

## DL – Unit 5 (Deep Generative Models) – END-SEM PYQ Answers

### MAY-JUNE 2023

**Q5a** Explain Boltzmann machine in details.

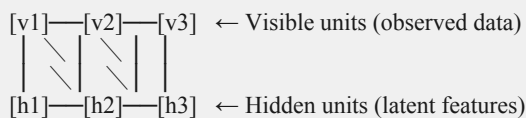
[6 Marks]

#### Boltzmann Machine

A Boltzmann Machine is a type of stochastic (probabilistic) recurrent neural network introduced by Hinton and Sejnowski in 1985. It is an energy-based model that learns the probability distribution over its inputs. The name comes from the Boltzmann distribution from statistical mechanics, which is used to define the probability of each state of the network.

#### Architecture

Boltzmann Machine (fully connected):



All units are binary (0 or 1).

Connections are bidirectional and symmetric ( $w_{ij} = w_{ji}$ ). Visible-visible AND hidden-hidden connections exist.

#### Energy Function

$$E(v, h) = -\sum_i \sum_j w_{ij} \cdot v_i \cdot h_j \\ - \sum_i b_i \cdot v_i \\ - \sum_j c_j \cdot h_j$$

$w_{ij}$  = weight between unit  $i$  and unit  $j$

$b_i$  = bias of visible unit  $i$

$c_j$  = bias of hidden unit  $j$

Probability of a state:  $P(v, h) = (1/Z) \cdot \exp(-E(v, h))$   $Z$  = partition function (normalisation constant)  $= \sum_{\{v, h\}} \exp(-E(v, h))$

#### Objectives

- **Learn the distribution:** Adjust weights  $W$  so that the model assigns high probability  $P(v)$  to configurations that resemble training data.
- **Maximise log-likelihood:** Training minimises the energy of training data configurations and increases the energy of configurations the model generates (fantasy particles).
- **Feature discovery:** Hidden units learn to represent latent features of the data without supervision.

#### Problems with Full Boltzmann Machines

- **Intractable partition function:** Computing  $Z$  requires summing over all  $2^{(n_v + n_h)}$  possible states — exponential in the number of units.
- **Slow training:** Exact gradient requires computing statistics over all possible states, which is NP-hard. Contrastive Divergence (CD) is used as an approximation.
- **Restricted Boltzmann Machine (RBM):** Removes hidden-to-hidden and visible-to-visible connections, making inference tractable via block Gibbs sampling.

*Note: The Restricted Boltzmann Machine (RBM) is the practical version used in modern deep learning. Stacking multiple RBMs gives a Deep Belief Network (DBN). RBMs are trained layer by layer greedily, and the*

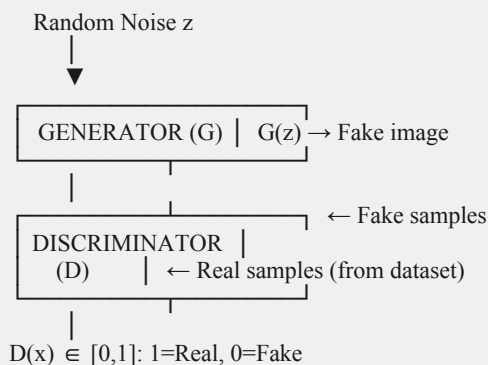
resulting weight initialisation is used to fine-tune the full network via backpropagation.

**Q5b** Explain GAN (Generative Adversarial Network) architecture with an example. **[6 Marks]**

### GAN — Generative Adversarial Network

GANs were introduced by Ian Goodfellow et al. in 2014. The core idea is to frame generative modelling as a two-player minimax game between two neural networks: a Generator (G) that produces fake data, and a Discriminator (D) that tries to distinguish real data from fake. The adversarial dynamic drives both networks to improve until G produces samples indistinguishable from real data.

#### Architecture



G tries to maximise  $\log(D(G(z)))$  → fool D. D tries to maximise  $\log(D(x)) + \log(1-D(G(z)))$  → distinguish real vs fake

#### Minimax Objective

$$\min_G \max_D V(D,G) = E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$
  
 At Nash Equilibrium:  $P_{\text{generated}} = P_{\text{real}}, D(x) = 0.5$  everywhere

#### Training Process

- **Step 1 — Train Discriminator:** Fix G; feed a batch of real images (label=1) and a batch of G(z) (label=0) to D. Update D's weights to correctly classify both.
- **Step 2 — Train Generator:** Fix D; sample noise z and generate G(z). Pass to D and compute the loss as if these were real (label=1). Update G's weights to fool D.
- **Repeat:** Alternate Steps 1 and 2 until D outputs ~0.5 for all samples (cannot distinguish real from fake).

