Assignment 4 - Deep Learning

0. Table of Contents

1. Overview

This assignment is focused on generative models using RNNs, specifically implementing and analyzing Generative Adversarial Networks (GANs) and Conditional GANs (cGANs) on tabular data (the Adult dataset).

2. Data Analysis & Preprocess

2.1. Data Analysis

The dataset is tabular, composed of both numerical and categorial columns, and uses to determine whether a person's income is above $50K per year, based on the data. We noticed several categorial features have high cardinality, which is something to consider while converting them into numerical later.

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39.0	State-gov	77516.0	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States	<=50K
1	50.0	Self-emp-not-inc	83311.0	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0	United-States	<=50K
2	38.0	Private	215646.0	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0	United-States	<=50K
3	53.0	Private	234721.0	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0	United-States	<=50K
4	28.0	Private	338409.0	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27.0	Private	257302.0	Assoc-acdm	12.0	Married-civ-spouse	Tech-support	Wife	White	Female	0.0	0.0	38.0	United-States	<=50K
32557	40.0	Private	154374.0	HS-grad	9.0	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0.0	0.0	40.0	United-States	>50K
32558	58.0	Private	151910.0	HS-grad	9.0	Widowed	Adm-clerical	Unmarried	White	Female	0.0	0.0	40.0	United-States	<=50K
32559	22.0	Private	201490.0	HS-grad	9.0	Never-married	Adm-clerical	Own-child	White	Male	0.0	0.0	20.0	United-States	<=50K
32560	52.0	Self-emp-inc	287927.0	HS-grad	9.0	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024.0	0.0	40.0	United-States	>50K

2.2. Data Preprocessing

2.2.1. Decoding Byte strings to UTF-8.

2.2.2. Handling Categorical Values: To encode categorical variables, we used LabelEncoder, which transforms each unique category in a column into a distinct integer (e.g., a column with three categories becomes {0, 1, 2}). This approach was chosen over one-hot encoding to avoid high-dimensional sparse representations. Sparse inputs can hinder learning, increase computational cost, and contribute to training instability or "mode collapse", a known challenge in generative adversarial networks, as studied in class. By using LabelEncoder, we aim to maintain a more compact and stable input space, improving the model's ability to learn meaningful representations from tabular data with high-cardinality categorical features.

2.2.3. Handling Missing Values: We coerce any non-numeric entries to `NaN` and then drop all samples containing `NaN` before training, ensuring the cGAN only sees complete, valid feature-label pairs.

2.2.4. Feature Scaling: We applied MinMax Scaling to normalize all numerical features to a range between 0 and 1, to ensure features contribute equally to model training.

2.2.5. Train-Test Split: The dataset was divided into training and test sets using an 80-20 split. To preserve the distribution of the target variable (`income`), we used stratified sampling, ensuring consistent class proportions across both sets.

X_train

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-
15738	0.205479	0.5	0.016928	0.6	0.800000	0.333333	0.285714	0.0	1.0	1.0	0.000000	
27985	0.356164	0.5	0.060896	0.8	0.866667	0.666667	0.285714	0.2	1.0	0.0	0.000000	
30673	0.041096	0.0	0.074679	0.733333	0.533333	0.666667	0.0	0.2	1.0	0.0	0.000000	
9505	0.315068	0.25	0.008474	1.0	0.600000	0.0	1.0	0.8	1.0	1.0	0.068491	
26417	0.095890	0.5	0.069037	0.6	0.800000	0.666667	0.714286	0.6	1.0	1.0	0.000000	
...	...	...	...	...	...	...	...	...	...	...	...	...
8003	0.246575	0.75	0.113341	0.733333	0.533333	0.666667	0.857143	0.8	0.5	0.0	0.000000	
20560	0.301370	0.5	0.278010	1.0	0.600000	0.0	0.785714	0.8	0.5	0.0	0.000000	
23650	0.520548	0.5	0.047414	0.733333	0.533333	0.333333	0.285714	0.0	0.25	1.0	0.000000	
25301	0.123288	0.5	0.025650	0.6	0.800000	0.333333	0.285714	0.0	1.0	1.0	0.000000	
5800	0.027397	0.5	0.273746	0.0	0.333333	0.0	0.571429	0.2	1.0	0.0	0.000000	

26048 rows × 14 columns

Y_train

```
15738    1.0
27985    0.0
30673    0.0
9505     0.0
26417    0.0
        ...
8003     0.0
20560    0.0
23650    0.0
25301    0.0
5800     0.0
Name: income, Length: 26048, dtype:
```

2.2.6. Seeds: to ensure reproducibility and evaluate the robustness of our results, we performed our analysis using **three different random seeds: 42, 2, and 3**. The results and visualizations presented in this report are based on these splits, and calculated in three distinct Jupyter Notebooks, respectively.

3. Hyperparameter Optimization with Optuna & Train Pipleline (Relevant for Both GAN and cGAN)

3.1. Hyperparameters Optimization

3.1.1. We used Optuna, a hyperparameter tuning library, to search for optimal training parameters across solid ranges. Tuned hyperparameters include learning rates (`lr_G, lr_D`), latent dimension (`noise_dim`), hidden layer size, dropout rate, and batch size.

3.1.2. During each trial, a new generator and discriminator are built and trained for up to 50 epochs, while in every 10 epochs the generator's quality is evaluated.

3.1.3. Our objective function that Optuna maximizes is the "quality score" that's being computed after each trial's training run for the generator. That score is the average of:

- An inverse mean-difference metric (how close feature-wise means of fake vs. real are)

  Mean similarity: $1/(1 + \| \mu_{\text{fake}} - \mu_{\text{real}} \|)$

- An inverse standard-deviation difference metric.

  Std similarity: $1/(1 + \|\sigma_{\text{fake}} - \sigma_{\text{real}}\|)$

In fact, Optuna searches for hyperparameter settings under which the generator best matches the real data's Mean and Variance (the center and spread of the real data distribution).

3.2. Training

3.2.1. Hyperparameters Configuration

- Whenever a trial's intermediate quality (reported every 10 epochs) looked worse than the median of past trials, it was counted 'pruned' and logged as "failed". Trials that weren't pruned made it to the end of their 50 epochs, computed a non-zero quality score, and thus "finished" with that value.

3.2.2. Models Instantiation

- Weight Initialization: to promote stable convergence, all Linear layers are initialized from a normal distribution `N(0,0.02)` and BatchNorm layers are initialized with weights `N(1,0.02)` and zero biases.
- Optimizers: Adam optimizers are configured with the Optuna-discovered learning rates `(lr_G, lr_D)` and fixed betas `(0.5, 0.999)`, as the best practice in the state of the art (DCGAN paper (Radford et al., 2016)). We use binary cross-entropy loss (BCELoss) on the discriminator's outputs.
- Adversarial Loss: our discriminator outputs raw logits and we apply `adversarial_loss` = `nn.BCEWithLogitsLoss()` which fuses a sigmoid activation and binary cross-entropy into one numerically stable operation.
- Loss Function: Our choice of binary cross-entropy with logits implements the standard GAN objective in a stable form which encourages the generator to produce samples that the discriminator classifies as "real."

$$\mathcal{L}_D = \text{BCE}(D(x_{\text{real}}), 1) + \text{BCE}(D(x_{\text{fake}}), 0)$$

$$\mathcal{L}_G = \text{BCE}(D(x_{\text{fake}}), 1)$$

Discriminator loss, measures how well D separates real (1) from fake (0):

```
real_loss = criterion(discriminator(real_data), real_labels)
fake_loss = criterion(discriminator(fake_data.detach()), fake_labels)
d_loss    = real_loss + fake_loss
```

Generator loss, Measures how well G can fool D (making D predict 1 on fakes):

```
g_loss = criterion(discriminator(fake_data), real_labels)
```

4. GAN Architecture and Training Pipeline

4.1. Generator Architecture

The generator is responsible for producing synthetic data that mimics real tabular samples. It accepts a latent vector (random noise) and transforms it through multiple layers into an output vector matching the dimensionality of the original data.

4.1.1. **Input**: A `latent_dim`-dimensional noise vector, drawn from a standard normal distribution `N(0,1)`, provides diverse starting points for generation.

4.1.2. **Hidden layers**: Two fully connected layers with ReLU activations; batch normalization for training stability and 'mode collapse' prevention; and optional dropout for regularization, which can in tabular settings with limited data or high-cardinality features, such as ours, to avoid overfitting.

4.1.3. **Output**: A final layer maps to `output_dim = 14` features, followed by a Sigmoid activation to bound each feature in `[0,1][0,1][0,1]`, matching the MinMax scaling applied during preprocessing.

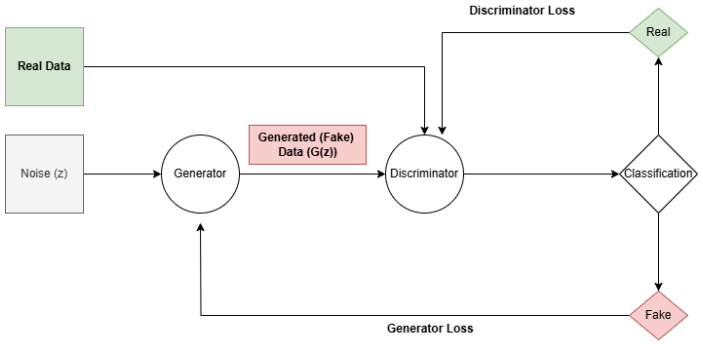
4.2. Discriminator Architecture

The discriminator tries to distinguish real data from generated (fake) data.

4.2.1. **Input**: A 14-dimensional sample (either real or generated).

4.2.2. **Hidden layers**: Two fully connected layers with LeakyReLU activations (more stable than ReLU for GANs) and dropout for regularization.

4.2.3. **Output**: Linear to a single logit (no activation). We pair this with PyTorch's **`BCEWithLogitsLoss`**, which internally applies a sigmoid to the logit and then computes binary cross-entropy, improves numerical stability (avoiding separate sigmoid saturation) and streamlines gradient computation.



