Study on Neural Network Development Tools for Web Applications and an Attempt to Advance PHP in Machine Learning Field

Majdi Awad
Abu Dhabi, United Arab Emirates
Abu.hihad.developer@gmail.com

***Abstract*— A study and discussion of the best programming languages and tools to empower web applications with artificial neural networks.**

***Keywords*— neural networks, web applications, machine learning, python, tensorflow, keras, pytorch, deep learning, artificial intelligence, integration, optimization, performance, frameworks, training models, predictive modeling, web development, data processing, backpropagation, supervised learning, unsupervised learning, reinforcement learning, cognitive computing, api integration, data visualization, scalability, model deployment, user experience, feature engineering, data preprocessing, pattern recognition, neural network architectures, hyperparameter tuning, convolutional neural networks (cnns), recurrent neural networks (rnns), natural language processing (nlp), image recognition, object detection, sentiment analysis, transfer learning, web services, model evaluation, error analysis, explainable ai, cloud computing, deployment strategies, real-time inference, adaptive systems, edge computing, interpretability, model maintenance.**

I. INTRODUCTION

Artificial Neural Networks (ANNs) have emerged as powerful tools driving advancements in artificial intelligence (AI) and machine learning (ML). Their application in diverse domains has expanded significantly, and the integration of ANNs within web applications stands as a pivotal consideration in contemporary development. Amidst the myriad of programming languages available, the choice of the right language for implementing an ANN within a web application becomes a critical decision, influencing functionality, performance, and scalability.

In this article, I will navigate the intricate landscape of programming language selection, focusing on Python, C++, and PHP, to ascertain the optimal language for creating and deploying an ANN as an integral component of a web application. Through an empirical exploration encompassing analysis, experimentation, and evaluation, I aim to provide insights, guidelines, and considerations crucial for developers, researchers, and practitioners in making informed decisions regarding language selection when incorporating ANNs into web-based frameworks.

The significance of ANNs in driving intelligent behavior within web applications cannot be understated. These networks, inspired by biological neural systems, excel in recognizing patterns, processing complex data, and making predictions based on learned information. Their deployment within web environments enables the realization of sophisticated functionalities, ranging from personalized user experiences to real-time data analysis and decision-making.

As an advocate for harnessing the capabilities of ANNs within web applications, I embarked on a comprehensive exploration, conducting an in-depth examination of Python, C++, and PHP, considering their respective strengths, weaknesses, and suitability in the context of ANNs. Drawing upon my experience in software development, AI research, and web application design, I endeavor to offer nuanced perspectives and evidence-based insights into the considerations pivotal to this decision-making process.

The journey of selecting the appropriate programming language for implementing an ANN in a web application encompasses multifaceted criteria. These criteria encompass not only the technical capabilities of the language but also factors such as community support, ease of integration, available libraries and frameworks, performance optimization, and the alignment of language features with the demands of the application at hand.

As I delve into the intricate facets of Python, C++, and PHP, I will scrutinize their suitability in terms of ease of implementation, computational efficiency, scalability, maintainability, and compatibility with existing web frameworks. This analysis will be accompanied by practical demonstrations, code examples, and performance benchmarks, providing a holistic view of the capabilities and limitations of each language within the context of deploying ANNs in web applications.

The overarching goal of this exploration is to empower developers and decision-makers with comprehensive insights, aiding them in navigating the labyrinth of programming language choices when embarking on the integration of ANNs within web-based systems. Through a structured and rigorous examination, I strive to distill actionable recommendations and best practices, facilitating informed decision-making and fostering advancements in the realm of intelligent web applications.

Therefore, by the culmination of this exploration, I aim to offer a comprehensive guide, enriched with empirical evidence and practical wisdom, aiding stakeholders in answering the pivotal question: "Which programming language – Python, C++, or PHP – should be considered to create and run an ANN as an integral component of a web application?"

II. UNDERSTANDING ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) constitute a pivotal component of modern machine learning and artificial intelligence paradigms. Their inception was inspired by the human brain's neural structure, seeking to emulate its complex interconnectedness to process information and make decisions. Through my research, I've delved into the intricate mechanisms and diverse types of ANNs, discovering their multifaceted applications across various domains.

At their core, ANNs are computational models composed of interconnected nodes, or "neurons," organized into layers. These neurons mimic the behavior of biological neurons by receiving input, processing it through an activation function, and producing an output signal. This interconnected structure enables ANNs to learn patterns and relationships within data, a process known as training.

ANNs manifest in different architectures, each tailored for specific tasks and data types. The Feedforward Neural Network (FNN) is among the foundational architectures, with information flowing unidirectionally from input to output layers. Its simplicity and effectiveness in handling structured data make it suitable for tasks like classification and regression.

On the other hand, Recurrent Neural Networks (RNNs) exhibit a cyclic structure, enabling them to process sequential data by preserving information across time steps. This characteristic renders them apt for tasks involving sequential data, such as natural language processing and time series analysis.

Furthermore, Convolutional Neural Networks (CNNs) excel in handling grid-like data, such as images, through specialized layers like convolutional and pooling layers. These architectures possess the ability to automatically learn spatial hierarchies of features, contributing significantly to image classification, object detection, and computer vision tasks.

The diversity of ANNs grants them applicability across a spectrum of fields. In medicine, they aid in disease diagnosis from medical images, predict patient outcomes, and assist in drug discovery. Their prowess in natural language processing facilitates sentiment analysis, machine translation, and chatbots, revolutionizing communication and information retrieval systems. Industries like finance leverage ANNs for fraud detection, stock market prediction, and risk assessment.

One of the remarkable strengths of ANNs lies in their ability to learn from data. Through a process called training, ANNs adjust their internal parameters iteratively, minimizing the difference between predicted and actual outputs. This adaptive learning capability empowers ANNs to generalize patterns from training data to new, unseen data, enhancing their predictive accuracy and utility.

However, the effectiveness of ANNs heavily relies on data quality, quantity, and representation. Insufficient or biased data can lead to inaccurate predictions or reinforce existing prejudices present in the data, highlighting the importance of robust data preprocessing and ethical considerations in AI deployment.

Another aspect contributing to their strength is their ability to handle high-dimensional and complex data. ANNs can automatically extract relevant features from raw data, mitigating the need for manual feature engineering in many cases. This feature extraction capability, coupled with their scalability, allows ANNs to tackle large-scale problems effectively.

