Last Name: _____

First Name: _____

Student Number: _____

Applied Fundamentals of Machine Learning

Date :	October 25th, 2021
Duration:	**120 minutes**
	(90 min + 30 min for Crowdmark submission)
Aids Allowed:	**All Quercus Notes**
Instructors:	Kaveh Hassani, Jaspreet Sahota
TAs:	Ali Khodadai (Q2,3), Martiya Zare Jahromi (Q4,5), Geoff Donoghue (Q6,7), Gianluca Villani (Q8,9)

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This test consists of 8 questions on 22 pages (including this one), printed on both sides of the paper. When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above. Bubble in your student number in either pen or pencil.

Answer each question directly on the test paper, in the space provided, using either a blue or black pen or a pencil. If you need more space for one of your solutions, use the extra pages at the end of the test paper and *indicate clearly the part of your work that should be marked.*

Write up your solutions carefully! Marks cannot be awarded for solutions that are not understandable by the grader, and may be deducted if you make false assertions. If you are giving only one part of an answer, indicate clearly what you are doing. Part marks might be given for incomplete solutions.

MARKING GUIDE

1: _____/10

2: _____/ 8

3: _____/11

4: _____/10

5: _____/14

6: _____/12

7: _____/10

8: _____/10

TOTAL: _____/85

Good Luck!

OVER...

Question 1. [10 MARKS]

Circle the best answer for each of the questions below. Do not circle more than one answer per question.

Part (a) [1 MARK]

Which of these problems is best described as an *unsupervised* learning problem?

(A) Generating a random number
(B) Clustering an unlabelled data set * (2 marks)
(C) Finding the shortest path between two points of a graph
(D) Predict house prices from the size of house, given a dataset of house prices/sizes

Part (b) [1 MARK]

Which of these best describes the concept of *inductive bias*?

(A) Bias of a machine learning algorithm learned from a training data set
(B) The assumptions we incorporate in our model to help learn to solve a problem * (2 marks)
(C) Learning from an unbalanced data set
(D) A model with high precision but low recall

Part (c) [1 MARK]

Which of the following activation functions are **not** compatible with back-propagation?

(A) $f(x) = \tanh(x)$
(B) $f(x) = (1 + e^{-x})^{-1}$
(C) $f(x) = \text{sign(x)}$ * (2 marks)
(D) $f(x) = \max(0, \text{x})$

Part (d) [1 MARK]

Which of the following is true about *convolutional neural networks*?

(A) Pooling/stride reduce the number of channels of an input image/feature map.
(B) Convolving an input image/feature map having any number of channels with a single kernel produces a feature map with *one* channel. * (2 marks)
(C) The order of operations used in a convolutional neural network is typically "Convolution-Pooling-Activation".
(D) Convolution is used to reduce the height/width of an input image/feature map.

Part (e) [1 MARK]

Which *loss* would be most suitable for training a neural network for *regression*?

(A) Delta rule
(B) Back-propagation
(C) Mean squared error * (2 marks)
(D) Cross entropy

Part (f) [1 MARK]

Suppose we have an input image of size 3×15×15 and we apply a kernel of size 3×4×4 with padding of size 2 and stride of 1, followed by a pooling with kernel size of 4×4 and strides of 4. What are the dimensions of the output?

(A) 1×8×8
(B) 3×8×8
(C) 1×4×4 * (2 marks)
(D) 3×4×4

CONT'D...

Part (g) [1 MARK]

Which of these computes the gradients in PyTorch?

(A) `optimizer.zero_grad()`

(B) `optimizer.step()`

(C) `loss.backward()` * (2 marks)

(D) `loss = criterion(model(inputs), labels)`

Part (h) [1 MARK]

Which of the following is **correct**?

(A) We optimize the weights on training set and the hyper-parameters on the test set.

(B) We can increase the learning rate as training progresses.

(C) Smaller batch sizes typically allow using larger learning rates.