#### Example — Handwritten Digit Generation (MNIST)

- **Generator:** Receives 100-dimensional Gaussian noise vector z. Passes through 3 fully-connected (or transposed-convolutional) layers to produce a 28×28 pixel image.
- **Discriminator:** Receives a 28×28 image. Passes through 3 layers and outputs a single probability (real/fake).
- **After training:** G learns to produce sharp, realistic handwritten digits without ever seeing them during G's training — only D's gradient signal guides it.

*Note: Common GAN problems: Mode collapse (G learns to generate only a few modes of the data distribution); Training instability (loss oscillates rather than converging). Solutions: Wasserstein GAN (WGAN) uses Earth Mover distance as loss; progressive growing of GANs (ProGAN) for high-resolution images; spectral*

normalisation for D.

**Q5c** Do GANs find real or fake images? If yes explain it in detail.

[6 Marks]

### How GANs Determine Real vs. Fake Images

This is fundamentally the role of the Discriminator network D. The Discriminator is a binary classifier trained to output a high probability (close to 1) when it sees a real image from the training dataset, and a low probability (close to 0) when it sees a fake image produced by the Generator.

#### Discriminator's Decision Process

Real image  $x \rightarrow [\text{Conv Layer 1}] \rightarrow [\text{Conv Layer 2}] \rightarrow [\text{Dense}] \rightarrow D(x) \approx 1.0$

Fake image  $G(z) \rightarrow [\text{Conv Layer 1}] \rightarrow [\text{Conv Layer 2}] \rightarrow [\text{Dense}] \rightarrow D(G(z)) \approx 0.0$

Loss for D:  $L_D = -[\log D(x_{\text{real}}) + \log(1 - D(G(z)))]$  = Cross-entropy of real=1 prediction + fake=0 prediction

#### What the Discriminator Learns

- **Early training:** The Discriminator easily tells apart blurry, noisy fake images from sharp real images, achieving near-perfect accuracy.
- **Mid training:** As G improves, D's task becomes harder. It must detect subtle artifacts in generated images — texture inconsistencies, unnatural blending, weird backgrounds.
- **Equilibrium:** Ideally, G generates images so realistic that D cannot do better than random guessing (50%/50%), meaning the generated distribution perfectly matches the real distribution.

#### Features the Discriminator Uses to Detect Fakes

- Statistical texture patterns that differ between real photos and generated images.
- High-frequency details (sharpness, fine edges) that generators often smooth over.
- Semantic consistency — e.g., faces with asymmetric eyes, fingers with wrong counts.
- Global structure vs. local detail coherence — GANs sometimes produce locally realistic but globally inconsistent images.

*Note: In modern GAN variants (StyleGAN2, BigGAN), the discriminator uses techniques like minibatch discrimination (comparing statistics across a batch to prevent mode collapse) and spectral normalization (constraining the Lipschitz constant of D to stabilize training). The discriminator's learned representations are also useful as feature extractors for downstream tasks.*

**Q6a** Differentiate generative and discriminative models in GAN.

[6 Marks]

### Generative vs. Discriminative Models

Property	Generative (G) vs Discriminative (D) in GAN
Goal	Generator: Learn $P(x)$ — the data distribution — to synthesise new samples. Discriminator: Learn $P(y x)$ — the conditional probability of a label given input — to classify real vs. fake.
Input	Generator: Random noise vector $z$ (usually Gaussian or Uniform). Discriminator: An image $x$ (either real or generated).
Output	Generator: A synthesised data sample $G(z)$ (e.g., a $256 \times 256$ image). Discriminator: A

	scalar probability $D(x) \in [0,1]$ indicating likelihood of being real.
Architecture	Generator: Upsampling network — uses transposed convolutions to expand noise into a full image. Discriminator: Downsampling classifier — uses standard convolutions to reduce an image to a scalar.
Objective	Generator: Minimise $\log(1 - D(G(z)))$ i.e. maximise the chance D thinks it produced a real image. Discriminator: Maximise $\log D(x_{\text{real}}) + \log(1 - D(G(z)))$ i.e. correctly label real and fake.
Analogy	Generator: A forger creating counterfeit currency. Discriminator: A detective trying to detect counterfeit currency.

