MUHAMMAD ALTHAF DHIAULHAQ / 1103194046

# TECHNICAL REPORT PYTORCH

## 1. Introduction

PyTorch is an open-source machine learning framework developed by Facebook's AI Research Lab (FAIR). It provides a flexible and dynamic approach to building and training neural networks, making it popular among researchers and practitioners in the field of deep learning. This technical report provides an overview of PyTorch, its key features, and its usage in developing deep learning models.

## 2. Key Features of PyTorch

### 2.1. Dynamic Computational Graph

Unlike other frameworks that use static computational graphs, PyTorch employs a dynamic computational graph. This allows users to define and modify their neural network models on the fly, enabling more flexibility in experimenting with different architectures and adapting models during runtime.

### 2.2. GPU Acceleration

PyTorch supports seamless GPU acceleration, allowing users to leverage the power of graphics processing units for faster training and inference. It provides easy-to-use interfaces for moving tensors and models to GPUs, making it effortless to utilize parallel computing capabilities.

### 2.3. Automatic Differentiation

PyTorch's automatic differentiation engine, known as Autograd, enables efficient computation of gradients for backpropagation. By tracking operations on tensors, it automatically computes and propagates gradients through the network, simplifying the process of training complex models.

### 2.4. Extensive Neural Network Library

PyTorch offers a rich library of pre-defined layers, loss functions, and optimization algorithms, making it easier to build and train neural networks. It also supports various neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers.

### 2.5. Integration with Python Ecosystem

Being built on Python, PyTorch seamlessly integrates with popular Python libraries such as NumPy and SciPy. This integration enables smooth data handling, preprocessing, and interoperability with other scientific computing tools.

## 3. PyTorch Workflow

### 3.1. Define the Model

PyTorch allows users to define their neural network models as Python classes by subclassing the `torch.nn.Module` class. This includes defining the layers, forward pass, and any additional customizations required.

### 3.2. Data Preparation

PyTorch provides utilities for loading and preprocessing data, including built-in datasets and data loaders. Users can also customize their data loading pipeline by creating custom `Dataset` classes and implementing transformations.

### 3.3. Training

During the training phase, users define the loss function, select an optimization algorithm (such as stochastic gradient descent), and iterate over the training data. PyTorch's Autograd automatically calculates gradients, which are used to update the model's parameters through the optimizer.

### 3.4. Evaluation

After training, models can be evaluated using test data. PyTorch provides tools for calculating various evaluation metrics, such as accuracy, precision, recall, and F1 score.

### 3.5. Saving and Loading Models

PyTorch allows users to save trained models and their associated parameters to disk. These saved models can later be loaded and used for inference or further training.

## 4. PyTorch Ecosystem

### 4.1. TorchVision

TorchVision is a PyTorch library providing datasets, models, and transformations specifically for computer vision tasks. It includes popular datasets like ImageNet and CIFAR, pre-trained models, and vision-specific data augmentation techniques.

### 4.2. TorchText

TorchText is a PyTorch library designed for natural language processing (NLP) tasks. It provides utilities for tokenizing text, building vocabularies, and creating datasets for tasks such as sentiment analysis, machine translation,