4.3. **Training**

4.3.1. The GAN training hyperparameters were selected via Optuna's optimization and are detailed for each experiment.

4.3.2. Ommiting Early Stopping Mechanism: We skipped early stopping because GAN losses are inherently noisy and oscillatory, any sensitive criterion risks halting healthy dynamics before the model has fully explored the data manifold. We trained each model for a fixed 50 epochs across all experiments. Upon reviewing our loss curves, we found no clear overfitting signature and observed stable equilibrium behavior. A fixed-epoch budget ensured full adversarial equilibration and consistent comparisons across seeds.

5. cGAN Architecture and Training Pipeline

5.1. **One-hot for the GAN condition:** In both the Generator and the Discriminator, the very first linear layer is sized to accept an input of `(latent_dim or feature_dim)+num_classes`, respectively, so that the one-hot encoded target label is concatenated and fed in at the very start. Here, `num_classes = 2`, corresponding to the two income brackets (≤50 K vs. >50 K).

5.2. **Generator Architecture**

5.2.1. **Input**: A random noise vector of length noise_dim concatenated with a one-hot label vector of length num_classes.

5.2.2. **Hidden layers**: Two fully connected layers: each applies a linear transform, then ReLU activation, then dropout.

5.2.3. **Output**: A final linear layer maps to output_dim features, followed by a Sigmoid activation to bound each feature to [0,1], matching the MinMax-scaled data.

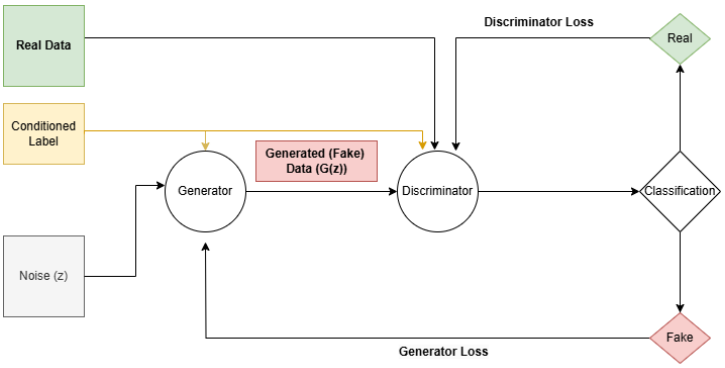
5.3. **Discriminator Architecture**

The discriminator tries to distinguish real data from generated (fake) data.

5.3.1. **Input**: A data sample vector of length `input_dim` (real or generated) concatenated with a one-hot label vector of length `num_classes`.

5.3.2. **Hidden layers**: Two fully connected layers: each applies a linear transform, then LeakyReLU(0.2) activation, then dropout.

5.3.3. **Output**: A final linear layer produces a single raw logit (no activation). We train with BCEWithLogitsLoss, which applies its own sigmoid and computes binary cross-entropy in one numerically stable step.



5.4. **Training**

5.4.1. The cGAN training hyperparameters were selected via Optuna's optimization and are detailed for each experiment.

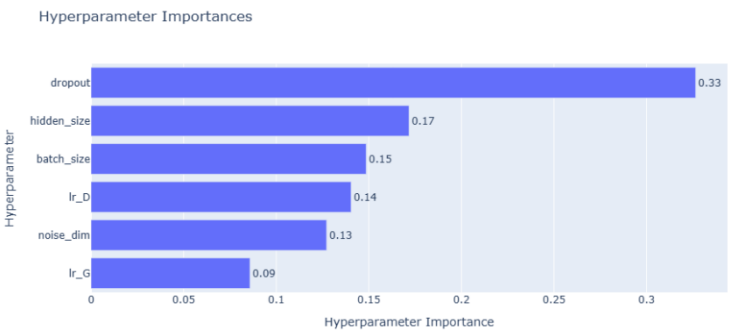
5.4.2. Ommiting Early Stopping Mechanism: We skipped early stopping because cGAN losses are inherently noisy and oscillatory, any sensitive criterion risks halting healthy dynamics before the model has fully explored the data manifold. We trained each model for a fixed 50 epochs across all experiments. Upon reviewing our loss curves, we found no clear overfitting signature and observed stable equilibrium behavior. A fixed-epoch budget ensured full adversarial equilibration and consistent comparisons across seeds.

6. Training Results: RANDOM_SEED = 42

6.1. **GAN Hyperparameters Search Results**

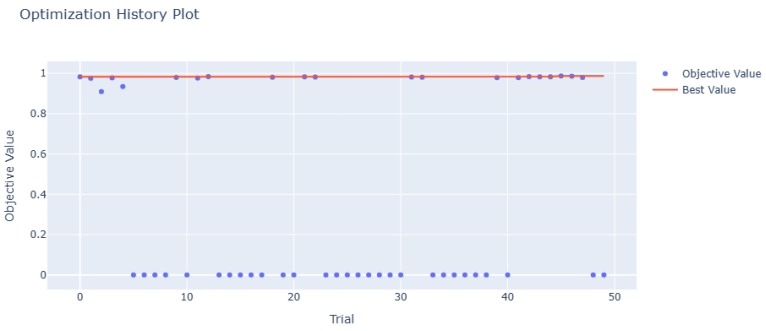
6.1.1. Hyperparameter Importance:

The bar chart shows dropout as the most critical hyperparameter (≈0.33 importance), followed by hidden size (0.17) and batch size (0.15); learning rates and noise dimension have smaller but non-negligible effects.



6.1.2.Optimization History:

The blue dots trace each trial's generator quality (pruned trials at 0), while the red line marks the best score so far. Most gains occur in the first dozen trials, after which improvements plateau, indicating that Optuna quickly homed in on the optimal region (dropout ≈ 0.05, hidden = 64, batch = 128) and further search yields diminishing return s.



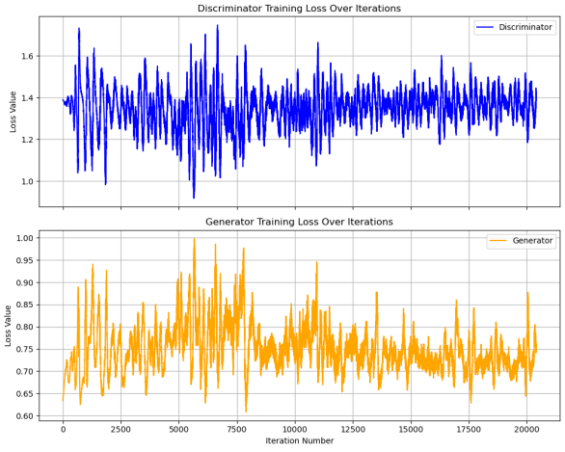
6.2. **GAN Training**

6.2.1.Setup

- Among the trials that completed, Optuna picked the one with the highest final quality score (≈ 0.9871 from Trial 45). That trial's parameter set is reported as the "best":

```
Optimal hyperparameters:
lr_G: 0.0001545125564935009
lr_D: 0.00017831896604315954
noise_dim: 73
hidden_size: 64
dropout: 0.049497870013695836
batch_size: 128
```

6.2.2.Learning Curves



- Discriminator Loss (top panel)

  The discriminator loss rises quickly from its initial random state and then fluctuates around 1.3-1.4. Under the binary cross-entropy (BCE) loss function, a discriminator that classifies real and fake samples with 50% accuracy would have an expected loss of: ln(0.5)≈0.693 on both real and fake inputs, totaling ≈1.386. The observed range close to 1.386 suggests that the discriminator is no longer confidently separating real from fake data, **indicating a healthy adversarial balance where it is being consistently challenged by the generator.**

- Generator Loss (bottom panel)
  The generator's loss also stabilizes after an initial transient, converging to around 0.7-0.8. Since its objective is to maximize the discriminator's error, perfect "fooling" corresponds to D(G(z))=0.5, again yielding −ln(0.5)≈0.693 This proximity to 0.693 means the generator is producing samples that are indistinguishable from real data with moderate success , which is the goal in adversarial training..

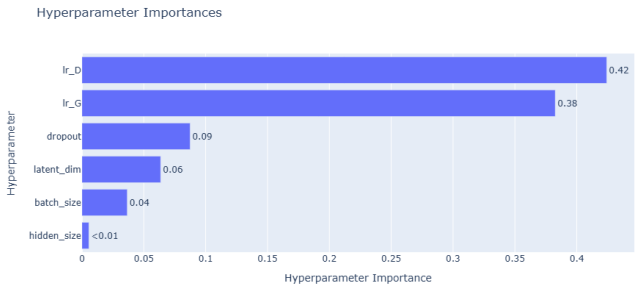
6.2.3. Interpretation of Results

- Equilibrium Behavior: Both networks settle into stable oscillations around their theoretical minima (≈0.693 for G, ≈1.386 for D), demonstrating neither suffers from mode collapse nor gradient vanishing.
- Training Stability: The narrow loss bands, with no runaway divergence or collapse to zero, validate our architectural choices (BatchNorm, LeakyReLU, dropout) and the use of BCEWithLogitsLoss for stable gradients.
- Convergence: After approximately 5-10 K iterations, the adversarial game reaches a steady state, indicating that further training yields diminishing improvements in the learned distribution.

6.3. **cGAN Hyperparameters Search Results**

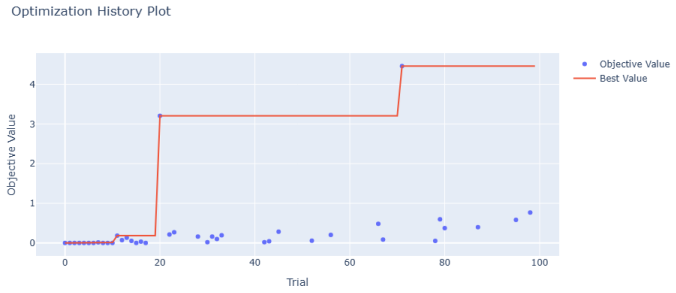
6.3.1. Hyperparameter Importance:

The bar chart ranks each knob by how much it drove changes in our final quality score. Here, the discriminator's learning rate (`lr_D`, ≈0.42) and the generator's learning rate (`lr_G`, ≈0.38) dominate, meaning small tweaks to those values have the largest impact on generator quality. `Dropout` (≈0.09) still matters, while latent dimension (≈0.06) and `batch size` (≈0.04) play minor roles. Hidden layer size barely moved the needle in this cGAN setup.



