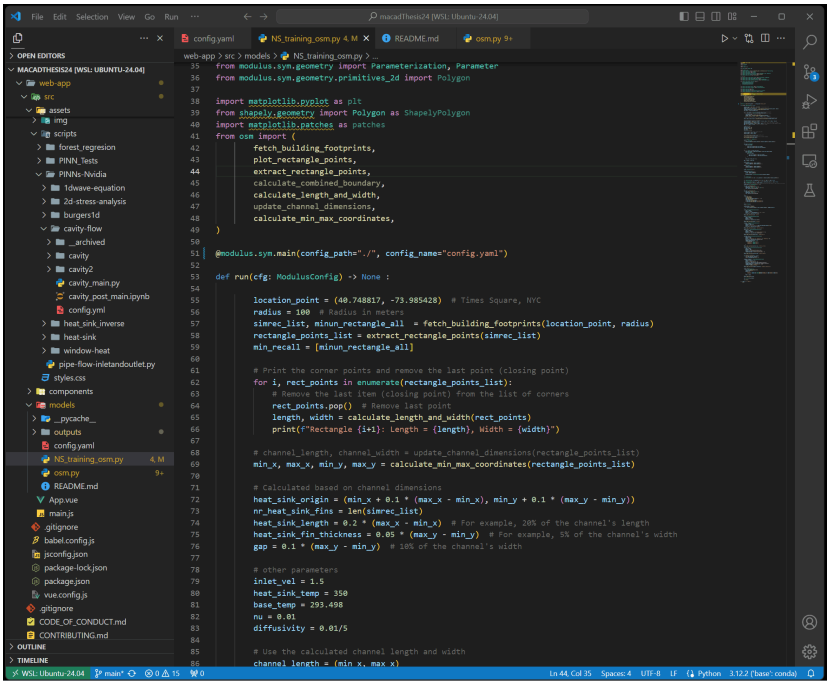
WIP- Urban Wind Flow Modeling with Physics-Informed Neural Networks (PINNs)

Author: Andres Roncal
Institution: IAAC: AI 2023-24
Date: August 29, 2024
Repository: GitHub Link



Abstract

This study presents a web-based platform designed to simulate urban wind flow using Physics-Informed Neural Networks (PINNs). The platform integrates real-time data from the NOAA, OpenWeatherMap, and OpenStreetMap, providing accurate and dynamic simulations of wind behavior in complex urban environments. The methodology encompasses advanced PINN model development, and frontend and backend integration utilizing technologies like Vue.js, Three.js, WSL, NVIDIA GPU, Ubuntu-24.04, JAX, TensorFlow, OpenStreetMap, OpenWeatherMap, NVIDIA Modulus, and PyTorch. The results demonstrate the efficacy of PINNs in capturing intricate wind patterns, offering valuable insights for urban planners, architects, and researchers dedicated to promoting sustainable and comfortable urban spaces. Future work will explore incorporating additional environmental factors and enhancing computational efficiency, solidifying the platform's role in informed urban development and environmental management.

Introduction

Urbanization has led to increasingly complex cityscapes, where the interaction between built environments and natural elements profoundly affects human comfort and environmental sustainability. Wind flow within urban areas influences various factors, including air quality, thermal comfort, energy efficiency, and structural integrity. Understanding and accurately modeling urban wind flow are essential for designing cities that are resilient, sustainable, and conducive to healthy living.

Traditional methods for simulating wind flow, such as Computational Fluid Dynamics (CFD), often require significant computational resources and rely on proprietary software, limiting accessibility and scalability. Recent advancements in machine learning, particularly the development of Physics-Informed Neural Networks (PINNs), offer promising alternatives by integrating physical laws directly into the learning process, thus reducing computational costs while maintaining high accuracy.

