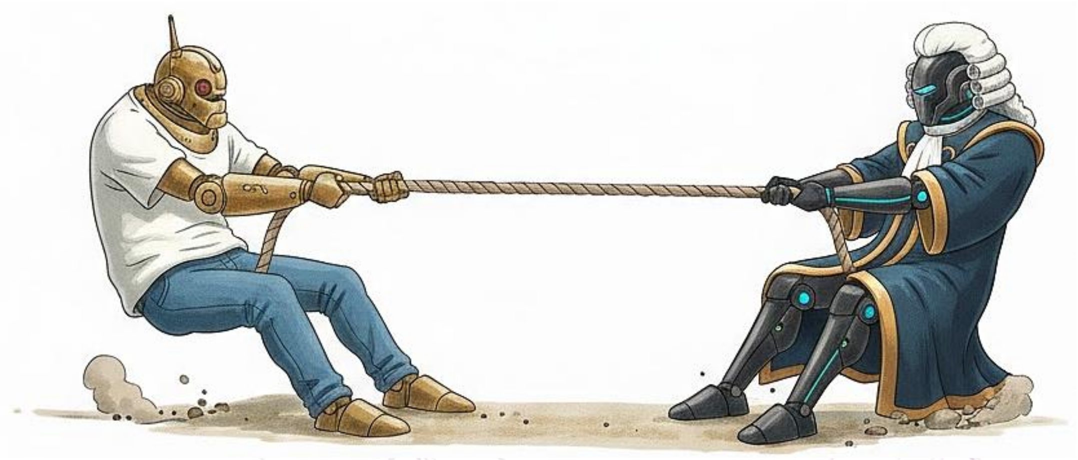
TAAP

INSTITUTO UNIVERSITÁRIO DE LISBOA



Advanced Topics in Deep Learning

MINI-PROJECT 01

IMPLEMENTING A CONDITIONAL GENERATIVE ADVERSARIAL NETWORK

António Cruz (140129)**, Cátia Brás** (120093)**, Ricardo Kayseller** (95813)

Table of Contents

1. Introduction

Generative Adversarial Networks (GANs) have become a fundamental class of generative models for learning complex data distributions and synthesizing realistic samples. In the classical (unconditional) GAN framework, the generator receives only random noise as input and learns to produce samples that resemble the training data, while the discriminator learns to distinguish real data from generated samples.

Although this setting allows the model to generate realistic images, it provides no control over the semantic content of the generated samples. In the context of datasets with labeled structure, such as MNIST, an unconditional GAN may generate any digit between 0 and 9, but the user cannot specify which digit should be produced.

Conditional Generative Adversarial Networks (cGANs) extend the original GAN formulation by incorporating additional side information, such as class labels, into both the generator and the discriminator.

By conditioning the generation process on a label y, the generator learns a mapping $G(z,y)$ that aims to produce samples consistent with the desired class, while the discriminator is trained to assess not only whether an image is real or fake, but also whether it matches the provided condition. This conditioning mechanism enables controlled generation and significantly increases the practical usefulness of GANs in applications where semantic attributes matter, such as digit synthesis, object class control, and, more generally, text-to-image generation.

In this project, we investigate the implementation and behavior of conditional GANs on the MNIST dataset of handwritten digits. Starting from an unconditional DCGAN baseline, we progressively introduce conditioning mechanisms that allow explicit control over the generated digit class.

Beyond basic conditioning via concatenation of labels, we explore stabilization strategies inspired by literature, such as Spectral Normalization applied to the discriminator, to mitigate training instabilities and mode collapse. The experimental analysis focuses on both qualitative and quantitative aspects, including visual inspection of generated samples, class-conditional control checks, intra-class diversity, and discriminator behavior during training.

The main objective of this work is to demonstrate that conditional adversarial training enables controllable image generation, to analyze the limitations of simple conditioning strategies, and to assess how architectural and training refinements improve stability, diversity, and label consistency.

This study provides practical insights into the challenges of training cGANs and highlights the importance of appropriate conditioning and regularization techniques for achieving reliable class-conditional generation.

2. Environment Setup & Configuration

2.1 Global Variables and Training Hyperparameters

```
# Global variables

SEED = 42

LIVE_MONITOR = False
EMIT_INTERVAL = 1

DATASET_PATH = "../../dataset/"

BATCH_SIZE = 128
LATENT_DIM = 100
NUM_CLASSES = 10

# Path to calibrated classifier checkpoint
CLASSIFIER_CHECKPOINT = "../../drafts/draft_01/classifier/mnist_cnn_calibrated_best.ckpt"

# NUM_STEPS = 15005 corresponds to ~32 epochs (60000 images / 128 batch_size ≈ 469 steps/epoch)
NUM_STEPS = 15005
SAVE_INTERVAL = 1000

# TTUR: Two Time-Scale Update Rule
# D learns faster than G, so we use different learning rates
LR_D = 4e-4  # Discriminator learning rate
LR_G = 1e-4  # Generator learning rate (4x slower)

# Adam betas optimized for GAN training
# Lower β1 (0.5 vs default 0.9) reduces momentum, stabilizes adversarial updates
ADAM_BETAS = (0.5, 0.999)

# Label smoothing for BCE/LSGAN (use 0.9 instead of 1.0 for real labels)
LABEL_SMOOTHING_REAL = 0.9

# WGAN-GP: number of critic steps per generator step
N_CRITIC = 5

MODEL_OUTPUT_PATH = "model/"
D_MODEL_NAME = "D_DRAFT_01"
G_MODEL_NAME = "G_DRAFT_01"

NUM_EVAL_SAMPLES = 10000

# Strategies to benchmark
BENCHMARK_STRATEGIES = ["bce", "lsgan", "hinge", "wgan-gp"]
```

2.2 Setup and Training Strategy

This project adopts a systematic experimental design to analyze the behavior and stability of conditional Generative Adversarial Networks (cGANs) on the MNIST dataset. Beyond implementing a single adversarial formulation, multiple loss strategies are benchmarked, including the standard binary cross-entropy (BCE) loss, Least Squares GAN (LSGAN), hinge loss, and Wasserstein GAN with Gradient Penalty (WGAN-GP).

This design choice follows the theoretical and empirical insights discussed in the lecture materials on GAN losses and training stability, which highlight that different adversarial objectives lead to substantially different optimization dynamics, convergence behavior, and robustness to mode collapse. In particular, WGAN-GP is included due to its smoother loss landscape and explicit enforcement of the Lipschitz constraint, which is known to improve training stability compared to classical GAN formulations.

Training is performed using the Two Time-Scale Update Rule (TTUR), where the discriminator is updated with a learning rate four times higher than the generator (LR_D = 4×10^{-4}, LR_G = 1×10^{-4}). This choice is directly motivated by best practices discussed in the GAN training “recipe” guidelines, which emphasize that an overly weak discriminator fails to provide informative gradients to the generator, while an overly strong discriminator can lead to vanishing gradients.

The Adam optimizer is used with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, following the DCGAN and subsequent GAN literature, as these hyperparameters are empirically known to stabilize adversarial training by reducing excessive momentum in the discriminator updates.

