Savitribai Phule Pune University, Pune

Final Year Information Technology (2019 Course)

414447: Lab Practice IV

Teaching Scheme:	Credit Scheme:	Examination Scheme:
Practical (PR):02 hrs/week	01 credits	PR: 25 Marks TW: 25 Marks

Prerequisites: Python programming language

Course Objectives:
The objective of the course is
1. To be able to formulate deep learning problems corresponding to different applications.
2. To be able to apply deep learning algorithms to solve problems of moderate complexity.
3. To apply the algorithms to a real-world problem, optimize the models learned and report on the expected accuracy that can be achieved by applying the models.

Course Outcomes:
On completion of the course, students will be able toCO1. Learn and Use various Deep Learning tools and packages.
CO2. Build and train a deep Neural Network models for use in various applications.
CO3. Apply Deep Learning techniques like CNN, RNN Auto encoders to solve real word Problems.
CO4. Evaluate the performance of the model build using Deep Learning.

Guidelines for Instructor's Manual
The faculty member should prepare the laboratory manual for all the experiments, and it should be made
available to students and laboratory instructor/assistant
Guidelines for Student's Lab Journal
1. Students should submit term work in the form of a handwritten journal based on a specified list of assignments.
2. Practical Examination will be based on the term work.
3. Candidate is expected to know the theory involved in the experiment.
4. The practical examination should be conducted if and only if the journal of the candidate is complete in all respects.

Guidelines for Lab /TW Assessment
1. Examiners will assess the term work based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Appropriate knowledge of usage of software and hardware related to the respective laboratory should be checked by the concerned faculty member.

Guidelines for Laboratory Conduction
As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers of the program in a journal may be avoided. There must be hand-written write-ups for every assignment in the journal. The DVD/CD containing student’s programs should be attached

to the journal by every student and the same to be maintained by the department/lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Practical Examination

1. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement.
2. Student's understanding of the fundamentals, effective and efficient implementation can be evaluated by asking relevant questions based on implementation of experiments he/she has carried out.

List of Laboratory Assignments

Mapping of course outcomes for Group A assignments: CO1, CO2,CO3,CO4

1. Study of Deep learning Packages: Tensorflow, Keras, Theano and PyTorch. Document the distinct features and functionality of the packages.

Note: Use a suitable dataset for the implementation of following assignments.

2. Implementing Feedforward neural networks with Keras and TensorFlow
    a. Import the necessary packages
    b. Load the training and testing data (MNIST/CIFAR10)
    c. Define the network architecture using Keras
    d. Train the model using SGD
    e. Evaluate the network
    f. Plot the training loss and accuracy
3. Build the Image classification model by dividing the model into following 4 stages:
    a. Loading and preprocessing the image data
    b. Defining the model's architecture
    c. Training the model
    d. Estimating the model's performance
4. Use Autoencoder to implement anomaly detection. Build the model by using:
    a. Import required libraries
    b. Upload / access the dataset
    c. Encoder converts it into latent representation
    d. Decoder networks convert it back to the original input
    e. Compile the models with Optimizer, Loss, and Evaluation Metrics
5. Implement the Continuous Bag of Words (CBOW) Model. Stages can be:
    a. Data preparation
    b. Generate training data
    c. Train model
    d. Output
6. Object detection using Transfer Learning of CNN architectures
    a. Load in a pre-trained CNN model trained on a large dataset
    b. Freeze parameters (weights) in model's lower convolutional layers
    c. Add custom classifier with several layers of trainable parameters to model
    d. Train classifier layers on training data available for task
    e. Fine-tune hyper parameters and unfreeze more layers as needed

Reference Books:

1. Hands-On Deep Learning Algorithms with Python: Master Deep Learning Algorithms with Extensive

Math by Implementing Them Using TensorFlow
2. Python Deep Learning, 2nd Edition by Ivan Vasilev , Daniel Slater , GianmarioSpacagna,, Peter Roelants,
Valentino Zocca
3. Natural Language Processing with Python Quick Start Guide by Mirant Kasliwal

Laboratory Assignment No. 1

Aim: Study of Deep learning Packages: Tensorflow, Keras, Theano and PyTorch. Document the distinct features and functionality of the packages.

Objective: Introduction to various deep learning tools and how to use them.

Software used: Windows/Linux/ Mac, TensorFlow, Keras, PyTorch

Theory:

Tensorflow:

What Is TensorFlow?

TensorFlow is an open source software library released in 2015 by Google to make it easier for developers to design, build, and train deep learning models. TensorFlow originated as an internal library that Google developers used to build models in-house, and we expect additional functionality to be added to the open source version as it is tested and vetted in the internal flavour.

