Physics-Informed Neural Networks Modeling for Tumor Proliferation.

Mohammed Yusuf Mujawar

Deep Generative Modeling for Physical Processes

Masters in Computer Science

The University of Alabama

mmujawar@crimson.ua.edu

***Abstract*—This paper examines the use of Physics-Informed Neural Networks (PINNs) to model the growth of glioblastoma, a type of brain tumor. By applying the Diffusion-Reaction Equation, we blend traditional machine learning methods with the rules of physics to better predict how these tumors grow. This approach helps us understand tumor growth under different conditions and offers insights that might lead to better treatments.**

A significant part of this study was to use PINNs to tackle the issue of limited data availability, which is common in medical research, especially when studying aggressive tumors like glioblastoma. Our model accurately simulated tumor growth by closely following the known behaviors of such tumors, ensuring that our predictions were realistic.

The findings highlight the potential of PINNs not only in medical imaging but also in creating personalized treatment plans. The advantage of PINNs in needing less data while still providing accurate simulations could be particularly useful for creating detailed models tailored to individual patients.

Keywords—PINN, Glioblastoma, Tumor, Brain, Diffusion-Reaction Equation, PDE, Loss Function, and Boundary Conditions.

I. INTRODUCTION

With the increase in data and high performance computing resources the increase in the usage of Machine Learning and Deep Learning has also seen substantial increase. But not every field in which we are implementing these technologies has enough data to train the models with high accuracy on which we can depend and many neural network have difficulty to converge.[1]. PINN was developed by Raissi et al., so that it can follow the rule of physics and work with equations in solving problems which are more difficult for traditional neural networks [1]. Since it uses equations it doesn't require much data as that of traditional neural networks. PINNs use loss functions to adhere to the physical laws and equations. I have selected Diffusion-Reaction Equation to model the growth of brain tumor specifically Glioblastoma which a cancer that effects the brain cells and spinal cord. Glioblastoma affects brain cells and destroys healthy tissues causing Memory loss, Vision difficulties, Speech impediment among other symptoms.[3]. Getting enough data from patients with Glioblastoma tumor to train traditional neural networks with high-efficiency is very challenging and leads to models that perform mediocre. Partial Differential Equations such as Diffusion-Reaction can used to reproduce the growth of tumor cells in humans in PINN. Diffusion-Reaction Equation is used to model the reaction of various components and the rate on how it propagates. It can model the rate at which the tumor will spread across brain and how it effects different parts of the brain such as white matter, grey matter, and cerebrospinal fluid.[4].

The aim of this paper is to demonstrate a PINN model which can be used to model the growth of glioblastoma tumor in brain with higher accuracy and efficiency than traditional method. By applying the Diffusion-Reaction Equation, this project aims to map the progression and effects of glioblastoma more precisely, offering potential insights that could lead to more effective treatment strategies. The significance of this study lies in its potential to provide a robust modeling tool that requires fewer data inputs and offers reliable simulations of tumor dynamics, which are critical for developing personalized medicine approaches and enhancing patient care in oncology.

II. DIFFERENTIAL EQUATIONS OVERVIEW

In this project, I utilized the Diffusion-Reaction Equation to model the complex behavior of glioblastoma tumor growth within the brain. The Diffusion-Reaction Equation is a fundamental type of partial differential equation (PDE) that describes the dynamics of concentration profiles in a physical system where chemicals react and diffuse. For our paper, these equations models how tumor cells proliferate and invades the healthy brain tissue [3]-[5].

The specific form of the Diffusion-Reaction Equation used is given as follows [4]:

$$\partial u(x,y,t)/\partial t = \nabla\cdot(D(x,y)\nabla u(x,y,t)) + \rho u(x,y,t)\,(1-u(x,y,t))\,(x,y,t)$$

Here, u represents the tumor cell density at a given point in space and time, D is the coefficient for tissue density which reflects the rate at which tumor cells spread through the brain tissue ρ is the proliferation rate which describes how quickly the tumor cells reproduce through natural processes. For every patient suffering from Glioblastoma the coefficient for diffusion. The term $\nabla\cdot(D\nabla u)$ captures the diffusion of tumor cells, while $\rho u(1-u)$ models logistic growth, capturing the natural limits on growth as the tumor reaches the carrying capacity of its environment.