The batch size is set to 128, representing a compromise between training stability and computational efficiency. Larger batch sizes generally provide more stable gradient estimates for both generator and discriminator, while still fitting comfortably within GPU memory constraints. The dimensionality of the latent space is fixed to 100, following common practice in DCGAN-style architectures, which provides sufficient capacity for modeling the variability of handwritten digits without introducing unnecessary complexity.

Training is defined in terms of a fixed number of optimization steps rather than epochs. Specifically, the model is trained for 15,005 steps. This value originates from the reference Keras implementation that was converted to PyTorch and corresponds approximately to 32 epochs over the MNIST training set (MNIST contains 60,000 samples, and with a batch size of 128, one epoch corresponds to roughly 469 iterations). The step-based formulation is consistent with how GAN experiments are typically reported in the literature, as it provides finer control over the balance between generator and discriminator updates, enables predictable checkpointing and sampling intervals, and facilitates fair comparisons across different training configurations.

The additional offset of five steps ensures that the final checkpoint aligns exactly with the predefined sampling interval of 1,000 steps, allowing the evolution of generated samples to be consistently monitored at fixed milestones throughout training.

2.3 Real-Time Monitoring Dashboard

A real-time monitoring dashboard was developed and integrated into the training pipeline to support continuous inspection of adversarial dynamics during benchmarking. When enabled, a lightweight local server streams training diagnostics such as generator sample grids, discriminator and generator loss trajectories, and per-strategy progress markers. This design aligns with best-practice recommendations for GAN training, where losses alone are often insufficient to assess convergence or detect failure modes. The live monitor improves experimental transparency, facilitates early detection of mode collapse or imbalance between networks, and ensures consistent reporting across multiple loss strategies within the same benchmarking framework.

3. GAN Loss Strategies

3.1 GAN Loss Strategies and Benchmarking Framework

To systematically study the impact of different adversarial objectives on training stability and sample quality, the training pipeline was designed around a modular loss-strategy framework.

Instead of hard coding a single GAN loss, a common abstract interface was defined to encapsulate the discriminator and generator loss components, enabling multiple loss formulations to be benchmarked under identical architectural and optimization conditions.

Four widely adopted loss strategies were implemented and evaluated:

1. **Binary Cross-Entropy (BCE) Loss**
   This strategy corresponds to the original GAN formulation, where the discriminator is trained as a probabilistic classifier distinguishing real from generated samples. The generator is optimized to maximize the probability that fake samples are classified as real. To improve training stability, label smoothing is applied to real samples, reducing overconfidence in the discriminator and mitigating sharp gradients that may destabilize the adversarial dynamics. This formulation directly reflects the classical minimax game described in the original GAN literature.

2. **Least Squares GAN (LSGAN)**
   The LSGAN objective replaces the binary cross-entropy loss with a least-squares regression loss. Instead of learning to output hard binary decisions, the discriminator is encouraged to regress towards continuous target values for real and fake samples. This formulation has been shown to reduce vanishing gradients and produce smoother optimization landscapes, often resulting in more stable convergence during training. As in the BCE strategy, label smoothing is applied to the real targets to further regularize the discriminator.

3. **Hinge Loss**
   The hinge loss formulation adopts a margin-based objective, where the discriminator enforces a separation between real and fake samples using a hinge function. Rather than predicting explicit probabilities, the discriminator outputs real-valued scores, and the generator is trained to maximize these scores for generated samples. This loss is commonly used in modern high-performance GAN architectures' and aligns with the design patterns discussed in contemporary GAN literature, particularly in combination with architectural constraints such as spectral normalization.

4. **Wasserstein GAN with Gradient Penalty (WGAN-GP)**
   The WGAN-GP strategy reformulates the adversarial game in terms of the Wasserstein distance between the real and generated data distributions. Instead of a classifier, the discriminator acts as a critic that assigns scalar scores to samples. To enforce the required Lipschitz continuity constraint, a gradient penalty term is added, penalizing deviations of the gradient norm from unity along linear interpolations between real and fake samples. Additionally, the critic is updated

multiple times per generator update, following the two-scale update principle, to ensure sufficiently strong approximation of the Wasserstein distance during training.

By implementing these loss strategies within a unified interface, the project enables a fair and controlled comparison of fundamentally different adversarial objectives. All strategies share the same generator and discriminator architectures, optimization settings, and training schedule, ensuring that observed differences in convergence behavior, stability, and sample quality can be attributed primarily to the choice of loss function rather than to confounding implementation details.

This benchmarking setup reflects the theoretical and practical considerations emphasized in the course materials, where the choice of adversarial loss is a central factor in the stability and effectiveness of GAN training.

4. Data Loading

4.1 Data Loading and Preprocessing

MNIST images were converted to tensors and normalized to the range [-1, 1] using a mean of 0.5 and standard deviation of 0.5. This preprocessing step is required when the generator outputs images through a Tanh activation, ensuring that real and generated samples lie on the same numerical scale.

The training split contains 60,000 images, and batching was performed with shuffling enabled. The DataLoader used drop_last=True to maintain a constant batch size across iterations, simplifying adversarial training dynamics and logging.

5. Generator & Discriminator

5.1 Conditional Generator

The conditional generator was implemented using a PixelShuffle-based upsampling architecture. Class conditioning was introduced through a learned embedding of the digit label into the latent space, combined multiplicatively with the noise vector to modulate the generation process.

The model first projects the conditioned latent code into a 7×7×128 feature map, then performs two stages of sub-pixel upsampling (7→14→28) using convolution + PixelShuffle blocks with Batch Normalization and ReLU activations. To reduce checkerboard and dot artifacts that can emerge from upsampling, ICNR initialization was applied to the convolutional layers preceding PixelShuffle.

Finally, the generator outputs a 28×28 grayscale image through a Tanh activation, which is consistent with preprocessing that normalizes MNIST images to the range [-1, 1].

The conditional discriminator was implemented using a label-map concatenation strategy, where each class label was embedded into a 28×28 spatial map and concatenated with the input image as an additional channel. The architecture consisted of two spectral-normalized convolutional blocks that progressively downsampled the input (28→14→7), followed by a spectral-normalized Multilayer Perceptron that produced a single authenticity score. Spectral normalization was applied to stabilize adversarial training by constraining the discriminator's sensitivity, reducing oscillations and improving robustness to mode collapse.

For benchmarking across different adversarial objectives (BCE, LSGAN, hinge, and WGAN-GP), the discriminator was configured to output raw scores (logits), allowing each loss strategy to apply its appropriate formulation without relying on a fixed sigmoid output.

5.3 Weight Initialization

Model parameters were initialized following the DCGAN guidelines, which recommend sampling convolutional and fully connected weights from a Normal(0, 0.02) distribution and initializing batch normalization layers with weights drawn from Normal(1, 0.02) and zero bias. This initialization scheme has been shown to promote stable adversarial dynamics during the early stages of training and to reduce the likelihood of vanishing gradients or premature mode collapse.

Label embedding layers were initialized using the same Normal(0, 0.02) distribution to ensure comparable scale between latent and conditional representations. For layers wrapped with Spectral Normalization, initialization was applied to the underlying un-normalized weight parameters (weight_orig) rather than the normalized weights, ensuring consistent parameter scaling at initialization while preserving the Lipschitz constraint enforced during training.

