**Comparing Classical and AI Algorithms for Simulation of Dispersive and Nonlinear Media**

**Coding Skills Required: Intermediate**

**Goals**

The goal of this project is to compare the runtime and accuracy of three methods for simulating optical propagation in dispersive nonlinear materials: physical experiments, the split-step Fourier method, and a deep neural network approximating the split-step Fourier method that you will build.

**Introduction**

The propagation of light in optical fibers is governed by the Nonlinear Schrödinger Equation (NLSE), a complex equation with coupled dispersion and nonlinearity effects, for which no analytical solution exists. The widely adopted approach for solving the NLSE is the Split-Step Fourier Method (SSFM), a time-consuming iterative process. Interestingly, each step in SSFM involves a linear operation (dispersion) followed by a nonlinear operation (nonlinearity), resembling a typical neural network block. In this project, you will explore the design and implementation of a neural network to approximate the solution of the NLSE more efficiently and compare the performance of physical experiments, the traditional SSFM, and your neural network approximation.

**Tasks**

**1. Understand Physical Experiments**

Read paper “”. Start with learning the fundamentals of the Nonlinear Schrödinger Equation (NLSE) and the Split-Step Fourier Method (SSFM) by studying the Chapter 2 of “Nonlinear Fiber Optics” by Agrawal.

**2. Implement Split-Step Fourier Method**

Code the split-step Fourier method for simulating optical propagation. Using your SSFM code, generate the following figures from Agrawal book: Figure 3.1, Figure 4.1, Figure 4.2, Figure 4.6.

**3. Develop Neural Network for Split-Step Approximation**

Gain an understanding of the neural network solver designed for 1D-NLSE by reviewing paper1 and paper2. Create your own PyTorch implementation of the NN solver for 1D-NLSE. Compare the performance of NN solver and SSFM for 1D-NLSE in terms of both speed and accuracy using different input waveforms.

**4. Quantify Acceleration from Neural Network**

Benchmark end-to-end runtime of physical experiments, split-step simulations, and neural network approximations. Quantify acceleration and accuracy tradeoffs.

**5. (Time Allowing) Generalize SSFM and the NN solver to 2D**

Building upon the knowledge and skills acquired in tasks 1 and 2, extend the implementation of both SSFM and NN solver for 2D-NLSE. To validate your SSFM implementation, use an image as the input, and disable the nonlinearity and attenuation term to see if you can reproduce the results of PST and VEViD. Test the noise propagation and reversibility of both SSFM and the NN solver.

**Deliverables**

PowerPoint slides, final presentation, and GitHub repository.

**Resources**

**Contact**

Yiming Zhou (Ph.D. student at Jalali-Lab): yimingz0416@g.ucla.edu

Callen MacPhee (Ph.D. student at Jalali-Lab): cmacphee@g.ucla.edu