Adaptive Deep Learning with Batch Feature Re-Engineering and Differential Dynamical Systems

Ruixin Chen
Katz School of Science and Health
Yeshiva University
New York, NY
rchen4@mail.yu.edu

David Li
Katz School of Science and Health
Yeshiva University
New York, NY
david.li@yu.edu

***Abstract*—This paper presents a novel deep learning framework that integrates class information directly into batch feature engineering to enhance model interpretability and adaptability. Unlike conventional machine learning models, this approach introduces an innovative training paradigm and evaluation methodology, enabling the model to adapt seamlessly to new data distributions without requiring re-training. By coupling the power of differential equation dynamics simulation, the proposed framework bridges the gap between data-driven learning and mechanistic modeling, offering robust predictive capabilities. A case study on infectious disease modeling demonstrates the effectiveness of this approach, showcasing superior performance compared to traditional methods in handling evolving data distributions and capturing complex system dynamics. This work lays the foundation for adaptive modeling in dynamic, real-world scenarios.**

***Index Terms*—Batch Feature Re-Engineering, Class Information, Differential Equations**

I. Introduction

In recent years, deep learning has revolutionized predictive modeling by leveraging vast amounts of data and intricate neural network architectures. However, traditional machine learning models often struggle to adapt to dynamically changing data distributions without significant retraining efforts. This limitation poses a significant challenge in fields such as epidemiology, where data evolve continuously due to factors like population behavior changes, new pathogen characteristics, or policy interventions. In such contexts, integrating mechanistic insights from physics-based models with the adaptability of deep learning can offer transformative solutions.

Differential equations have long been a cornerstone in modeling dynamic systems, providing a mathematical framework to describe the underlying processes driving system behavior. When combined with data-driven methods, they have the potential to enhance the interpretability and robustness of predictions. Nevertheless, existing attempts at combining these paradigms often treat them as separate components, failing to exploit their full synergistic potential.

This paper introduces a novel framework by combining adaptive deep learning and differential equations, that addresses these challenges. The key innovation lies in the integration of posterior probability into the feature engineering process, which enhances the model's ability to capture the evolving statistical characteristics of the data. Furthermore, the proposed framework incorporates a new training paradigm and evaluation methodology, allowing the model to adapt to new data distributions without requiring retraining. By coupling this approach with differential equation dynamics simulation, the model effectively bridges the gap between data-driven learning and mechanistic modeling, enabling robust and adaptive predictions.

To demonstrate the efficacy of the proposed framework, a case study on infectious disease modeling is presented. The results show that this approach outperforms traditional methods in predictive accuracy, adaptability, and the ability to handle evolving data distributions. This paper aims to provide a foundation for future research in integrating deep learning with mechanistic models, paving the way for innovative solutions in complex, real-world scenarios.

The remainder of this paper is organized as follows: Section II reviews related work and highlights the limitations of existing approaches. Section III details the proposed framework, including its mathematical formulation and implementation. Section IV presents the case study and experimental results. Finally, Section V concludes the paper and discusses potential directions for future research.

II. Related Work and Limitations of Existing Approaches

The integration of machine learning (ML) and mechanistic modeling has gained increasing attention across various scientific disciplines, particularly in applications requiring predictive insights into dynamic systems. Traditional approaches to predictive modeling can be broadly categorized into three domains: purely data-driven methods, mechanistic modeling using differential equations, and hybrid models that attempt to combine the two. While each has demonstrated success in specific contexts, limitations persist that hinder their applicability to rapidly changing real-world scenarios.

A. Purely Data-Driven Methods

Data-driven models, particularly deep learning frameworks, have demonstrated remarkable success in fields such as image recognition, natural language processing, and disease forecasting. These methods excel in capturing complex patterns and

non-linear relationships from large datasets. However, their reliance on static training data makes them inherently limited when applied to dynamically evolving systems. Retraining data-driven models for new distributions or incorporating domain knowledge remains computationally expensive and often impractical in real-time scenarios. Furthermore, these models are typically opaque, offering little insight into the underlying processes driving their predictions, which is a critical requirement in scientific and policy-making domains.

