HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 1 (2023)

Enhancing Simulations of Superparamagnetic Magnetic Drug Delivery to Predict Efficacy of Treatment

Anas Owais[1], Ashita Biria[2] and Tara Pathak[2]

[1]Middlesex County Academy
[2]Ronald McNair High School

ABSTRACT

Globally, leukemia and skin cancers are the most common types of cancers. However, current measures (radiation, chemotherapy, etc.) cannot exercise high accuracy when targeting such tumors and cancers, causing extensive damage to the surrounding tissue, and leading to the adverse effects commonly associated with treatments such as chemother-apy. Magnetic drug delivery applies anticancer nanoparticles as a component vehicle for charged, targeted treatment. Building upon prior research, a stochastic numerical model was enhanced to simulate the motion of a superparamag-netic cluster suspended in different types of flow while being guided by an external magnet to travel to a target. The model supplanted the assumed Carreau blood flow to a realistic Hagen-Poiseuille flow under the influence of a mag-netic field. The specific application of such clusters magnetic drug targeting, with clusters in the range of 10–200 nm radii. Using a magnetic dipole model for the external magnet close to the surface of the skin, the time of arrival at the target was calculated through a physics-informed neural network by defining the pathway for the clusters. Variations in the release position, background flow, magnetic field strength, number of clusters, and stochastic effects are as-sessed to see how they affect the capture rate. The capture rate is found to depend weakly on variations in the velocity profile, and strongly on the cluster size, the magnetic moment, and the distance between the magnet and the blood vessel wall. The model is validated with existing data and good agreement with experimental results is shown.

Introduction

There are many potential benefits to the application of magnetic drug delivery [1, 2, 3] in the treatment of cancer patients, many of which point to a substantially higher efficacy in the treatment. Some of the potential benefits include improved treatment efficiency and a significant reduction in adverse effects, as opposed to more traditional forms of treatment like chemotherapy. Magnetic drug delivery is an approach in which superparamagnetic iron-oxide nanopar-ticles containing treatment drugs are injected into the bloodstream. Then, using an external magnet, the nanoparticles are guided toward the targeted tumor (see Figure 1). This method of anticancer drug delivery contrasts traditional forms of cancer treatment, in which the drugs are delivered intravenously and then spread throughout the body until they sufficiently accumulate within the tumors. Traditional forms of cancer treatment allow for the drugs to dissemi-nate throughout the body, affecting healthy tissue and causing numerous adverse reactions. This traditional method is not nearly as efficient as it could be: if the drugs were instead localized to the specific tumor, it would allow for the medication to directly target the tumor and concentrate its effects onto its immediate vicinity instead of causing sys-tematic changes within the body. These factors allow for magnetic drug delivery to remain as the more promising and efficient approach.

However, there are certain drawbacks [4] related to magnetic drug delivery which limit its current potential. Since this approach relies heavily on an external magnet, one large concern is the ability for the drugs to remain within the tumor after the external magnetic field is removed. Other factors, such as the velocity of the bloodstream and the distance of the magnet from the tumor may affect the efficacy of magnetic drug delivery.

In order to evaluate the optimal conditions for this type of treatment to be most effective in treating the tumor, machine learning and neural networks can be utilized. Machine learning [5] is a powerful tool that allows for software applications to optimize accuracy when predicting outcomes, given set datasets to use as a reference for the predictions. Neural networks [6], specifically, are a subset of machine learning, in which a computer learns to do a certain task by analyzing examples. Neural networks are composed of node layers, input layers, output layers, and middle layers: each layer can contain a unique number of individual binary units, called neurons, which can process inputs and provide outputs. Additionally, in order to account for the values in the dataset which will help the program make predictions based on the observed patterns, libraries compatible with neural networks were utilized. One such library utilized in the program was Tensorflow. Tensorflow [7] is an open-source software library used for neural networks and is well-suited for dataflow programming involving multiple factors (which is consistent with our use in finding multiple factors such as loss, accuracy, etc.). Tensorflow offers a plethora of resources, tools, and libraries throughout the communities that support it, and therefore provides an extremely useful tool in building and using machine learning apps. Using this library, variables such as layers, epoch, optimizers, etc., can be tested to find the best combination for the lowest lost data value and the greatest accuracy value. Loss and accuracy are inversely proportional, as the loss function is a function that compares the target and predicted output values and measures how well the neural network models the training data. Decreasing loss and increasing accuracy is vital as it will provide a more accurate picture of the viability of magnetic drug delivery and how it can be made more effective for several cancer patients.