This equation is particularly suited to model glioblastoma due to the aggressive nature of the tumor, characterized by rapid growth and significant infiltration into surrounding neural tissue. The ability to accurately simulate both the diffusion and proliferation of tumor cells is important in predicting tumor growth patterns and potential responses to treatment. By incorporating the Diffusion-Reaction Equation within a PINN, we leverage the strengths of machine learning to solve this complex PDE across irregular and complex geometries of the brain, informed directly by patient-specific data, which is very difficult to achieve in traditional methods.

The application of this equation through the PINN framework allows for enhanced computational efficiency and robustness in handling boundary conditions and variations in tissue properties, which are often challenging to manage with traditional numerical simulation techniques. The model's flexibility and adaptability make it a powerful tool for exploring new therapies and surgical interventions designed for every individual patient's anatomy and tumor characteristics.

Diffusion-Reaction Equation is used in a multitude of fields beyond its application to tumor growth modeling. In ecology, it's used to understand how species populations change over time and space, which helps in studying the spread of invasive species and the effects of habitat fragmentation. In the field of chemistry, this equation helps model how the concentration of substances changes through reactions and diffusion, important for situations like pollution spread or optimizing chemical reactors. Additionally, in developmental biology, Diffusion-Reaction models are crucial for explaining natural patterns like the stripes on animals or the structures of shells, showing how complex patterns can emerge from simple biochemical interactions.

III. Background or Related Work

The integration of neural networks with differential equations, particularly through the development of PINNs, marks a huge achievement in computational science. This approach allows us in embedding of governing physical laws directly into the architecture of neural networks, enhancing their ability to solve complex differential equations across various scientific fields. Here we go through some of the key research papers that emphasize the importance and areas of application, especially in the modeling of intricate systems like tumor growth.

Introduced by Raissi et al., PINNs have set a new standard in the convergence of machine learning and physical sciences. These neural networks integrate differential equation solvers into the training process, which constrains the solution to satisfy both the data and the physics of the problem. This dual compliance is critical for problems where experimental data might be incomplete or noisy, ensuring that the neural network predictions remain physically plausible [1]. The use of loss function and custom activation in PINN allows us to make models that are specific to our requirements and can solve problem from different fields, and are not just black box models such as support vector machine, random forests, etc. According to [1] PINN use automatic differentiation to train neural networks by using their derivatives with respect to space and time .

Focusing on glioblastoma, an aggressive type of brain tumor, Andy Zhu in [5], used PINN to model glioblastoma growth in brain using Diffusion-Reaction equation. With the implementation of PINN instead of traditional neural network Andy Zhu saw a huge improvement in the performance in getting the desired results if implemented correctly and efficiently. From this paper it is was evident that in implementation of PINN it took very less computational power and showed high accuracy. These neural networks facilitate rapid parameter inference, which is essential for adapting models in real-time to reflect patient-specific tumor dynamics. PINN provided the solution for the inference inverse problems in one hour which is 20-40 times faster than the methods used earlier. This capability is critical for clinical settings where timely and precise modeling can inform and potentially improve treatment outcomes [5].

Corentin Martens et al., in [2] talk about how the use of mathematics in medical use specifically in tumor growth has always been put forward for many decades but with the advancement in the computational power and the development of different neural networks has made this a practically possible. In [2], Martens et al, state how use of MRI is not that practical when we want to predict or estimate the growth of tumors in patients and hence they decided to use mathematical models in Convolutional Neural Network to generate 1200 synthetic tumors using 6 healthy people's brain profile. They developed two CNN's to construct complete tumor with cell-density distribution and to assess the model parameters using limited data available as imaging contours. The mathematical model can ultimately provide an accurate depiction of the tumor's spatial and temporal development, based on the estimated distribution of cell density and the values of the parameters [2].

Beyond medical applications, PINNs are being explored in fields like ecology for modeling population dynamics, in engineering for material fatigue analysis, and in meteorology for weather prediction. The flexibility and robustness of PINNs in handling nonlinearities and multi-physics problems make them a promising tool for these complex applications.

Despite their successes, the deployment of PINNs comes with its own challenges. The accuracy of a PINN heavily relies on the correct formulation of the physics-based component of the loss function and the quality of the available data. Ongoing research is directed towards improving the efficiency of these networks in learning from sparse data and extending their application to even more complex multi-scale phenomena.

