ECE 6524 – Deep Learning – Spring 2024
No. MIDTERM-6524

ECE 6524 – Deep Learning

Midterm Exam

Name ______________________________________

Date ______________________________________

- This is a take-home exam. **Due midnight Sunday, 3/17/2024 (NO EXTENSION).**
- There are THREE problems in the exam. Should you have any questions, please contact me at xuan@vt.edu.
- Adherence to the Honor Code is expected in the exam:

  "As a Hokie, I will conduct myself with honor and integrity at all times. I will not lie, cheat, or steal, nor will I accept the actions of those who do."

ECE 6524 (Deep Learning) - Midterm Exam

Problem 1. Machine Learning Basics (40 points)

Part A (15 points). Assuming the loss function $L(\mathbf{w})$ is a **quadratic** function, a simple gradient descent algorithm updates the weights ($\mathbf{w}$) with a learning rate η as follows:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}.$$

If $\mathbf{w}$ is one-dimensional, what is the optimal learning rate, η_{opt}, defined as being the learning rate moving the loss ($L(\mathbf{w})$) to its minimum in precisely one step? What is the largest learning rate that can be used without causing divergence?

(Show your work in detail; *Hint: using Taylor expansion to represent the loss function.*)

Part B (15 points)**.** One of the simplest and most common kinds of parameter norm penalty is the L_2 parameter norm penalty commonly known as **weight decay**. This regularization strategy drives the weights closer to the origin by adding a regularization term $\Omega(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ to the loss function ($L(\mathbf{w})$). The loss function $L(\mathbf{w})$ is assumed to be a **quadratic** function.

(1) *Mathematically prove* that updating $\mathbf{w}$ using the stochastic gradient descent (SGD) method is a weight decay process;

(2) *Mathematically prove* that the solution to the regularized version of $L(\mathbf{w})$ (i.e., $\mathbf{w}' = \text{argmin}_{\mathbf{w}}(L(\mathbf{w}) + \alpha * \Omega(\mathbf{w}))$ is a rescaled version of the optimum solution $\mathbf{w}^* = \text{argmin}_{\mathbf{w}}(L(\mathbf{w}))$. Specifically, the component of $\mathbf{w}^*$ that is aligned with the *i*-th eigenvector of $\mathbf{H}$ is rescaled by a factor of $\frac{\lambda_i}{\lambda_i + \alpha}$. $\mathbf{H}$ is the Hessian matrix of $L(\mathbf{w})$ with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$ and λ_i is the *i*-th eigenvalue of $\mathbf{H}$.

(Show your work in detail; *Hint: using Taylor expansion to represent the loss function.*)

Part C (10 points). Mathematically prove that the **Kaiming He Weight Initialization** scheme [**Ref. 1**] for neural networks with the ReLU activation function is: $w \sim N(0, 2/n_l)$, where n_l is the number of inputs in layer l. (Show your work in detail)

[Ref. 1] K. He, X. Zhang, S. Ren and J. Sun: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification (2015). (Published as a conference paper at the International Conference on Computer Vision (ICCV), 2015.) (https://arxiv.org/abs/1502.01852)

Problem 2. Multilayer Perceptrons (MLPs) (30 points)

Part A (10 points)**.** Describe and explain **in detail** the backpropagation algorithm using a ***computational graph*** approach. Specifically, explain (1) how downstream gradients are calculated from upstream gradients systematically in the backpropagation algorithm, and (2) what patterns in computational graphs can be used to facilitate our understanding of the backpropagation algorithm for multilayer perceptrons (MLPs)?

Part B (20 points)**.** A simple multilayer perceptron (MLP) neural network is given in **Fig. 1**. The activation function is the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$); the loss function ($L$) is defined by $L = \frac{1}{2}\sum_{i=1}^{N}(d_i - o_i)^2$, where the target values are d_i, $i = 1, \ldots, N$. (1) Draw the **backward computational graph** (in the backward/reverse direction, i.e., from the output to the input) for computing gradients with the backpropagation algorithm; (2) following the backpropagation algorithm, compute the following gradients: $\frac{\partial L}{\partial o_i}, \frac{\partial L}{\partial w_{ij}^{(3)}}, \frac{\partial L}{\partial h_i}, \frac{\partial L}{\partial z_i}, \frac{\partial L}{\partial w_{ij}^{(2)}}, \frac{\partial L}{\partial g_i}, \frac{\partial L}{\partial y_i}, \frac{\partial L}{\partial w_{ij}^{(1)}}, i = 1, \ldots, N; j = 1, \ldots, N$.