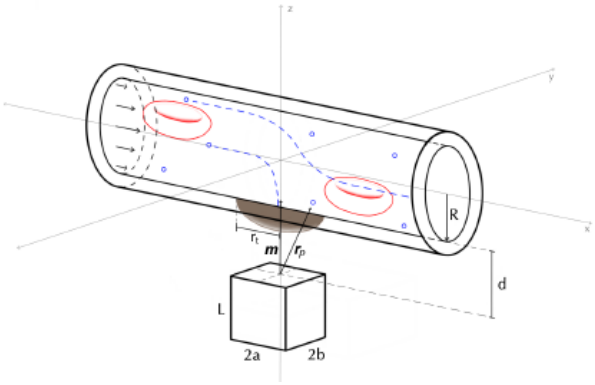

Figure 1: Model for magnetic drug delivery.

Methodology

Research Design

The primary focus of this research is to simulate magnetic drug delivery in order to potentially discover the most optimal conditions that would allow for the highest rate of success. Due to involving multiple, repetitive calculations, machine learning was used to create a neural network for the simulation. All code was written on Visual Studio Code, and incorporated libraries such as Tensorflow, Pandas, and NumPy to assist with coding and calculations. Specifically, a neural network was utilized as it would be able to take previously generated data sets and learn from them, allowing it to make accurate predictions when provided with an unknown set of parameters. The goal was to optimize the neural network such that it would have a minimum overall loss whilst still maintaining its accuracy.

The main variables that would be used as parameters for the research included the position of the particles in a three-dimensional space (in terms of X, Y, and Z), probability of success, the size of the magnet, and average time, with all of the data being in numerical form. It should be noted that the probability is calculated by combining physical methods, Carreu Flow, and numerical methods, and that the numerical dataset used for the code is derived from the mathematical models provided by [2]. Over 10,000 data sets were derived from the established equations, of which around 5,000 were randomly selected to be used as training data for the algorithm. In addition, all the data that was generated was rescaled within the code to call between (0, 1). This was done to allow for the reduction of loss, as the original data consisted of multiple scales and thus prevented the lowering of data loss.

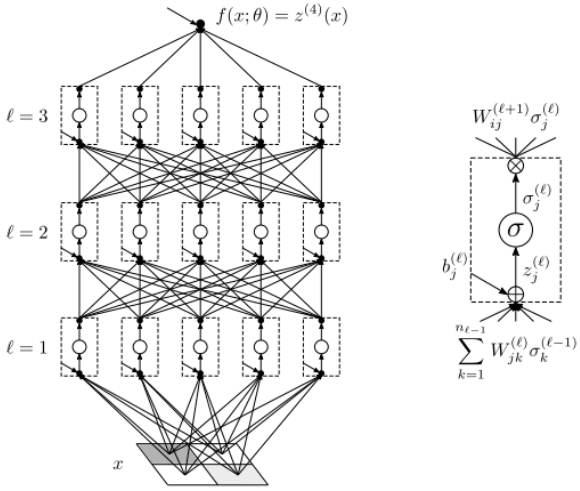

Figure 2: Depiction of a neural network; includes the input layer, middle layer (which is singular in this specific case), and output layer. The figure on the left showcases the different layers and highlights their interconnected nature. Each node is interconnected allowing for data transmission, and the figure on the right represents the connection of one node to several other nodes.