This study introduces an open-source web application that leverages PINNs for real-time urban wind flow simulation. By integrating comprehensive datasets from NOAA, OpenWeatherMap, and OpenStreetMap, the platform provides detailed and dynamic visualizations of wind behavior across various urban scenarios. The application aims to serve as a collaborative tool for professionals and researchers, facilitating informed decision-making in urban planning and contributing to the mitigation of urban heat islands and improvement of natural ventilation strategies.

The following sections detail the significance of accurate urban wind flow modeling, the theoretical foundation and advantages of using PINNs, the objectives guiding this project, the methodology employed, and the results achieved. The paper concludes with discussions on the implications of the findings and potential avenues for future research and development.

Importance of Urban Wind Flow Modeling

Environmental and Health Impacts

Urban wind flow plays a critical role in dispersing pollutants, regulating temperature, and maintaining air quality within city environments. Inadequate ventilation can lead to the accumulation of harmful pollutants, exacerbating health issues such as respiratory diseases and contributing to the urban heat island effect, where metropolitan areas experience significantly higher temperatures than surrounding rural regions. Effective wind flow modeling allows for the identification and mitigation of areas prone to poor air circulation, thereby improving public health and environmental conditions.

Energy Efficiency and Sustainability

Proper understanding and utilization of wind patterns can significantly enhance the energy efficiency of buildings and urban layouts. By strategically designing structures and open spaces that harness natural ventilation, reliance on mechanical cooling and heating systems can be reduced, leading to lower energy consumption and decreased greenhouse gas emissions. Accurate wind flow simulations support architects and urban planners in creating designs that align with sustainability goals and adapt to the challenges posed by climate change.

Urban Planning and Infrastructure Development

Wind flow considerations are essential in the planning and development of urban infrastructures such as skyscrapers, bridges, and public spaces. Accurate modeling helps in assessing the structural loads and ensuring the safety and comfort of pedestrians by preventing issues like wind tunnels and vortex shedding, which can cause discomfort or even structural damage. Integrating wind flow analyses early in the design process facilitates the creation of resilient and comfortable urban environments.

Challenges with Traditional Modeling Approaches

Conventional CFD methods, while accurate, are computationally intensive and often inaccessible due to high costs and the need for specialized expertise. These limitations hinder widespread adoption and real-time application in dynamic urban planning scenarios. The emergence of PINNs addresses these challenges by offering a more efficient, flexible, and accessible approach to fluid dynamics simulation, enabling broader usage and integration into various stages of urban development processes.

Physics-Informed Neural Networks (PINNs)

Conceptual Overview

Physics-Informed Neural Networks represent a paradigm shift in computational modeling by embedding physical laws directly into the neural network architecture. Unlike traditional machine learning models that rely solely on data-driven approaches, PINNs incorporate governing equations such as partial differential equations (PDEs) into the loss function during training. This integration ensures that the model's predictions inherently respect the underlying physics of the system being studied.

Advantages over Traditional Methods

- Reduced Data Dependency: PINNs require less observational data compared to purely data-driven models because they leverage known physical laws to guide learning, making them particularly useful in scenarios where data is scarce or expensive to obtain.
- Enhanced Generalization: By incorporating physical constraints, PINNs demonstrate improved generalization capabilities across different scenarios and conditions, maintaining accuracy even when extrapolating beyond the training data range.
- Computational Efficiency: PINNs can be trained and executed more efficiently than traditional CFD simulations, enabling faster computations and real-time applications, which are crucial for responsive urban planning and design processes.
- Flexibility and Adaptability: The neural network framework allows for easy adaptation to complex geometries and varying boundary conditions, making PINNs suitable for modeling the intricate and diverse structures found in urban environments.

Implementation in Fluid Dynamics

In the context of fluid dynamics, PINNs are trained to solve the Navier-Stokes equations, which describe the motion of fluid substances such as air and water. By minimizing the residuals of these equations across a computational domain, the network learns to predict velocity, pressure, and other relevant flow properties accurately. This approach enables the simulation of complex flow phenomena, including turbulence and vortex formation, which are critical for understanding and optimizing urban wind flow patterns.

