Harrison Varnado

CS 4267: Introduction to Machine learning
Final Exam Practice test- Part1

1.Consider K-means algorithm for the following **distance** matrix where K=2 and the initial points are chosen as *a* and *b*.

	a	b	c	d	e	f
a	0	9	7	5	4	2
b	9	0	2	1	7	3
c	7	2	0	3	5	4
d	5	1	3	0	4	5
e	4	7	5	4	0	3
f	2	3	4	5	3	0

a. (4 points) What are the initial assignments of the data points to clusters?

C1: Data points of Cluster having *a*	C2: Data points of Cluster having *b*
a, e, f	b, c, d

Do not forget to list b and e in clusters.

(b) Using the clustering results above, calculate the intra-cluster distances for cluster C1 and C2. (6 points)

$d(a,e) = 4$ $D_{c_1} = 4 + 2 + 3 = 9$

$d(a,f) = 2$

$d(e,f) = 3$

$d(b,c) = 2$ $D_{c_2} = 2 + 1 + 3 = 6$

$d(b,d) = 1$

$d(c,d) = 3$

2.In a dataset, there are 1000 samples from apples and oranges. The dataset is split into two parts as 90% and 10% for training and testing, respectively. In the testing data, there are equal numbers of for apples and oranges. A supervised machine learning algorithm is trained to detect apples and oranges using the training dataset. When it is tested with the testing data, it finds the following results:

- It classifies 5 apples as oranges.
- It classifies 40 oranges as oranges.

Fill out the confusion matrix for the given information above:

Model 1	Predicted Class		
		Apple (+)	Oranges (-)
Actual Class	Apple (+)	a 45	b 5
	Oranges (-)	c 10	d 40

TP FN
FP TN

What are the a, b, c, d values in the confusion matrix?

$a=45$ $b=5$ $c=10$ $d=40$

b)Compute the accuracy, sensitivity (TPR), specificity (TNR), and precision for the confusion matrix. Show your below and write your results.

$$\text{accuracy} = \frac{45+40}{45+5+10+40} = \frac{85}{100} = 0.85$$

$$TPR = \frac{45}{45+5} = \frac{45}{50} = 0.9$$

$$TNR = \frac{40}{40+10} = \frac{40}{50} = 0.8$$

$$\text{precision} = \frac{45}{45+10} = \frac{45}{55} = 0.\overline{81}$$

3.Consider the following training set and a test object **X(O=Sunny, T=Mild, W=True).** Use Naïve Bayesian Classifier for the following questions assuming attributes are independent.

ID	Outlook (O)	Temperature (T)	Windy (W)	Play Golf (Class)
1	Rainy	Hot	False	No
2	Rainy	Mild	False	No
3	~~Overcast~~	~~Hot~~	~~False~~	~~Yes~~
4	~~Sunny~~	~~Hot~~	~~True~~	~~Yes~~
5	~~Sunny~~	~~Cool~~	~~False~~	~~Yes~~
6	Sunny	Cool	True	No
7	~~Overcast~~	~~Cool~~	~~False~~	~~Yes~~
8	Rainy	Mild	False	No
9	Rainy	Cool	False	No
10	~~Sunny~~	~~Mild~~	~~False~~	~~Yes~~

a) **(8 points)** Predict the class of X by writing the following probabilities or expressions:

a)	P(Play=Yes)	$5/10 = 1/2$
b)	P(Play=No)	$5/10 = 1/2$
c)	P(O=Sunny|Play=Yes)	$3/5$
d)	P(T=Mild|Play=Yes)	$1/5$
e)	P(W=True|Play=Yes)	$2/5$
f)	P(O=Sunny|Play=No)	$1/5$
g)	P(T=Mild|Play=No)	$2/5$
h)	P(W=True|Play=No)	$1/5$
i)	P(X|Play=Yes)	$3/5 \cdot 1/5 \cdot 1/5 = 3/125$
j)	P(X|Play=No)	$1/5 \cdot 2/5 \cdot 1/5 = 2/125$
k)	P(X|Play=Yes)*P(Play=Yes)	$3/125 \cdot 1/2 = 3/250$
m)	P(X|Play=No)* P(Play=No)	$2/125 \cdot 1/2 = 2/250$