B. Mechanistic Modeling Using Differential Equations

Mechanistic models based on differential equations have been a mainstay for modeling dynamic systems, including fluid dynamics, infectious disease spread, and chemical reactions. These models benefit from interpretability and the ability to describe system behaviors using physical laws or biological principles. However, they often struggle to accommodate the stochastic nature of real-world data and require extensive domain expertise to calibrate. Moreover, differential equation models are sensitive to parameter estimation, and their predictive accuracy declines when faced with highly noisy or incomplete data.

C. Hybrid Models

Recent advances have focused on combining machine learning with mechanistic modeling to address their individual limitations. For example, neural ordinary differential equations (Neural ODEs) [1] embed neural networks within the framework of differential equations, allowing for data-driven parameterization of dynamic systems. Similarly, physics-informed neural networks (PINNs) [2] leverage known physical laws as constraints to enhance the learning process. While these approaches have shown promise, they typically involve complex architectures and are computationally intensive. Additionally, many hybrid models are tailored to specific applications, making them difficult to generalize across domains or adapt to new data distributions without extensive reconfiguration.

D. Limitations of Existing Approaches

Despite the progress made, several limitations remain:

- Lack of Adaptability: Most existing models require retraining or manual reconfiguration when the underlying data distribution changes. This limitation is especially problematic in domains where data evolve rapidly, such as infectious disease modeling during outbreaks.
- Inefficient Integration of Probabilistic Insights: Few approaches directly incorporate probabilistic information, such as posterior distributions, into the feature engineering or learning process. This oversight reduces the models' ability to handle uncertainty and variability in real-world data.
- High Computational Cost: The coupling of deep learning and differential equations often results in computationally demanding frameworks that are difficult to scale or deploy in real-time applications.
- Limited Generalizability: Many hybrid models are tailored for specific scenarios and struggle to generalize to new problems or domains without significant customization.

The proposed framework addresses these challenges by seamlessly integrating class information into batch feature engineering, offering a new paradigm for training and evaluation. By combining these innovations with differential equation dynamics simulation, the model achieves adaptability to new data distributions without retraining. This work represents a significant step forward in the development of robust, scalable, and interpretable predictive models for complex dynamic systems.

III. The Proposed Framework

Training and evaluating machine learning models typically involves multiple steps. Initially, the data is divided into training and test sets. The training set is used to train the model, while the test set is subsequently utilized to evaluate the model's performance. This approach assumes that the training set and test set share the same distribution.

However, once the model is deployed in production, the distribution of new, real-world data may shift, necessitating model retraining. In many real-world applications, the patterns and relationships present in the training data may not remain static over time. For example, in applications like financial forecasting, e-commerce recommendation systems, or infectious disease modeling, the factors influencing the system can evolve due to new market trends, changes in user behavior, or the emergence of novel pathogens. Retraining the model ensures it can adapt to these changes and maintain high predictive accuracy.

Retraining is often considered bad practice in industry due to its high computational cost, inefficiency, and potential for model instability. It can disrupt real-time applications, complicate version control, and require costly labeled data. Frequent retraining also risks overfitting to recent trends, making the model less generalizable. Adaptive or robust models that self-update without full retraining are typically more practical and scalable in dynamic industrial environments.

A. The Traditional Deep Neural Network and Batch Training

A Deep Neural Network (DNN) is a computational model composed of multiple layers of interconnected neurons, capable of learning complex patterns and representations from data. Batch training is a critical process in training DNNs, where the dataset is divided into smaller subsets, called batches, to optimize the model's parameters iteratively. This approach balances computational efficiency and convergence stability, as it reduces memory usage compared to processing the entire dataset at once (full-batch training) and minimizes the high variance observed in single-sample updates (stochastic gradient descent) [3]. During batch training, gradients of the loss function are computed for each batch and used to update the weights [4], often leveraging techniques like momentum and learning rate schedules to enhance convergence [5]. This

strategy is integral to the scalability and performance of DNNs, especially when dealing with large datasets.

