Ramakrishna Mission Vivekananda Educational & Research Institute
Belur Math, Howrah, West Bengal
School of Mathematical Sciences, Department of Computer Science

Assignment - 1 XOR Classification (Theory + Practice)
M.Sc. Computer Science and Big Data Analytics — Date: 8 Feb 2026
Course : **CS411: Applications of Computer Vision and Deep Learning**
Deadline: 23rd-Feb-2026, 11:59 P.M.
Instructor: Jimut Bahan Pal — Max marks: 75

Instructions: Read Carefully and Attempt All Questions

Submission Format:

- Complete all theory/proof questions in rough draft first, then write clean final versions
- Scan your handwritten work using a mobile scanner to create a single PDF (max 15 MB)
- Ensure the scan is clear and legible with good contrast between paper and ink
- Submit **one PDF** along with your **Jupyter notebook**
- Compress both files as **Name_ROLL.zip**

Submission Process:

- Email your submission to **jimutbahanpal@yahoo.com**
- Add **jpal.cs@gm.rkmvu.ac.in** as CC (Carbon Copy)
- Optionally CC your personal email to confirm delivery

Academic Integrity:
This assignment is designed to be completed without AI assistance. While you may use LLMs or agentic AI tools, you must:

- Fully understand any AI-generated code
- Be prepared to explain and justify your solutions during the viva
- Note: Inability to justify your work during the viva will result in negative marking

Deadline:
Submissions received after the deadline will incur a penalty of **-5 marks per day**. Plan accordingly and start early!

1. The multi-layer perceptron shown in Figure 1, has the following weight initializations: (27)

$$w_{11}^{(1)} = 1,\ w_{21}^{(1)} = 1,\ \text{and } w_{31}^{(1)} = -0.5.$$
$$w_{12}^{(1)} = 1,\ w_{22}^{(1)} = 1,\ \text{and } w_{32}^{(1)} = -1.5.$$

Say, we are feeding the x_1 and x_2 the values of the XOR function's input, and expecting the MLP to classify the XOR function correctly as shown in Table 1.

Say, the bottom neuron of the second layer is named as z_1 and the top neuron in the second layer is named as z_2, then do the following:

   1. Find out the activations for the z_1, z_2 and the output neurons. [3+3]

2. Find out what are the values of the weights of $w_{31}^{(2)}$, $w_{21}^{(2)}$, and $w_{11}^{(2)}$ [3+3+3]
3. Find the values of z_1 and z_2's output as well for all the combinations of the input. [3+3]
4. Draw the visualizations for each of the hyperplane for each of the layers, and see if the output is being separated using a linear hyperplane. [3+3]

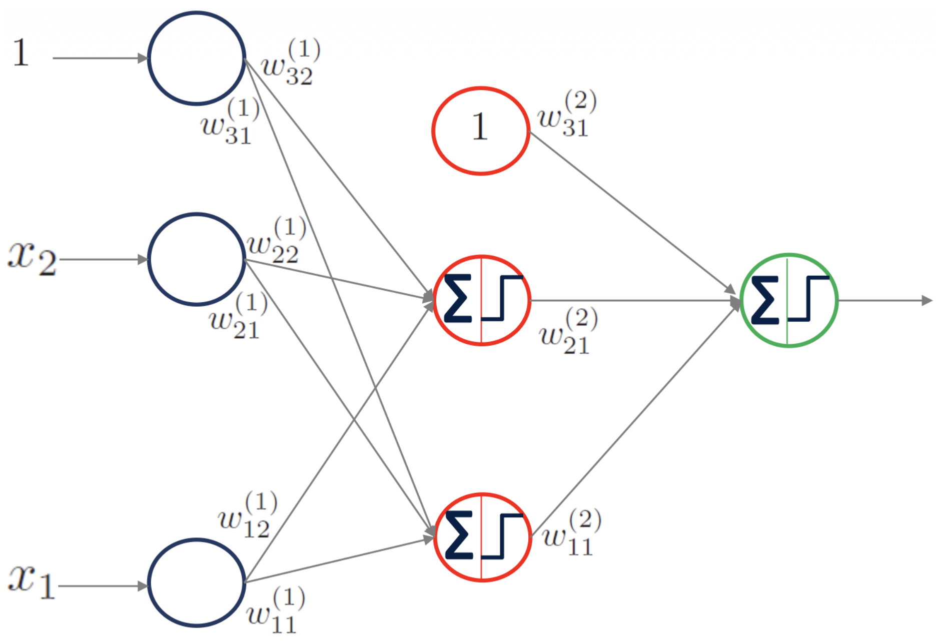

Figure 1: Multi Layer Perceptron (Credits: Dripta Mj's slides)

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: XOR Truth Table

2. **Implementation Guidelines:** (48)

   You must refer to the notebook **ACVDL_Lec_3_Pytorch_count_1gt0.ipynb** shared in class and use it as a skeleton for this assignment. Use Google Colab for faster and easier implementations.

   You may discuss the logic with your peers, but you must code individually. Do not copy another person's work. Any use of LLMs is prohibited unless you fully understand the generated code. Read the questions carefully and ask for clarification if needed. Failure to follow instructions may result in mark deduction.

   Tasks:

   1. Create the model, print the model, visualize using `torchsummary` and `graphviz` [4+1+2+2]
   2. Initialize the model weights as described in Q1 [4]
   3. Create an XOR dataset with minor Gaussian noise added to each point to generate multiple samples [4]
   4. Split the data into train (60%), validation (10%), and test (30%) sets [4]
   5. Plot the dataset points in a 2D scatter plot [2]
   6. Create a DataLoader for the input-output pairs [3]

7. Implement the training pipeline with the following functions: [2+2+2]
   (a) `train_epoch(train_loader, model, optimizer, epoch)`
   (b) `val_epoch(val_loader, model, optimizer, epoch)`
   (c) `test_epoch(test_loader, model, optimizer, epoch)`
8. Store training and validation losses. Track and plot the evolution of weights $w_{31}^{(2)}$, $w_{21}^{(2)}$, and $w_{11}^{(2)}$ across epochs. Freeze the initial weights for comparison. [3+3+3]
9. Implement a master training function which controls all the other functions:
   `def train_val_test(train_loader, val_loader, test_loader, model, optimizer, n_epochs, resume):` [5]
10. Save the final trained model [2]
11. Load the saved model and evaluate on the test dataset [2]
12. Plot the loss curves and weight evolution over 1000 epochs [3]
13. Display the confusion matrix showing model performance on the test dataset [3]
14. Create a function to load the trained model and evaluate on the test dataset, displaying the confusion matrix on demand [4]