207547183
206477952

Deep Learning

Assignment 3 - experiment with generative architectures

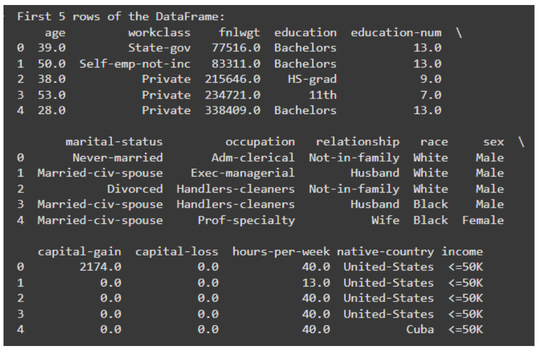
Introduction:

The purpose of this assignment was to explore and experiment with generative models, specifically GANs and cGANs, using tabular data. GANs (Generative Adversarial Networks) are a type of neural network consisting of a Generator, which creates synthetic data, and a Discriminator, which evaluates whether the data is real or synthetic, with both networks trained adversarially. cGANs (Conditional GANs) are an extension of GANs that incorporate conditional inputs, allowing the generation of synthetic data tailored to specific categories or features by conditioning both the Generator and Discriminator on auxiliary information.

Dataset Description:

Data Analysis

The dataset used for this assignment is the Adult dataset, provided in the ARFF format. It contains information about individuals, including demographic and work-related attributes, with the goal of predicting whether an individual's income exceeds $50,000 annually.

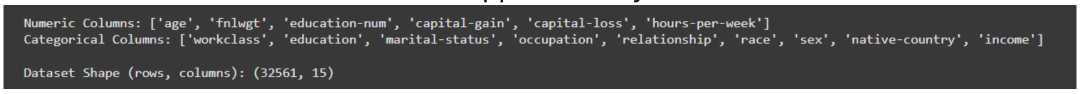


```
First 5 rows of the DataFrame:
    age         workclass    fnlwgt  education  education-num  \
0  39.0         State-gov   77516.0  Bachelors           13.0
1  50.0  Self-emp-not-inc   83311.0  Bachelors           13.0
2  38.0           Private  215646.0    HS-grad            9.0
3  53.0           Private  234721.0       11th            7.0
4  28.0           Private  338409.0  Bachelors           13.0

       marital-status         occupation   relationship   race     sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2            Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female

   capital-gain  capital-loss  hours-per-week native-country income
0        2174.0           0.0            40.0  United-States  <=50K
1           0.0           0.0            13.0  United-States  <=50K
2           0.0           0.0            40.0  United-States  <=50K
3           0.0           0.0            40.0  United-States  <=50K
4           0.0           0.0            40.0           Cuba  <=50K
```

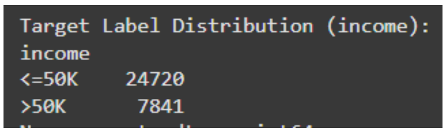
Dataset Summary:

- Total Instances: 32,561
- Total Features: 15
    - 6 Numeric Columns: age, fnlwgt, education-num, capital-gain, capital-loss, and hours-per-week
    - 9 Categorical Columns: workclass, education, marital-status, occupation, relationship, race, sex, native-country, and income
- Target Column: income
    - possible values: <=50K and >50k

  Imbalanced distribution: Approximately **76%** for <=50K and **24%** for >50k.



```
Numeric Columns: ['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
Categorical Columns: ['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'native-country', 'income']

Dataset Shape (rows, columns): (32561, 15)
```



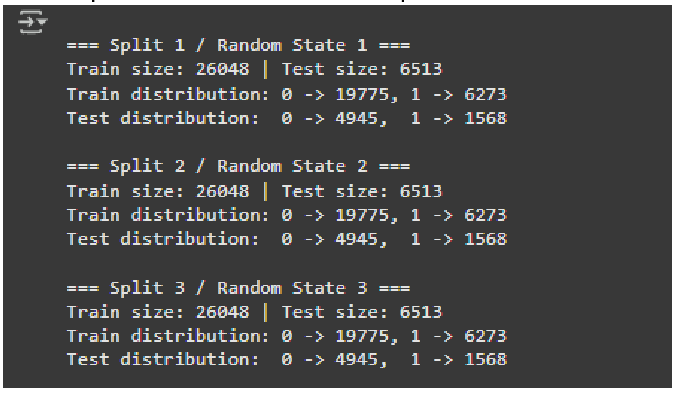
```
Target Label Distribution (income):
income
<=50K    24720
>50K      7841
```

Train-Test Split (Repeated 3 Times)

To ensure fair evaluation and consistent label distribution across training and testing datasets, we performed a stratified train-test split. This approach maintained the original label proportions in both the training and test sets.

Implementation Details:

The income column was transformed using LabelEncoder. Then, the encoded values were converted into a two-column, one-hot vector: 0 for incomes ≤ $50K and 1 for incomes > $50K.
We used StratifiedShuffleSplit to perform a train-test split with:
80% - training data
20% - test data
To ensure reproducibility, we used a fixed set of random seeds ([1, 2, 3]) for generating three splits. Each random seed produced a distinct train-test split.



```
=== Split 1 / Random State 1 ===
Train size: 26048 | Test size: 6513
Train distribution: 0 -> 19775, 1 -> 6273
Test distribution:  0 -> 4945,  1 -> 1568

=== Split 2 / Random State 2 ===
Train size: 26048 | Test size: 6513
Train distribution: 0 -> 19775, 1 -> 6273
Test distribution:  0 -> 4945,  1 -> 1568

=== Split 3 / Random State 3 ===
Train size: 26048 | Test size: 6513
Train distribution: 0 -> 19775, 1 -> 6273
Test distribution:  0 -> 4945,  1 -> 1568
```

Data Preprocessing

To prepare the dataset for model training, we performed preprocessing on the features by scaling continuous features and encoding categorical features. These steps ensure that the data is in a suitable format for input into the GAN and CGAN architectures.

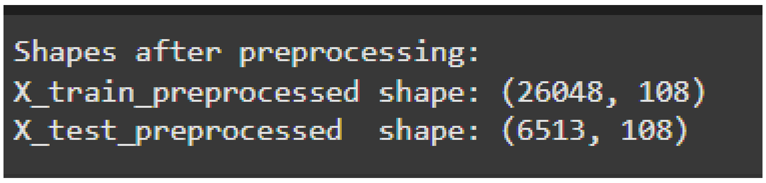
Scaling Continuous Features

We used **MinMaxScaler**, which scales the features to a range of [0, 1].
The scaler was fitted on the training data (X_train[continuous_features]) to compute the scaling parameters (minimum and maximum values for each feature).
The same scaler was then applied to both the training and test data, ensuring consistent scaling across the splits.

Encoding Categorical Features

Categorical features were converted into numerical representations using **OneHotEncoder**:
The encoder was fitted on the training data (X_train[categorical_features]), creating one-hot encoded vectors for each category.
It was then used to transform both the training and test data.
Combining Processed Features

After preprocessing, the scaled continuous features and one-hot encoded categorical features were combined into a single array.



```
Shapes after preprocessing:
X_train_preprocessed shape: (26048, 108)
X_test_preprocessed  shape: (6513, 108)
```

Model Architecture and Components:

<u>**GAN Architecture**</u>

- **Autoencoder:** Learns compressed representations of real data for use in GAN training.
- **Generator:** Creates synthetic embeddings that resemble real data embeddings.
- **Discriminator:** Differentiates between real and synthetic embeddings, guiding the Generator during training.

Autoencoder - Architecture:

Encoder:
The encoder compresses the input into a lower-dimensional latent space (embedding). It includes:
A dense layer mapping the input to 128 hidden units.
ReLU activation for non-linearity.
Dropout with a rate of 0.2 to prevent overfitting.
A linear layer that reduces the dimension to the specified embedding size (embed_dim).
Batch normalization to stabilize learning and improve convergence.

Decoder:
The decoder reconstructs the original input from the latent embedding. It includes:
A dense layer expanding the latent space back to 128 units.
ReLU activation and dropout.
A final layer mapping the output to the original data dimension.
A Sigmoid activation to ensure normalized outputs in the range [0, 1].

Forward Pass:
The input is passed through the encoder to produce a latent embedding, which is then fed into the decoder to reconstruct the input.

Generator - Architecture:

Input: Random noise vector (noise_dim).
Hidden Layer: A dense layer with 128 units, followed by:
Batch normalization to stabilize training.
LeakyReLU activation with a slope of 0.2 for better gradient flow.
Dropout with a rate of 0.2 to prevent overfitting.
Output Layer: A dense layer that maps the hidden representation to the latent embedding dimension (embed_dim).

Forward Pass:
Random noise is mapped to the latent embedding space, which serves as input for the discriminator.

Discriminator - Architecture:

Input: Embedding of size data_dim.
Two hidden layers, each with:
128 units.
LeakyReLU activation with a slope of 0.2.
Batch normalization to stabilize training.
Dropout with a rate of 0.2 to reduce overfitting.
Output: A single neuron that outputs the classification score.

Forward Pass:
The discriminator evaluates whether an input embedding is real or generated.

Layer #	Layer Type	Input Size	Activation Function	Notes
1	Noise	noise_dim	-	Input random noise
2	Dense	128	LeakyReLU (0.2)	Has BatchNorm1d
3	Dense	128	LeakyReLU (0.2)	Has Dropout (0.2)
4	Dense	128	LeakyReLU (0.2)	Has Dropout (0.2)
5	Output	embed_dim (Generator) / 1 (Discriminator)	Sigmoid (for Generator)	Output: embedding (Generator) / Real-Fake score (Discriminator)

cGAN Architecture

- **Autoencoder**: Learns compressed representations of real data for use in cGAN training
- **generator**: Creates synthetic embeddings conditioned on additional input, ensuring synthetic data closely resembles real data embeddings.
- **Discriminator**: Differentiates between real and synthetic embeddings, leveraging conditional input to guide the Generator during training.

Autoencoder - Architecture:

Encoder:
The encoder compresses the input into a lower-dimensional latent space (embedding). It includes:
A dense layer mapping the input to 128 hidden units.
LeakyReLU activation with a slope of 0.2 for non-linearity.
Dropout with a rate of 0.1 to prevent overfitting.
A linear layer that reduces the hidden dimension to half (hidden_dim // 2).
Another LeakyReLU activation.
A final linear layer that maps to the specified embedding size (embed_dim).

Decoder:
The decoder reconstructs the original input from the latent embedding. It includes:
A dense layer expanding the latent space back to half the hidden dimension (hidden_dim // 2).
LeakyReLU activation with a slope of 0.2.
Dropout with a rate of 0.5.
A dense layer expanding back to 128 units.

Another LeakyReLU activation.
A final layer mapping the output to the original data dimension with a Sigmoid activation to ensure normalized outputs in the range [0, 1].

Forward Pass:
The input is passed through the encoder to produce a latent embedding, which is then fed into the decoder to reconstruct the input.

Generator - Architecture:

• Input: Concatenation of random noise vector (noise_dim) and conditional input (condition_dim).
• Hidden Layer:

- A dense layer with 256 units.
- Batch normalization to stabilize training.
- LeakyReLU activation with a slope of 0.2.
- Dropout with a rate of 0.5.
- A second dense layer reducing the hidden dimension to half (hidden_dim // 2).
- Batch normalization and another LeakyReLU activation.
  • Output Layer: A dense layer that maps the hidden representation to the latent embedding dimension (embed_dim), followed by a Tanh activation.

Forward Pass:
The concatenated random noise and conditional input are mapped to the latent embedding space, which serves as input for the discriminator.

Discriminator - Architecture:

• Input: Concatenation of the embedding (size data_dim) and the conditional input (condition_dim).
• Hidden Layers:

- A dense layer with 256 units using spectral normalization.
- LeakyReLU activation with a slope of 0.2.
- Dropout with a rate of 0.4.
- A second dense layer reducing the hidden dimension to half (hidden_dim // 2), also using spectral normalization.
- Another LeakyReLU activation and a dropout rate of 0.5.
  • Output: A single neuron that outputs the classification score.

Forward Pass:
The embedding and conditional input are concatenated and passed into the first hidden layer. The output is then passed through the second hidden layer and finally fed into the output layer, producing a classification score indicating whether the embedding is real or synthetic.

Layer #	Architecture	Layer Type	Input Size	Output Size	Activation and Notes
G1	Generator	**Input** (concatenate noise **z** and condition **c**)	(N, noise_dim + condition_dim)	--	No activation; just preparing combined input.
G2	Generator	**Linear** + **BatchNorm1d** + **LeakyReLU(0.2)** + **Dropout(0.5)**	(N, noise_dim + condition_dim)	(N, 256)	Hidden dimension = 256; dropout probability = 0.5
G3	Generator	**Linear** + **BatchNorm1d** + **LeakyReLU(0.2)**	(N, 256)	(N, 128)	Hidden dimension = 128
G4	Generator	**Output**: **Linear** → embed_dim,	(N, 128)	(N, embed_dim)	Final embedding output in the range [-1, 1] via Tanh.
D1	Discriminator	Input (concatenate data x or generated x_hat with condition c)	(N, data_dim + condition_dim)	--	No activation; just preparing combined input.
D2	Discriminator	Linear + SpectralNorm + LeakyReLU(0.2) + Dropout(0.4)	(N, data_dim + condition_dim)	(N, 256)	Hidden dimension = 256; dropout probability = 0.4.
D3	Discriminator	Linear + SpectralNorm + LeakyReLU(0.2) + Dropout(0.5)	(N, 256)	(N, 256)	Hidden dimension = 128; dropout probability = 0.5.
D4	Discriminator	Output: Linear → 1	(N, 128)	(N, 128)	Real or fake logit score; typically used with a sigmoid-based loss (for example, BCEWithLogitsLoss).