Class of X is	yes

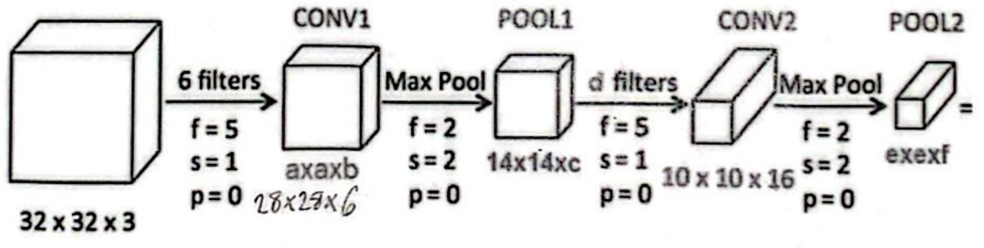
4.Fill the blanks below in the process of training a Convolutional Neural Networks (CNN).

1. Download big datasets
2. Design CNN architecture
3. Initialize weights
4. For t = 1 to T:
1.
   - i. form minibatch
   - ii. Copute loss + gradient
   - iii. update weights
5. Apply trained model to task

b) b) Briefly, describe how the minibatch work in a CNN.

mini batches assist in training. By splitting the train set into smaller sets (mini batches) it can complete training quicker and reduce the chance of over training by rotating data

c) In CNN, the convolution and pooling layers may change the size of the feature map. A simple CNN is given below with input image size of 32x32x3. In each layer, several filters are applied with different filter size (f), stride (s), and padding (p). Compute the missing values (a, b, c, d, e, f).



a = 28 c = 6 e = 5
b = 6 d = 16 f = 16

5. Explain Adaptive Boosting Algorithm

AdaBoost has multiple weak learners and takes their outputs and combines them into a weighted sum. It is typically a binary output

Harrison Varnadoe

CS4267: Machine Learning
Sample Final Exam Part 2
Kennesaw State University

1. For the following code matrix, determine how x would be classified if weak decision makers yields

[+1, -1, -1, +1, +1, -1, +1].

$$
W = \begin{array}{c} \\ \\ \\ \\ \end{array}
\begin{array}{ccccccc}
d_1 & d_3 & d_{3:4} & d_2 & d_{2,4} & d_{2:3} & d_{2:4} \\
-1 & \boxed{-1} & \boxed{-1} & -1 & -1 & \boxed{-1} & -1 \\
-1 & \boxed{-1} & \boxed{-1} & \boxed{+1} & \boxed{+1} & +1 & \boxed{+1} \\
-1 & +1 & +1 & -1 & -1 & +1 & \boxed{+1} \\
\boxed{+1} & \boxed{-1} & +1 & -1 & \boxed{+1} & \boxed{-1} & \boxed{+1}
\end{array}
$$

$C_1 = -1+1+1-1-1+1-1 = -1$

$C_2 = -1+1+1+1+1-1+1 = 3$

$C_3 = -1-1-1-1-1-1+1 = -5$

$C_4 = 1+1-1-1+1+1+1 = 3$

C_2

2.For the following, weak classifier outcomes use the fixed combination rules: sum, median, minimum, maximum, product.

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2

C_1:

sum = 0.2

median = 0.2

min = 0.0

max = 0.4

product = $0.2 \cdot 0 \cdot 0.4 = 0$

C_2: Sum = 0.5

med = 0.5

min = 0.4

max = 0.6

prod = $0.5 \cdot 0.6 \cdot 0.4 = 0.12$

C_3:

sum = 0.3

med = 0.3

min = 0.2

max = 0.4

prod = 0.032

5.Apply convolution for the following matrix using the following kernel with strides of 1 and 2 as well as without padding and with padding.

0	0	0	0	0	0	0	0	0	0
0	1	3	5	0	0	0	0	0	0
0	3	5	3	5	6	2	1	1	0
0	5	2	4	0	0	0	0	1	0
0	7	1	0	6	3	8	6	8	0
0	8	2	0	0	1	2	4	2	0
0	5	4	3	2	1	8	7	3	0
0	8	7	6	5	4	3	4	1	0
0	1	0	0	1	1	0	0	1	0
0	0	0	0	0	0	0	0	0	