This design choice aligns with best practices in stabilizing GAN training, as discussed in the DCGAN and SNGAN literature, where careful weight initialization and spectral normalization jointly contribute to improved convergence behavior and reduced training oscillations.

6. Training & Evaluation

6.1 Class Consistency

Model Instantiation, Loss Strategy Selection

Models were instantiated on the selected compute device, with the generator using its internal initialization (including ICNR initialization for PixelShuffle upsampling) and the discriminator initialized using DCGAN-style weight initialization to improve early training stability. A modular loss-strategy interface was used to select the adversarial objective (BCE, LSGAN, hinge, or WGAN-GP).

For WGAN-GP, the discriminator instance was passed into the strategy object to enable computation of the gradient penalty term, enforcing the Lipschitz constraint required by the Wasserstein formulation.

Training used the Two Time-Scale Update Rule (TTUR), configuring separate Adam optimizers for discriminator and generator with a higher discriminator learning rate (LR_D > LR_G) and GAN-optimized momentum parameters (β_1=0.5, β_2=0.999). This setup supports stable adversarial dynamics and enables fair benchmarking across different loss formulations under consistent optimization conditions.

Class-Consistency Evaluation (Label-Alignment Metric)

To complement distribution-based metrics (FID/KID) and qualitative inspection, we introduced a class-consistency evaluation to explicitly measure whether the conditional generator obeys the requested label. This metric quantifies label alignment by computing the percentage of generated images whose predicted class (from an external classifier) matches the conditioning label y.

This is particularly relevant for conditional GANs on MNIST because a model can produce visually plausible digits while still ignoring conditioning information (a known failure mode in cGANs).

Metric definition

For each class $y \in \{0, \ldots, 9\}$, we generate N samples using $G(z,y)$, then evaluate them with a pretrained MNIST classifier $C(\cdot)$.

We also report per-class consistency, which is useful to detect asymmetric failures (e.g., digits that collapse into visually similar classes such as 3/5/8 or 4/9).

Why this evaluation is important

- Directly targets the conditioning objective: Unlike FID/KID, this metric evaluates whether the generator respects the provided label.
- Detects “conditional collapse”: The generator may learn a narrow subset of digits and still appear stable, but class-consistency immediately exposes label leakage or label ignoring.
- MNIST-appropriate: A digit classifier is trained on the same domain (handwritten digits), making it a strong and interpretable control metric.

Classifier choice and calibration: We used the same calibrated MNIST classifier employed previously (in the project pipeline) to ensure consistency across experiments. The classifier architecture (CNN) was re-instantiated without Lightning dependencies, and the weights were loaded from a Lightning-style checkpoint (state_dict). A minor compatibility step was included to remove a potential 'model.' prefix in checkpoint keys, which commonly appears depending on how the model was saved.

Before using the classifier for evaluation, we verified that it achieved high accuracy on real MNIST to ensure it is a reliable oracle for class-consistency. If the checkpoint is missing, the evaluation is safely skipped (to avoid breaking the benchmark pipeline).

This evaluation was applied to each trained generator configuration (different loss strategies / regularization setups), enabling a clear comparison of how each strategy affects label compliance, not only visual fidelity.

6.2 Benchmarking

6.3 Comparative Analysis of Loss Functions

6.3.1 Experimental Setup

All adversarial objectives (BCE, LSGAN, Hinge, and WGAN-GP) were evaluated under a controlled and identical architectural setup. In particular, the Discriminator shared the same network design across all experiments, incorporating:

- Spectral Normalization (SN) applied to all convolutional and fully connected layers.
- Dropout (p = 0.25) as a regularization mechanism.
- Two Time-Scale Update Rule (TTUR), with the Discriminator learning rate four times larger than that of the Generator.
- Conditional input via spatial concatenation of the class embedding with the image.

This ensured a fair comparison across loss strategies, while also imposing a strong regularization regime on the Discriminator.

6.3.2 Sample Fidelity and Distributional Alignment (FID & KID)

The quantitative evaluation using FID and KID revealed a clear ranking among the adversarial objectives:

- LSGAN achieved the best overall performance, with FID = 6.30 and KID = 0.0034 ± 0.0017, indicating the strongest alignment between generated and real MNIST distributions.
- BCE followed closely, with FID = 7.74 and KID = 0.0048 ± 0.0020, confirming its role as a strong baseline for low-resolution image generation.
- Hinge loss exhibited a noticeable degradation, obtaining FID = 10.44 and KID = 0.0070 ± 0.0022, suggesting that margin-based objectives may require higher model capacity or different regularization to be competitive in this setting.
- WGAN-GP performed substantially worse, with FID = 68.03 and KID = 0.0691 ± 0.0078, indicating poor distributional matching under the adopted configuration.

These results suggest that, for MNIST and the current cGAN architecture, least-squares objectives provide more stable and effective gradients for training than Wasserstein-based formulations.

6.3.3 Conditional Consistency

Beyond visual fidelity, the conditional nature of the model was evaluated via a class-consistency metric, measuring the percentage of generated samples whose predicted class matched the conditioning label:

- **BCE**: 97.64%
- **LSGAN**: 97.38%
- **Hinge**: 97.18%
- **WGAN-GP**: 71.02%

The high consistency scores obtained by BCE, LSGAN, and Hinge confirm that the conditioning mechanism is correctly learned and that the Generator reliably respects the class information. In contrast, the markedly lower consistency of WGAN-GP indicates that, in this regime, the Wasserstein objective fails to preserve class-conditional structure, leading to semantically inconsistent samples.

6.3.4 Computational Cost

The computational overhead varied significantly across strategies:

- BCE / LSGAN / Hinge: approximately 280–290 seconds of training time.
- WGAN-GP: approximately 1,360 seconds, reflecting the additional cost of multiple updates from critic per generator step (n_critic = 5) and the computation of the gradient penalty.

Thus, WGAN-GP incurred a substantially higher computational cost without corresponding gains in sample quality or conditional consistency in this experimental setting.

6.3.5 On the Degradation of WGAN-GP under SN with Dropout

The poor performance of WGAN-GP should be interpreted in light of the interaction between multiple regularization mechanisms applied simultaneously to the Discriminator:

- Spectral Normalization (SN) explicitly constrains the Lipschitz constant of the Discriminator.
- Gradient Penalty (GP) enforces an additional Lipschitz constraint by penalizing deviations of the gradient norm from unity.
- Dropout introduces stochasticity into the Discriminator's function, perturbing the smooth gradient structure assumed by the Wasserstein formulation.

The combination of SN + GP + Dropout can therefore lead to over-regularization of the Discriminator, limiting its capacity to provide informative gradients to the Generator. Since WGAN-GP relies critically on well-behaved and accurate gradient estimates, this mismatch degrades both visual fidelity and conditional alignment. In contrast, BCE and LSGAN objectives are less sensitive to such constraints and remain stable under strong regularization, which explains their superior empirical performance in this setup.

The benchmark indicates that for conditional MNIST generation under a strongly regularized Discriminator, LSGAN offers the most favorable trade-off between fidelity, conditional consistency, and computational efficiency, with BCE remaining a competitive baseline. Although WGAN-GP is theoretically attractive for high-dimensional and complex image distributions, its direct application in this low-resolution, heavily regularized setting proved suboptimal.