*Note: Outside the GAN context: generative models (VAE, Boltzmann machines, normalizing flows) model the joint distribution  $P(x, y)$  and can generate new samples. Discriminative models (CNN classifier, SVM, logistic regression) only model  $P(y|x)$  and cannot generate data. GANs bring both together in an adversarial framework.*

**Q6b** What are applications of GAN? Explain any four in detail.

**[6 Marks]**

### Applications of Generative Adversarial Networks

- **1. Image Synthesis and Super-Resolution:** GANs generate photorealistic images from scratch. SRGAN (Super-Resolution GAN) upscales low-resolution images to 4× higher resolution by learning to hallucinate plausible high-frequency details. This is used in satellite imagery enhancement, medical scan upscaling, and entertainment (upsampling old films).
- **2. Image-to-Image Translation (pix2pix, CycleGAN):** Conditional GANs learn mappings between image domains. Examples: converting satellite maps to street maps, turning sketches into photorealistic faces, converting daytime images to nighttime, and horse-to-zebra style transfer. CycleGAN achieves this without paired training examples by using cycle-consistency loss.
- **3. Data Augmentation for Imbalanced Datasets:** In medical imaging, rare diseases may have very few training examples. GANs can generate synthetic training images of rare conditions (e.g., diabetic retinopathy, skin lesions) to balance the dataset and improve classifier performance, reducing the reliance on expensive data collection.
- **4. Deepfakes and Face Generation:** StyleGAN and Progressive GAN generate photorealistic human faces that do not exist. Applications include anonymizing training data (privacy-preserving ML), creating virtual avatars, game character generation, and film production (de-aging actors). This also raises serious ethical concerns about misinformation.
- **5. Drug Discovery and Molecular Generation:** GANs (e.g., MolGAN) generate novel molecular structures with desired properties (binding affinity, solubility, toxicity) by learning the distribution of known drug-like molecules. This accelerates pharmaceutical discovery significantly.
- **6. Text-to-Image Synthesis:** GANs conditioned on text descriptions generate images matching those descriptions. DALL-E (earlier versions) and GigaGAN are examples. This enables creative tools where users describe scenes in natural language and the system renders them.

*Note: GANs are also used in video generation (VideoGAN), 3D object generation (3D-GAN), music synthesis (MuseGAN), and anomaly detection (training a GAN on normal data and flagging inputs that the discriminator identifies as anomalous).*

**Q6c** Write Short Note on Deep generative model and Deep Belief Networks.

**[6 Marks]**

## Deep Generative Models

A deep generative model is a probabilistic model that uses a deep neural network to learn the underlying probability distribution  $P(x)$  of the training data. Once trained, the model can generate new samples from that distribution. The key varieties include:

- **Variational Autoencoders (VAE):** Encode input into a latent distribution (mean and variance), sample from it, and decode back. Training maximises the Evidence Lower Bound (ELBO). VAEs produce continuous, smooth latent spaces ideal for interpolation.
- **Generative Adversarial Networks (GAN):** Covered in detail above. Use adversarial training.
- **Normalizing Flows:** Learn an invertible mapping between data and a simple distribution (e.g., Gaussian). Allow exact likelihood computation. Examples: RealNVP, Glow.
- **Diffusion Models:** Learn to reverse a gradual noise-adding process. State-of-the-art for image generation (DALL-E 2, Stable Diffusion, Midjourney).

## Deep Belief Networks (DBN)

A Deep Belief Network (DBN) is a generative probabilistic model composed of multiple layers of stochastic, latent variables. It was introduced by Hinton et al. in 2006 and was one of the breakthrough models that reignited interest in deep learning.

### Architecture

DBN Structure (3 hidden layers):

```
[Visible Layer] ← input data (e.g., image pixels)
  ↓ (RBM 1 — trained first)
[Hidden Layer 1] ← learns low-level features (edges)
  ↓ (RBM 2 — trained second)
[Hidden Layer 2] ← learns mid-level features (shapes)
  ↓ (RBM 3 — trained third)
[Hidden Layer 3] ← learns high-level features (objects)
```

Top two layers form an undirected RBM. Lower layers form a directed generative model (sigmoid belief net).