Role of NVIDIA Modulus and PyTorch

The implementation of PINNs in this project utilizes NVIDIA Modulus, a framework specifically designed for developing and training physics-informed neural networks efficiently. Coupled with PyTorch, a widely-used deep learning library, the development process benefits from robust computational capabilities and flexibility. NVIDIA Modulus provides optimized routines and support for multi-GPU training, significantly accelerating the development and deployment of complex simulation models required for real-time urban wind flow analysis.

Objectives

Primary Goals

- Develop an Accessible Simulation Platform: Create an open-source, web-based application capable of simulating urban wind flow in real-time, making advanced modeling tools available to a broad audience, including urban planners, architects, and environmental researchers.
- Integrate Multisource Data: Seamlessly combine historical and real-time wind data from NOAA and OpenWeatherMap with detailed urban geometry from OpenStreetMap to produce accurate and context-specific simulations.
- Leverage Advanced Computational Techniques: Utilize Physics-Informed Neural Networks, powered by NVIDIA Modulus and PyTorch, to achieve high-fidelity simulations that adhere to fundamental fluid dynamics principles while maintaining computational efficiency.
- Enhance Urban Planning Practices: Provide actionable insights through detailed visualizations and analyses that support sustainable urban design decisions, aiming to improve pedestrian comfort, reduce energy consumption, and mitigate environmental issues such as the urban heat island effect.
- Foster Collaborative Development: Establish a platform that encourages community contributions, facilitating continuous improvement and adaptation to emerging needs and technologies within the urban planning and environmental modeling domains.

Secondary Goals

- Optimize Computational Performance: Explore techniques to enhance the speed and scalability of simulations, enabling their application to large-scale and complex urban scenarios without prohibitive computational costs.
- Expand Environmental Factors: Plan for the incorporation of additional variables such as temperature, humidity, and pollution levels to provide a more comprehensive environmental analysis and support multifaceted urban planning strategies.
- Ensure User-Friendly Interface: Design an intuitive and interactive frontend that allows users of varying technical backgrounds to easily input data, run simulations, and interpret results effectively.
- Validate and Benchmark Models: Conduct extensive testing and validation against empirical data and established benchmarks to ensure the reliability and accuracy of the simulation outputs.
- Document and Disseminate Findings: Produce detailed documentation and reports outlining methodologies, results, and best practices, contributing to the broader body of knowledge and facilitating replication and extension of the work by others in the field.

Data Collection and Processing

Data Sources

To achieve accurate and context-specific simulations, the platform integrates data from multiple reputable sources:

- OpenWeatherMap (OWM):
    - Supplies real-time weather data, ensuring simulations reflect current atmospheric conditions.

    - The API provides up-to-date information on wind speed, direction, temperature, and humidity.
    - Real-time data integration allows for dynamic simulations that can adapt to changing weather patterns instantly.
- OpenStreetMap (OSM):
    - Offers detailed and open-source geographic data, including precise 3D models of urban structures.
    - The Overpass API facilitates the extraction of building geometries and other relevant spatial information.
    - Data is essential for constructing accurate computational domains that reflect the complexities of urban landscapes.

Data Processing Pipeline

The data collected undergoes a systematic processing workflow to ensure compatibility and optimal performance within the simulation framework:

- Data Extraction:
    - Automated scripts retrieve data from respective APIs, handling authentication, error checking, and data formatting.
    - Geospatial queries are constructed to obtain specific urban areas based on user-defined coordinates and parameters.
- Spatial Alignment and Transformation:
    - Geographic data is transformed into appropriate coordinate systems suitable for computational modeling.
    - Alignment procedures ensure that weather data corresponds accurately to spatial locations within the urban geometry.
    - Spatial interpolation methods are applied to estimate wind conditions across different points within the simulation domain.
- Data Integration:
    - Processed datasets are combined into a unified format, serving as input for the PINN model.
    - Metadata and auxiliary information, such as terrain elevation and land use patterns, are incorporated to enhance simulation fidelity.
- Validation and Quality Assurance:
    - Cross-referencing with additional data sources and empirical observations to validate accuracy.
    - Statistical analyses assess data reliability and identify potential sources of error or uncertainty.
    - Continuous monitoring ensures data remains up-to-date and reflects current environmental conditions.