Kernel=. $\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$

s=1 p=0

-2	2	1	0	0	-1
3	6	5	4	-8	-19
1	4	3	-3	-7	-7
-4	-2	3	6	1	4
-11	-16	-14	-9	-4	0
11	8	4	9	15	17

s=2 p=0

-2	1	0
1	3	-7
-11	-14	-4

s=1 p=1

-8	-11	-13	-14	-13	-9	-4	-2
-3	-2	2	1	0	0	-1	-1
0	3	6	5	4	-8	-19	-12
-3	1	4	3	-3	-7	-7	-5
-1	-4	-2	3	6	1	4	4
-5	-11	-16	-14	-9	-4	0	-1
8	11	8	4	9	15	17	9
15	21	18	15	12	11	8	5

s=2 p=1

-8	-13	-13	-4
0	6	-4	-18
-1	-2	6	4
8	8	9	17

Harrison Varnado

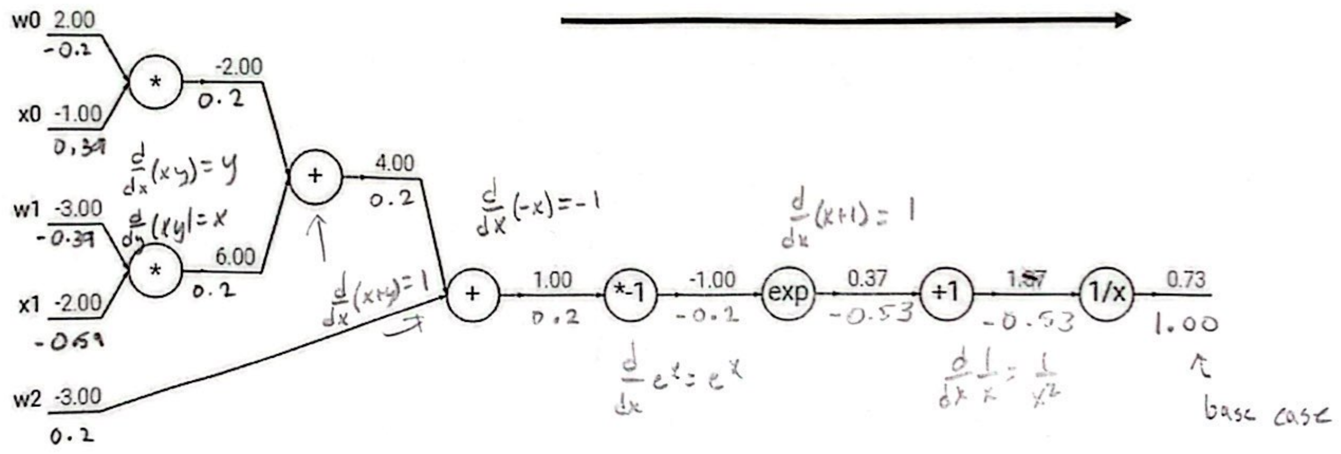
6. Apply pooling of (2,2) with stride=2 for the following matrix.

4	3	5	0	2	0	0	0
3	5	3	5	6	2	1	1
6	2	4	6	9	2	0	1
7	1	0	3	3	8	6	8
8	2	0	0	1	2	4	2
5	4	3	2	1	8	7	3
8	7	6	5	4	3	4	1
4	1	0	1	1	0	0	1

max pooling

$$\begin{matrix} 5 & 5 & 6 & 1 \\ 7 & 6 & 9 & 8 \\ 8 & 3 & 8 & 7 \\ 8 & 6 & 4 & 4 \end{matrix}$$

7.. For the following computational graph, show the computations for the forward pass and gradients for the backward pass.



8. Describe, and compare ensemble methods: bagging, boosting (e.g., AdaBoost), stacking, mixture of experts, cascading.

Bagging - considers homogenous weak learners, trains them independantly, + uses an averaging function to combine the outputs

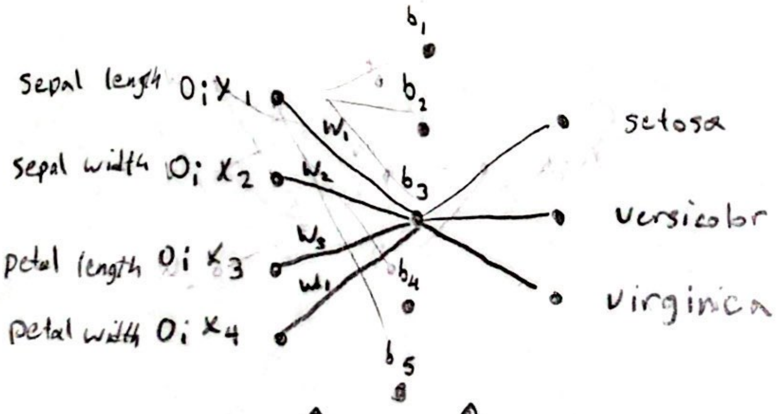
Boosting - takes weak learners + learns them sequentially + adaptivly + combines them deterministically

Stacking - takes weak learners + trains them in parralell + combines them by traing a meta-model to output a prediction based on the weak models

mixture of experts - weights are gating / input dependant

Cascading - only proceeds if the previous model is unsure

9.Develop a multi-layer perceptron with weights for the XOR function.



