National University of Computer and Emerging Sciences

FAST School of Computing **Spring-2024** **Islamabad Campus**

AI-4009: Generative AI Sessional-II Exam

Date: 4th April, 2024

Course Instructor

Dr. Akhtar Jamil

Total Time: 1 Hour

Total Marks: 50

____________________ ____________ ____________ ________________

Student Name Roll No. Course Section Student Signature

Do not write anything on the question paper except the information required above.

Instructions:

1. Read the question carefully, understand the question, and then attempt your answers in the provided answer booklet.
2. Verify that you have **two (2)** printed pages including this page. There are **Four (4)** questions.
3. Calculator sharing is strictly prohibited.
4. Write concise answers where necessary

Q1: Write short answers to the following questions [10 x 2 = 20]

1. Why do latent variable models approximate the expected log-likelihood rather than computing the actual probability directly?

In latent variable models, directly calculating the actual probability of the observed data involves integrating over all possible values of the latent variables, which can be mathematically intractable or computationally prohibitive, especially in high-dimensional spaces. This integration is necessary because the latent variables are not directly observed, yet they influence the generation of the observed data. The true likelihood function of the observed data thus involves summing or integrating over these hidden variables to account for all their possible configurations.

To manage this complexity, we approximate the expected log likelihood instead of calculating the actual probability. This approximation makes the problem more tractable by allowing us to work with simpler forms that can be efficiently computed.

2. What will be the impact if the KL Divergence between $q_{\emptyset}(z|x)$ and $P(z)$ is high?

If the distance between two terms is too high, then the model will generate garbage images if a random Z is taken as input to generate an image.

3. Explain the concept of uniform dequantization in the context of applying flow-based models.

Uniform dequantization is a technique used to adapt flow-based models for discrete data. Flow-based models, which are designed to model distributions of continuous data, rely on the ability to perform exact density estimation and to invertibly map between data spaces. However, many types of data

encountered in practice, like images, are inherently discrete, with pixel values typically represented as integers within a certain range (e.g., 0 to 255 for 8-bit images).
The process of uniform dequantization involves adding a small amount of uniform noise to each discrete data point. Specifically, for a discrete data point y, noise u sampled from a uniform distribution U over an interval [0,1] or [−0.5, 0.5] is added to y to produce a continuous variable x=y+u. This noise addition effectively spreads the discrete data points across the continuous interval between their original integer values, smoothing the data distribution and making it continuous.

4. Why VAEs generally generate blurry images as output?

Variational Autoencoders (VAEs) tend to generate blurry image outputs primarily due to their underlying objective function, which balances reconstruction accuracy with a regularization term that encourages the learned latent space to follow a specific distribution, typically a Gaussian. This regularization term, which promotes the smoothing of the latent space to ensure a continuous and complete representation, often leads to the averaging of similar data points. When generating new samples, the decoder part of the VAE thus tends to produce outputs that are averages of similar training examples, resulting in images that lack the sharpness and detail of the original data. Additionally, the use of a pixel-wise loss function, such as mean squared error, in the reconstruction objective can further exacerbate the blurriness by emphasizing the overall structure at the expense of high-frequency details.

5. Why GANs are considered to be robust against the overfitting problem?

Since we do not feed the real data to the generator, it reduces the risk of memorizing the training dataset, thereby enhancing the model's generalization capabilities.

6. Can we use all available labels in the dataset to train a discriminator in the GAN model or it is always designed to be binary (to distinguish between fake or real)? Explain.

Yes, it's possible to extend beyond binary classification in more complex GAN variants, incorporating multiple labels or attributes into the training process. Using all available labels in the dataset to train a discriminator in a GAN model can enrich the learning process, enabling the generation of more diverse and high-quality data. It can also help the discriminator become more robust by giving it a deeper understanding of the data's underlying structure and characteristics.

7. How image de-duplication process can help decrease the likelihood that GAN memorizes and directly replicates its training images?

Image de-duplication is a process that removes duplicate or highly similar images from a dataset. In the context of training Generative Adversarial Networks (GANs), de-duplication plays a crucial role in promoting the generation of novel images and reducing the likelihood that the GAN simply memorizes and replicates its training images. Here’s how image de-duplication helps in this context:
- Enhances generalization by forcing the GAN to learn broader dataset features instead of memorizing specific images.
- Reduces overfitting by removing bias towards repeated patterns, helping the model to better generalize to unseen data.
- Improves model robustness by presenting a more challenging and varied set of training examples, enhancing discriminator accuracy.

- Encourages creative generation by pushing the generator to explore the dataset's underlying space and produce varied outputs.
- Prevents mode collapse by ensuring a broad representation of the dataset's variance, encouraging diversity in generated images.