Challenges and Solutions

- Handling Large and Complex Datasets:
    - Implemented efficient data structures and storage solutions to manage extensive spatial and temporal data.
    - Utilized parallel processing and optimized algorithms to expedite data processing tasks.
- Ensuring Data Consistency and Accuracy:

    - Developed robust validation protocols, including consistency checks and anomaly detection mechanisms.
    - Engaged in periodic updates and synchronization across data sources to maintain temporal coherence.
- Integrating Diverse Data Formats:
    - Employed standardized data models and transformation utilities to harmonize disparate data formats.
    - Leveraged geospatial libraries and tools (e.g., GDAL, GeoPandas) for seamless spatial data manipulation.

Model Development

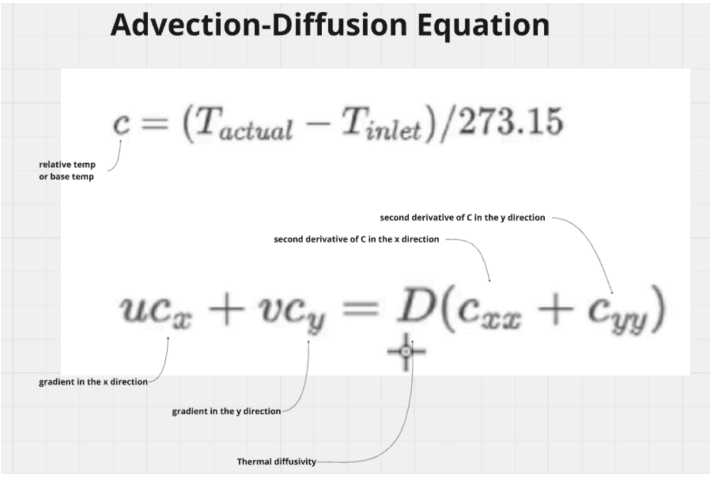
Mathematical Foundation

The core of the simulation model is grounded in the Navier-Stokes equations and Advection-Diffusion equation, which describe the motion of fluid substances. These equations account for various forces acting on the fluid, including viscosity, pressure gradients, and external forces, providing a comprehensive description of fluid flow behavior.

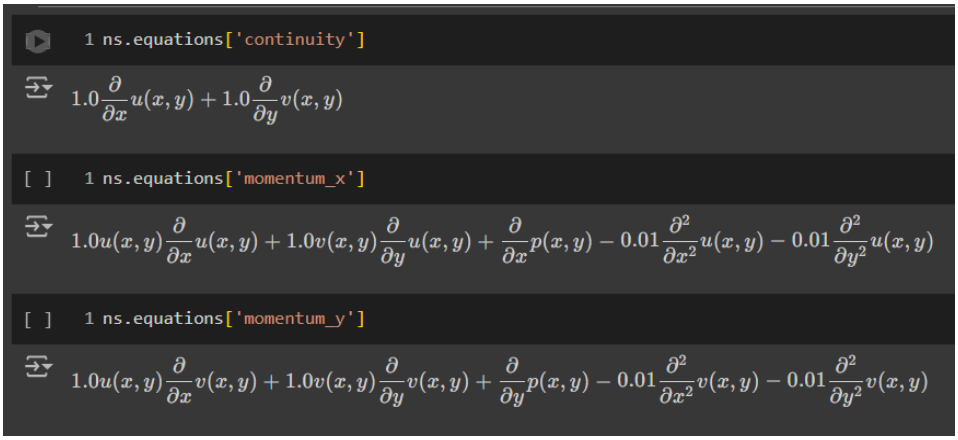
Navier-Stokes Equations:

PINN Architecture

The PINN model is constructed by embedding the Navier-Stokes equations into the loss function of a deep neural network. The network approximates the solution to these equations across the computational domain defined by the urban geometry.