Nevertheless, ANNs are not without limitations. Their black-box nature often makes it challenging to interpret their decision-making processes, raising concerns about transparency and trust in critical applications like healthcare and judiciary systems. Additionally, training deep and complex networks demands significant computational resources and time, posing challenges in resource-constrained environments.

In conclusion, artificial neural networks represent a powerful paradigm in the realm of artificial intelligence, embodying the complexity and adaptability of the human brain in processing information. Their diverse architectures and applications underscore their significance across various domains, revolutionizing industries and shaping the technological landscape. Despite their challenges, the continuous advancements in neural network research promise further innovations and improvements, fostering their integration into diverse facets of our lives.

Artificial Neural Networks (ANNs) have emerged as instrumental tools in shaping the landscape of modern web applications, revolutionizing their functionalities and enhancing user experiences. Through extensive research and analysis, I have gained insights into the pivotal role that ANNs play in various facets of web applications, spanning from content personalization to user behavior analysis and beyond.

The advent of web applications has witnessed a paradigm shift in how information is accessed, processed, and disseminated across digital platforms. ANNs, with their ability to discern intricate patterns and extract meaningful insights from vast amounts of data, have become indispensable in augmenting the capabilities of these applications.

One prominent area where ANNs significantly contribute to web applications is in content recommendation systems. Leveraging user data such as browsing history, preferences, and interactions, ANNs enable personalized content delivery. These networks employ sophisticated algorithms to analyze user behavior, predict preferences, and suggest tailored content, thereby enhancing user engagement and retention on various web platforms. The intricate neural architectures of these systems allow for adaptive learning, continually refining recommendations based on real-time user interactions.

Moreover, ANNs play a pivotal role in natural language processing (NLP) within web applications. Through techniques like sentiment analysis, text summarization, and language translation, neural networks facilitate effective communication and information retrieval. Sentiment analysis, for instance, employs ANNs to discern emotions and opinions expressed in user-generated content, enabling web applications to gauge user sentiment and tailor responses accordingly. Similarly, language translation services leverage ANNs to deliver accurate and contextually relevant translations, fostering seamless global interactions within web platforms.

In the realm of user behavior analysis, ANNs serve as robust tools for predictive modeling and user segmentation. By analyzing vast datasets encompassing user interactions, preferences, and engagement metrics, these networks can predict future user behavior, anticipate trends, and segment users into distinct cohorts based on their characteristics and interactions. This analytical prowess empowers web applications to optimize user experiences, customize interfaces, and strategize targeted marketing campaigns, thereby maximizing user satisfaction and engagement.

Furthermore, ANNs contribute significantly to web security through anomaly detection and pattern recognition. With the exponential growth in cyber threats and malicious activities targeting web applications, neural networks offer sophisticated solutions to identify irregularities and detect potential security breaches. These networks learn normal patterns of user behavior and system interactions, enabling them to flag anomalies or suspicious activities that deviate from established norms, thereby fortifying the security infrastructure of web applications.

The integration of ANNs into web applications is not devoid of challenges. The computational complexity and resource-intensive nature of training neural networks pose hurdles in real-time deployment within web environments. Additionally, ensuring data privacy and ethical considerations surrounding the utilization of user data for training ANNs remain paramount concerns in the development and deployment of web applications leveraging these technologies.

The role of Artificial Neural Networks in web applications is paramount, reshaping the landscape of digital interactions and user experiences. From content personalization and natural language processing to user behavior analysis and security enhancement, ANNs empower web applications to deliver tailored and intuitive experiences to users worldwide. Despite the challenges, ongoing advancements in neural network research promise continued innovation, fostering the evolution of web applications towards more intelligent, adaptive, and user-centric platforms.

Artificial Neural Networks (ANNs) stand as pivotal elements reshaping the landscape of web development, imbuing web applications with a suite of transformative characteristics and capabilities. My extensive research has unraveled the intricate workings and multifaceted nature of ANNs, highlighting their indispensable role in augmenting web development across various domains. In this exploration, I delve into the key characteristics and capabilities of ANNs that underpin their significance in web development, elucidating their impact on content delivery, user experience enhancement, data analysis, and security fortification.

Central to the effectiveness of ANNs in web development is their ability to process and analyze complex data patterns. These networks are adept at handling vast volumes of data, learning from it, and discerning intricate relationships and patterns that may elude conventional algorithms. In the context of web development, this capability is leveraged for various purposes, ranging from content personalization to predictive analytics and beyond.

Content personalization represents a cornerstone application of ANNs in web development. By analyzing user behavior, preferences, and historical interactions, neural networks empower web applications to curate personalized content tailored to individual users. Through recommendation systems built on ANNs, web platforms can deliver customized experiences, suggesting relevant articles, products, or services based on a user's past engagements. This personalized content delivery enhances user engagement, satisfaction, and retention, thereby bolstering the overall success of web applications.

Furthermore, the adaptive learning mechanisms inherent in ANNs contribute significantly to enhancing user experiences within web applications. These networks possess the ability to continuously learn and adapt based on user feedback and interactions. For instance, in user interface design, ANNs can analyze user behavior patterns to optimize layouts, features, and functionalities, ensuring a more intuitive and user-friendly experience. Such adaptive capabilities enable web applications to evolve dynamically, catering to evolving user preferences and expectations.

In the realm of data analysis, ANNs serve as powerful tools for extracting insights and making informed decisions within web applications. Through techniques like predictive modeling and clustering, neural networks analyze vast datasets to forecast trends, predict user behavior, and segment user cohorts. This analytical prowess aids web developers in optimizing marketing strategies, tailoring content delivery, and streamlining user interfaces based on data-driven insights derived from ANNs.

Moreover, ANNs play a crucial role in bolstering the security infrastructure of web applications. With cyber threats posing significant risks to online platforms, neural networks offer robust solutions for anomaly detection and threat identification. By learning normal patterns of user behavior and system interactions, ANNs can detect anomalies or suspicious activities that deviate from established norms, thereby fortifying web application security and mitigating potential threats.

Despite their myriad strengths, integrating ANNs into web development poses certain challenges. The computational complexity and resource-intensive nature of training neural networks necessitate significant computational power and storage capabilities. Additionally, ensuring the ethical and responsible use of user data for training ANNs remains a paramount concern in the development and deployment of web applications leveraging these technologies.

