A Visualization of Deep learning and Machine learning Models

Aaryadev Ghosalkar
Savitribai Phule Pune University

Data Visualisation

07 September 2023

Abstract

This project focuses on the integration of PyTorch with Python libraries like NumPy and Matplotlib to develop an instructive tool for visualizing the impact of common hyperparameters in machine learning models. While designed primarily as a teaching aid, the project delivers an interactive, user-friendly tool that empowers learners to explore and fine-tune hyperparameters. It enhances their comprehension of deep learning models and optimization techniques.

By bridging PyTorch with Python packages, this project underscores the pedagogical value of visualization in model refinement. It offers a valuable resource for educators and learners in the machine learning community, highlighting the significance of accessible tools for effective model evaluation and optimization.

Introduction

In the rapidly evolving field of machine learning and deep learning, gaining a deep understanding of the intricacies of model training and hyperparameters is essential for aspiring data scientists and engineers. The project "A Visualization of Deep Learning and Machine Learning Models" aims to bridge the gap between theoretical knowledge and practical comprehension for students, particularly those in undergraduate STEM programs. This project offers a valuable learning tool that empowers students to grasp the impact of hyperparameters on model performance through visualization, making complex concepts more accessible.

Traditional learning resources often present hyperparameters in a theoretical manner, which can pose challenges for students seeking a more intuitive understanding of their practical effects. This project addresses this gap by providing an interactive application that allows users to experiment with hyperparameters in a hands-on way. It currently supports linear regression and basic Artificial Neural Networks (ANN), aligning with the skill level of undergraduates with basic knowledge of machine learning and deep learning.

The core functionality of the project centers on an application where users can set hyperparameters and witness the model's training process. Through this tool, users can observe the model's predictions evolve across epochs, providing a dynamic and visual perspective on the learning process. As an aid tailored to the educational needs of students, this project offers a valuable resource for those striving to comprehend and harness the power of hyperparameters in the world of machine learning and deep learning.

Methodology

Introduction

In this section, we'll provide a concise overview of the methods and techniques used in the development of our interactive learning tool. We'll cover the choice of development tools, application architecture, algorithm flow, user interface design, testing, and other key aspects that shaped the creation of this educational resource. This methodology section offers a glimpse into the technical approach that underpins our project, giving you insights into how the tool was brought to life.

Development tools and environments

- Python
- PyTorch (Backend Deep learning framework)
- Numpy
- Matplotlib (Plotting and graphics backend

Application Architecture

Our interactive learning tool comprises three interconnected sections, each playing a vital role in delivering a comprehensive educational experience.

Frontend:

The frontend serves as the user interface, developed using the Tkinter library, which allows users to interact with the application.Users can select a machine learning model of their choice, set hyperparameters, and initiate the training process. The frontend communicates with the trainer (section 2) by passing user-defined hyperparameters for model training.

Trainer (PyTorch-based)

The trainer section, built using PyTorch, is responsible for training the selected machine learning model. It handles model training, including the generation of synthetic data and default settings for missing hyperparameters. The trainer receives hyperparameters from the frontend, trains the model in each epoch, and passes the model's output and training loss to the plotting section (section 3) for visualization.

Plotting (Matplotlib-based):

The plotting section utilizes the Matplotlib library to create dynamic visualizations of the training process. It receives data from the trainer in real-time, visualizes model predictions, and displays the training loss as an animated plot. Data on model outputs and loss are continuously sent from the trainer, and the plotting section updates the visualizations to reflect the ongoing training progress.

This three-tier architecture ensures a seamless flow of information and control between the user interface (frontend), the machine learning model trainer, and the dynamic visualization of training progress. It empowers users to experiment with hyperparameters, observe model behavior during training, and gain valuable insights into the effects of parameter adjustments on machine learning outcomes.

