# **Deep Learning Programming Assignment**

| **Ex. No.** | **Topic** |
| --- | --- |
| 1 | Feed Forward & Back-Propagation Learning Algorithm |
| 2 | ANN for MNIST digit Classification |
| 3 | CNN for MNIST digit Classification |
| 4 | ResNet-152 for Binary Classification of Skin Lesions |
| 5 | Autoencoder for dimensionality reduction |
| 6 | VAE for fashionMNIST (Synthetic Data Generation) |
| 7 | Deep fake generation using GAN |
| 8 | RNN+LSTM |
| 9 | UNet model for segmentation / YOLO for object detection |

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **Lab - 01 Feed Forward & Back-Propagation Learning Algorithm**

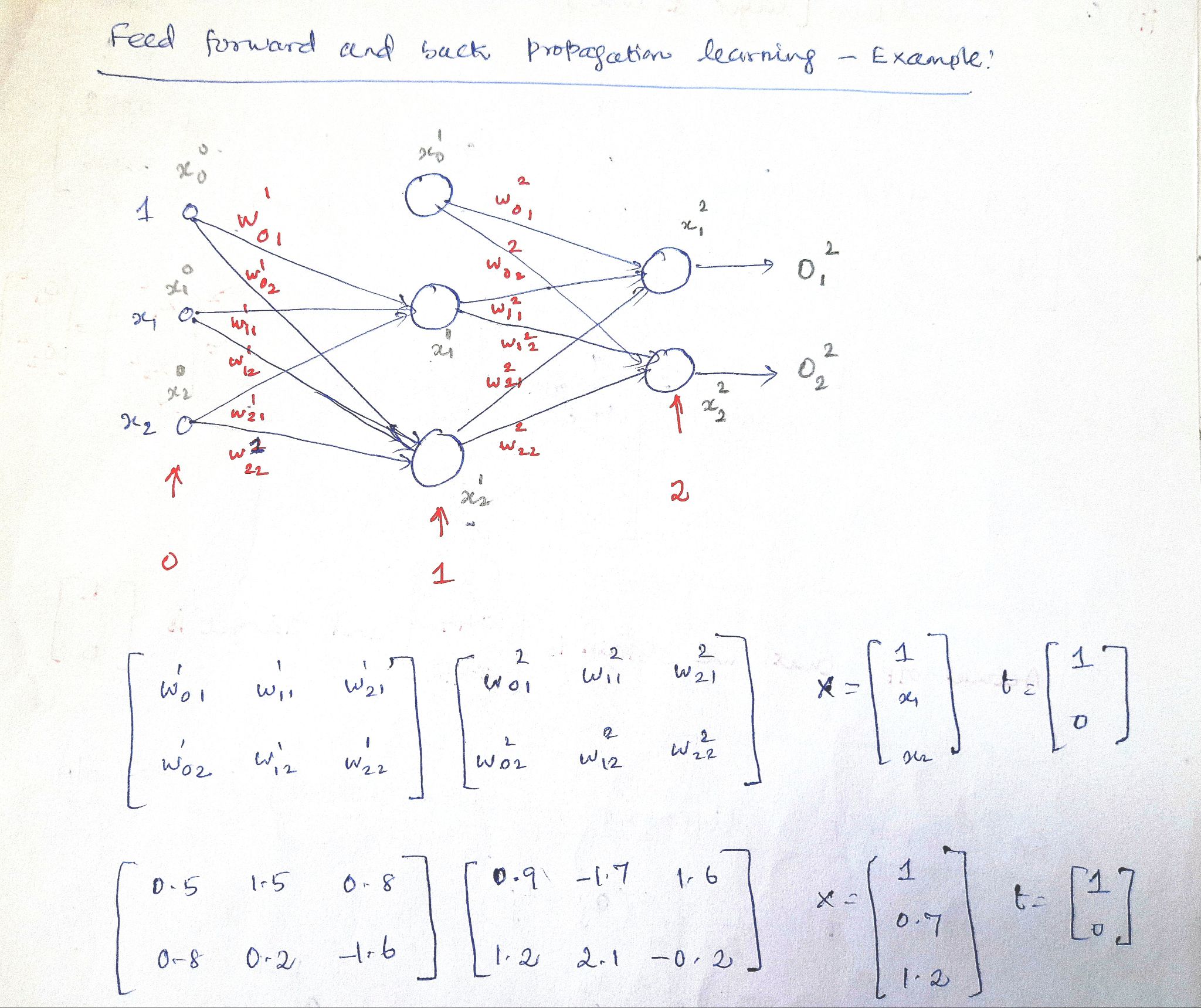
1. Implement the simple neural network algorithm from scratch in Python.

* Initialize the weights with [0 0 0] and a learning rate of 0.0001.
* For each iteration, calculate the output of the simple neural network for each input in the training set.
* Use MSE to computer the error for all samples
* Update the weights using the gradient descent procedure.
* Repeat the above steps until the simple neural network converges or a maximum number of iterations is reached.
* Test the trained simple neural network on a separate test set, explain how you came up with the test set.
* Use the step function as an activation function in the output layer and sigmoid function for other layers.

Use the IRIS Dataset for the above, considering all four features: sepal length, sepal width, petal length, and petal width, but only two classes - **Setosa, and Versicolor**. Drop the feature vectors of the other class.

