**Ramakrishna Mission Vivekananda Educational & Research Institute**

Belur Math, Howrah, West Bengal

**School of Mathematical Sciences, Department of Computer Science**

Assignment - 1 (Theory + Practice)

M.Sc. Computer Science and Big Data Analytics Date: 11 Feb 2024

Course : **CS411: Applications of Computer Vision and Deep Learning**

Deadline: 3rd-March-2024, 11:59 P.M.

Instructor: Jimut Bahan Pal Max marks: 140

**Instructions**: **Attempt all the questions**; try to solve the theory/proof questions in rough first, and later correctly write it on a sheet of paper and make a PDF document using any mobile scanner. The document should be within 15 MB, and the writing should be visible, with clear contrast between the paper and the ink. **One single PDF should be submitted** alongside the **jupyter-notebook**. Submit the code and the PDF writings as **Name_ROLL.zip** and send it to the reply email on **jimutbahanpal@yahoo.com** AND **jpal.cs@gm.rkmvu.ac.in** as Carbon Copy (CC). You may also CC it to your other personal email address to check whether the email has gone to the address. **Submission after the deadline will fetch you -5 marks per day, so start early!**

1. (25)

```
x = torch.tensor([0.1, 0, 0.2, 1], dtype=torch.float32, requires_grad=True)
z = torch.tensor([-1, 1, -1, 0], dtype=torch.float32, requires_grad=True)
a = x + 10%4 + z
b = a**(4*x)
c = (b + 3)*5
y = c.mean()
y.backward()
```

You must compute the x.grad and z.grad (i.e., the gradient of the leaf variables) by hand. Please show all the workings in the pdf submitted. You need to explain what is happening by drawing diagrams of the computation tree, and you can check the code in pytorch too. Proper variables and calculations should be shown in the PDF and should be consistent with the output in Pytorch. **Please refer to ACVDL_Lec_2_Pytorch_Fundamentals.ipynb notebook for this question and gradient computation.**

2. (a) Please refer to the notes of Lecture-3 and Lecture-5 for this. See Figure a; you need to derive the change in weight, i.e., $\Delta w_{11}$'s equation. Please absorb any constant inside the learning rate. Derive to find $\Delta w_{11} = \eta(t_1 - o_1)o_1(1 - o_1)x_1$. **(30)** (35)

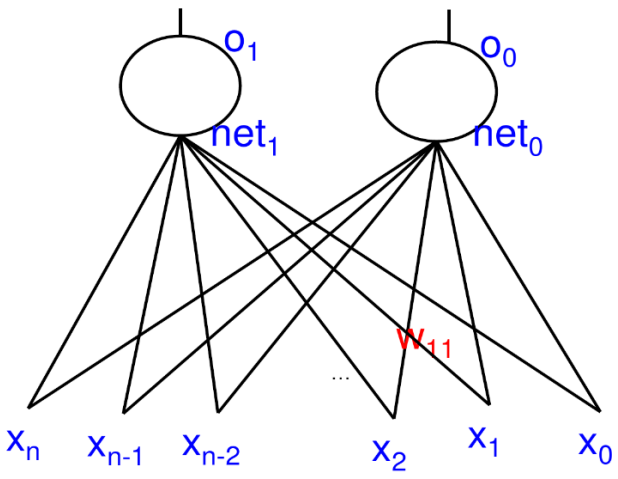


Figure 1: Figure of a neural network having sigmoid as activation.

Use the same set of procedures followed in the class to produce the final results. Here, the target vector is $<t_1,t_0>$and the observed vector is $<o_1,o_0>$. Use the total sum of squared loss, i.e, TSS = $\frac{1}{2}[(t_1 - o_1)^2 + (t_2 - o_2)^2]$

(b) Also, try to figure out all the the possibilities of vanishing gradient for this problem. **(5)**