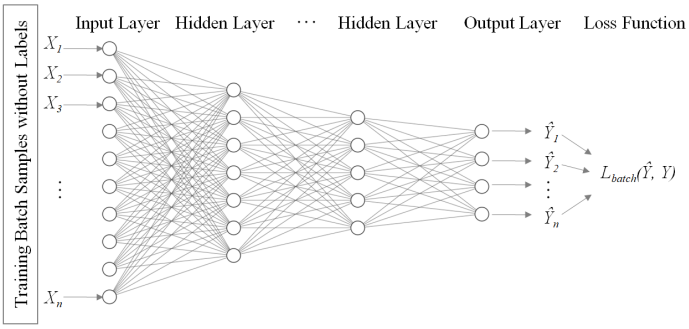

Fig. 1: The Traditional Deep Neural Network and Batch Training

B. Class Information in Features, Good or Bad?

Class information refers to data or features that capture the distribution of the classes, such as:

- Class probabilities or frequencies.
- Conditional probabilities, like $P(y|x)$, where y is the class label and x is the feature set.
- Context-dependent adjustments based on the dataset or task.

When training a deep neural network (DNN) using batches, the input features should not directly contain class information because:

- Prevention of Label Leakage. Class information in the input features provides the network with direct access to the labels it is supposed to predict. Consequently, the model achieves artificially high accuracy during training but fails to generalize to unseen data, leading to poor performance during testing.
- Model Bias. Class information as a feature can artificially inflate the apparent accuracy of the model during training because the input explicitly carries clues about the target, making evaluation unreliable.
- Batch Updates and Gradient Descent. During batch training, the model uses gradients calculated from predictions and actual labels. If the input features already encode class information, the gradients might primarily reflect the "shortcuts" in the data instead of learning true patterns.

However, in many real-world scenarios, class information from new data can be highly beneficial for adapting a trained model based on historical data, leading to more accurate predictions for newly encountered instances. Often, this class information is readily available and can be leveraged effectively to enhance predictive performance.

For instance, in infectious disease modeling, predictive models deployed into production frequently experience performance degradation due to shifting data distributions caused by the dynamic and mechanistic nature of diseases. The class information of new patient cases often differs significantly from that of the historical training data. However, this new information is typically accessible, either through statistical analyses of recent patient data—reflecting the class distribution of the immediate past—or through simulations of differential dynamical systems, which estimate the class distribution for the near future, corresponding to the prediction target.

Similar challenges arise in other fields, such as financial forecasting and weather forecasting, where shifting data distributions necessitate the incorporation of new class information for improved prediction accuracy.

The traditional approach involves retraining the model using recent data. However, this method fails to fully capture emerging trends in the near future, leading to recurring performance issues as the data distribution continues to shift. The proposed approach addresses this by embedding class information directly into the feature set of the batch samples, enabling the model to utilize this information effectively for improved predictions.

C. Batch Feature Re-Engineering with Batch Class Information

Embedding batch class information dynamically into a model training for batch data involves incorporating batch class-specific probabilities, weights, or contextual information in each batch that the model can adaptively use during training. This approach ensures that the model has a built-in awareness of class information without explicitly adjusting weights or retraining.

In static methods (like using class weights or oversampling), the adjustments are predefined and do not adapt during training. Dynamic embedding allows the model to learn the influence of batch class-specific information in a feature-specific or context-specific way, and adjust the importance of classes adaptively as training progresses. Assume that we have