(D) Using a standard adaptive learning rate method doubles the memory usage. * (2 marks)

Part (i) [1 MARK]

Assume an input image of size 64×64 with 3 channels, and a model with a single convolutional layer having 2 kernels of size 3x3, and 3 kernels of size 5×5. What are the total number of convolutional weights you need to learn? Assume there are no bias terms.

(A) $(2 \times 3 \times 3 \times 3) + (3 \times 3 \times 5 \times 5)$ * (2 marks)

(B) $(64 \times 64 \times 3) + (2 \times 3 \times 3) + (3 \times 5 \times 5)$

(C) $(2 \times 3 \times 3) + (3 \times 5 \times 5)$

(D) $64 \times 64 \times 3 \times 3 \times 5 \times 5$

Part (j) [1 MARK]

Which of the following is **incorrect** about convolutional neural networks?

(A) They are more efficient compared to fully-connected networks.

(B) As we go deeper, height and weight are decreased while depth is increased.

(C) We can use either convolution with strides or pooling layers to reduce the depth. * (2 marks)

(D) Kernels of size 3×3 can approximate bigger kernel sizes.

Question 2. [8 MARKS]

You are given a data set to solve a problem and split this data into training and test data sets. You decide to train a deep neural network (NN) to solve the problem, however you are not sure what batch size to use.

Part (a) [4 MARKS]

Having a larger batch size gives us a better approximation of the gradient at a specific point in the optimization space (i.e. the space of points corresponding to model parameters that will be learned). Given this, is it always a good strategy to train with a large batch size? Why or why not?

No. (1 marks)

You could run out of GPU memory (1 marks)

Larger batch size could lead to higher test error due to the model training not converging. Smaller batch size means more noise in training, which can aid optimization by navigating local minimums more effectively. (2 marks)

Part (b) [2 MARKS]

Now suppose that you train many different NN models (each having different hyper-parameters), and you select the model with the best accuracy on the test set. Explain potential flaws with this methodology.
There's a risk of over-fitting to the test data (1 marks)
There's no reason to believe the NN would be the best model. Other models should be considered depending on the underlying data. (1 marks)

Part (c) [2 MARKS]

Propose a more effective methodology and also explain its drawbacks.
Use a train/test/validation data set. Or any split involving a validation set (1 marks)
Accept either of the following drawbacks, max possible marks for this question is 2. (marks)
less data for training (1 marks)
computational cost of exploring more model and more hyperparameters (1 marks)

Question 3. [11 MARKS]

Provide concise answers to the following questions regarding training of neural networks.

Part (a) [2 MARKS]

Why is backpropagation easier in the output layer (i.e. the delta rule), than for hidden layers?
can calculate partial derivative with respect to error easily since only each neuron affects the error of only one output directly (2 marks)

Part (b) [2 MARKS]

In the weight update rule for backpropagation/the delta rule, why do we *subtract* the gradient from the weights rather than *add* the gradient to the weights?
direction of the gradient is the direction in which the function increases most quickly, we want to move in the direction that decreases the loss (2 marks)

Part (c) [2 MARKS]

How does using *momentum* help training?
it adds a memory of previous step to the optimizer / prevents the optimization from changing direction too quickly, speeding up training in some loss landscapes (2 marks)
or if they only say "it speeds up training" (1 marks)

Part (d) [2 MARKS]

What are **two** important characteristics of *all* activation functions used in modern neural networks trained with backpropagation.
differentiable (1 marks)
non-linear (1 marks)

Part (e) [3 MARKS]

