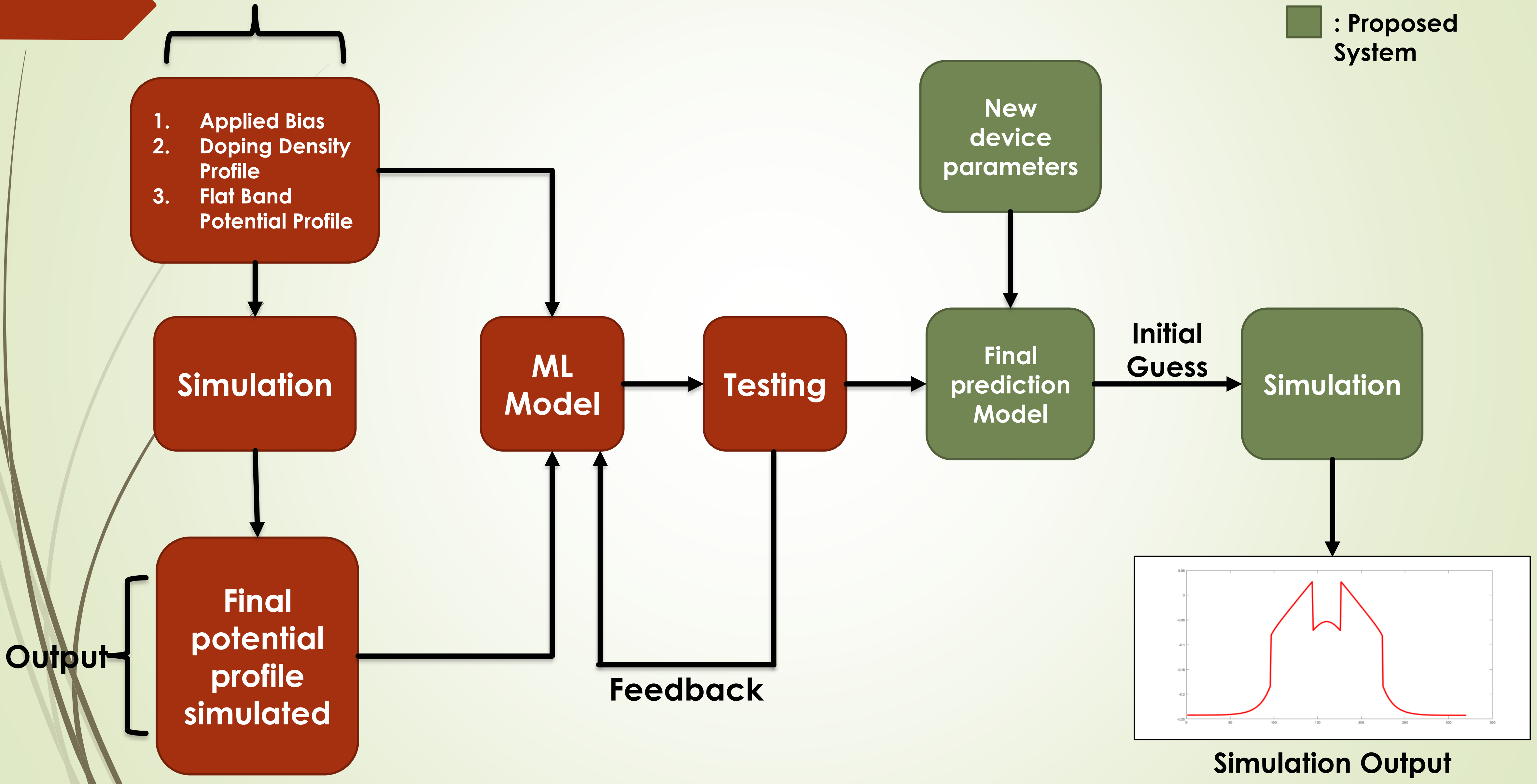
Problem Statement and Solution Proposed

- NEGF Framework, although captures the microscopic phenomenon is computationally expensive.
- Simulating a single sample of data took around $10^2 - 10^5$ seconds and made use of GPU resources extensively.
- Even so there were some cases, where it failed to reach convergence altogether!
- This is a huge hindrance for research and development making use of device simulation using NEGF.

- Tong Wu and Jing Guo, in their paper "Speed up quantum transport device simulation on Ferro-electric tunnel junction with Machine Learning methods", have used various regression models on heavily feature engineered data to predict device characteristics with very high accuracy.

- We shall make such a model to predict said "Potential profile" of the device from the inputs and feed that as an initial condition of the device characteristics to the NEGF simulator for faster and more efficient convergence to actual characteristics.

PROPOSED SYSTEM



Pre-processing the Dataset

The Dataset looked like this →

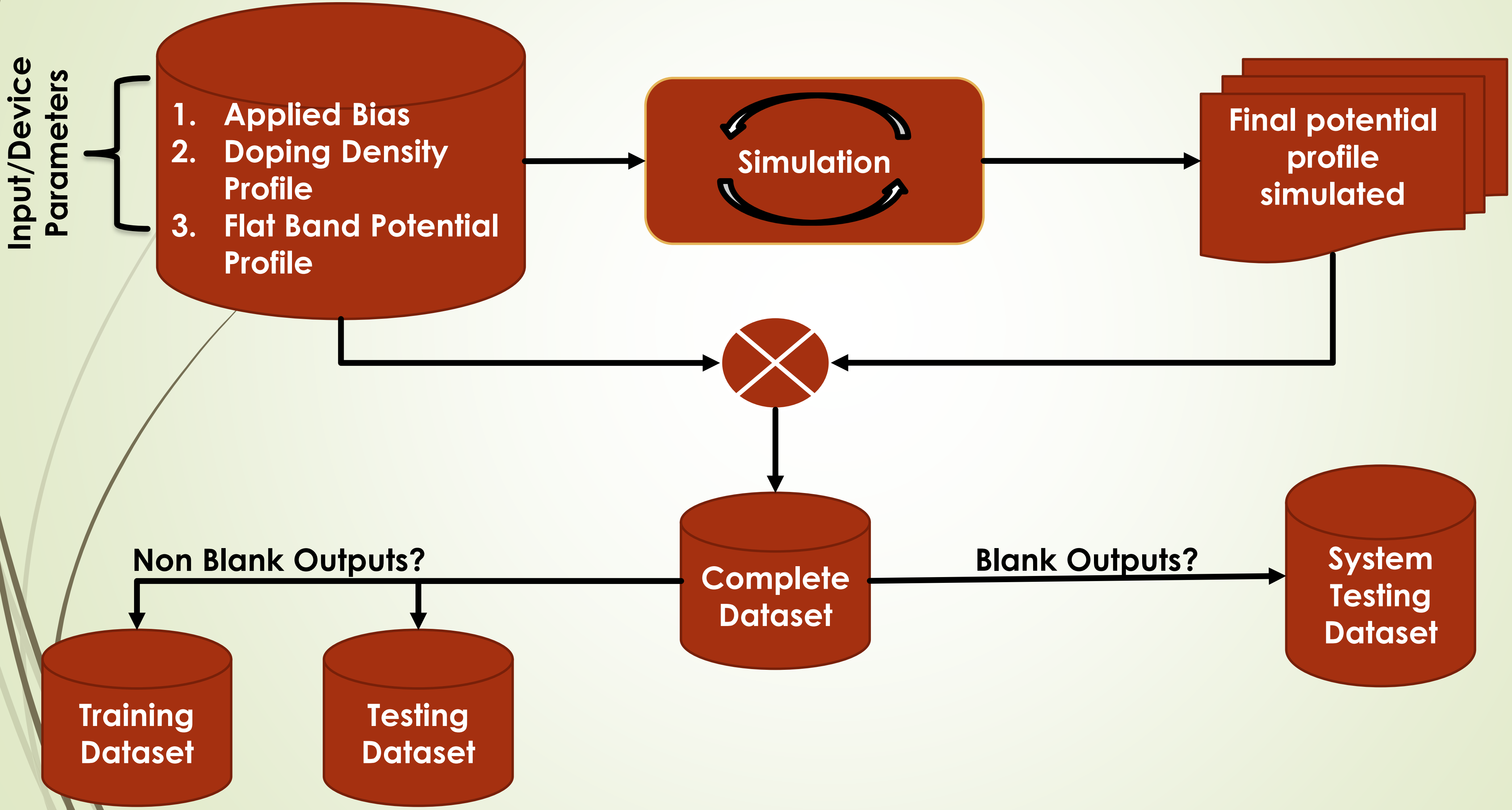
Applied Bias	Doping Profile	Flat Band Profile	Potential Profile
0.65 V	[1 x 320] vector	[1 x 320] vector	[1 x 320] vector