In conclusion, the key characteristics and capabilities of Artificial Neural Networks constitute a cornerstone in advancing web development practices. From content personalization and user experience enhancement to data analysis and security fortification, ANNs empower web applications to deliver tailored, intuitive, and secure experiences to users worldwide. As research in neural networks progresses, their integration into web development continues to evolve, promising further innovations and advancements in the digital landscape.

III. Considerations for Language Selection

To start, I must define the criteria upon which I will evaluate programming languages, and they are as follows:

- Library and Framework Support
- Community and Ecosystem
- Performance
- Integration with Web Technologies
- Scalability
- Deployment and Hosting
- Ease of Development
- Security
- Compatibility with Other Components
- Documentation and Learning Resources
- Availability of sufficient experience
- Costs and Licensing
- Market Trends and Industry Standards
- Learning Curve
- Long-Term Maintenance

When it comes to developing Artificial Neural Networks (ANNs), Python, C++, and PHP each offer distinct advantages and considerations.

Python is highly favored in AI and ANN development due to its extensive ecosystem. It boasts libraries like TensorFlow, PyTorch, and Keras, specifically designed for machine learning and neural networks. Python's simplicity and readability make it beginner-friendly and accelerate development. Its vibrant community contributes to libraries and provides ample resources for learning and problem-solving. However, Python's interpretative nature might lead to slower execution compared to lower-level languages like C++ in certain scenarios. Additionally, Python's Global Interpreter Lock (GIL) can limit its performance in multithreaded scenarios.

C++, known for its performance and efficiency, is suitable for complex neural network computations. It offers low-level memory access and control, which is beneficial for optimizing resource-intensive algorithms. There are libraries available, such as TensorFlow and Caffe, that provide bindings to C++ for neural network development. However, its steep learning curve and complex syntax might slow down development cycles compared to Python. C++ requires a deeper understanding of memory management and low-level concepts, which could be challenging for some developers.

On the other hand, PHP, primarily used for web development, may integrate neural networks into web applications. Its ease of deployment on web servers makes it convenient for web-based projects. However, PHP lacks native libraries and frameworks specifically designed for neural network development when compared to Python or C++. While suitable for web-related tasks, PHP might not offer the performance required for heavy computational tasks involved in training complex neural networks.

IV. Comparative Analysis

Python, C++, and PHP each offer distinct advantages and considerations when it comes to creating artificial neural networks (ANNs) for websites and web applications. In terms of library and framework support, Python stands out with its rich selection of machine learning libraries like TensorFlow and PyTorch, providing comprehensive tools for ANN development. C++ boasts high-performance libraries, but its machine learning ecosystem isn't as extensive. PHP, while versatile for web development, has limited native support for ANNs.

Community and ecosystem heavily favor Python due to its extensive community engagement and diverse range of resources. C++ has a robust community but is more oriented towards other domains, and PHP's community, while sizable, is less focused on machine learning.

Performance-wise, C++ shines for its speed and efficiency, making it suitable for performance-critical applications. Python, though slower, offers simplicity and ease of use. PHP is typically not the choice for high-performance computing tasks.

Integration with web technologies leans towards PHP due to its native integration with web servers and frameworks. Python has a decent integration thanks to frameworks like Django and Flask, while C++ might require more effort in this regard.

Scalability considerations often favor Python and C++ due to their flexibility and performance, while PHP might face limitations in handling large-scale applications.

Deployment and hosting ease vary; PHP has straightforward deployment due to its common usage in web hosting environments. Python has become more accessible, but C++ might involve more complex deployment processes.

Ease of development tends to favor Python due to its simple syntax and readability. C++ requires more attention to memory management and syntax intricacies, while PHP provides a more straightforward but less versatile development experience.

Security-wise, Python and C++ provide more control over security measures due to their nature as lower-level languages, while PHP, though secure when used correctly, might face more vulnerabilities due to its design.

Compatibility with other components tends to favor Python due to its extensive library support and robust integration capabilities. C++ also offers high compatibility but might require more effort for certain integrations, whereas PHP's compatibility might be limited in comparison.

Documentation and learning resources heavily favor Python due to its abundant tutorials, documentation, and active community support. C++ and PHP also have good resources but might not be as comprehensive in the machine learning domain.

Availability of sufficient experience generally favors Python due to its popularity in the machine learning and web development communities. C++ has experienced developers, especially in performance-driven applications, while PHP developers might have fewer experiences with ANNs.

Costs and licensing considerations usually lean towards Python and PHP due to their open-source nature, while C++ might involve more licensing considerations for certain libraries or tools.

Market trends and industry standards heavily favor Python for machine learning applications, with C++ being prevalent in performance-critical sectors. PHP is more commonly associated with web development but less so in machine learning.

The learning curve varies, with Python offering a relatively gentle learning curve compared to C++, known for its complexity, and PHP, which can be simpler but might lack support for ANNs.

Long-term maintenance considerations often favor Python due to its readability and extensive support. C++ might require more effort for maintenance, while PHP might have more straightforward maintenance but might face limitations in scalability and performance over time.

Assigning specific percentages for each language across these criteria can be quite subjective and can vary based on individual use cases and preferences. However, considering the overall comparative analysis provided, I can say:

- Python: 45%
- C++: 30%
- PHP: 25%

V. Web Programming Languages

Most sources concerned with classifying and ranking programming languages based on their strength, popularity, and prevalence indicate that Python occupied the first position in 2023, while C++ held the third position. Meanwhile, PHP secured the thirteenth position. I believe this is relatively natural. However, what about sources specifically focused on ranking and evaluating programming languages relevant to web applications.

The landscape drastically differs concerning web application programming. Despite Python maintaining its lead, PHP occupies the second position in most sources, while C++ tends to vanish from certain rankings.

Indeed, an incredibly insightful statistic from W3Techs indicates that 43.1% of websites on the internet are built using WordPress (2023), which operates primarily with PHP. This percentage does not cover websites built using Joomla, Drupal, Magento, OpenCart, Laravel, CodeIgniter, or others. According to the same source, websites built using PHP constitute 76.7% of the total websites on the internet.

These relatively precise percentages prompt us to question the underlying reasons behind the limited popularity of PHP in the field of artificial neural networks, despite the urgent need for it in today's era.

VI. The obstacles facing PHP

Creating neural networks for web applications using PHP faces several obstacles despite its widespread usage in web development. PHP, renowned for its simplicity and ease of use, encounters challenges when tasked with handling complex tasks like implementing neural networks due to its inherent design limitations and the specialized demands of machine learning applications.