### Training Procedure — Greedy Layer-wise Pre-training

- **Step 1:** Train the first RBM on the raw input data. The trained hidden layer becomes the 'visible' layer for the next RBM.
- **Step 2:** Train the second RBM on the hidden representations from Step 1. Repeat for all layers.
- **Step 3:** Fine-tune the entire network using backpropagation with the greedy pre-training weights as initialisation. This was the key insight — greedy pre-training provides a good starting point in weight space.

### Why DBNs Were Important

- Demonstrated that deep networks could be trained effectively by initialising with unsupervised pre-training — solving the vanishing gradient problem of the time.
- Showed that deep architectures learn hierarchical representations (edges → shapes → objects).
- Sparked the modern deep learning renaissance (2006–2012), leading up to AlexNet (2012).

*Note: DBNs have largely been superseded by modern architectures trained end-to-end with batch normalisation, ReLU activations, and dropout — which can be trained from random initialisation without greedy pre-training. However, the hierarchical representation learning insight from DBNs remains foundational to our understanding of why deep learning works.*

## NOV-DEC 2023

**Q5a** Explain Deep generative model with example.

[6 Marks]

**[REPEATED]** Deep Generative Models are covered in Q6c of May-June 2023. Key examples: VAE, GAN, Normalizing Flows, Diffusion Models. Please refer to that section.

**Q5b** How does GAN training scale with batch size?

[6 Marks]

### GAN Training and Batch Size

Batch size is a critical hyperparameter in GAN training that affects both training stability and output quality. Unlike supervised learning, where larger batches generally converge faster, GAN training has a more nuanced relationship with batch size.

- **Discriminator estimate quality:** A larger batch provides the discriminator with a better estimate of the true data distribution at each step, making D's gradient signal to G more accurate. This generally improves training stability.
- **Mode collapse risk:** Smaller batches introduce more gradient noise, which can sometimes help G explore different modes of the distribution — preventing mode collapse. Very large batches can accelerate mode collapse by making D's signal too precise early in training.
- **Memory constraints:** GAN training stores both G and D in GPU memory along with gradients for both. Effective batch size is often smaller than in supervised learning for the same GPU.
- **Minibatch Discrimination:** A technique where D is given statistics computed across the batch (standard deviation of features) to detect mode collapse. This only works with batch sizes large enough to compute meaningful statistics.
- **BigGAN scaling insight:** BigGAN (2019) found that very large batch sizes (2048+) with large model capacity significantly improved image quality and diversity on ImageNet. Large batches allowed the model to see more diverse training samples per gradient update.

*Note: The practical recommendation is to use the largest batch size that fits in GPU memory, combine with techniques like spectral normalisation and gradient penalty (WGAN-GP) for stability, and monitor both the generator and discriminator losses to detect divergence or mode collapse early.*

**Q5c** List the applications of GAN network with description.

[6 Marks]

**[REPEATED]** Applications of GAN are covered comprehensively in Q6b of May-June 2023. Refer to that section for six detailed applications including image synthesis, image-to-image translation, data augmentation, deepfakes, drug discovery, and text-to-image synthesis.

**Q6a** Draw and explain architecture of Boltzmann machine.

[6 Marks]

**[REPEATED]** Boltzmann Machine architecture and explanation are covered in Q5a of May-June 2023, including the energy function, probability formula, and RBM as the practical variant.

**Q6b** Explain different types of GAN.

[6 Marks]