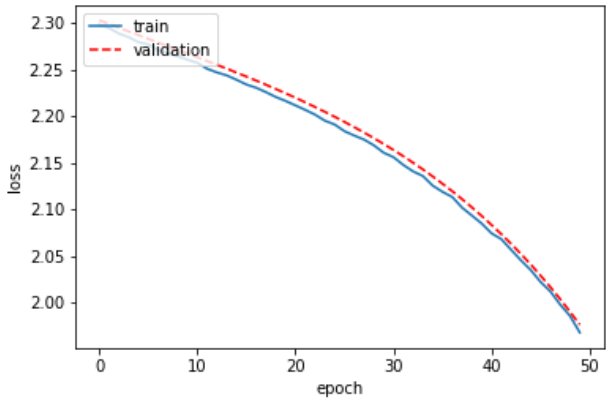
Assume we have a dataset of 20,000 examples and we sample mini-batches of size 20. How many iterations are there in one epoch if we train with: (a) stochastic gradient descent, (b) mini-batch gradient descent, and (c) batch gradient descent? Note we are using the strict definition of stochastic gradient descent here.
20,000 (1 marks)
1000 (1 marks)
1 (1 marks)

Question 4. [10 MARKS]

During hyperparameter tuning of a neural network model, we encountered different loss vs. epoch plots. Each of the following plots corresponds to a set of hyperparameters. Pick the best set of hyperparameters among the plots below (Write "The best" in front of the set name). For the remaining sets, suggest a change in hyperparameters() for each that may result in better performance.

Part (a) [2 MARKS]

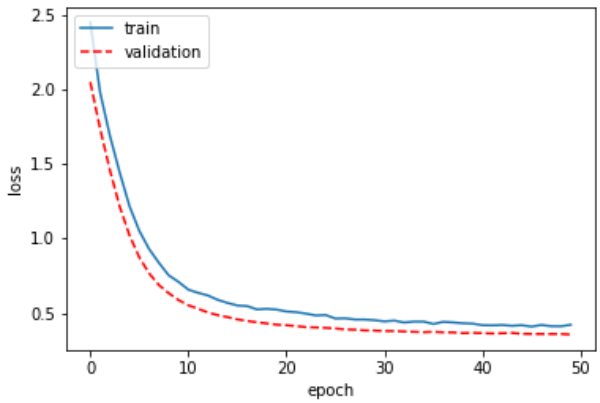
SET A



Use more epochs (1 marks)
Use larger learning rate (1 marks)

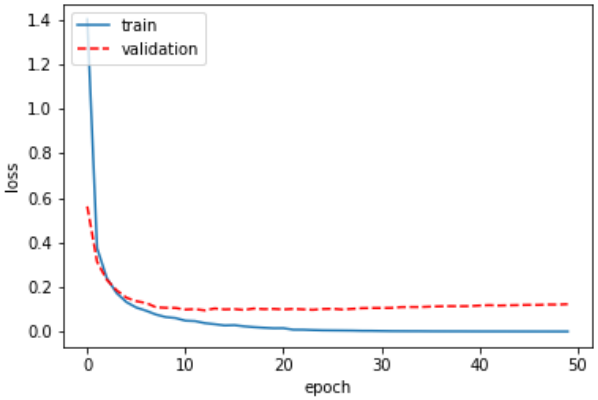
Part (b) [2 MARKS]

SET B



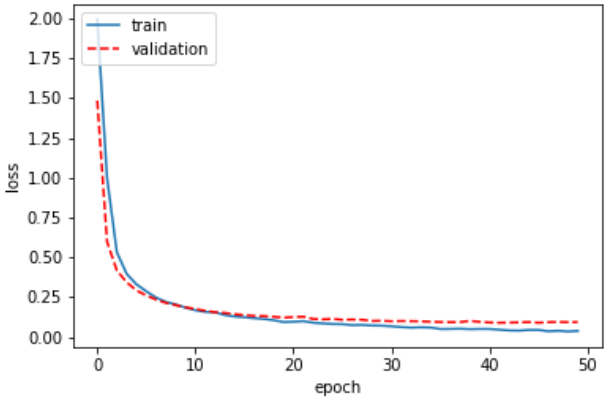
Increase model capacity (e.g. number or layers, parameters, etc.) as model is underfitting. (2 marks)

