Fourier Neural Operator

Table of Contents

Summary

The Fourier Neural Operator (FNO) is an advanced neural network architecture designed to learn mappings between function spaces, particularly useful in solving partial differential equations (PDEs). By integrating Fourier analysis into neural networks, the FNO leverages the Fourier transform to capture spatial relationships among input features, offering a balance between quasi-linear time complexity and state-of-the-art approximation capabilities. This approach has demonstrated notable success in various scientific and engineering domains, including fluid dynamics, weather forecasting, solar physics, and medical imaging, due to its efficient modeling of complex systems.

The FNO employs the Fourier transform in its layers to enhance the representation of input data, which is initially lifted to a higher-dimensional space. This transformation allows the model to handle high-dimensional PDEs more efficiently than traditional methods, such as convolutional and recurrent neural networks, by focusing on essential frequency components. The inverse Fourier transform is then applied to project the data back to the original dimension, ensuring that the transformed data aligns with the desired output space. This spectral approach mitigates the challenges associated with low-frequency bias and high-frequency noises, which are common in operator learning.

Despite its numerous advantages, including improved convergence, computational efficiency, and robustness to outliers, the FNO faces challenges related to com-

putational and memory requirements, risk of overfitting, and limited interpretability. Addressing these limitations involves exploring techniques like model compression, fine-tuning strategies, and data augmentation. Additionally, ensuring discretization invariance remains a critical aspect for the generalization of FNO models across different spatial resolutions.
The future of FNO research includes optimizing these models for complex systems, extending the F-principle to operator learning, and enhancing their applicability in real-world problems such as engineering, climate science, and medical diagnostics. As the field progresses, it will also be crucial to address potential societal implications and ethical considerations associated with the deployment of FNOs in various applications.

Theoretical Background

The Fourier Neural Operator (FNO) is grounded in the integration of Fourier analysis into neural network frameworks, specifically designed to learn mappings between function spaces, a fundamental concept in operator learning[1]. In this context, operators refer to mappings between function spaces, such as differentiation and integration, which are common in solving partial differential equations (PDEs)[2]. By leveraging the Fourier transform, the FNO aims to capture the spatial relations among input features, offering a quasi-linear time complexity and state-of-the-art approximation capabilities[3].

Fourier Transform and Neural Networks

The Fourier transform is frequently utilized in spectral methods for solving differential equations, as differentiation is equivalent to multiplication in the Fourier domain[3]. This property has also been instrumental in the development of deep learning, evidenced by its application in convolutional neural networks to accelerate computations[3]. Neural network architectures that involve Fourier transforms or sinusoidal activation functions have shown promising results in various studies, including those by Bengio et al. (2007), Mingo et al. (2004), and Sitzmann et al. (2020)[3]. Recent advancements have extended these spectral methods to neural networks, paving the way for the development of the FNO[3].

Fourier Features in Neural Networks

Incorporating Fourier features into neural network inputs has been shown to expedite the convergence of high frequencies in computer vision tasks[4]. This approach has been explored in Physics-Informed Neural Networks (PINNs) to mitigate the impact of low-frequency preferences, as highlighted by Karniadakis et al. (2021)[4]. Unlike the Fourier transform employed in FNO, Fourier features elevate input features to more discernible high-dimensional representations using their Fourier series, enhancing the high-frequency performance of neural networks[4].

Addressing Low-Frequency Bias

One of the challenges in operator learning is the low-frequency bias, which refers to a lack of precision in high-frequency predictions rather than zero magnitudes in these predictions[4]. Spectral analysis has shown that flow fields with minimal high-frequency details can introduce high-frequency noises during training. Techniques like SpecBoost have been developed to mitigate these high-frequency noises by enhancing the low-frequency PDEs[4]. Understanding and addressing these biases are crucial for improving the accuracy and efficiency of neural operators in solving PDEs.

Future Directions

The future directions for Fourier Neural Operators (FNOs) encompass a broad array of research trajectories aimed at enhancing the accuracy, efficiency, and applicability of these models. One of the primary goals in this direction is to improve the prediction accuracy and efficiency in more complex or 3D systems. The current research represents only a humble starting step towards that end[5].

Optimizing for Different PDE Coefficients

An interesting trend observed is how the advantage of FNOs varies with different coefficient values of partial differential equations (PDEs). For example, in the diffusion equation, a larger coefficient indicates a more rapidly changing system. It has been found that for larger dataset sizes, the differences between certain configurations (e.g., C1 and C0) are relatively small. However, the gap becomes more significant as the diffusion coefficient increases, with C1 achieving a 40-50% error reduction compared to C0 in a 5-step rollout[5].

