UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 324 — Introduction to Machine Intelligence
Midterm Examination
November 20, 2019
2:15 pm – 3:45 pm
(90 minutes)

Examiner: J. Rose

This is a "closed book" examination; no aids are permitted.

No calculators or other electronic devices are allowed.

All questions are to be answered on the examination paper. If the space provided for a question is insufficient, you may use the last page to complete your answer. If you use the last page, please direct the marker to that page and indicate clearly on that page which question(s) you are answering there. It is acceptable to use a pencil.

The grades associated with each question are given in square brackets next to each question number, and for portions of questions.

The examination has 20 pages, including this one.

First Name: _____ Last Name: _____

Student Number: _____

MARKS

1	2	3	4	5	6	7	8	9	Total
/8	/9	/8	/10	/9	/7	/6	/9	/8	/74

Question 1 [8 Marks]

(a) [2] Describe, in words, what it means when a neural network **fails** to generalize.

Please write your answer here:

(b) [4] Consider the following scenario: you've trained a neural network **binary classifier** with training data, and tuned its hyperparameters successfully with validation data. After this you discover that the network *failed to generalize* on the *test data* compared to results from the *validation data*. You then do a new training run which measures the accuracy of all three datasets (training, validation and test) at the end of every epoch, through 100 epochs of training, using the same final tuned hyperparameters. You are to draw a plot with three curves – the training, validation and test accuracy versus epoch – which would illustrate the behavior described. Be sure to label the axes (including the scale numbers) and curves.

Please draw your answer here:

Question 1 continues:

(c) [2] Give one possible reason why the failure (from validation to test) described in (b) might have occurred.

Please write your answer here:

Question 2 [9 Marks]

(a) [4] Consider a neural network that produces a single numerical output, z. During training of the network, as you know, a *loss function* is used that takes as its input z, and the label of a training data sample, L. Consider the following general form of loss function:

$$Loss = (z - L)^n$$

where n is an integer. What restrictions must be placed on n for this function to work properly as a loss function in the context of neural net training? Explain your answer.

(b) [3]

i) Why is data that is processed by neural networks often normalized?

ii) Will a neural network still succeed in training even when its inputs are not normalized? Why or why not?

Question 2 continues:

(c) [2] Consider a neural net which predicts, using data gathered from a person, whether or not that person has a life-threatening disease. A positive result from the neural net classifier means that the person is predicted to have the disease. There are four kinds of outcomes based on such a prediction - a false positive (FP), a false negative (FN), a true positive (TP) and a true negative (TN). Recall that, in class we defined several metrics relating to these outcomes, as follows (where 'FP' means the number of false positives, etc.):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Sensitivity/Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

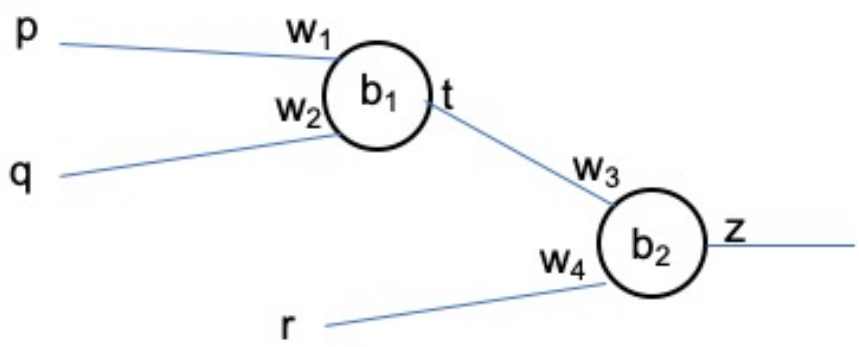
$$Precision = \frac{TP}{TP + FP}$$

Which of these outcome metric(s) is the most important one to use to judge the success of the neural net classifier? Why?

Please write your answer here:

Question 3 [8 Marks]

Consider the neural net illustrated below with two neurons. It has three inputs p, q and r, and one output, z. Each neuron is a standard linear neuron *without* an activation function. The lines annotated with w_i indicate the weight applied to each input. The bias of each neuron is given in the circle representing the neuron; b_1 is the bias for the neuron with output t and b_2 is for the neuron with output z. This network is to be trained on a dataset using a *squared error* loss.



3(a) [6] You are to determine the expression for the gradient of parameter w_1 with respect to the loss, *for a single training example* with inputs p, q and r and a label L. Make use of the chain rule to simplify the process of derivation. Your answer must be expressed only in terms of the inputs, weights and biases. Be sure to show your derivation steps.

Please write your derivation and solution here (and continue on next page):

Your answer to question 3(a) continues:

3(b) [2]: Assume that a ReLU activation function has been placed on the output of the neuron z. Give the expression for the gradient of w_1, as in part (a), that accounts for this change. Your answer may reference the solution given in part (a).