6.4 Benchmarking Summary Table

6.4.1 Interpretation of Results

LSGAN achieved the best overall performance, obtaining the lowest FID (6.30) and KID (0.0034), indicating the closest alignment between generated and real MNIST distributions. Compared to BCE, which achieved FID = 7.74 and KID = 0.0048, the improvement is consistent with the theoretical motivation of least-squares objectives to mitigate gradient saturation and provide smoother optimization dynamics for the Generator.

BCE remained a strong baseline, producing competitive fidelity scores with only a small degradation relative to LSGAN. This confirms that the original GAN objective remains effective for low-resolution domains such as MNIST when combined with architectural stabilization techniques (Spectral Normalization, Dropout, TTUR).

Hinge loss showed a clear drop in performance, with FID = 10.44 and KID = 0.0070. While hinge-based objectives are widely used in high-capacity GANs (e.g., BigGAN, SAGAN), their benefits did not translate as effectively to this lower-resolution setting and architecture, suggesting that hinge loss may require either larger model capacity or alternative regularization schemes to be competitive.

WGAN-GP performed significantly worse, with FID = 68.03 and KID = 0.0691, while also being by far the most computationally expensive method (≈ 1,360 seconds vs. ≈ 280–290 seconds for the others). This indicates that, under the current configuration, the Wasserstein objective with gradient penalty fails to provide useful learning signals for conditional MNIST generation.

6.4.2 Computational Trade-offs

The training time highlights an important practical trade-off:

- BCE, LSGAN, and Hinge converge in comparable time (≈ 280–290 s),
- WGAN-GP is almost 5× slower, due to:
    - multiple critic updates per generator step (n_critic = 5),
    - the cost of computing gradient penalties,
    - higher memory and computational overhead.

Given that WGAN-GP also produced substantially worse sample quality, it is dominated by the simpler objectives in this experimental regime.

7. Results Analysis

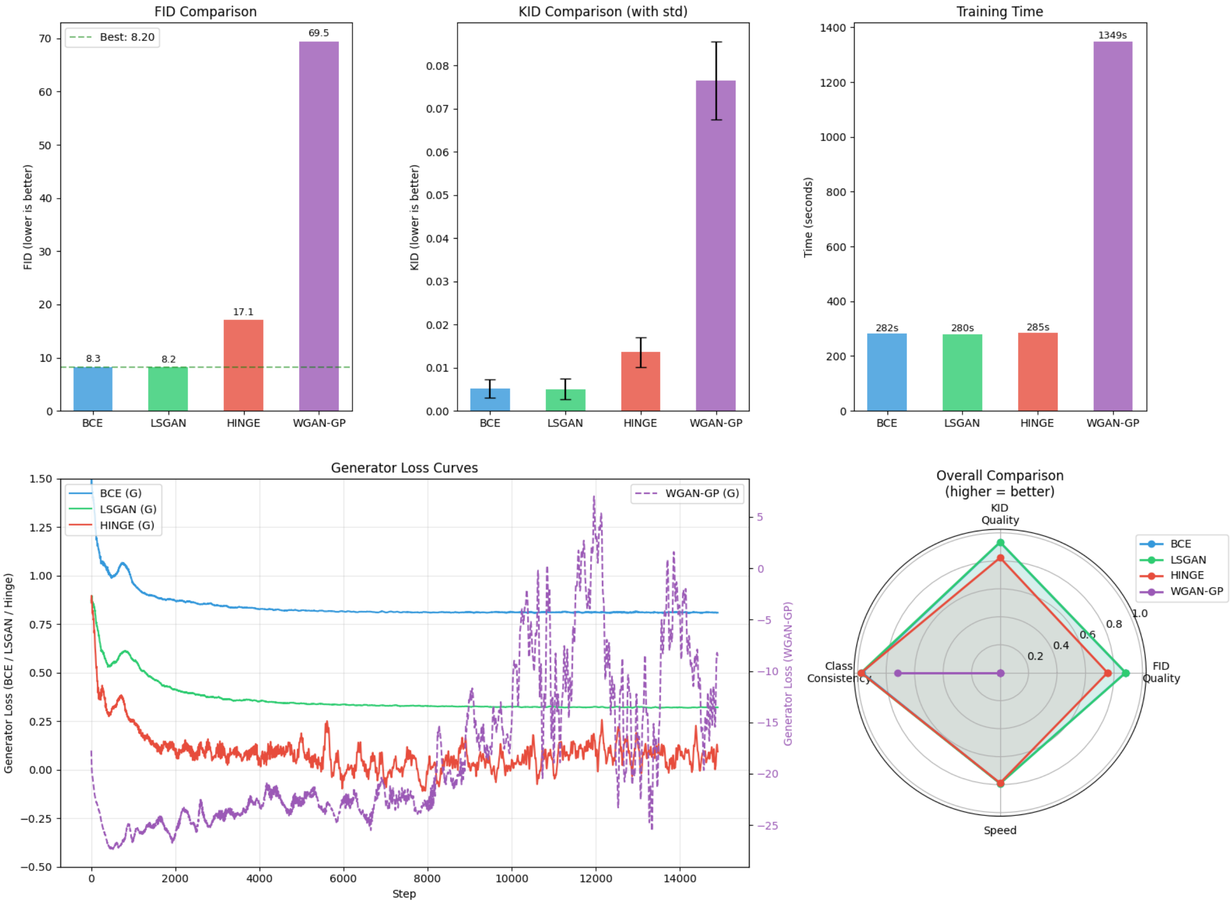


Figure 7.1 - Benchmark comparison across loss strategies. Top row: FID scores, KID scores with standard deviation, and training time. Bottom left: smoothed generator loss curves during training (WGAN-GP on secondary axis due to different scale). Bottom right: radar chart summarizing FID quality, KID quality, class-consistency, and training speed (higher values indicate better performance).

7.1 Visual Analysis of Benchmark Results

Figure 7.1 presents a consolidated comparison of the four adversarial objectives (BCE, LSGAN, Hinge, and WGAN-GP) across fidelity metrics (FID, KID), computational cost, and training dynamics.

7.1.1 FID Comparison (Top-Left)

The FID bar chart confirms the quantitative ranking observed in the benchmark table:

- LSGAN achieves the lowest FID (≈ 6.3), indicating the best alignment between the generated and real MNIST distributions.
- BCE follows closely (≈ 7.7), remaining a strong baseline but slightly inferior to LSGAN.
- Hinge loss shows a noticeable degradation (≈ 10.4), suggesting that margin-based objectives are less well suited to this architecture and dataset scale.
- WGAN-GP performs substantially worse (≈ 68.0), reflecting poor sample fidelity under the current training configuration.

The dashed horizontal line marking the best FID visually highlights the clear separation between LSGAN/BCE and the remaining objectives.