3. This part of the assignment takes you through building a classification network pipeline using pytorch from scratch. You must use the Google Colaboratory platform or any other GPU compute resource and submit the notebook for this task. You can tune the model's parameters to run it in decent-sized images without further hurting its performance. Everything in this assignment is left to the doer's choice to tune and play with for a better learning experience. Investing at least 5-10 hours for this assignment is required. The notebook should be self-explanatory, and you must demo the assignment in class using a loaded model that you will train and save over time. The focus should be on visualizing the qualitative results and easy demo within 5-10 minutes. Please practice the demo 1-2 times before the final one. (80)

   Accessing the data inside Google Colab. You can access the data here: `https://drive.google.com/drive/u/2/folders/1Q_hzYMevVidcCHbW8SH-n3ztilmW3Z1B`. Or you can use this command as it is in colab.

```
! pip install --upgrade --no-cache-dir gdown
! gdown 1M8BY0yFrLX3liHUOHnBTes8HMa8507BS
! unzip -qq animals.zip
```

   This dataset has 3000 images uniformly distributed across the classes for easy implementation in the Google Colaboratory platform.

   The data is divided into 3 folders, with cats, dogs, and pandas as class labels, corresponding to 1000 images from individual classes. You must refer to the **ACVDL_Lec_3_Pytorch_count_1gt0.ipynb** notebook shared in class and use it as a skeleton for this assignment. You can refer to a friend and discuss the logic for this task, but you should code individually and not copy from another person's work. The current assignment is divided into the following parts:

   (a) Use a seed value of 42 for Numpy, random, and torch, and split the dataset into 70-10-20 images uniformly over all the classes separately. This will ensure that 700 images (i.e., 70%) of each class are used for training, 10% for validation, and 20% for testing. Please refer your friend to check if the images are consistent for individual splits. Please plot the first 5 images of train, validation, and test from each class (i.e., 45 images in total). Label the training set, validation set, and test set accordingly in the notebook while plotting. **(10)**

   (b) Extend the current data-loader to take in images and return the class value as one-hot-vector of size 3. **(5)**

   (c) Build a CNN model from scratch using the Pytorch torch.nn module and use proper convolutional, batch norm, ReLU, max pool, and sigmoid layers when necessary. You can tune the architecture, but no pre-trained model should be used for this task. The model's output should be of size 3, and a softmax activation should be required for this task. **(15)**

   (d) Use data augmentation (you can choose any existing libraries like albumentations; code that does data augmentation from scratch will receive extra credit). Visualize the augmentations according to a given input image example. Use at least 3 data augmentation for a given image. Use random augmentations. Use relevant input size. Standard input sizes are 256x256, 128x128, and 64x64, but please don't go below this since the images will become pixelated. You can check with the Google Colaboratory runtime to see which image is optimal and create a model according to that. See that each epoch is run within 3-4 minutes to have a decent number of epochs, at least 30, during training. **(10)**

   (e) Extend the current code by modifying the training pipeline to support the validation function. Check that there is no gradient update during this function's implementation. **(5)**

(f) Use relevant metrics for classification and dump the metrics for each epoch for training validation as separate text files. Plot the metrics at the end of the training. You may choose any number of epochs that give optimal results on the validation set. Accuracy, precision, and recall are good metrics to start with, but you can choose any other relevant metrics with proper reference and justification. Use proper optimizer and learning rate. Tune the hyper-parameters to get optimal results, and plot the graphs according to the hyper-parameters tuned. **(10)**

(g) Try to see the misclassified examples and come up with an explanation for why things are not working. (You might get extra credit if you find innovative techniques for visualizations.) **(5)**

(h) Report the test metrics across the untouched 200 images and display the first 10 images from each class. **(5)**

(i) Make a procedure to load the model and use it for the experiments in the test set. You must show the notebook's test set by loading the model in the fly. **(5)**

(j) Make confusion metrics and confusion-metrics-like-plots for the randomly picked images and their softmax scores as probability maps below each image. This will help us explain which cat image is classified as a cat, dog, or panda, which dog image is classified as a cat, dog, or panda, and vice versa. Please use proper plots for this by picking randomly from the test set and testing your model. **(10)**

(k) Students who use different loss functions other than the cross-entropy loss function, which achieves better results than before, will get extra credit. This also should be referred to as why you chose the particular loss function. **(bonus 10)**

---