CSC6022 Homework 2

Homework Due Date: **Nov 12, 2025**

Instructions (Please read carefully)

- Submit your answers as an electronic copy only in **pdf** format on Blackboard.
- No late submissions will be accepted. Zero credit will be assigned for late submissions. Email request for late submission will not be replied.
- Please type in Latex or provide handwritten submissions scanned to **pdf**.
- Explicitly mention your collaborators if any. Collaborations should be limited to discussing and learning from each other but please do your own work and write your own codes. We will actively monitor any attempt to copy solutions from each other or from the internet.
- The full score of this homework is 150 pts.

1 Backpropagation for a MLP [15 pts]

Consider the following classification MLP with one hidden layer:

$$
\begin{aligned}
\boldsymbol{x} &= \text{ input } \in \mathbb{R}^D \\
\boldsymbol{z} &= \mathbf{W}\boldsymbol{x} + \boldsymbol{b}_1 \in \mathbb{R}^K \\
\boldsymbol{h} &= \text{ReLU}(\boldsymbol{z}) \in \mathbb{R}^K \\
\boldsymbol{a} &= \mathbf{V}\boldsymbol{h} + \boldsymbol{b}_2 \in \mathbb{R}^C \\
\mathcal{L} &= \text{ CrossEntropy } (\boldsymbol{y}, \mathcal{S}(\boldsymbol{a})) \in \mathbb{R}
\end{aligned}
$$

where $\boldsymbol{x} \in \mathbb{R}^D, \boldsymbol{b}_1 \in \mathbb{R}^K, \mathbf{W} \in \mathbb{R}^{K \times D}, \boldsymbol{b}_2 \in \mathbb{R}^C, \mathbf{V} \in \mathbb{R}^{C \times K}$, where D is the size of the input, K is the number of hidden units, and C is the number of classes. Show that the gradients for the parameters and input are as follows:

$$
\begin{aligned}
\nabla_{\mathbf{V}} \mathcal{L} &= \left[\frac{\partial \mathcal{L}}{\partial \mathbf{V}}\right]_{1,:} = \boldsymbol{u}_2 \boldsymbol{h}^\top \in \mathbb{R}^{C \times K} \\
\nabla_{\boldsymbol{b}_2} \mathcal{L} &= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_2}\right)^\top = \boldsymbol{u}_2 \in \mathbb{R}^C \\
\nabla_{\mathbf{w}} \mathcal{L} &= \left[\frac{\partial \mathcal{L}}{\partial \mathbf{W}}\right]_{1,:} = \boldsymbol{u}_1 \boldsymbol{x}^\top \in \mathbb{R}^{K \times D} \\
\nabla_{\boldsymbol{b}_1} \mathcal{L} &= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_1}\right)^\top = \boldsymbol{u}_1 \in \mathbb{R}^K \\
\nabla_{\boldsymbol{x}} \mathcal{L} &= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}\right)^\top = \mathbf{W}^\top \boldsymbol{u}_1 \in \mathbb{R}^D
\end{aligned}
$$

where the gradients of the loss wrt the two layers (logit and hidden) are given by the following:

$$\boldsymbol{u}_2 = \nabla_{\boldsymbol{a}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{a}}\right)^\top = (\boldsymbol{p} - \boldsymbol{y}) \in \mathbb{R}^C$$
$$\boldsymbol{u}_1 = \nabla_{\boldsymbol{z}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}}\right)^\top = \left(\mathbf{V}^\top \boldsymbol{u}_2\right) \odot H(\boldsymbol{z}) \in \mathbb{R}^K$$

with H is the Heaviside function. Note that, in our notation, the gradient (which has the same shape as the variable with respect to which we differentiate) is equal to the Jacobian's transpose when the variable is a vector and to the first slice of the Jacobian when the variable is a matrix.

2 Gradient derivation for a two-layer neural network [15 pts]

Consider a two-layer neural network N with the following forward propagation steps:

$$E(\mathbf{o}, \hat{\mathbf{o}}) = -\sum_{i=1}^{K} o_i \log \hat{o}_i \quad \text{(cross-entropy loss)}$$
$$\hat{\mathbf{o}} = \text{softmax}(\mathbf{z}_2), \quad \text{where } \mathbf{z}_2 = \mathbf{W}_2 \mathbf{h} + \mathbf{b}_2$$
$$\mathbf{h} = \text{ReLU}(\mathbf{z}_1), \quad \text{where } \mathbf{z}_1 = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

Here,
- $\mathbf{x} \in \mathbb{R}^n$ is the input,
- $\mathbf{o} \in \mathbb{R}^K$ is the one-hot encoded target output,
- $\hat{\mathbf{o}} \in \mathbb{R}^K$ is the predicted probability output,
- $\mathbf{W}_1 \in \mathbb{R}^{d \times n}$, $\mathbf{b}_1 \in \mathbb{R}^d$ are parameters of the first layer,
- $\mathbf{W}_2 \in \mathbb{R}^{K \times d}$, $\mathbf{b}_2 \in \mathbb{R}^K$ are parameters of the second layer.