GAN

Training Process Gan:

We trained a **GAN model integrated with an Autoencoder** to improve the quality of synthetic data generation. The model consists of three main components: an **Autoencoder**, a **Generator**, and a **Discriminator**. The training process was carried out over multiple epochs and involved the following key steps:

1. **Training the Autoencoder:**
   - Real samples are passed through the Autoencoder to produce embeddings and reconstructions.
   - The reconstruction loss is calculated as the difference between the reconstructed samples and the original real samples.
   - The Autoencoder's weights are updated using backpropagation to minimize this loss, enabling it to learn meaningful embeddings of the data.
2. **Training the Discriminator:**
   - The Discriminator is trained to differentiate between real and synthetic samples.
   - Real samples are passed directly into the Discriminator, while fake samples are generated by feeding random noise into the Generator and decoding the resulting embeddings using the Autoencoder.

    - The Discriminator loss is computed as the sum of:
        - Loss for real samples (should classify as real).
        - Loss for fake samples (should classify as fake).
    - The Discriminator's weights are updated to enhance its ability to correctly classify real and synthetic data.
3. **Training the Generator:**
    - Random noise is fed into the Generator to create synthetic embeddings.
    - These embeddings are decoded by the Autoencoder into fake samples, which are then evaluated by the Discriminator.
    - The Generator loss is calculated as its inability to fool the Discriminator into classifying the fake samples as real.
    - The Generator's weights are updated to improve its ability to generate realistic synthetic data.

We conducted multiple experiments with different hyperparameter combinations to evaluate the model's performance in generating synthetic data that closely resembles real data.

Optimizers and Loss Functions

To ensure efficient and stable training, we used separate **Adam optimizers** for the Autoencoder, Generator, and Discriminator, each with tailored learning rates:

- **Autoencoder and Generator Optimizer:** Learning rate (gen_lr) set to **0.0001**.
- **Discriminator Optimizer:** Learning rate (disc_lr) set to **0.00001**.

The loss functions used during training were:

- **Reconstruction Loss (MSE):** For the Autoencoder, to minimize the difference between the original and reconstructed samples.
- **GAN Loss (BCEWithLogitsLoss):** For the Generator and Discriminator, to evaluate their adversarial interaction.

These optimizers and loss functions were chosen to provide a balanced training process, ensuring stability and preventing issues like mode collapse or oscillations.

Fixed Parameters

The following parameters were kept constant during training to ensure consistent comparisons:

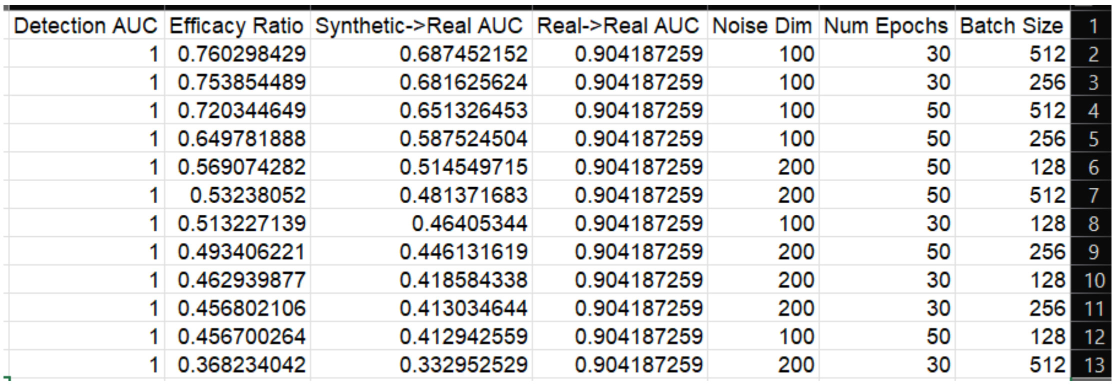
- **gen_lr:** 0.0001 for the Generator and Autoencoder.
- **disc_lr:** 0.00001 for the Discriminator, ensuring it learns at a slower pace to avoid overpowering the Generator.
- **embed_dim:** The latent embedding dimension, set to **32**.

207547183
206477952

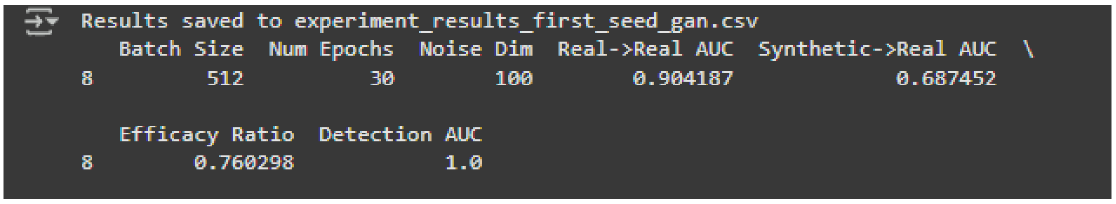
Three different seeds: 1,2,3.

First seed = 1

Results:



Detection AUC	Efficacy Ratio	Synthetic->Real AUC	Real->Real AUC	Noise Dim	Num Epochs	Batch Size	1
1	0.760298429	0.687452152	0.904187259	100	30	512	2
1	0.753854489	0.681625624	0.904187259	100	30	256	3
1	0.720344649	0.651326453	0.904187259	100	50	512	4
1	0.649781888	0.587524504	0.904187259	100	50	256	5
1	0.569074282	0.514549715	0.904187259	200	50	128	6
1	0.53238052	0.481371683	0.904187259	200	50	512	7
1	0.513227139	0.46405344	0.904187259	100	30	128	8
1	0.493406221	0.446131619	0.904187259	200	50	256	9
1	0.462939877	0.418584338	0.904187259	200	30	128	10
1	0.456802106	0.413034644	0.904187259	200	30	256	11
1	0.456700264	0.412942559	0.904187259	100	50	128	12
1	0.368234042	0.332952529	0.904187259	200	30	512	13



```
Results saved to experiment_results_first_seed_gan.csv
   Batch Size  Num Epochs  Noise Dim  Real->Real AUC  Synthetic->Real AUC  \
8         512          30        100        0.904187             0.687452

   Efficacy Ratio  Detection AUC
8        0.760298            1.0
```

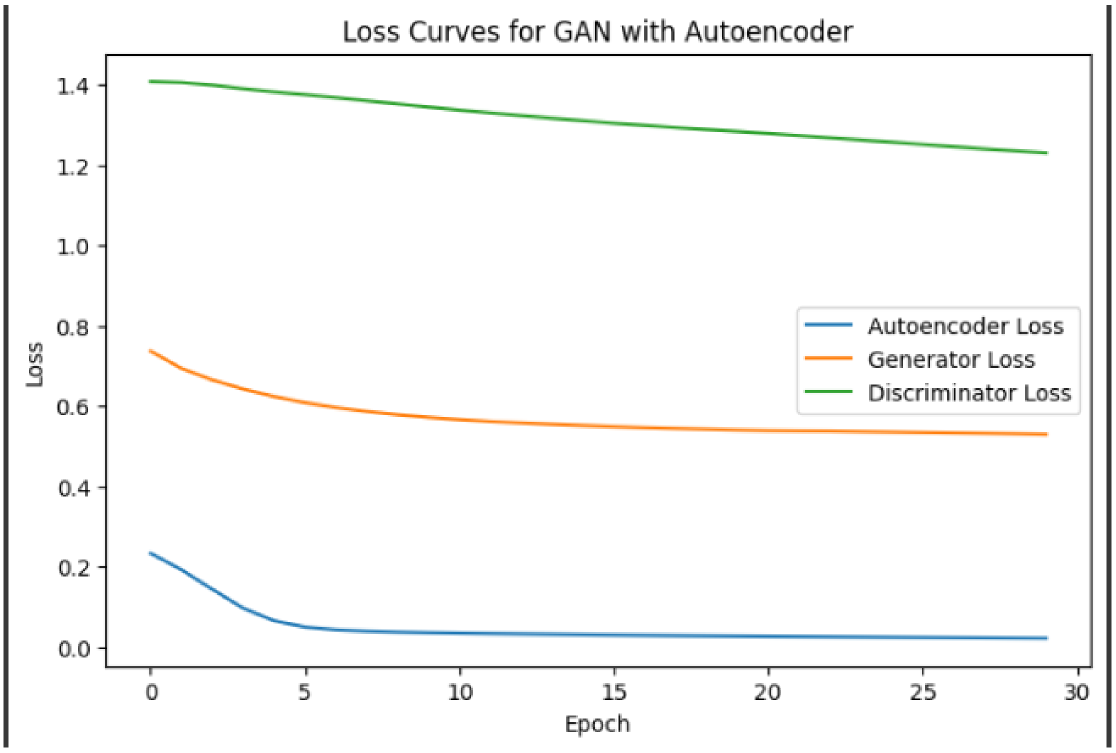
Analysis of the Best Results:

The best configuration is achieved with:

Batch Size: 512

Num Epochs: 30

Noise Dim: 100



The graph illustrates the training dynamics of the GAN, including the Generator, Discriminator, and Autoencoder, over 30 epochs:

1. Autoencoder Loss
   - Starts around 0.25 in the first epoch.
   - Declines steadily and reaches about 0.02 by epoch.
   - This consistent drop to a low value indicates that the Autoencoder is successfully learning to reconstruct the real data.
2. Generator Loss
   - Begins at about 0.76 and steadily decreases to around 0.53 by epoch 30.
   - The downward trend implies the Generator is improving its ability to produce more realistic samples that fool the Discriminator.
3. Discriminator Loss
   - The total Discriminator loss starts around 1.41 and gradually falls to roughly 1.22.
   - This decreasing trend shows the Discriminator remains capable of distinguishing real from synthetic data while not completely overpowering the Generator.

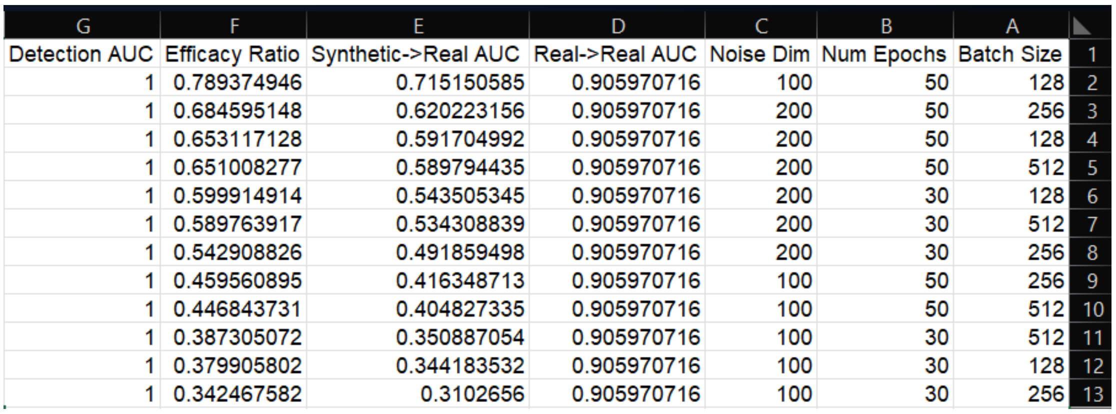
Overall, all three loss curves move downward in a relatively stable fashion, suggesting that the GAN is training effectively without mode collapse or instability. The Generator gets progressively better at "fooling" the Discriminator, while the Autoencoder continues to refine its reconstructions, and the Discriminator still improves its classification accuracy.

Model Evaluation Metrics Detection and Efficacy:

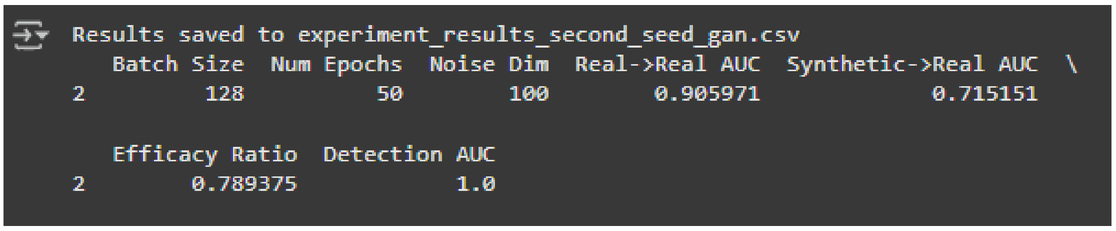
- **Real → Real AUC (0.9042):** Indicates excellent performance of the classifier when both training and testing are done on real data, serving as the gold-standard baseline.
- **Synthetic → Real AUC (0.6875):** Reflects moderate success of a classifier trained on synthetic data at recognizing patterns in real data, leaving some room for improvement.
- **Efficacy Ratio (0.7603):** Suggests that the synthetic data retains roughly 76% of the predictive capability found in the real dataset, highlighting its practical utility for model training.
- **Detection AUC (1.000):** Shows that real and synthetic data can be perfectly distinguished—a common outcome in GAN settings, as achieving complete indistinguishability is inherently difficult.