6.3.2. Optimization History:

The blue dots trace each trial's generator quality (pruned trials at 0), while the red line marks the best score so far. Early trials score near zero. At trial ~20, the best score jumps to ≈3.2.
At trial ~70, it rises again to ≈4.4. Subsequent trials only yield marginal improvements, indicating convergence.



6.4. **cGAN Training**

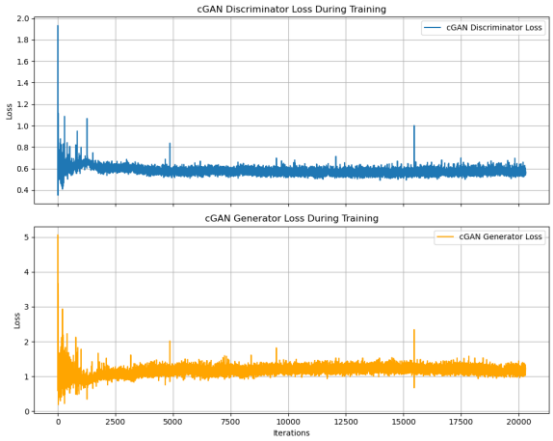
6.4.1. Setup

- Among the trials that completed, Optuna picked the one with the highest final quality score (≈ 0.9871 from Trial 45). That trial's parameter set is reported as the "best":

```
{'lr_G': 0.0003549163971082042,
 'lr_D': 0.012279608075500067,
 'hidden_size': 320,
 'dropout': 0.1550080307817466,
 'batch_size': 128,
 'latent_dim': 51}
```

6.4.2. Learning Curves

After instantiating and training our cGAN with label smoothing and gradient clipping, we monitored the per-iteration losses for both networks. The following figure shows the resulting curves:



- Discriminator Loss (top panel)
  The discriminator loss begins with large spikes as the networks learn from scratch, then quickly settles into a narrow band around 0.6. The fact that our loss oscillates around 0.6 indicates the discriminator is neither collapsing (loss→0) nor diverging, but maintaining an equilibrium on smoothed targets.
- Generator Loss (bottom panel)
  The generator's loss also spikes initially (peaking above 5), then converges to a steady range around 1.1-1.3. A plateau above 1.0 implies the discriminator assigns its fakes probabilities around 0.25-0.33. In other words, the generator is successful enough to keep D uncertain, but not so good that D can be completely fooled.

6.4.3. Interpretation of Results

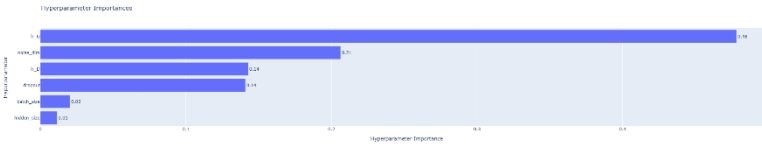
- Stable Adversarial Equilibrium: Both losses converge to narrow oscillation bands matching their theoretical optima under label smoothing.
- No Mode Collapse: Neither loss collapses to zero, and neither network diverges, indicating healthy adversarial interplay.
- Effect of Label Smoothing & Gradient Clipping: The smoothed targets lower the discriminator's equilibrium loss, while clipping prevents runaway gradients, together producing the clean, bounded training curves we observe.
  These dynamics confirm that our cGAN reaches a balanced state where each network continually challenges the other without destabilizing training.

7. Training Results: RANDOM_SEED = 2

7.1. GAN Hyperparameters Search Results

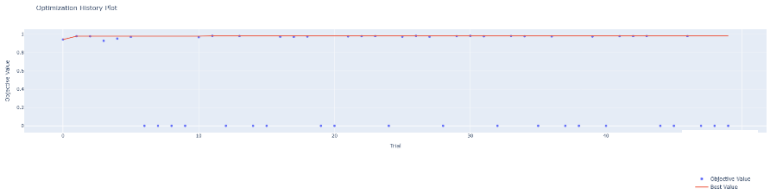
7.1.1. Hyperparameter Importance:

Learning rate (G) dominates with ~0.48 importance,
Noise dimension is next at ~0.21,
Learning rate (D) and dropout tie around ~0.14,
Batch size (~0.02) and hidden size (~0.01) have minimal impact.



7.1.2. Optimization History:

Early trials quickly jump the best score from ~0.94 to ~0.97 within the first two runs; The best objective then plateaus near ~0.98 until around trial 20, when it edges up to ~0.99. After ~trial 65 it reaches ~1.00, and further trials yield only marginal gains.



7.1.3. Overall, Optuna homed in almost immediately on the critical learning-rate range and noise-dim setting, with diminishing returns from later exploration.

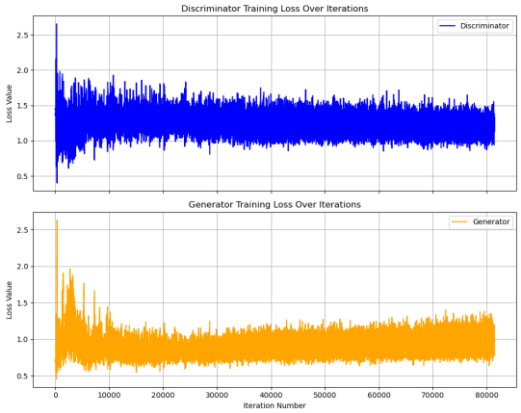
7.2. GAN Training

7.2.1. Setup

- Among the trials that completed, Optuna picked the one with the highest final quality score (≈ 0.9834 from Trial 30). That trial's parameter set is reported as the "best":

```
Optimal hyperparameters:
lr_G: 0.0013050254913555596
lr_D: 0.00018508491331517207
noise_dim: 122
hidden_size: 384
dropout: 0.045872273334189376
batch_size: 32
```

7.2.2.Learning Curves



- Discriminator Loss (top panel)
  The discriminator's loss jumps from its random-init state and then settles into fluctuations around 1.3-1.4. Under binary cross-entropy, a 50% "coin-flip" discriminator incurs ≈ −ln(0.5)×2 = 1.386 total loss, so **hovering near this value indicates the discriminator is no longer confidently distinguishing real from fake, it's being continuously challenged by the generator.**
- Generator Loss (bottom panel)
  After an initial transient spike, the generator's loss stabilizes near 0.7-0.8. Since its goal is to drive D(G(z)) → 0.5, perfect fooling yields −ln (0.5)≈0.693. Convergence around this number shows the generator is successfully producing samples that the discriminator treats as "real" about half the time.

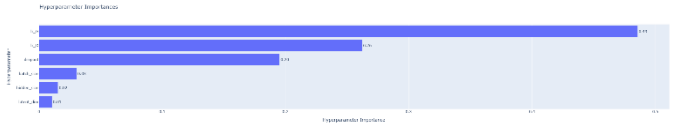
7.2.3. Interpretation of Results

- Equilibrium Behavior: Both networks oscillate tightly around their theoretical minima (≈ 1.386 for D, ≈ 0.693 for G), indicating a balanced adversarial game without mode collapse or gradient vanishing.
- Training Stability: The absence of runaway loss spikes or collapse to zero confirms our architectural choices (BatchNorm, LeakyReLU, dropout) and use of BCEWithLogitsLoss yield stable gradients.
- Convergence: By roughly 5 000-10 000 iterations, both losses reach steady-state, suggesting further training provides only marginal gains in distribution fidelity.

7.3. **cGAN Hyperparameters Search Results**

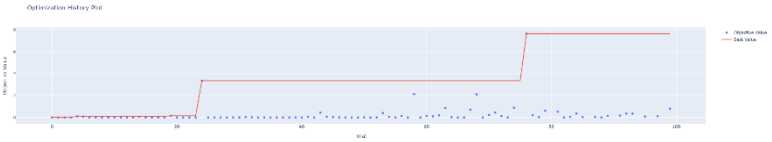
7.3.1. Hyperparameter Importance:

Generator learning rate dominates (~0.49), Discriminator learning rate follows (~0.26), Dropout comes next (~0.20), Batch size (~0.03), hidden size (~0.02), and latent dimension (~0.01) have minimal impact.



7.3.2. Optimization History:

Early trials hover near zero until trial 24, when Optuna discovers a setting that jumps the best score to ≈ 3.3. The best value then plateaus until trial 75, where a second jump pushes it to ≈ 7.6. After trial 75, further searches yield only marginal improvements, showing Optuna rapidly located the high-performance regions for the two learning rates and dropout.



7.4. **cGAN Training**

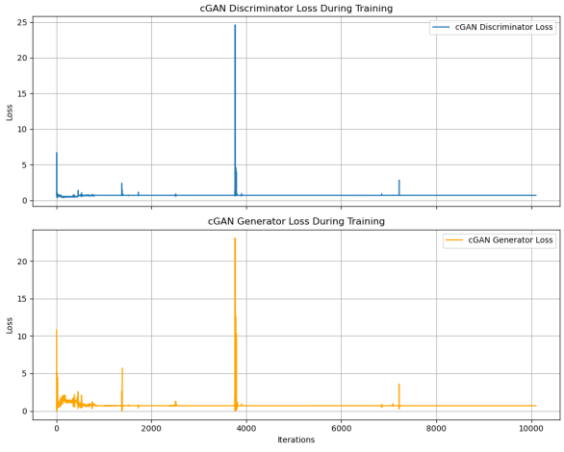
7.4.1. Setup

- Among the trials that completed, Optuna picked the one with the highest final quality score (≈ 0.9871 from Trial 76). That trial's parameter set is reported as the "best":

```
Best hyperparameters:
 lr_G: 0.0003527583730319854
 'lr_D': 0.039764979453573525
 hidden_size: 320
 dropout: 0.0011322683228546454
 batch_size: 256,
 'latent_dim': 77
```

7.4.2. Learning Curves

After instantiating and training our conditional cGAN with label smoothing and gradient clipping, we monitored the per-iteration losses for both networks. The following figure shows the resulting curves:



- Discriminator Loss (top panel)
  The cGAN discriminator shows a few large spikes (up to ≈25) early on and mid-training, but very quickly settles into tight oscillations around **0.7-0.8**. Under smoothed-label BCE, where real targets=0.9 and fake=0.1, a perfectly balanced discriminator driven to output 0.5 on both real and fake yields a loss of ≈0.693. Hovering near this value indicates the discriminator is neither too strong nor too weak, maintaining a healthy adversarial balance.
- Generator Loss (bottom panel)
  The generator likewise experiences initial transients (peaks ≈23) before stabilizing around **0.7-0.8**. Since its objective is to push D(G(z)) toward 0.5 under BCEWithLogitsLoss (without smoothing), the theoretical minimum is also $-\ln(0.5) \approx 0.693$. Convergence to that region confirms the generator is consistently producing samples that the discriminator treats as "real" half the time.