One of the primary hindrances is PHP's historical orientation as a scripting language tailored for web development rather than for scientific computing or machine learning tasks. Unlike Python, which boast robust libraries and frameworks specifically designed for machine learning, PHP lacks mature, dedicated libraries and tools for neural network creation. The absence of comprehensive machine learning-focused packages in the PHP ecosystem significantly hampers developers' ability to efficiently implement neural networks within web applications.

Moreover, PHP's performance characteristics present a significant hurdle in the context of neural network computation. While PHP excels in web server interactions and handling web requests, it often falls short in terms of raw computational speed and memory management, crucial aspects for training and running neural networks efficiently. The language's interpreted nature, compared to compiled languages, inherently adds overheads that might not be conducive to computationally intensive tasks like training intricate neural network models.

Additionally, the machine learning community predominantly gravitates towards languages like Python due to their extensive support for data manipulation, scientific computing, and a wide array of machine learning libraries like TensorFlow, PyTorch, and scikit-learn. This trend has led to an ecosystem rich in resources, documentation, and community support, factors that significantly expedite the development and deployment of machine learning models. The absence of a comparable ecosystem in PHP for machine learning impedes its adoption in this domain.

Furthermore, the evolution of neural networks and machine learning technologies requires constant updates, improvements, and the integration of new algorithms and methodologies. Python, being the language of choice for many machine learning researchers and practitioners, tends to receive these advancements first, leading to quicker adoption and integration into existing frameworks and libraries. PHP's lack of a specialized focus on machine learning hinders its ability to keep pace with the rapid advancements in the field, further diminishing its suitability for neural network development.

Interoperability and compatibility also pose significant challenges. Integrating PHP-based neural networks seamlessly into existing web applications might entail complexities in ensuring compatibility with other components or systems. The need for seamless integration with various databases, web servers, and frontend technologies adds layers of complexity that may not align well with PHP's core strengths, potentially leading to increased development time and effort.

Moreover, the expertise and skill set required to develop and optimize neural networks differ substantially from traditional web development. The machine learning domain demands specialized knowledge in linear algebra, calculus, statistics, and a deep understanding of neural network architectures. While PHP developers might possess strong web development skills, they may not necessarily have the requisite expertise in machine learning, making it challenging to bridge this knowledge gap and produce efficient, well-optimized neural network solutions within PHP.

VII. Prospects for PHP

Despite the prevailing perception of PHP as primarily a web development language, its application in creating artificial neural networks for applications and websites presents a promising landscape with several positive attributes and future prospects.

PHP's widespread use in web development, accounting for a significant majority of websites on the internet, underscores its ubiquity and familiarity among developers. Leveraging this extensive user base, the integration of neural networks within PHP-based web applications holds the potential to democratize machine learning by making it accessible to a broader community of developers. This accessibility can spark innovation and experimentation, leading to novel applications and functionalities powered by neural networks in the web development sphere.

Moreover, PHP's simplicity and ease of use serve as a double-edged sword in the context of neural network implementation. While PHP may lack specialized machine learning libraries, its straightforward syntax and user-friendly nature could lower the entry barriers for web developers looking to venture into machine learning. The simplicity of PHP may facilitate a smoother learning curve for developers transitioning from web development to machine learning, enabling them to grasp the fundamentals of neural networks more easily.

PHP's versatility in interfacing with various databases, web servers, and frontend technologies also presents a unique advantage in the integration of neural networks within existing web applications. Its compatibility with different systems and frameworks can streamline the process of incorporating machine learning functionalities into PHP-powered websites, offering a seamless user experience without significant infrastructural overhauls.

Furthermore, although PHP may not have an extensive repertoire of machine learning-specific libraries comparable to Python, efforts are underway to bridge this gap. Community-driven initiatives and emerging projects aim to develop and expand machine learning capabilities within PHP. These initiatives, though in their nascent stages, showcase the growing interest and potential for PHP to evolve into a more machine learning-friendly ecosystem. As these projects mature and gain traction, PHP's standing in the realm of neural network development could witness a noteworthy transformation.

The performance concerns often associated with PHP in the context of machine learning may also witness amelioration. Ongoing advancements in PHP, coupled with optimizations and improvements in runtime environments, could potentially enhance PHP's computational capabilities. While PHP may not match the raw computational speed of some other languages, optimizations and enhancements might bridge the performance gap to a certain extent, making it more viable for running neural networks efficiently.

In conclusion, while PHP faces challenges in establishing itself as a frontrunner in neural network development due to its origins in web development and the absence of dedicated machine learning libraries, its extensive user base, simplicity, versatility, and ongoing community-driven efforts paint an Optimistic picture for its future in this domain. As the landscape evolves, PHP stands poised to capitalize on its strengths and carve a niche for itself in the creation of artificial neural networks for applications and websites, offering unique opportunities for innovation and accessibility in machine learning for web developers.

VIII. Solutions for PHP

Improving PHP's performance for creating artificial neural networks (ANNs) within web applications involves a multifaceted approach encompassing both technical and non-technical strategies. On the technical front, optimizing PHP code by employing efficient algorithms and data structures tailored for neural network operations is crucial. Utilizing PHP extensions or libraries that offer optimized mathematical computations can significantly enhance performance. Tools like PHP-ML, though in their nascent stages, provide basic machine learning functionalities and can be leveraged to kickstart ANN development within PHP environments, optimizing execution and resource utilization.

Moreover, enhancing PHP's performance in ANN creation for web applications involves architectural optimizations. Utilizing caching mechanisms such as OpCode caching or data caching with tools like Memcached or Redis can alleviate the computational burden on PHP scripts by reducing redundant computations and database queries. Additionally, adopting asynchronous processing techniques or implementing task queues using technologies like RabbitMQ or Redis can offload heavy ANN computations to background processes, thereby improving responsiveness and scalability of web applications.

On the non-technical front, scaling PHP's performance in ANN creation involves considering infrastructure enhancements. Upgrading server hardware or migrating to more powerful hosting environments can offer improved computational resources and faster execution times for PHP scripts handling ANN operations. Employing content delivery networks (CDNs) for serving static assets and optimizing network latency can enhance overall web application performance, indirectly benefiting ANN-related functionalities.

Furthermore, investing in developer expertise and training in machine learning concepts can optimize ANN creation within PHP applications. Enabling developers to leverage best practices in neural network design, optimization techniques, and model evaluation can lead to more efficient ANN implementations. Encouraging a culture of continuous learning and staying updated with advancements in PHP and machine learning can further enhance performance by leveraging the latest tools and methodologies available within the ecosystem.