Please find the dataset here - [Iris Dataset](https://docs.google.com/spreadsheets/d/1lbzA7pHBmqSoWA8qTvSugeTZbR-7OJwfl3XEIfZtomY/edit?usp=sharing)

1. Implement the feedforward and backpropagation learning algorithm for multi layer perceptrons in Python for the question provided in the attached image.
2. Use the weights and biases as given.
3. Implement the forward pass.
4. Compute the loss between the predicted output and the actual output using an appropriate loss function (MSE).
5. Compute the gradients of the loss function with respect to the weights and biases using the chain rule.
6. Update the weights and biases.
7. Iterate over multiple times (epochs), performing forward propagation, loss calculation, backpropagation, and parameter updates in each iteration till convergence (the actual output is the same as the target output).



## 

## **Lab - 02 ANN for MNIST digit Classification**

Q1. Classify MNIST digits using Fully Connected Neural network.

Dataset : download from internet source

* Plot few samples from dataset
* Train the network
* Test on the test dataset
* Calculate test accuracy on test set

## **Lab - 03 CNN for MNIST digit Classification**

Q1. Classify MNIST digits using CNN.

Dataset : download from keras/pytorch/standard source.

* Plot few samples from dataset
* Train the network
* Test on the test dataset
* Calculate test accuracy on test set

## **Lab - 04 Binary Classification of Skin Lesions**

1. In this lab assignment, implement the **ResNet-152(or ResNet 101)** model, a deep convolutional neural network, for classifying skin lesions into benign or malignant (binary classification).
   1. Use the dataset provided, write **a custom dataset class** to load the dataset properly.
   2. **Do not use prebuilt ResNet architecture** that comes with tensorflow/pytorch frameworks, instead **create architecture using basic layers and functions from the framework** like Conv2d, Linear, Relu, Pooling and others required.
   3. Print model parameters and architecture to verify it matches the standard one.
   4. **Experiment with different values of hyperparameters, describe the observation at each step of experimentation.** (*important* - must be documented in the submission file)
   5. After training the model, evaluate the model’s performance on the test dataset provided.
   6. Calculate classification metrics such as **accuracy, precision, recall, and F1-score.**

## **Lab - 05 AutoEncoders**

1. MNIST dataset compression and reconstruction using PCA and AutoEncoder.
2. Train a PCA model to compress the digits to 4 dimensional latent vectors.
3. Train an AutoEncoder model to compress the digits to 4 dimensional latent vectors.
4. Visualize the latent vectors using TSNE( <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>) for both the cases.
5. Show reconstruction for 20 latent samples using both models.
6. What is the compression ratio achieved?

## **Lab - 06 Variational AutoEncoders for Synthetic Data Generation**

1. Train a VAE model for generating new data similar to training set.
2. Train the model on fashionMNIST dataset.
3. Generate 30 new samples using the learned model and plot them.

Dataset: **fashionMNIST** <https://github.com/zalandoresearch/fashion-mnist?tab=readme-ov-file#get-the-data>

## **Lab - 07 Deep Fake Generation: GAN model on CelebA dataset**

1. Train a GAN model on celebrity faces dataset.
2. Use basic layers from the frameworks to write the discriminator and generator.
3. Understand the flow of data from input to output while training and testing.
4. Visualize samples from the dataset after loading the dataset.
5. After each epoch, plot a few samples to track the training progress.
6. Generate 50 samples from the trained model.

**Dataset: CelebA**

Use only images from the dataset to train the GAN model, resize the image to 64x64 for training to minimize computational requirements.

* **Main source of the dataset:**

<https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

* **PyTorch:** <https://pytorch.org/vision/stable/generated/torchvision.datasets.CelebA.html>
* **Tensorflow:**

<https://www.tensorflow.org/datasets/catalog/celeb_a>

## **Lab - 08 Sentiment Classification using RNN, LSTM**

1. Train a RNN based sentiment analysis model for classification of movie reviews.
2. Explore and learn about the different preprocessing steps in the Natural Language Processing(NLP) domain.
3. Apply suitable preprocessing steps for this sentiment analysis assignment.
4. Build and train a RNN model using basic layers from the framework.
5. Test model on the test set using suitable evaluation metrics.

2. Train a LSTM based model for the same sentiment analysis problem.

1. Build and train a LSTM model using basic layers from the framework.
2. Test model on the test set using suitable evaluation metrics.

**Dataset: Stanford Sentiment Treebank 2**

*Original dataset link:* [*https://huggingface.co/datasets/stanfordnlp/sst2*](https://huggingface.co/datasets/stanfordnlp/sst2)

**Dataset Zip Link:** [https://drive.google.com/file/d/1TytoIgt7KI9Ep9bo8bs\_X0HSSnBJX0oi/](https://drive.google.com/file/d/1TytoIgt7KI9Ep9bo8bs_X0HSSnBJX0oi/view?usp=sharing)

## **Lab - 09 Segmentation using U-Net**

Brain Tumor Segmentation using U-Net on LGG MRI Dataset

1. Train a Simple U-Net Model for Brain Tumor Segmentation

a. Explore and learn about different preprocessing steps in medical imaging, specifically for MRI scans.

b. Apply suitable preprocessing steps such as normalization, resizing, and data augmentation for this segmentation task.

c. Build and train a U-Net model using basic layers from the framework for brain tumor segmentation.

d. Test the model on the test set using suitable evaluation metrics such as Dice coefficient and IoU.

**Dataset: LGG MRI Dataset**

**Dataset Link** : <https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation>

**Dataset Zip Link:**

<https://drive.google.com/drive/folders/1F7pobjXSJk99EUtph3_gdq_b0ZgaG30Y?usp=sharing>