## Types of Generative Adversarial Networks

- **Vanilla GAN (Original GAN):** The original architecture by Goodfellow (2014). Generator and discriminator are simple fully-connected MLPs. Trained on MNIST. Suffers from mode collapse and training instability.
- **DCGAN (Deep Convolutional GAN):** Replaces fully-connected layers with convolutional layers (in D) and transposed convolutional layers (in G). Uses batch normalisation and LeakyReLU in D. Much more stable and produces higher quality images than vanilla GAN.
- **Conditional GAN (cGAN):** Both G and D receive an additional class label (or any conditioning information) as input. Allows controlled generation — e.g., 'generate a cat image' vs 'generate a dog image'. Conditioning is achieved by concatenating the label to the input.
- **CycleGAN:** Uses two GANs (G: domain A→B and F: domain B→A) with a cycle-consistency loss ( $F(G(x)) \approx x$ ). Enables image-to-image translation without paired training examples.
- **StyleGAN / StyleGAN2:** Controls image style at different levels of detail by injecting the latent code into the generator at multiple points using adaptive instance normalisation (AdaIN). Produces state-of-the-art photorealistic face generation. Introduced W and W+ latent spaces.
- **Progressive GAN (ProGAN):** Starts training at low resolution (4×4) and progressively adds layers to both G and D to grow resolution to 1024×1024. Dramatically stabilises high-resolution training.
- **WGAN (Wasserstein GAN):** Replaces the binary cross-entropy loss with the Wasserstein (Earth Mover) distance. Provides meaningful gradients even when G and D distributions have no overlap. The discriminator becomes a 'critic' (not bounded to [0,1]). Requires weight clipping or gradient penalty (WGAN-GP).
- **InfoGAN:** Maximises mutual information between a subset of noise variables (c) and the generated output. Learns disentangled representations where each  $c_i$  controls a meaningful factor (e.g., digit class, rotation).

*Note: The evolution of GAN architectures reflects a progression from training stability (DCGAN, WGAN) to controllability (cGAN, InfoGAN, StyleGAN) to scalability (BigGAN, ProGAN). Modern image generation has largely shifted to Diffusion Models (Stable Diffusion), which are more stable to train and achieve higher diversity than GANs.*

**Q6c** Explain Deep Belief Network with diagram.

[6 Marks]

**[REPEATED]** Deep Belief Network is covered in full in Q6c of May-June 2023, including the layered architecture diagram, greedy layer-wise pre-training steps, and historical significance.

## MAY-JUNE 2024

**Q5a** State and explain different types of GAN.

[6 Marks]

**[REPEATED]** Types of GAN are covered comprehensively in Q6b of Nov-Dec 2023. Refer to that section for all eight types with explanations (Vanilla GAN, DCGAN, cGAN, CycleGAN, StyleGAN, ProGAN, WGAN, InfoGAN).

**Q5b** What is Boltzmann machine? Explain its objectives.

[6 Marks]

**[REPEATED]** Boltzmann Machine architecture and its objectives are covered in Q5a of May-June 2023. Key objectives: learning data distribution  $P(x)$ , maximising log-likelihood, and discovering latent features.

**Q5c** Write short Note on Deep generative model and Deep Belief Networks. [6 Marks]

**[REPEATED]** *Deep Generative Models and Deep Belief Networks are both covered in Q6c of May-June 2023. Refer to that section for VAE, GAN, Normalizing Flows, Diffusion Models, and the full DBN greedy pre-training procedure.*

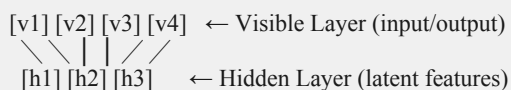
**Q6a** Define Boltzmann machine? State and Explain its types. [6 Marks]

### Boltzmann Machine — Types

A Boltzmann Machine is an energy-based stochastic neural network (see Q5a, May-June 2023 for the full definition). The key types are:

- **Full Boltzmann Machine:** All units (both visible and hidden) are connected to each other — both visible-visible and hidden-hidden connections exist. This makes inference computationally intractable (exponential in number of units) and is not used in practice.
- **Restricted Boltzmann Machine (RBM):** Connections exist only between the visible and hidden layers — no intra-layer connections. This bipartite structure makes the hidden units conditionally independent given the visible units (and vice versa), enabling efficient block Gibbs sampling for inference and training via Contrastive Divergence.

RBM Architecture:



NO connections within visible or hidden layers. Bipartite graph — enables efficient inference.

- **Deep Boltzmann Machine (DBM):** Multiple hidden layers, all connections are undirected (bidirectional). Allows top-down and bottom-up inference. More expressive than RBM but harder to train. Uses variational inference.
- **Convolutional RBM:** Uses convolutional filters instead of fully-connected weights, making it suited for image data. Preserves spatial structure and has fewer parameters.