At the end, enhancing PHP's performance in creating artificial neural networks for web applications necessitates a holistic approach encompassing technical optimizations in code, architecture, and infrastructure alongside non-technical strategies focusing on developer expertise and staying abreast of advancements in machine learning and PHP ecosystem. By amalgamating these strategies, PHP's capabilities in handling ANN operations within web applications can be significantly enhanced, offering improved performance and scalability.

IX. PHP library for ANNs (PHPNeuroForge)

Creating PHPNeuroForge stemmed from recognizing PHP's prevalence in web development and the increasing importance of artificial neural networks (ANNs) in web applications. Despite existing libraries, my pursuit was fueled by a strong desire to elevate PHP's role in crafting neural networks for web-based projects. PHPNeuroForge represents a new iteration rather than a pioneering solution, aiming to refine and advance PHP's usage within the realm of ANNs.

Choosing to embark on PHPNeuroForge was driven by the realization that despite available options, there's untapped potential for PHP-centric neural network libraries. Merging PHP's capabilities, beloved by a vast developer community, with the transformative power of neural networks motivated this initiative. It doesn't seek to reinvent the wheel but to introduce innovative dimensions, fostering an environment where ANNs seamlessly integrate into PHP-based endeavors.

An inherent aspect of PHPNeuroForge is its open-source nature, a deliberate choice to foster global collaboration among PHP developers. Embracing an open-source approach aims to cultivate an ecosystem where developers contribute, refine, and augment PHPNeuroForge seamlessly. This inclusivity not only nurtures collective knowledge but also ensures continual evolution, injecting PHPNeuroForge with diverse perspectives and fresh functionalities.

The vision for PHPNeuroForge extends beyond code; it aims to be a community-driven initiative where PHP enthusiasts and ANN aficionados converge. This open-source ethos promotes transparency, knowledge-sharing, and collective innovation, creating an environment where expertise and creativity converge to shape a robust neural network library for PHP.

PHPNeuroForge aims to bridge the gap between PHP and neural network development, democratizing the use of ANNs within the PHP ecosystem. This journey isn't solitary but an inclusive expedition, inviting PHP enthusiasts, machine learning aficionados, and developers to contribute their expertise, ideas, and innovations. PHPNeuroForge embodies a collaborative vision, showcasing the potential unlocked when the PHP community unites to pioneer advancements in neural network integration for web-centric domains.

X.PHPNeuroForge Structure

Creating a library structure akin to NumPy for PHP involves organizing functionalities into modules or classes for seamless usage. Here's a broad outline of how you might structure the PHPNeuroForge library:

1) *Core Module/Package:*
    - A) Neural Network Class:
        - a) Contains methods for creating and manipulating neural networks.
        - b) Submodules might include:
            - i) Layers:
                - (1) Classes for different types of layers (e.g., input, hidden, output).
            - ii) Activation Functions:
                - (1) Classes/functions for various activation functions (ReLU, sigmoid, tanh, etc.).
            - iii) Loss Functions:
                - (1) Classes/functions for different loss functions (MSE, cross-entropy, etc.).
            - iv) Optimizers:
                - (1) Classes for optimization algorithms (gradient descent, Adam, RMSprop, etc.).
            - v) Regularization Techniques:
                - (1) Classes for regularization methods (L1/L2 regularization, dropout, etc.).
            - vi) Utilities:
                - (1) Helper functions or classes for matrix operations, data preprocessing, etc.
2) *Data Handling Module/Package:*
    - A) Data Structures:
        - a) Classes for handling data efficiently (e.g., matrices, tensors).
    - *B) Data Preprocessing:*
        - a) Functions/classes for data normalization, encoding, splitting, etc.
3) Utilities Module/Package:
    - A) Math Functions:
        - a) Classes or functions for mathematical operations.
    - B) File I/O:
        - a) Classes/functions for reading/writing neural network models or data.
4) *Documentation and Examples:*
    - A) Documentation:
        - a) Detailed documentation explaining the library's functionalities, methods, and examples.
    - B) Example Scripts:
        - a) Practical examples demonstrating the usage of various functionalities within the library.
5) Tests:
    - A) Test Suites:
        - a) Unit tests for each module or class to ensure functionality and avoid regressions.

The structure should prioritize clarity, modularity, and ease of use. This layout allows users to import specific components they need, maintains code organization, and enables straightforward expansion with additional functionalities or modules. Remember, adjusting this structure to best fit the specifics of PHP and your intended user base is crucial for PHPNeuroForge's usability and success.

XI.PHPNeuroForge Core

The core folder stands at the nucleus of PHPNeuroForge, housing crucial elements vital for crafting neural networks within PHP. Each file within this directory embodies specific functionalities, handpicked and meticulously designed to form a cohesive and comprehensive neural network toolkit.

In NeuralNetwork.php, I crafted the centerpiece—the NeuralNetwork class. This class forms the backbone of the library, orchestrating network creation, training, and usage. The constructor, chosen by me, initializes the network, configuring its layers, activation functions, and optimizer. The method forwardPass, designed for executing forward computations, applies activation functions and performs layer-wise operations, producing essential outputs. Furthermore, I aimed to include additional methods for training, prediction, backpropagation, and evaluation to ensure a robust network management framework.

Within Layers.php, I designated classes representing distinct network layers. From the InputLayer, responsible for initial data intake, to the HiddenLayer, managing intricate computations, and the OutputLayer, culminating the network's output, each class delineates specific functionalities intrinsic to its corresponding network layer.

ActivationFunctions.php houses fundamental activation functions integral to neural networks. I specifically selected functions like ReLU, sigmoid, and tanh for their crucial roles in introducing non-linearity, facilitating complex learning behaviors within network layers.

In LossFunctions.php, I included essential algorithms like Mean Squared Error (MSE) and Cross-Entropy to compute loss between predicted and actual values. This selection enables effective performance assessment during network training, aiding optimization strategies.

Optimizers.php presents diverse optimization algorithms, including Gradient Descent, Adam, and RMSprop. I incorporated these classes to offer developers a range of optimization techniques crucial for refining network parameters and enhancing training efficiency.

RegularizationTechniques.php features L1 and L2 regularization alongside Dropout techniques, selected by me to address overfitting concerns. These classes play a pivotal role in maintaining model simplicity, improving generalization, and preventing excessive reliance on training data.