From reviewing all these papers we saw how integrating neural networks with differential equations through PINNs is rapidly evolving, pushing the boundaries of what can be achieved in scientific computing with low computational power. As these methods continue to mature, they promise to unlock new possibilities in not only medical sciences but also in any field where modeling complex systems is very important.

IV. Methods

A. Description of the Differential Equations

In this section we will see the differential equation that we selected and how it is transformed to solve our problem and the parameters set up in the neural network used. The Diffusion-Reaction Equation which we discussed earlier in this paper is in a general form and from [4] we saw how it is modified to adapt to simulate the glioblastoma growth in brain and can be seen as follows:

$$\partial u(x, y,t)/\partial t - D(x, y)\partial^2 u(x, y,t) /\partial x^2 - D(x, y)\partial^2 u(x, y,t)/\partial x^y - \rho u(x, y,t)\,(1-u(x, y,t)) = 0.$$

For the sake of implementation [4], and [5] have set up the values for the parameters in the above equation with the known rate of glioblastoma tumor and how it effects different parts of the brain. Hence, [5] considered the parameters of high grade tumor which has its white matter coefficient of the tissue density *D(x, y)* from the random tumor dispersal given as 0.13 mm^2 $days^{-1}$, for grey matter it is given as 0.013 mm^2 $days^{-1}$, and finally the *D* for cerebrospinal fluid is set as 0. These considerations are also made for the rate at which this tumor will affect parts of the brain and is given as proliferation rate ρ and is set to 0.025. These values were observed in patients MRI who are suffering from high grade glioblastoma tumor. The use *D(x, y)* in the equation is very useful as it allows us to set the parameters of individual patients giving us more detailed estimation on how the tumor is going to effect the patient which will indeed result in more efficient and effective treatment.

Now, we know that PINN is used to solve the above equation and the solution for that equation can be represented as follows:

$$u(x, y,t) = PINN(x, y, t) = A_n\sigma\,(A_{n-1}\sigma(\ldots\sigma(A_1[x, y,t] + B_1)\ldots + B_{n-1}) + B_n$$

where A_i and B_i denote matrices and bias vectors, respectively, and σ represents the sigmoid activation function.

B. Boundary conditions

Boundary conditions are essential for ensuring that a model's predictions conform to physical constraints at the spatial domain's edges throughout the simulation. The function get_boundary_points is designed to sample these critical areas by creating linearly spaced values across each dimension, forming grids that combine these values. This method effectively maps out the spatial boundaries at various times, converting these grids into column vectors and assigning them to a computational device to enhance performance. Setting requires_grad to True is crucial here, as it enables the network to learn from these points by allowing for the backpropagation of errors during training.

The function get_initial_points focuses on the system's starting state at t=0. By generating evenly spaced points across the spatial domain and associating these points with the initial time, the function ensures that each spatial point is paired with t=0. This setup is vital for the network to match the known conditions at the start of the simulation accurately, providing a solid foundation for the model to build upon as it learns the dynamics of the system.

Beyond the boundaries and initial conditions, a model needs to understand and learn from the dynamics governed by differential equations within the domain's interior. The function get_interior_points addresses this by sampling a comprehensive set of points across all dimensions, creating a three-dimensional meshgrid that spans the entire volume of the domain over time. This extensive interior sampling is essential for training the network on a representative set of data points, ensuring that the model's predictions remain consistent with the underlying physics. As with the other functions, enabling gradient computation through requires_grad is critical for effective learning.

By integrating these constraints directly into the training process through these specialized functions, PINNs effectively leverage both the known physical laws and any available observational data. This method allows the network to predict behavior accurately, even in complex scenarios where direct measurements might be sparse. This structured approach not only enhances the reliability of the model but also broadens its applicability, making it a powerful tool in solving real-world problems across various scientific and engineering domains. Through the meticulous setup of initial, boundary, and interior conditions, PINNs ensure that every aspect of the physical reality is considered, leading to robust and reliable model predictions.