1. Removing Non-converging samples and placing them in a Simulation testing set
2. Making 12 prototype functions of Applied Bias voltage feature
3. Normalizing Doping concentration profile values
4. Making [Flat Band – Potential] profile feature
5. Splitting in Train and Test sets

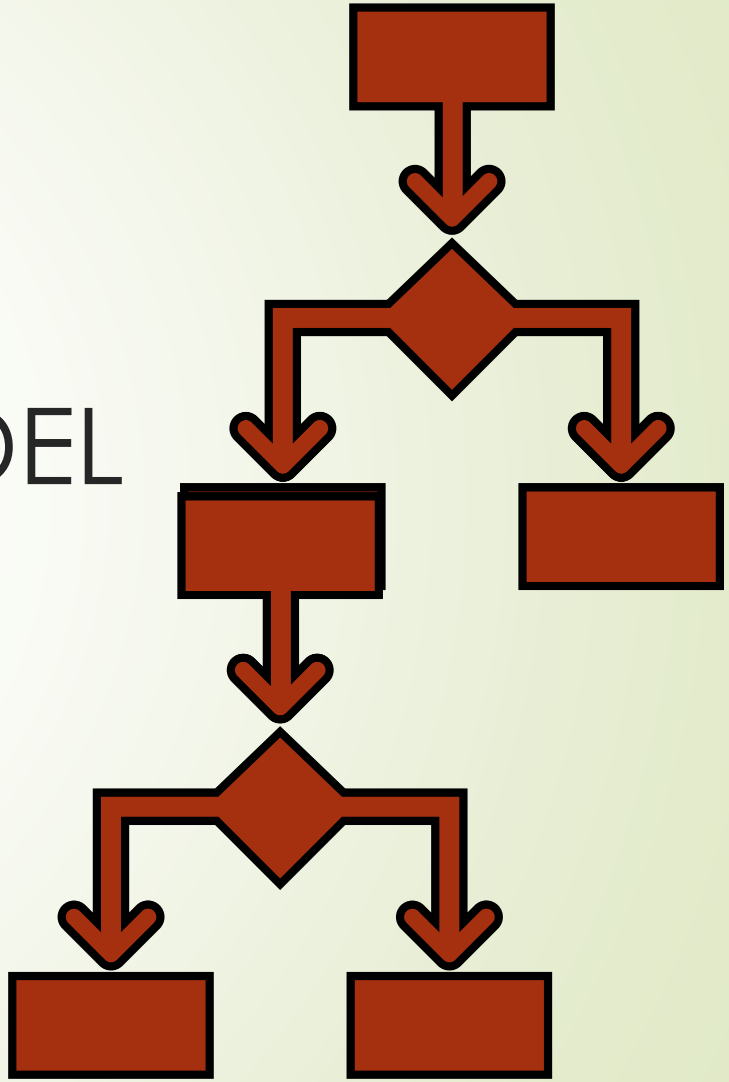
The 12 prototype functions of X feature were →

$$[X, X^{-1}, X^{0.5}, X^{-0.5}, X^2, X^{-2}, X^3, X^{-3}, \ln(X), (\ln(X))^{-1}, e^X, e^{-X}]$$