Second seed = 2

Results:



G	F	E	D	C	B	A	
Detection AUC	Efficacy Ratio	Synthetic->Real AUC	Real->Real AUC	Noise Dim	Num Epochs	Batch Size	1
1	0.789374946	0.715150585	0.905970716	100	50	128	2
1	0.684595148	0.620223156	0.905970716	200	50	256	3
1	0.653117128	0.591704992	0.905970716	200	50	128	4
1	0.651008277	0.589794435	0.905970716	200	50	512	5
1	0.599914914	0.543505345	0.905970716	200	30	128	6
1	0.589763917	0.534308839	0.905970716	200	30	512	7
1	0.542908826	0.491859498	0.905970716	200	30	256	8
1	0.459560895	0.416348713	0.905970716	100	50	256	9
1	0.446843731	0.404827335	0.905970716	100	50	512	10
1	0.387305072	0.350887054	0.905970716	100	30	512	11
1	0.379905802	0.344183532	0.905970716	100	30	128	12
1	0.342467582	0.3102656	0.905970716	100	30	256	13



```
Results saved to experiment_results_second_seed_gan.csv
   Batch Size  Num Epochs  Noise Dim  Real->Real AUC  Synthetic->Real AUC  \
2         128          50        100        0.905971             0.715151

   Efficacy Ratio  Detection AUC
2        0.789375            1.0
```

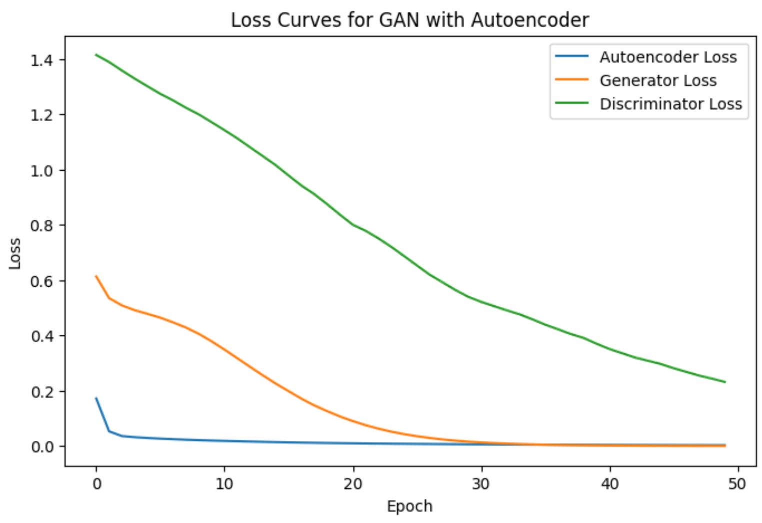
Analysis of the Best Results:

The best configuration is achieved with:

Batch Size: 128

Num Epochs: 50

Noise Dim: 100



The graph illustrates the training dynamics of the GAN, including the Generator, Discriminator, and Autoencoder, over 50 epochs:

1. **Autoencoder Loss (Blue):**
   - Starts around *0.25* in the first steps and rapidly falls below *0.01* by later epochs.

- This steep drop indicates that the Autoencoder quickly learns to reconstruct real samples with high fidelity and converges to a very low error.

2. **Generator Loss (Orange):**
   - Begins near *0.72* in the early steps and steadily decreases.
   - Such a large reduction suggests the Generator becomes progressively better at producing "realistic" synthetic samples that can fool the Discriminator.
3. **Discriminator Loss (Green):**
   - Initially around *1.40* and steadily declines to roughly *0.20–0.30* toward the end.
   - The lower Discriminator loss implies it is becoming more confident and accurate at distinguishing real data from the Generator's outputs.

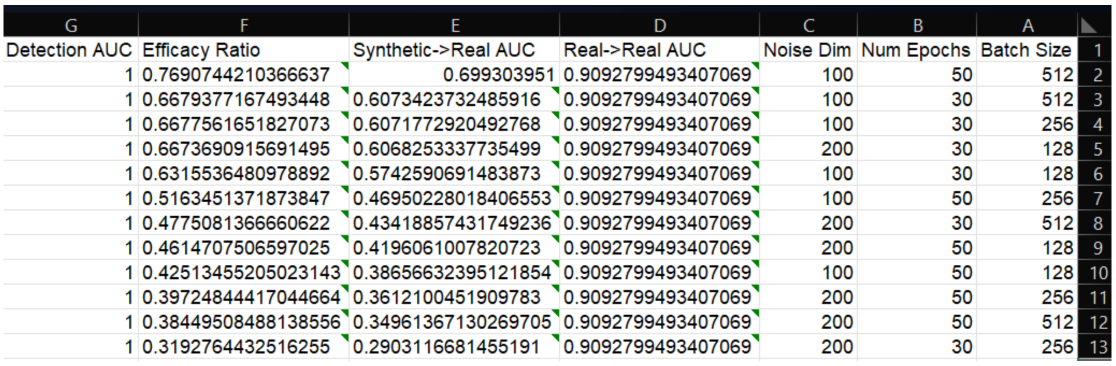
Because both the Generator's and Discriminator's losses continue to decrease without diverging or oscillating wildly, the training appears stable. The Generator does not overwhelm the Discriminator (nor vice versa), and the Autoencoder reliably reconstructs training samples. Overall, this balance suggests that the model reaches an equilibrium wherein the Generator's outputs become increasingly realistic, and the Discriminator remains sufficiently trained to differentiate them from genuine data.

Model Evaluation Metrics Detection and Efficacy:

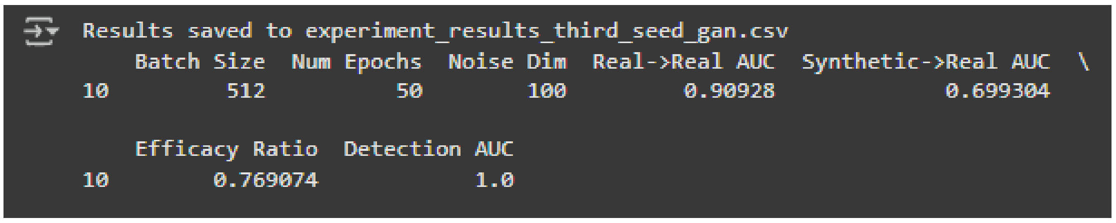
- **Real → Real AUC (0.9060):** Demonstrates strong baseline accuracy when both training and testing occur on real data.
- **Synthetic → Real AUC (0.7152):** Indicates a notable improvement in the classifier's ability to generalize to real data patterns from synthetic training, showing continued progress.
- **Efficacy Ratio (0.7894):** Implies that the synthetic data preserves roughly 78.9% of the real data's predictive power, underscoring its usefulness for training.
- **Detection AUC (1.0):** Confirms that real and synthetic data remain distinguishable, which is common in GAN setups where perfect indistinguishability is difficult to achieve.

Third seed = 3

207547183
206477952



G	F	E	D	C	B	A	
Detection AUC	Efficacy Ratio	Synthetic->Real AUC	Real->Real AUC	Noise Dim	Num Epochs	Batch Size	1
1	0.7690744210366637	0.699303951	0.9092799493407069	100	50	512	2
1	0.6679377167493448	0.6073423732485916	0.9092799493407069	100	30	512	3
1	0.6677561651827073	0.6071772920492768	0.9092799493407069	100	30	256	4
1	0.6673690915691495	0.6068253337735499	0.9092799493407069	200	30	128	5
1	0.6315536480978892	0.5742590691483873	0.9092799493407069	100	30	128	6
1	0.5163451371873847	0.46950228018406553	0.9092799493407069	100	50	256	7
1	0.4775081366660622	0.43418857431749236	0.9092799493407069	200	30	512	8
1	0.4614707506597025	0.4196061007820723	0.9092799493407069	200	50	128	9
1	0.42513455205023143	0.38656632395121854	0.9092799493407069	100	50	128	10
1	0.39724844417044664	0.3612100451909783	0.9092799493407069	200	50	256	11
1	0.38449508488138556	0.34961367130269705	0.9092799493407069	200	50	512	12
1	0.3192764432516255	0.2903116681455191	0.9092799493407069	200	30	256	13



```
Results saved to experiment_results_third_seed_gan.csv
    Batch Size  Num Epochs  Noise Dim  Real->Real AUC  Synthetic->Real AUC  \
10         512          50        100         0.90928             0.699304

    Efficacy Ratio  Detection AUC
10        0.769074            1.0
```

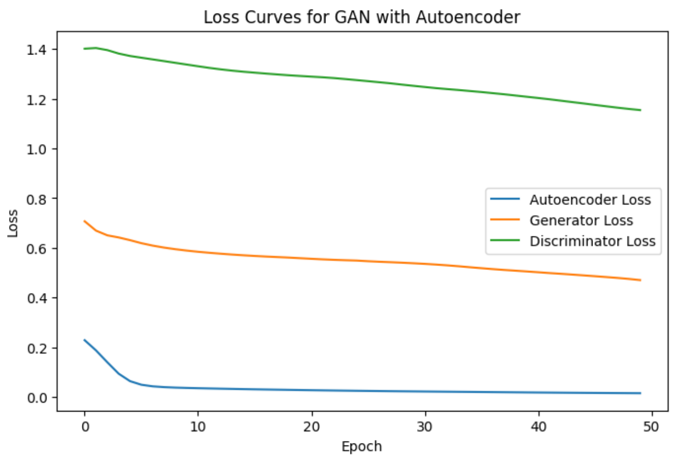
Analysis of the Best Results:

The best configuration is achieved with:

Batch Size: 512

Num Epochs: 50

Noise Dim: 100



The graph illustrates the training dynamics of the GAN, including the Generator, Discriminator, and Autoencoder, over 50 epochs:

1. **Autoencoder Loss (Blue):** Quickly falls from about 0.2 to near 0.01 within the first few epochs, then remains low and stable. This rapid decrease indicates that the Autoencoder is effective at learning to reconstruct real data.

2. **Generator Loss (Orange):** Starts relatively high (around 0.8) and steadily decreases to roughly 0.4 by epoch 50. This downward trend suggests the Generator becomes increasingly adept at producing realistic samples that challenge the Discriminator.

3. **Discriminator Loss (Green):** Gradually decreases from approximately 1.4 to about 1.0 over the 50 epochs, reflecting its improved ability to distinguish real data from the Generator's outputs.

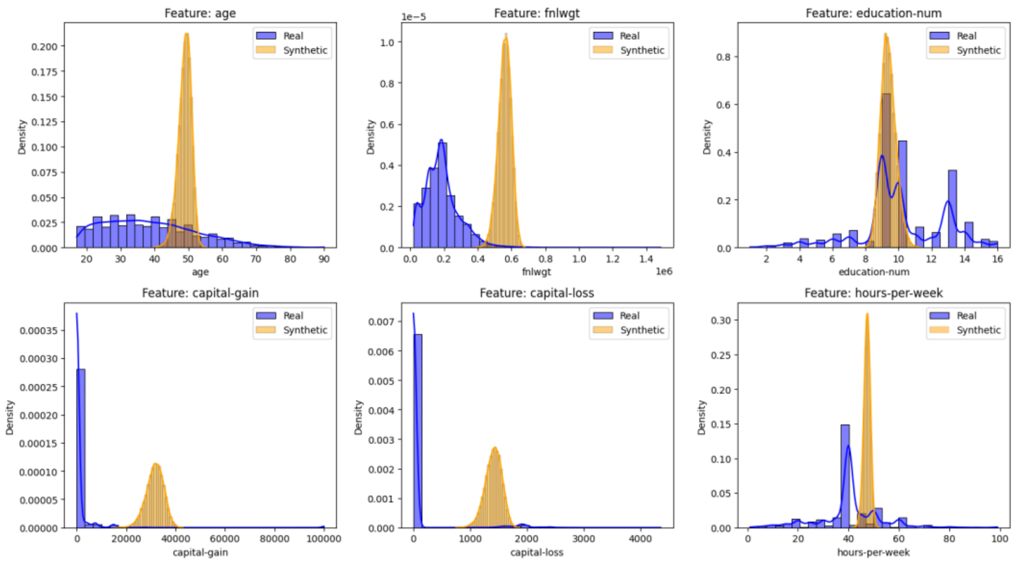
Overall, both the Generator and Discriminator continue to improve without destabilizing the training process. The Generator's outputs become more convincing over time, while the Discriminator remains effective at identifying synthetic versus real inputs.

Model Evaluation Metrics Detection and Efficacy:

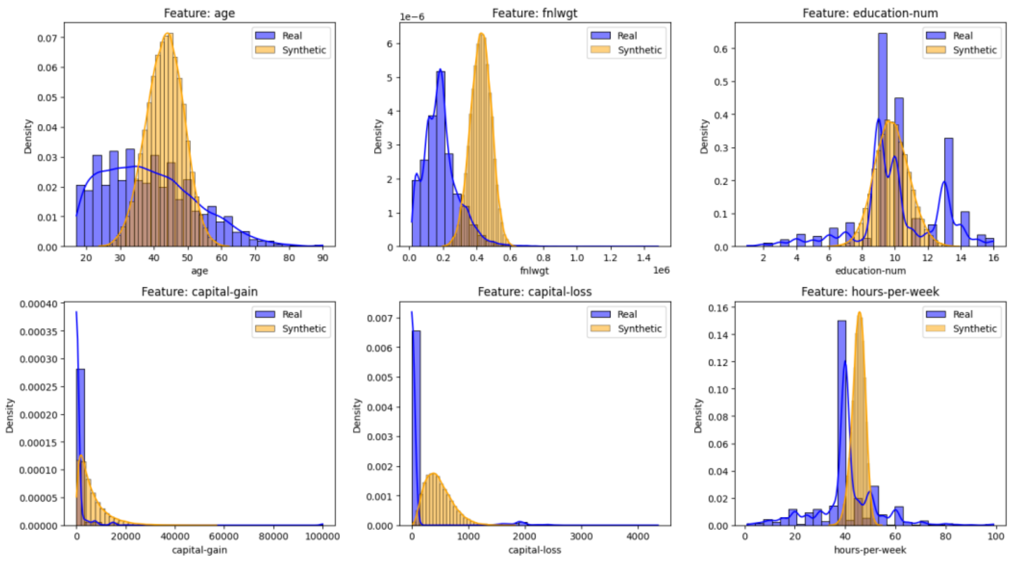
4. **Real → Real AUC (0.9093):** Demonstrates strong baseline performance of a classifier trained and tested on real data.
5. **Synthetic → Real AUC (0.6993):** Indicates moderate success of a classifier trained on synthetic data in recognizing patterns in real data, though there's room for further improvement.
6. **Efficacy Ratio (0.7691):** Suggests that the synthetic data retains about 76.9% of the predictive power of the real dataset, making it reasonably effective for training.
7. **Detection AUC (1.0):** Confirms that real and synthetic samples are distinguishable, which is not uncommon in GAN setups, as perfect indistinguishability is difficult to attain.