Lastly, Utilities.php houses indispensable helper functions and classes vital for various network operations. From matrix multiplication functions aiding complex computations to data preprocessing methods, this file offers foundational support crucial for network operations and data manipulation.

Collectively, the core folder and its contents constitute an intricately crafted toolbox, meticulously curated to empower developers within the PHPNeuroForge ecosystem. These components offer an array of tools and methodologies, providing the essential infrastructure needed to craft sophisticated neural networks for diverse applications within the PHP landscape.

XII.PHPNeuroForge Data Handling

Within the PHPNeuroForge library, the data_handling folder represents a dedicated module focused on managing data structures and preprocessing techniques essential for effective neural network operations. Each file within this directory, crafted by me, encapsulates distinct functionalities crucial for handling and preparing data before it enters the neural network framework.

Starting with DataStructures.php, I designed this file to host classes representing pivotal data structures used extensively in neural network computations. The Matrix class encapsulates methods for matrix operations such as addition, multiplication, and dimension handling, catering to the fundamental mathematical operations indispensable in neural network computations. Simultaneously, the Tensor class provides functionalities for tensor operations, encompassing slicing, reshaping, and other tensor-specific manipulations, catering to higher-dimensional data handling within the network.

Moving on to DataPreprocessing.php, I assembled a comprehensive toolkit essential for preparing data before its integration into the neural network. The DataPreprocessing class offers a suite of methods encompassing data normalization, encoding, and splitting. These methods are crucial for ensuring that the input data adheres to appropriate formats and scales, enhancing the network's ability to learn effectively from diverse datasets. Additionally, auxiliary functions like normalizeData, encodeData, and splitData augment the preprocessing toolkit, empowering developers to handle data-specific requirements with ease. These functions provide normalization within specified ranges, categorical data encoding, and dataset splitting into training, validation, and test sets, respectively.

Collectively, the data_handling folder and its constituent files represent a vital module within PHPNeuroForge, fostering a robust environment for managing data and preprocessing techniques crucial for neural network functionality. These functionalities, meticulously designed and organized, empower developers within the PHPNeuroForge ecosystem to manipulate, preprocess, and optimize data effectively, ensuring its compatibility and readiness for utilization within neural networks.

XIII.PHPNeuroForge Utilities

Within the PHPNeuroForge library, the utilities folder serves as a dedicated repository housing essential helper functionalities crucial for diverse neural network operations. Each file within this directory, meticulously designed by me, encapsulates distinct yet interrelated functionalities, supporting developers in managing mathematical computations, file input/output operations, and more.

Starting with MathFunctions.php, I crafted this file to accommodate fundamental mathematical operations integral to neural network computations. The functions within this file, such as matrixMultiply, facilitate matrix multiplications essential for layer computations, while elementwiseOperation allows element-wise manipulations crucial for various mathematical operations within neural networks. Additionally, activationDerivative computes derivatives of activation functions, a pivotal component in backpropagation during network training. Each function is extensively commented, providing detailed insights into their usage, parameters, and examples, aiding developers in understanding and utilizing mathematical operations effectively within their neural network implementations.

Moving to FileIO.php, this file houses classes and methods dedicated to file input/output operations specific to neural networks. The ModelIO class features methods like saveModel and loadModel, designed to serialize and deserialize neural network models, enabling developers to save and retrieve models effortlessly. Meanwhile, the DataIO class offers functionalities such as readCSV and writeCSV for handling CSV file operations, crucial for managing and preprocessing datasets before feeding them into neural networks. These functions are meticulously documented, empowering developers with clear guidance on utilizing file operations for seamless integration of data and model management within their neural network projects.

Collectively, the utilities folder and its contents within PHPNeuroForge constitute a comprehensive toolkit, meticulously curated to augment and streamline neural network development. These functionalities, encompassing mathematical computations and file handling operations, are meticulously designed and extensively commented, serving as a robust support system for developers within the PHPNeuroForge ecosystem. They empower developers to effectively manage mathematical computations, handle data, and seamlessly integrate neural network models, facilitating a more efficient and streamlined development process.

XIV.PHPNeuroForge Documents & Examples

Within the PHPNeuroForge library, the docs_examples folder represents a pivotal repository curated by me to provide comprehensive documentation and practical examples crucial for developers engaging with the library. Each file within this directory is strategically designed to serve distinct purposes, aiding developers in understanding, integrating, and leveraging PHPNeuroForge effectively within their projects.

Commencing with README.md, I crafted this file as a centerpiece offering an overarching view of PHPNeuroForge. Through this markdown file, I meticulously detailed the library's authorship, established in 2024 by me, Majdi M. S. Awad, along with the version as 1.0.0 and the licensing information under the MIT License. It serves as an introductory guide, presenting an overview of the library's purpose, installation guidelines, usage instructions, and the underlying licensing structure. This document, meticulously composed, functions as a gateway for developers, elucidating the library's essence and serving as a starting point for their exploration and utilization.

Next, example_usage.php stands as a practical compendium, meticulously designed to showcase a myriad of examples outlining the functionalities of PHPNeuroForge. Curated by me, this PHP script encompasses numerous functions, each dedicated to a specific aspect or feature of the library. Through these functions, I selected and illustrated diverse use cases, including creating neural network instances, training networks with varying datasets, evaluating network performance, and other pertinent functionalities. While not containing a massive 500 examples, it is structured to aid developers in navigating and understanding the library's capabilities efficiently. This file serves as a practical guide, offering tangible code snippets to facilitate developers' comprehension and implementation of PHPNeuroForge functionalities within their projects.

Lastly, installation_guide.md is meticulously crafted to furnish developers with a comprehensive step-by-step manual for installing PHPNeuroForge. Authored by me, this markdown document delineates the installation process in detail, meticulously outlining the procedures required to integrate the library seamlessly into their PHP projects. Through this document, developers are guided through essential steps, from acquiring PHPNeuroForge, whether via direct download or Composer installation, to incorporating the necessary library files within their projects. It stands as a valuable resource, meticulously structured to streamline the setup process and ensure developers embark on their PHPNeuroForge journey without hurdles.

Collectively, the contents within the docs_examples folder serve as an invaluable resource center, meticulously curated and structured to empower developers engaging with PHPNeuroForge. These documents, meticulously composed and detailed, cater to various aspects, from providing an overarching introduction and installation guidance to furnishing practical code examples. They stand as pillars supporting developers in comprehending, integrating, and harnessing the capabilities of PHPNeuroForge effectively within their PHP projects, ensuring a smoother and more efficient development experience.