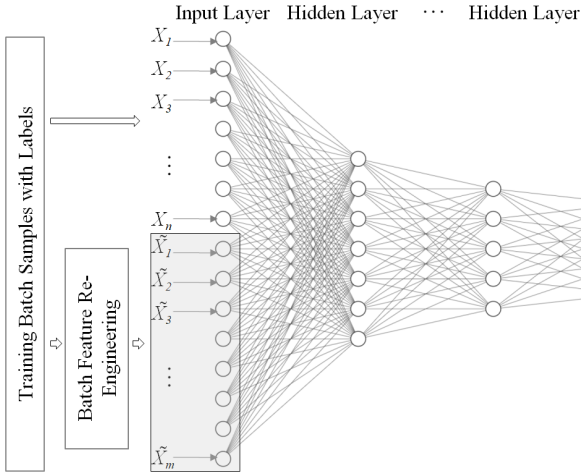

Fig. 2: Batch Feature Re-Engineering with Class Information of the Batch

the feature set,

$$X = \{X_1, X_2, \ldots, X_n\} \tag{1}$$

where n is the original feature set size.

The new feature set embeded with class information is,

$$\tilde{X} = \{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_m\} \tag{2}$$

where m is the new feature set size.

Some of the techniques to dynamically embed batch class information into the new batch features are,

- Class Likelihoods. One simple way to embed batch class information is to add features that represent the classes distribution or likelihood for each sample within this batch.

$$\tilde{X}_P = \left\{ P_{batch}(y=i) = \frac{N_i}{N} = \frac{N_i}{\sum_{j=1}^{c} N_j} \right\} \tag{3}$$

  where N is the batch size, N_i is the count of the i-th class in this batch, c is the total number of classes.
- Class Likelihood Ratios. Another way is to create new features by class likelihood ratios.

$$\tilde{X}_R = \text{Likelihood Ratios} = \left\{ \frac{P_{batch}(y=i)}{P_{batch}(y=j)} \right\} \tag{4}$$

  where $i, j \in [1, c], i \neq j$.
- Combine Class Likelihoods and Ratios with Original Features. For example, if $X = \{x_1, x_2, \ldots, x_n\}$ is a feature vector, then we can add new features as

$$\begin{aligned} \tilde{X}_F &= \tilde{X}^{(i)}_{P \text{ or } R} \times X \\ &= \{\tilde{X}^{(i)}_{P \text{ or } R} \times x_1, \tilde{X}^{(i)}_{P \text{ or } R} \times x_2, \ldots, \tilde{X}^{(i)}_{P \text{ or } R} \times x_n\} \end{aligned} \tag{5}$$

  where $\tilde{X}^{(i)}_{P \text{ or } R}$ can be any one feature of $\tilde{X}_P$ or $\tilde{X}_R$.

D. Training Strategy with Batch Feature Re-Engineering

The proposed batch feature re-engineering method embeds class information from each batch into the corresponding features, enabling the model to leverage this information for more accurate predictions. However, if traditional random shuffling and sampling techniques are used to create batches, the class information within each batch will tend to mirror the overall class distribution of the entire training set. This limits the diversity of class information the model encounters during training. Consequently, the model learns to handle only the most common class patterns in the training set while gaining little exposure to less frequent class scenarios.

To address this limitation, this study introduces a novel training strategy that ensures that the model is fully trained to handle a wide range of possible class information scenarios.

Instead of randomly sampling the training set into batch, one can randomly generate the class likelihoods of $\tilde{X}_p$.

$$\tilde{X}_p = \{p_1, p_2, \ldots, p_c\} \tag{6}$$

where $0 \leq p_i \leq 1$ and $\sum_{i=1}^{c} p_i = 1$.

This random class likelihoods can be generated by the softmax function from a list of random values $\{z_1, z_2, \ldots, z_c\}$.

$$p_i = \sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{c} e^{z_j}} \tag{7}$$

With the pre-defined batch size N and the random class likelihoods of (6), one can sample the training set to generate a batch.