Network Structure:



- Input Layer: Receives spatial coordinates (x, y, z) and temporal information (t).
- Hidden Layers: Composed of fully connected layers with activation functions (e.g., ReLU, tanh) that capture complex, nonlinear relationships.
- Output Layer: Predicts fluid properties such as velocity components (u, v, w) and pressure (p).

Loss Function Components:

- Physics Loss: Measures the residuals of the Navier-Stokes equations, enforcing adherence to physical laws.
- Boundary Condition Loss: Ensures that the solutions satisfy specified boundary conditions derived from real-world data (e.g., no-slip conditions on building surfaces).
- Initial Condition Loss: Aligns the model's initial state with observed or prescribed fluid states.
- Data Loss (Optional): Incorporates observational data points to guide the model where data is available, enhancing accuracy.

Training Process:

- The network is trained by minimizing the combined loss function using optimization algorithms such as Adam or L-BFGS.
- Training data consists of collocation points sampled across the spatial and temporal domain.
- Regularization techniques are applied to prevent overfitting and ensure smooth and physically plausible solutions.

Computational Framework

NVIDIA Modulus:

- Provides a high-level API for constructing and training PINNs efficiently.
- Supports automatic differentiation, enabling precise computation of derivatives required for evaluating PDE residuals.
- Optimized for GPU acceleration, significantly reducing training times and enabling handling of large-scale problems.

PyTorch Integration:

- Offers a flexible and extensible platform for implementing custom neural network architectures and loss functions.
- Facilitates seamless integration with NVIDIA Modulus and other deep learning tools.
- Enables easy experimentation with different network configurations and training strategies.

Handling Complex Geometries:

- The computational domain is discretized using mesh generation techniques that conform to intricate urban structures.
- Adaptive sampling strategies focus computational resources on areas with high flow complexity or interest.
- Level-set methods and signed distance functions are employed to accurately represent building boundaries within the simulation domain.

Model Validation and Testing

- Benchmarking: The model's performance is evaluated against established CFD simulations and analytical solutions for standard flow scenarios.
- Cross-Validation: Partitioning data into training and validation sets to assess the model's generalization capabilities.
- Sensitivity Analysis: Examining the influence of various parameters (e.g., viscosity, inlet velocities) on the simulation outcomes to ensure robustness.
- Error Metrics: Utilizing metrics such as Mean Squared Error (MSE), L2 norm, and relative error to quantify accuracy and identify areas for improvement.

Performance Optimization

- Parallel Computing: Leveraging multiple GPUs and parallel processing techniques to expedite training and inference.
- Model Pruning and Compression: Reducing network complexity without compromising accuracy to enhance computational efficiency.
- Hyperparameter Tuning: Systematically adjusting learning rates, network depths, and activation functions to achieve optimal performance.
- Caching and Data Streaming: Implementing efficient data handling mechanisms to minimize I/O bottlenecks during simulation runs.

System Integration and Deployment

Frontend Development

Technologies Used:

- Vue.js: Provides a progressive framework for building user interfaces, enabling reactive and component-based development.
- Three.js: Facilitates the rendering of complex 3D visualizations directly within the browser, essential for depicting intricate wind flow patterns over urban terrains.
- Geolib: Supports precise geographic calculations, aiding in accurate mapping and spatial analyses.

User Interface Design:

- Interactive Controls: Allow users to input geographic coordinates, adjust simulation parameters, and select different time frames for analysis.
- Real-Time Visualization: Displays dynamic simulations of wind flow, including velocity vectors and pressure contours, overlaid on 3D models of urban areas.
- Data Overlay: Integrates additional layers such as temperature, humidity, and pollutant concentrations for comprehensive environmental assessment.
- Responsive Design: Ensures accessibility across various devices and screen sizes, facilitating wide usability.