C. Physical Significance

Integrating this modified Diffusion-Reaction equation which is tailored for estimating the growth of glioblastoma tumor within a PINN framework enhances the model's ability to make predictions that are not only data-driven but also deeply rooted in physical laws. This dual approach ensures that the simulations are realistic and grounded in the actual physical processes of tumor growth and diffusion, providing a more reliable tool for medical researchers and clinicians. By solving this complex PDE across the irregular and complex geometries of the brain, informed directly by patient-specific data, the PINN framework offers a powerful method for exploring new therapies and surgical interventions that are tailored to the unique anatomical and pathological characteristics of each patient's tumor.

V. Implementation and Experimentation

A. Neural Network Architecture

Input Layers: The input layer of the network is composed of three neurons, directly mapping to the spatial and temporal variables: x, y, and t. This configuration is critical for the network's capacity to process multidimensional data, as it establishes the basis for the network to interpret both spatial coordinates and temporal evolution. The inclusion of these three variables at the input stage is what enables the model to handle the spatiotemporal aspects of tumor growth, ensuring that the

network can accommodate the necessary dimensions for accurate modeling of the physical phenomena under study.

Hidden Layers: In constructing the neural network for this project, The implemented structure is capable of work with the complexity of tumor growth data. This network is built with four hidden layers, and each of these layers has 160 neurons through which the data is passed. It's a robust setup, designed to tackle the complex task at hand. For the activation function, the hyperbolic tangent, or Tanh for short is utilized. This function is key to introducing non-linearity into the processing stages. Non-linearity empowers the network to understand more than just simple, straight-line relationships—it allows the network to recognize and learn from the complex, multi-dimensional patterns hidden in the data which is useful in recognizing complex data available in brain [4], [5].

The decision to employ Tanh comes down to the need for a model that mirrors the unpredictable nature of glioblastoma growth through the human brain. Tumor growth is far from linear; it's simply erratic, influenced by a multitude of biological variables, and it requires a model that can recognize such complexity with high accuracy. By infusing the network with the capacity to handle non-linear phenomena, the model is equipped to learn the irregularities of tumor cells as they grow and invade healthy tissue. This is crucial because capturing these detailed patterns can significantly improve the predictions the model makes, bringing us closer to understanding the real-world behavior of aggressive tumors such as glioblastoma tumor [4]-[6].

Output Layer: The output layer of the neural network consists of just one neuron. Its function is to output the predicted tumor cell density at specific spatial coordinates *x*, and *y* and a given time t. This single-neuron design is not a simplification but rather a precise alignment with the model's objective, which is to estimate a singular, continuous value—the density of tumor cells. This output is essentially the result of the network's complex internal processes, in a single metric [4], [5].

B. Libraries

1) PyTorch

PyTorch is a comprehensive library for building deep learning models that provides tools for everything from basic tensor operations to training complex neural networks [8]. It is particularly valued for its dynamic computation graph and robust automatic differentiation capabilities. In this project, PyTorch is used to define and train the PINN. It handles the creation of neural network layers, performs forward and backward passes, and manages gradient calculations essential for updating model weights during training. It facilitates the integration of differential equations into the neural network by enabling custom gradient computations, which are crucial for ensuring that the network adheres to the physical laws described by the Diffusion-Reaction Equation [8], [4].

2) NumPy

NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is primarily used for handling numerical operations that are not directly related to the neural network training but are essential for data preprocessing, post-processing, and analytical tasks. While its direct role in neural network training might be limited, NumPy underpins data manipulation and transformation tasks, ensuring efficient handling of numerical data [9].

3) Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications. We use Matplotlib for visualizing data and results, such as displaying the initial condition of the tumor or plotting the evolution of tumor density over time. This visualization is crucial for validating the model's performance and understanding the tumor growth dynamics. Effective visualizations help in interpreting the model's output, providing insights into the effectiveness of the training process and the accuracy of the neural network in modeling tumor growth [10].

4) Torch Autograd

torch.autograd is PyTorch's automatic differentiation engine that powers neural network training by providing the gradients of tensors. It is essential for implementing custom gradient-based optimization algorithms. This module is used extensively to compute the derivatives required by the Diffusion-Reaction Equation. These derivatives are part of the loss function used to train the PINN. The ability to calculate derivatives efficiently and accurately is critical for ensuring that the PINN adheres to the underlying physical equations, which directly influences the reliability and stability of the model predictions [8], [11], [4].