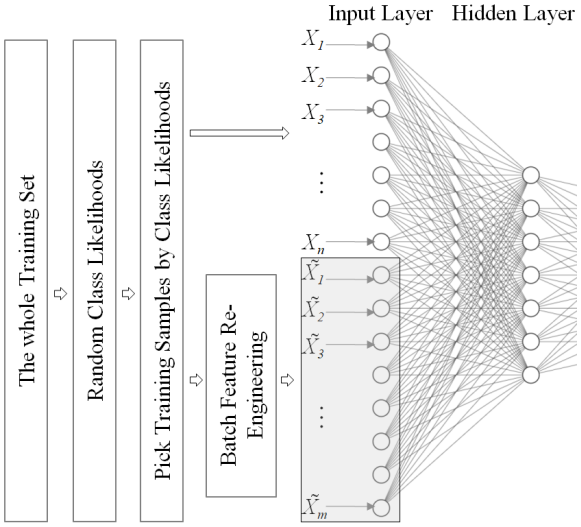

Fig. 3: Training Strategy with Batch Feature Re-Engineering

It is important to note that each batch is assigned its own randomized class likelihoods, enabling the model to learn and utilize this information to enhance prediction accuracy.

E. Evaluation Strategy with Batch Feature Re-Engineering

Unlike traditional model evaluation, the goal here is to assess the model's accuracy under the assumption that class information for the testing samples is already available.

This approach mirrors real-world scenarios, such as infectious disease prediction, where the most recent class information is accessible. By incorporating this information, it is expected that the pre-trained model will achieve improved prediction accuracy.

To evaluate the trained model, the actual testing set is first analyzed to determine the class likelihoods. These likelihoods are then used in conjunction with the batch feature re-engineering method to generate new features for the testing data, enabling a more accurate evaluation of the model. The workflow is shown in Fig. 4.

F. Prediction with Class Information from Differential Dynamical Systems

Now the question is, when we use the trained model to predict, how to get the class information of the unseen new data?

There are several options. Meta-learning techniques incorporates class-specific information dynamically into the prediction process. Instead of treating class information as static or predefined, meta-learning enables the model to adaptively learn how class characteristics interact with input features. By embedding class-specific knowledge as part of the learning process, the model can generalize better across various scenarios, particularly when class distributions shift over time. This approach is especially useful in applications like personalized

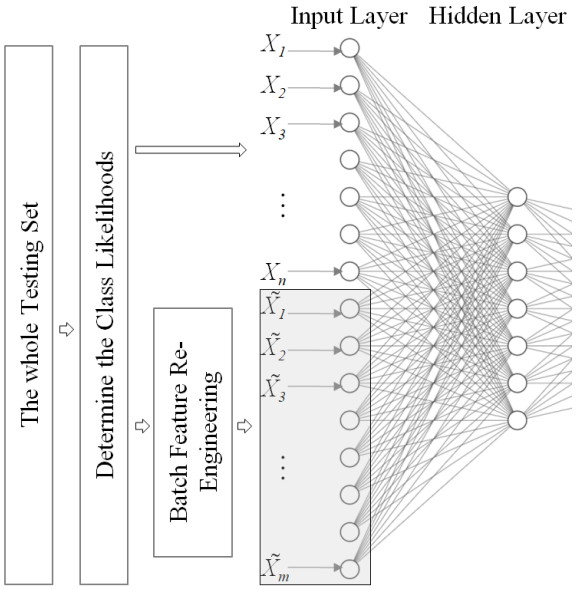

Fig. 4: Evaluation Strategy with Batch Feature Re-Engineering

medicine, financial forecasting, and dynamic systems modeling, where class information is subject to frequent changes and must be continuously integrated to improve predictive performance.

Alternatively, in many dynamical systems such as infectious disease, one can estimate the class information directly from the dynamics by the differential equations [6]. There are several variations of compartmental models, where SEIRD is one of them, which are used in modelling of infectious disease using differential equations. These types of models divide the population into groups or compartments and the dynamics of these groups are expressed with the help of a system of differential equations.