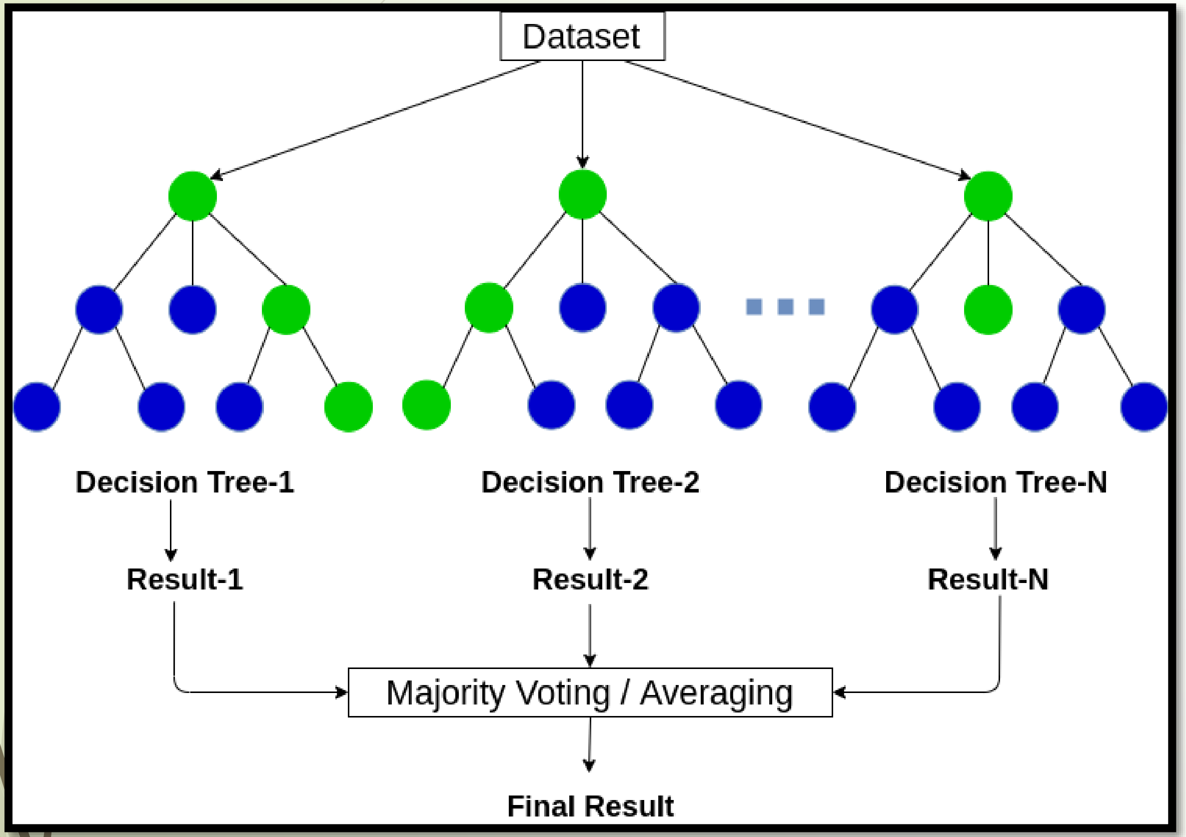
Making the datasets



Random FOREST MODEL



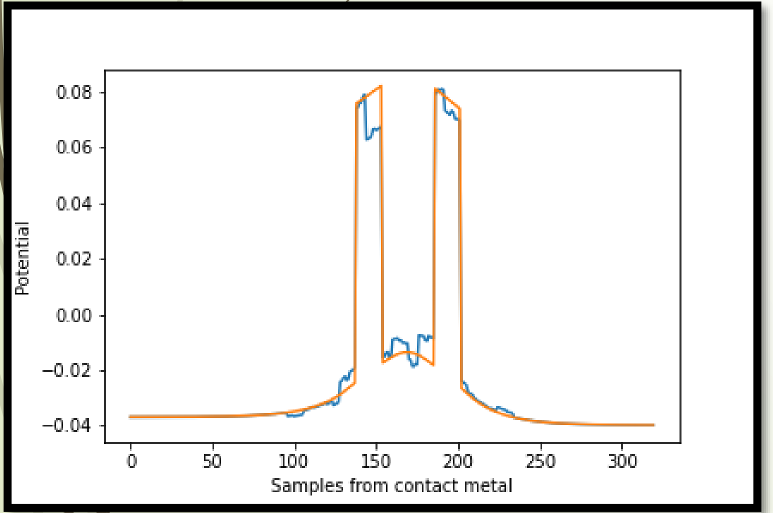
Random Forest Model - Implementation



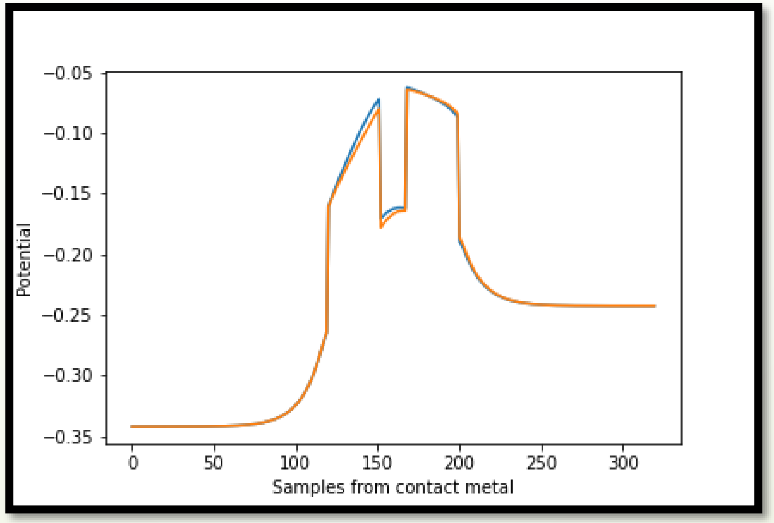
- **N_estimators** = 500 trees.
- "**Squared_error**" cost criterion.
- **Bootstrap** and **Out-of-Bag** score set to true with a random_state of 3.
- **Verbose** set to **3** and **n_jobs** set to take in all available threads for processing.

Random Forest Model - Results

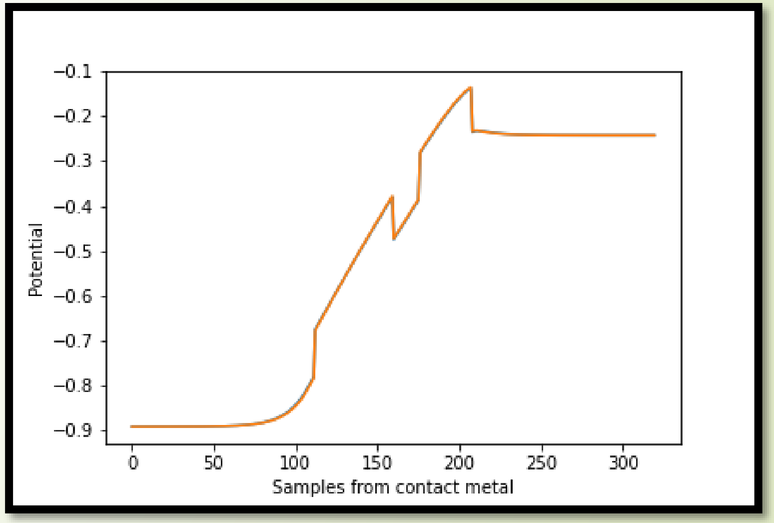
Set	Mean Absolute Error
Training set	0.0019165
Test set	0.0115169



Applied Bias Voltage = 0V

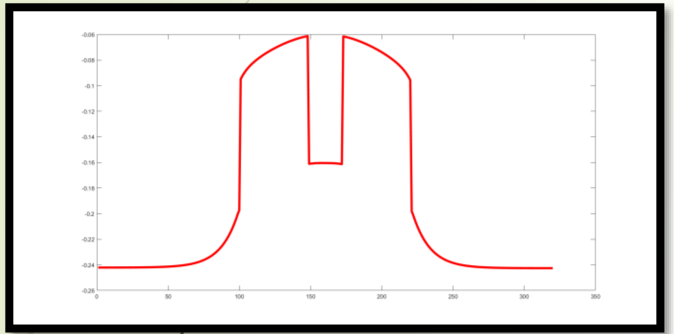


Applied Bias Voltage = 0.1V

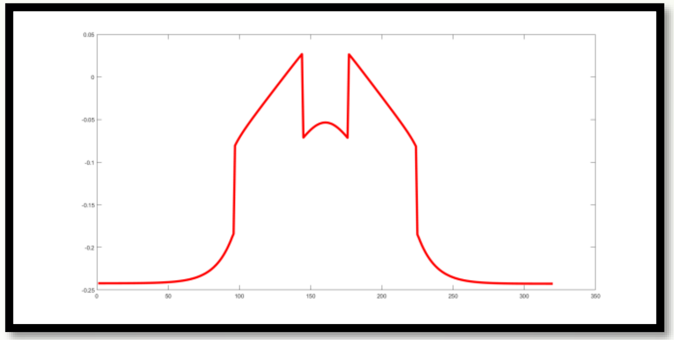


Applied Bias Voltage = 0.65V

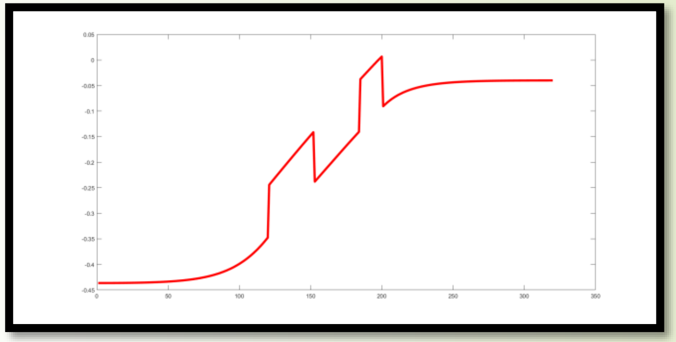
Random Forest Model – SIMULATION Results



