**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |
| **Course/Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |

**CS-477 Computer Vision**

**Lab#8: introduction to Pytoroch**

Tutorial from pytorch site

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **PLO4-CLO4** | **PLO5-CLO5** | **PLO8-CLO6** | **PLO9-CLO7** |
| **Name** | **Reg. No** | **Investigation**  **(5 marks)** | **Modern Tool Usage**  **(5 marks)** | **Ethics**  **(5 marks)** | **Individual and Team Work**  **(5 marks)** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Lab#8: introduction to Pytorch**

**Objectives**

* Understand PyTorch’s Tensor library and neural networks at a high level.
* Introduction to autogradaint
* Training a classifier

**Lab Instructions**

* This lab activity comprises of following parts: Lab Exercises, and Post-Lab Viva/Quiz session.
* The lab report shall be uploaded on LMS.
* Only those tasks that are completed during the allocated lab time will be credited to the students. Students are however encouraged to practice on their own in spare time for enhancing their skills.

**Lab Report Instructions**

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

* Lab objectives
* Python codes
* Results (graphs/tables) duly commented and discussed
* Conclusion

# PyTorch Introduction

PyTorch is an open-source machine learning framework based on the Torch library. It is primarily developed by Facebook's AI Research lab. PyTorch is one of the most popular deep learning frameworks, alongside TensorFlow. It is known for its ease of use, flexibility, and dynamic computational graph.

PyTorch is a Python library that provides a high-level interface for building and training deep learning models. It is built on top of Torch, a scientific computing library for Lua. PyTorch offers a number of advantages over other deep learning frameworks, including:

* **Ease of use:** PyTorch is very easy to use, especially for those who are already familiar with Python. It has a simple and intuitive API, and it is well-documented.
* **Flexibility:** PyTorch is very flexible. It allows users to define their own custom models and training loops. It also supports a wide range of hardware platforms, including CPUs, GPUs, and TPUs.
* **Dynamic computational graph:** PyTorch has a dynamic computational graph, which means that the graph can be modified at runtime. This makes it easy to experiment with different model architectures and training algorithms.

### PyTorch Features

PyTorch offers a number of features that make it a popular choice for deep learning research and development, including:

* **Tensors:** Tensors are the core data structure in PyTorch. Tensors are multi-dimensional arrays that can be used to represent a wide variety of data, such as images, text, and audio.
* **Automatic differentiation:** Automatic differentiation is a technique for calculating the gradients of a function with respect to its inputs. Automatic differentiation is essential for training deep learning models.
* **Modules:** Modules are reusable building blocks for deep learning models. Modules can be combined to create complex models.
* **Optimizers:** Optimizers are algorithms for updating the parameters of a deep learning model during training. PyTorch provides a variety of optimizers to choose from.
* **Datasets and DataLoaders:** PyTorch provides a number of built-in datasets and DataLoaders. DataLoaders are used to load and batch data efficiently.
* **Model serialization:** PyTorch models can be serialized to disk or memory. This makes it easy to save and load models, and to deploy them to production environments.

### PyTorch Applications

PyTorch can be used to build a wide variety of deep learning models, including:

* **Image classification:** PyTorch can be used to build image classification models that can identify objects in images.
* **Object detection:** PyTorch can be used to build object detection models that can locate and identify objects in images.
* **Semantic segmentation:** PyTorch can be used to build semantic segmentation models that can segment images into different objects and regions.
* **Natural language processing:** PyTorch can be used to build natural language processing models that can understand and generate human language.
* **Speech recognition:** PyTorch can be used to build speech recognition models that can transcribe audio into text.
* **Machine translation:** PyTorch can be used to build machine translation models that can translate text from one language to another.

PyTorch is a powerful and flexible deep learning framework that is well-suited for a wide variety of tasks. It is easy to use and offers a number of features that make it a popular choice for both research and development. If you are interested in learning more about deep learning, PyTorch is a great place to start.

## PyTorch Tensors

Tensors are the fundamental data structure in PyTorch. They are n-dimensional arrays that can be used to represent a wide variety of data, such as images, text, and audio. Tensors are like NumPy arrays, but they have the additional advantage of being able to run on GPUs. This makes them ideal for training and deploying deep learning models.

Tensors can be created in PyTorch using a variety of methods, including:

* From NumPy arrays: You can convert a NumPy array to a PyTorch tensor using the torch.from\_numpy() function.
* From scratch: You can create a new tensor from scratch using the torch.Tensor() function.
* From random data: You can create a tensor filled with random data using the torch.randn() function.