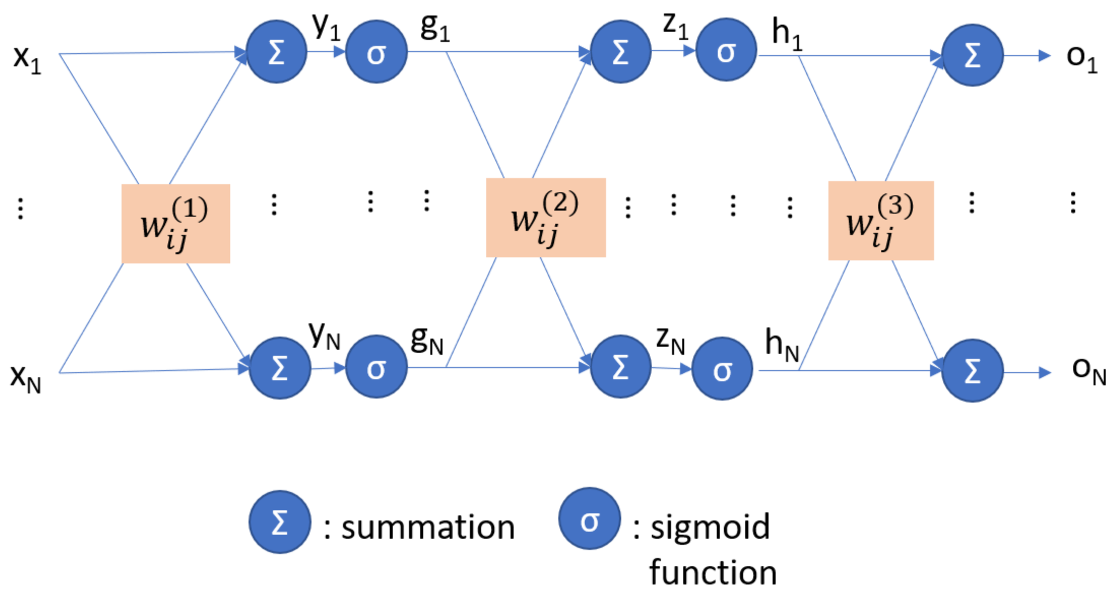

Fig. 1. A simple multilayer perceptron (MLP) neural network. Note that $w_{ij}^{(k)} (k = 1,2,3; i = 1, \ldots, N; j = 1, \ldots, N)$ are weights.

Problem 3. Convolutional Neural Networks (30 points)

CNNs for sports video classification

Please read a tutorial on using Keras to train a CNN for binary classification of cats and dogs: "Cats and Dogs Image Classification using Keras" (https://pythonistaplanet.com/image-classification-using-deep-learning/). Specifically, you need to understand some basics on building a simple CNN using Keras and how to use Keras *ImageDataGenerator* to generate training, validation and test data.

Part A (20 points). You are asked to design, implement, optimize and evaluate a convolutional neural networks (CNN) to classify sports videos. The sports video dataset consists of five types of sports events in particular: baseball, basketball, boxing, football and volleyball, which can be downloaded from the Canvas system for training. Several testing images and video clips are also provided for you to test your optimized CNN.

A simple frame-by-frame approach to video classification: videos can be understood as a series of individual images; in this problem we will simply treat video classification as performing image classification a total of *N* times, where *N* is the total number of frames in a video. To extract the image frames from a video, you may use the following (skeleton) code example with cv2 (Computer Vision Library):

```
import cv2
cap = cv2.VideoCapture('test_video_filename.mp4')
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret != True:
      break
     # Process frame from here
    ...
```

Part B (10 points) – Multiresolution CNNs. Implement, optimize and evaluate a multiresolution CCN as outlined in **Ref. 2** to classify the above-mentioned 5-type sports videos. (*hint:* for cropping and resizing an image, you may use the Cropping2D() and Resizing() layers in Keras/TensorFlow.)

For both Part A and Part B, report the architecture of the CNNs (your optimized CNNs), the performance of the CNNs (including training, validation and testing performances), and the convergence of the CNNs.

Write a report to present your results and discussions, and submit your implementation codes in separate files.

[Ref. 2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 1725-1732, doi: 10.1109/CVPR.2014.223.

END of the Midterm Exam