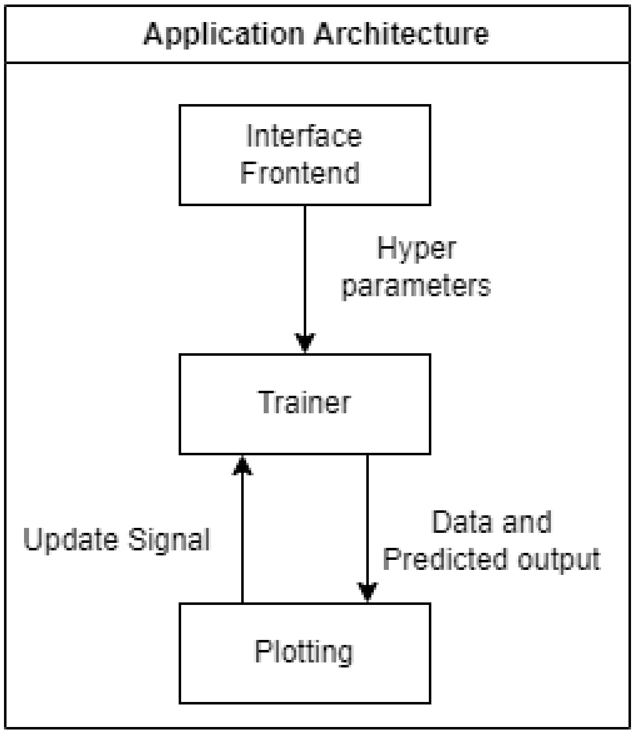
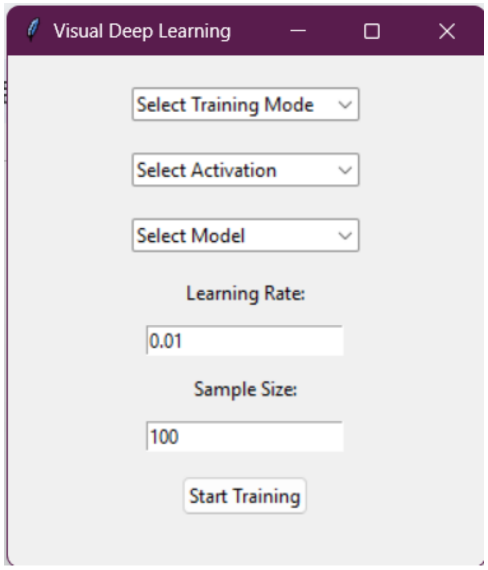


Figure 1: Application architecture diagram

- **Hyper parameters:** are the selected by the user in the application frontend this is done using various input elements in tkinter module
- **Data and predicted output:** are sent every epoch to the plotting tool, to draw the data as well as the models prediction, the data is static during training and only a reference is sent to plotting backend
- **Update Signal** is sent upon completion of 1 animation frame this indicates that the trainer should let the model train for another epoch. This loop continues until the maximum number of epochs is reached, which is a hyper parameter indicated by the user

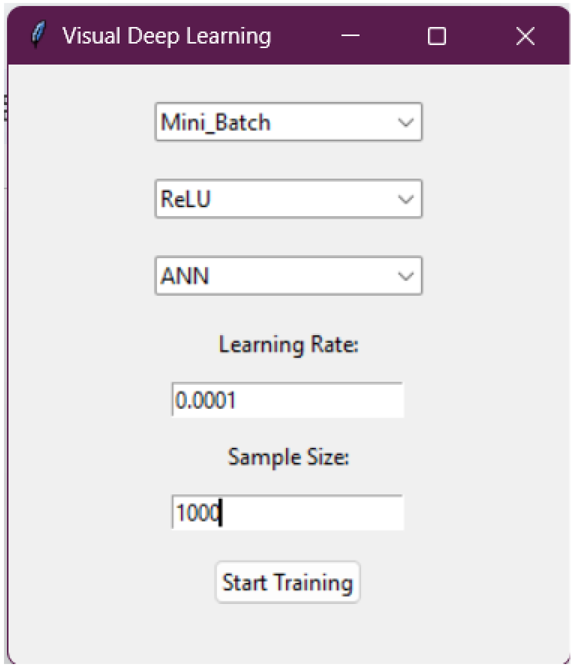
This application is open source under MIT license and all source code including presentation slides are uploaded to GitHub, the source can be found at this link Project Source (https://github.com/aaryadevg/DV-Project-pytorch)

