Summer Project Summary

Name: Kshitish Kumar Ratha
Registration Number: MS22174

This project details the design and implementation of Kinone, a deep learning framework built from scratch using Python and NumPy. The primary motivation was to deconstruct the "black box" of production-grade frameworks like PyTorch and TensorFlow, creating a transparent, auditable, and educational tool. The goal was to gain a first-principles understanding of the internal mechanics of gradient-based learning by implementing every core component, from automatic differentiation to complete neural network models.

Framework Architecture and Implementation

The Kinone framework is a layered system designed for clarity and modularity.

Core Autograd Engine The foundation of Kinone is a reverse-mode automatic differentiation (autograd) engine. This engine operates on a define-by-run paradigm, dynamically constructing a computational graph as operations are performed. The core data structure is a `Tensor` class, which wraps a NumPy array and tracks the operations that create it. Every mathematical operation is a `Function` subclass, defining both a `forward` pass (the computation) and a `backward` pass (the analytical gradient). When the `backward()` method is called on a final scalar loss, the engine traverses the graph in reverse, applying the chain rule at each step to propagate gradients back to all model parameters. This process is a from-scratch implementation of backpropagation.

Neural Network Primitives and Models Built upon the autograd engine is a comprehensive library of neural network layers and operations. This includes fundamental building blocks like 2D convolutions, pooling, and batch normalization. These primitives are encapsulated in a `Module` system (inspired by PyTorch) that handles parameter tracking and state management (e.g., training vs. evaluation modes). The framework's expressive power was validated by implementing canonical deep learning architectures, including the influential ResNet and the modern, efficient EfficientNet.

Training Pipeline and Tooling To support an end-to-end machine learning workflow, Kinone is equipped with a custom data handling pipeline, including a `DataLoader` and specific logic for the NIH Chest X-ray dataset. Standard optimization algorithms, such as SGD with momentum and the widely used Adam optimizer, were implemented to update model parameters based on the computed gradients. To enhance usability and interoperability, the framework includes an exporter to the Open Neural Network Exchange (ONNX) format, allowing models trained in Kinone to be deployed in standard inference engines. A live monitoring dashboard built with Streamlit was also developed for real-time tracking of training progress.

Verification and Testing Strategy

Correctness was paramount. A rigorous testing strategy was employed to validate every component of the framework. The most critical aspect was ensuring the mathematical correctness of the autograd engine. This was achieved by systematically verifying the analytical gradient of every implemented operation against a numerical approximation calculated using the finite-difference method. This exhaustive comparison ensures that the gradients used for training are precise, providing a high degree of confidence in the framework's reliability.

Conclusion

The project successfully achieved its goal of creating a functional deep learning framework from first principles. The result, Kinone, serves as a powerful educational tool for understanding the core mechanics of deep learning. Key learnings from the process include a deeper, practical understanding of backpropagation, numerical stability considerations, and the software design patterns that make modern frameworks scalable and flexible.