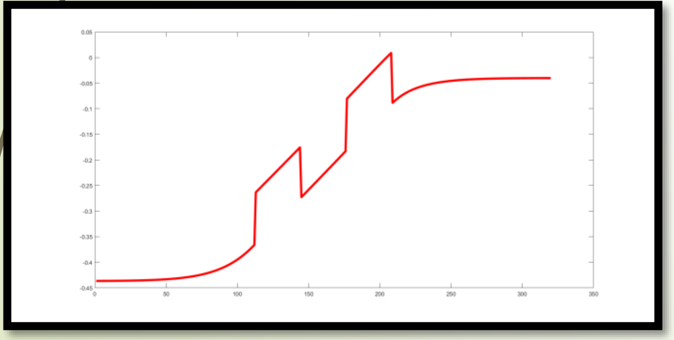
Device 1



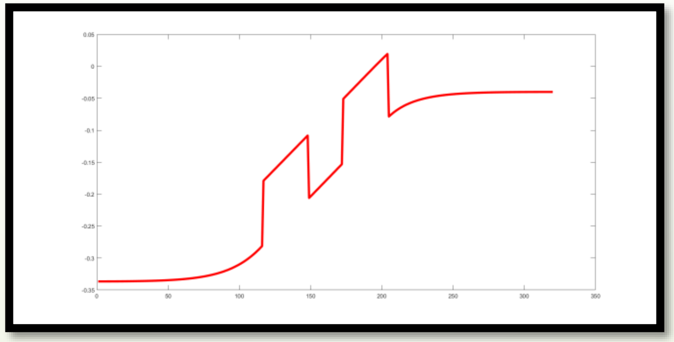
Device 2



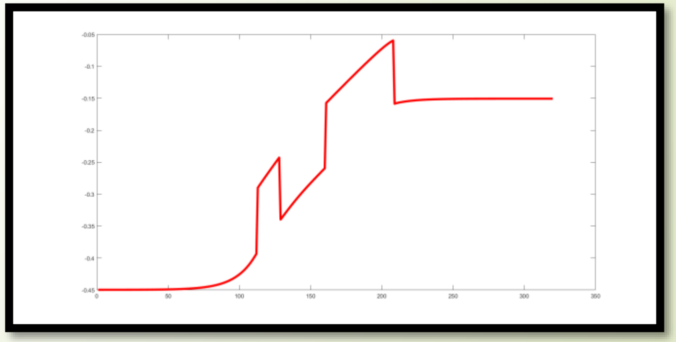
Device 3



Device 4



Device 5



Device 6

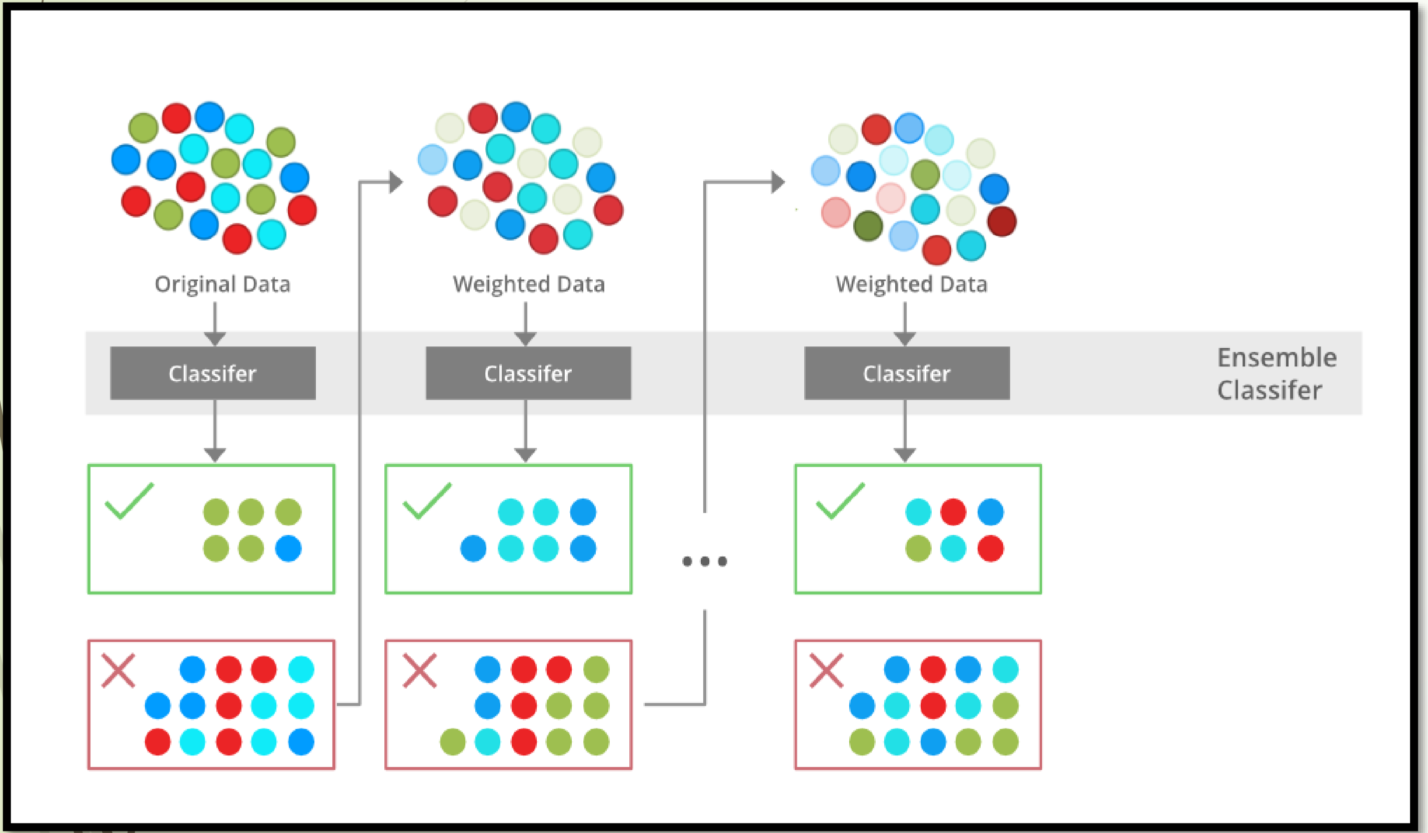
Random Forest Model – SIMULATION TIME

Device characteristics number	Simulation time (seconds)
Device 1	1086.252
Device 2	351.682
Device 3	218.322
Device 4	245.566
Device 5	137.86
Device 6	156.497

For every device data, we simulated the output by using the predicted [Flat Band – Potential] profile output from our Random forest and fed it as U_guess to our simulator.