8. How semantic hashing is performed in the image de-duplication process?

Autoencoder compresses and then reconstructs the images, helping to remove noise and unnecessary details.
Binarization and semantic hashing:
After training, the latent spaces are used to represent each image.
Z are made binary (0 or 1) by thresholding: values above the threshold are set to 1, and those below are set to 0.
Result of binarization is like semantic hashing where similar images are likely to have similar binary codes, allowing for efficient comparison and deduplication.

9. Consider a Maxout layer that has 12 units with 4 pieces. Calculate the output of Maxout layer (y) when the following input is fed to it.

$$\boldsymbol{x} = [3, -1, 2, 6, 4, 5, -2, 0, 1, 7, 9, 8]$$
$$y = [3, 6, 1, 9]$$

10. How Cycle Consistency Losses can be calculated in CycleGANs? Write its formulation.

CycleGANs consist of two mapping functions, $(G: X \rightarrow Y)$ and $(F: Y \rightarrow X)$, where (G) attempts to translate images from domain (X) to domain (Y), and (F) translates images from domain (Y) to domain (X). The cycle consistency loss consists of two parts:

1. Forward cycle consistency loss is defined as:

$$\mathrm{L}(G, F, \mathrm{Dx}, \mathrm{X}, \mathrm{Y}) = E_{x \sim p_{\text{data}}(x)}\left[|F(G(x)) - x|_1\right]$$

2. Backward cycle consistency loss is defined as:

$$\mathrm{L}(G, F, \mathrm{Dy}, \mathrm{Y}, \mathrm{X}) = E_{y \sim p_{\text{data}}(y)}\left[|G(F(y)) - y|_1\right]$$

The total cycle consistency loss is the sum of both the forward and backward cycle consistency losses:

$$\mathrm{L}cycle(G, F) = \mathrm{L}(G, F, \mathrm{Dy}, \mathrm{Y}, \mathrm{X}) + \mathrm{L}(G, F, \mathrm{Dx}, \mathrm{X}, \mathrm{Y})$$

Q2: [5+5]

a) Given a 4x4 image of 3 bits as shown below. Calculate the entropy of this image.
Hint: Calculate histogram.

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

x_i	r_i	Prob(x_i)
0	2	2/16 = 0.125
1	2	2/16 = 0.125
2	2	2/16 = 0.125
3	2	2/16 = 0.125

4	**2**	**2/16 = 0.125**
5	**2**	**2/16 = 0.125**
6	**2**	**2/16 = 0.125**
7	**2**	**2/16 = 0.125**

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)]$$

$$H(X) = -\sum_{i=1}^{8} 0.125 \log_2(0.125)$$
$$H(X) = -8 \times 0.125 \times (-3) = 3$$

b) What are Variational Autoencoders. How can we train a VAE and then use it for classification task?

Variational Autoencoders (VAEs) are a class of generative models. VAEs learn the parameters of probability distributions representing the data in a latent space. This allows VAEs to generate new data points similar to the ones in the training set.

A VAE consists of two main components: an encoder and a decoder.

Encoder: This part of the model takes an input x and encodes it into a latent space representation z. The encoder outputs parameters (mean μ and variance σ) of a Gaussian distribution representing possible values in the latent space.

Decoder: The decoder part takes a sampled point from the latent space and attempts to reconstruct the original input x. The goal of the reconstruction process is to be as accurate as possible, which trains the model to learn a meaningful representation of the data.

Training a VAE involves optimizing both the encoder and the decoder. The loss function is a combination of both Reconstruction Loss and KL Divergence

Classification
Once the VAE is trained, you can use the encoder part of the VAE as feature extractor that can serve as input for classification tasks. For classification, you can train a separate classifier on the latent representations produced by the encoder. Depending on the performance, you might need to fine-tune the classifier or the entire model by adjusting hyperparameters to improve the classification accuracy.

Q-3: [5+5]

a) Consider a corpus containing the following sentences (documents):

1. The quick brown fox jumps over the lazy dog.

2. Lazy foxes lie low.
3. The quick yellow bird flies high.
4. High and low, the bird flies.
5. A quick bird jumps over lazy dogs.

Calculate the following:
TF("quick", Document1)
IDF("quick", Corpus)
TF-IDF("quick", Document1, Corpus)

TF("quick", Document 1) = 1/9 = 0.111
IDF("quick", Corpus) = log(5/3) ≈ 0.511
TF-IDF("quick", Document1, Corpus) = TF x IDF ≈ 0.057

b) Explain the working of Mini Batch GANs. What problem this type of GAN actually tackles that is generally available in standard GANs?

Working of Mini Batch GANs
Mini Batch Generative Adversarial Networks (GANs) adjust the standard GAN framework to improve the learning process, specifically addressing common issues like mode collapse and training instability. The core innovation in Mini Batch GANs is in how the discriminator processes information.