Extending the F-Principle

Another promising area of research involves extending the F-principle, which is traditionally based on learning between data, to operator learning. This extension could potentially improve the performance of FNOs by allowing them to better handle the learning of mappings between function spaces. Future work aims to further develop this theory and design criteria for grid conversion to optimize training times and improve model performance[6].

Real-World Applications and Efficiency

The optimization of FNOs through incremental learning could lead to their application in solving larger real-world problems, thereby advancing fields like engineering, weather forecasting, and climate science. This could result in significant reductions in training times and computational resource requirements, making FNOs a more viable option for practical applications[7].

Handling Rapidly Changing Systems

In rapidly changing systems, the distribution of the system state can become unfamiliar over time, increasing the risk of overfitting and poor performance on unseen validation data. Pretrained models that have already been exposed to a large amount of data can mitigate this risk to some extent, suggesting that future research could focus on improving the robustness of FNOs in such scenarios[5].

Societal Implications

While there might be potential negative societal consequences of this research, they are not immediately apparent. As such, future studies might also need to consider ethical implications and address any societal concerns that arise from the deployment of FNOs in various fields[7].

Structure and Architecture

Fourier Neural Operator (FNO) architectures leverage the Fast Fourier Transform (FFT) to efficiently process and learn from complex data. A typical FNO architecture includes several key components that work in tandem to transform input data and produce the desired output.

Lifting Layer

The input function is initially lifted to a higher-dimensional space using a neural network. This step is crucial for preparing the input data for subsequent Fourier transformations [8][9].

Fourier Layers

Following the lifting layer, a sequence of Fourier layers is applied to the transformed input function. Each Fourier layer is parameterized by a vector of Fourier coefficients, a bias vector, and a weight matrix [8]. The output of each layer is determined by applying an activation function component-wise, with the ReLU function commonly used for this purpose [8].

Inverse Fourier Transform

In the final stage, after the sequence of Fourier layers, an inverse Fourier transform is applied to project the data back to the original dimension. This process ensures that the transformed data aligns with the desired output space [9][10].

Non-linear Activation

The non-linear activation functions employed in FNOs are essential for introducing non-linearity into the mapping from input to output. This non-linear transformation enables the model to learn more complex patterns in the data. Typical activation functions include the ReLU and sigmoid functions [8][10].

Discretization Invariance

A desirable property for FNO architectures is discretization invariance, allowing the model to act on any discretization of the source term and be evaluated at any point in the domain. This property is essential for generalizing the model to unseen data and enhancing its transferability to different spatial resolutions [8].

Limitations and Extensions

One main limitation of FNO architectures is the requirement for FFT to be performed on a uniform grid and rectangular domains. This constraint can be addressed using embedding techniques that transform input functions to a uniform grid, although these methods may introduce additional computational costs and potential loss of accuracy [8]. Recent advancements have proposed various approaches to extend FNOs to more general domains, such as zero padding, linear interpolation, and encoding the geometry into a regular latent space using neural networks [8].

Mathematical Formulation

The Fourier Neural Operator (FNO) leverages the principles of Fourier transforms to perform efficient and accurate approximations of complex systems. In mathematical terms, the FNO applies the concept that "convolution in the time domain is multiplication in the frequency (Fourier) domain" to perform operations on functions in the frequency domain[11].
[
\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)
]
where (\mathcal{F}) denotes the Fourier transform operator[11].
In the FNO framework, we consider the learning of surrogate models in data-driven settings, where the data might not always be available. In such scenarios, physics-informed settings are used by optimizing the physics-informed equation loss. The output function can be explicitly written using predefined formulas, and the derivatives are computed using the chain rule with auto-differentiation[9]. This method ensures that the residual error is minimized to identify the parameter representing the solution function. For instance, the Geo-PINO method is an optimization-based spectral method designed for general geometry[9].
For specific applications, such as fluid dynamics, Fourier neural operators have been employed to create reduced-order models that are less computationally intensive compared to traditional numerical PDE solvers. This is particularly useful in modeling turbulent flows and high-dimensional PDEs, where traditional solvers require fine discretization[8]. For example, in fluid dynamics, Fourier neural operators can accelerate the evaluation of solution operators, making them useful for forecasting and optimization in engineering problems like the design of aircraft and wind turbines[8].
The coefficients in these models are often generated with zero Neumann boundary conditions on the Laplacian, forming prototypical models for physical systems like subsurface flows and material microstructures in elasticity. Solutions are typically obtained using second-order finite difference schemes on a grid, with varying resolutions downsampled from the dataset[3].
Another key aspect of the FNO involves using operator learning to build reduced-order models. This method is advantageous in situations requiring multiple evaluations of the solution operator, such as in engineering design problems. For example, the Fourier neural operator has been used to speed up the sampling process in diffusion models, allowing for efficient data generation and optimization[8].
To further enhance the performance of neural operators, novel strategies such as pretraining models in lower dimensions and transferring the weights to multidimensional models have been proposed. This approach has shown significant improvements in low-data regimes, particularly for systems with slow rates of change[5]. Such pretrained models have demonstrated error reductions of up to 70% compared to models trained from random initialization, especially in autoregressive scenarios with multiple prediction steps[5].