The SEIRD model is shown as below,

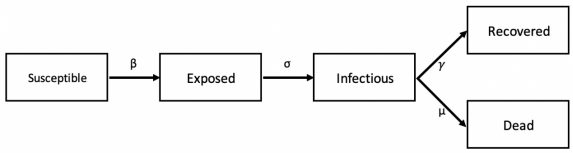

Fig. 5: SEIRD Model

The dynamics is represented by the differential equations,

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta SI}{N} \\ \frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E \\ \frac{dI}{dt} = \sigma E - \gamma I - \mu I \\ \frac{dR}{dt} = \gamma I \\ \frac{dD}{dt} = \mu I \\ N = S + E + I + R + D \end{cases} \tag{8}$$

where,

- These variables (S, E, I, R and D) represent the number of people in each compartment at a particular time.
- β is infection rate or the rate of spread
- σ is the incubation rate or the rate of latent individuals becoming infectious (average duration of incubation is $1/\sigma$)
- γ is the recovery rate or mortality rate. If the duration of indection is D then γ = 1/D
- μ is the mortality rate due to the disease

The four parameters can be estimated by minimizing the error between the numerical solution of the differential equations and the real statistical data.

$$\{\beta, \sigma, \gamma, \mu\} = \arg\min_{\beta,\sigma,\gamma,\mu} MSE(SEIRD, data) \tag{9}$$

With the parameter values in the differential equations, one can get the numerical solution of the dynamical system.

For example, given the values

$$\{\beta, \sigma, \gamma, \mu\} = \{0.25, 0.08, 0.02, 0.004\}$$

and the initial conditions with 10000 people with 1 infectious, one can generate the numerical solution as below,

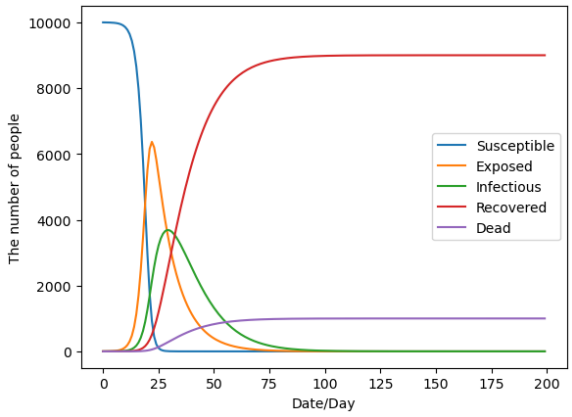

Fig. 6: The Numerical Solution of SEIRD Model

With the numerical solution, one can estimate the class information on any given date in the past and in the future, while the statistical data of the recent patient cases can only estimate the class information for the recent past.

Once we have the class information for the given date, we can use it to generate the new batch features on the new data. The trained model should be able to use the class information to predict the class with more accuracy.

IV. Case Study and Experimental Results

This study uses the dataset of diagnosis of COVID-19 and its clinical spectrum [7]. Based on the results of laboratory tests commonly collected for a suspected COVID-19 case during a visit to the emergency room, the goal is to predict the test result for SARS-Cov-2 (positive/negative).

The comparison of the two confusion matrices highlights the performance differences between the Simple Model (left) and the Adaptive Model (right). The simple model accurately predicts 100 out of 109 samples for Class 0 but misclassifies 9 samples as Class 1. For Class 1, the simple model performs

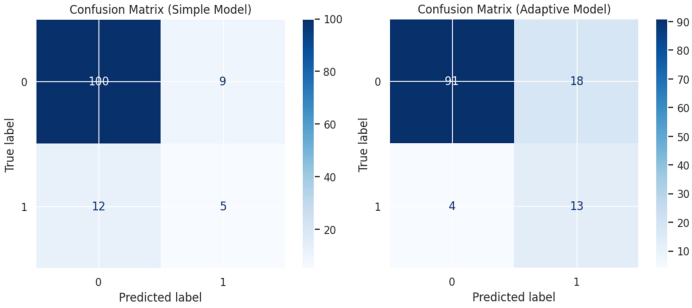

Fig. 7: The Confusion Matrix Comparison

poorly, correctly identifying only 5 samples while misclassifying 12 samples as Class 0. This suggests the simple model is heavily biased toward Class 0, likely due to class imbalance or its inability to adapt to changes in class distributions.

