xTorch: A High-Level C++ Extension Library for PyTorch (LibTorch)

Tags

Summary

xTorch is a C++ library that extends PyTorch's C++ API (LibTorch) with high-level abstractions and utilities, aiming to provide a more user-friendly and productive experience for C++ developers working with deep learning. It bridges the usability gap in PyTorch's C++ API, which had become cumbersome for end-to-end model development after 2019.

Statement of Need

xTorch addresses the lack of high-level APIs in LibTorch for C++ developers, which is critical for high-performance machine learning, robotics, embedded applications, and large-scale deployment scenarios. By reintroducing high-level utilities that were deprecated in the Python API post-2019, xTorch enables C++ developers to build, train, evaluate, and deploy models more intuitively and efficiently.

Functionality

xTorch provides:

- High-level neural network module definitions (e.g., XTModule, ResNetExtended, XTCNN)
- A simplified training loop with the Trainer class, handling loss computation, metrics, and callbacks
- Enhanced data handling with ImageFolderDataset, CSVDataset, and OpenCV-backed transformations
- Utility functions for logging, metrics computation, and device management
- Extended optimizers like AdamW, RAdam, and learning rate schedulers
- Model serialization and TorchScript export helpers (save_model(), export_to_jit())
- Inference utilities for loading models and making predictions

The library is modular and extensible, built on top of LibTorch, and supports both CPU and CUDA devices.

Example Use

"'cpp // Example: CNN Training Pipeline auto trainData = xt::datasets::ImageFolder("data/train", xt::transforms::Compose({ xt::transforms::Resize({224, 224}), xt::transforms::ToTensor(), xt::transforms::Normalize({0.5, 0.5, 0.5}, {0.5, 0.5, 0.5}) }));

auto trainLoader = xt::data::DataLoader(trainData, 64, true);

auto model = xt::models::ResNet18(10); auto optimizer = xt::optim::Adam(model.parameters(), 1e-3); auto criterion = xt::loss::CrossEntropyLoss();

xt::Trainer trainer; trainer.setMaxEpochs(20) .setOptimizer(optimizer) .setCriterion(criterion) .fit(model, trainLoader);

// Export model to TorchScript xt::export_to_jit(model, “model.pt”);