There are more connections here, just immitted for legibility

KENNESAW STATE
UNIVERSITY

CS 4267
MACHINE LEARNING

PROJECT 3
Deep Learning for Classification

INSTRUCTOR

Emin Mary Abraham

Harrison Varnadoe
000873793

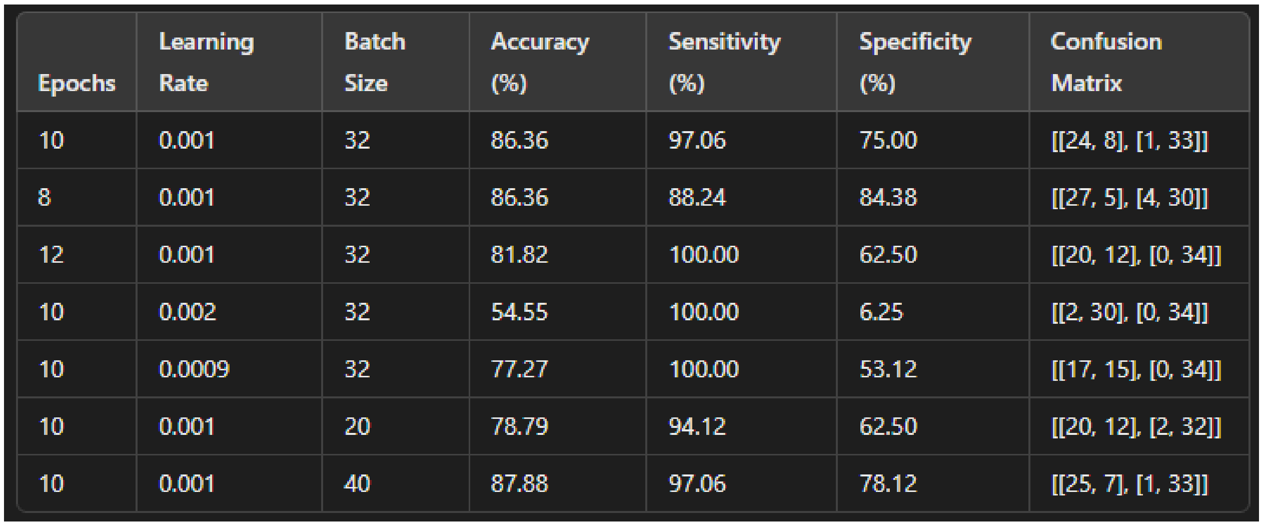
1. ABSTRACT

In this project I utilized the ResNet CNN from pytorch and retrained it to classify fundus images from IDRiD to determine whether they contained a reasonable mixture of disease stratification representative of diabetic retinopathy (DR). The Images where already split into training and testing set but I did some work to sort out the different classifications. The CNN was a binary classifier that either returned a positive value for DR or a negative value for non DR.

2. RESULTS AND DİSCUSİON

2.1 Results

I tested with the default values of epochs = 10, learning rate = 0.0001, and mini batch size = 32. From there, I tweaked the different values and observed the differences. The values were recorded into a text file which was passed into ChatGPT to format into a table.



Epochs	Learning Rate	Batch Size	Accuracy (%)	Sensitivity (%)	Specificity (%)	Confusion Matrix
10	0.001	32	86.36	97.06	75.00	[[24, 8], [1, 33]]
8	0.001	32	86.36	88.24	84.38	[[27, 5], [4, 30]]
12	0.001	32	81.82	100.00	62.50	[[20, 12], [0, 34]]
10	0.002	32	54.55	100.00	6.25	[[2, 30], [0, 34]]
10	0.0009	32	77.27	100.00	53.12	[[17, 15], [0, 34]]
10	0.001	20	78.79	94.12	62.50	[[20, 12], [2, 32]]
10	0.001	40	87.88	97.06	78.12	[[25, 7], [1, 33]]

2.2 Discussion

As can be seen in the table, lowering the epochs to 8 yielded similar results and setting the batch size to 40 showed marginal improvements. However, the values I landed on for the defaults seemed to be the most consistent in my testing. The learning rate seems to be the most sensitive variable as changing it slightly has drastic impacts on the model’s performance. Increasing the batch size past 40 also decreased the models performance. With all of the values I settled on, I am sure there is still room for improvement.