7.1.2 KID Comparison with Variance (Top-Center)

The KID plot (with standard deviation bars) mirrors the FID ranking:

- LSGAN again yields the lowest KID (≈ 0.0034 ± 0.0017), reinforcing its superior sample quality.
- BCE remains competitive (≈ 0.0048 ± 0.0020), with overlapping confidence intervals but consistently higher mean error than LSGAN.
- Hinge exhibits higher divergence (≈ 0.0070 ± 0.0022).
- WGAN-GP shows both the highest mean KID (≈ 0.069) and the largest variance, indicating unstable generation quality across subsets.

The relatively small variance for BCE and LSGAN suggests stable convergence, while the larger spread for WGAN-GP is consistent with less stable training dynamics.

7.1.3 Training Time (Top-Right)

The training time plot highlights the computational cost of each objective:

- BCE, LSGAN, and Hinge complete within a narrow range (≈ 280–290 seconds).
- WGAN-GP requires ≈ 1,360 seconds, almost 5× longer than the other methods.

This large overhead is explained by:

- multiple critic updates per generator step (n_critic = 5),
- gradient penalty computation at each discriminator update,
- higher memory and compute cost per iteration.

Crucially, this increased cost does not translate into better fidelity, making WGAN-GP dominated in this experimental setting.

7.1.4 Generator Loss Curves (Bottom-Left)

The generator loss trajectories reveal qualitative differences in optimization dynamics:

- BCE and LSGAN exhibit smooth, gradually stabilizing loss curves, indicative of steady adversarial learning and convergence.
- Hinge loss shows noisier behavior, with frequent oscillations around zero, reflecting the margin-based objective's sensitivity to discriminator strength.
- WGAN-GP displays highly volatile generator loss values, with large-magnitude oscillations over time. This behavior is consistent with the Wasserstein objective being sensitive to the balance between:
    a. critic capacity,
    b. gradient penalty strength,
    c. regularization mechanisms such as Spectral Normalization and Dropout.

The instability observed here correlates with the degraded FID/KID results for WGAN-GP.

7.1.5 Overall Comparison Radar Plot (Bottom-Right)

The radar chart summarizes four dimensions: FID quality, KID quality, class-consistency, and speed.

- LSGAN dominates across all axes, achieving strong fidelity, high class-consistency, and competitive speed.
- BCE follows closely, slightly behind in fidelity but comparable in class-conditioning accuracy and computational efficiency.
- Hinge shows weaker fidelity despite similar speed, indicating that its benefits do not materialize in this low-resolution conditional setting.
- WGAN-GP is strongly penalized on both quality and speed, and exhibits a marked drop-in class-consistency, reflecting poorer alignment between generated digits and conditioned labels.

7.1.6 Discussion and Implications

Across all visual and quantitative indicators, LSGAN consistently provides the best trade-off between stability, fidelity, and computational efficiency. BCE remains a robust baseline, confirming that classical GAN objectives remain competitive when combined with architectural stabilization techniques (Spectral Normalization, TTUR, conditional inputs).

The poor performance of WGAN-GP in this benchmark is particularly noteworthy. While theoretically appealing, WGAN-GP appears mismatched with the chosen regularization stack (Spectral Normalization + Dropout) and the low-resolution MNIST domain. The interaction between gradient penalties and spectral constraints likely over-regularizes the critic, weakening the learning signal provided to the generator. This highlights that WGAN-GP is not universally superior and requires careful adaptation of architecture and hyperparameters to outperform simpler objectives.

These results justify the selection of LSGAN as the preferred adversarial objective for conditional MNIST generation in this project and underline the importance of empirical benchmarking when choosing GAN loss formulations.

8. Visualizations (Per-Class, Fixed-Z)

8.1 Per-Class Diversity Analysis

Table X reports the per-class standard deviation of generated samples for each digit (0–9) under the four adversarial objectives. Higher values indicate greater intra-class diversity, i.e., a wider variety of writing styles, stroke thicknesses, and shapes within the same digit class.

8.1.1 Comparison Across Loss Functions

Across nearly all digits, BCE and LSGAN consistently achieve the highest intra-class diversity, with very similar magnitudes. For instance:

- **Digit 0**: BCE ≈ 0.673 vs. LSGAN ≈ 0.674
- **Digit 3**: BCE ≈ 0.626 vs. LSGAN ≈ 0.602
- **Digit 8**: BCE ≈ 0.629 vs. LSGAN ≈ 0.608

This indicates that both objectives can produce stylistically diverse samples within each class, rather than collapsing to a narrow prototype. The marginal advantage of BCE on some digits (e.g., 2, 3, 6, 8) is small and does not change the overall ranking.

Hinge loss exhibits systematically lower diversity across most classes. While the drop is moderate, it is consistent (e.g., digit 4: 0.548 vs. BCE 0.568; digit 7: 0.550 vs. BCE 0.568), suggesting a mild tendency toward more concentrated modes. This aligns with earlier observations that hinge objectives can favor sharper but less diverse outputs when model capacity or regularization is limited.

WGAN-GP displays the lowest diversity for every digit, with a substantial margin. For example:

- Digit 1: WGAN-GP ≈ 0.362 vs. BCE/LSGAN ≈ 0.473
- Digit 4: WGAN-GP ≈ 0.466 vs. BCE/LSGAN ≈ 0.57
- Digit 9: WGAN-GP ≈ 0.455 vs. BCE ≈ 0.577

This pattern indicates a pronounced mode contraction under WGAN-GP in this setup, where samples for each class cluster are generated around a limited set of visual patterns. This is consistent with the previously observed degradation in FID/KID and the unstable generator dynamics for WGAN-GP.

8.1.2 Class-Specific Observations

Some digits appear intrinsically easier to diversify across all objectives:

- Digits 0, 3, 6, and 8 consistently show higher diversity across BCE, LSGAN, and Hinge. These digits naturally admit multiple writing styles (loops, stroke openings, curvature), which the models successfully capture.
- Digit 1 exhibits the lowest diversity across all methods, reflecting the limited stylistic variability of the digit “1” in MNIST (mostly vertical strokes with minor variations). The gap between WGAN-GP and the other methods is particularly pronounced here, indicating near-prototype collapse.

8.1.3 Interpretation and Link to Global Metrics

The per-class diversity trends reinforce the conclusions drawn from FID/KID and class-consistency:

- LSGAN and BCE achieve a favorable balance between fidelity and diversity, producing not only realistic digits but also a wide range of styles within each class.
- Hinge loss trades off some diversity for sharper but more constrained generation, which may explain its weaker global FID/KID.
- WGAN-GP suffers from both low fidelity and low diversity, suggesting that, under the current architecture and regularization regime (Spectral Normalization + Dropout + Gradient Penalty), the critic becomes overly constrained, leading to a weak learning signal and partial mode collapse.

This per-class analysis provides important complementary evidence that the superiority of LSGAN (and the competitiveness of BCE) is not limited to global distributional metrics but extends to intra-class variability, which is crucial for conditional generation tasks.

Conversely, the consistently low per-class diversity of WGAN-GP explains, at a qualitative level, its poor FID/KID scores and supports the conclusion that WGAN-GP is ill-suited to the present experimental configuration.