Visualizations comparing real vs. synthetic feature distributions

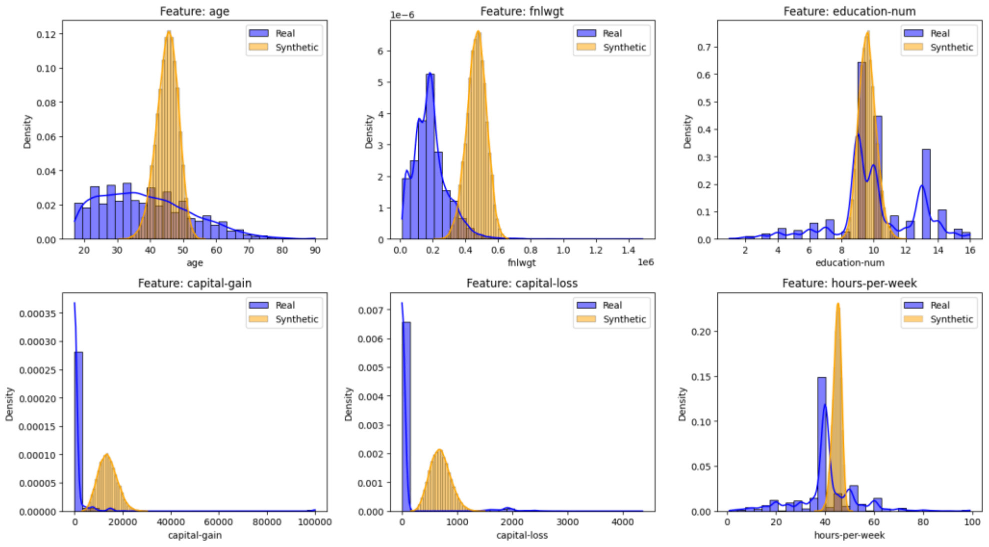
Seed 1:



Seed = 2:



Seed = 3:



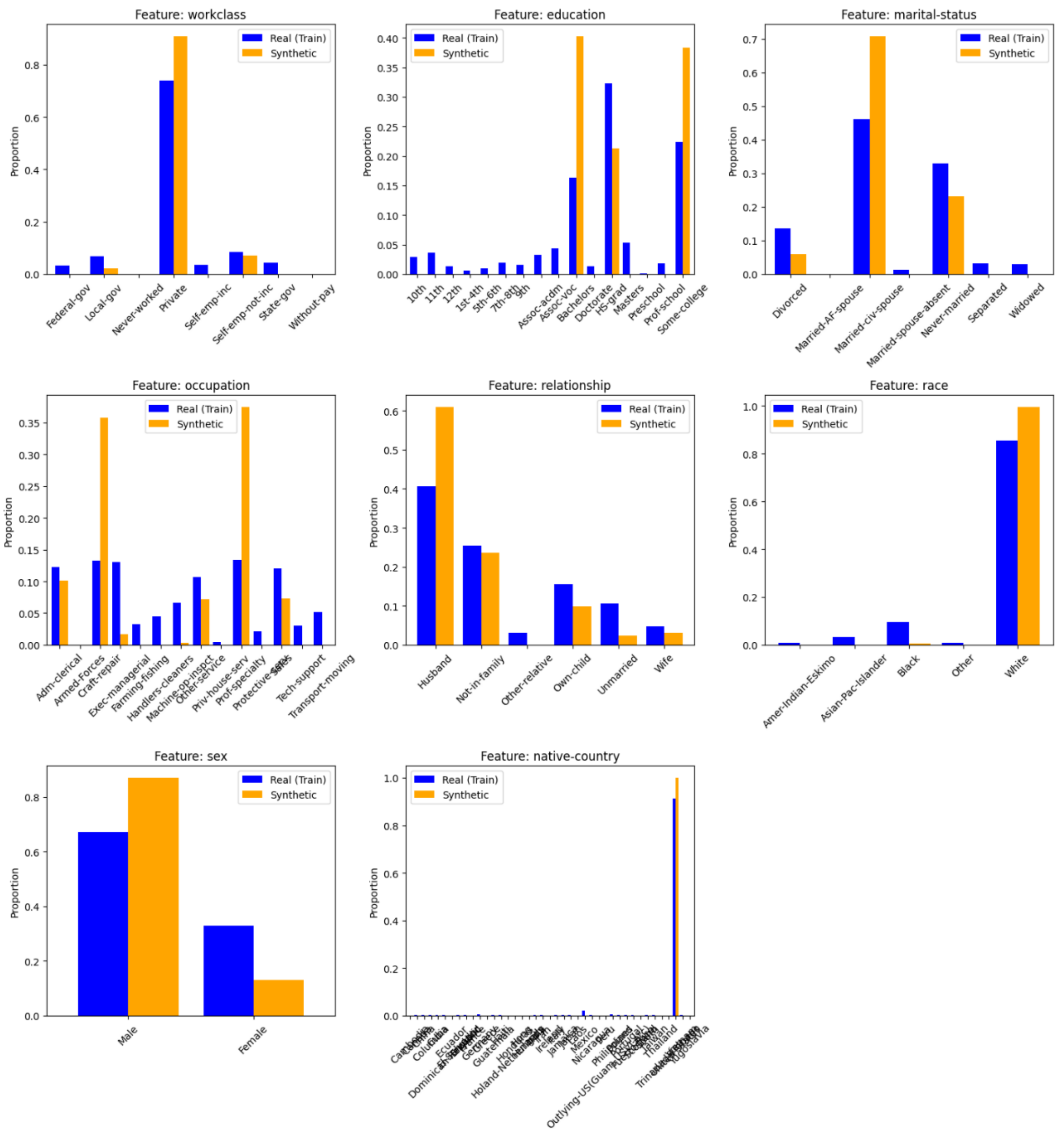
Looking across all three random seeds, we can observe consistent patterns in how the synthetic data (orange) compares to the real data (blue). The age distribution consistently shows the synthetic data being more concentrated around ages 45-50, while the real data spreads more evenly across ages 20-60. For final weight (fnlwgt), all three seeds show the same misalignment.

The education-num feature shows good alignment at the primary peak (around 9-10) across all seeds, though the synthetic data consistently fails to capture the secondary peaks present in the real data at higher values. Capital-gain and capital-loss features show similar challenges across all seeds - the real data has sharp spikes near zero, while the synthetic data produces smoother, more spread-out distributions that don't capture this pattern.

Hours-per-week shows the most consistency between real and synthetic data across all seeds, with both showing a strong peak around 40 hours (standard full-time work week). However, in all cases, the synthetic data produces a sharper, more concentrated peak compared to the real data's wider distribution that better captures part-time and overtime hours.

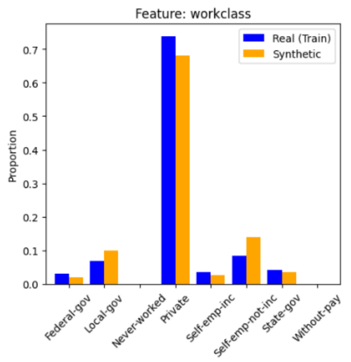
This consistency across different random seeds suggests that these patterns are inherent to the synthetic data generation process rather than random variations, highlighting both the strengths and limitations of the current approach.

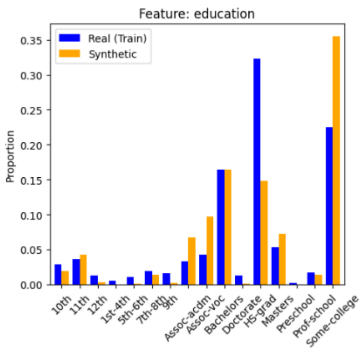
Seed 1:

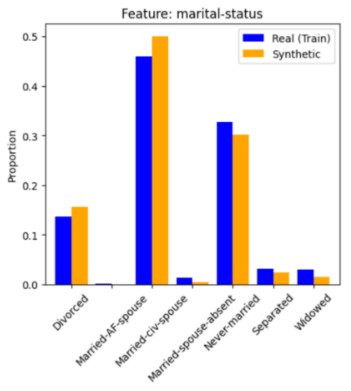


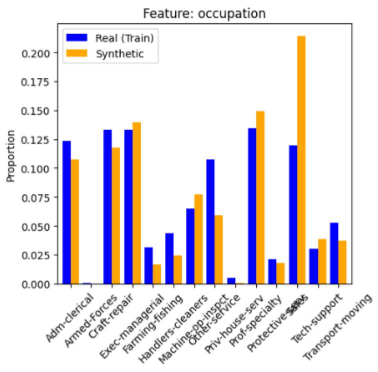
207547183
206477952

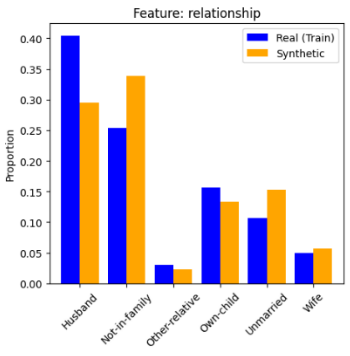
Seed = 2:

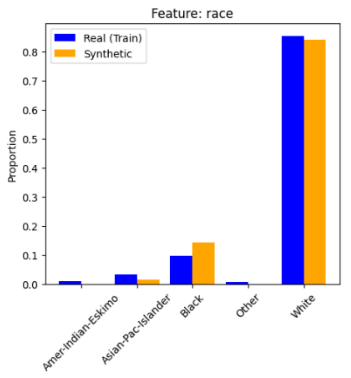


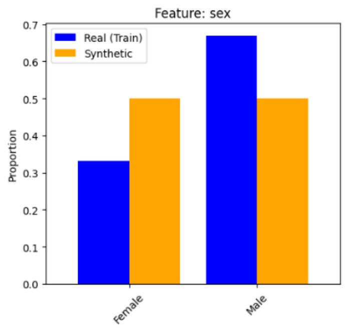


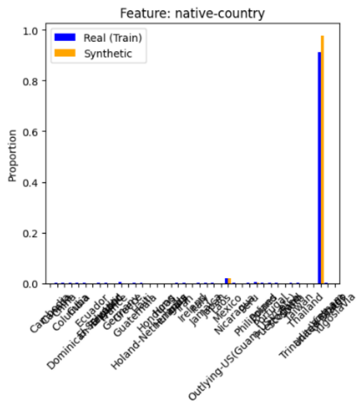




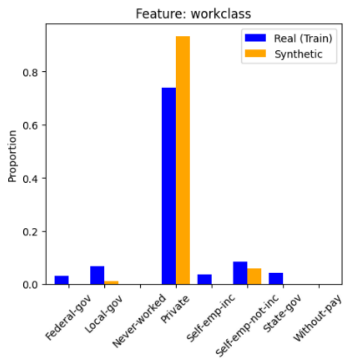


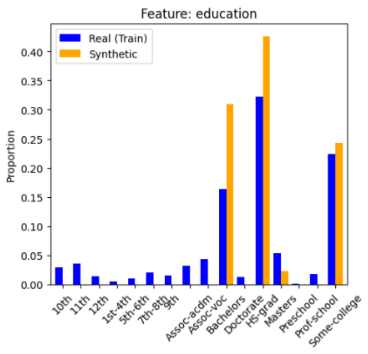


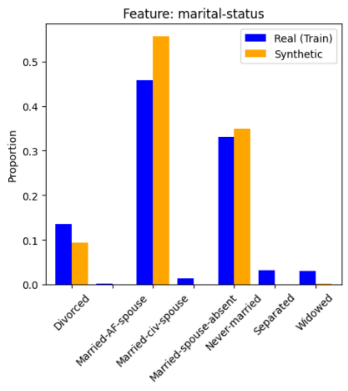


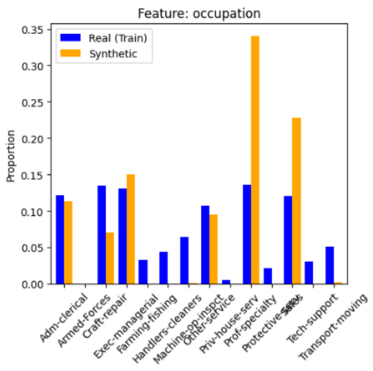


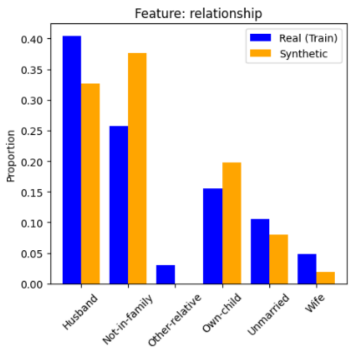
Seed = 3:

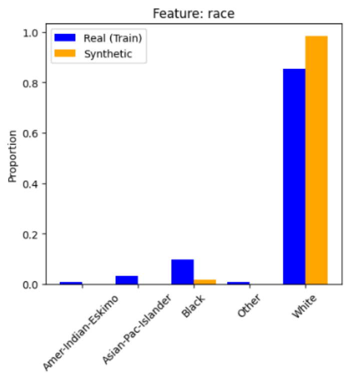


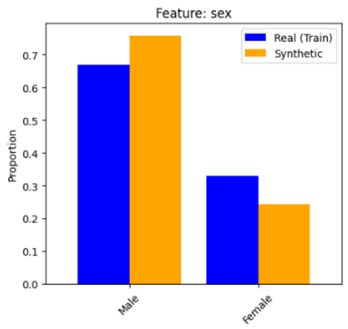


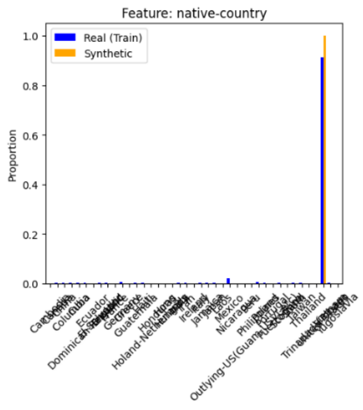












Looking across all three random seeds, we can observe consistent patterns in how the synthetic data (orange) compares to the real data (blue) for categorical features. In workclass, "Private" consistently dominates in both distributions, though with varying proportions across seeds. Education shows similar patterns across seeds, with some fluctuation in categories like "HS-grad" and "Some-college."