Applications

Fourier Neural Operators (FNOs) have demonstrated significant potential across various scientific and engineering domains due to their ability to model complex systems efficiently.

Fluid Dynamics

FNOs have been applied to solve fundamental Partial Differential Equations (PDEs) in fluid dynamics, such as the Navier-Stokes equations. These equations are pivotal in predicting the behavior of fluid flows, which is critical in fields like aerospace engineering and climate modeling. Zhao et al. (2023) utilized Local Convolution Enhanced Global Fourier Neural Operators to predict multiscale dynamic spaces, enhancing the ability to model fluid dynamics accurately [10]. Furthermore, the 2D incompressible Navier-Stokes dataset, as referenced in recent studies, demonstrates the practical application of FNOs in modeling fluid viscosity and velocity fields [4].

Weather Forecasting

One prominent application of FNOs is in weather forecasting. Recent studies by Lam et al. (2023) have shown that FNOs can learn skillful medium-range global weather forecasts, providing a robust alternative to traditional numerical weather prediction models [8]. The ability of FNOs to handle large-scale spatial-temporal data makes them particularly suited for this task.

Solar Physics

In solar physics, FNOs have been employed to model solar wind parameters and simulate solar flare events. Yang and Shen (2019) demonstrated the use of artificial neural networks, including FNOs, to model the global distribution of solar wind parameters using multiple observations [10]. Similarly, Yamasaki et al. (2022) conducted data-constrained magnetohydrodynamic simulations of solar flares, highlighting the applicability of FNOs in space weather prediction [10].

Medical Imaging

In the domain of medical imaging, FNOs are used for high-resolution tasks such as semantic segmentation, object detection, and disease localization. The work by Nehme et al. (2020) on 3D localization microscopy using deep learning techniques underscores the potential of FNOs in achieving dense and precise image localization [12]. Additionally, they are employed in various medical image classification tasks, such as detecting COVID-19 infection using lung CT images and segmenting pneumonia lesions from chest X-rays [12].

Computational Efficiency

FNOs provide computational efficiency by enabling additional non-linear processing stages, which enhance their capability to model complex functions and solve a diverse array of PDE solutions effectively. This advantage is crucial for applications requiring high computational performance, such as real-time 2D/3D registration in medical imaging and large-scale weather prediction simulations [10].

Advantages and Limitations

Advantages

Improved Convergence

Fourier Neural Operators (FNOs) exhibit enhanced convergence properties, especially when dealing with deep networks. They empower the training of networks with more than 100 layers, as demonstrated with the Stochastic Gradient Descent (SGD) algorithm. For instance, a Highway Network with a depth of fifty layers has shown improved rates of convergence compared to thinner and deeper architectures[12]. This improvement is critical in achieving better performance and faster training times.

Effective for Large-Scale Problems

FNOs are particularly useful in scenarios where the problem size is extremely large and exceeds human reasoning abilities. They excel in applications such as sentiment analysis, ad matching on social media platforms, and webpage ranking calculations[12]. This capacity makes them suitable for large-scale data-intensive fields, including healthcare and environmental science, where data heterogeneity and volume pose significant challenges[12].

Flexibility and Hierarchical Feature Extraction

FNOs offer flexibility in fine-tuning strategies, which can be adjusted based on the availability of training data. For example, tuning a smaller subset of parameters may achieve lower errors when few training samples are available, whereas more trainable parameters can result in a lower error with sufficient training data[5]. This flexibility suggests that FNOs can effectively perform hierarchical feature extraction, which is crucial for various complex applications[5].