The parameter set of the model is:

$$\Theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}.$$

The ReLU function is applied element-wise:

$$\text{ReLU}(z) = \max(0, z), \quad \text{and we define } \mathrm{I}\{z > 0\} \text{ to be the indicator function applied elementwise.}$$

(a) Compute the gradient of the loss E with respect to the output layer pre-activation $\mathbf{z}_2$.

(b) Using your result in (a), derive the gradients with respect to the output layer weights and bias, i.e. $\frac{\partial E}{\partial \mathbf{W}_2}$ and $\frac{\partial E}{\partial \mathbf{b}_2}$.

(c) Backpropagate the error to the hidden layer by computing $\frac{\partial E}{\partial \mathbf{h}}$ and $\frac{\partial E}{\partial \mathbf{z}_1}$.

(d) Finally, derive the gradients with respect to the input layer weights and bias, i.e. $\frac{\partial E}{\partial \mathbf{W}_1}$ and $\frac{\partial E}{\partial \mathbf{b}_1}$.

3 Convolutional Neural Network [15 pts]

(a) Consider the convolutional network defined by the layers in the left column below. Fill in the size of the activation volumes at each layer, and the number of parameters at each layer.

You can write your answer as a multiplication (e.g. $128 \times 128 \times 3$).

- CONV5-N denotes a convolutional layer with N filters, each of size $5 \times 5 \times D$, where D is the depth of the activation volume at the previous layer. Padding is 2, and stride is 1.
- POOL2 denotes a 2×2 max-pooling layer with stride 2 (pad 0).
- FC-N denotes a fully-connected layer with N output neurons.

Please fill in the following table.

Layer	Activation Volume Dimensions (memory)	Number of parameters
INPUT	$32 \times 32 \times 1$	0
CONV5-10		
POOL2		
CONV5-10		
POOL2		
FC-10		

(b) Give a reason why one would use a 1×1 convolution.

4 CNNs and Vision Transformers (ViTs) [10 pts]

In class, we have compared the differences between CNNs and ViTs in dealing with image data, focusing on their distinct inductive biases and how they impact performance on tasks such as image classification.

1. What are the two main operations used in CNNs to process images? (Single Choice)
   A. Matrix multiplication and addition
   B. Convolution and pooling
   C. Self-attention and layer normalization
   D. Gradient descent and backpropagation

2. Why do Vision Transformers (ViTs) need positional encodings? (Single Choice)
   A. To add color information to patches
   B. To remember where each image patch came from
   C. To reduce computation time
   D. To make the model smaller

3. Which architecture typically works better with small datasets? (Single Choice)
   A. Vision Transformer
   B. CNN
   C. Both work equally well
   D. Neither works well

4. Patch Processing: An image of size 224×224 is divided into 16×16 patches for a Vision Transformer. 1) How many patches will there be? 2) If each patch is flattened into a vector, what will be its length for an RGB image?

5. For each feature below, indicate whether it applies to:
   (C) CNNs only
   (V) Vision Transformers only
   (B) Both architectures
   (N) Neither architectures

   1. _____ Processes image patches in sequential order
   2. _____ Can handle patches in parallel fashion
   3. _____ Uses weight sharing across spatial locations
   4. _____ Requires positional encoding of patches
   5. _____ Naturally preserves local spatial relationships
   6. _____ Naturally preserves global spatial relationships

5 Using Backpropagation for GANs [20 pts]

In this question, we will work out backpropagation with Generative Adversarial Networks (GANs). (Hint: You don't need to understand what is GAN; you only need to use the following definition to compute the gradient using backpropagation.)
GAN: GAN is a generative model that consists of a Generator and a Discriminator playing a game. The Generator takes some noise (e.g., Gaussian) as input, and its goal is to produce something similar to a target distribution (from which we observe the training samples and name the training samples as true datasets). The Discriminator takes as input a batch consisting of a mix of samples from the true dataset and the Generator's output, and its goal is to correctly classify whether its input comes from the true dataset or the Generator.

Definitions:

- $X^1, \ldots, X^n$ is a minibatch of n samples from the target data generating distribution. For this question, we suppose that each X^i is a k dimensional vector. For example, we might be interested in generating a synthetic dataset of customer feature vectors in a credit scoring application
- $Z^1, \ldots, Z^n$ is a minibatch of n samples from some predetermined noise distribution.
- The generator $G(.;\theta_g) : \mathcal{Z} \to \mathcal{X}$ is a neural network.
- The discriminator $D(.;\theta_d) : \mathcal{X} \to (0,1)$ is a neural network.