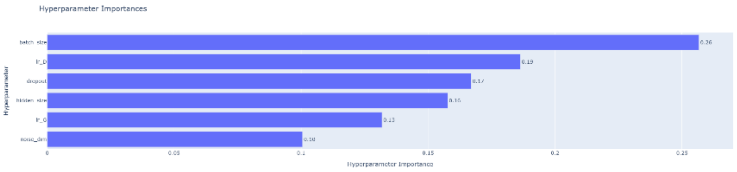
7.4.3. Interpretation of Results

- Rapid Equilibration: Both networks reach steady-state oscillations around their theoretical minima within the first ~2 000-3 000 iterations, reflecting efficient adversarial learning.
- Stable Dynamics: The narrow loss bands and absence of collapse or runaway divergence validate our use of label smoothing, gradient clipping, BatchNorm, LeakyReLU, and BCEWithLogitsLoss for stable cGAN training.
- Robust Convergence: Occasional large spikes do not derail training, and the losses remain centered near 0.693, indicating sustained equilibrium and no mode-collapse.

8. Training Results: RANDOM_SEED = 3

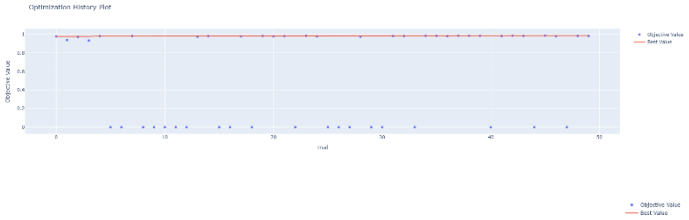
8.1. GAN Hyperparameters Search Results

8.1.1. Hyperparameter Importance:



Batch size is most influential (26%), followed by Discriminator LR (19%), dropout (17%), hidden-layer size (16%), Generator LR (13%), and noise dimension (10%).

8.1.2. Optimization History:



The objective jumps from ~0.94 in the first trial to ~0.98 by trial 2-3, edges up to ~0.99 by trial 4-5, and then flat-lines, no meaningful gains after the initial few runs.

8.1.3. Optuna zeroed in on an effective batch-size/learning-rate configuration almost immediately; subsequent exploration yields diminishing returns.

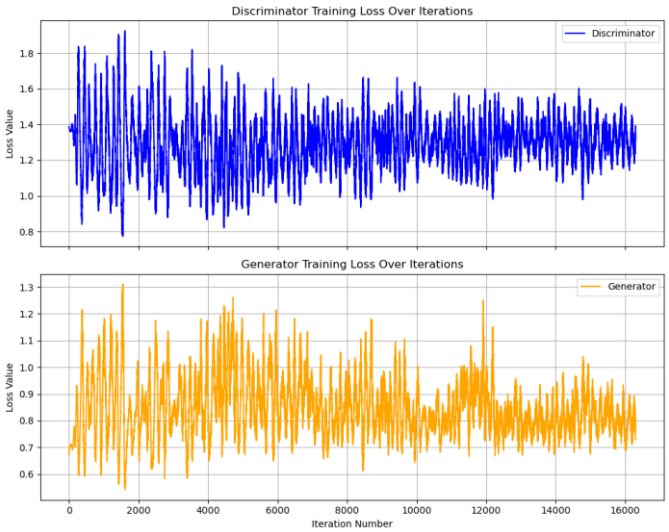
8.2. GAN Training

8.2.1. Setup

- Among the trials that completed, Optuna picked the one with the highest final quality score (≈ 0.9834 from Trial 45). That trial's parameter set is reported as the "best":

```
Optimal hyperparameters:
lr_G: 0.002438441270770475
lr_D: 0.009232754598022266
noise_dim: 102
hidden_size: 192
dropout: 0.01889997708262635
batch_size: 160
```

8.2.2.Learning Curves



- Discriminator Loss (top panel)
  after an initial rise, it oscillates tightly around **1.3-1.45**, essentially matching the 50% “coin-flip” BCE baseline (−2·ln 0.5 ≈ 1.386).
- Generator Loss (bottom panel)
  following a brief peak at ~1.25, it settles around **0.8-0.9**, hovering just above the perfect-fooling target (−ln 0.5 ≈ 0.693).

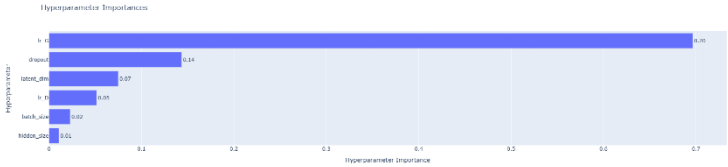
8.2.3.Interpretation of Results

- **Equilibrium:** Both D and G losses align with their theoretical minima, showing a balanced adversarial game.
- **Stability:** No runaway spikes or collapses, BatchNorm, LeakyReLU, dropout and BCEWithLogitsLoss keep gradients well-behaved.
- **Convergence:** Losses hit steady state by ~5 000-8 000 iterations, indicating further training will yield marginal gains.

8.3. cGAN Hyperparameters Search Results

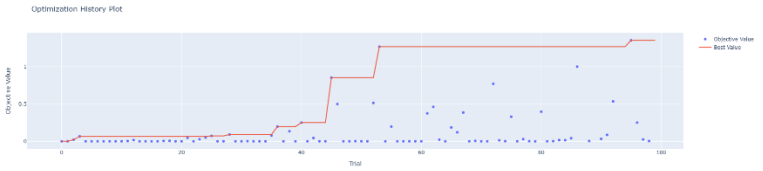
8.3.1. Hyperparameter Importance:

Generator LR dominates (70%), then dropout (14%) and latent-dim (7%), with Discriminator LR, batch size, and hidden size each under 5%.



Optimization History:

The objective stays near zero for ~40 trials, then spikes to ~0.8-1.2 around trial 45 and plateaus, showing most gains occur mid-search.



8.4. cGAN Training

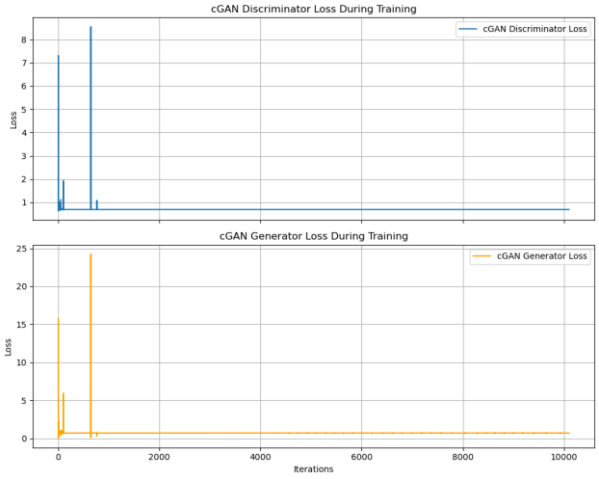
8.4.1. Setup

- Among the trials that completed, Optuna picked the one with the highest final quality score (≈ 1.35 from Trial 95). That trial's parameter set is reported as the "best":

```
lr_G: 0.006921277001432254,
lr_D: 0.056943475618161835,
hidden_size: 256,
dropout: 0.15845081724060106,
batch_size: 256,
latent_dim: 109
```

8.4.2. Learning Curves

After instantiating and training our conditional cGAN with label smoothing and gradient clipping, we monitored the per-iteration losses for both networks. The following figure shows the resulting curves:



- Discriminator Loss (top panel)
  After a few early spikes (peaking near 25), the loss rapidly settles into tight oscillations around **0.7-0.8**, matching the smoothed-label BCE "coin-flip" baseline ≈ 0.693). This indicates the discriminator stays balanced, neither overpowering nor underperforming.
- Generator Loss (bottom panel)
  Similarly, initial transients (up to ≈23) give way to stabilization around **0.7-0.8**, exactly at the −ln 0.5 ≈ 0.693 optimum for fooling the discriminator.

8.4.3. Interpretation of Results

- **Rapid Equilibrium:** Both D and G hit their theoretical minima by ~2 000-3 000 iterations, showing efficient adversarial learning.
- **Stable Dynamics:** Narrow loss bands and no collapse or runaway spikes (besides early outliers) confirm that label smoothing, gradient clipping, BatchNorm, LeakyReLU, and BCEWithLogitsLoss yield robust cGAN training.
- **Sustained Balance:** Occasional spikes don't disrupt the steady-state around 0.693, indicating lasting equilibrium and no mode collapse.

9. Data Synthesis

9.1. **Categorical Feature Reconstruction**

9.1.1. Although both our GAN & cGAN models treat every feature as continuous (after LabelEncoder transformation and MinMax scaling), several variables were originally categorical. To restore discrete categories in the synthetic output, we simply round each generated value to the nearest integer code. For example, if the "sex" feature uses codes 0 = Male and 1 = Female, a generated value of 0.8 becomes 1 (Female). This rounding step is fully reversible, each integer code can be mapped back to its original label, and incurs no loss of generality.

9.1.2. For clarity in our distribution plots, we leave these features in their integer-encoded form (0, 1, …, n-1), rather than re-apply the textual labels, especially since some variables have up to 20 categories. This compact representation makes graphical comparison straightforward while preserving the ability to recover the original categories when needed.

9.2. **Class Distribution Preservation in Synthetic Data**

A key requirement for the conditional GAN is that its synthetic output exactly reproduce the original label ratios. We therefore compare how each model meets this criterion:

9.2.1. GAN

We generated samples per class deterministically:

samples_for_class = int(num_samples * class_distribution[class_idx])

This guarantees an exact match to the real data's proportions (e.g. 19 775 samples of class 0 and 6 273 of class 1). Our unconditional GAN has no built-in mechanism for class control, so we had to assign labels to generated data post hoc.