On a high level, TensorFlow is a Python library that allows users to express arbitrary computation as a graph of data flows. Nodes in this graph represent mathematical operations, whereas edges represent data that is communicated from one node to another. Data in TensorFlow is represented as tensors, which are multidimensional arrays.

Installing TensorFlow

We use a Python package installation manager called Pip.

 # Ubuntu/Linux 64-bit
$ sudo apt-get install python-pip python-dev
 # Mac OS X
$ sudo easy_install pip

We can use the following commands to install TensorFlow. Note the difference in Pip package naming if we would like to install a GPU-enabled version of TensorFlow.

$ pip install --upgrade tensorflow # for Python 2.7
$ pip3 install --upgrade tensorflow # for Python 3.n
$ pip install --upgrade tensorflow-gpu # for Python 2.7 # and GPU
$ pip3 install --upgrade tensorflow-gpu # for Python 3.n # and GPU

If you installed the GPU-enabled version of TensorFlow, you'll also have to take a couple of additional steps. Specifically, you'll have to download the CUDA Toolkit 8.03 and the latest CUDNN Toolkit.
Install the CUDA Toolkit 7.0 into /usr/local/cuda. Then uncompress and copy the CUDNN files into the toolkit directory. Assuming the toolkit is installed in/usr/local/cuda, you can follow these instructions to accomplish this:
$ tar xvzf cudnn-version-os.tgz
$ sudo cp cudnn-version-os/cudnn.h /usr/local/cuda/include
$ sudo cp cudnn-version-os/libcudnn* /usr/local/cuda/lib64

You will also need to set the LD_LIBRARY_PATH and CUDA_HOME environment variables

to give TensorFlow access to your CUDA installation. Consider adding the commands below to your *~/.bash_profile*. These assume your CUDA installation is in */usr/local/cuda*:

export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/cuda/lib64"

export CUDA_HOME=/usr/local/cuda

Note that to see these changes appropriately reflected in your current terminal session, you'll have to run:

$ source ~/.bash_profile

Running TensorFlow:

```
import tensorflow as tf
deep_learning = tf.constant('Deep Learning')
session = tf.Session()
session.run(deep_learning)
a = tf.constant(2)
a = tf.constant(2)
multiply = tf.mul(a, b)
session.run(multiply)
```

Output:

'Deep Learning'

TensorFlow Operation

Category	Examples
Element-wise mathematical operations	`Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...`
Array operations	`Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...`
Matrix operations	`MatMul, MatrixInverse, MatrixDeterminant, ...`
Stateful operations	`Variable, Assign, AssignAdd, ...`
Neural network building blocks	`SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...`
Checkpointing operations	`Save, Restore`
Queue and synchronization operations	`Enqueue, Dequeue, MutexAcquire, MutexRelease, ...`
Control flow operations	`Merge, Switch, Enter, Leave, NextIteration`

Keras:

Keras is an open source deep learning framework for python. It has been developed by an artificial intelligence researcher at Google named **Francois Chollet**.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

Features

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features −

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

Benefits

Keras is highly powerful and dynamic framework and comes up with the following advantages −

- Larger community support.
- Easy to test.
- Keras neural networks are written in Python which makes things simpler.
- Keras supports both convolution and recurrent networks.
- Deep learning models are discrete components, so that, you can combine into many ways.

Keras Installation Steps

Step 1: Create virtual environment

Virtualenv is used to manage Python packages for different projects. It is always recommended to use a virtual environment while developing Python applications.

Windows

Windows user can use the below command,

```
py -m venv keras
```

Step 2: Activate the environment

This step will configure python and pip executables in your shell path.

Windows

Windows users move inside the “kerasenv” folder and type the below command,

```
.\env\Scripts\activate
```

Step 3: Python libraries

Keras depends on the following python libraries.

- Numpy
- Pandas
- Scikit-learn

- Matplotlib
- Scipy
- Seaborn

pip install numpy, pandas, scipy, matplotlib, scikit-learn, seaborn

Keras Installation Using Python

Now, install the Keras using same procedure as specified below –

```
pip install keras
```

Quit virtual environment

```
deactivate
```

PyTorch:

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs. PyTorch is an optimized Deep Learning tensor library based on Python and Torch and is mainly used for applications using GPUs and CPUs. PyTorch is favored over other Deep Learning frameworks like TensorFlow and Keras since it uses dynamic computation graphs and is completely Pythonic.

The two main features of PyTorch are:

- Tensor Computation (similar to NumPy) with strong GPU (Graphical Processing Unit) acceleration support
- Automatic Differentiation for creating and training deep neural networks

Common PyTorch Modules

In PyTorch, modules are used to represent neural networks.