Example Code Snippet:

```
<template>
  <div id="app">
    <AppSidebar :weather="weatherData"
@update-coordinates="updateCoordinates" />
    <ThreeDScene :geometry="urbanGeometry"
:windData="windSimulationResults" />
  </div>
</template>

<script>
import AppSidebar from './components/AppSidebar.vue';
import ThreeDScene from './components/ThreeDScene.vue';

export default {
  name: 'App',
  components: {
    AppSidebar,
    ThreeDScene
  },
  data() {
    return {
      weatherData: {},
      urbanGeometry: {},
      windSimulationResults: {}
    };
  },
  methods: {
    updateCoordinates(coords) {
      this.fetchWeatherData(coords);
      this.fetchUrbanGeometry(coords);
      this.runWindSimulation();
    },
    fetchWeatherData(coords) {
      // Fetch weather data from OpenWeatherMap API
    },
    fetchUrbanGeometry(coords) {
      // Fetch building data from OpenStreetMap via Overpass API
    },
    runWindSimulation() {
      // Invoke backend API to run PINN simulation
    }
```

```
  }
};
</script>
```

Progress and Future Steps

Real-Time Visualization: The Journey Ahead:

While the concept of real-time visualization is an integral part of the project's goals, the current focus remains on integrating the model and building the necessary backend infrastructure. Thus far, foundational aspects like PINN model development, data collection, preprocessing, and initial backend setup using Flask/FastAPI have been completed. The visualization component, though not yet operational, represents a crucial milestone for upcoming work. In this next phase, efforts will concentrate on refining the frontend interface using Vue.js and Three.js to allow users to input coordinates, adjust simulation parameters, and visualize urban wind flows.

Model Integration and Advanced Techniques:

The immediate objective is to fully integrate the PINN model for pedestrian wind assessment. This process involves continuous testing of different techniques, including the Navier-Stokes equations and advection-diffusion models. As the real-time data retrieval from OpenWeatherMap and OpenStreetMap APIs becomes more stable, the next step will be to enhance the accuracy of the wind flow simulations and streamline data processing.

Data Overlay and Interactive Controls:

Upon successful model integration, the platform will move towards incorporating additional environmental layers such as temperature, humidity, and pollutant concentrations. This feature will enrich the simulations, providing a multi-faceted view of urban wind patterns. The design of interactive controls will allow users to adjust parameters, further enhancing the platform's adaptability to various urban planning scenarios.

Responsive Design and Frontend Development:

The current stage involves building the user interface using Vue.js and Three.js for 3D rendering, with the plan to create a responsive design accessible across different devices. Future work will focus on refining this interface to include dynamic elements, such as adjustable time frames for analysis and responsive visual feedback.

Challenges and Continuous Testing:

The integration of complex pedestrian wind assessment models into urban environments has surfaced several challenges, notably related to computational resource demands. To address this, ongoing optimization efforts will focus on techniques like parallel processing, GPU acceleration, and possibly exploring cloud-based scaling solutions. Additionally, the accuracy of the simulations relies on high-quality input data, necessitating continuous validation and integration from various sources to ensure reliability.

System Integration and Expansion of Features:

Following model integration, the focus will shift to system integration, connecting the frontend with the backend for seamless user interaction. Plans include the development of adaptive algorithms to dynamically update simulations based on real-time data. Future iterations will aim to incorporate additional environmental factors and advanced controls for scenario comparisons, fostering a more comprehensive and versatile urban wind flow simulation platform.

Collaborative Community and Open-Source Enhancements:

A significant aspect of future work is to promote the platform as an open-source tool. By fostering community involvement, the goal is to encourage contributions, such as new features, optimizations, and educational content, ultimately expanding the platform's capabilities.

Conclusion

The development of an open-source, PINN-based web application for urban wind flow simulation represents a significant advancement in accessible and efficient environmental modeling tools. By effectively integrating real-time and historical data with advanced neural network methodologies, the platform offers precise and dynamic insights crucial for sustainable urban planning and design.