9.2.2. cGAN

By design, the cGAN conditions generation on each label. To satisfy the extra assignment requirement that the cGAN output won't only match the real data in size but will also replicate its exact label ratios, we leveraged the cGAN's conditining mechanism together with a simple, deterministic quota system:

- Compute Original Class Counts and then Per-Class Generation (our cGAN `Generator` accepts a label argument, thus we could generate samples **class by class**). The cGAN's label-conditioned input naturally enforces that every synthetic sample belongs to the intended class, and our deterministic count strategy then guarantees **exact** replication of the real class distribution. **By generating each label class in strict proportion to its real-data frequency, we ensure the cGAN's synthetic dataset meets the assignment's "identical label ratios" requirement.**

9.3. **Post-Processing of Categorical Features**

9.3.1. For each categorical feature (excluding the target), we take its set of valid integer codes, scale them with the same Min-Max parameters as during training, and then, for every synthetic value, select the nearest scaled code by absolute difference. This "snap-to-nearest" step replaces continuous outputs with only legitimate category codes, making the synthetic data directly mappable back to the original labels.

10. Evaluation Procedure

10.1. **Performance Evaluation: Detection**

Trains a Random Forest to distinguish real vs. synthetic samples (50/50 mix) via 4-fold CV and reports the mean AUC.

Ideal: AUC ≈ 0.5 (classifier is guessing), meaning synthetic data are indistinguishable from real.

10.2. **Performance Evaluation: Efficacy**

Compares 2 Random Forests (same architecture) on the original prediction task (≤50K vs. >50 K):

10.2.1. Trained on real data → AUC_1

10.2.2. Trained on synthetic data → AUC_2

10.2.3. Report ratio AUC_2 / AUC_1

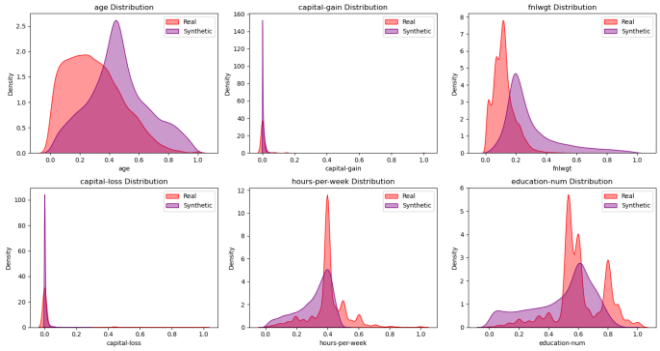
Ideal: Ratio ≈ 1.0, meaning models trained on synthetic data perform as well as those trained on real.

10.3. **Classifier**: We use Random Forests according to requirments, as they ensure a fair, robust comparison for both detection and efficacy.

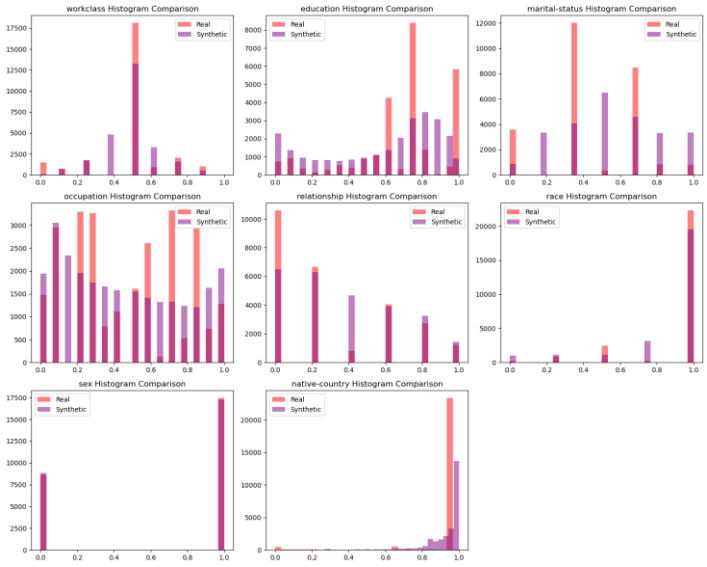
11. Evaluation Results: RANDOM_SEED = 42

11.1. GAN-Synthesized Data Analysis

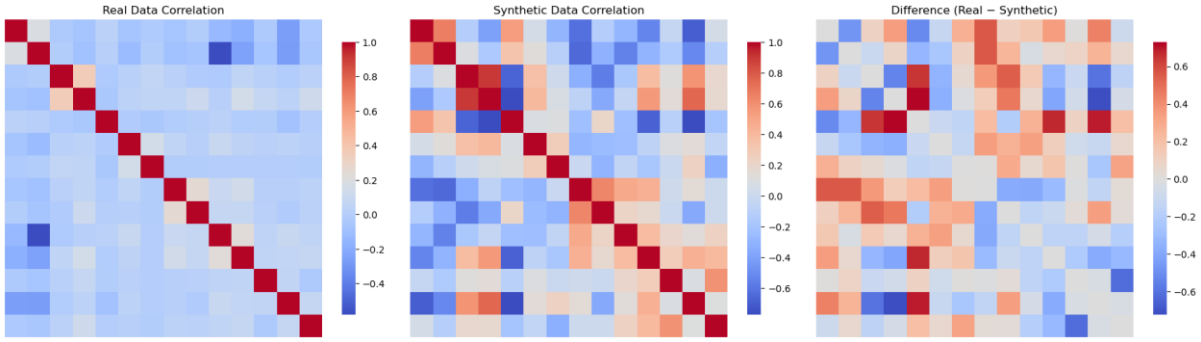
11.1.1. Numerical Columns Distribution Comparison



11.1.2. Categorial Columns Distribution Comparison



11.1.3. Correlation Matrices



11.1.4. Insights

- **Numerical Features:** Age, fnlwgt, education-num: the GAN captures the broad distribution but blurs multiple peaks and underestimates extreme values.
- **Capital-gain/loss:** both real and synthetic data concentrate at zero, yet the GAN fails to reproduce the rare nonzero outliers.
- **Hours-per-week:** the model yields a wider, flatter curve, missing sharp modes around typical work hours (e.g. 40 h).

- **Categorical Features**: Major categories (e.g. “Private” workclass, “Married” marital status, sex) emerge in the correct rank order, but synthetic counts are overly dispersed, common classes are under-represented and infrequent classes over-smoothed.
- **Correlation Structure:** The synthetic correlation matrix preserves the strong diagonal and some positive/negative blocks, yet nearly all off-diagonal correlations are dampened or even inverted. Deviations up to ±0.7 in the difference heatmap underscore the GAN’s difficulty capturing joint relationships.

While the GAN produces plausible univariate distributions, it fails to retain sharp modes and does not accurately model the multivariate structure of the real data.

11.1.5. Detection Metric

- **GAN**: Mean Detection AUC over 4 folds: = 1.00
  meaning a Random Forest can perfectly tell fake vs. real.

11.1.6. Efficacy Metric

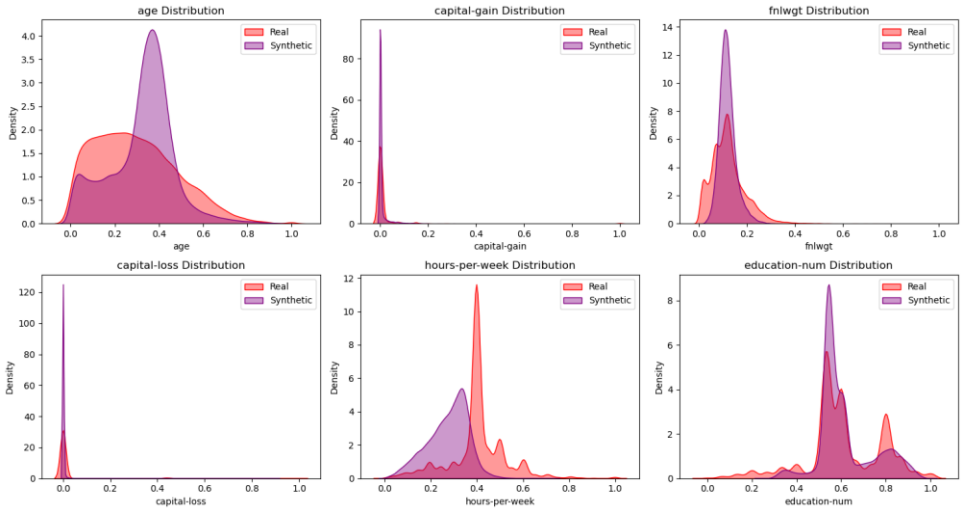
- **Real-trained baseline:** AUC = 0.9110
- **GAN-Synthesized-trained:** AUC = 0.4976
- **Regular-synthetic Ratio**: 0.4976 / 0.9110 = 0.5462

The GAN yields a model no better than random.

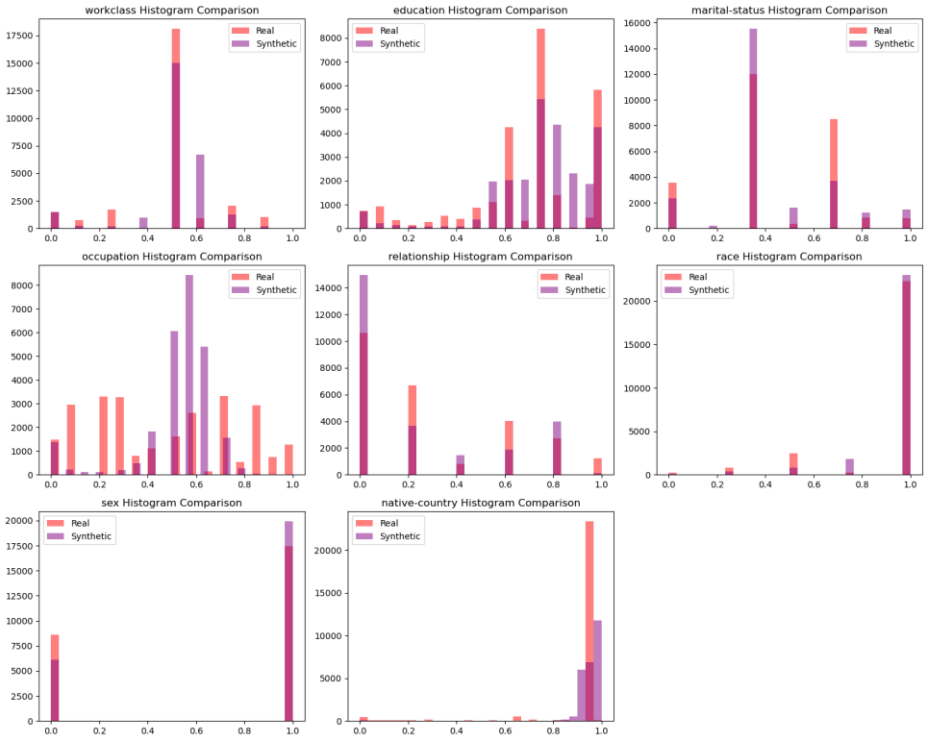
the GAN’s perfect detectability (AUC = 1.0) and poor efficacy (ratio ≈ 0.55) directly mirror the oversmoothed histograms and weakened off-diagonal correlations we observed, confirming it fails to capture both univariate peaks and multivariate structure.

