Technical Documentation: Modified Difference Boosting Neural Network (DBNN) for PyTorch

Ninan Sajeeth Philip*
Artificial Intelligence Research and Intelligent Systems (airis4D))
Thelliyoor -689548
India

March 17, 2025

Contents

*nsp@airis4d.com
Phone: +91 9496552479, +91 9497552476

1 Introduction

This document provides a detailed technical overview of the modified Difference Boosting Neural Network (DBNN) optimized for the PyTorch platform. The DBNN is a probabilistic neural network that leverages pairwise feature interactions to model complex data distributions. The modifications include GPU acceleration, adaptive learning, and improved handling of high-cardinality features.

2 Architecture Overview

The DBNN architecture consists of the following key components:

- **Feature Pair Processing**: The model processes features in pairs, computing likelihoods for each pair.
- **Histogram and Gaussian Models**: The model supports both histogram-based and Gaussian-based likelihood computations.
- **Adaptive Learning**: The model adapts its learning rate and feature selection based on performance.
- **Invertible DBNN**: An optional invertible model for feature reconstruction.

3 Mathematical Description

3.1 Likelihood Computation

The likelihood of a feature pair (x_i, x_j) given a class c is computed as:

$$P(x_i, x_j|c) = \sum_{k=1}^{K} w_{c,k} \cdot \text{PDF}(x_i, x_j|\theta_{c,k})$$

where $w_{c,k}$ are the weights for class c and feature pair k, and PDF is the probability density function (either histogram-based or Gaussian).

3.2 Posterior Probability

The posterior probability of class c given the feature vector $\mathbf{x}$ is:

$$P(c|\mathbf{x}) = \frac{P(\mathbf{x}|c) \cdot P(c)}{\sum_{c'} P(\mathbf{x}|c') \cdot P(c')}$$

where $P(c)$ is the prior probability of class c.

3.3 Weight Update

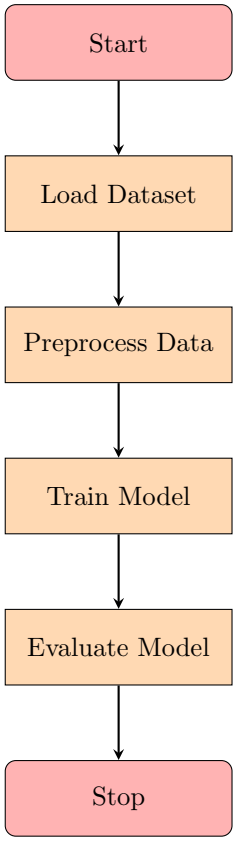
The weights are updated using the following rule:

$$w_{c,k} \leftarrow w_{c,k} + \eta \cdot (1 - \frac{P(c|\mathbf{x})}{P(c'|\mathbf{x})})$$

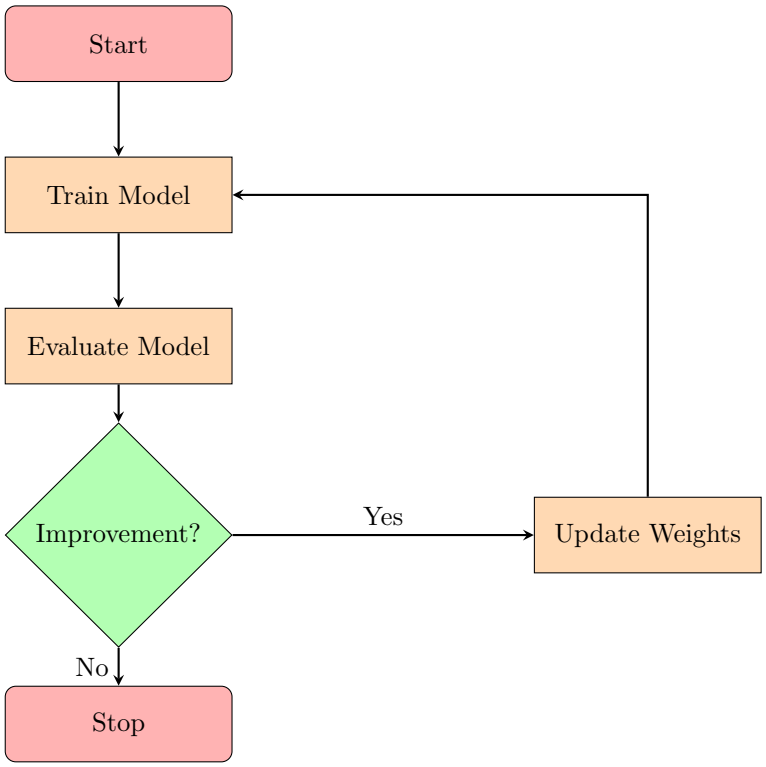
where η is the learning rate, and c' is the predicted class.

4 Flowcharts

4.1 Training Process



4.2 Adaptive Learning



5 Features

5.1 GPU Acceleration

The model is optimized for GPU acceleration using PyTorch's CUDA backend. This allows for faster computation of likelihoods and weight updates.

5.2 Adaptive Learning

The model includes an adaptive learning mechanism that adjusts the learning rate and feature selection based on performance. This helps in converging to a better solution faster.

5.3 Invertible DBNN

The model includes an optional invertible DBNN for feature reconstruction. This allows for the reconstruction of input features from the model's predictions.

5.4 High-Cardinality Feature Handling

The model includes improved handling of high-cardinality features by removing or rounding features with too many unique values.

6 Conclusion

The modified DBNN optimized for PyTorch provides a powerful tool for probabilistic modeling of complex data distributions. The inclusion of GPU acceleration, adaptive learning, and improved feature handling makes it suitable for a wide range of applications.