XGBOOST MODEL

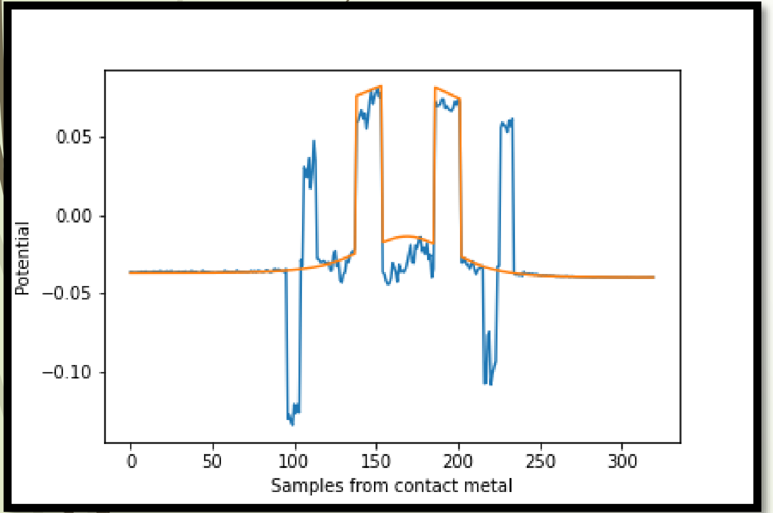
XGBoost Model - Implementation



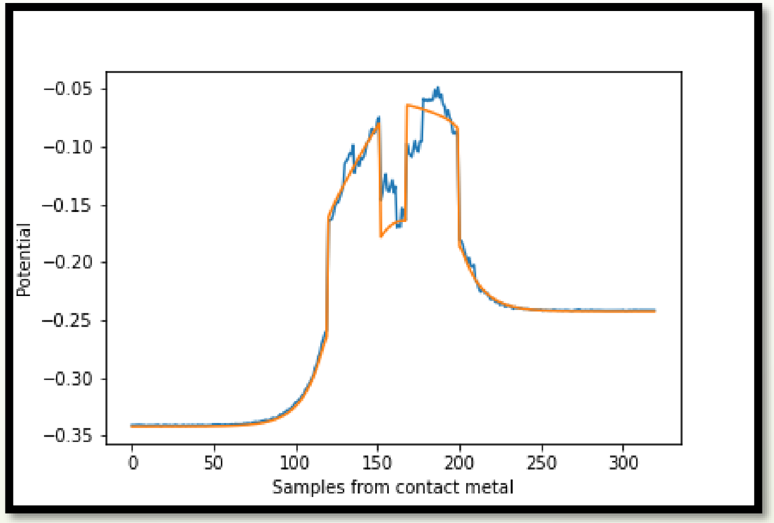
- **N_estimators** = 500 trees.
- **Learning_rate** of **0.1** at each stage of boosting.
- **Subsampling** set for **0.6** for increasing generalization of dataset.
- **N_threads** set to **-1** for using all available threads.
- **MultiOutputRegressor** for creating a multi-to-multi mapping of the model.

XGBoost Model - Results

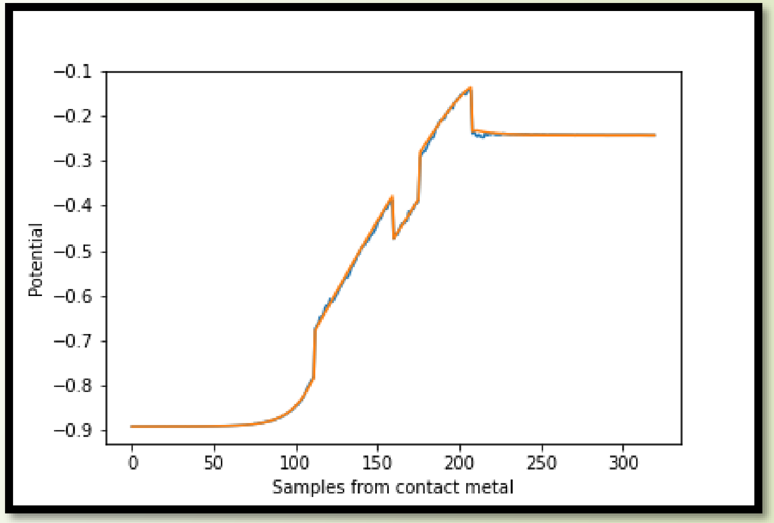
Set	Mean Absolute Error
Training set	0.0033216
Test set	0.0112141