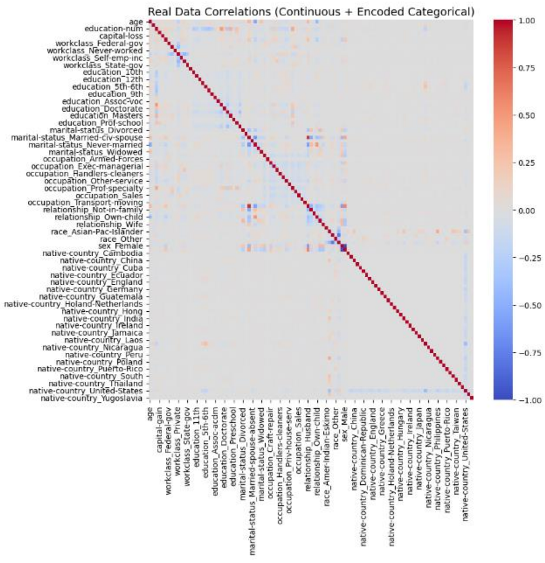
For relationship status, the synthetic data shows varying degrees of overrepresentation across seeds. Race distributions maintain good alignment across all seeds, with "White" being the predominant category. Native-country shows remarkably consistent behavior across all seeds, with both real and synthetic data having almost identical high concentrations in one category.

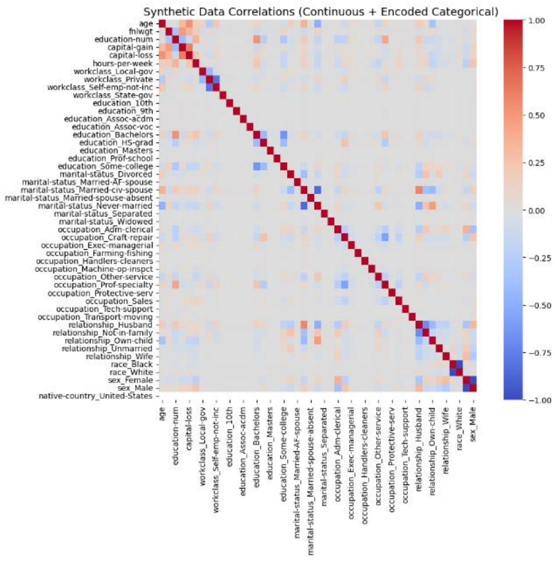
The most noticeable variations between seeds appear in the sex distribution - while all seeds maintain the male majority, the exact proportions and the gap between male and female categories show more variation. Occupation categories also show some inconsistency across seeds, particularly in professional and technical categories.

207547183
206477952

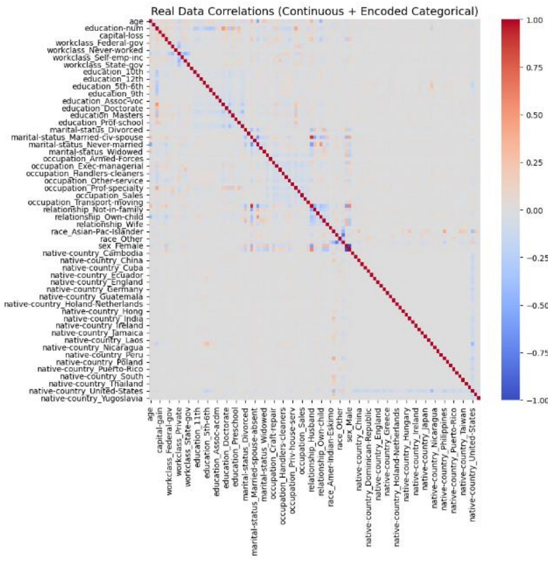
This seed-to-seed comparison suggests that while the synthetic data generally captures the main categorical patterns of the real data, some features (like native-country and race) are more stable across different random initializations than others (like sex and occupation).

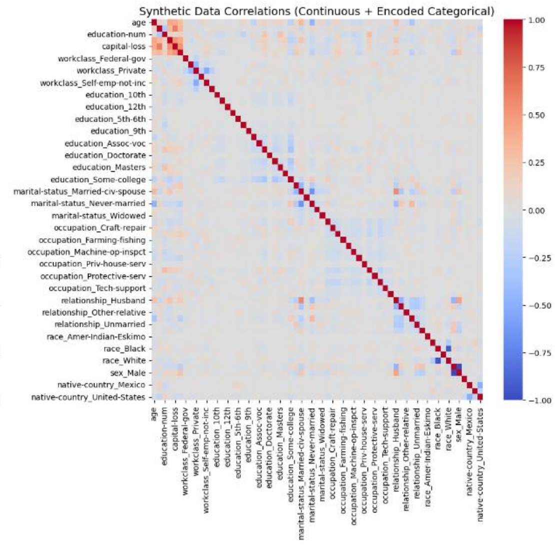
Seed = 1:



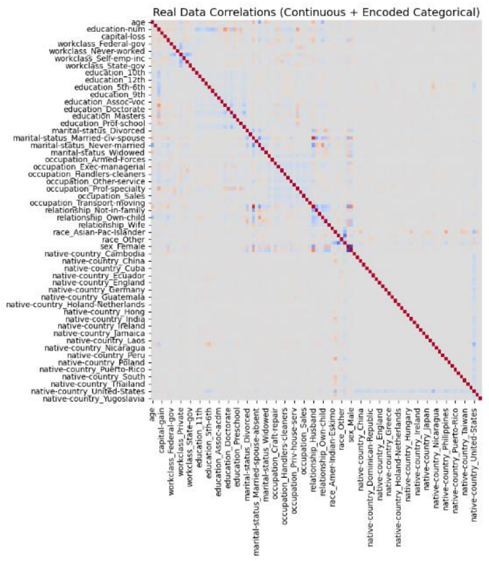


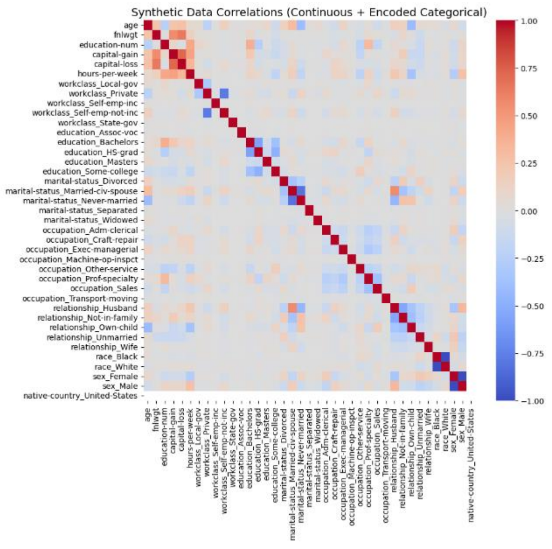
Seed = 2:





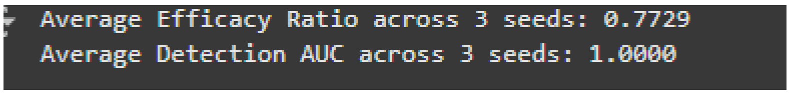
Seed = 3:





These correlation matrices compare how variables interrelate in the real dataset (left) versus the synthetic dataset (right). In both matrices, the diagonal (dark red) indicates perfect self-correlations, which is expected. Off the diagonal, the real dataset exhibits a greater number of strong correlations, suggesting richer and more nuanced relationships among variables. While the synthetic dataset does capture the main correlation patterns, it tends to be sparser and shows weaker relationships overall. Consequently, although the synthetic data preserves basic variable interactions, it does not fully replicate the subtle correlations found in the real dataset.

Conclusion from the 3 Seeds:



```
Average Efficacy Ratio across 3 seeds: 0.7729
Average Detection AUC across 3 seeds: 1.0000
```

Based on the three random seeds' results and the metrics shown (Average Efficacy Ratio: 0.7729, Average Detection AUC: 1.0000), we can conclude that the synthetic data generator produces relatively consistent results across different seeds. The perfect Detection AUC (1.0000) suggests the synthetic data maintains clear distinguishability from real data, while the Efficacy Ratio of 0.7729 indicates that the synthetic data captures about 77% of the real data's patterns on average.Nonetheless, there is still room for refinement to further narrow the gap between synthetic and real data performance. The generator shows particular strength in replicating certain features while consistently struggling with others across all seeds. This consistency across seeds suggests these are inherent characteristics of the model rather than random variations.

The evaluation across three different seeds underscores both the promise and the remaining challenges of the synthetic data generation process.

207547183
206477952

CGAN

Training Process CGAN:

We trained a **Conditional GAN (CGAN) model integrated with an Autoencoder** to enhance the quality of synthetic data generation. The CGAN architecture includes three key components: a Conditional Autoencoder, a Conditional Generator, and a Conditional Discriminator. These components were trained over multiple epochs, following these main steps:

1. **Training the Conditional Autoencoder:**
   - Real samples and their corresponding conditional labels are passed through the Autoencoder to generate embeddings and reconstructions.
   - The reconstruction loss is calculated as the difference between the reconstructed samples and the original real samples.
   - The Autoencoder's weights are updated using backpropagation to minimize this loss, allowing it to learn meaningful embeddings conditioned on the labels.

2. **Training the Conditional Discriminator:**
   - The Discriminator is tasked with distinguishing between real and synthetic samples, conditioned on the input labels.
   - Real samples, along with their labels, are directly passed to the Discriminator.
   - Fake samples are created by feeding random noise and conditional labels into the Generator, and these embeddings are decoded into fake samples by the Autoencoder.
   - The Discriminator loss is calculated as the sum of:
     - Loss for real samples, which should be classified as real.
     - Loss for fake samples, which should be classified as fake.
   - The Discriminator's weights are updated to improve its ability to correctly classify real and synthetic samples, conditioned on the labels.

3. **Training the Conditional Generator:**
   - Random noise and conditional labels are fed into the Generator to create synthetic embeddings.
   - These embeddings are decoded into fake samples using the Autoencoder, which are then evaluated by the Discriminator.
   - The Generator loss is calculated based on its ability to fool the Discriminator into classifying the fake samples as real.
   - The Generator's weights are updated to improve its ability to generate realistic and label-consistent synthetic data.

We conducted experiments across different hyperparameter combinations to evaluate the CGAN's performance in generating synthetic data that closely resembles real data, while maintaining label consistency.

Optimizers and Loss Functions

To ensure stable and efficient training, we used separate Adam optimizers for the Autoencoder, Generator, and Discriminator, each with tailored learning rates:

- **Autoencoder and Generator Optimizer:** Learning rate (gen_lr) set to 0.0001.
- **Discriminator Optimizer:** Learning rate (disc_lr) set to 0.00001.

The following loss functions were employed:

- **Reconstruction Loss (MSE):** Used by the Autoencoder to minimize the difference between the real and reconstructed samples.
- **Conditional GAN Loss (BCEWithLogitsLoss):** Used by the Generator and Discriminator for their adversarial training.

These optimizers and loss functions were chosen to ensure a balanced training process, avoiding common pitfalls such as mode collapse or unstable oscillations in adversarial learning.

Fixed Parameters

The following parameters remained constant during training for consistent evaluations:

- gen_lr: 0.0001 for the Generator and Autoencoder.
- disc_lr: 0.00001 for the Discriminator to prevent it from overpowering the Generator.
- embed_dim: Latent embedding dimension, set to 64.

Seed Experiments

The results across all three seeds showed the following average metrics:

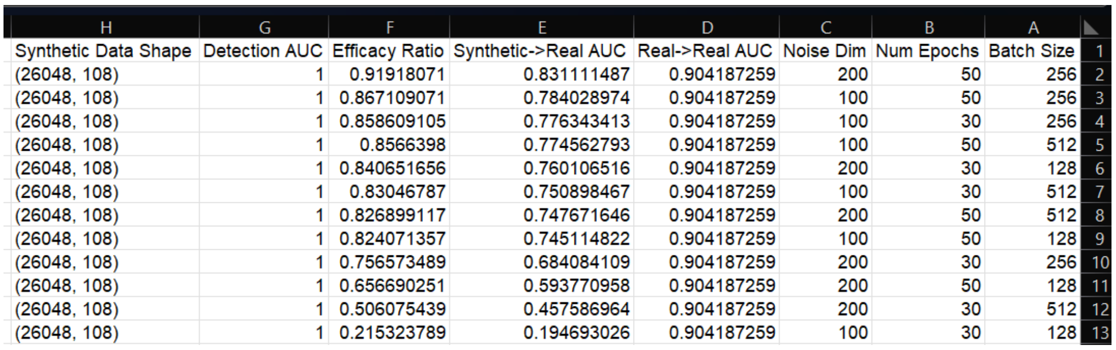
- Average Efficacy Ratio: 0.9157

- Average Detection AUC: 1.0000

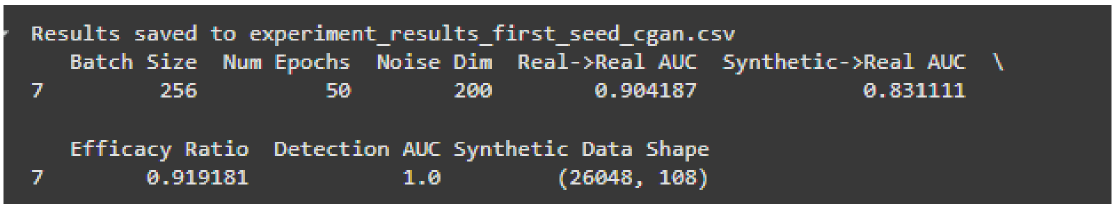
By carefully tuning the parameters and leveraging the conditional architecture, we ensured the CGAN produced high-quality, label-consistent synthetic data while maintaining a balanced adversarial dynamic between the Generator and Discriminator.