Autograd

The autograd module is PyTorch's automatic differentiation engine that helps to compute the gradients in the forward pass in quick time. Autograd generates a directed acyclic graph where the leaves are the input tensors while the roots are the output tensors.

Optim

The Optim module is a package with pre-written algorithms for optimizers that can be used to build neural networks.

nn

The nn module includes various classes that help to build neural network models. All modules in PyTorch subclass the nn module.

Installation

- Install Python 3

- Install pip.
- Now we can install with the following options:
    - No CUDA: Download the PyTorch from the official site(No CUDA) and install it.
    - With CUDA: Download the PyTorch from the official site(With CUDA) and install it.

VERIFICATION

To ensure that PyTorch was installed correctly, we can verify the installation by running sample PyTorch code. Here we will construct a randomly initialized tensor.

From the command line, type:

```
python
```

then enter the following code:

```
import torch
x = torch.rand(5, 3)
print(x)
```

The output should be something similar to:

```
tensor([[0.3380, 0.3845, 0.3217],
        [0.8337, 0.9050, 0.2650],
        [0.2979, 0.7141, 0.9069],
        [0.1449, 0.1132, 0.1375],
        [0.4675, 0.3947, 0.1426]])
```

Conclusion: We learnt about the various deep learning training tools.

Laboratory Assignment No. 2

Aim: Implementing Feedforward neural networks with Keras and TensorFlow

a. Import the necessary packages
b. Load the training and testing data (MNIST/CIFAR10)
c. Define the network architecture using Keras
d. Train the model using SGD
e. Evaluate the network
f. Plot the training loss and accuracy

Objective: To learn how to develop a feedforward neural network and how to optimize it for better performance.

Infrastructure: Computer/ Laptop/ Virtual Machine

Software used: Jupyter Notebook/Google Colab, Tensorflow, Kearas

Theory:

What is Feed forward neural networks?

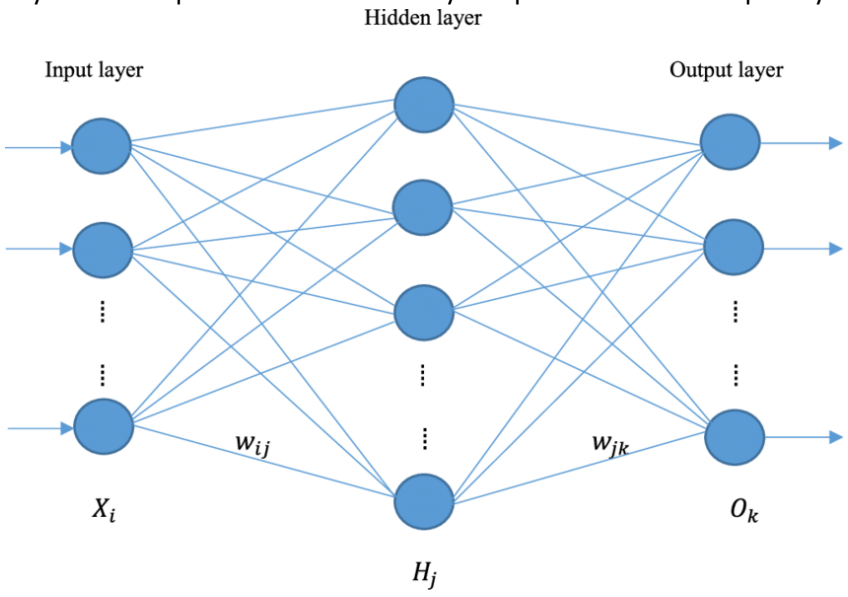
Deep feedforward networks, also often called feedforward neural networks, or multilayer perceptron's. The goal of a feedforward network is to approximate some function f ∗ . For example, for a classifier, y = f ∗(x) maps an input x to a category y. A feedforward network defines a mapping y= f(x; θ) and learns the value of the parameters θ that result in the best function approximation.

These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.

Structure of a feed forward Neural network

The basic structure consists of the following layers:

1. Input layer: It is where the user accepts the input for the neural network.
2. Hidden layer: This is the layer where all the computation required for the prediction are done.
3. Output layer: The output from the hidden layer is provided at the output layer.



The nodes are connected with the help of edges. The edges are represented as $W_{i,j}$ where i represent the node where the edge starts from and j represent the node where the edge ends.

The nodes compute the output for the next layer by summation of the product of the node input and the weight associated with the node edge, which is then applied to an activation function to decide whether the node should fire or not for the input.

SGD

Stochastic gradient descent (SGD) and its variants are probably the most used optimization algorithms for machine learning in general and for deep learning in particular. A crucial parameter for the SGD algorithm is the learning rate.