Applied Bias Voltage = 0V

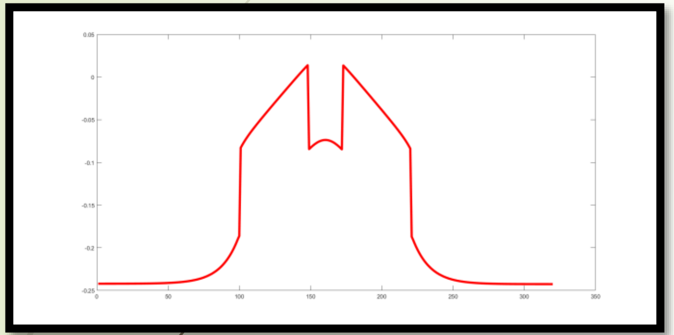


Applied Bias Voltage = 0.1V

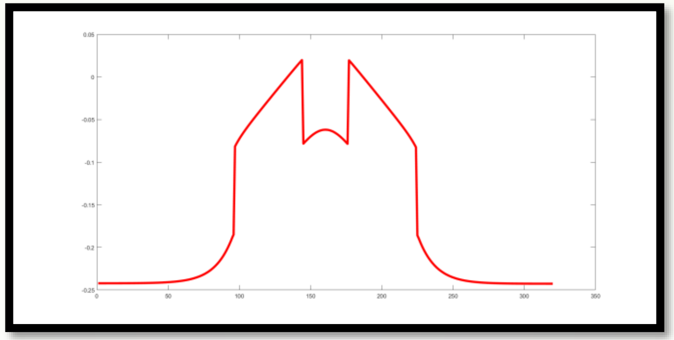


Applied Bias Voltage = 0.65V

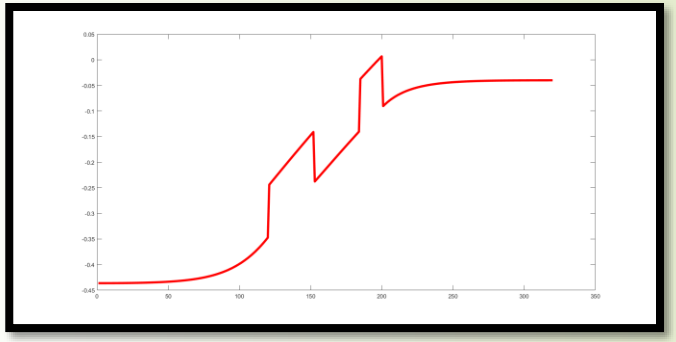
XGBOOST – SIMULATION Results



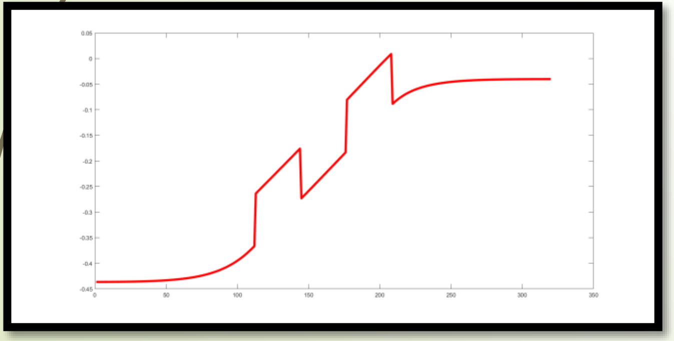
Device 1



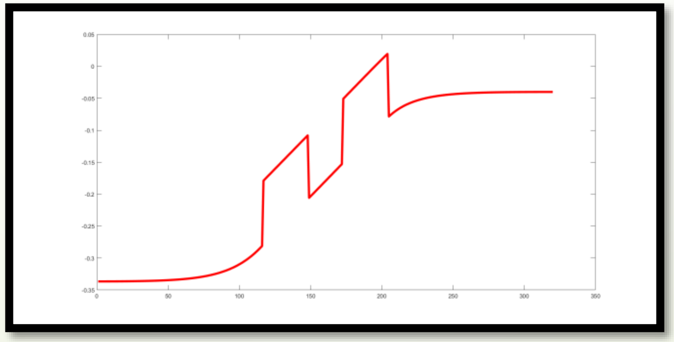
Device 2



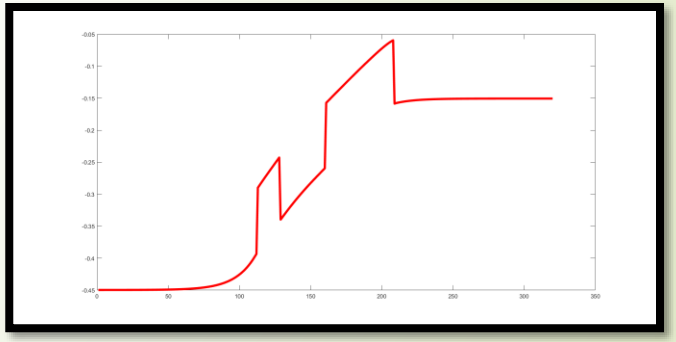
Device 3



Device 4



Device 5



Device 6

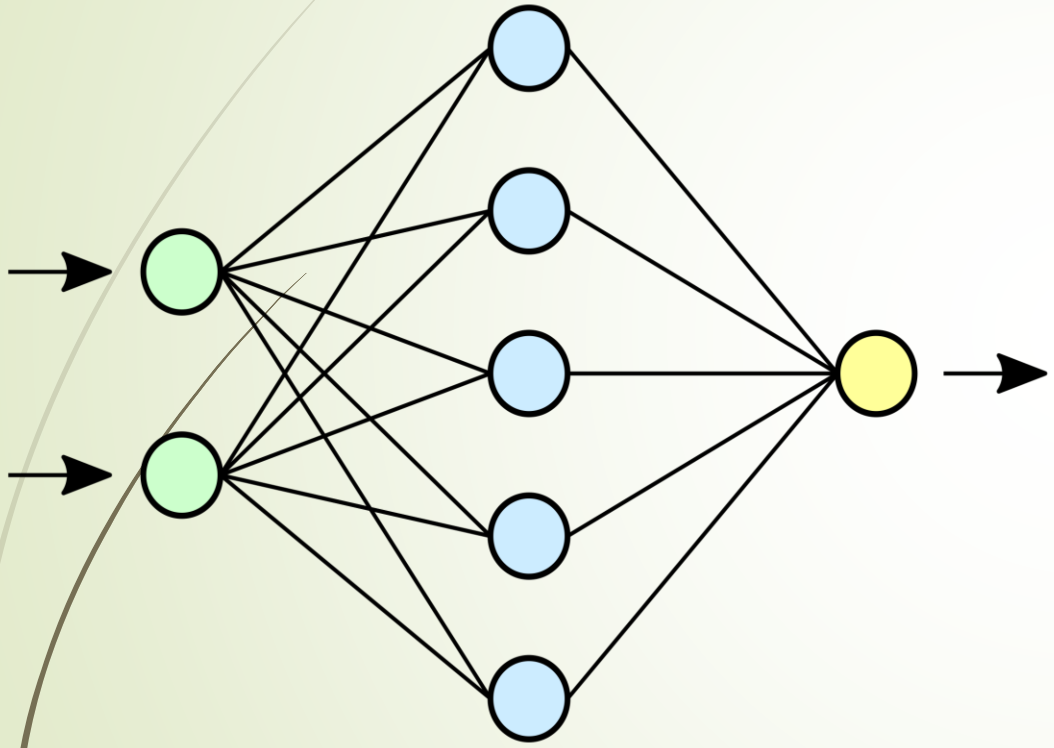
XGBOOST – SIMULATION TIME

Device characteristics number	Simulation time (seconds)
Device 1	1198.152
Device 2	997.101
Device 3	261.791
Device 4	292.663
Device 5	161.488
Device 6	180.461

For every device data, we simulated the output by using the predicted [Flat Band – Potential] profile output from our XGBoost and fed it as U_guess to our simulator.

Neural network MODEL

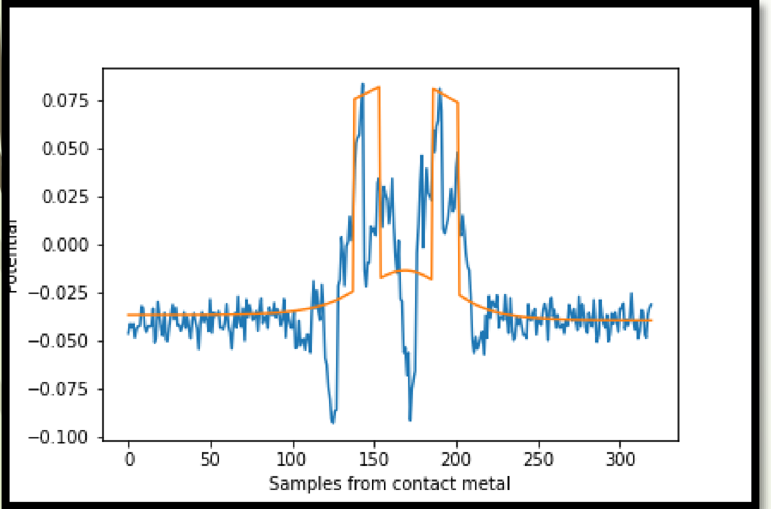
Neural Network Model - Implementation



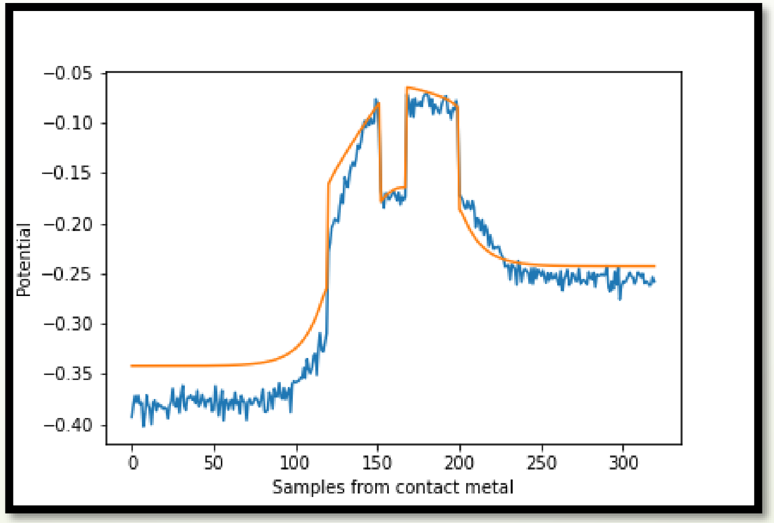
- **Sequential layers** of **652 → 512 → 512 → 320** perceptrons.
- **ReLU** activation functions in between perceptrons.
- Loss function of "**Mean Squared Error**".
- **Adam** optimizer with a **learning rate** of **0.0005** for weight optimization of perceptrons.
- **55 epochs** with a training batch size of **10 samples**.
- Early stopping condition when **training_loss < 0.0015** and **r1_score > 0.95**.

Neural Network Model - Results

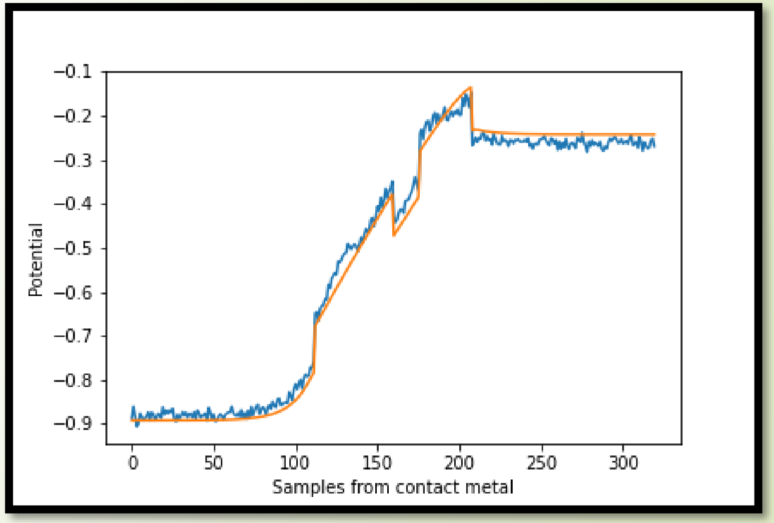
Set	Mean Absolute Error
Training set	0.0226036
Test set	0.0263556