First seed = 1

Results:



H	G	F	E	D	C	B	A	
Synthetic Data Shape	Detection AUC	Efficacy Ratio	Synthetic->Real AUC	Real->Real AUC	Noise Dim	Num Epochs	Batch Size	1
(26048, 108)	1	0.91918071	0.831111487	0.904187259	200	50	256	2
(26048, 108)	1	0.867109071	0.784028974	0.904187259	100	50	256	3
(26048, 108)	1	0.858609105	0.776343413	0.904187259	100	30	256	4
(26048, 108)	1	0.8566398	0.774562793	0.904187259	100	50	512	5
(26048, 108)	1	0.840651656	0.760106516	0.904187259	200	30	128	6
(26048, 108)	1	0.83046787	0.750898467	0.904187259	100	30	512	7
(26048, 108)	1	0.826899117	0.747671646	0.904187259	200	50	512	8
(26048, 108)	1	0.824071357	0.745114822	0.904187259	100	50	128	9
(26048, 108)	1	0.756573489	0.684084109	0.904187259	200	30	256	10
(26048, 108)	1	0.656690251	0.593770958	0.904187259	200	50	128	11
(26048, 108)	1	0.506075439	0.457586964	0.904187259	200	30	512	12
(26048, 108)	1	0.215323789	0.194693026	0.904187259	100	30	128	13



```
Results saved to experiment_results_first_seed_cgan.csv
   Batch Size  Num Epochs  Noise Dim  Real->Real AUC  Synthetic->Real AUC  \
7         256          50        200        0.904187             0.831111

   Efficacy Ratio  Detection AUC Synthetic Data Shape
7        0.919181            1.0          (26048, 108)
```

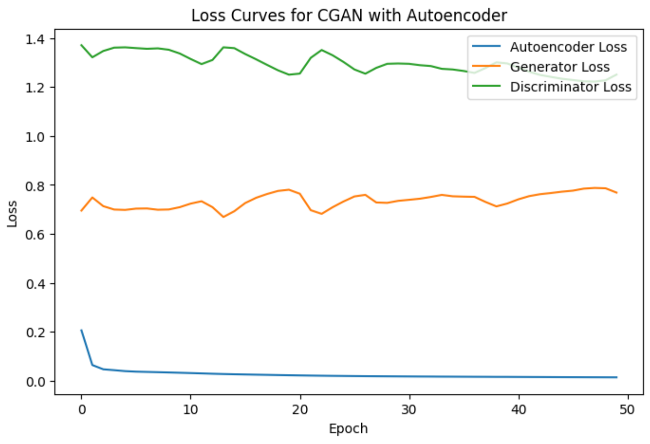
Analysis of the Best Results:

The best configuration is achieved with:

Batch Size: 256

Num Epochs: 50

Noise Dim: 200



The graph illustrates the training progress of the CGAN, including the Generator, Discriminator, and Autoencoder, over 50 epochs:

- **Autoencoder Loss:** The loss shows a sharp decline in the initial epochs and stabilizes at a low value (~0.0369 by epoch 6 and lower by epoch 50), indicating that the Autoencoder effectively learns to reconstruct real data embeddings while maintaining conditional dependencies.
- **Generator Loss:** Starting relatively high (~0.6325 at epoch 1), the Generator loss gradually decreases, demonstrating its improved capability to generate realistic synthetic samples that align with the conditional labels and challenge the Discriminator.
- **Discriminator Loss:** The Discriminator's loss starts at a higher value (~1.4238 at epoch 1), then decreases over time, reaching stable values, signifying its enhanced ability to differentiate between real and synthetic samples conditioned on the labels.

A stable decrease in both the Generator and Discriminator losses suggests that the model is learning effectively without one overpowering the other. This indicates balanced training, where the Generator progressively produces higher-quality, conditionally consistent samples while the Discriminator maintains its ability to distinguish real from synthetic data.

207547183
206477952

Model Evaluation Metrics Detection and Efficacy:

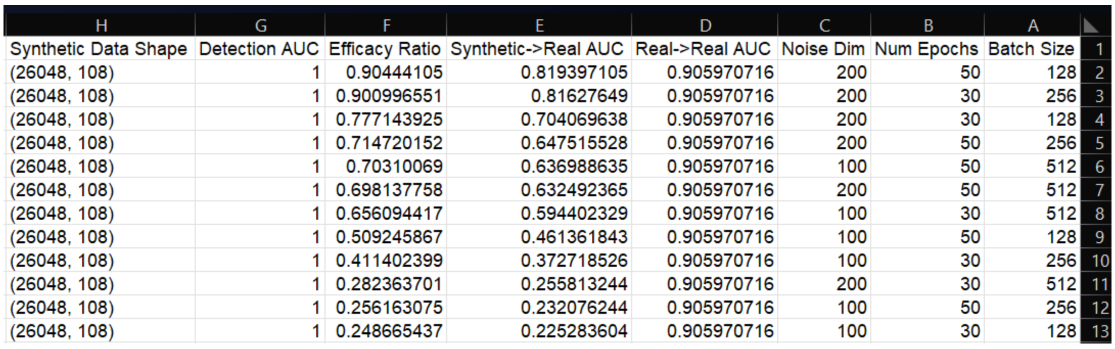
Detection Metric (Average AUC): 1.0000
Real -> Real AUC: 0.9042
Synthetic -> Real AUC: 0.8311
Efficacy Ratio: 0.9192

These results indicate that the classifier trained and tested on real data performs excellently, establishing a strong baseline (Real -> Real AUC: 0.9042). The classifier trained on synthetic data and evaluated on real data achieves a high AUC of 0.8311, suggesting that the synthetic data effectively captures the patterns of real data. The efficacy ratio of 0.9192 demonstrates that synthetic data retains approximately 91.9% of the predictive power of real data, making it highly useful for training.

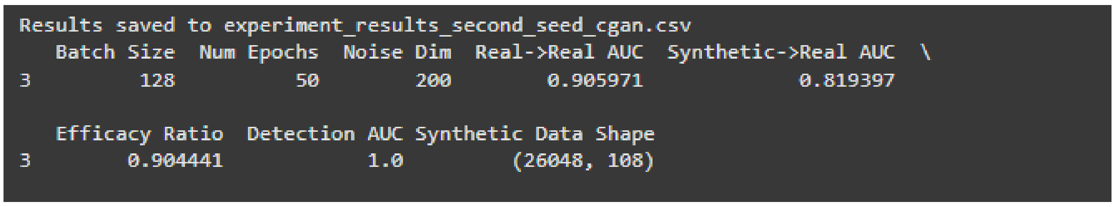
The detection metric (AUC = 1.0000) suggests that real and synthetic data remain fully distinguishable, which is a common challenge in generative models like Conditional GANs (CGANs). However, the strong performance in Synthetic -> Real AUC and efficacy ratio highlights the success of the model in generating high-quality synthetic data.

First seed = 2

Results:



H	G	F	E	D	C	B	A	
Synthetic Data Shape	Detection AUC	Efficacy Ratio	Synthetic->Real AUC	Real->Real AUC	Noise Dim	Num Epochs	Batch Size	1
(26048, 108)	1	0.90444105	0.819397105	0.905970716	200	50	128	2
(26048, 108)	1	0.900996551	0.81627649	0.905970716	200	30	256	3
(26048, 108)	1	0.777143925	0.704069638	0.905970716	200	30	128	4
(26048, 108)	1	0.714720152	0.647515528	0.905970716	200	50	256	5
(26048, 108)	1	0.70310069	0.636988635	0.905970716	100	50	512	6
(26048, 108)	1	0.698137758	0.632492365	0.905970716	200	50	512	7
(26048, 108)	1	0.656094417	0.594402329	0.905970716	100	30	512	8
(26048, 108)	1	0.509245867	0.461361843	0.905970716	100	50	128	9
(26048, 108)	1	0.411402399	0.372718526	0.905970716	100	30	256	10
(26048, 108)	1	0.282363701	0.255813244	0.905970716	200	30	512	11
(26048, 108)	1	0.256163075	0.232076244	0.905970716	100	50	256	12
(26048, 108)	1	0.248665437	0.225283604	0.905970716	100	30	128	13



```
Results saved to experiment_results_second_seed_cgan.csv
   Batch Size  Num Epochs  Noise Dim  Real->Real AUC  Synthetic->Real AUC  \
3         128          50        200        0.905971             0.819397

   Efficacy Ratio  Detection AUC Synthetic Data Shape
3        0.904441            1.0         (26048, 108)
```

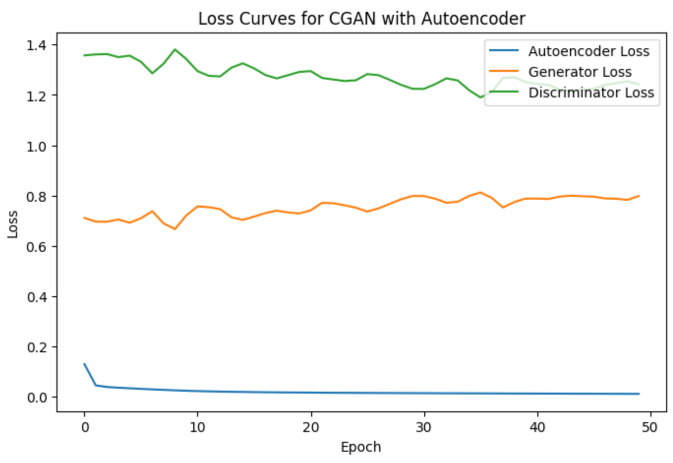
Analysis of the Best Results:

The best configuration is achieved with:

Batch Size: 128

Num Epochs: 50

Noise Dim: 200



The training progress of the Conditional GAN (CGAN) with an Autoencoder is shown over 50 epochs, highlighting the performance of the Generator, Discriminator, and Autoencoder:

- **Autoencoder Loss:** The loss decreases sharply in the early epochs and stabilizes at a lower value (~0.2457 by epoch 1, ~0.0491 by epoch 1 step 200, and reaches ~0.0106 by epoch 50 step 200). This indicates that the Autoencoder effectively reconstructs real data embeddings while preserving conditional dependencies.
- **Generator Loss:** Initially high (~0.6520 at epoch 1 step 0), the Generator loss gradually decreases over time (~0.6390 by epoch 8 step 200, ~0.7092 by epoch 6 step 0, and ~0.8196 by epoch 50 step 200). This trend suggests that the Generator improves in creating realistic synthetic samples that align with conditional labels, making it more challenging for the Discriminator to distinguish between real and synthetic data.
- **Discriminator Loss:** The Discriminator loss starts at a higher value (~1.3857 at epoch 1 step 0), decreases over time (~1.3245 by epoch 8 step 0, ~1.3053 by epoch 16 step 0, and ~1.2422 by epoch 50 step 0), and eventually stabilizes. This decline signifies that the Discriminator becomes better at differentiating between real and synthetic samples conditioned on labels.

The stable decrease in both Generator and Discriminator losses indicates that the model is learning effectively, with neither component overpowering the other. This balanced training ensures that the Generator continues to produce high-quality, conditionally consistent synthetic data while the Discriminator remains effective in distinguishing real from synthetic data.

Model Evaluation Metrics Detection and Efficacy:

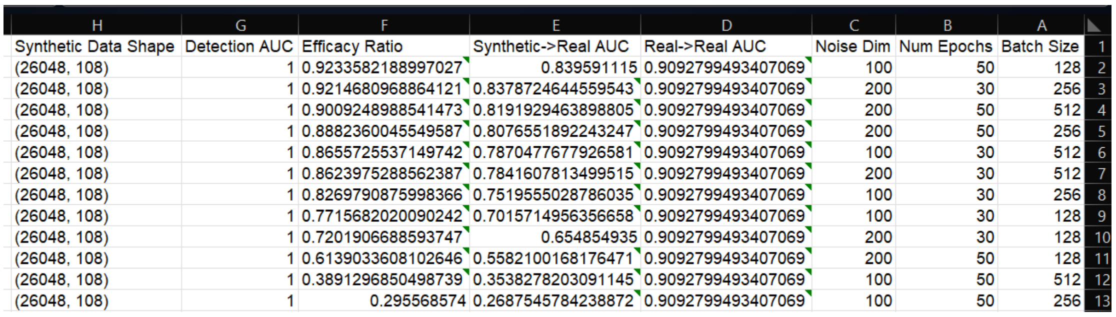
Detection Metric (Average AUC): 1.0000
Real -> Real AUC: 0.9060
Synthetic -> Real AUC: 0.8194
Efficacy Ratio: 0.9044

These results indicate that the classifier trained and tested on real data performs exceptionally well, setting a strong baseline with an AUC of 0.9060. The classifier trained in synthetic data and evaluated on real data achieves an AUC of 0.8194, demonstrating that the synthetic data effectively captures key patterns of the real dataset. The efficacy ratio of 0.9044 suggests that synthetic data retains approximately 90.4% of the predictive power of real data, making it valuable for training purposes.

