# ECE 449 Machine Learning (ZJU-UIUC Institute)
# Assignment 2

**Due Date:** April 20, 2021.

## 1 Overview

Welcome to ECE449 assignment 2!

In this assignment, you will complete 4 Python programming tasks and answer 6 conceptual questions. Assignment 2 has 100 pts in total: 60 pts from programming tasks and 40 pts from conceptual questions.

## 2 Programming Questions

### 2.1 Programming environment update

In assignment 2, you need PyTorch when answering programming questions. You can install PyTorch in your ece449_hws Conda environment. We recommend the following steps:

1. Launch Anaconda Prompt, activate your environment by typing the following command in your Anaconda Prompt: conda activate ece449_hws

2. Check if your machine has CUDA compatible GPU devices. A quick verification is to type in thecommand: nvcc -V in your terminal. If you see version info from NVIDIA CUDA compiler driver(nvcc), then your machine has CUDA compatible GPU; if you see error message saying command not found, then your machine does not have a CUDA compatible environment (either because your CUDA driver installation is not working or you didn't install CUDA on your computer, or because your machine does not have CUDA compatible hardware).

3. If you have a working CUDA environment, please type in the following command in your terminal toinstall PyTorch, make sure that the cudatoolkit version in your command matches CUDA version shown in your machine from nvcc -V output(below is an example for CUDA 10.2):

```
conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch
```

If your machine does not support CUDA, please type in command shown below to install PyTorch:

```
conda install pytorch torchvision torchaudio cpuonly -c pytorch
```

For more info about PyTorch installation please visit: https://pytorch.org/.

Now you should have PyTorch installed on your machine. A quick verification is to launch a Python interpreter in your Anaconda Prompt by typing in python, then type in the following Python code:

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

If you see a torch tensor, you have verified that your PyTorch installation is working.

You can also verify if CUDA is available for your PyTorch installation by using the Python code below:

```
torch.cuda.is_available()
```

### 2.2 Programming questions brief descriptions

There are 4 Python script files you need to work on in this assignment.

The first Python script file, task1_template.py, is about implementing a simple and small multi-layer perceptron using Numpy. You will implement the following functions:

relu [3pts], prelu [2pts], DNNforward [5pts], DNNbackward [5pts], and train [5pts].

The second Python script file, task2_template.py, is about implementing a simple and small multi-layer perceptron using PyTorch. You will implement the following functions:

get_DNN [5pts] and train_torch [5pts].

The third Python script file, task3_template.py, is about implementing a famous Convolutional Neural Network, LeNet-5, using PyTorch. You will implement the following functions:

get_lenet [5pts] and train_cnn [15pts].

The fourth Python script file, task4_template.py, is about implementing another famous Convolutional Neural Network, ResNet-18, using PyTorch. You will implement the following functions and/or classes:

Residual [5pts] and get_resnet [5pts].

The programming part has 60 pts in total.

We have provided you the 4 Python script templates.

We have also provided you 4 Jupyter notebook files:

task1_run.ipynb, task2_run.ipynb, task3_run.ipynb, task4_run.ipynb.

You can use these 4 Jupyter notebook files to check the correctness of your implementations in the 4 tasks.

We recommend that you work in the 4 tasks in order. When you begin working on a task, please first launch the Jupyter notebook for that task and read the task instructions carefully, then open the .py script and write your implementations.

The 4 Jupyter notebooks are all set to active reloading mode, which means that once your implementation modifications are saved to your .py script file, your modifications will be immediately updated in the Jupyter notebook, which is very convenient for you to check and debug your code.

### 2.3 Autograder notice

We use autograder to grade your implementations, and here are some things we want to highlight:

1. Do NOT modify the names of any functions we grade. Autograder will give 0 for functions with namesmodified.
2. Do NOT modify the input interface of any functions, Autograder will give 0 for functions with inputinterfaces modified. However, in some cases you are allowed to modify the return values of some functions.
3. Do NOT import extra libraries. The Python libraries given in the .py scripts should be adequate foryou to obtain a correct answer.

### 2.4 Collaboration policy

You should not look at python code solutions from other students, the Python implementations should be your own work.