**Q6b** Explain Discriminator network. [6 Marks]

### Discriminator Network in GANs

The Discriminator (D) is one of the two networks in a GAN. Its sole task is binary classification: given an image, decide whether it came from the real training dataset (label 1) or was generated by the Generator (label 0). The discriminator's gradient signal is what drives the Generator to improve.

#### Architecture (for image data)

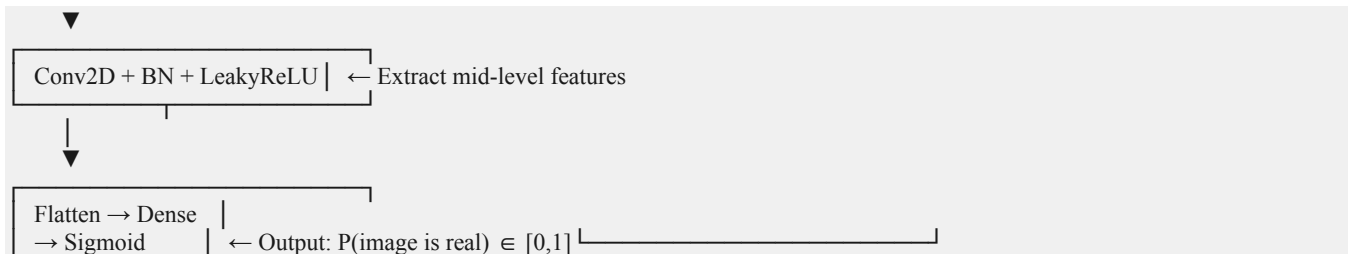
Input image (real or fake)



Conv2D + LeakyReLU  
stride=2, no padding

← Extract low-level features  
(no pooling — strided conv)





### Key Design Choices

- **LeakyReLU instead of ReLU:** Prevents dying neuron problem and allows gradients to flow even for negative activations, critical for training stability.
- **No pooling layers:** Strided convolutions are used for down-sampling (following the DCGAN guideline) to let the network learn its own spatial reduction.
- **Batch Normalisation (except first and last layers):** Stabilises training but is not applied to the input or output layers to prevent information loss.
- **Output: Sigmoid vs no activation (WGAN):** Standard GAN uses sigmoid to output a probability. WGAN uses a linear output (critic score) to avoid gradient saturation.

### Loss Function

Standard GAN Loss for D:

$$L_D = -E[\log D(x_{\text{real}})] - E[\log(1 - D(G(z)))]$$

Minimising  $L_D \rightarrow D$  correctly predicts 1 for real, 0 for fake.

Gradient penalty (WGAN-GP) adds:

$$L_{GP} = \lambda \cdot E[\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1]^2 \text{ where } \hat{x} \text{ is a random interpolation between real and fake samples.}$$

*Note: The discriminator's weights are often trained for more steps (k steps) per generator step to keep D's estimate accurate and provide a strong learning signal to G. In practice, k=1 to 5 discriminator updates per generator update is common.*

**Q6c** Enlist and Explain applications of GAN.

**[6 Marks]**

**[REPEATED]** Applications of GAN are covered in Q6b of May-June 2023 with six detailed examples. Additional applications to note: Video generation, anomaly detection, music synthesis (MuseGAN), and 3D object generation (3D-GAN).

**MAY-JUNE 2025 [REPEATED]** All questions repeated.

**NOV-DEC 2025**

**Q5a** What is a Boltzmann Machine? Describe its structure and components.

**[6 Marks]**

**[REPEATED]** Boltzmann Machine structure and components are covered in Q5a of May-June 2023 (energy function, visible/hidden units, training objectives, RBM as practical variant) and in May-June 2024 Q6a (types: Full BM, RBM, DBM, Convolutional RBM).

**Q5b** List at least five real-world applications of GANs and describe any one in detail. **[6 Marks]**

**[REPEATED]** GAN applications are covered in Q6b of May-June 2023 with six detailed applications. Refer to that section for image synthesis/super-resolution, image-to-image translation (CycleGAN), data augmentation for medical imaging, deepfakes/face generation, drug discovery (MolGAN), and text-to-image synthesis.