Mini-Batch Discrimination Technique
Mini Batch GANs incorporate a technique known as mini-batch discrimination. This technique allows the discriminator to look at multiple examples (a mini-batch) at once, rather than making decisions based on single samples. The idea is to give the discriminator context about the diversity (or lack thereof) of samples it's evaluating, helping it to distinguish between real and fake batches more effectively.

Calculating a Diversity Score
The discriminator calculates a score that reflects the diversity of the samples in a mini-batch. If the generator is producing varied and realistic samples, the diversity score will be higher, indicating a batch of samples that resembles the variation seen in real data. Conversely, a low diversity score suggests that the generator's outputs are too similar to each other, signaling a problem like mode collapse.

Based on the diversity score and the discriminator's ability to distinguish real from fake samples considering batch context, the feedback to the generator is adjusted. The generator then uses this feedback to update its parameters, aiming to produce more diverse and realistic samples in the next iteration.

Mini Batch GANs Solution:
The primary issue with standard GANs is the lack of diversity in the generated samples, often referred to as mode collapse. In standard GANs, the generator may learn to produce only a small set of highly realistic outputs that consistently fool the discriminator, neglecting the variety present in the real data distribution. This leads to poor generalization and limits the usefulness of the generated data.

Mini Batch GANs considering multiple instances within a mini-batch, the discriminator becomes more adept at recognizing small differences and patterns across a wider range of data. This forces the generator to create more varied outputs to successfully fool the discriminator.

Q-4: [5+5]

a) Write down at least three limitations of CycleGAN titled "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"

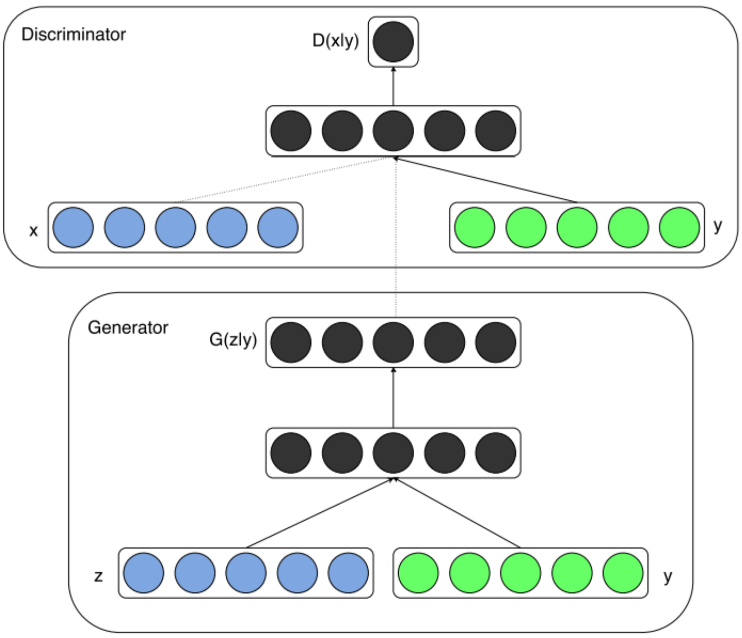
a) The model was trained on specific synsets (wild horse and zebra) from ImageNet, which does not include images of a person riding a horse or zebra. This limitation in the diversity of the training data can restrict the model's generalization ability to unseen or varied scenarios.

b) The method may incorrectly swap labels, such as tree and building labels, in the output of tasks like photos→labels. This indicates a challenge in maintaining semantic consistency without explicit paired guidance.

c) Although unpaired data is abundantly available, solely relying on it can limit achieving the high precision and reliability of model.

b) With the help of a diagram explain the working of conditional GANs. Write their objective function.

- **Conditional Generative Adversarial Nets** (Conditional GANs) are an extension of the original Generative Adversarial Networks (GANs) framework
- It incorporates conditional information into the data generation process.
- Both the generator and discriminator are provided with additional conditional data
  - class labels or part of data features
- This allows the generated data to be more specific to the given condition
  - More controlled and diverse data generation.



- Generator:
  - The generator G takes a noise vector z and conditional information y to produce data G(z|y)
  - Not only produces realistic output but also matches the given condition.
- **Discriminator**:
  - The discriminator (D) also receives the conditional information y alongside the real data or the generated data from the generator.
  - Its task is to determine whether the given data is real or fake and whether it corresponds to the given condition.
  - The discriminator assesses D(x, y), where (x) is either real or generated data.
- **Objective Function**:
  - The loss function encourages the generator to create data that can fool the discriminator into believing it is real and correctly conditioned.
  - Distinguish between real and fake data and also ensure that the generated data adheres to the conditional context.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$