Hypothesis

After finding certain values for the layers, learning rate, and epoch that return a low loss value, it is hypothesized that the loss can be minimized even more by altering the value of the epoch. If a combination of an epoch of 3000 with constant values for the other two variables returns a loss of 0.0005, doubling the epoch to 6000 will ultimately cut down the loss by half, hence making it 0.00025. In addition, when the loss is showing a constant downward trend with a specific combination of variables, increasing the epoch will allow for the minimization of loss mathematically proportional to the ratio by which the original epoch was increased.

The most efficient optimizer for reducing loss is hypothesized to be the Adagrad optimizer. Unlike the other options, Adagrad is unique in the sense that it abolishes the need to manually modify the learning rate [8]. Unlike other optimizers, it does not use a single learning rate value for each iteration. The value of this is that it accounts for sparseness (primarily zero values) and denseness (primarily non-zero values) of data by modifying the learning rate to fit with the condition of the current parameter. Therefore, using the Adagrad optimizer with this specific dataset would return the lowest loss value in comparison to the other optimizers.

The most effective activation function to utilize in the neural network for reducing loss would be the sigmoid function. Since the neural network is meant to enhance the previous stochastic model, the sigmoid function would provide a reasonable prediction within bounds of 0 and l to produce probability as an output. It would also be more efficient than other activation functions, such as ReLu, which would require more computational resources being

dedicated towards computations and still incur a higher net loss [9]. Therefore, using the sigmoid activation function with this specific dataset would return the lowest loss value in comparison to the others.

Data Analysis Methods

There were two main aspects of the data which needed to be analyzed in context of the problem we were trying to solve: the performance of the neural network itself and the capture rate of the magnetically enhanced drugs coming into contact with the tumor site. The performance of the neural network directly affected the capture rate.

Results and Findings

Hyperparameters

The first step of the process was to create a neural network that could be used for simulation. This initial neural network used two to five layers, each consisting of 50 neurons, and the ability to stop if no significant improvement occurred. It used 3000 epochs (one epoch is when an entire dataset is passed forward and backward through the neural network once) and a learning rate of 0.45. It used these parameters to generate and train a neural network. Once this training process was completed, the model was evaluated using 10 sets of data randomly selected from the 5000 samples. The value loss and accuracy were calculated by comparing the results that the neural network output. Once the neural network was created: the layers, epochs, and learning rate were all treated as variables and changed in order to find the combination which provided the lowest loss.

Initially, the testing was done manually by changing one parameter at a time and then running the program to see if there was any improvement. However, with the vast amount of potential combinations, testing the parameters by hand was inefficient and time-consuming. Instead, the code was adapted into a hypertuner model. A hypertuner neural network generates hyperparameters, parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning, for a neural network. It is able to tune parameters such as the number of neurons, activation function, optimizer, learning rate, batch size, and epochs in order to decrease the loss of a neural network. These hyperparameters control the model parameters that a learning algorithm eventually ends up learning, and thus, are vital to the efficiency of a neural network. The hypertuner model found that the optimal number of neurons per layer was 64, with the best learning rate being between 0.01 and 0.001. Regarding epochs, the hypertuner would run a set amount of epochs (input by the user) and then select the most efficient one and retest up to that number. Thus, the best epoch value fluctuated between iterations of the code. Even with the use of the hyperparameters, the loss was not lowered substantially enough for the accuracy to improve. This outlined the flaw with the neural network; while it was the only way to create a model that could predict the outcome of new data, its accuracy relied on the way it was created. Thus, in order to decrease the loss further and create an effective model, other aspects of the neural network must be optimized.