## 3 Conceptual Questions

There are 6 questions you need to answer in the conceptual question part. You can write down your answer on a few sheets of paper and scan them to a single PDF file, or you can use a digital pen to work on these questions

and export your handwritten answers to a single PDF file. We also accept answers in pdf format generated using LATEX.

## 3.1 Problem 1

Imagine that you are working on a big and complicated Convolutional Neural Network(CNN) with many convolutional building blocks shown on the right. The convolutional building block has three layers: a convolution layer with kernel $K \in \mathbb{R}^{k \times k}$, stride=1, and no padding, a sigmoid activation layer(with sigmoid function $f(x) = \frac{1}{1+e^{-x}}$), and a 2×2 average pooling layer. The building block input is $X \in \mathbb{R}^{(2n+k-1)\times(2m+k-1)}$, the output from convolution layer is $C \in \mathbb{R}^{2n \times 2m}$, the output from activation layer is $S \in \mathbb{R}^{2n \times 2m}$, and the building block output from the 2×2 average pooling layer is $P \in \mathbb{R}^{n \times m}$.

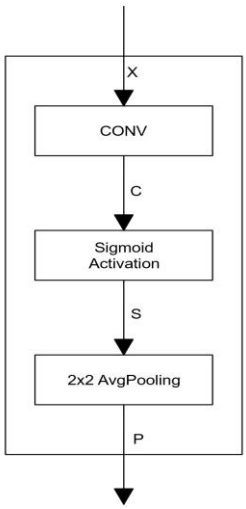


Figure 1: Convolutional Building Block For Problem 1

Given $X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,2m+k-1} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,2m+k-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2n+k-1,1} & x_{2n+k-1,2} & \cdots & x_{2n+k-1,2m+k-1} \end{bmatrix}$, $S = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,2m} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,2m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{2n,1} & s_{2n,2} & \cdots & s_{2n,2m} \end{bmatrix}$,

and $\frac{\partial loss}{\partial P} = \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,m} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,m} \end{bmatrix}$, write down the expressions for $\frac{\partial loss}{\partial S}$, $\frac{\partial loss}{\partial C}$, and $\frac{\partial loss}{\partial K}$, using $g_{i,j}$ for $1 \le i \le n$, $1 \le j \le m$; $s_{d,e}$ for $1 \le d \le 2n$, $1 \le e \le 2m$; and $x_{a,b}$ for $1 \le a \le (2n+k-1)$, $1 \le b \le (2m+k-1)$. [8 pts]

## 3.2 Problem 2

### 3.2.1 Problem 2.1

Please state at least two advantages of one-stage models compared with multi-stage models in object detection tasks. [3 pts]

### 3.2.2 Problem 2.2

Please state briefly why Non-maximum suppression(NMS) is generally needed for bounding-box based approaches in object detection tasks. [3 pts]

### 3.3 Problem 3

Imagine we have a Convolutional Neural Network with its structure shown below:

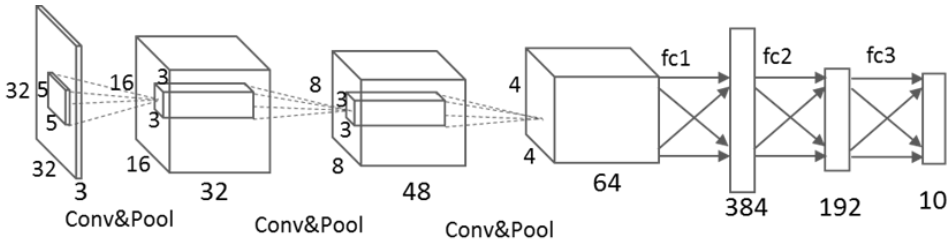


Figure 2: CNN For Problem 3

Please calculate the total number of parameters (including all weights and all biases, suppose biases are used wherever possible) of this CNN. [6 pts]

### 3.4 Problem 4

Suppose we have a neural network model with a softmax layer shown as below:

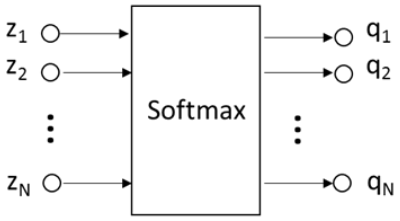


Figure 3: Softmax layer For Problem 4

Where $q_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$ for $1 \leq i \leq N$.