The adaptive model correctly predicts 91 out of 109 samples for Class 0 but misclassifies 18 samples as Class 1. The performance on Class 0 is slightly worse in comparison. However, the performance for Class 1 significantly improves: 13 samples are correctly classified compared to only 5 for the simple model, and 4 samples are misclassified as Class 0. This shows that the adaptive model better balances predictions across both classes, particularly improving recall for the minority class (Class 1).

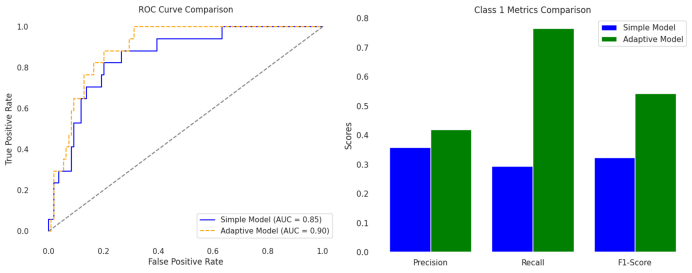

Fig. 8: The ROC/AUC and Metrics Comparison

The ROC curve for the adaptive model consistently outperforms the simple model across various false positive rates. The AUC 0.90 and 0.85 indicating that the adaptive model has better overall discriminatory power between the two classes. This also suggests the adaptive model is better at differentiating between positive and negative classes, especially for shifted data. Precision, recall, and F1-Score of adaptive model are significantly higher than the simple model.

Overall, the improvements indicate that the adaptive model better accommodates class shifts in the data, likely leveraging additional information for better predictions on underrepresented classes, which is crucial for shifted new data distributions.

V. Discussion and Conclusion

In this study, we proposed an innovative deep learning framework that integrates batch feature re-engineering and differential dynamical systems to enhance adaptability and performance. Unlike conventional deep learning techniques that assume static input features throughout the learning process, our approach iteratively transforms and refines the input space in batch-wise operations, adapting to the changing dynamics of the learning task. By embedding the principles of differential dynamical systems into the model architecture, we successfully characterized temporal changes and complex dependencies in the data.

The integration of feature re-engineering as a part of the batch learning process allowed our method to adaptively improve performance on tasks with dynamically evolving patterns. This is particularly effective in cases where data shifts or feature redundancy compromise the performance of standard approaches. Results demonstrated improved learning stability and generalization ability, particularly in dynamic or non-stationary environments.

Moreover, by employing the differential dynamical systems approach, we observed that the model could interpret data as a sequence of states governed by a dynamical trajectory, adding robustness to forecasting and sequential data analysis tasks.

In conclusion, our work bridges the gap between static deep learning frameworks and dynamic real-world scenarios by incorporating batch feature re-engineering and differential dynamical systems. This fusion allows models to self-adapt to changing input distributions while effectively capturing temporal structures in the data. Empirical evaluations underscore the advantage of the proposed method over traditional techniques, proving its potential to enhance performance in dynamic and complex systems.

Future directions include extending this framework to broader applications such as time-series forecasting, control systems, and dynamic graph analysis. Furthermore, integrating methods to reduce the computational cost and exploring self-supervised learning strategies could unlock new opportunities for broader applicability.

References

[1] Chen, Ricky T. Q. & Rubanova, Yulia (2018). Neural Ordinary Differential Equations. Advances in Neural Information Processing Systems.

[2] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Partial Differential Equations. Journal of Computational Physics, 378, 686-707.

[3] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. Proceedings of the International Conference on Learning Representations (ICLR).

[4] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. Nature, 323(6088), 533-536.

[5] Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2018). Don't Decay the Learning Rate, Increase the Batch Size. International Conference on Learning Representations.

[6] Kermack, W. O., & McKendrick, A. G. (1927). A Contribution to the Mathematical Theory of Epidemics. Proceedings of the Royal Society of London.

[7] Einstein Data4U (2020). COVID-19 Dataset: Diagnosis and Clinical Spectrum. Kaggle Dataset. https://www.kaggle.com/datasets/einsteindata4u/covid19