ChemNODE-Ammonia: A Neural ODE Surrogate for Stiff Ammonia-Air Combustion Kinetics

Aditya Prasad (22110018)

Department of Mechanical Engineering, IIT Gandhinagar

Instructor: Prof. Shubhangi Bansude

Contents

1. Abstract

Ammonia (NH_3) is emerging as a critical carbon-free fuel for power generation and marine propulsion. Its potential has garnered significant industrial interest; for instance, **Toyota** is actively researching ammonia-fueled internal combustion engines as a pathway to carbon neutrality. However, the simulation of ammonia combustion is exceptionally computationally expensive. This is due to the extreme **numerical stiffness** inherent in detailed chemical mechanisms, where reaction timescales span from nanoseconds (radical formation) to seconds (bulk heat release). This stiffness forces traditional ODE solvers to take prohibitively small time steps, making multi-dimensional Computational Fluid Dynamics (CFD) simulations intractable.

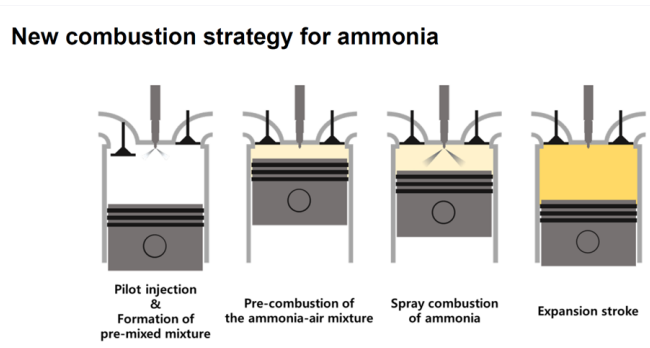


Figure 1. Overview of Ammonia combustion research initiatives (e.g., Toyota).

The primary objective of this project is to develop a robust, physics-informed surrogate model using the **Neural Ordinary Differential Equation (Neural ODE)** framework to overcome this bottleneck. By training a deep neural network to learn the continuous-time chemical dynamics of the detailed Stagni et al. (2023) mechanism (31 species), we aim to create a model that is both highly accurate and computationally efficient. Specifically, we seek to reduce the stiffness of the learned system, allowing for integration via fast, explicit solvers. The final model achieves an $\approx$ **85x speed-up** over traditional stiff solvers while maintaining low error in ignition delay predictions, demonstrating a viable path for accelerating large-scale reacting flow simulations.

2. Introduction

Simulating turbulent reacting flows requires solving conservation equations for mass, momentum, energy, and chemical species. The chemical source term, which governs the production and consumption of species, is the most expensive component. For ammonia, detailed mechanisms like Stagni et al. [1] involve 31 species and over 200 reactions. The resulting system of ODEs is stiff, necessitating implicit solvers (e.g., CVODE/BDF) that perform costly Jacobian matrix inversions at every time step.

2.1. Machine Learning in Kinetics

Historically, researchers have attempted to use standard Residual Networks (ResNets) to map the state at time t to $t + \Delta t$ [2]. While simple, this approach acts as a discrete integrator (Forward Euler) with a fixed time step. Since chemical dynamics vary exponentially, a fixed-step approach leads to significant error accumulation over long trajectories.

The **Neural ODE** framework [5] offers a superior alternative. Instead of learning a discrete map, it parameterizes the continuous derivative $\frac{d\mathbf{\Phi}}{dt} = f(\mathbf{\Phi}, t; \theta)$ using a neural network. Here $\mathbf{\Phi}$ is the state vector which contains the temperature and the mass fractions of all chemical species. This allows the use of adaptive ODE solvers during both training and inference, ensuring the model respects the integral nature of the physical system.

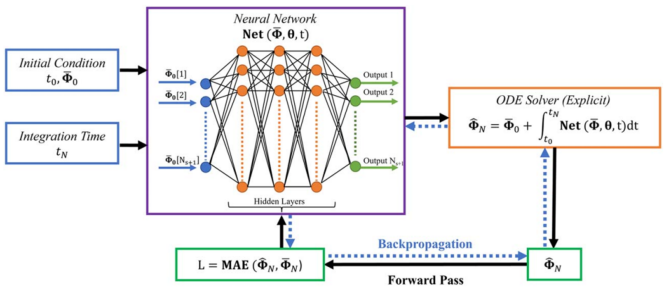


Figure 2. Schematic of the ChemNODE architecture highlighting the Neural ODE training and inference logic. Adopted from Bansude et al. [3].