Application Screenshots



Learning Rate: 0.01

Sample Size: 100



Mini_Batch

ReLU

ANN

Learning Rate: 0.0001

Sample Size: 1000

Figure 2: User Interface

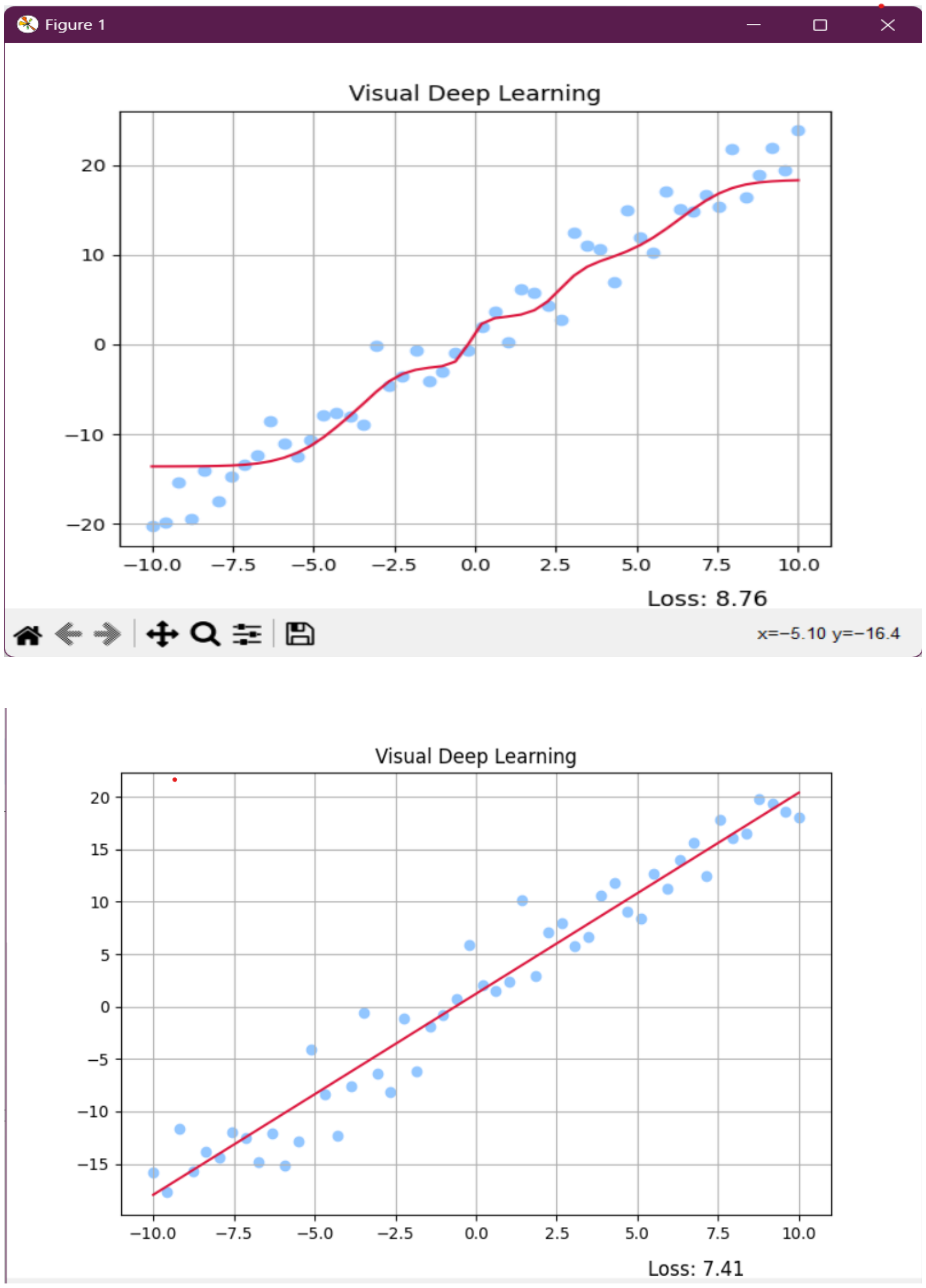


Figure 3: Plotting Showcase