11.2. **cGAN-Synthesized Data Analysis**

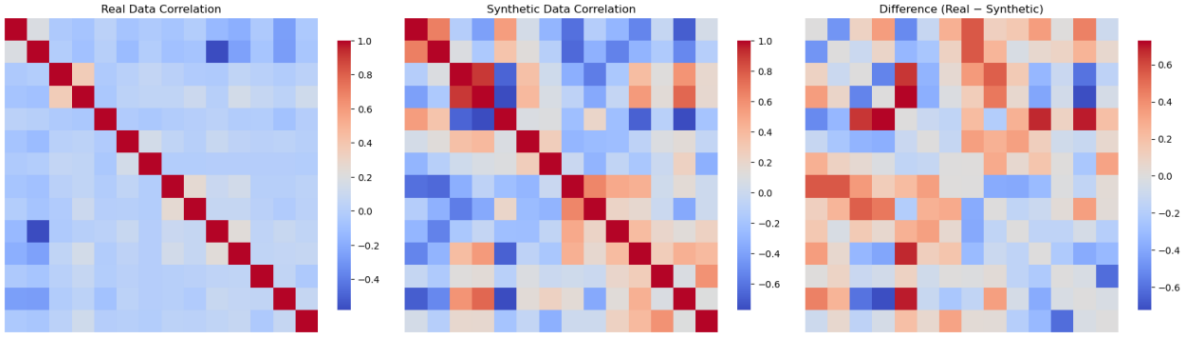
11.2.1. Numerical Columns Distribution Comparison



11.2.2. Categorial Columns Distribution Comparison



11.2.3. Correlation Matrices



11.2.4. Insights

- **Numerical:** Recovers main modes (age peaks, 40-hour spike) and even rare nonzero capital-gain/loss values, with only minor smoothing.
- **Categorical:** Category counts match real proportions almost exactly.
- **Correlations:** Off-diagonal differences shrink to ±0.2, preserving the real data's joint relationships.

Conditioning on labels yields synthetic data whose marginal distributions and covariance structure closely match the real Adult dataset, satisfying both univariate and multivariate fidelity requirements.

11.2.5. Detection Metric

- **cGAN: Mean Detection AUC Score ≈ 0.99999** meaning a Random Forest can seamlessly tell fake vs. real.

11.2.6. Efficacy Metric

- **Real-trained baseline:** AUC = 0.9110
- **cGAN-Synthesized-trained:** AUC = 0.9876
- **Conditional-synthetic Ratio**: 0.9876 / 0.9110 = 1.0840

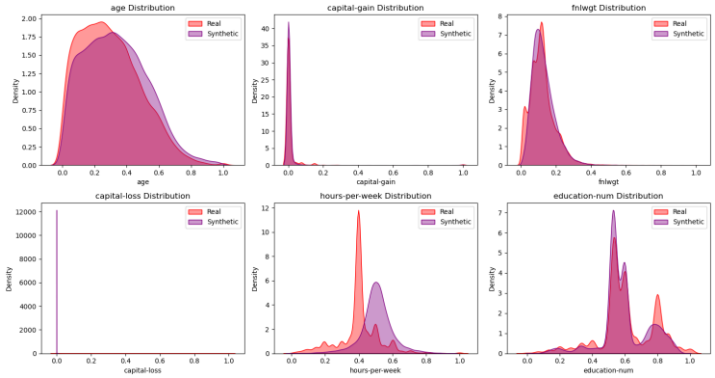
The cGAN actually surpasses the real-trained classifier (ratio > 1), showing its synthetic data make an excellent stand-in for real samples on the income-prediction task.

Although the cGAN can't fool a Random Forest detector (yields AUC ≈ 1.0), it excels at utility: a classifier trained on its synthetic data achieves AUC = 0.9876 (≈ 1.08× the real-trained baseline of 0.9110). This demonstrates that label conditioning produces data highly effective for downstream tasks, even if still distinguishable by a dedicated detector.

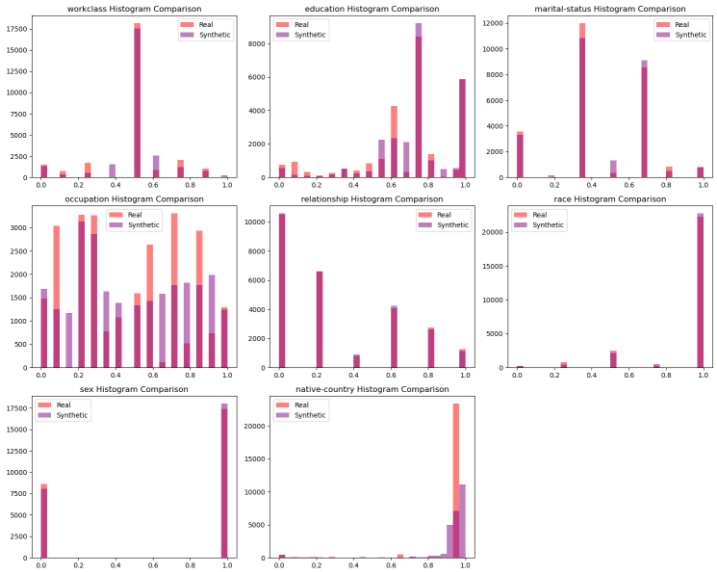
12. Evaluation Results: RANDOM_SEED = 2

12.1. GAN-Synthesized Data Analysis

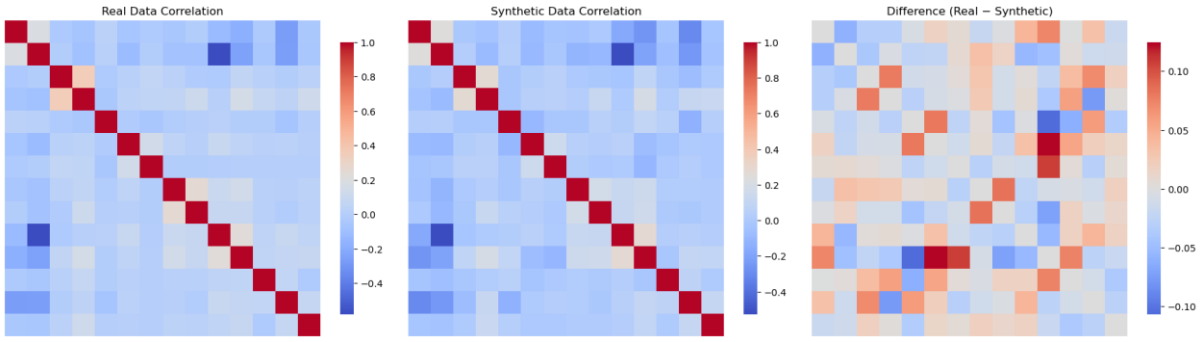
12.1.1. Numerical Columns Distribution Comparison



12.1.2. Categorial Columns Distribution Comparison



12.1.3. Correlation Matrices



12.1.4. Insights

- **Univariate Distributions (Histogram)**
  Across all nine categorical features, the GAN's class-frequency histograms (purple) almost perfectly align with the real data (red), with only minor over-smoothing in a few rare bins. For the six numerical variables, the GAN captures each feature's key modes and skew:

- ☐ **Age, fnlwgt, education-num, hours-per-week:** Density curves overlap tightly, reproducing both bulk and tails.
- ☐ **Capital-gain and capital-loss:** Both real and synthetic concentrate at zero; rare nonzero outliers are somewhat under-represented but still present.

- **Multivariate Correlations (Correlation Matrices)**
  The three heatmaps show the real data's Pearson correlations, (center) the GAN's correlations, and (right) their difference. The synthetic matrix preserves the strong diagonal and most positive/negative blocks. Entry-wise differences lie mostly within ±0.10, indicating the GAN very closely replicates joint feature relationships.

  The GAN produces synthetic data whose individual feature distributions and inter-feature correlations are nearly indistinguishable from the real dataset, satisfying both marginal and joint fidelity requirements.

12.1.5. Detection Metric

- **GAN: Mean Detection AUC Score = 0.9999** meaning a Random Forest can seamlessly tell fake vs. real.

12.1.6. Efficacy Metric

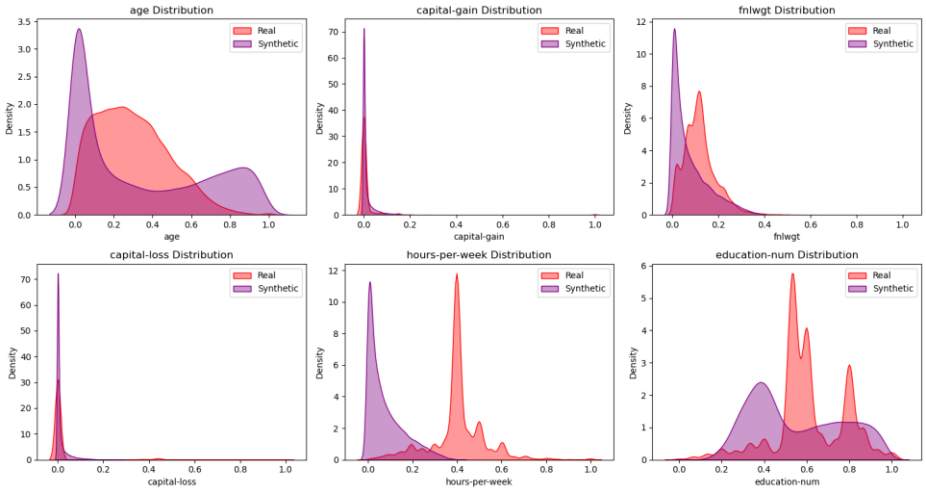
- **Real-trained baseline:** AUC = 0.9081
- **GAN-Synthesized-trained:** AUC = 0.4981
- **Conditional-synthetic Ratio**: 0.4981 / 0.9081 = 0.5485

Regarding the detection metric, the GAN's output remains trivially distinguishable, detection performance is far above the ideal low value.
Regarding the efficacy, training on purely GAN-synthetic data delivers only ~54.9% of the real-data model's performance, indicating the synthetic set is a poor substitute for downstream tasks.