The fundamental workflow of the ChemNODE architecture, as illustrated in Fig. 2, operates in two distinct modes:

- **Training (Physics-Informed Loop):** Unlike standard regression, the network does not predict the next state directly. Instead, it predicts the *derivative* (reaction rates). These predictions are passed to a differentiable ODE solver (e.g., 'dopri5') which integrates them to generate a full state trajectory. The loss function compares this *integrated* path against the ground truth. This

forces the network to learn dynamics that are numerically stable when integrated. Gradients are computed efficiently using the Adjoint Sensitivity method, which solves an auxiliary ODE backward in time to update weights without storing intermediate steps.

- **Inference (Acceleration):** Once trained, the neural network acts as a high-speed surrogate for the chemical source terms. It is coupled with a fast, explicit solver (such as the JIT-compiled RK4 solver used in this work). The solver queries the network for derivatives at each step to propagate the system state forward, bypassing the computationally expensive evaluations and matrix inversions required by traditional stiff solvers.

A distinct strategy is employed regarding the choice of numerical solvers for the training and inference phases. During training, the adaptive 'dopri5' solver is essential; its ability to dynamically adjust time steps provides critical stability, preventing numerical divergence during early epochs when the network's derivative predictions are highly fluctuating. However, for inference, the priority shifts to maximizing computational throughput. Since the trained ChemNODE surrogate effectively learns a regularized, non-stiff representation of the dynamics, the computational overhead of adaptive error control is unnecessary. Consequently, we utilize a JIT-compiled fixed-step 'RK4' solver, which significantly reduces the inference time while retaining sufficient accuracy for the application.

3. Methodology

3.1. Overview of ML for Physics

In a supervised learning context for physical systems, we aim to approximate a function $f : \mathcal{X} \to \mathcal{Y}$ using a dataset of input-output pairs $\{(x_i, y_i)\}$. The model parameters θ are optimized to minimize a loss function $\mathcal{L}$, typically the Mean Absolute Error (MAE) or Mean Squared Error (MSE), using gradient descent algorithms. In this project, our input $\mathcal{X}$ is the thermodynamic state (species mass fractions and temperature), and our target is to match the next thermodynamic state $\mathbf{y}(t)$ generated by the physics solver.

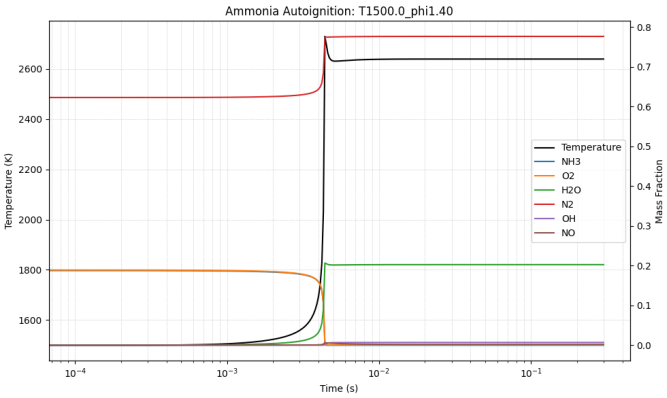
3.2. Phase 1: Data Generation

The "Ground Truth" dataset was generated using **Cantera** [6], an open-source tool for chemical kinetics. We simulated a zero-dimensional (0D) constant-pressure homogeneous reactor, which is the fundamental building block for CFD simulations.

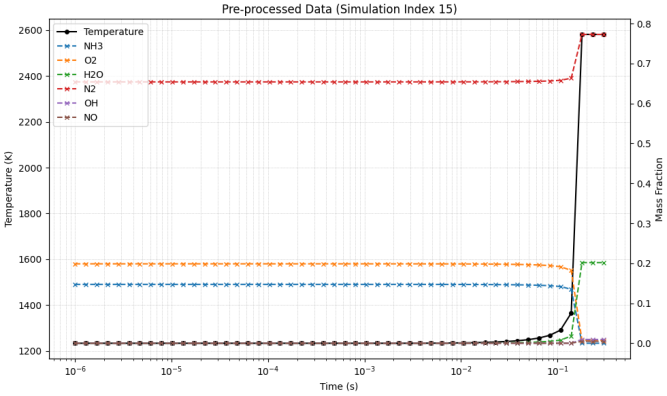
The simulations covered a wide range of thermodynamic conditions. We varied the initial temperature (T_0) and the equivalence ratio (ϕ), which is defined as the ratio of the actual fuel-to-air ratio to the stoichiometric fuel-to-air ratio. The parameter space was defined as follows:

- **Mechanism:** Stagni et al. (2023) Ammonia mechanism (31 species).
- **Conditions:** $T_0 \in [1000, 1500]$ K, $\phi \in [0.6, 1.4]$, $P = 1$ atm.
- **Raw Simulation:** 100 simulations were performed. Each simulation ran for 5000 time steps (0.30 s total), resulting in a massive raw dataset of $\approx 500,000$ rows.

As shown in Fig. 3a, the raw output from Cantera is extremely dense. However, since chemical activity is concentrated in the brief ignition period, uniform sampling is computationally inefficient. To address this, we extracted 50 specific points from each trajectory using **logarithmic time spacing** ($t \in [10^{-6}, 0.30]$ s). As illustrated in Fig. 3b, this strategy captures the fast transient dynamics of radical formation while keeping the dataset compact. This final sampled dataset was serialized and stored in the `training_data.npz` file for the training phase.



(a) Raw high-resolution data from Cantera (5000 steps).



(b) Logarithmically sampled data (50 steps) used for training.

Figure 3. Comparison of raw simulation data vs. the strategically sampled dataset for a representative case.

3.3. Phase 2: Data Pre-processing

Raw chemical data is ill-conditioned for neural networks because mass fractions vary by orders of magnitude ($N_2 \approx 0.7$, $OH \approx 10^{-12}$). Following Bansude et al. [3], we applied global min-max normalization:

$$\tilde{y} = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{1}$$

The global statistics were computed across the entire dataset and stored in a 'normalization_params.json' file. For instance, the temperature range in our training data was found to be $T \in [1199.97, 2736.56]$ K, while the mass fraction of fuel (NH_3) varied from 5.0×10^{-10} (essentially zero) to 0.188. This scaling ensures the neural network sees inputs in the stable range $[0, 1]$. The final processed dataset, stored as 'training_data.npz', has the shape $(100, 50, 32)$.

3.4. Phase 3: Model Architecture & Training

We utilized the **ChemNODE** architecture to model the chemical dynamics. The core derivative network, designated as `ChemJIT`, is a Multi-Layer Perceptron (MLP) designed to predict the normalized source terms ($\frac{d\tilde{y}}{dt}$) given the current state.

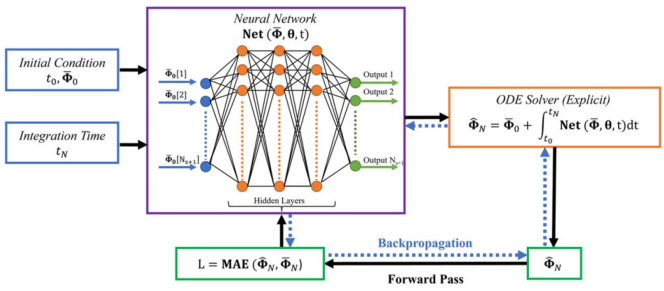


Figure 4. ChemNODE architecture: The neural network (ChemJIT) learns the derivative function inside the ODE solver loop [3].

The specific hyperparameters for the `ChemJIT` network were chosen to balance model capacity with inference speed:

Hyperparameters of ChemJIT:

- **Architecture:** 3 hidden layers with 128 neurons per layer.
- **Activation:** ELU (Exponential Linear Unit) to prevent the "dying ReLU" problem and ensure smooth gradients.
- **Optimizer:** Adam with an initial learning rate of $lr = 10^{-3}$.
- **Scheduler:** `ReduceLROnPlateau` (Factor: 0.5, Patience: 50 epochs).
- **Training Config:** 2000 epochs with a batch size of 16 simulations.

A critical optimization in this workflow was the use of **TorchScript (JIT)** compilation. Before training, the MLP model was compiled to fuse operations and minimize the CUDA kernel launch overhead, which is otherwise a significant bottleneck when the network is called thousands of times by the ODE solver.