Consider an objective function as follows:

$$L(\theta_d, \theta_g) = \frac{1}{n} \sum_{i=1}^{n} \log D\left(X^i\right) + \log\left(1 - D\left(G\left(Z^i\right)\right)\right). \tag{1}$$

The training of such a GAN system proceeds as follows: given the generator's parameters, the discriminator is optimized to maximize the above objective Eq. (1). Then, given the discriminator's parameters, the generator is optimized to minimize the above objective Eq. (1). This process is iteratively repeated. Once training is completed, we only require the generator to generate samples that are similar to the distribution of interest: we sample a point from our noise distribution and map it to a sample using our generator.

The Discriminator (The generator architecture is defined analogously)

- Consider the discriminator to be a network with layers indexed by $1, 2, .., L_d$ for a total of L_d layers.
- Let the discriminator's weight matrix for layer l be W_d^l; let us assume there are no biases for simplicity
- Let the activations produced by a layer l be given by A_d^l, and the pre-activation values by z_d^l.
- Let $g_d^l(.)$ be the activation function at layer l

1. Given that the discriminator is a binary classifier, please verify why the goal of the discriminator is to maximize the above objective Eq. (1).
2. Write down $\nabla_{\theta_d} L(X; \theta_d, \theta_g)$ in terms of $\nabla_{\theta_d} D(.)$
3. Write down
$$\frac{\partial L(\theta_d, \theta_g)}{\partial z_d^{L_d}}$$
taking help from your answer in the previous subpart. Remember that the activation function in the last layer of the discriminator is a sigmoid function as the output of the discriminator $\in (0,1)$.

4. Write down recursively
$$\frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^l}$$
in terms of $\frac{\partial L(\theta_d, \theta_g)}{\partial z_d^{l+1}}$. Hint: Your answer will contain $w_d^{l+1}, g_d'^l(.)$ and z_d^l.

5. Let the output of the generator be $g\left(z_i; \theta_g\right)$. Write down
$$\frac{\partial L\left(\theta_d, \theta_g\right)}{\partial g\left(z_i; \theta_g\right)}$$
in terms of w_d^1, and $\frac{\partial L(\theta_d, \theta_g)}{\partial z_d^1}$.

6. Now we move to the generator. The goal of the generator is to minimize the above objective function Eq. (1). Write down $\nabla_{\theta_g} L\left(\theta_d, \theta_g\right)$ in terms of $\nabla_{\theta_g} g(.)$ and in terms of $\frac{\partial L(\theta_d, \theta_g)}{\partial g(z_i; \theta_g)}$ calculated in the previous part.

7. Write down a simple mini-batch gradient descent update rule for θ_d^{t+1} in terms of $\nabla_{\theta_d} L\left(\theta_d, \theta_g\right)$, fixed learning rate α and the current parameters θ_d^t

8. Write down a simple mini-batch gradient descent update rule for θ_g^{t+1} in terms of $\nabla_{\theta_g} L\left(\theta_d, \theta_g\right)$, fixed learning rate α and the current parameters θ_g^t

6 Programming I [45 pts]

6.1 MLP [15 pts]

1. Write a program to fit a single hidden layer neural network (ten hidden units) via back-propagation and weight decay.

2. Apply it to 100 observations from the model
$$Y = \sigma\left(a_1^T X\right) + \left(a_2^T X\right)^2 + 0.30 \cdot Z,$$
where σ is the sigmoid function, Z is standard normal, $X^T = (X_1, X_2)$, each X_j being independent standard normal, and $a_1 = (3, 3), a_2 = (3, -3)$. Generate a test sample of size 1000, and plot the training and test error curves as a function of the number of training epochs, for different values of the weight decay parameter. Discuss the overfitting behavior in each case.

3. Vary the number of hidden units in the network, from 1 up to 10 , and determine the minimum number needed to perform well for this task.

6.2 CNN [30 pts]

Goals In this question you will practice writing codes and understand the architecture of Convolutional Neural Networks and get practice with training them. You will also gain experience with a major deep learning framework, such as **TensorFlow** or **PyTorch**. We specially thank Stanford 231n as the source of this question.

For this assignment, you can choose to use either PyTorch of Tensorflow and complete the corresponding prepared Jupyter notebooks that are given to you.

6.2.1 Download CIFAR10 Data

Please download the CIFAR 10 dataset from the website:
`https://www.cs.toronto.edu/~kriz/cifar.html`

6.2.2 Step-by-step instructions

You can find the following content from your given Jupyter notebooks. Please follow the instructions and complete the functions.

6.2.3 Requirement of your submissions

Please submit all the .py file in a zip file.

7 Programming of Attention Mechanism [30 pts]

Goals In this question you will practice writing codes and understand the architecture of Transformer Model. We specially thank Harvard NLP group as the source of this question.

For this assignment, you should complete the corresponding prepared Jupyter notebooks that are given to you by writing code of attention mechanism yourself.

7.1 Requirement of your submissions

Please submit all the .ipynb file in a zip file.