Part (c) [2 MARKS]
SET C



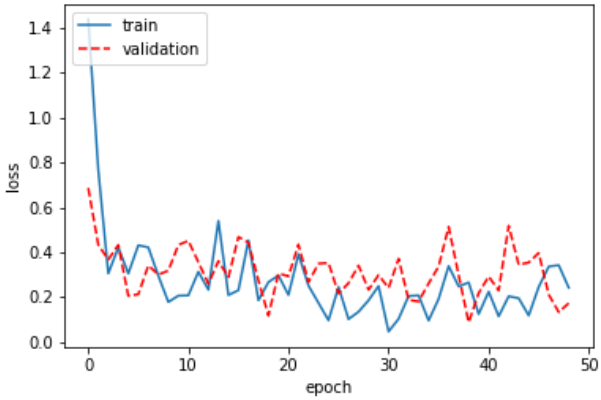
Overfitting. Suggestion: Reduce the capacity of the model (Number of layers/ layers width/ parameters). (2 marks)

Part (d) [2 MARKS]
SET D



The best set of hyperparameters (2 marks)

Part (e) [2 MARKS]
SET E



Fluctuation. Suggestion: smaller learning rate and/or larger batch size (2 marks)

Question 5. [14 MARKS]

Given the code below, for a binary neural network classifier, answer the following questions.

```
    def __init__(self):
        super(MLP, self).__init__()
        self.firstLayer = nn.Linear(20, 10)
        self.middleLayer = nn.Linear(10, 10)
        self.lastLayer = nn.Linear(10, 1)
    def forward(self, input):
        activation1 = self.firstLayer(input)
        activation1 = F.relu(activation1)
        activation2 = self.middleLayer(activation1)
        activation2 = F.relu(activation2)
        activation3 = self.middleLayer(activation2)
        activation3 = F.relu(activation3)
        activation4 = self.lastLayer(activation3)
        return activation4
```

Part (a) [2 MARKS]

What is the input size of the network?

Partial marks for 20 (1 marks)

Full marks for (20, B), where B is the batch size. Max possible marks 2. (2 marks)

Part (b) [1 MARK]

How many layers are in the network?

4 (1 marks)

CONT'D...

Part (c) [1 MARK]

What is the total number of parameters in the network (including Biases)?
No partial marks. (marks)
441 = (20*10 + 10) + (10*10 + 10) + (10*10 + 10) + (10*1 + 1) (1 marks)

Part (d) [4 MARKS]

Which loss function should you use and Why?
We should use BCEWithLogitsLoss() (1 marks)
Reason: We need a binary cross entropy loss because the network is a binary classifier (1 marks)
Reason: No sigmoid is applied in the code above so we need BCEWithLogitsLoss() (1 marks)
BCEWithLogitsLoss() is more numerically stable then BCELoss() (1 marks)

Part (e) [2 MARKS]

How would you modify this network (including its loss) in order to classify days of the week (Mon, Tue, Wed, Thur, Fri, Sat, Sun)?
Set self.lastLayer to nn.Linear(10, 7) (1 marks)
Use torch.nn.CrossEntropyLoss() loss (1 marks)

Part (f) [4 MARKS]

Below, you may find a part of the code used for training a Neural Network. There are 5 blank spaces (#___) for comments in the following code. From the list of the comments below, please put the number of each comment in the right place. (Number 1 is done as an example)

1. # backward pass
2. # update the weights
3. # forward pass
4. # reset gradients to zero
5. # loss calculation

```
for epoch in range(num_epochs):
        ...
        for i, data in enumerate(train_loader, 0):
            ...
            optimizer.zero_grad()  #___

            outputs = net(inputs)  #___

            loss = criterion(outputs, labels.float())  #___

            loss.backward()  #___

            optimizer.step()  #___
            ...
```