5) Standard Python Libraries

These Libraries provide a wide range of functionalities such as "os" for interacting with the operating system, "time" for timing the execution of code, "copy" for creating deep copies of data structures, and "functools" for higher-order functions and operations on callable objects. "os" is used for directory management like creating a directory for saving outputs, "time" to monitor the duration of training epochs, copy to manage versions of the neural network during training, and "functools" for enhancing the functionality of functions, especially when dealing with partial functions or decorators [12]. These utilities enhance the code's functionality, improve its efficiency, and enable robust management of the file system and execution states, which are crucial for large-scale training tasks. Each of these libraries contributes to different aspects of the modeling process, from the core computations and visualization to utility functions and system operations, forming an integrated framework that supports the sophisticated requirements of modeling tumor growth with PINNs [4]-[12].

C. Training Strategies

Loss Function: The training of the model revolves around a bespoke loss function, which is integral to the learning process. This function is a composite of three distinct components: it quantifies the model's deviation from the Diffusion-Reaction equation evaluates how well the model adheres to the specified boundary conditions, and assesses the accuracy of the model's output against the known initial condition. Each component of this loss function is assigned an equal weight of 1.0, signifying

that they are all equally critical for the model's training. This equal weighting ensures that the model is penalized uniformly for discrepancies in any of these aspects, promoting a balanced convergence towards accurate simulations of the physical system being modeled [4], [5]. The Loss function is defined as follows:

$$LOSS_{PDE}(x,y,t) = \left(\frac{\partial u(x,y,t)}{\partial t} - D(x,y)\frac{\partial^2 u(x,y,t)}{\partial x^2} - D(x,y)\frac{\partial^2 u(x,y,t)}{\partial x^y} - \rho u(x,y,t)\,(1-u(x,y,t))\right)^2$$

The Diffusion-Reaction equation is transformed into above equation as a Loss function for residual of the PDE.

Optimizer: For the optimization process, the model employs the Adam optimizer, which is selected for its proficiency in handling the complexities of deep neural networks. The learning rate, set at 0.002, is chosen to strike a balance between the speed of convergence and the stability of the training. The Adam optimizer is favored in many deep learning scenarios due to its adaptive learning rate mechanism, which can adjust the size of the updates to the model's weights based on the evaluation of the loss function, thereby often leading to more efficient convergence to the optimal solution [4], [5], and [7].

Epochs: The neural network undergoes training over 20,000 epochs, a duration determined through initial experimental analysis. This specific number of epochs was chosen after observing that further training showed negligible improvement in the network's performance, indicating a plateau. Such a plateau typically suggests that the model has reached its capacity to learn from the data under the current configuration, and additional training would not yield significant gains in accuracy or loss reduction [4].

D. Dataset

The dataset used to train this Physics-Informed Neural Network (PINN) isn't directly imported from external data sources like databases or data files. Instead, the dataset is dynamically generated within the code using functions designed to create sample points based on specified domains and initial conditions. As we have seen that PINN when compared to traditional neural networks doesn't need much data to be able to give more accurate results. Here's a breakdown of howashe dataset is supplied to the model through the code:

We define spatial and temporal domains. These domains determine the range over which the data points for training the model will be generated.The function get_initial_points generates grid points at the initial time (t=0) across the spatial domain. The function returns tensors representing coordinates where the initial condition of the differential equation will be applied. This is crucial for setting up the problem state at the beginning of the simulation [4], [5].

The function get_boundary_points generates points along the spatial domain's boundaries at various times. These points are essential for applying boundary conditions necessary for the differential problem [4], [5].

The function get_interior_points produces points inside the domain across the given time range. These points are where the network will learn the behavior of the solution to the differential equation during training [4], [5].

An initial_condition function is used to apply a specific initial state to the initial points. This function is crucial as it defines how the solution looks at the start of the simulation, influencing how well the model will predict future states based on the physics defined in the differential equation [4].

E. Challenges and Solutions:

One of the main hurdles was the brain's complex geometry, which can be difficult to mesh in traditional simulations. The PINN approach avoids this by learning to approximate solutions directly from scattered data points, eliminating the need for conventional meshing. Given the limited availability of high-quality, comprehensive datasets on glioblastoma, the model had to maximize learning by generating data using the PDE. This was addressed by leveraging the physics-informed component of the PINN, which guides the network towards physically plausible solutions even in the face of limited data.