XV.PHPNeuroForge Test

The tests folder within the PHPNeuroForge library serves as a dedicated repository meticulously designed by me to house a comprehensive suite of unit tests, ensuring the functionality and integrity of the library's components. Each file within this directory is purposefully crafted to host test cases for distinct modules or functionalities, enabling developers to validate and confirm the correctness of PHPNeuroForge's operations seamlessly.

Commencing with NeuralNetworkTest.php, this file contains a series of meticulously designed test methods specifically tailored to evaluate and validate the functionalities of the Neural Network module within PHPNeuroForge. Authored by me, these tests encompass scenarios for creating neural networks, training them with various datasets, performing predictions, and assessing accuracy. Leveraging PHPUnit assertions, such as assertEquals and assertTrue, I've structured these tests to verify the expected behaviors of the Neural Network functionalities, ensuring reliability and correctness.

Moving on to MathFunctionsTest.php, this script is meticulously crafted to encompass an array of test cases meticulously designed by me to assess the mathematical functionalities embedded within PHPNeuroForge. Through carefully selected scenarios, these tests scrutinize critical mathematical operations like matrix multiplication, element-wise operations, and activation function derivatives. Utilizing PHPUnit's assertion methods, I've validated these functions' outputs against expected results, ensuring accuracy and consistency in mathematical computations.

Lastly, FileIOTest.php stands as a meticulously composed suite of tests focused on validating the File I/O functionalities within PHPNeuroForge. Authored by me, these tests encompass scenarios for saving and loading neural network models and handling data input/output operations like reading and writing from CSV files. Leveraging assertions such as assertInstanceOf and assertEquals, I've rigorously verified the accuracy of these file-related functionalities, ensuring seamless operations while handling models and data within PHPNeuroForge.

Collectively, the contents within the tests folder epitomize a robust suite of unit tests meticulously designed and structured to validate every aspect of PHPNeuroForge. These test scripts, meticulously crafted by me, serve as a critical safeguard, ensuring the library's functionalities remain reliable, accurate, and consistent across diverse scenarios. By executing these tests, developers gain confidence in PHPNeuroForge's functionalities, fostering a solid foundation for reliable and efficient neural network development within the PHP ecosystem.

XVI.PHPNeuroForge Installation

I have published PHPNeuroForge V1.0.0 on Github and you can check the source on https://github.com/studentsworldgithub/PHPNeuroForge. Installing PHPNeuroForge manually without Composer involves downloading the library's source code and including it in your PHP project directly. Here's a step-by-step guide:

Download PHPNeuroForge from Github then extract the files inside your project. After that you will be able to include PHPNeuroForge in Your Project. Once included, you can start using the PHPNeuroForge classes and functionalities within your PHP scripts.Instantiate classes, call methods, and utilize the functionalities provided by PHPNeuroForge based on your requirements.By following these steps, you can manually download the PHPNeuroForge source code, include the required PHP files in your project, and utilize the functionalities provided by the library directly within your PHP scripts. Adjust the file paths and inclusions according to your project's directory structure and usage needs.

XVII.PHPNeuroForge Sample Projects

To demonstrate the effectiveness of the library and its ability to assist web developers in creating artificial neural networks, it was essential for me to experiment with the library. This involved creating three neural networks using it and ensuring their proper functionality. The following is the project completed utilizing the PHPNeuroForge. https://github.com/studentsworldgithub/PHPNeuroForge---Deep-Learning-Models-for-Accurate-Classification-of-RGB-and-HeB-Stars.

The use of PHP for this project opens up numerous opportunities as it facilitates accessibility for non-specialists. We can now develop a comprehensive website and seamlessly integrate this neural network into it, presenting its results attractively across various internet browsers. This can be achieved without excessive effort, multiple tools, intricate programming languages, or complex integrations. We can easily embed it within straightforward websites built using WordPress, Joomla, OpenCart, Magento, or similar platforms.

XVIII.Pythons issues with Web

The Python language stands out as one of the most prevalent programming languages in the fields of artificial intelligence and machine learning. This is attributed to several reasons elaborated upon in this paper. However, despite its prominence, Python remains a language that is not specialized for web development, and some of its drawbacks include:

1) Performance: Python can be slower compared to some other languages like C++ or Java. In high-performance scenarios, such as handling a massive number of concurrent requests or performing complex calculations, this can be a drawback.
2) Asynchronous Programming: Although Python has libraries like asyncio for asynchronous programming, it's not as deeply ingrained in the language as in some others like Node.js. This might affect performance in applications that heavily rely on asynchronous operations.
3) Threading Limitations: Due to the Global Interpreter Lock (GIL), Python has limitations in utilizing multiple cores efficiently for CPU-bound tasks. This can impact performance in certain scenarios where parallel processing is crucial.
4) Not Native to Web Browsers: While Python can be used for server-side scripting, it's not natively supported in web browsers. JavaScript remains the primary language for front-end web development.
5) Less Common in Some Web Development Areas: While Python is widely used in web development, especially with frameworks like Django and Flask, in certain specialized areas like real-time systems or high-frequency trading where low-level languages are preferred, Python might not be the first choice.
6) Limited Mobile Development: Although there are ways to use Python in mobile development through frameworks like Kivy or platforms like BeeWare, it's not as popular as languages like Java or Swift for native mobile app development.
7) Difficulty in Switching Versions: Python has undergone significant changes between versions 2 and 3, and despite efforts to encourage migration to Python 3, there are still legacy systems using Python 2. This can lead to compatibility issues when integrating older systems with newer ones or when using libraries that haven't transitioned to Python 3.
8) Resource Consumption: Python's runtime can consume more memory compared to some other languages. This can be a concern for applications running on systems with limited resources or when scaling up to handle a large number of simultaneous requests.
9) Rendering for Search Engine Crawlers: Some Python web frameworks (like those using heavy client-side rendering or single-page applications) might present challenges for search engine crawlers to navigate and index content. If the content isn't easily accessible or visible without executing JavaScript, it might affect SEO.
10) Speed and Performance: As mentioned earlier, Python might have performance issues compared to some other languages. Slow-loading websites can negatively impact SEO rankings as search engines prioritize user experience, including page load times, when determining rankings.
11) Dynamic URLs and URL Structure: Python-based web frameworks might produce dynamic URLs with query parameters that aren't as descriptive or user-friendly. Clean, static URLs tend to perform better in SEO. However, this issue can often be addressed through URL rewriting and structuring.
12) Handling Redirects and Canonicalization: Ensuring proper handling of redirects, canonicalization, and avoiding duplicate content is crucial for SEO. Python developers need to implement these practices correctly within their web applications to avoid SEO penalties.
13) Content Management and SEO-Friendly Markup: Python frameworks might require more effort to implement SEO-friendly markup, structured data, and content management systems compared to some specialized CMS platforms that are specifically designed with SEO in mind.
14) Server-Side Rendering: Python's server-side rendering capabilities might differ across frameworks. Ensuring that content is readily available and rendered on the server side can positively impact SEO by providing search engines with easily readable and indexable content.
15) Caching Strategies: Python web applications might need robust caching strategies to improve performance. However, improperly configured caching mechanisms could lead to issues with outdated content being indexed or displayed, impacting SEO.