Applied Bias Voltage = 0V

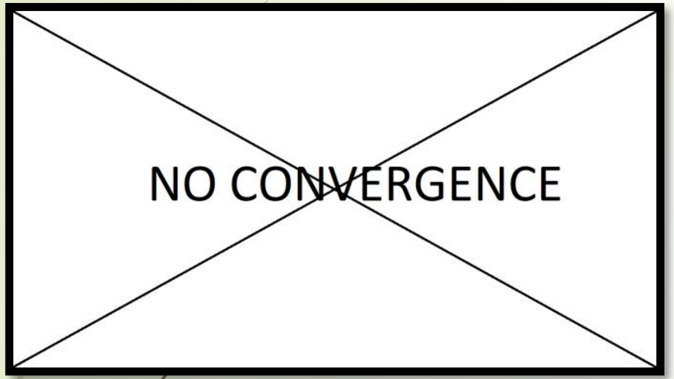


Applied Bias Voltage = 0.1V

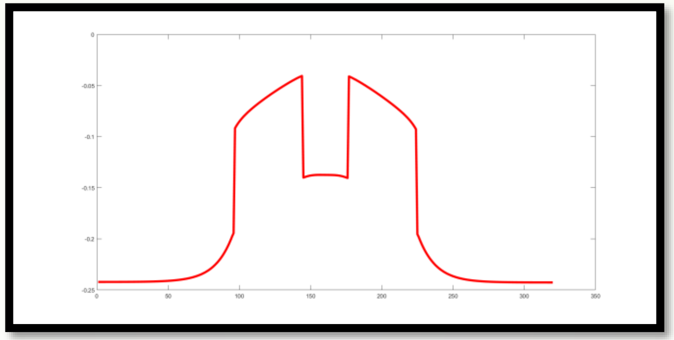


Applied Bias Voltage = 0.65V

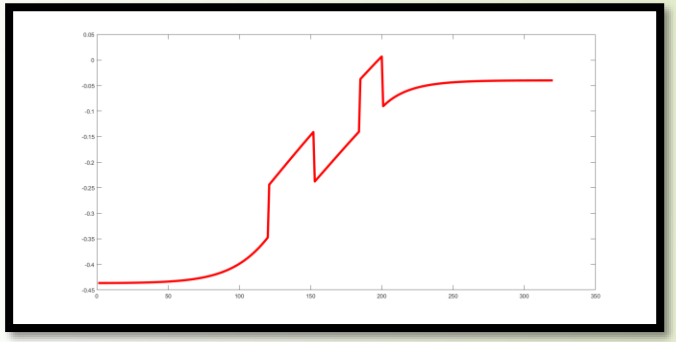
NEURAL NETWORK – SIMULATION Results



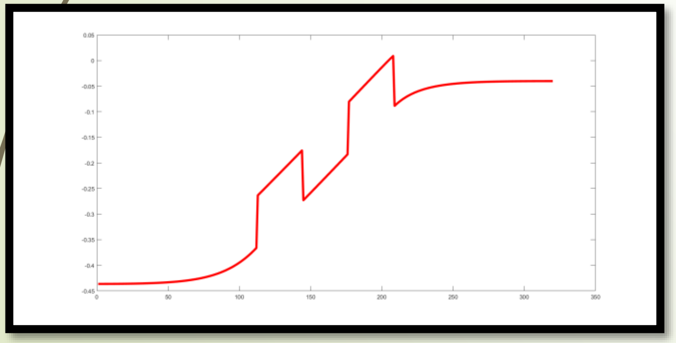
Device 1



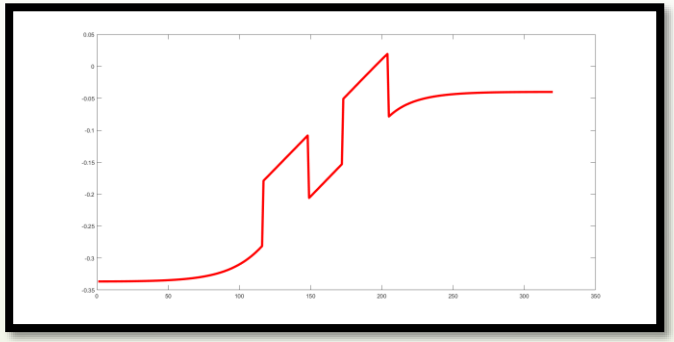
Device 2



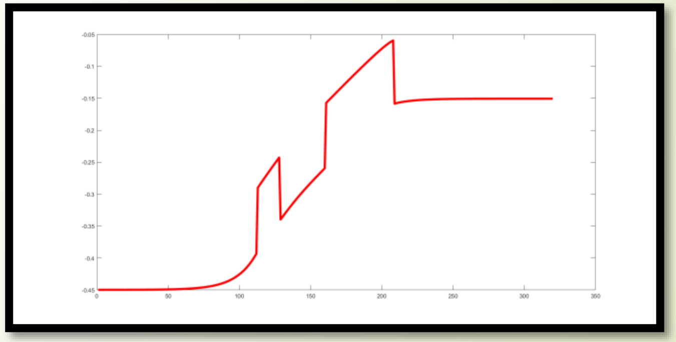
Device 3



Device 4



Device 5



Device 6

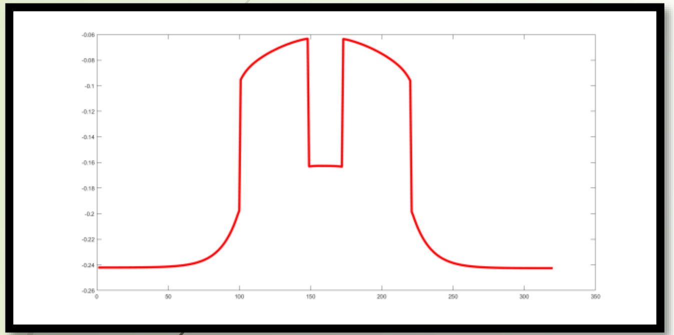
NEURAL NETWORK – SIMULATION TIME

Device characteristics number	Simulation time (seconds)
Device 1	No convergence!
Device 2	250.217
Device 3	281.561
Device 4	321.142
Device 5	150.449
Device 6	168.545

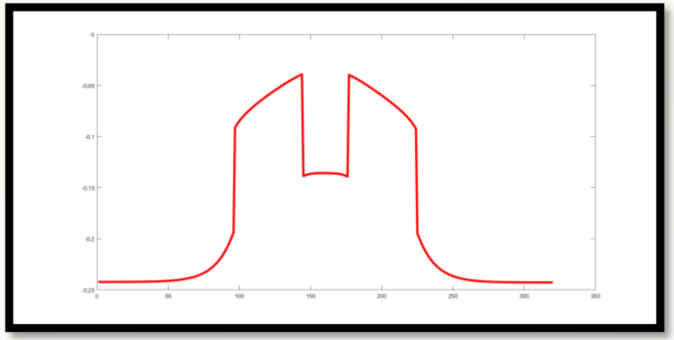
For every device data, we simulated the output by using the predicted [Flat Band – Potential] profile output from our Neural Network and fed it as U_guess to our simulator.