Accurately implementing initial and boundary conditions required careful formulation within the loss function to ensure that they are respected throughout the training process. This involved developing custom functions to calculate losses specifically related to these conditions.

Using PyTorch and matplotlib, we visualized the initial conditions and the evolution of the tumor density over time. These visualizations not only served to verify the accuracy of the model's predictions but also provided intuitive, visual feedback on the tumor growth dynamics.

By integrating these complex systems into a cohesive neural network model using PyTorch, this project highlights the potential of advanced machine learning techniques to transform medical research and treatment planning. The careful construction and training of the network ensures that the model not only predicts glioblastoma progression accurately but does so in a way that is grounded in the actual physical processes occurring within the brain.

VI. Results, Discussion and Conclusion

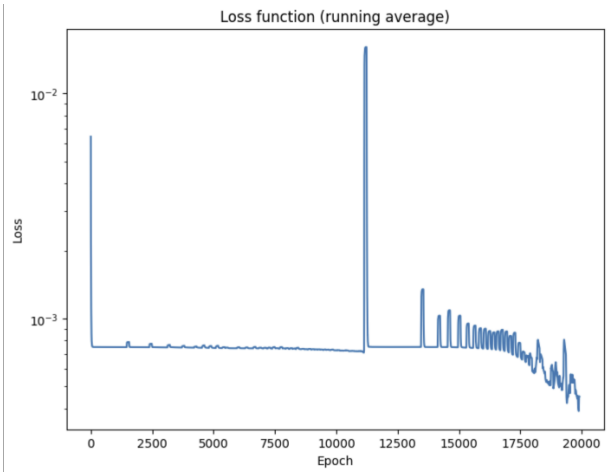

Figure 1. Graph for Loss Function.

In the Figure 1, we see the model's learning curve, depicted through changes in the loss function as the training progress through 20,000 epochs. The initial sharp decline in the graph indicates a strong start in the training, with the model quickly picking up patterns from the dataset. As time goes on, the occasional peaks in the graph represent moments where the model recalibrates, adjusting its internal parameters in response to the complexity of the data it's processing. By the end of the training, the loss levels out, suggesting the model has reached a point where it's learning as much as it can from the given information, with further training unlikely to yield improvements.

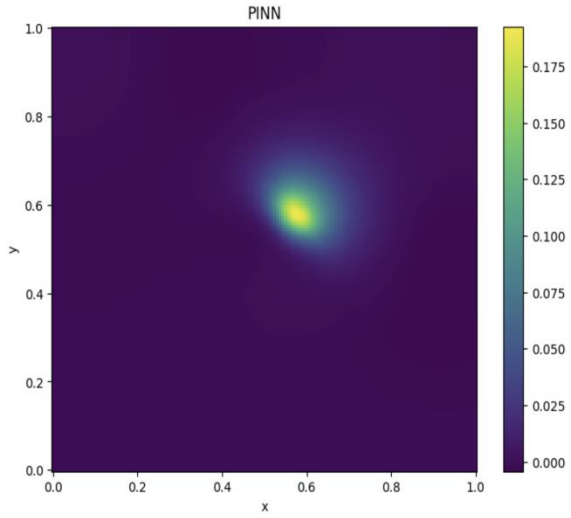

Figure 2. PINN representing initial tumor size

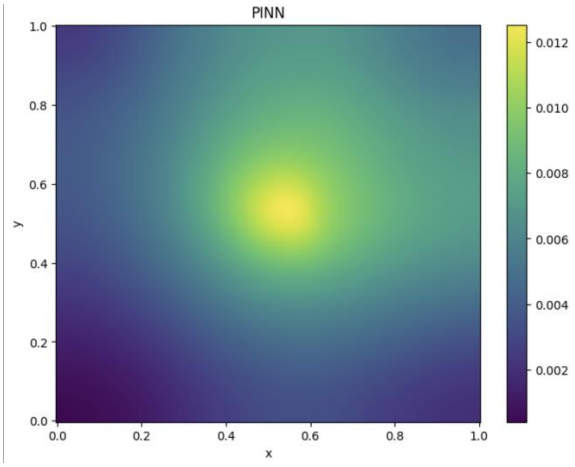

Figure 3. PINN representing tumor growth after training

The Figure 2 and Figure 3 display snapshots of the tumor model at different stages. The Figure 2, which captures the model's initial tumor state, which shows a concentrated area that represents where the tumor begins. The accuracy of this depiction is crucial—it's the starting line from which the model projects the tumor's future growth. The Figure 3 reveals how the model expects the tumor to spread, painting a gradient that fades outward from the center, illustrating the tumor's expansion and how its density changes over space. Figure 3 is a visualization of the model's understanding of tumor behavior, indicating areas that could potentially see more aggressive growth.