8.2 Generated Samples

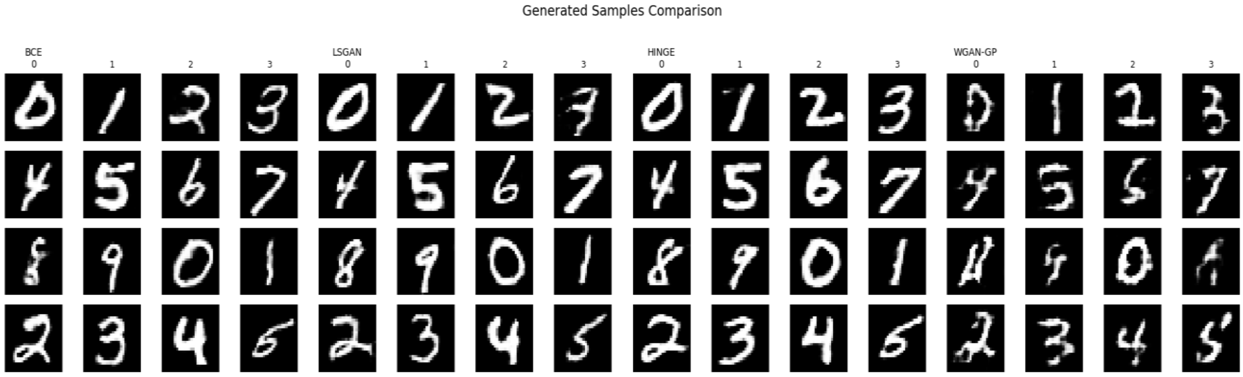
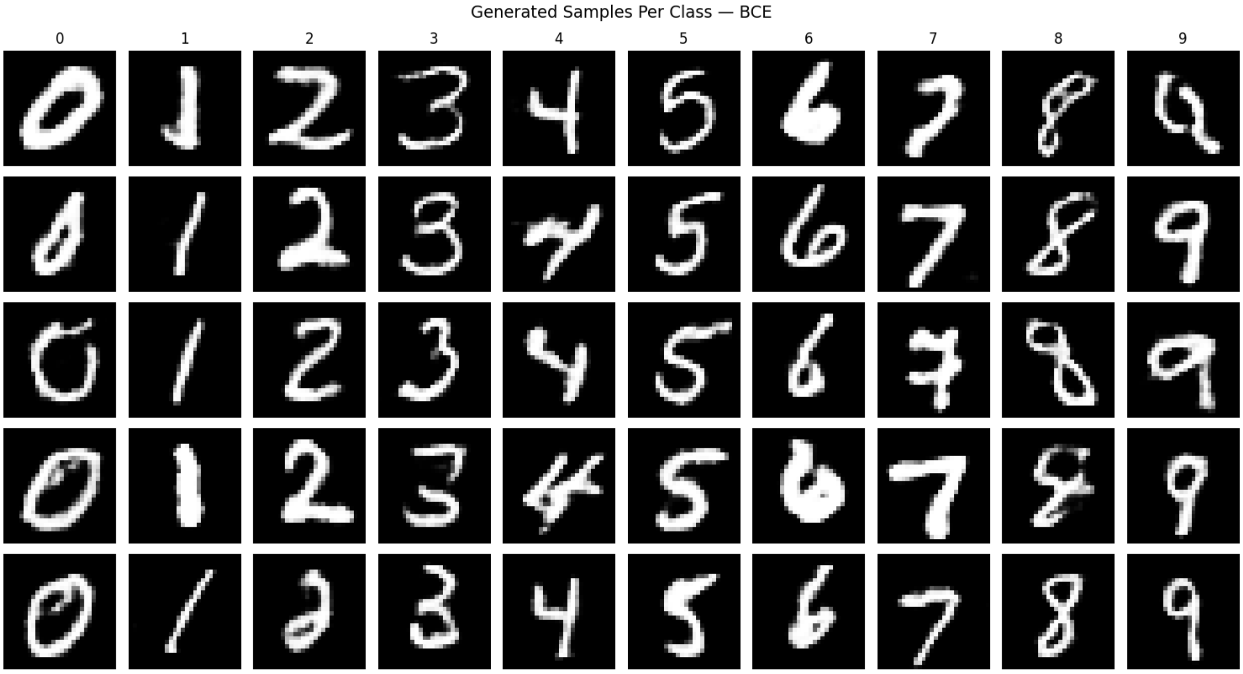
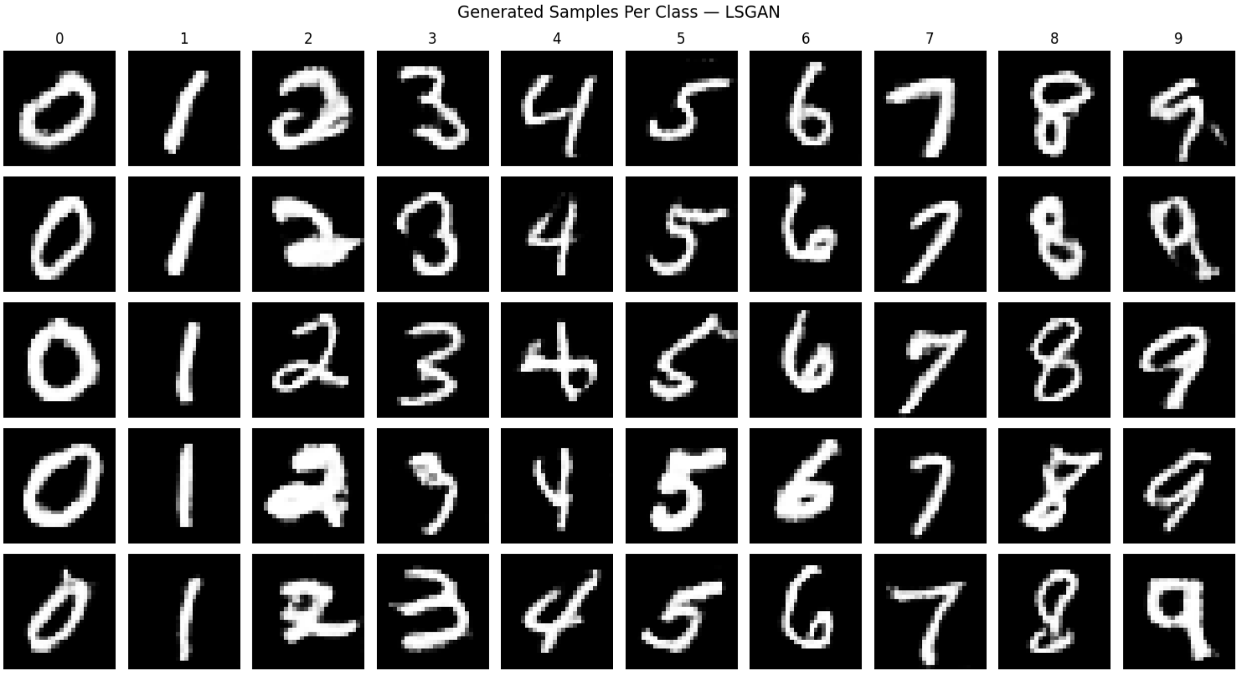


Figure 8.1 - Generated samples from each loss strategy using balanced labels (digits 0-9). Each column group shows 16 samples from a single strategy, with digit labels indicated in the header row.