**Q5c** Describe the difference between generative and discriminative phases in Deep Belief Networks (DBNs). [5 Marks]

### Generative vs Discriminative Phases in Deep Belief Networks

A Deep Belief Network (DBN) has a two-phase training procedure that exploits both unsupervised generative learning and supervised discriminative fine-tuning. Understanding the distinction between these two phases reveals why the greedy pre-training strategy was so groundbreaking in 2006 — it solved the initialisation problem that had plagued deep network training for a decade.

#### Phase 1 — Generative Pre-training (Unsupervised)

In the generative phase, each pair of adjacent layers is trained as a Restricted Boltzmann Machine (RBM) using Contrastive Divergence — an unsupervised algorithm that adjusts weights to make the model assign high probability to configurations that look like the training data.

Greedy layer-wise generative pre-training:

Step 1: Train RBM\_1 on raw input data (pixels).

RBM\_1 learns to reconstruct the input distribution  $P(v)$ .  
Hidden activations of RBM\_1 become 'data' for Step 2.

Step 2: Train RBM\_2 on the hidden representations from RBM\_1.

RBM\_2 learns higher-level features (shapes from edges).

Step 3: Train RBM\_3 on hidden representations from RBM\_2.

RBM\_3 learns even more abstract features (objects from shapes).

Objective at each step:  $\max P(v)$  = maximise log-likelihood of data.

This is generative because the model learns to MODEL the input distribution and can GENERATE new samples by sampling down from the top RBM.

- **What is learned:** A hierarchy of increasingly abstract features — edges → shapes → object parts — purely from the structure of the input data, with no labels. The top-level RBM forms an associative memory that models the joint distribution of the highest-level features.
- **Result:** The stacked RBMs provide an excellent initialisation for the full deep network — the weights are already in a region of parameter space that captures useful structure in the data, far better than random initialisation.

#### Phase 2 — Discriminative Fine-tuning (Supervised)

Once all layers have been pre-trained generatively, a classification layer (usually a softmax) is added on top, and the entire network is fine-tuned end-to-end using standard backpropagation with labeled data.

Discriminative fine-tuning:

[Input] → [Layer 1 (from RBM\_1)] → [Layer 2 (from RBM\_2)]  
→ [Layer 3 (from RBM\_3)] → [Softmax classifier]

$$\hat{y} = P(\text{class} \mid \text{input})$$

Loss: Cross-entropy  $L = -\sum y_i \cdot \log(\hat{y}_i)$

All weights updated via backpropagation — discriminative objective. The generative pre-training provides the starting point (initialisation).

- **What is optimised:** The discriminative conditional probability  $P(\text{label} \mid \text{input})$  — not the generative joint  $P(\text{input}, \text{label})$ . The model's goal shifts from modelling the data to correctly classifying it.

- **Why fine-tuning works so well:** The generatively pre-trained weights are already in a good part of parameter space. Fine-tuning only needs to make small adjustments to adapt these general representations to the classification task — it does not need to learn the representations from scratch.

### Why This Two-Phase Approach Was Revolutionary

- **Solved the vanishing gradient problem (2006 context):** Before batch normalisation and ReLU were widely used, training deep networks from random initialisation caused gradients to vanish in lower layers. Generative pre-training gave all layers meaningful starting weights, so gradients had something useful to work with from the start.
- **Unsupervised pre-training as regularisation:** The generative phase effectively constrains the model to learn representations that are useful for reconstructing the input — a form of regularisation that prevents overfitting even with limited labeled data.

*Note: In modern deep learning, DBN-style greedy pre-training is rarely used because batch normalisation, ReLU activations, residual connections, and large labeled datasets have made it unnecessary. However, the insight that unsupervised pre-training on unlabeled data can dramatically improve supervised fine-tuning lives on in modern self-supervised learning — BERT's masked language modelling pre-training is conceptually a direct descendant of DBN generative pre-training.*

**Q6a** What is the role of the discriminator in a GAN? What are the inputs and outputs of a discriminator network? **[6 Marks]**

**[REPEATED]** Discriminator network role, inputs (real/fake images), outputs (probability scalar), architecture, loss function, and training strategy are covered in full in May-June 2024 Q6b. Refer to that section.

**Q6b** Explain the following Terms: i) Deep Belief Network ii) Deep Generative Model **[6 Marks]**

**[REPEATED]** Deep Belief Network (DBN) is covered in Q6c of May-June 2023 with architecture diagram and greedy pre-training procedure. Deep Generative Models (VAE, GAN, Normalizing Flows, Diffusion Models) are covered in the same section and in ND2023 Q5a. Refer to those sections.