These visualizations are a testament to the model's ability to not only interpret the initial setup of the tumor but also to project its development through time, underscoring the potential applications in clinical planning. They confirm that the model can provide a detailed picture of tumor dynamics, an essential feature for any tool intended to aid in medical treatment strategies.

CONCLUSION

In this research I have successively demonstrated that Physics-Informed Neural Networks (PINNs) are powerful tools for modeling the complex mechanisms of glioblastoma tumor growth within the brain. By using the Diffusion-Reaction Equation into the PINN framework, the research has bridged the gap between data-driven machine learning techniques and the deterministic nature of physical laws governing biological processes.

The deployment of a custom loss function, reflecting the residual of the governing PDE along with the initial and boundary conditions, ensured that the network's training was robust and grounded in physical accuracy. The model's predictions have been validated against known conditions, and the visualizations of the tumor's evolution provide compelling evidence of the network's ability to predict the aggressive spread of glioblastoma.

Furthermore, the research has shed light on the potential of PINNs to advance the field of medical imaging and personalized treatment to treat patients. By capturing the growth of high-grade tumors with minimal data requirements, the project paves the way for developing predictive models that are not only scientifically robust but also clinically relevant.

This research has been one of methodical learning, and application of PINN. The knowledge acquired through the process of implementing PINNs has been substantial, and it can be used for various other applications aside from this project.

This research success gives lot of hope to apply PINN in solving various other medical conditions using PDEs. It is evident that continued exploration and development of generative models are very promising. The goal is to develop PINN models that can contribute meaningfully to the fields of medicine and beyond, enhancing our ability to predict, understand, and ultimately control complex systems with a precision that was previously unattainable.

ACKNOWLEDGMENT

I'd like to thank Prof Dr. Jiaqi Gong for his invaluable support during this research. His course on Deep Generative Modeling for Physical Processes was crucial in helping me understand neural networks and their application in creating generative models. His clear guidance and expertise in teaching have been essential in helping me grasp how neural networks function and how they can be used to solve scientific problems using PINNs, which I've applied in this project. His help has been a foundation for my learning, and for that, I'm truly grateful.

REFERENCES

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686-707, 2019, ISSN 0021-9991.*)*

[2] Martens C, Rovai A, Bonatto D, Metens T, Debeir O, Decaestecker C, Goldman S, Van Simaeys G. Deep Learning for Diffusion-Reaction Glioma Growth Modeling: Towards a Fully Personalized Model?. 2022 May 20;14(10):2530. doi: 10.3390/cancers14102530. PMID: 35626134; PMCID: PMC9139770.

[3] https://www.mayoclinic.org/diseases-conditions/glioma/symptoms-causes/syc-20350251

[4] Maczuga, P., Sikora, M., Skoczeń, M., Rożnawski, P., Tłuszcz, F., Szubert, M., Łoś, M., Dzwinel, W., Pingali, K., & Paszyński, M. (2023). "Physics Informed Neural Network for 2D Transient Problems (PINN-2DT) Compatible with Google Colab".

[5] A. Zhu, "Accelerating Parameter Inference in Diffusion-Reaction Models of Glioblastoma Using Physics-Informed Neural Networks," 2023. Project advisors: J. Vo and Dr. J. Lowengrub.

[6] https://medium.com/@omkar.nallagoni/activation-functions-with-derivative-and-python-code-sigmoid-vs-tanh-vs-relu-44d23915c1f4

[7] Adam Optimizer - https://pytorch.org/docs/stable/generated/torch.optim.Adam.html

[8] PyTorch - https://pytorch.org/

[9] Numpy - https://numpy.org/doc/stable/

[10] Matplotlib - https://matplotlib.org/

[11] Autograd - https://pytorch.org/docs/stable/autograd.html

[12] Python Standard Libraries - https://docs.python.org/3/library/index.html