We want our softmax layer to learn a mapping from $Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}$ to $P = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}$,

where we have

$$\sum_{i=1}^{N} p_i = 1$$

Given the objective function

$$E = -\sum_{i=1}^{N} (p_i \times ln(q_i))$$

Please prove: $\frac{\partial E}{\partial z_x} = q_x - p_x$, for $1 \leq x \leq N$. [8 pts]

### 3.5 Problem 5

Suppose we have a simple neural network shown as below:

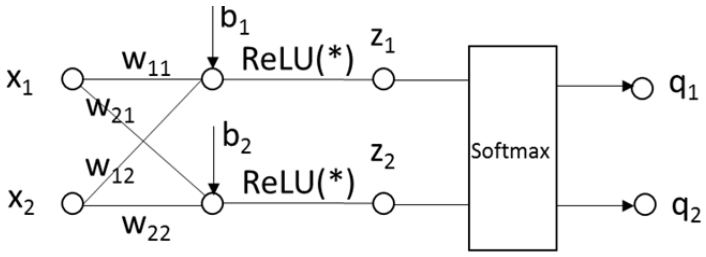


Figure 4: Neural Network For Problem 5

The initial parameters of this simple neural network model are:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} =, B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

We use batch size = 1, learning rate $\alpha = 0.02$, objective function $E = -p_1 ln(q_1) - p_2 ln(q_2)$, in the first training iteration our input sample ( $X, P)$ is: $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, P = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$.

If we use gradient decent formula

$$W' = W - \alpha \times \frac{\partial E}{\partial W}, \quad B' = B - \alpha \times \frac{\partial E}{\partial B}$$

during model training, please calculate the new parameters in our simple neural network $W$ and $B$ after the first iteration in training. [7 pts]

### 3.6 Problem6

In recurrent neural network, define loss function:

$$L = \sum_{t=0}^{T} L_t$$

Differentiate $L_t$ with respect to W:

$$\frac{\partial L_t}{\partial W^o} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial W^o} \quad \text{.....(1)}$$

$$\frac{\partial L_t}{\partial W^i} = \sum_{t=0}^{T} \sum_{k=0}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left( \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W^i} \quad \text{.....(2)}$$

$$\frac{\partial L_t}{\partial W^h} = \sum_{t=0}^{T} \sum_{k=0}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left( \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W^h} \quad \text{.....(3)}$$

Define:

$$h_t = \sigma(W^i x_t + W^h h_{t-1})$$

where σ denotes sigmoid activation function.

Based on equation (2) or (3), explain why there will be gradient exploding and gradient vanishing. [5pts]

## 4 Submission

You only need to submit 5 files to BlackBoard: the 4 Python script files with your programming implementations and a PDF file with your answers to the conceptual questions.

Pay attention that your 4 .py files should be renamed in the format of task[i]_[your_studentID].py, and your PDF file should be renamed in the format of conceptual_[your_studentID].pdf. For example, a student whose ID number is 3180100111 should submit the following 5 files:

task1_3180100111.py, task2_3180100111.py task3_3180100111.py,
task4_3180100111.py conceptual_3180100111.pdf

You can directly submit the five files to BlackBoard, or you can submit a zip file containing all the five files you need to submit.

If you want to submit a zip file, please name it as [your_studentID]_assignment2.zip. For example, a student whose ID number is 3170100111 should submit the zip file 3170100111_assignment2.zip to BlackBoard. Please make sure that you add the 5 files directly to your zip file, not a folder containing your 5 files. A quick verification for this is that if you unzip your zip file, the five files should appear in the same folder as your zip file. **Even if you submit a zip file, renaming the 5 files inside that zip file is required.** Otherwise our autograder will not be able to recognize the files you submit, which may even result in a 0 score for the file not properly renamed.