Please write your answer here:

Question 4 [10 Marks]

In convolutional neural networks, the *receptive field* of a kernel in a convolutional layer is defined as the amount of the network *input* that the kernel receives, directly or indirectly through other layers, to perform *one* dot product on. For example, in the 2D CNN that you used in Assignment 4, the receptive field of a kernel (in a convolutional layer) would be the number and region of pixels in the original image that the kernel operates on, either directly or indirectly. Consider the following PyTorch code of a model for a CNN, from the CIFAR10 classifier discussed in class:

```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)

        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Here the definitions of the three relevant PyTorch methods from the PyTorch documentation:

```
class nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,
                      padding=0, dilation=1, groups=1, bias=True)

class nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1,
                         return_indices=False, ceil_mode=False)

class nn.Linear(in_features, out_features, bias=True)
```

Question 4 continues.

(a) [2] What is the size (in number of pixels) of the receptive field of the kernels on the *first* convolutional layer? What is the 2-dimensional shape of that receptive field?

(b) [6] What is the size (in number of pixels) of the receptive field of kernels on the *second* convolutional layer? What is the 2-dimensional shape of that receptive field? Show how you arrived at your answer, and be sure to state any assumptions you make.

Question 4 continues.

(c) [2] When you are designing a CNN for an image classification problem, how should the choices of receptive fields (that would be implied by your convolutional layer design) be influenced by the nature of the classification problem?

Question 5 [9 Marks]

This question concerns the method of Stochastic Gradient Descent.

(a) [3] In the context of neural network training, what is *stochastic* gradient descent and why is it used?

(b) [2] What are the negative effect(s) when the batch size is too small?

(c) [2] What are the negative effect(s) when the batch size is too large?

Question 5 continues.

(d) [2] If Batch Normalization is used inside a neural net training system, how would it be affected by a batch size that was too small?

Please write your answer here:

Question 6 [7 Marks]

Consider the following PyTorch code:

```
import torch
import torch.nn as nn

class My_Model(nn.Module):

    def __init__(self, isize,firstL,scale):

        super(My_Model, self).__init__()
        self.fc1 = nn.Linear(isize, firstL)
        secondL = int(firstL/scale)
        self.fc2 = nn.Linear(firstL,secondL)
        thirdL = int(secondL/scale)
        self.fc3 = nn.Linear(secondL,thirdL)
        self.nonlin = nn.ReLU()
        self.out = nn.Sigmoid()

    def forward(self, din):

        x = self.nonlin(self.fc1(din))
        x = self.nonlin(self.fc2(x))
        x = self.fc3(x)

        return self.out(x)

inst = My_Model(3,4,2)
```

See Question 4 for the form of the nn.Linear function. You are to draw an illustration of the neural network instantiated by the last line of the above code - i.e. an illustration of the model's forward function computation. Your illustration should include enough additional written information such that someone could, using just your picture and written information, write a Python implementation that replicates the functionality of the forward function (without the benefit of PyTorch, or any other library). This includes providing the correct mathematical form of all the computations expressed in the code.

Please put your rough work below, and final answer on the next page:

Question 6 continued; place your answer here:

Question 7 [6 Marks]

Generative models, as described in the lectures about autoencoders, are neural networks that *create* examples of data, rather than classifying data. In this question you are to provide the model definition, in PyTorch, of the following generative model: The input to the model is a 9-element vector of numbers, and the output is a 32x32 gray-scale picture. The model should be a fully-connected MLP with three layers (where each layer includes a non-linear activation function); other than this the architecture should be of your own choosing. Note the following:

i The question is asking only for the model, not the training code.

ii The outputs should consist of values between 0 and 1, that could be converted to a grayscale range (which you do not have to do).

iii Make sure that the output of the model has the correct 32x32 shape.

Answer:

Question 8 [9 Marks]

Recall the concept of Word Vectors (also called word embeddings) that were used as the basis for Assignment 5. Word vectors are created through the training of a neural network on a corpus of many sentences. One such method was described in class, in which the neural network was trained to predict the word immediately following a given word.

Another method is to train a neural network as follows: Given a sequence of three consecutive words in the corpus (for example: WordA WordB WordC), predict the middle word (WordB), given the first and last word (WordA and WordC) as inputs.

a) [7] Draw the architecture of the neural network using this different method with the following specifications:

   i) The size of the vocabulary to be trained is 100 words.

   ii) The size of the word vector is 5.

   Explain how the network works in writing, and be sure to describe any special requirements or constraints that are necessary on the network parameters.

   Put your rough work here and your full answer on the next page:

Your answer to question 8(a) should go here:

Question 8, continued

b) [2] Consider the following sentence, which is from the training text corpus that could be used to train this network:

```
The machine learns what it needs to learn.
```

Give **two** examples of a training sample (input data and label) that could be taken from this sentence.

Place your answer here:

Question 9 [8 Marks]

Transfer Learning. Please read all three parts of question before answering.

(a) [2] What is Transfer Learning?

(b) [4] Give an example where transfer learning could be used, and describe how it would be done in general terms.

(c) [2] List two advantages that Transfer Learning provides.

This page has been left blank intentionally. You may use it for answers to any question in this examination.