Actually, Python itself doesn't directly cause SEO issues; rather, it's the implementation, configuration, and design choices made within Python-based web applications that can impact SEO. By employing best practices, optimizing code, and using appropriate frameworks, developers can mitigate these issues and create Python-based websites that perform well in terms of SEO.

XIX.C++ issues with Web

A rational individual does not dispute the strength, quality, and performance of the C++ language. Personally, I can confidently state that it is one of the best programming languages ever. However, employing it in the realm of web development raises several substantial issues, including, but not limited to, the following:

1) Memory Management: C++ requires manual memory management, which can lead to memory leaks, segmentation faults, and other errors if not handled properly. In web applications where scalability and stability are crucial, managing memory can be a challenge.
2) Performance vs. Development Time: While C++ is highly performant and efficient, achieving that performance often involves writing more lines of code compared to higher-level languages. In web development where faster development cycles are often desired, this can be a drawback.
3) Lack of High-Level Web Frameworks: C++ doesn't have as many high-level web frameworks as languages like Python, JavaScript, or Ruby. This means developers might have to handle lower-level networking and server-side tasks themselves, potentially increasing development time and complexity.
4) Platform Dependence and Portability: C++ code is less portable compared to some other languages due to its

platform-dependent nature. Web applications often need to run on various operating systems and devices, and achieving seamless portability can be more challenging with C++.

5) Slower Development for Web Applications: Building web applications in C++ might take more time and effort compared to languages that have extensive web-oriented libraries and frameworks. This might hinder rapid development and iteration cycles common in web development.
6) Concurrency and Scalability: While C++ supports multithreading and concurrency, managing these aspects manually can be complex and error-prone. This can become a challenge when building highly scalable web applications that require handling multiple concurrent requests efficiently.
7) Community and Resources: While C++ has a large and dedicated community, its focus has historically been on systems programming, game development, and other areas rather than web development. This might result in fewer readily available resources specifically tailored for web development in C++.

I don't need to present more issues related to the C++ language in the field of web development, as its usage may seem somewhat uncommon, despite the existence of several frameworks employed in the market.

XX.Conclusion

Specialized web developers often possess expertise in programming languages and technologies specific to the web, such as PHP, JavaScript, CSS, among others. However, it's less common to find web developers with extensive experience in languages like Python or C++. Consequently, the idea of using PHP for building neural networks and implementing them in web applications and sites seems more practical, straightforward, and cost-effective. Moreover, utilizing PHP significantly reduces the effort, cost, and complexities involved in integration, compatibility, hosting, and related areas.

With the advancement and widespread integration of artificial intelligence across various domains, it's truly surprising that PHP lacks libraries and tools similar to those possessed by Python. This absence persists despite some timid attempts in this direction.

The library developed in this paper is nothing but another attempt to enhance the role of PHP in the realm of developing intelligent web applications using artificial neural networks.

Improving the performance of PHP for Artificial Neural Networks (ANNs) development involves several technical approaches and considerations. Here's a number of suggestions to enhance PHP's performance in the field of ANNs:

1) I recommend implementing effective caching mechanisms for network weights, model structures, and intermediate computations. Utilize techniques like opcode caching (e.g., OPcache) to cache compiled PHP code, reducing execution time and overhead.
2) I recommend exploring parallel processing capabilities to handle multiple computations concurrently. Utilize threading or asynchronous processing techniques to execute multiple network evaluations simultaneously, leveraging multi-core systems for performance gains.
3) I advise optimizing data handling and preprocessing. Use efficient data structures and algorithms to handle input data, reducing overhead during data processing and enhancing overall performance.
4) I recommend profiling the code to identify performance bottlenecks. Optimize critical sections by refining algorithms, reducing unnecessary computations, and fine-tuning resource-intensive operations.
5) Consider using Just-In-Time (JIT) compilation techniques if available. JIT compilers can dynamically optimize and compile code during runtime, potentially improving execution speed for certain computations.

By implementing these technical strategies and leveraging external libraries or extensions and creating new specialized PHP libraries, PHP developers can significantly improve the performance of PHP for ANNs development, making computations faster and more efficient.

XXI.References

[1] w3techs, https://w3techs.com/technologies/details/cm-wordpress, 2023.

[2] w3techs, https://w3techs.com/technologies/details/pl-php, 2023.

[3] stackscale, https://www.stackscale.com/blog/most-popular-programming-languages, 2023.

[4] Yılmaz Yörü, *The Pros And Cons of The C++ Programming Language*, embarcadero.com, 2023, https://blogs.embarcadero.com/the-pros-and-cons-of-the-c-programming-language/.

[5] Yulia Gavrilova, *Pros and Cons of Python Programming Language*, serokell.io/, 2023, https://serokell.io/blog/python-pros-and-cons.

[6] I. Stančin and A. Jović, *An overview and comparison of free Python libraries for data mining and big data analysis*, IEEE, 2019.

[7] Lwin Khin Shar, Lionel C. Briand, and Hee Beng Kuan Tan, *Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning*, IEEE, 2014.

[8] Marcus Christie , Suresh Marru , Eroma Abeysinghe , Dimuthu Upeksha , Sudhakar Pamidighantam , Stephen Paul Adithela , Eldho Mathulla , Aarushi Bisht , Shivam Rastogi , and Marlon Pierce, *Django Content Management System Evaluation and Integration with Apache Airavata*, ACM, 2020.

[9] Liton Chandra Voumik, R. Karthik, A. Ramamoorthy, and Anurag Dutta, A Study on Mathematics Modeling using Fuzzy Logic and Artificial Neural Network for Medical Decision Making System, IEEE, 2023.