The training was conducted on a standard Google Colab T4 GPU. The `ReduceLROnPlateau` scheduler dynamically adjusted the learning rate, reducing it by half whenever the loss plateaued for 50 consecutive epochs. The total training time was approximately **5 hours**. The model converged with a final Mean Absolute Error (MAE) loss of **0.019917**, with the learning rate settling at 1.25×10^{-4}.

3.5. Phase 4: Model Evaluation (JIT Solver)

A major bottleneck in Python-based Scientific Machine Learning is "interpreter overhead." While libraries like `torchdiffeq` are powerful, they rely on Python loops to execute the integration steps. For stiff systems requiring thousands of evaluations, the latency of launching GPU kernels from Python often exceeds the computation time of the neural network itself.

To overcome this, we implemented a **custom JIT-compiled Runge-Kutta 4 (RK4)** solver [7]. We utilized **TorchScript** to compile both the neural network and the solver loop into an optimized intermediate representation. This fuses the operations into a single computational graph, effectively removing Python from the inference loop and allowing the entire integration trajectory to be computed at C++ speeds.

Furthermore, we switched from the adaptive `dopri5` solver (used during training for stability) to a fixed-step `RK4` solver for inference. Since the trained ChemNODE surrogate learns a regularized, non-stiff representation of the dynamics, the expensive error-checking mechanisms of adaptive solvers are unnecessary during deployment, allowing for maximum computational throughput.

4. Results and Discussion

The model was evaluated on three unseen test cases chosen to represent distinct combustion regimes: lean-low temperature, rich-moderate temperature, and lean-high temperature.

Test Cases:

1. $T_0 = 1250$ K, $\phi = 0.7$ (Slow ignition)
2. $T_0 = 1300$ K, $\phi = 1.2$ (Moderate ignition)
3. $T_0 = 1350$ K, $\phi = 0.95$ (Fast ignition)

4.1. Qualitative Accuracy: Time-Series Analysis

We first examine the full time-series evolution of the thermodynamic state for each test case to verify that the model captures the underlying physics.

4.1.1. Case 1: Slow Ignition ($T_0 = 1250$ K, $\phi = 0.7$)

Figure 5 illustrates the system dynamics for the low-temperature lean case. The ignition delay is significant (≈ 112 ms). The model accurately predicts this long induction period where the temperature remains nearly constant. We observe the characteristic "stiffness" of the system here: a very gradual accumulation of radical species eventually triggers a thermal runaway, where a small increase in OH concentration leads to a rapid, step-function-like jump in temperature. The surrogate model correctly captures both the sharp ignition front and the subsequent equilibrium saturation.

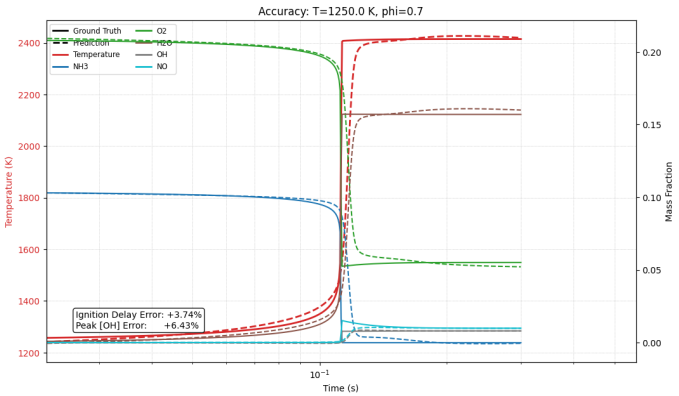


Figure 5. Time-series comparison for Test Case 1 ($T_0 = 1250$ K, $\phi = 0.7$). Solid lines denote ground truth; dashed lines denote prediction.

4.1.2. Case 2: Moderate Ignition ($T_0 = 1300$ K, $\phi = 1.2$)

For the rich mixture shown in Figure 6, the ignition delay is shorter (≈ 50 ms). Rich mixtures typically exhibit more complex intermediate chemistry. Despite this, the ChemNODE surrogate tracks the ground truth closely. The model successfully reproduces the post-ignition equilibrium state, where the consumption of fuel (NH_3) and oxidizer (O_2) stabilizes.