The grid illustrates representative conditional samples produced by each model after training. BCE and LSGAN generate clear, well-formed digits with consistent class alignment and visible stylistic variation. Hinge loss produces recognizable digits but with slightly reduced smoothness and higher intra-class variability in stroke continuity. In contrast, WGAN-GP exhibits visibly degraded samples, with less stable digit structure and occasional artifacts, reflecting the weaker quantitative performance observed under this training configuration.

8.3 Per-class Grid





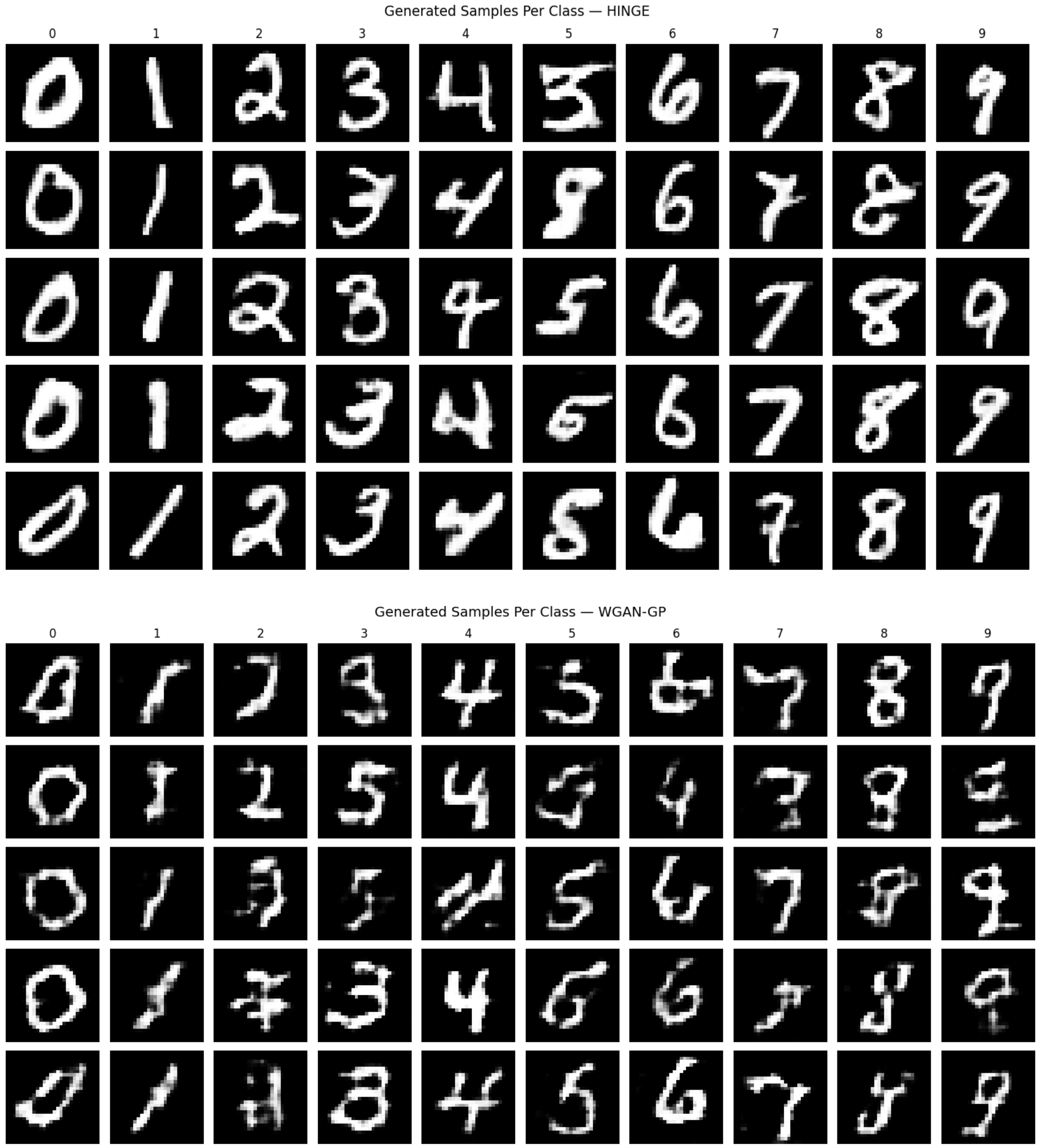
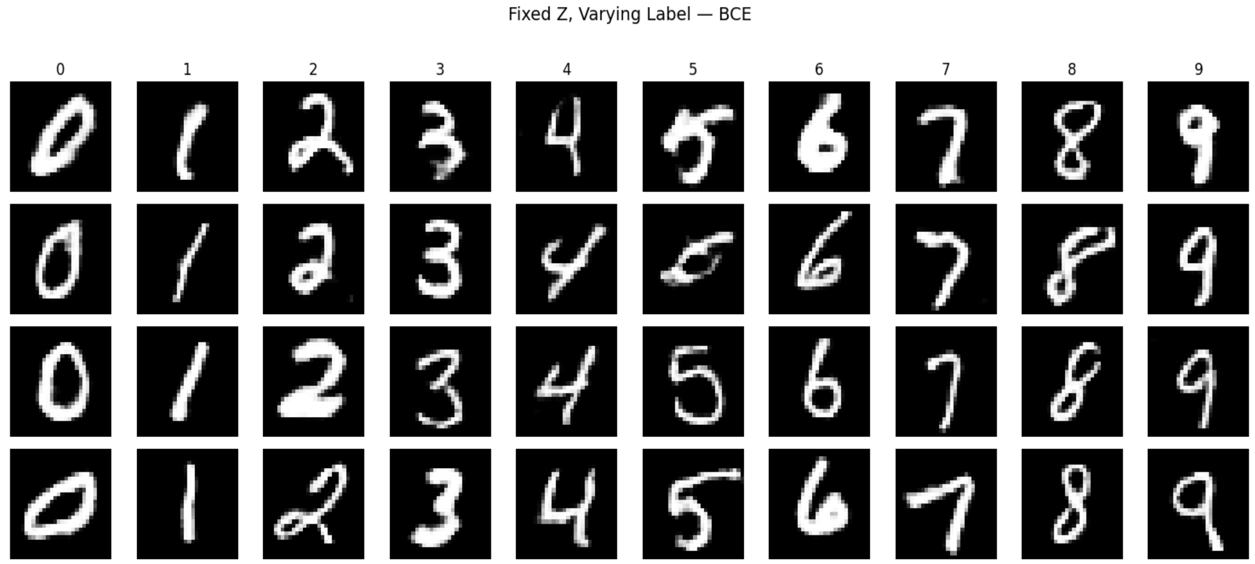

Generated Samples Per Class — HINGE
0
1
2
3
4
5
6
7
8
9
Generated Samples Per Class — WGAN-GP
0
1
2
3
4
5
6
7
8
9