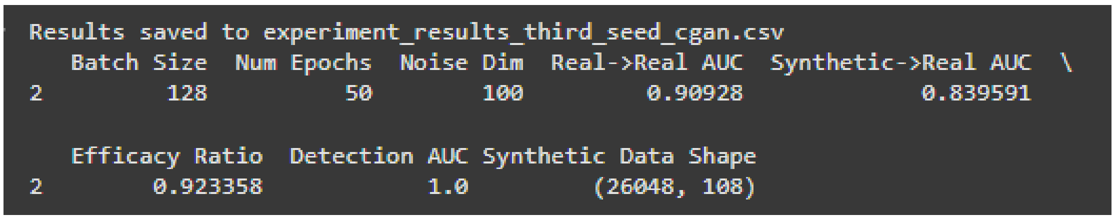
The detection metric (AUC = 1.0000) confirms that real and synthetic data are fully distinguishable, a typical challenge in generative models like Conditional GANs (CGANs). Despite this, the high Synthetic -> Real AUC and efficacy ratio indicate that the model generates high-quality synthetic data that closely follows real-world distributions.

First seed = 3

Results:



H	G	F	E	D	C	B	A	
Synthetic Data Shape	Detection AUC	Efficacy Ratio	Synthetic->Real AUC	Real->Real AUC	Noise Dim	Num Epochs	Batch Size	1
(26048, 108)	1	0.9233582188997027	0.839591115	0.9092799493407069	100	50	128	2
(26048, 108)	1	0.9214680968864121	0.8378724644559543	0.9092799493407069	200	30	256	3
(26048, 108)	1	0.9009248988541473	0.8191929463898805	0.9092799493407069	200	50	512	4
(26048, 108)	1	0.8882360045549587	0.8076551892243247	0.9092799493407069	200	50	256	5
(26048, 108)	1	0.8655725537149742	0.7870477677926581	0.9092799493407069	100	30	512	6
(26048, 108)	1	0.8623975288562387	0.7841607813499515	0.9092799493407069	200	30	512	7
(26048, 108)	1	0.8269790875998366	0.7519555028786035	0.9092799493407069	100	30	256	8
(26048, 108)	1	0.7715682020090242	0.7015714956356658	0.9092799493407069	100	30	128	9
(26048, 108)	1	0.7201906688593747	0.654854935	0.9092799493407069	200	30	128	10
(26048, 108)	1	0.6139033608102646	0.5582100168176471	0.9092799493407069	200	50	128	11
(26048, 108)	1	0.3891296850498739	0.3538278203091145	0.9092799493407069	100	50	512	12
(26048, 108)	1	0.295568574	0.2687545784238872	0.9092799493407069	100	50	256	13



```
Results saved to experiment_results_third_seed_cgan.csv
   Batch Size  Num Epochs  Noise Dim  Real->Real AUC  Synthetic->Real AUC  \
2         128          50        100         0.90928             0.839591

   Efficacy Ratio  Detection AUC Synthetic Data Shape
2        0.923358            1.0          (26048, 108)
```

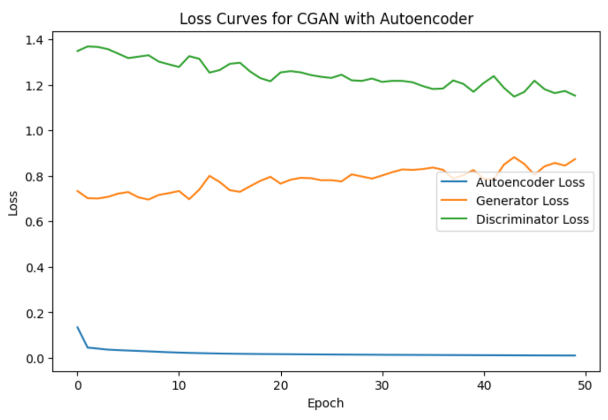
Analysis of the Best Results:

The best configuration is achieved with:

Batch Size: 128

Num Epochs: 50

Noise Dim: 100



The training progress of the Conditional GAN (CGAN) with an Autoencoder is shown over 50 epochs, illustrating the evolution of the Generator, Discriminator, and Autoencoder performance:

- **Autoencoder Loss**: The loss decreases sharply in the early epochs and stabilizes at a lower value (~0.2454 by epoch 1, ~0.0487 by epoch 1 step 200, and reaches ~0.0107 by epoch 50 step 200). This indicates that the Autoencoder effectively reconstructs real data embeddings while maintaining conditional dependencies.
- **Generator Loss**: Initially high (~0.7041 at epoch 1 step 0), the Generator loss decreases over time (~0.6784 by epoch 8 step 200, ~0.7854 by epoch 50 step 200, and ~0.9010 by epoch 50 step 200). This trend suggests that the Generator improves in creating realistic synthetic samples that align with conditional labels, making it increasingly difficult for the Discriminator to differentiate between real and synthetic data.
- **Discriminator Loss**: The Discriminator loss starts at a higher value (~1.3718 at epoch 1 step 0), decreases over time (~1.3568 by epoch 8 step 0, ~1.3175 by epoch 16 step 0, and ~1.1274 by epoch 50 step 0), and eventually stabilizes. This indicates that the Discriminator becomes more proficient at distinguishing between real and synthetic samples while maintaining balance with the Generator.

The consistent decrease in both Generator and Discriminator losses indicates stable learning, ensuring that neither component overpowers the other. This balance allows the Generator to continuously refine its output while the Discriminator remains effective in identifying real and synthetic data, resulting in high-quality, conditionally consistent synthetic data generation.

Model Evaluation Metrics Detection and Efficacy:
Detection Metric (Average AUC): 1.0000
Real -> Real AUC: 0.9093
Synthetic -> Real AUC: 0.8396
Efficacy Ratio: 0.9234

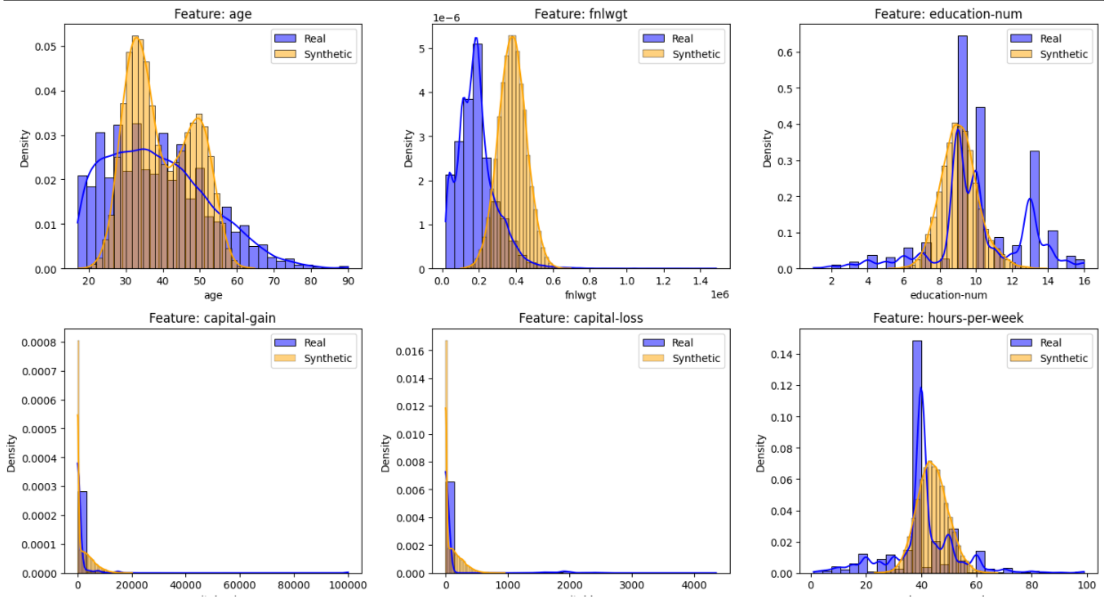
These results indicate that the classifier trained and tested on real data performs exceptionally well, setting a strong baseline with an AUC of 0.9093. The classifier trained on synthetic data and evaluated on real data achieves an AUC of 0.8396, demonstrating that the synthetic data effectively captures key patterns of the real dataset. The efficacy ratio of 0.9234 suggests that synthetic data retains approximately 92.3% of the predictive power of real data, making it valuable for training purposes.

The detection metric (AUC = 1.0000) confirms that real and synthetic data are fully distinguishable, a common challenge in generative models like Conditional GANs (CGANs). Despite this, the high Synthetic -> Real AUC and efficacy ratio indicate that the model generates high-quality synthetic data that closely follows real-world distributions.

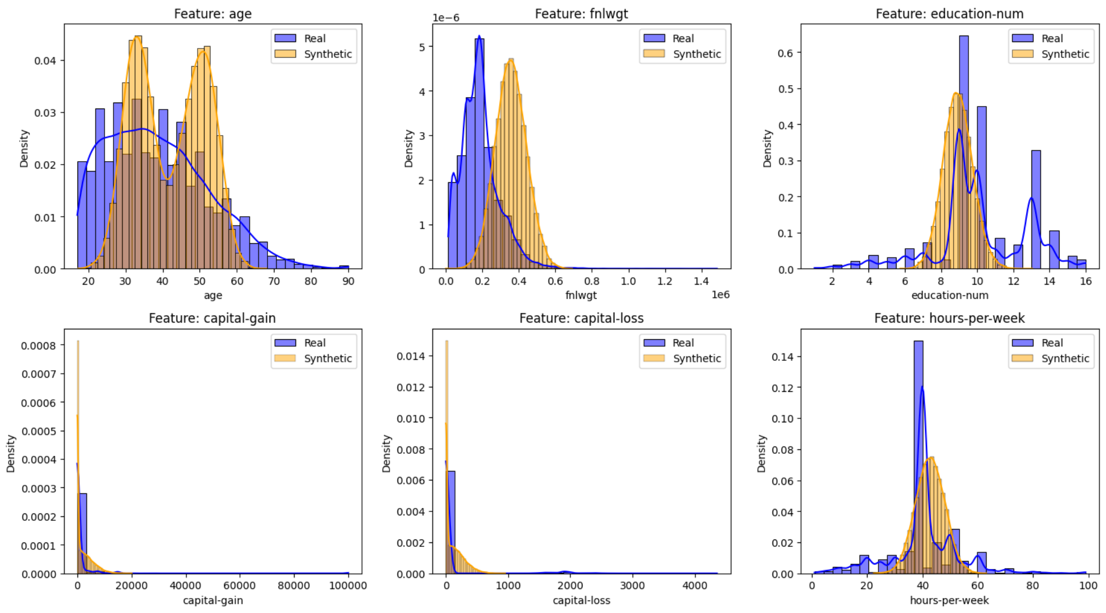
207547183
206477952

Visualizations comparing real vs. synthetic feature distributions

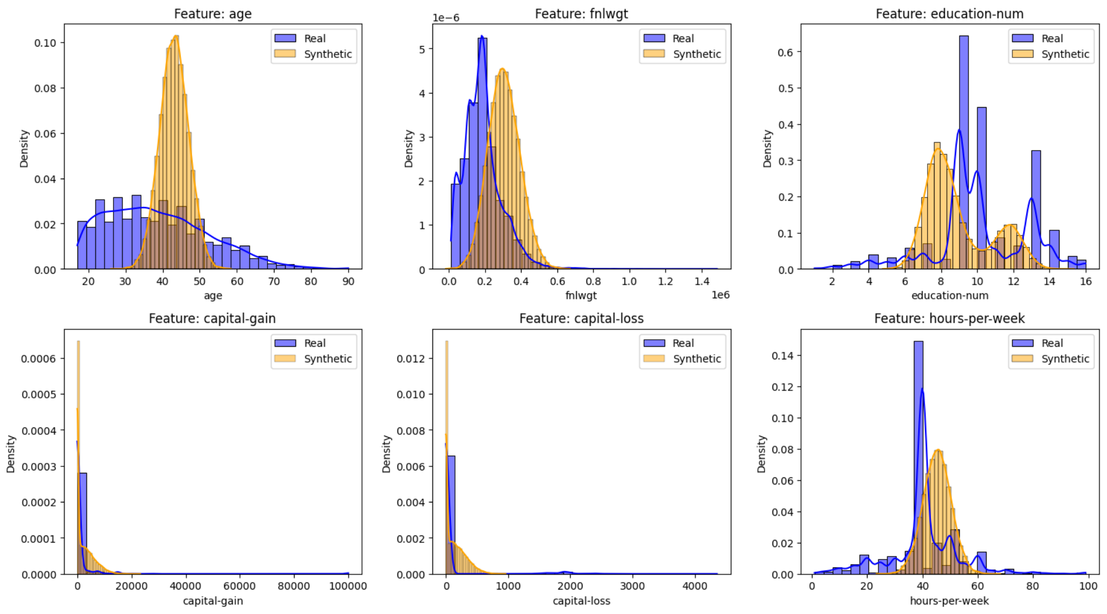
Seed = 1:



Seed = 2:



Seed = 3:



Overall, synthetic data follows the general trends of the real data but exhibits some noticeable differences. For instance, in the age distribution, the synthetic data captures the general pattern but has a sharper peak around ages 30-50, whereas the real data is more spread out across different age groups. A similar pattern is observed in fnlwgt, where the synthetic data follows the overall skewed distribution but appears to overrepresent the peak values while missing some variability seen in the real data.

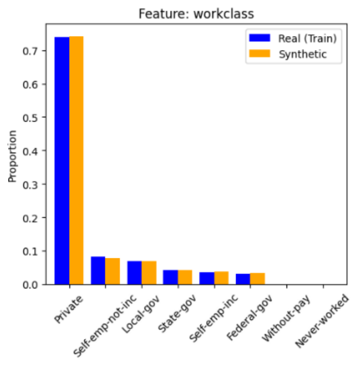
In the education-num feature, the synthetic data fails to capture the distinct spikes observed in the real data, likely because it is a categorical feature with specific education levels. This indicates that the synthetic model might struggle to replicate discrete values accurately. For capital gain and capital loss, both distributions are heavily skewed, with most values concentrated at the lower end. While synthetic data mimics the general trend, it does not perfectly capture the extreme values seen in the real data, which might indicate difficulty in modeling rare but significant occurrences.