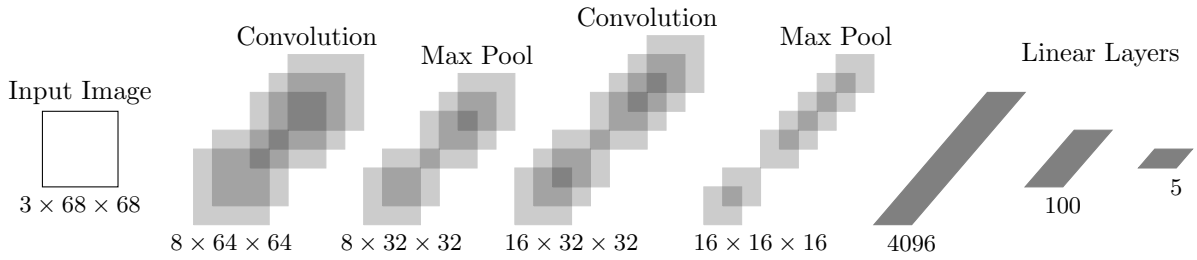

CONT'D...

Question 6. [12 MARKS]

Consider the convolutional neural network (CNN) architecture below.

Part (a) [6 MARKS]

Complete the PyTorch code to achieve the architecture described in the diagram below. You may assume that all the necessary packages have already been imported, and that the code has no syntax errors. Note that the size mentioned under each block is the output size of that operation. For example, $8 \times 64 \times 64$ is the size of the feature map that is generated by applying the first convolution to the input image.




```
class CNNClassifier(nn.Module):
    def __init__(self):
        super(CNNClassifier, self).__init__()
        # Conv2d: in_ch, out_ch, kernel_size, stride, padding
        self.conv1 = nn.Conv2d(____,____,____,____, 1)
        # Pool: kernel_size, stride
        self.pool = nn.MaxPool2d(____,____)
        self.conv2 = nn.Conv2d(____,____,3, ____, ____)
        self.fc1 = nn.Linear(____,____)
        self.fc2 = nn.Linear(____,____)

    def forward (self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(____, ____)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Please fill in the blanks (___) in the code block above.
(3, 8, 7, 1) (1 marks)
(2,2) (1 marks)
(8, 16, 1, 1) (1 marks)
(4096, 100) (1 marks)
(100, 5) (1 marks)
(-1, 4096) (1 marks)

Part (b) [3 MARKS]

While training this CNN, your GPU runs out of memory. What are three things you could do to avoid this error?

1 mark for any of the following. Max possible marks is 3 (marks)
Lower the number of hidden layers/Reduce model size (1 marks)
Add more pooling layers, reduce dimensionality CNN layers (1 marks)
Reduce batch size (1 marks)
Apply transfer learning. Tune only a subset of the CNN layers (1 marks)

Part (c) [3 MARKS]

What changes would you need to make to remove the max pooling layers and still maintain the overall network architecture (i.e. deminsionality of Conv layers)? Indicate which lines of code need to be updated and provide the changes.

1 mark for having the right approch: i.e. remove pooling and increase stride. 2 more marks for getting the correct details below. Max possible marks is 3. (1 marks)
Remove line 7, and remove self.pool() function in 13 and 14 (1 marks)
Set each nn.Conv2d() stride from 1 to 2 (1 marks)

Question 7. [10 MARKS]

The following code is to be used for MNIST classification with images of 28×28 pixels. You may assume that all necessary packages have been imported and that the code has no syntax errors.

```
mnist_train = mnist_data[:3200]
mnist_val = mnist_data[3200:4224]

class MNISTClassifier(nn.Module):
    def __init__(self):
        super(MNISTClassifier, self).__init__()
        # Conv2d: in_ch, out_ch, kernel_size, stride, padding
        self.conv1 = nn.Conv2d(1, 4, 5)
        # Pool: kernel_size, stride
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(5, 10, 3)
        self.fc1 = nn.Linear(250, 32)
        self.fc2 = nn.Linear(32, 10)

    def forward (self, x):
        x = self.pool(F.sigmoid(self.conv1(x)))
        x = self.pool(F.sigmoid(self.conv2(x)))
        x = x.view(-1, 250)
        x = F.sigmoid(self.fc1(x))
        x = self.fc2(x)
        return x