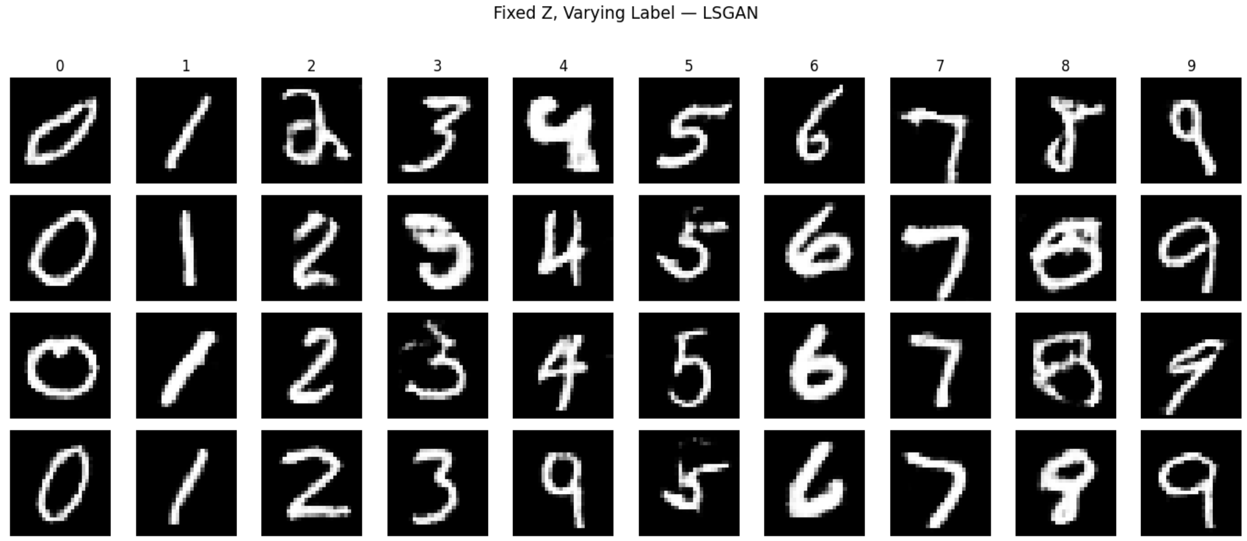
Figure 8.2 - Per-class generation grid showing five samples per digit (0-9) with different noise vectors. Each column corresponds to a conditioned class, demonstrating intra-class diversity and label consistency.

8.4 Fixed-Z

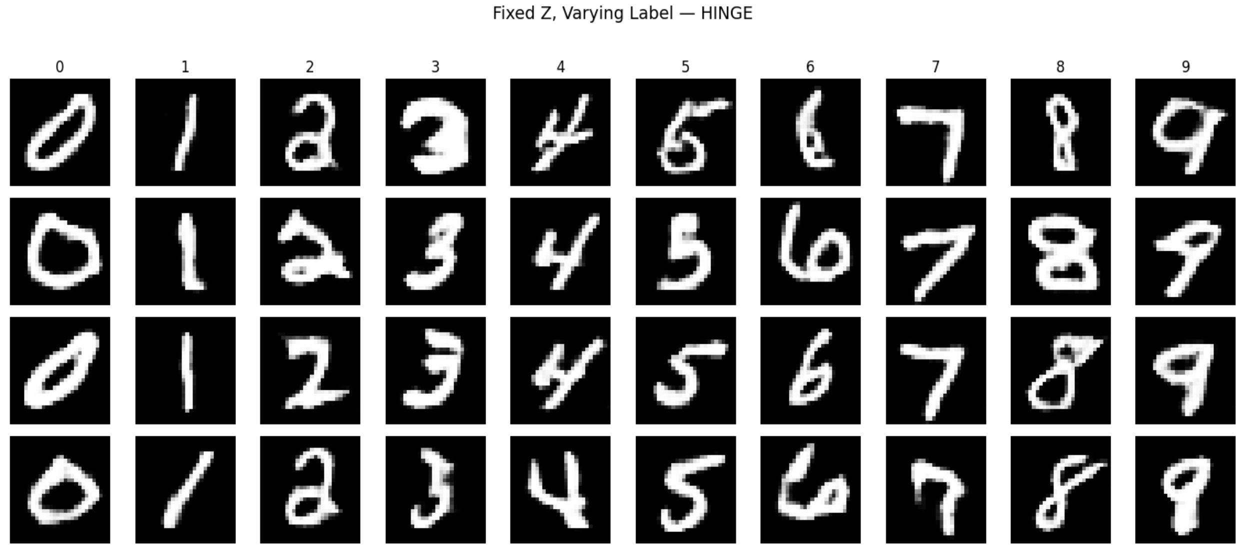
A key property of conditional GANs is the ability to disentangle the latent noise vector z from the class label y. To verify that our cGAN has learned this separation, we fix a single noise vector and vary the conditioned label across all digits (0-9). If conditioning is correctly learned, the same z should produce digits that share stylistic characteristics (such as stroke thickness, slant, or size) while differing only in their identity. This visualization provides qualitative evidence that the generator uses z to control appearance



Fixed Z, Varying Label — BCE



Fixed Z, Varying Label — LSGAN



Fixed Z, Varying Label — HINGE

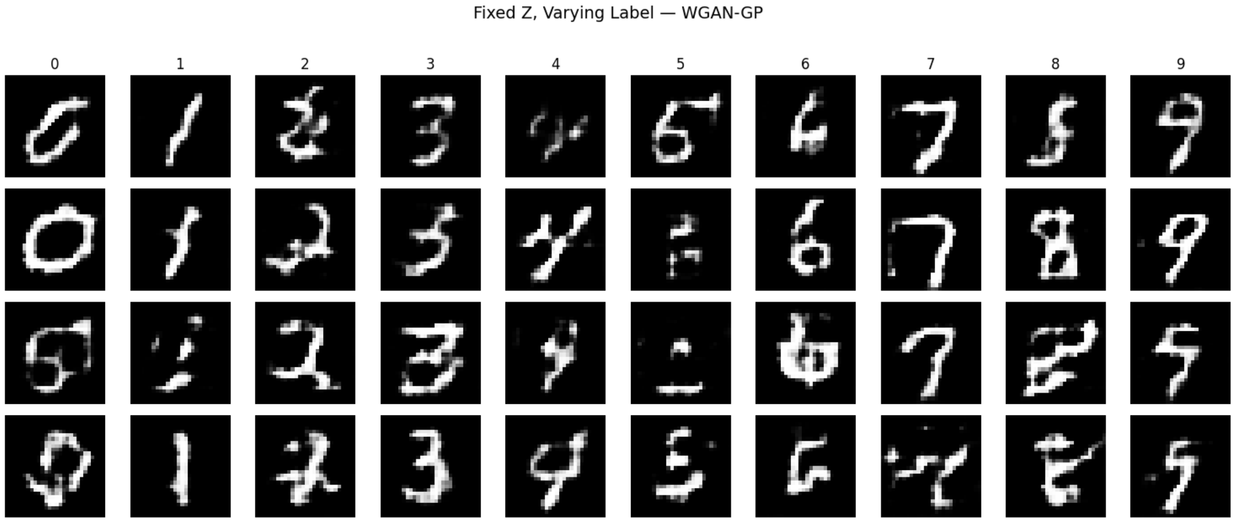


Figure 8.3 - Controllability demonstration using fixed noise vectors. Each row uses the same z while varying the label from 0 to 9, showing that the generator disentangles style (controlled by z) from digit identity (controlled by label).