3. CODE

```python
import os
import numpy as np
import torch
import torch.nn as nn
from torchvision import datasets, transforms, models
from torchvision.models import ResNet18_Weights
from torch.utils.data import DataLoader, SubsetRandomSampler
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

def filter_files(root_dir):
    files = []
    for file in os.listdir(root_dir):
        files.append(file)
    return files

train_DR = filter_files('DS_IDRID/Train/DR')
train_Non_DR = filter_files('DS_IDRID/Train/NonDR')
test_DR = filter_files('DS_IDRID/Test/DR')
test_Non_DR = filter_files('DS_IDRID/Test/NonDR')

train_data = datasets.ImageFolder(root='DS_IDRID/Train', transform=transform)
test_data = datasets.ImageFolder(root='DS_IDRID/Test', transform=transform)

def get_indices(file_list, dataset):
    indices = []
    dataset_files = [os.path.basename(path) for path, _ in dataset.samples]

    for file in file_list:
        if file in dataset_files:
            index = dataset_files.index(file)
            indices.append(index)
    return indices

train_indices = get_indices(train_DR, train_data) + get_indices(train_Non_DR,
train_data)
test_indices = get_indices(test_DR, test_data) + get_indices(test_Non_DR,
test_data)
```

```
def run_cnn(num_epochs = 10, learn_rate = 0.001, batch_size = 32):
    train_sampler = SubsetRandomSampler(train_indices)
    test_sampler = SubsetRandomSampler(test_indices)

    train_loader = DataLoader(train_data, batch_size=batch_size,
sampler=train_sampler)
    test_loader = DataLoader(test_data, batch_size=batch_size,
sampler=test_sampler)

    model = models.resnet18(weights=ResNet18_Weights.DEFAULT)

    num_features = model.fc.in_features
    model.fc = nn.Linear(num_features, 2)
    model = model.to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=learn_rate)

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)

            outputs = model(images)
            loss = criterion(outputs, labels)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        accuracy = 100 * correct / total
    model.eval()
    y_true = []
    y_pred = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)

            outputs = model(images)
            _, predicted = torch.max(outputs, 1)
            y_true.extend(labels.cpu().numpy())
            y_pred.extend(predicted.cpu().numpy())

    accuracy = accuracy_score(y_true, y_pred)
    sensitivity = recall_score(y_true, y_pred, pos_label=1)  # DR is positive
class
    specificity = recall_score(y_true, y_pred, pos_label=0)  # Non-DR is
negative class

    print(f"Number of Epochs: {num_epochs}")
    print(f"Learning Rate: {learn_rate}")
    print(f"Mini Batch Size: {batch_size}")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Sensitivity: {sensitivity * 100:.2f}%")
    print(f"Specificity: {specificity * 100:.2f}%")
    conf_matrix = confusion_matrix(y_true, y_pred)
    print("Confusion Matrix:")
    print(conf_matrix)

    with open("output.txt", "a") as file:
        file.write(f"Number of Epochs: {num_epochs}\n")
        file.write(f"Learning Rate: {learn_rate}\n")
        file.write(f"Mini Batch Size: {batch_size}\n")
        file.write(f"Accuracy: {accuracy * 100:.2f}%\n")
        file.write(f"Sensitivity: {sensitivity * 100:.2f}%\n")
        file.write(f"Specificity: {specificity * 100:.2f}%\n")
        file.write("Confusion Matrix:")
        file.write(np.array2string(conf_matrix))
        file.write("\n\n")

run_cnn() #default values
run_cnn(8) #lower epochs
run_cnn(12) #higher epochs
run_cnn(learn_rate=0.002) #higher learning rate
run_cnn(learn_rate=0.0009) #lower learning rate
run_cnn(batch_size=20) #smaller batch size
run_cnn(batch_size=40) #larger batch size
```