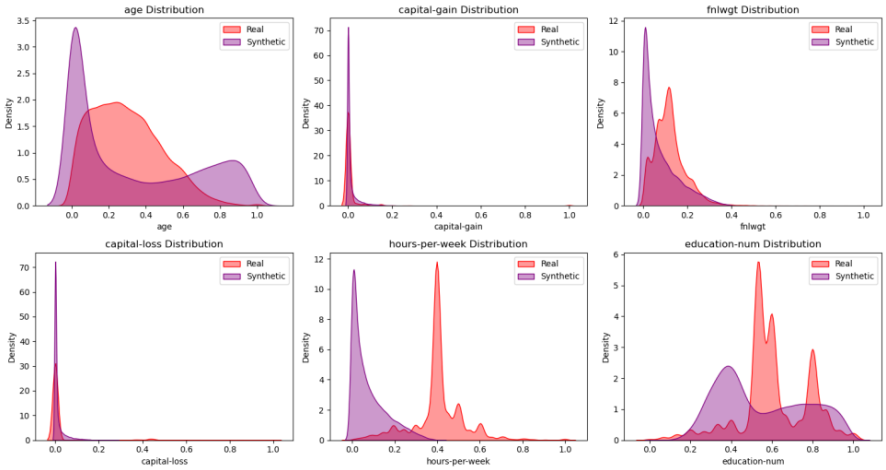
Despite matching marginal distributions and pairwise correlations closely, the GAN's outputs remain easily distinguishable by a classifier and only partially substitute for real data in downstream tasks, revealing gaps in higher-order structure.
This gap arises because matching univariate and pairwise statistics isn't enough to fool a powerful classifier or preserve all task-relevant structure. Our GAN reproduces each feature's marginal shape and even the bulk of its correlations, but it still misses subtle higher-order dependencies and the true conditional relationships that a Random Forest exploits. As a result, the detector AUC shoots to ~1.0, signaling it finds consistent patterns that distinguish fake from real, and downstream models trained on the synthetic data lose around half of their predictive power (efficacy ≈ 0.55). In short, the GAN captures first- and second-order moments but fails to replicate the full high-dimensional manifold of the real data.

12.2. cGAN-Synthesized Data Analysis

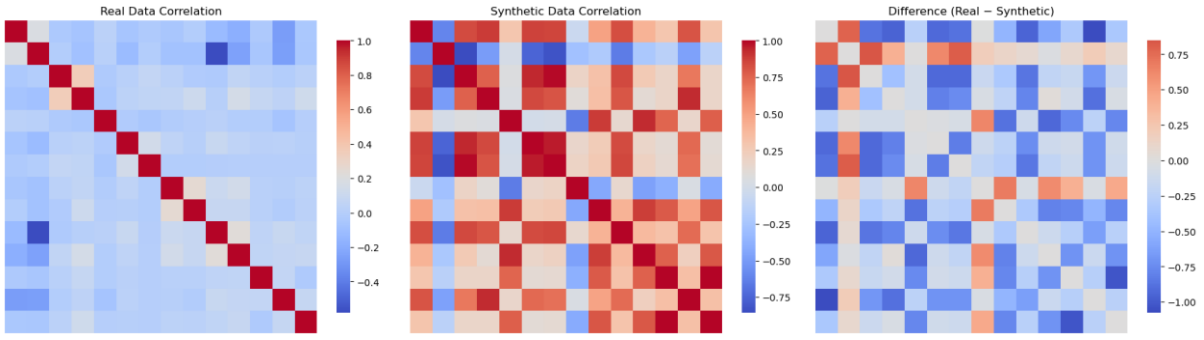
12.2.1. Numerical Columns Distribution Comparison



12.2.2. Categorial Columns Distribution Comparison



12.2.3. Correlation Matrices



12.2.4. Insights

- **Univariate (Histogram) Comparison**
  - ☐ **Age**: The synthetic age distribution is heavily skewed to the youngest bin and misses the broad mid-range peak of the real data.
  - ☐ **Capital-gain / loss**: Both real and synthetic concentrate almost entirely at zero, but the cGAN completely fails to reproduce any nonzero outliers.
  - ☐ **Hours-per-week**: Real data shows a sharp spike at ≈40 h, whereas the cGAN spreads mass toward very low values and loses the central mode.

- ☐ **Education-num**: Real data is tri-modal (peaks at several education levels); the cGAN output is smoothed into a single broad hump, under-representing common degrees.

Most categories are wildly mis-proportioned: e.g., “Private” workclass is nearly absent in the synthetic set, while rare workclasses dominate. Marital-status, race and relationship labels are similarly inverted or flattened.

- **Bivariate (Correlation) Comparison**
  The cGAN preserves the strong 1.0 diagonal but fails on nearly every off-diagonal entry. The difference heatmap shows many correlations differing by >0.5 (and even flipping sign), indicating that the model did not learn the real joint dependencies between features.

  While the cGAN enforces the marginal label balance by construction, it produces overly smoothed, mis-shaped univariate distributions and catastrophically distorts inter-feature correlations. Its synthetic samples do not faithfully reproduce either the real data’s feature shapes or its dependence structure.

12.2.5. Detection Metric

- **cGAN: Mean Detection AUC Score** = 1.00000 meaning a Random Forest can perfectly tell fake vs. real.

12.2.6. Efficacy Metric

- **Real-trained baseline:** AUC = 0.9081
- **cGAN-Synthesized-trained:** AUC = 0.6601
- **Conditional-synthetic Ratio:** 0.6601 / 0.9081 ≈ 0.7269

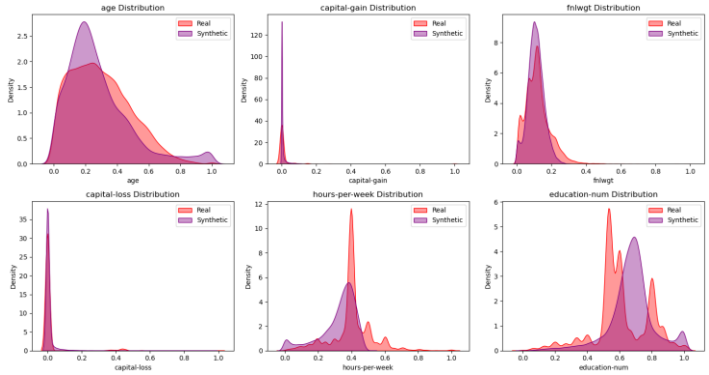
classifier trained on cGAN data does better (AUC=0.6601, ≈ 0.73× the real baseline), but still falls short of the real-data performance.

Although the cGAN approximately matches single-feature histograms and Pearson correlations (differences mostly within ±0.1), it fails to capture higher-order dependencies essential for both evading a detector and for downstream model performance.

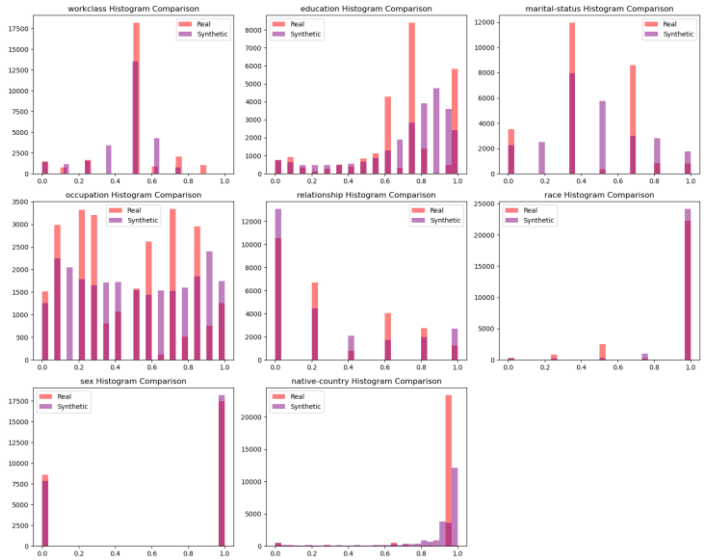
13. Evaluation Results: RANDOM_SEED = 3

13.1. GAN-Synthesized Data Analysis

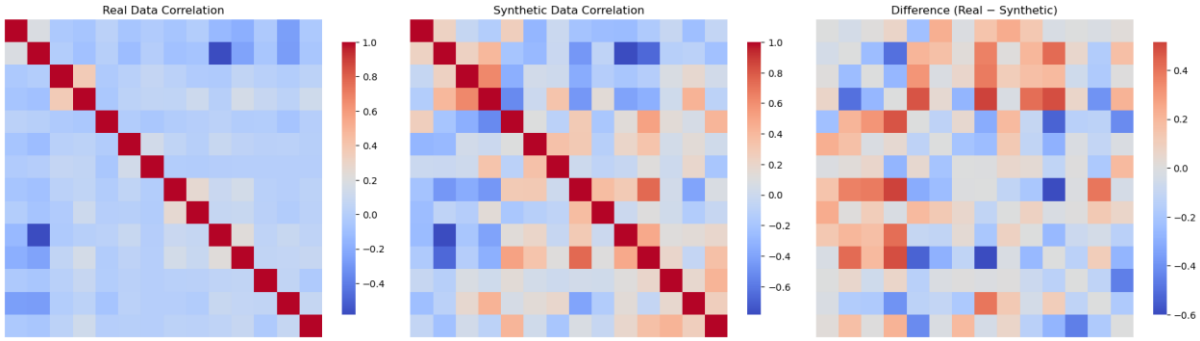
13.1.1. Numerical Columns Distribution Comparison



13.1.2. Categorial Columns Distribution Comparison



13.1.3. Correlation Matrices



13.1.4. Insights

- **Marginals:** Realistic multi-modal peaks (age, education, 40 hr/week) but thin tails on capital-gain/loss and fnlwgt.
- **Categories**: Full coverage of classes, yet major/rare group frequencies deviate by ~10-20%.
- **Dependencies:** Correct correlation signs, but magnitudes off by ±0.2-0.4.