Data Augmentation and Overfitting Mitigation

To avoid overfitting, data augmentation techniques are used to expand the training dataset size artificially. This method is essential for training models on a sizeable amount of data, ensuring that the model generalizes well to unseen data[12]. These augmentation techniques play a pivotal role in mitigating overfitting, especially when training data is scarce[5].

Robustness to Outliers

FNOs demonstrate robustness to outliers by employing specific loss functions that measure absolute deviation. This characteristic is vital in tasks requiring robustness to outliers or when it is essential to evaluate absolute deviations accurately[8].

Limitations

Computational and Memory Requirements

One of the primary limitations of FNOs is their intensive memory and computational requirements. Due to their complex architectures and large numbers of parameters, deploying these models on machines with limited computational power, such as those in the healthcare field, can be challenging[12]. However, advancements in hardware-based parallel processing solutions like FPGAs and GPUs offer potential solutions to these computational issues[12].

Risk of Overfitting

The risk of overfitting is a significant challenge, particularly when the downstream task has scarce training data[5]. Although data augmentation techniques can mitigate this risk, careful consideration is required to balance the number of trainable parameters and the available training samples to prevent overfitting effectively.

Dependence on Pretraining and Fine-Tuning Strategies

The advantage of pretraining FNO models is highly dependent on the availability of training data in the downstream task. Various fine-tuning strategies must be explored to determine the best way to freeze or tune the transferred weights from a pretrained model[5]. This dependence can complicate the model training process, requiring extensive experimentation to identify the optimal strategy.

Limited Interpretability

Despite their powerful performance, FNOs, like many deep learning models, may lack interpretability. In cases where human experts need to understand the decisions made by the model, this lack of interpretability can be a significant drawback[12].

Comparison with Other Models

The Fourier Neural Operator (FNO) distinguishes itself from other neural network architectures by leveraging the Fourier transform to handle high-dimensional partial differential equations (PDEs). This approach provides several advantages and unique challenges compared to traditional models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

CNNs and FNO

Convolutional Neural Networks (CNNs) are well-known for their weight-sharing mechanism, which significantly reduces the number of trainable parameters and helps improve generalization by avoiding overfitting[12]. CNN layers consist of convolutional filters that generate feature maps from input data. However, CNNs struggle with capturing long-range dependencies and high-dimensional data typical in PDE applications. FNO, on the other hand, employs the Fourier transform to effectively handle high-dimensional data by truncating the Fourier series at a maximal number

of modes, thereby focusing on essential frequency components[7].
Despite their strengths, CNNs are prone to overfitting when dealing with complex data, whereas FNOs aim to mitigate this by leveraging spectral information, which provides a more compact representation of the solution space[12]. This spectral approach allows FNOs to capture intricate patterns in high-dimensional PDEs more efficiently than CNNs.

RNNs and FNO

Recurrent Neural Networks (RNNs) are primarily used for sequential data and are adept at capturing temporal dependencies, making them suitable for applications in speech processing and natural language processing (NLP)[12]. RNNs employ a hidden state that is updated sequentially, capturing short-term dependencies effectively. However, their applicability to high-dimensional PDEs is limited due to their inherent sequential nature, which can lead to inefficiencies in capturing spatial dependencies.
The FNO addresses these limitations by transforming the problem into the frequency domain, where spatial dependencies can be more easily managed[7]. This transformation is particularly advantageous for PDEs, as it allows for the simultaneous handling of multiple spatial dimensions without the need for sequential processing inherent to RNNs.

Multigrid Methods and FNO

Traditional multigrid (MG) methods are highly efficient for solving PDEs by iteratively refining the solution on progressively finer grids[6]. These methods are well-established and provide robust solutions for linear systems by employing techniques like the Jacobi iteration for error correction. The FNO integrates the multigrid approach by constructing a multi-scale model that converts high-frequency components into more manageable low-frequency components[6]. This integration allows the FNO to leverage the strengths of MG methods while maintaining the flexibility and adaptability of neural networks.

