**University of Central Missouri**

**Department of Computer Science & Cybersecurity**

**CS5720 Neural network and Deep learning**

**Spring 2025**

**Home Assignment 5. (Cover Ch 11, 12)**

**Student name: Yaswanth Polarouthu**

**Submission Requirements:**

* Total Points: 100
* Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
* Submit your GitHub link and video on the BB.
* Comment your code appropriately ***IMPORTANT.***
* Make a simple video about 2 to 3 minutes which includes demonstration of your home assignment and explanation of code snippets.
* Any submission after provided deadline is considered as a late submission.

**1. GAN Architecture**

Explain the adversarial process in GAN training. What are the goals of the generator and discriminator, and how do they improve through competition? Diagram of the GAN architecture showing the data flow and objectives of each component.

**2. Ethics and AI Harm**

Choose one of the following real-world AI harms discussed in Chapter 12:

* Representational harm
* Allocational harm
* Misinformation in generative AI

Describe a real or hypothetical application where this harm may occur. Then, suggest **two harm mitigation strategies** that could reduce its impact based on the lecture.

**3. Programming Task (Basic GAN Implementation)**

Implement a simple GAN using PyTorch or TensorFlow to generate handwritten digits from the MNIST dataset.

**Requirements**:

* Generator and Discriminator architecture
* Training loop with alternating updates
* Show sample images at Epoch 0, 50, and 100

**Deliverables**:

* Generated image samples
* Screenshot or plots comparing losses of generator and discriminator over time

**4. Programming Task (Data Poisoning Simulation)**

Simulate a data poisoning attack on a sentiment classifier.  
Start with a basic classifier trained on a small dataset (e.g., movie reviews). Then, poison some training data by flipping labels for phrases about a specific entity (e.g., "UC Berkeley").

**Deliverables**:

* Graphs showing accuracy and confusion matrix before and after poisoning
* How the poisoning affected results

**5. Legal and Ethical Implications of GenAI**

Discuss the legal and ethical concerns of AI-generated content based on the examples of:

* Memorizing private data (e.g., names in GPT-2)
* Generating copyrighted material (e.g., Harry Potter text)

Do you believe generative AI models should be restricted from certain data during training? Justify your answer.

**6. Bias & Fairness Tools**

Visit [Aequitas Bias Audit Tool](http://www.datasciencepublicpolicy.org/projects/aequitas/).  
Choose a bias metric (e.g., false negative rate parity) and describe:

* What the metric measures
* Why it's important
* How a model might fail this metric

**Optional**: Try applying the tool to any small dataset or use demo data.

ANSWERS

### **1. GAN Architecture**

In GANs, the **generator** creates fake data from random noise, while the **discriminator** tries to distinguish real from fake data. They compete: the generator improves by trying to fool the discriminator, and the discriminator improves by detecting fakes. Over time, both models become stronger until the fake data becomes indistinguishable from real.

### **2. Ethics and AI Harm –** Allocational Harm

**Example:** An AI hiring tool trained on biased data may reject qualified candidates from underrepresented groups.

**Mitigation Strategies:**

1. Remove or reduce bias in training data by balancing representation.
2. Use fairness metrics and auditing tools to evaluate model decisions across demographics.

**Q3: Basic GAN – Code (PyTorch)**

# Install required packages in Colab

# !pip install torch torchvision matplotlib

import torch

import torch.nn as nn

import torch.optim as optim

from torchvision import datasets, transforms

from torchvision.utils import save\_image

import matplotlib.pyplot as plt

import os

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

# Generator

class Generator(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.net = nn.Sequential(

nn.Linear(100, 256),

nn.ReLU(True),

nn.Linear(256, 512),

nn.ReLU(True),

nn.Linear(512, 784),

nn.Tanh()

)

def forward(self, x):

return self.net(x).view(-1, 1, 28, 28)

# Discriminator

class Discriminator(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.net = nn.Sequential(

nn.Flatten(),

nn.Linear(784, 512),

nn.LeakyReLU(0.2),

nn.Linear(512, 1),

nn.Sigmoid()

)

def forward(self, x):

return self.net(x)

# Training setup

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

train\_loader = torch.utils.data.DataLoader(datasets.MNIST('.', download=True, transform=transform), batch\_size=64, shuffle=True)

G = Generator().to(device)