Optimizer

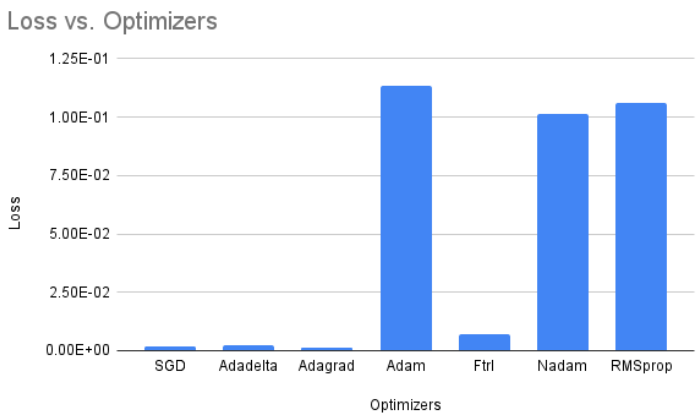

Figure 3: The various optimizers available in the Tensorflow library and the loss values they produced.

Table 1: Various optimizers and the loss values produced, alongside observations.

Optimizers	SGD	Adadelta	Adagrad	Adam	Ftrl	Nadam	RMSprop
Loss	1.71E-03	2.07E-03	1.31E-03	1.14E-01	7.25E-03	1.02E-01	1.06E-01
Observations	No notable observations.	Loss constantly decreased as epoch increased.	Loss almost decreased constantly as epoch increased, but there were some exceptions.	Loss was the same for every epoch.	Decreasing loss but slower than other optimizers.	Loss was the same for every epoch.	Similar loss value for each epoch

After testing out various optimizers, the one which gave the best results was the Adagrad optimizer (see Figure 3 and Table 1). Unlike other optimizers, Adagrad uses a unique learning rate with each iteration, in order to account for sparseness and denseness of data. The only optimizer that came close to the same loss value that Adagrad returned was SGD (the default optimizer used with the data). However, after consideration of the benefits of Adagrad as well as its successful loss value, it is concluded that the latter optimizer is the most efficient one and should be utilized within the code to find the lowest loss value.

Activation Function

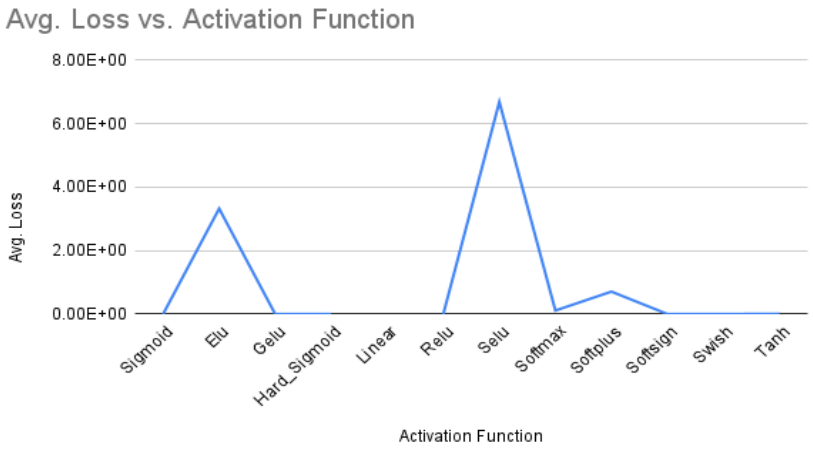

Figure 4: The various activation functions available in Tensorflow and the loss values they output.

Table 2: The best numerical loss values of the activation functions and notable observations.

Activation Function	Avg. Loss	Observations
Sigmoid	1.71E-03	Heavily compresses input data and approaches 0 as more is processed.
Elu	3.32E+00	Very similar for all epochs.
Gelu	1.75E-03	
Hard_Sigmoid	1.73E-03	
Linear	n/a	Approach or linearity + origin output an undefined loss.
Relu	2.06E-03	Doesn't activate all neurons at the same time; faster run-time but higher net loss.
Selu	6.69E+00	Smooter run; lower run time, however extremely high loss; very similar/same for all epochs.
Softmax	1.12E-01	Same for all epochs.
Softplus	7.08E-01	Same for all epochs; Smooter run; lower run time; however, extremely high loss.
Softsign	1.61E-03	Between 2000-3000 epochs, loss went down and then backed up minimally, and then reduced.
Swish	1.83E-03	Quickly drops after 200 epochs.
Tanh	1.04E-02	Scaled sigmoid; creates deeper difference as in graph.