The standard gradient descent algorithm updates the parameters θ of the objective $J(\theta)$ as,

$$\theta=\theta-\alpha\nabla\theta E[J(\theta)]\theta=\theta-\alpha\nabla\theta E[J(\theta)]$$

where the expectation in the above equation is approximated by evaluating the cost and gradient over the full training set.

MNIST/CIFAR10

MNIST: The MNIST data set of handwritten digits has a training set of 70,000 examples and each row of the matrix corresponds to a 28 x 28 image. The unique values of the response variable *y* range from 0 to 9.

CIFAR10: CIFAR-10 is an established computer-vision dataset used for object recognition. The data I'll use in this example is a subset of an 80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10 object classes (6000 images per class). Furthermore, the data were converted from RGB to gray, normalized and rounded to 2 decimal places (to reduce the storage size).

Implementation:

1. Import the necessary libraries.
2. Load the dataset from the libraries or from outside.
3. Build the Feed forward neural network using Keras.
4. Train the model with the dataset and use SGD as optimizer.
5. Evaluate the model for the accuracy and other evaluation metrics.
6. Plot the loss and accuracy function.

Conclusion: We developed a feed forward neural network for hand written digit recognition.

Laboratory Assignment No. 3

Aim: Build the Image classification model by dividing the model into following 4 stages:

a. Loading and preprocessing the image data
b. Defining the model's architecture
c. Training the model
d. Estimating the model's performance

Objective: To learn about CNN and how to develop a CNN for image recognition.

Infrastructure: Computer/ Laptop/ Virtual Machine

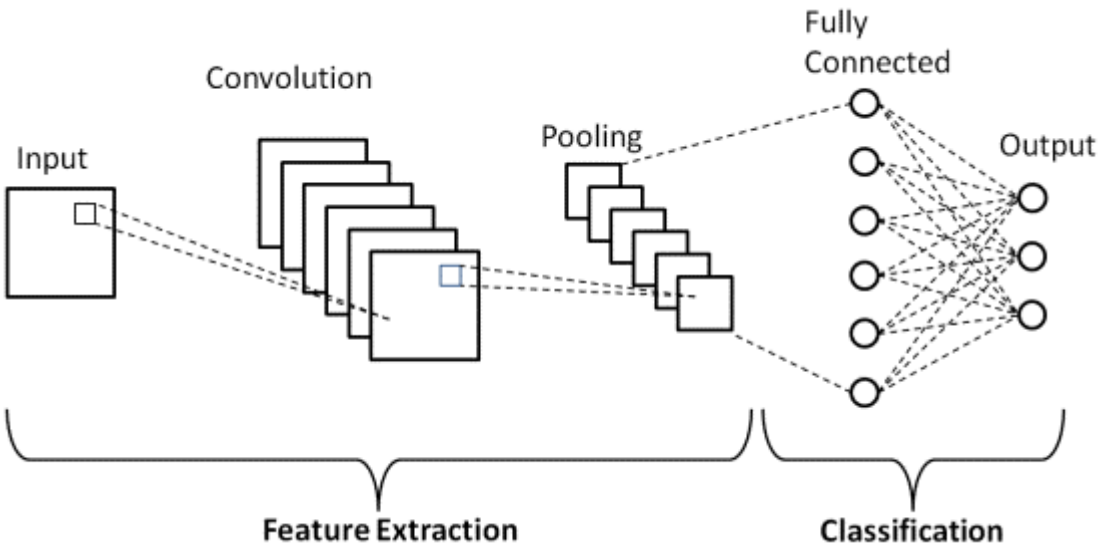
Software used: Jupyter Notebook/Google Colab, Tensorflow, Kearas

Theory:

CNN

Convolutional networks, also known as convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Architecture of CNN



There are two main parts to a CNN architecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction.
- The network of feature extraction consists of many pairs of convolutional or pooling layers.
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
- This CNN model of feature extraction aims to reduce the number of features present in a dataset. It creates new features which summarises the existing features contained in an original set of features.

Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers.
In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function.

1. Convolutional Layer
This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).
The output is termed as the Feature map which gives us information about the image such as the corners and edges.
2. Pooling Layer
The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling.
3. Fully Connected Layer
The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.
4. Dropout
Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.
5. Activation Functions
They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

Implementation:

1. Load the necessary libraries.
2. Import the dataset from the respective library or local dataset.
3. Design the neural network architecture and mention the number of layers, nodes, etc.
4. Then train the model with the imported dataset.
5. Evaluate the performance of the model.

Conclusion:
We learnt how to build and train a CNN to identify images.