#### Tensor Operations

PyTorch provides a variety of operations for tensors, including:

* Arithmetic operations: You can perform basic arithmetic operations on tensors, such as addition, subtraction, multiplication, and division.
* Element-wise operations: You can perform element-wise operations on tensors, such as taking the square root or exponent of each element.
* Linear algebra operations: You can perform linear algebra operations on tensors, such as matrix multiplication and inversion.

#### Tensor Broadcasting

Tensor broadcasting is a mechanism that allows PyTorch to perform computations on tensors of different shapes. Broadcasting works by automatically expanding the smaller tensor to the shape of the larger tensor. This allows you to perform operations on tensors of different shapes without having to explicitly reshape them.

#### Tensor Gradients

Gradients are the derivatives of a function with respect to its inputs. Gradients are essential for training deep learning models. PyTorch provides a built-in mechanism for calculating gradients, which is called automatic differentiation.

#### Tensor GPU Acceleration

One of the key advantages of PyTorch tensors is that they can be accelerated using GPUs. This can significantly improve the performance of deep learning models. To enable GPU acceleration, you need to set the device argument of the torch.Tensor() function to cuda.

PyTorch tensors are a powerful and flexible data structure that can be used to represent a wide variety of data and perform a wide range of computations. They are also easy to use and can be accelerated using GPUs. If you are interested in learning more about PyTorch, I recommend starting with the official PyTorch documentation.

## Lab Task 1

Run the following notebook and learn about tensor manipulation.

* <https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/3dbbd6931d76adb0dc37d4e88b328852/tensor_tutorial.ipynb>
* <https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html>

#### INTRODUCTION TO TORCH.AUTOGRAD

torch.autograd is PyTorch’s automatic differentiation engine that powers neural network training. In this section, you will get a conceptual understanding of how autograd helps a neural network train.

#### Background

Neural networks (NNs) are a collection of nested functions that are executed on some input data. These functions are defined by parameters (consisting of weights and biases), which in PyTorch are stored in tensors.

Training a NN happens in two steps:

**Forward Propagation**: In forward prop, the NN makes its best guess about the correct output. It runs the input data through each of its functions to make this guess.

**Backward Propagation**: In backprop, the NN adjusts its parameters proportionate to the error in its guess. It does this by traversing backwards from the output, collecting the derivatives of the error with respect to the parameters of the functions (gradients), and optimizing the parameters using gradient descent.

## Lab task 2

Run the following notebook and learn about tensor manipulation.

<https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html>

https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/\_downloads/8eed7e178f8fa30798f280ea82ff468b/autograd\_tutorial.ipynb

## NEURAL NETWORKS

Neural networks can be constructed using the torch.nn package.

Now that you had a glimpse of autograd, nn depends on autograd to define models and differentiate them. An nn.Module contains layers, and a method forward(input) that returns the output.

A typical training procedure for a neural network is as follows:

* Define the neural network that has some learnable parameters (or weights)
* Iterate over a dataset of inputs
* Process input through the network
* Compute the loss (how far is the output from being correct)
* Propagate gradients back into the network’s parameters
* Update the weights of the network, typically using a simple update rule: weight = weight - learning\_rate \* gradient

## Lab task 3

Run the following notebook and learn about neural network.

* [**https://pytorch.org/tutorials/beginner/blitz/neural\_networks\_tutorial.html**](https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html)
* **https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/\_downloads/c029676472d90691aa145c6fb97a61c3/neural\_networks\_tutorial.ipynb**

## TRAINING A CLASSIFIER

This is it. You have seen how to define neural networks, compute loss and make updates to the weights of the network.

Now you might be thinking,

Generally, when you have to deal with image, text, audio or video data, you can use standard python packages that load data into a numpy array. Then you can convert this array into a torch.\*Tensor.

* For images, packages such as Pillow, OpenCV are useful
* For audio, packages such as scipy and librosa
* For text, either raw Python or Cython based loading, or NLTK and SpaCy are useful

## Lab task 4

Run the following notebook and learn training a classifier.

* <https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html>
* https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/\_downloads/4e865243430a47a00d551ca0579a6f6c/cifar10\_tutorial.ipynb

# Helpful links

**Neural networks**

[**https://www.youtube.com/watch?v=CqOfi41LfDw**](https://www.youtube.com/watch?v=CqOfi41LfDw)

**backpropagation**

[**https://www.youtube.com/watch?v=IN2XmBhILt4**](https://www.youtube.com/watch?v=IN2XmBhILt4)

**Activation function**

**https://www.youtube.com/watch?v=68BZ5f7P94E**

* <https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html>
* https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/\_downloads/4e865243430a47a00d551ca0579a6f6c/cifar10\_tutorial.ipynb#scrollTo=PP9km88QkiZp