The data expressed by Figure 4 and Table 2 show a great variation in the loss values produced by different activation functions. Due to the high variation and large amount of activation functions, the three activation functions with the lowest average loss from the twelve displayed above were tested further with the hyperparameters produced by the hypertuner model. This was done to finally isolate one function as superior over the rest, and to clearly display which was the most efficient function.

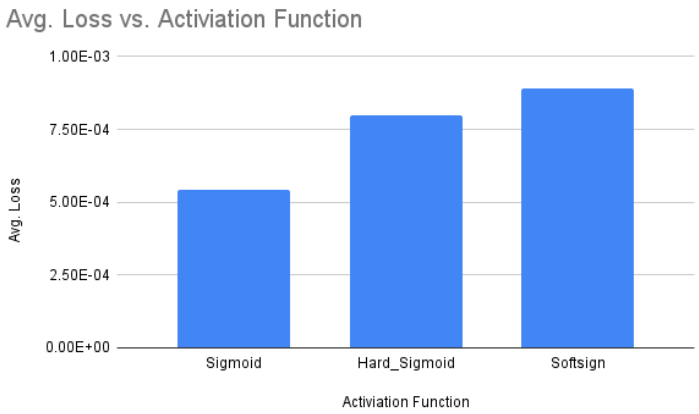

Figure 5: Average loss of the three most efficient activation functions from Table 2. All three activation functions underwent extensive additional testing to produce the above averages.

As Figure 5 clearly shows, testing the most efficient functions with the hyperparameters indicates that the Sigmoid activation function is the best.

Efficiency of the Neural Network

After testing multiple combinations of variables using the optimal optimizer and activation function, the range of the functional variables was isolated and identified. This gave us further insight on evaluating how accurate our initial hypotheses were. For example, while using a specific combination of variables, an almost constant decrease in loss for each iteration and epoch level was noted. Using this information, it was theorized that by doubling the epoch, the loss would be cut down by half, returning the lowest value. However, after inputting a doubled epoch, we noticed that loss cut down slower and the final loss value was much larger than the initial. Hence, the first hypothesis was disproved.

Additionally, while using the optimizer Adagrad, it was observed that the run time of the program would be affected, providing the loss at an inefficient rate. Especially with a high epoch, the program would run especially slow and modifying it to fit certain values was extremely time-consuming. Therefore, the best method of modifying the code would be to add a smaller amount of neural network layers, as the program would take a shorter time to compile. This is because each node in a neural network program is interconnected – thus, a larger number of layers would mean that the program would take longer to compile.

The second and third hypotheses proved accurate as the optimal optimizer proved to be Adagrad and the optimal activation function proved to be sigmoid [8, 9]. The descriptions of the benefits of using these specific functions proved to be correct. In each test case that tested loss value using these specific conditions, the program returned the lowest value of loss, proving that Adagrad and Sigmoid would enable for the production of the most accurate results.

Ultimately, the best loss value was found using all of these factors. By lowering the amount of layers being utilized from five to three and using the optimizer Adagrad and the activation function Sigmoid, a loss value of 0.00017 (simplified) with an accuracy of 50.9% was produced.

Table 3. The parameters used to achieve the lowest loss value.

Hyperparameters

Parameter	Value/Setting
Layers	50 50 1
Loss	1.70E-04
Accuracy	50.9%
Learning Rate	0.01
Epoch	8000
Activation Function	Sigmoid
Optimizer	Adagrad
Loss Function	MSE

The Physics-Informed Neural Network

By incorporating the results of the three separate tests, the most optimized version of the neural network was created. However, even though all aspects of it were enhanced, the best result produced was a loss of 1.70131×10-4 with an accuracy of around 50%. While the accuracy does fully reflect the full effectiveness of the neural network (i.e. the predictions were usually within 0.1 - 0.01 of the actual values), this small margin was still substantial enough for the neural network to be considered ineffective.