The project's success demonstrates the viability of leveraging machine learning and physics-informed approaches to tackle complex environmental challenges, providing a foundation for continued innovation and application across various domains. Future enhancements and community engagement promise to expand the platform's utility and impact, contributing to the creation of healthier, more resilient, and environmentally conscious urban spaces.

References

1. NVIDIA Modulus. (n.d.). *Physics-Informed Machine Learning with NVIDIA Modulus*. Retrieved from docs.nvidia.com
2. *Scientific Machine Learning Through Physics-Informed Neural Networks*. (2022). Journal of Computational Physics, 466, 110924. https://doi.org/10.1007/s10915-022-01939-z
3. *GPT-PINN: Generative Pre-Trained Physics-Informed Neural Networks*. (n.d.). Retrieved from arxiv.org
4. *Using Hybrid Physics-Informed Neural Networks for Complex Engineering Simulations*. (n.d.). Retrieved from arxiv.org
5. *Architectural Strategies for PINN Optimization*. (n.d.). Retrieved from docs.nvidia.com
6. *Scientific Machine Learning through PINNs*. (n.d.). Retrieved from physicsbaseddeeplearning.org
7. Zappitelli, R., Najjar, S. K., Bovo, R., Maqbool, T., & Kaeuffer, A. (2023). *Machine Learning for Pedestrian-Level Wind Comfort Analysis*. Buildings, 14(6), 1845. https://doi.org/10.3390/buildings14061845
8. *Deep learning to replace, improve, or aid CFD analysis in built environment applications: A review*. (2021). Building and Environment, 207, 108459. https://doi.org/10.1016/j.buildenv.2021.108459
9. *Evaluating the pedestrian level of service for varying trip purposes using machine learning algorithms*. (2024). Scientific Reports, 14, 53403. https://doi.org/10.1038/s41598-024-53403-7
10. NVIDIA Modulus. (n.d.). *NVIDIA Modulus - Advanced Topics*. Retrieved from courses.machinedecision.com
11. *Can Taichi play a role in CFD? | Taichi Docs*. (n.d.). Retrieved from docs.taichi-lang.org
12. *Table of Contents | NVIDIA Docs*. (n.d.). Retrieved from docs.nvidia.com

13. *Regression using LightGBM - GeeksforGeeks*. (n.d.). Retrieved from geeksforgeeks.org
14. *Random Forest Regression in Python - GeeksforGeeks*. (n.d.). Retrieved from geeksforgeeks.org
15. *Computer Fluid Dynamics Python at DuckDuckGo*. (n.d.). Retrieved from duckduckgo.com
16. *Deep Learning*. (n.d.). Retrieved from docs.nvidia.com
17. *Physics-Informed Machine Learning through Modulus | Physical Loss Code*. (n.d.). Retrieved from physicsbaseddeeplearning.org
18. *CFD Analysis in Built Environment Applications Using Deep Learning: A Review*. (2024). Journal of Building and Environment, 207, 108459. https://doi.org/10.1016/j.buildenv.2021.108459
19. *Evaluating the Pedestrian Level of Service for Varying Trip Purposes Using Machine Learning Algorithms*. (2024). Nature Communications, 14, 53403. https://doi.org/10.1038/s41598-024-53403-7
20. *Computer Fluid Dynamics Python at DuckDuckGo*. (n.d.). Retrieved from duckduckgo.com
21. *Deep Learning to Replace, Improve, or Aid CFD Analysis in Built Environment Applications: A Review*. (2024). Building and Environment, 207, 108459. https://doi.org/10.1016/j.buildenv.2021.108459
22. *So, what is a physics-informed neural network?* https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/

Contributor:
Andres Roncal

Thesis Advisor:
David Andres Leon

I would like to express gratitude to IAAC for providing the academic framework and resources necessary for this research. Special thanks to David Andres Leon and my friend Will Nash for guidance and support throughout the project's development.