def train(model, data, batch_size=32, num_epochs=4):
    train_loader = torch.utils.data.DataLoader(data, batch_size)
    critereon = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr = 0.01, momentum = 0.9)

    for epoch in range(num_epochs):
        for imgs, labels in train_loader:
        out = model(imgs)
        loss = critereon(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

model = MNISTClassifier()
train(model, mnist_train, num_epochs = 2)
```

Part (a) [2 MARKS]

What is the total number of iterations in the above code? You may assume the code works as intended.

$2 \times (3200/32) = 200$ (2 marks)

Part (b) [4 MARKS]

Determine the total number of parameters/weights in this model (including Biases).

Conv1: $(1 \times (5 \times 5) + 1) \times 4 = 104$ (1 marks)

Conv2: $(5 \times (3 \times 3) + 1) \times 10 = 460$ (1 marks)

FC1: $(250 + 1) \times 32 = 8032$ (1 marks)

FC2: $(32 + 1) \times 10 = 330$ (1 marks)

Total: 8926 (1 marks)

Part (c) [4 MARKS]

The above code yields an error that the dimensions are not correct. How can you change the first convolutional layer to correct the error? Show your work.

Change the out channel size in conv1 from 4 to 5. (marks)
Conv1: $(28 + (2 \times 0) - 5)/1 + 1 = 24$ (marks)
Pool1: $(24 + (2 \times 0) - 2)/2 + 1 = 12$ (marks)
Conv2: $(12 + (2 \times 0) - 3)/1 + 1 = 10$ (marks)
Pool2: $(10 + (2 \times 0) - 2)/2 + 1 = 5$ (marks)
FC1 input: final img size x conv layer output $= 5 \times 5 \times 10 = 250$ Correct. (marks)

Question 8. [10 MARKS]

Following are a series of questions based on deep learning architectures we've seen in the course.

Part (a) [2 MARKS]

One of the biggest challenges in deep learning has been the problem of *vanishing* or *exploding* gradients preventing us from training very deep models. Name **two** improvements that have helped address this problem.
improved initialization (1 marks)
ReLU activation function (1 marks)
residual connections (1 marks)
intermediate losses (GoogLeNet) (1 marks)
normalization/batch normalization (1 marks)

Part (b) [2 MARKS]

GoogLeNet (or the Inception architecture) has many fewer weights than AlexNet, and yet achieves a higher accuracy on the ImageNet test set. What two changes from AlexNet are most responsible for this?
1×1 convolutions (1 marks)
multiple filter sizes on one layer, with more small filters than large (1 marks)
increased depth of GoogLeNet (not a clear/major reason for better parameter efficiency, but we gave part marks since it's not in-plausible) (0.5 marks)

Part (c) [2 MARKS]

What are the two most important reasons AlexNet was able to demonstrate the potential of convolutional neural networks so successfully, as compared to LeNet decades before?
large datasets (1 marks)
large compute/gpu acceleration (1 marks)
ReLU (1 marks)
data augmentation (0.5 marks)
dropout/better regularization (0.5 marks)
learning a more complex problem (0.5 marks)

Part (d) [2 MARKS]

What is the (spatial) convolutional filter size used in the VGG architecture? How did the architecture justify using this filter size exclusively?
3x3 (or just 3 is acceptable) (1 marks)
stacking 3x3 layers can approximate any larger filter (1 marks)

Part (e) [2 MARKS]

Your model has a layer, named `layerX`, used as follows:

```
class MyModel(nn.Module):

    def forward(self, img):
        ...
        output = self.layerX(prev_output)
        ...
```

where `prev_output` is the output from the previous layer in the model. How would you change the code to add a residual (or skip) connection to `layerX`?
`output = prev_output + self.layerX(prev_output)` (2 marks)

Additional page for answers

Additional page for answers

Additional page for answers