The problem with the neural network now lay in the context of the problem that was being solved. The program was attempting to simulate magnetic particles in the vasculature of the human body as if the blood had Newtonian flow and did not account for how the magnet itself might influence the success rate. However, for the loss to be reduced further and the program's accuracy to be increased, the code would have to account for the flow of blood as well as the magnet in relation to the particles within the bloodstream. Thus, the neural network was discarded in favor of a physics-informed neural network, or PINN. PINNs are specialized forms of neural networks that apply physics to their learning cycles in order to create more accurate results for physics-based problems. These specialized neural networks are able to solve for partial differential equations, fractional equations, and integral-differential equations, and can complete various other high-level mathematical computations.

Equation 1: The equation by which blood flow is emulated in the neural network.

$$\gamma = [\frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})]$$

Equation 2: The mathematical representation of the magnetic field generated by the external magnet.

$$\frac{dx}{dt} = u(x) + \frac{2a_p^2\mu_0\chi}{3(3+\chi)\eta(x)}(H_e \cdot \nabla)H_e$$

To develop the PINN, two main physics equations were incorporated. Equation 1 deals with Carreau Flow to stimulate blood flow. The equation accounts for factors such as viscosity of the blood and the presence of other blood cells. Equation 2 relates to the magnetic field generated by the presence of an external magnet, and accounts for the magnets size, strength, and distance from the body. A standardized magnetic coefficient as well as constant size was used for the paramagnetic particles within the blood itself to minimize the impact changing these values would have on the capture rate. For more information on the equations utilized, reference [2].

By accounting for both of these factors, the learning process of the neural network and the subsequent results it generated would be much more efficient and accurate. This led to a substantially lower loss value as well as improved the accuracy of the model. The PINN was also coded to produce a graph (shown below) to display the numerical solution of the various partial differential equations incorporated into the code. Through the use of the PINN, the best results thus far were obtained.

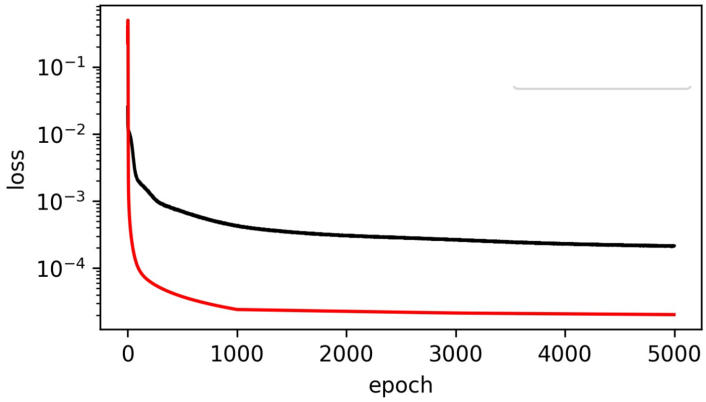

Figure 6: The PINN versus the results produced by the neural network without incorporating physics. The black line represents the neural network's loss, and the red line represents the loss of the physics-informed neural network. The graph demonstrates the significant difference between the loss values achieved using a regular neural network versus using a physics informed neural network. At every observable epoch, the PINN always outputs results with a lower value-loss than the NN.

With the loss at its lowest possible point, the capture rate for the medication can finally be determined.