**Q6c** Discuss the role of GANs in anomaly detection. How do they help identify outliers in data? **[5 Marks]**

### GANs for Anomaly Detection

Anomaly detection is the task of identifying data points that deviate significantly from what is considered 'normal'. Traditional approaches require labeled examples of anomalies — which are by definition rare and expensive to collect. GANs offer an elegant unsupervised alternative: train the GAN exclusively on normal data, then use the trained model to flag inputs it considers unusual.

#### The Core Idea

When a GAN is trained on normal data, both the Generator and Discriminator become experts on what normal data looks like. An anomalous input — something the GAN has never seen during training — will behave very differently from a normal sample when passed through the trained model. This difference in behaviour is the anomaly signal.

#### Method 1 — Discriminator Score

- **How it works:** After training, the Discriminator D has learned to distinguish the training distribution (normal data) from the Generator's output. Any real input that receives a low D score (i.e., D says 'this doesn't look like training data') is flagged as anomalous.
- **Limitation:** The Discriminator is trained to distinguish real from generated — not to detect out-of-distribution real samples. Its scores may be unreliable for this purpose without additional training.

## Method 2 — Reconstruction Loss (AnoGAN approach)

AnoGAN Anomaly Score:

For a test image  $x_{\text{test}}$ :

1. Find the latent vector  $z^*$  that best reconstructs  $x_{\text{test}}$ :

$$z^* = \operatorname{argmin}_z \|x_{\text{test}} - G(z)\|$$

(optimise  $z$  via gradient descent — the Generator is frozen)

2. Anomaly score  $A(x) = (1-\lambda) \cdot \|x_{\text{test}} - G(z^*)\| + \lambda \cdot \|f(x_{\text{test}}) - f(G(z^*))\|$

where  $f(\cdot)$  = intermediate discriminator feature map

Normal sample:  $z^*$  found easily  $\rightarrow$  small reconstruction error  $\rightarrow$  low A    Anomaly: no  $z^*$  fits well  $\rightarrow$  large reconstruction error  $\rightarrow$  high A

The intuition is elegant — if the test sample is normal, the Generator (trained on normal data) can reproduce it well from some latent vector  $z^*$ . If it is anomalous, no  $z$  in the normal latent space maps to it, so the reconstruction error is large. Large reconstruction error flags the sample as an anomaly.

## Method 3 — BiGAN / ALI (Encoder-Decoder GANs)

- **Architecture:** These GANs train both a Generator  $G (z \rightarrow x)$  and an Encoder  $E (x \rightarrow z)$  simultaneously so that the Discriminator distinguishes  $(\text{real\_image}, E(\text{real\_image}))$  from  $(G(\text{noise}), \text{noise})$ . After training, the anomaly score is the reconstruction error  $\|x - G(E(x))\|$  — how well the network can reconstruct the input through the encode-generate cycle.
- **Advantage:** No iterative optimisation at test time — the encoder directly maps any input to a latent code, making inference fast enough for real-time applications.

## Real-World Applications of GAN-based Anomaly Detection

- **Medical imaging:** Train on healthy MRI/CT scans. Lesions, tumours, or rare pathologies appear as anomalies because they are not representable in the normal data's latent space.
- **Industrial defect detection:** Train on images of defect-free products on a production line. Defective products score high anomaly scores without any labeled defect examples.
- **Network intrusion detection:** Train on normal network traffic patterns. Intrusions produce traffic patterns outside the normal distribution, flagged by the discriminator or reconstruction score.
- **Financial fraud:** Train on legitimate transactions. Fraudulent transactions that deviate from normal spending patterns receive high anomaly scores.

*Note: GAN-based anomaly detection has largely been superseded by modern methods based on Normalizing Flows (exact likelihood estimation) and Transformer-based autoencoders (which give better reconstruction quality). However, the GAN approach remains valuable when you need a generative model anyway (e.g., for data augmentation) and want to reuse it for anomaly detection at no extra training cost. The key practical challenge is mode collapse — if the GAN fails to cover all modes of the normal distribution, it will flag rare-but-normal samples as anomalies, producing false positives.*