D = Discriminator().to(device)

criterion = nn.BCELoss()

optimizer\_G = optim.Adam(G.parameters(), lr=0.0002)

optimizer\_D = optim.Adam(D.parameters(), lr=0.0002)

losses\_G, losses\_D = [], []

for epoch in range(101):

g\_loss\_epoch, d\_loss\_epoch = 0, 0

for i, (real, \_) in enumerate(train\_loader):

real = real.to(device)

batch\_size = real.size(0)

# Train Discriminator

D.zero\_grad()

real\_labels = torch.ones(batch\_size, 1, device=device)

fake\_labels = torch.zeros(batch\_size, 1, device=device)

noise = torch.randn(batch\_size, 100, device=device)

fake = G(noise)

out\_real = D(real)

out\_fake = D(fake.detach())

loss\_D = criterion(out\_real, real\_labels) + criterion(out\_fake, fake\_labels)

loss\_D.backward()

optimizer\_D.step()

# Train Generator

G.zero\_grad()

output = D(fake)

loss\_G = criterion(output, real\_labels)

loss\_G.backward()

optimizer\_G.step()

d\_loss\_epoch += loss\_D.item()

g\_loss\_epoch += loss\_G.item()

losses\_D.append(d\_loss\_epoch / len(train\_loader))

losses\_G.append(g\_loss\_epoch / len(train\_loader))

if epoch in [0, 50, 100]:

with torch.no\_grad():

sample = G(torch.randn(64, 100, device=device)).cpu()

save\_image(sample, f"epoch\_{epoch}.png", normalize=True, nrow=8)

# Plot loss

plt.plot(losses\_G, label="Generator Loss")

plt.plot(losses\_D, label="Discriminator Loss")

plt.legend()

plt.title("GAN Losses")

plt.xlabel("Epoch")

plt.ylabel("Loss")

plt.savefig("gan\_loss\_plot.png")

plt.show()

Output:

A black screen with white numbers

AI-generated content may be incorrect.

## **Q4: Data Poisoning – Code (Sentiment Classifier)**

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Original data

texts = [

"I love this movie", "It was a great film", "UC Berkeley is awesome",

"Terrible acting", "I hated it", "UC Berkeley ruined the movie"

]

labels = [1, 1, 1, 0, 0, 0] # 1 = positive, 0 = negative

# Vectorize

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(texts)

# Train baseline classifier

model = LogisticRegression()

model.fit(X, labels)

preds = model.predict(X)

acc\_before = accuracy\_score(labels, preds)

cm\_before = confusion\_matrix(labels, preds)

# Poison the data (flip sentiment for UC Berkeley)

texts\_poisoned = [

"I love this movie", "It was a great film", "UC Berkeley is awesome",

"Terrible acting", "I hated it", "UC Berkeley ruined the movie"

]

labels\_poisoned = [1, 1, 0, 0, 0, 1] # Flip labels for "UC Berkeley"

X\_pois = vectorizer.fit\_transform(texts\_poisoned)

model\_pois = LogisticRegression()

model\_pois.fit(X\_pois, labels\_poisoned)

preds\_pois = model\_pois.predict(X\_pois)

acc\_after = accuracy\_score(labels\_poisoned, preds\_pois)

cm\_after = confusion\_matrix(labels\_poisoned, preds\_pois)

# Plot confusion matrices

fig, axs = plt.subplots(1, 2, figsize=(10, 4))

sns.heatmap(cm\_before, annot=True, fmt='d', ax=axs[0], cmap="Blues")

axs[0].set\_title(f"Before Poisoning (Accuracy: {acc\_before:.2f})")

sns.heatmap(cm\_after, annot=True, fmt='d', ax=axs[1], cmap="Oranges")

axs[1].set\_title(f"After Poisoning (Accuracy: {acc\_after:.2f})")

plt.savefig("poisoning\_effect.png")

plt.show()

Output:

A screenshot of a graph

AI-generated content may be incorrect.

### **5. Legal and Ethical Implications of GenAI**

Generative AI can memorize and reproduce private or copyrighted data.  
**Yes, restrictions should exist.** This prevents privacy violations (e.g., names from training sets) and respects copyright laws (e.g., avoiding full reproduction of copyrighted books like Harry Potter). Responsible training practices are essential.

### **6. Bias & Fairness Tools –** False Negative Rate Parity

* **Measures:** Whether the false negative rate is equal across groups (e.g., gender, race).
* **Importance:** Prevents disadvantaged groups from being disproportionately denied opportunities (e.g., loans, parole).
* **Failure:** A model might fail if it predicts fewer positives for one group despite equal ground truth rates.