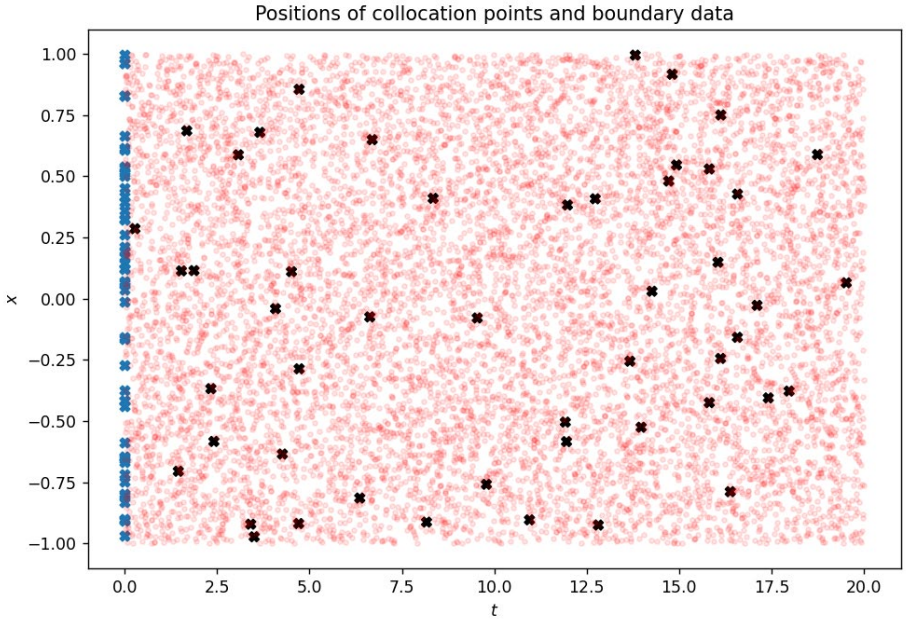

Figure 7: A graph output by the PINN that displays the numerical solutions to the differential equations that simulate the physics of this model.

Figure 7 is generated every time the PINN is run, and it represents the capture rate in a mathematical context. Boundary data is a pair of test data values at each end of a range: The data at the upper or lower limits of expectations that should be accepted. Immediate values before or beyond the limits of expectations that should be rejected. The code implements the collocation method, which enables it to choose a finite-dimensional space of candidate solutions (usually polynomials up to a certain degree) and a number of points in the domain (called collocation points), and to select that solution which satisfies the given equation at the collocation points. The values of "x" and "t" are then used by the program for computation (i.e., finding derivatives).

As for the capture rate, the black X symbols represent points in time when the magnet is able to effectively direct the medication to the tumor. The red dots all represent potential points of contact between the medication and the tumor. While it may appear to be ineffective at a glance, the fact that at least some of the medication is able to be redirected is promising of the potential viability of this drug delivery method. The medication that is not redirected will still operate in the same manner that current medications do, and thus will still benefit the patient. Over a longer interval of time, as the medication re-circulates within the body, the possibility also exists that it will be pulled towards the tumor during its second circulation. Overall, this output should be viewed as a great success.

Conclusion

Finding the most efficient combination of functions, optimizers, and parameters as well as the incorporation of differential equations to effectively model the physics involved with this treatment method has allowed for an accurate representation of the viability of magnetic drug delivery to be simulated. By lowering the loss value and increasing the accuracy of the results generated by the neural network as well as optimizing the learning process, the results output by the program is an accurate representation of what may occur when an external magnet is used for targeted drug delivery. Through the use of the neural network and hyperparameters, the program and its generated data have allowed for further insight into the ideal conditions for magnetic drug delivery. More important than that, however, is the fact that the neural network has shown that magnetic drug delivery, under the right conditions, is a viable method

to target tumors at the skin level. The true significance of this research lies in the evidence it provides regarding the efficacy of this approach for cancer treatment.

The results of this study indicate that there is great potential for the implementation of this treatment method. As such, future research can and should be conducted in order to further test the viability of this treatment method. The next steps would include creating an even more sophisticated version of the PINN. Through further analysis of the mathematical workings of the PINN as well as improved incorporation of fluid mechanics into the code, the accuracy and thus validity of the model could be significantly improved. Optimizing the model to be representative of a clinical scenario would be the end goal of this step in the research process. If an even more optimized model also points towards the viability of magnetic drug delivery, research should then be conducted on other aspects of this treatment method, including the types of medicines that can utilize magnetic targeting as well as how extended use of magnets could impact the human body. Future research would include finding a safe way to attach magnetic particles to the specific medications that this targeted delivery method would use and then inject them into a human body. Non-human test subjects would be the first in clinical trials, and success with them would allow for human testing to one day become a possibility. Potential side effects that may arise from this treatment method should also be investigated. In addition, an apparatus would need to be designed and built that would be able to direct a magnet at any part of the human body which may have a surface-level tumor. If all of these prerequisites are achieved, human trials may begin to examine the effectiveness of this treatment, but as the neural networks have all indicated, the chances of success are considered high.

