Project Proposal: Investigating the Transfer Learning capability of PINNsFormers

(Paul Dietze)

1. Introduction

This proposal is based on the paper **PINNsFormer: A Transformer-Based Framework For Physics-Informed Neural Networks** which was accepted at ICLR 2024. In this paper the authors propose a transformer-based architecture for approximating partial differential equations (PDEs), as opposed to more traditional PINN architectures. The code, as well as the trained models for this research are available in a GitHub Repo.

The project I would like to do would not replicate this paper or its results but rather investigate the PINNsFormers ability for transfer learning. Transformer based foundational models that are pretrained on a variety of data and then fine-tuned for a particular task have proven to be widely useful in many NLP (also vision/multimodal) tasks. For this reason, it would be very interesting to investigate how a PINNsFormer pre-trained on a family of PDEs with similar characteristics (e.g. various convection speeds) and then fine-tuned on a new PDE would compare with more traditional PINN models. It would be very interesting to see whether the transformer-based foundation model exhibits faster convergence, better final accuracy, or both.

The research questions I would like to address are the following:

1. Transfer learning efficacy: Does a PINNsFormer, pre-trained on multiple PDE instances, fine-tune faster and more accurately than a randomly initialized network or a standard MLP PINN?

2. **Generality:** Can a single PINNsFormer learn a "universal PDE prior"? For example, if pre-trained on 1D reaction PDEs, can it transfer effectively to a slightly modified PDE with a different reaction coefficient or boundary condition?

3. **Comparison to Baseline PINNs:** When is the transformer architecture truly beneficial, and when might a well-tuned MLP perform nearly as well? What are the boundary conditions or PDE regimes that showcase the biggest gap?

2. Methods

A possibility could be to focus just on one or two PDE families like: 1D Convection PDE and 1D Reaction PDE and create a dataset of solution snapshots using numerical solvers.

Dataset (for example):

- Pre-training: Multiple PDE instances (for instance, β=5,10,20,40 in convection; ρ=2,5,10 in reaction).
- Fine-tuning Data: A held-out PDE instance, e.g. $\beta = 60$ not included in pre-training, or a brand-new PDE from reaction if only convection PDEs were used for training.

Pre-training:

- Implement an extended training loop where, in each iteration, I randomly pick from the pool of PDE tasks (mini-batch PDE sampling).
- Input to PINNsFormer includes solution snapshots plus PDE parameter (possibly)
- Minimize the combined PDE losses across all PDE tasks.

Fine-tuning:

- Initialize the same PINNsFormer weights from pre-training.
- Train on the new PDE instance alone. Compare speed of convergence and final error to: (a) a PINNsFormer, (b) a standard MLP-based PINN, (c)?, (d)?...

Evaluation metrics:

- error vs. ground truth PDE solutions.
- Time to convergence: how many iterations are needed to reach a certain accuracy threshold.
- Ablation: Possibly test removing wavelet activation or self-attention to confirm the advantages of PINNsFormer.
- Others...

3. Computation

Clearly, training a foundation model is computationally expensive, which is why we would just focus on a small set of problems, just enough to investigate transfer learning. We would use Google Colab with GPU: multi-task training with a small transformer (2–4 layers, moderate hidden size) should be feasible within 8–12 hours of total GPU time if mini-batch sizes are kept modest.

If we were to realize that these resources are not enough, we could create a new Google Vertex AI account, which would give us 300 $ in Google Cloud credits, which we could spend on computational resources.

4. Why I really want to do this project.

Throughout the last semesters and my bachelor, I have been specializing in NLP. Using a transformer-based model would be a nice bridge from that area to the GNN in Science topics we covered in the lecture. The project is feasible in the allotted timeframe: we can easily generate synthetic data and run small scale transformers. The PINNsFormer code is provided, but would have to be adopted, for example to create a custom training loop for multi-task training. One could also add comparisons to other PINN variations if there is time and increase the scope of downstream experiments as needed. The idea is also novel enough, since we would try to essentially address the question (on a very small scale): Can we replicate the success of transformer-based foundation models in PDE solving?