Default simulator MODEL

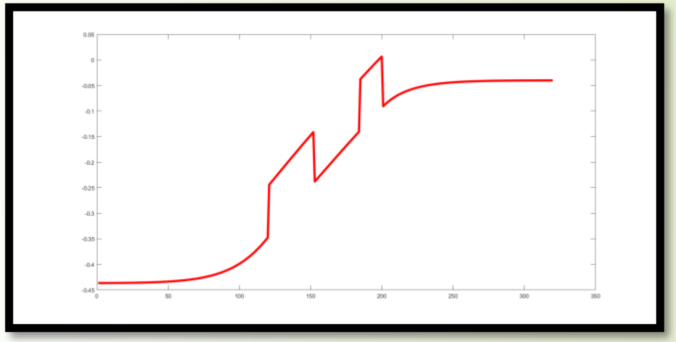
BASE SIMULATION – SIMULATION Results



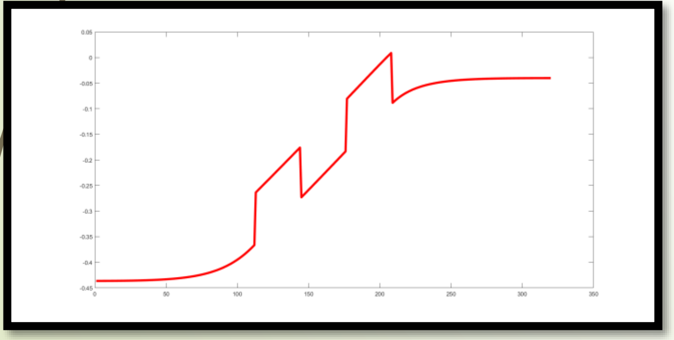
Device 1



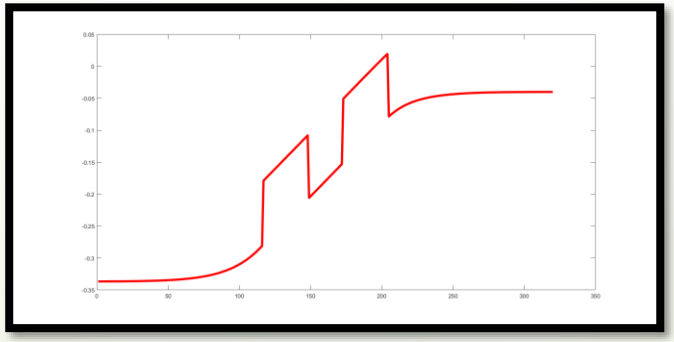
Device 2



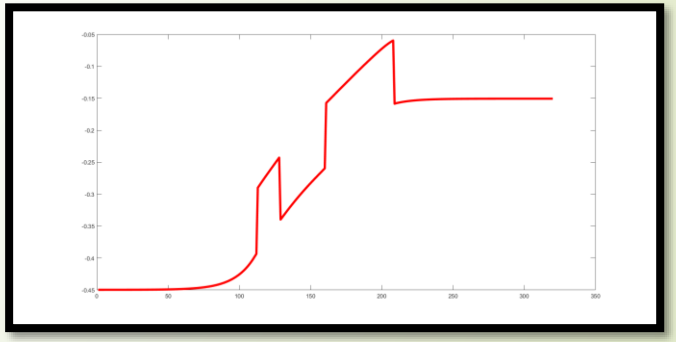
Device 3



Device 4



Device 5



Device 6

BASE SIMULATION – SIMULATION TIME

Device characteristics number	Simulation time (seconds)
Device 1	6763.822
Device 2	452.229
Device 3	307.347
Device 4	390.172
Device 5	193.026
Device 6	275.433

These are the ground truth result when the simulation is done with U_guess as a zero vector and these results are used for comparison of the performance of the prediction models.

Results

Percentage reduction of simulation time wrt. Default simulator when apply predicted initial guess →

Device	Random Forest % reduction	XGBoost % reduction	Neural Network % reduction
Device 1	83.94	82.28	-∞
Device 2	22.23	-120.49	44.67
Device 3	28.97	14.82	8.39
Device 4	37.06	24.99	17.69
Device 5	28.57	16.34	22.06
Device 6	43.18	34.48	38.81

Average Percentage reduction of simulation time →

	Random Forest	XGBoost	Neural Network
Average reduction	40.66	8.74	1.94

For the sake of comparison and penalizing neural network for not-converging we put in place of -∞ the worst % reduction that was found which was -120.49%

CONCLUSION

- Under these circumstances, our **Random Forest regressor** model worked the best with the least amount of training and test MAE, and providing a huge reduction in simulation time while being highly accurate to base simulation results.
- **XGBoost** gives a more generalized model with a lower test accuracy, but is much more computationally expensive and time taking than Random Forest Regressor.
- Since dataset is really small**, deep learning methods** by conventional means won't give us an optimal solution. But looking at simulation outputs, we see these methods sometimes work better than both XGBoost and Random Forests, only with exception of non-convergence in a single case.
- It was necessary to bootstrap data and add by considering subsamples from it as dataset was small and generating more examples for the data was computationally expensive and time taking. This limitation of having less data for training and testing must have been the major reason for failure of our Neural Network predictions as Deep learning networks require a huge base of data for training.

Future Scope

- Next, we should look forward to **tuning the hyper parameters of our Random Forest model** so as to give us better accuracy for the complete range of out dataset.

- With the possibility of **training even better Deep Learning networks**, using Random Forest to train even more data with faster computation time will help us getting our much needed bigger database and then using it to train a Neural Network to check whether it can out-perform the other models and help in further reduction of this generalized simulation.

- This being a **generalized process** which only required lengths of barriers, channels, contacts and wells, we can train even other multi-barrier devices on our network and see if the model lives up to decreasing their computational times.

Bibliography

- Wu, T. and Guo, J. (2020). Speed Up Quantum Transport Device Simulation on Ferroelectric Tunnel Junction With Machine Learning Methods. IEEE Transactions on Electron Devices, 67(11):5229–5235
- Chang, L. L. and Esaki, L. (1977). Tunnel triode—a tunneling base transistor. Applied Physics Letters, 31(10):687–689. eprint: https://doi.org/10.1063/1.89505.
- Gallagher, W. J., Kaufman, J. H., Parkin, S. S. P., and Scheuerlein, R. E. (1997). Magnetic memory array using magnetic tunnel junction devices in the memory cells.
- Gerra, G., Tagantsev, A. K., Setter, N., and Parlinski, K. (2006). Ionic polarizability of conductive metal oxides and critical thickness for ferroelectricity in BaTiO3. Phys Rev Lett, 96(10):107603. Place: United States.
- Ikeda, S., Miura, K., Yamamoto, H., Mizunuma, K., Gan, H. D., Endo, M., Kanai, S., Hayakawa, J., Matsukura, F., and Ohno, H. (2010). A perpendicular-anisotropy CoFeBMgO magnetic tunnel junction. Nat Mater, 9(9):721–724. Place: England
- Lam, K.-T., Cao, X., and Guo, J. (2013). Device Performance of Heterojunction Tunneling Field-Effect Transistors Based on Transition Metal Dichalcogenide Monolayer. Electron Device Letters, IEEE, 34:1331–1333.
- Datta, S. (2002). The non-equilibrium Green's function (NEGF) formalism: An elementary introduction. In Technical Digest - International Electron Devices Meeting, pages 703 – 706
- Venugopal, R., Ren, Z., Datta, S., Lundstrom, M. S., and Jovanovic, D. (2002). Simulating quantum transport in nanoscale transistors: Real versus mode-space approaches. Journal of Applied Physics, 92(7):3730–3739. eprint: https://doi.org/10.1063/1.1503165.

Thank You