Acknowledgements

All authors of this paper would like to express our gratitude towards those who played an integral role in making the research behind this paper a reality. First, we would like to thank Dr. Afkhami for providing us with his mentorship. Under his guidance, we were able to explore all the nuances of this topic to their full capacities and produce a thorough and complete paper. We would also like to thank Zhaoshu Cao for her assistance and guidance throughout our research. We extend our appreciation to NJIT and the Liberty Science Center Partners in Science program for providing us with the resources, connections, and opportunities needed to conduct this research. Special thanks to Dr. James Coilaizzi for his invaluable assistance in editing the paper itself to meet industry standards and present our research in the best manner possible.

References

1. Yue, P., Lee, S., Afkhami, S., & Renardy, Y. (2011). On the motion of superparamagnetic particles in magnetic drug targeting. Acta Mechanica, 223(3), 505–527. https://doi.org/10.1007/s00707-011-0577-9

2. Lee, M., Shelke, A., Singh, S., Fan, J., Zaleski, P., & Afkhami, S. (2022). Numerical simulation of superparamagnetic nanoparticle motion in blood vessels for magnetic drug delivery. Physical Review E, 106(1). https://doi.org/10.1103/physreve.106.015104

3. Price, P. M., Mahmoud, W. E., Al-Ghamdi, A. A., & Bronstein, L. M. (1AD, January 1). Magnetic Drug Delivery: Where the field is going. Frontiers. Retrieved August 19, 2022, from https://www.frontiersin.org/articles/10.3389/fchem.2018.00619/full of superparamagnetic nanoparticle motion in blood vessels for magnetic drug delivery. Physical Review E, 106(1). https://doi.org/10.1103/physreve.106.015104

4. F;, S.-F. E. K. R.-O. (n.d.). Advantages and disadvantages of using magnetic nanoparticles for the treatment of complicated ocular disorders. Pharmaceutics. Retrieved August 19, 2022, from https://pubmed.ncbi.nlm.nih.gov/34452117/#:~:text=However%2C%20there%20are%20some%20drawbacks,dimensions%20inside%20the%20human%20body.

5. Burns, E. (2021, March 30). What is machine learning and why is it important? SearchEnterpriseAI. Retrieved August 19, 2022, from https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values.

6. The University. (1978). What is ... Amazon. Retrieved August 19, 2022, from https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,that%20resembles%20the%20human%20brain.

7. Tensorflow. TensorFlow. (n.d.). Retrieved August 19, 2022, from https://www.tensorflow.org/

8. Doshi, S. (2020, August 3). *Various optimization algorithms for training neural network*. Medium. Retrieved August 19, 2022, from https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

9. SHARMA, S. (2017, September 6). Activation Functions in Neural Networks. Medium; Towards Data Science. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

10. Team, K. (n.d.). Simple. flexible. powerful. Keras. Retrieved August 19, 2022, from https://keras.io/

11. IBM. What are Neural Networks? (n.d.). Www.ibm.com. https://www.ibm.com/cloud/learn/neural-networks#:~:text=Neural%20networks%2C%20also%20known%20as

12. Accuracy and Loss - AI Wiki. (2022). Paperspace.com. https://machine-learning.paperspace.com/wiki/accuracy-and-loss#:~:text=Unlike%20accuracy%2C%20loss%20is%20not

13. Magnetic Nanoparticles for Drug Delivery. (2018, October 22). News-Medical.net. https://www.news-medical.net/health/Magnetic-Nanoparticles-for-Drug-Delivery.aspx#:~:text=Magnetic%20nanoparticles%20have%20been%20developed