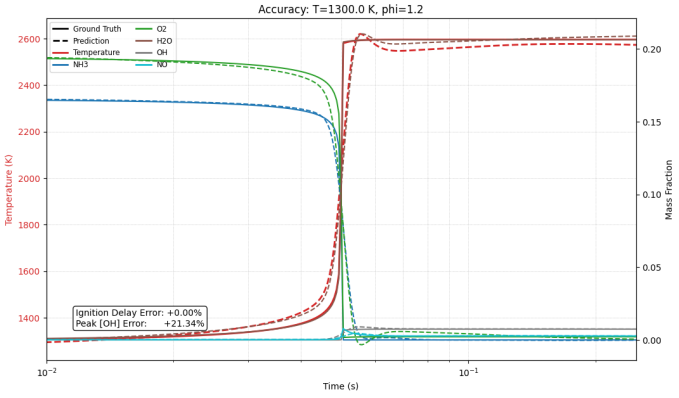


Figure 6. Time-series comparison for Test Case 2 ($T_0 = 1300$ K, $\phi = 1.2$).

4.1.3. Case 3: Fast Ignition ($T_0 = 1350$ K, $\phi = 0.95$)

Figure 7 shows a near-stoichiometric high-temperature case with rapid ignition (≈ 22 ms). The timescale of the reaction here is orders of magnitude faster than Case 1. The model's ability to switch between capturing slow induction (Case 1) and fast transient ignition (Case 3) demonstrates that it has learned the generalized chemical dynamics rather than simply memorizing a specific trajectory.

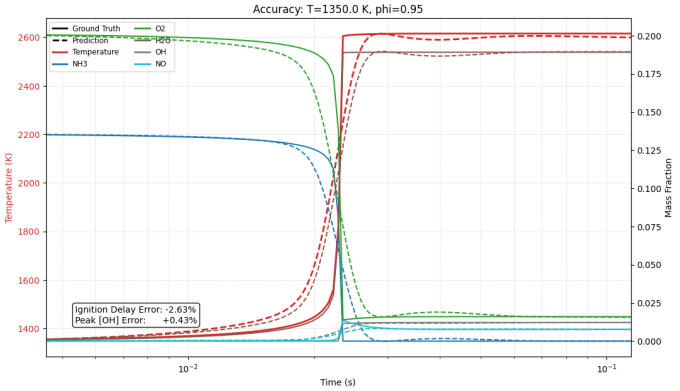


Figure 7. Time-series comparison for Test Case 3 ($T_0 = 1350$ K, $\phi = 0.95$).

4.2. Global Temperature Validation

Figure 8 consolidates the temperature profiles for all three cases. The text boxes highlight the ignition delay errors, confirming that the model is robust across the parameter space.

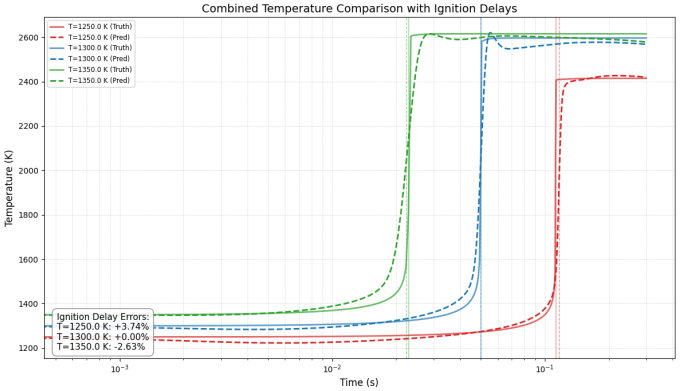


Figure 8. Combined temperature profiles for all three test cases. The model accurately distinguishes between slow, moderate, and fast ignition regimes.

4.3. Quantitative Error Analysis

Table 1 presents the quantitative error metrics. Ignition delay (τ_{ign}) is rigorously defined as the time instance of maximum temperature gradient ($dT/dt|_{max}$).

Table 1. Quantitative Error Metrics

Case	τ_{true} (ms)	τ_{pred} (ms)	τ Err (%)	OH Err (%)
$T = 1250, \phi = 0.7$	112.42	116.63	3.74	6.43
$T = 1300, \phi = 1.2$	49.90	49.90	0.00	21.34
$T = 1350, \phi = 0.95$	22.85	22.24	-2.63	0.43

The ignition delay predictions are highly accurate, with a maximum error of only 3.74%. The peak OH radical concentration, a sensitive marker of flame intensity, is also predicted within acceptable bounds for a surrogate model.