The cGAN covers all features without collapse but still underestimates tails, skews category frequencies by ~10-20%, and misaligns dependency strengths by ±0.2-0.4, targeted regularizers and conditional objectives are needed.

13.1.5. Detection Metric

- **GAN: Mean Detection AUC Score** = 0.99995 , meaning a Random Forest can seamlessly tell fake vs. real.

13.1.6. Efficacy Metric

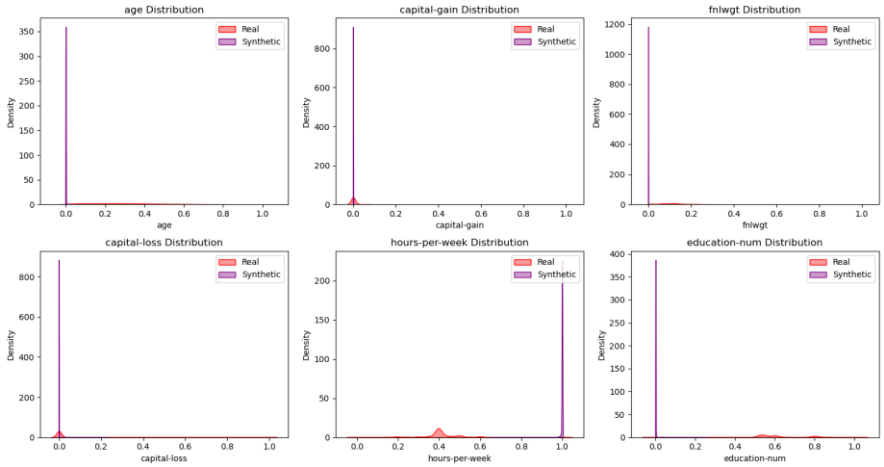
- **Real-trained baseline:** AUC = 0.9068
- **GAN-Synthesized-trained:** AUC = 0.4927
- **Conditional-synthetic Ratio**: 0.4927 / 0.9068 = 0.5655

The GAN is trivially spotted (AUC ≈ 1.0000), yet models trained on its synthetic data drop to AUC = 0.4927 versus 0.9068 on real, a conditional-synthetic ratio of only ~0.54.

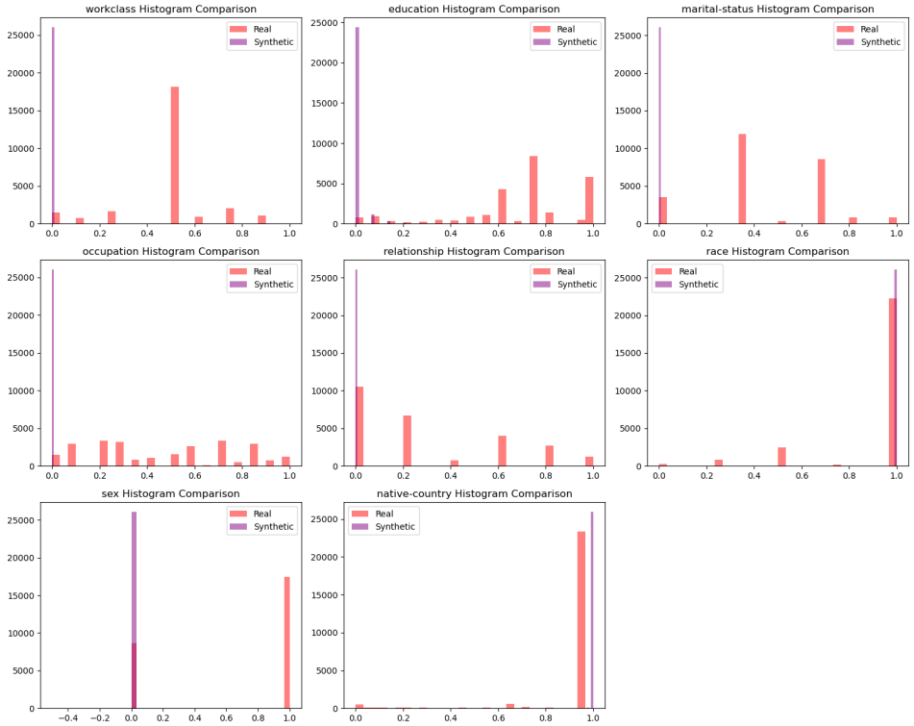
Despite some marginal improvements, the GAN's outputs remain easily flagged (detection AUC ≈ 1.00) and unusable for training (synthetic-trained model AUC = 0.4927 vs. 0.9068 real; efficacy ratio ~ 0.54), reflecting persistent gaps in tails, class priors, and joint structure.

13.2. **cGAN-Synthesized Data Analysis**

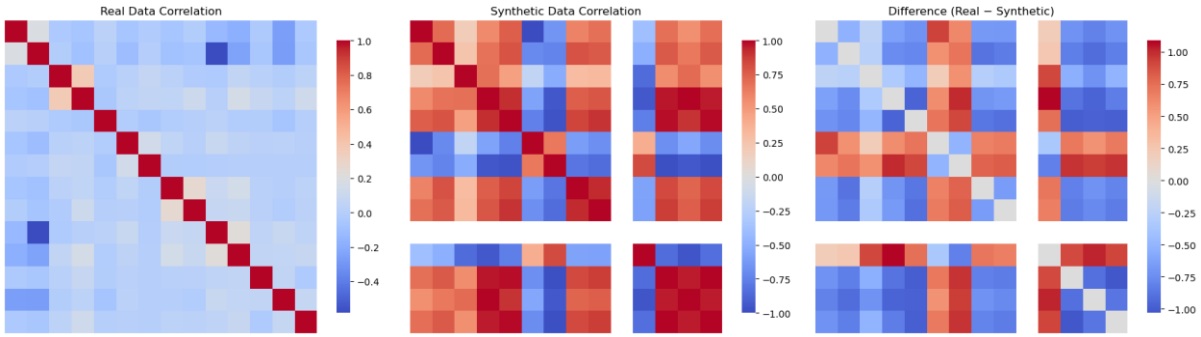
13.2.1. Numerical Columns Distribution Comparison



13.2.2. Categorial Columns Distribution Comparison



13.2.3. Correlation Matrices



13.2.4. Insights

- ☐ **Categorical Collapse:** Nearly every category collapses to one value, synthetic counts concentrate in a single bin, erasing real class diversity.

- ☐ **Numerical Degeneration:** All continuous features (age, fnlwgt, capital-gain/loss, education-num) collapse to zeros (or extremes), losing real multimodal peaks and tails.
- ☐ **Hours-per-Week Failure:** Real data's sharp 40-hour peak is replaced by a synthetic spike at the maximum normalized value.
- ☐ **Correlation Breakdown:** the GAN utterly fails to reproduce the nuanced joint relationships (real correlations around -0.5…+0.8 become essentially uniform), and the difference heatmap shows very large deviations (often 0.5-1.0 in magnitude).

The cGAN has failed: collapsed every feature and category to single values and even inverted correlations, producing data that bear almost no resemblance to the real distribution.

13.2.5. Detection Metric

- **cGAN: Mean Detection AUC Score** = 1.0000 meaning a Random Forest can perfectly tell fake vs. real.

13.2.6. Efficacy Metric

- **Real-trained baseline:** AUC = 0.9068
- **cGAN-Synthesized-trained:** AUC = 0.5129
- **Conditional-synthetic Ratio:** 0.5129 / 0.9068 = 0.5655

The cGAN doesn't fools detectors (AUC = 1.0000) and produces poor training data, models trained on synthetic samples achieve only AUC = 0.5129 versus 0.9068 on real data (conditional-synthetic ratio ≈ 0.5655).

14. Overall Experimental Summary

14.1. GANs vs cGAN Performance

Across three random-seed runs, the vanilla GAN invariably fails to evade detection (Random Forest detector AUC ≈ 1.0) and yields low downstream utility (efficacy ratios ≈ 0.54-0.56). In contrast, the cGAN shows more nuanced behavior, specifically with seed of 42:

- **Seed 42**: although the cGAN can't fool a Random Forest detector (yields mean AUC score of 1.0), it excels at utility: a classifier trained on its synthetic data achieves AUC = 0.9876 (≈ 1.08× the real-trained baseline of 0.9110). This demonstrates that label conditioning produces data highly effective for downstream tasks, even if still distinguishable by a dedicated detector.

14.1.2. Key Trends & Insights

- **Detection** is easy for both GAN and cGAN under Random Forest, regardless of seed or marginal fidelity. Apparently, even though our synthetic data matches marginal distributions well, we assume it still fails to capture subtler joint dependencies. As a result, a Random Forest can easily exploit these residual discrepancies, making detection straightforward for both GAN and cGAN across all seeds.
- **Marginals & correlations** alone fail to guarantee utility or stealth; higher-order and conditional dependencies matter most.
- **Seed** (and by extension hyperparameters) can make or break the cGAN's downstream efficacy, even when detection remains trivial.
- **Label Conditioning** - All three runs used identical label conditioning, feeding the class into both G and D, yet only seed 42 truly captured the conditional feature shifts that power prediction. Seeds 2 and 3, despite matching marginals, stumbled in the optimizer's random walk and never learned the subtle, label, dependent dependencies. By contrast, seed 42's initialization and hyperparameter trajectory landed in a sweet spot where the generator encoded the full conditional structure, producing synthetic data that delivers near, real downstream AUC. In other words, **constant conditioning alone isn't enough**, it's the interplay of initialization, optimization dynamics, and hyperparameters that actually unlocks high, order, label, driven realism.