Key Challenges

Designing robust stress tests that align well with applied requirements and provide comprehensive coverage of potential failure modes remains a significant challenge in the domain of machine learning (ML) and deep learning (DL). Underspecification imposes substantial constraints on the credibility of ML predictions, which necessitates reconsideration of its application in critical fields like medical imaging and self-driving cars [12].
One of the prominent challenges in deep learning is overfitting, where models perform exceptionally well on training data but fail to generalize to new, unseen data. This occurs when a model becomes too complex or is trained for too long, memorizing noise and specific examples instead of capturing genuine patterns, thus resulting in poor performance on real-world tasks [13][14]. The first step to address overfitting is to decrease the complexity of the model by removing layers or reducing the number of neurons to make the network smaller, though the exact amount to reduce is context-dependent [15][13].
Another related issue is the trade-off between overfitting and underfitting. While

overfitting involves the model capturing noise, underfitting happens when the network cannot model the training or test data adequately, leading to poor performance across both datasets. This can result from an overly simplistic model that fails to capture the underlying trends [14].
Furthermore, the discrepancy between training and validation performance is another indicator of overfitting. Methods to detect this include monitoring validation loss, analyzing learning curves, and comparing validation accuracy against training accuracy [14]. Ensuring robust generalization capabilities in neural networks is essential for their effectiveness on unseen data [6].
The invariance to discretization in neural operators also presents a challenge. Since data and functions are often accessed numerically through point-wise evaluations, ensuring that the neural operator can produce consistent results across different discretization levels is highly desirable. This property allows for the transfer of solutions between varying grid geometries and discretizations, which is crucial for the adaptability and robustness of the models [3][16].

Implementations and Libraries

The incorporation of new information into a plain deep learning (DL) model, often involving the fine-tuning of models with new classes, can result in poor performance with older classes if not handled properly[12]. Addressing this challenge typically involves training an entirely new model from scratch using both old and new data, which is computationally intensive and time-consuming. This issue is common across many fields, including Biology[12]. Several computational approaches and techniques have been developed to manage these challenges more effectively.

Model Compression Techniques

Model compression techniques have become essential for reducing computational costs while maintaining performance.

[12]

[12]

[12]

[12]

DL Frameworks and Libraries

Numerous DL frameworks and libraries have emerged, streamlining the training and deployment of DL models. Among these, TensorFlow is deemed the most effective and user-friendly based on star ratings on GitHub and expert backgrounds[12]. It supports multiple platforms and facilitates various DL tasks. Other popular frameworks include PyTorch, Keras, and MXNet.

Benchmark Datasets

Benchmark datasets are crucial for evaluating the performance of DL models across different tasks. Several datasets have been established and are widely used in the DL community. Some notable examples include ImageNet for image classification, COCO for object detection, and LibriSpeech for speech recognition.

Software Tools and Resources

The development and deployment of DL models often involve additional software tools and resources. For instance, Python interfaces with high-performance libraries like PETSc can be used for efficient linear and nonlinear solvers[8]. Several platforms and tools support collaborative work and code sharing, such as GitHub, which also serves as a repository for pre-trained models and associated scripts[17][18][19]. Tools like DagsHub and CatalyzeX facilitate code discovery and integration with research articles[19].
Furthermore, simple implementations of common operations, such as the 1D convolution operation in Python, are essential for understanding and developing custom DL models[20]. Such implementations are crucial for educational purposes and for building foundational knowledge in the field.

References

[1]: Neural operators - Wikipedia

[2]: Neural Operators: an Introduction — neuraloperator 0.3.0 documentation

[3]: [2010.08895] Fourier Neural Operator for Parametric Partial ...

[4]: Toward a Better Understanding of Fourier Neural Operators:

[5]: Pretraining a Neural Operator in Lower Dimensions - arXiv.org

[6]: MgFNO: Multi-grid Architecture Fourier Neural Operator for Parametric ...

[7]: Incremental Spatial and Spectral Learning of Neural Operators

[8]: A Mathematical Guide to Operator Learning - arXiv.org

[9]: Fourier Neural Operator with Learned Deformations for PDEs on General ...

[10]: Neural Operator for Accelerating Coronal Magnetic Field Model - arXiv.org

[11]: Intuitive Guide to Convolution – BetterExplained

[12]: Review of deep learning: concepts, CNN architectures, challenges ...

[13]: Guide to Prevent Overfitting in Neural Networks - Analytics Vidhya

[14]: Techniques To Prevent Overfitting In Neural Networks - Analytics Vidhya

[15]: 5 Techniques to Prevent Overfitting in Neural Networks

[16]: GitHub - raj-brown/fourier_neural_operator: Use Fourier transform to ...

[17]: [2402.11568] A novel Fourier neural operator framework for ...

[18]: [2404.07200] Toward a Better Understanding of Fourier Neural Operators ...

[19]: Fourier Neural Operator Networks: A Fast and General Solver for the ...

[20]: Choosing Your Neural Network for NLP: A Hands-On Guide to CNNs vs. RNNs ...