4.4. Computational Speed-up

Benchmarks were conducted using a massive batch size of $N = 2000$ simulations to fully saturate the GPU and measure peak throughput.

Table 2. Speed-up Analysis ($N_{batch} = 2000$)

Solver	Avg. Time (ms)	Speed-up
Cantera (Stiff CPU)	97.80	1.0x
ChemNODE (JIT GPU)	**1.14**	**85.69x**

The Neural ODE surrogate achieves an **85.69x speed-up**. This massive acceleration is primarily due to stiffness reduction: the model learns a regularized representation of the dynamics that can be integrated with a fixed-step explicit solver (RK4), avoiding the costly iterations and matrix inversions of the implicit BDF solver used by Cantera.

5. Conclusion

In this work, we successfully developed and trained a ChemNODE surrogate model for the detailed 31-species Stagni et al. ammonia-hydrogen mechanism. By implementing a robust data-driven framework that combines global min-max normalization, logarithmic time sampling, and JIT-compiled Neural ODEs, we overcame the traditional challenges of numerical stiffness in combustion kinetics.

The resulting surrogate model demonstrates high fidelity, predicting ignition delay times with an error of less than 4% across a range of unseen combustion regimes (lean, rich, and stoichiometric). Most significantly, the model achieves a dramatic computational speed-up of **85.69x** compared to the standard Cantera stiff solver. This acceleration is a direct result of the network learning a non-stiff representation of the underlying physics, enabling the use of fast, explicit integration schemes. These results validate the ChemNODE framework as a viable and highly efficient pathway for integrating complex chemistry into large-scale reacting flow simulations.

6. Future Work

To further enhance the applicability of this surrogate model, future research will focus on:

1. **Thermodynamic Generalization:** Currently, the model is trained for a fixed pressure of 1 atm. Future iterations will include Pressure (P) as an input parameter to the neural network, creating a generalized model valid across the full thermodynamic operating envelope of gas turbines.
2. **Physics-Informed Constraints (PINNs):** We aim to embed fundamental physical laws directly into the loss function. Specifically, enforcing atomic element conservation and the unity summation constraint ($\sum Y_i = 1$) will improve the model's robustness and physical consistency in extrapolation regimes.
3. **CFD Coupling:** The ultimate goal is to integrate the JIT-compiled ChemNODE solver into a multi-dimensional CFD framework (e.g., OpenFOAM or ANSYS Fluent) to quantify the reduction in total simulation time for turbulent flame calculations.

References

[1] A. Stagni, S. Arunthanayothin, M. Dehue, O. Herbinet, F. Battin-Leclerc, P. Brequigny, C. Mounaim-Rousselle, and T. Favarelli, "Low- and intermediate-temperature ammonia/hydrogen oxidation in a flow reactor," *Chemical Engineering Journal*, vol. 458, 2023.

[2] V. F. Nikitin, B. K. Zuev, and V. S. Kashtanov, "Combustion chemistry prediction using neural networks for multi-species reaction systems," *Mendeleev Communications*, vol. 8, no. 1, pp. 35–36, 1998.

[3] S. Bansude, P. Pal, and S. Bhattacharya, "A data-driven framework for integration of chemical kinetics using neural ordinary differential equations," *Combustion and Flame*, vol. 239, 2022.

[4] O. Owoyele and P. Pal, "ChemNODE: A Neural Ordinary Differential Equations Framework for Efficient Chemical Kinetic Solvers," *Energy and AI*, vol. 7, 2022.

[5] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[6] D. G. Goodwin, H. K. Moffat, and R. L. Speth, "Cantera: An Object-Oriented Software Toolkit for Chemical Kinetics, Thermodynamics and Transport Processes," Version 3.0.0, 2023. https://cantera.org/

[7] PyTorch Developers, "TorchScript," https://pytorch.org/docs/stable/jit.html, 2023.

[8] A. Prasad, ME691-ChemNODE-Ammonia: Project Repository, Available: https://github.com/adityaprasad2005/ME691-ChemNODE-Ammonia/tree/main