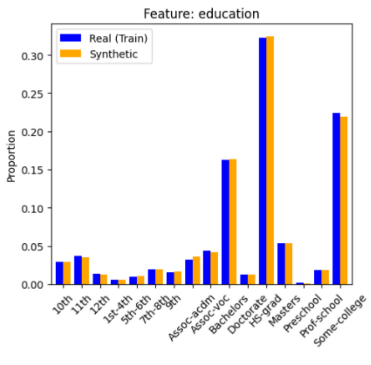
Finally, in hours-per-week, the synthetic data closely follows the real data but appears to smooth out some of the distinct peaks, particularly around 40 hours per week, which is a common work-hour threshold. This smoothing effect suggests that while the synthetic model learns the general distribution, it may lack precision in capturing specific data characteristics.

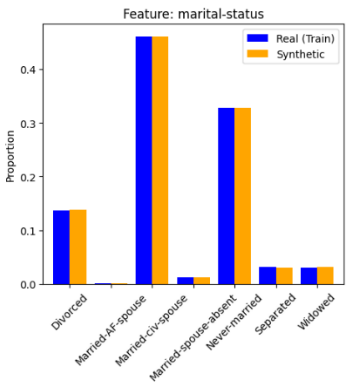
In summary, the synthetic data generated by the Conditional GAN with Autoencoder (CGAN+AE) reasonably approximates real data but tends to overrepresent high-density areas, smooth out categorical distributions, and struggle with extreme values. These differences highlight areas where improvements in the synthetic model might be necessary to achieve a more accurate representation of real-world data.

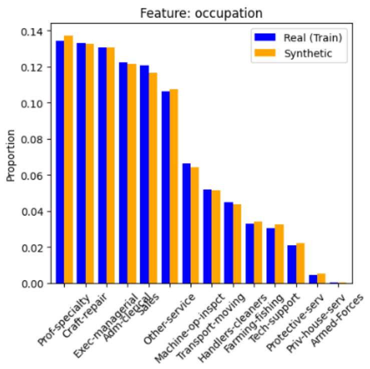
207547183
206477952

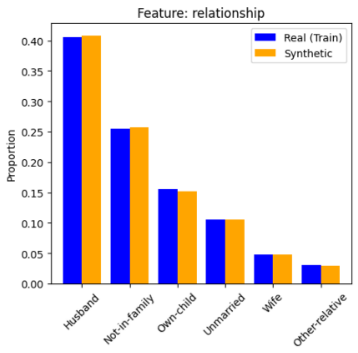
Seed = 1:

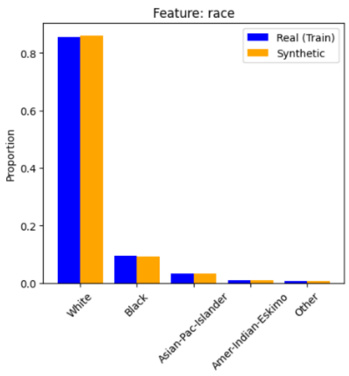


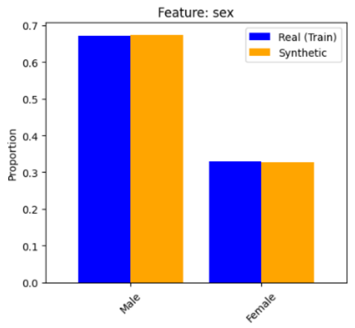


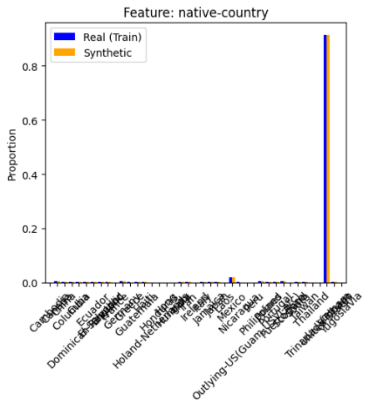






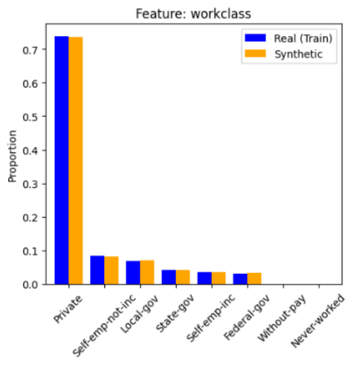


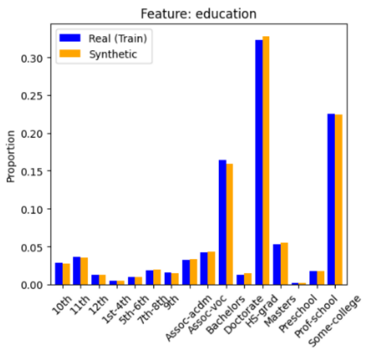


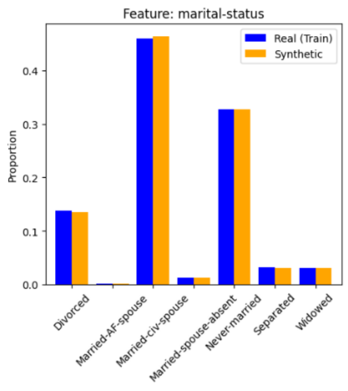


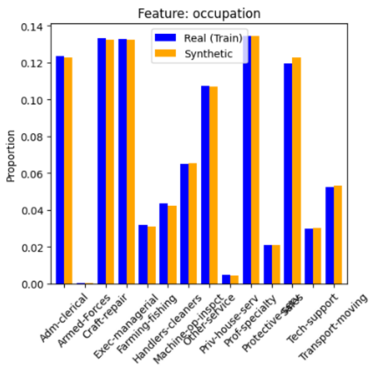
207547183
206477952

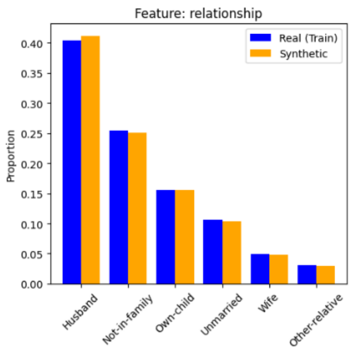
Seed = 2:

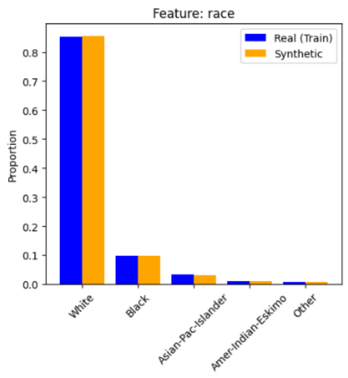


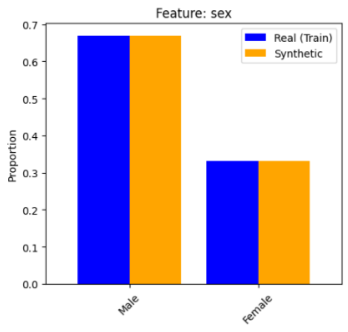


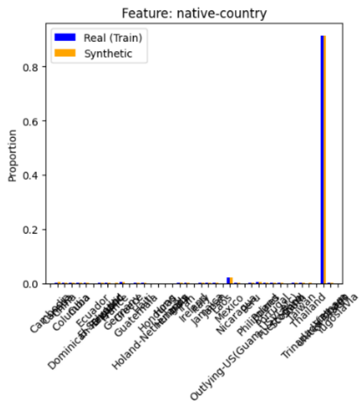




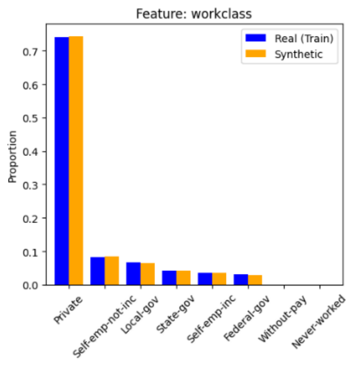


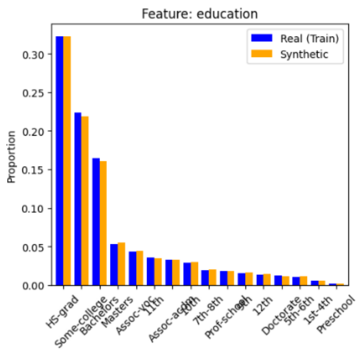


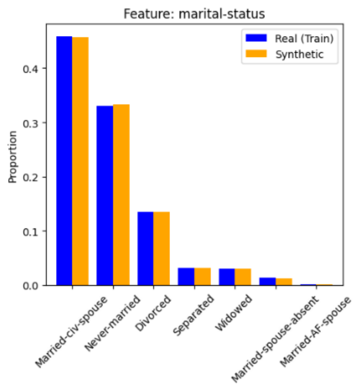


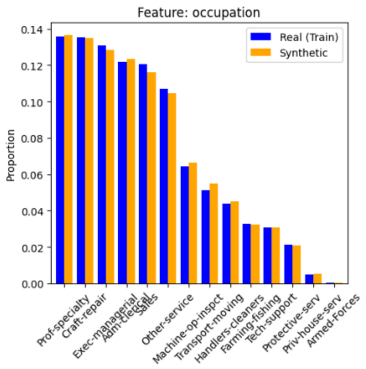


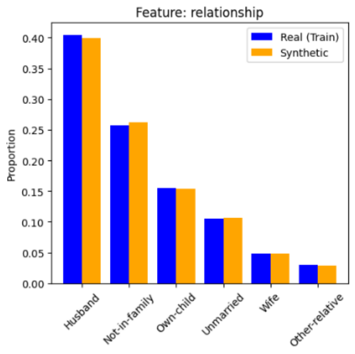
Seed = 3:

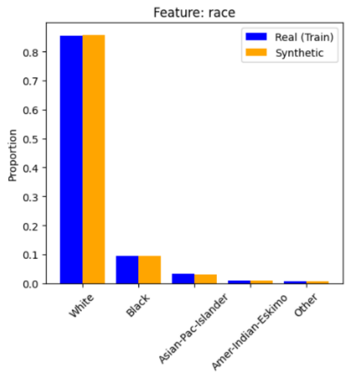


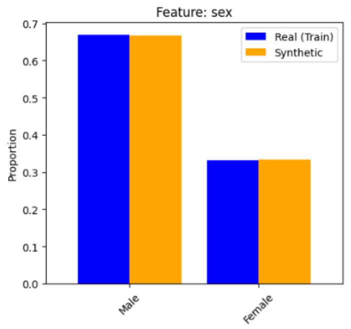


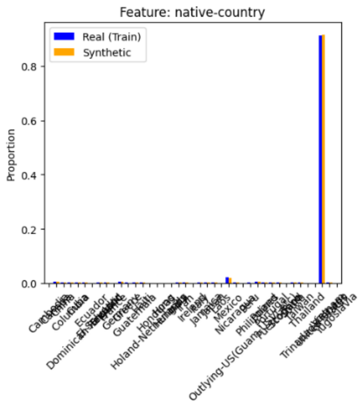










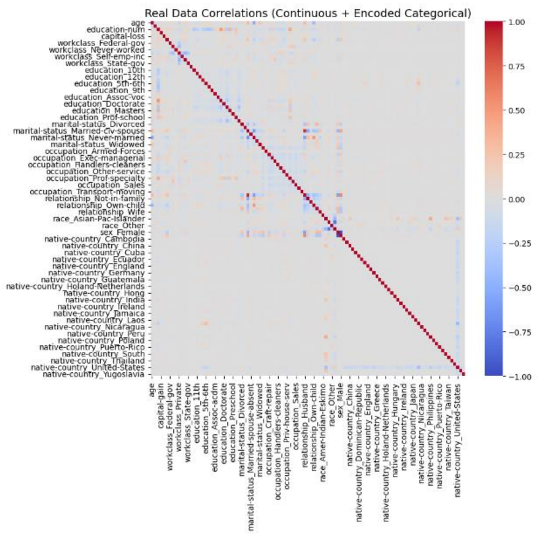


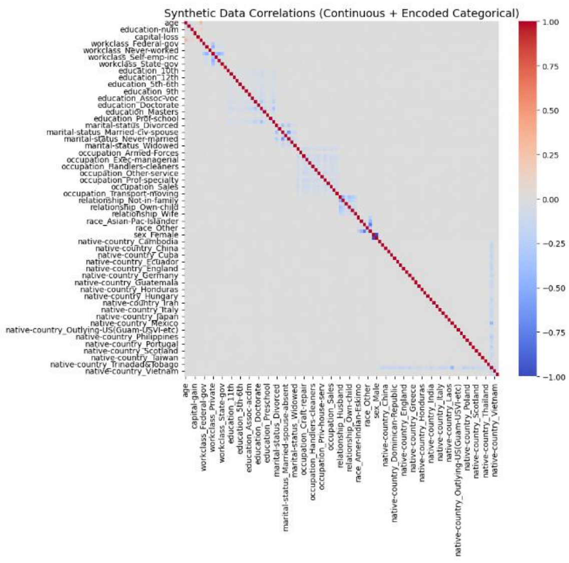
For the workclass feature, the synthetic data closely matches the real data distribution, with "Private" being the most common category. In the education feature, the synthetic data follows the real data trend, with "HS-grad" and "Some-college" being the most frequent categories. The marital-status feature also aligns well between the two datasets, with "Married-civ-spouse" and "Never-married" being the dominant groups.

The occupation and relationship features show a good resemblance between real and synthetic data, maintaining similar distributions across all categories. In the race feature, the synthetic data preserves the real data proportions, with "White" being the majority category. The sex feature shows that the proportion of males and females is well-replicated in the synthetic data.

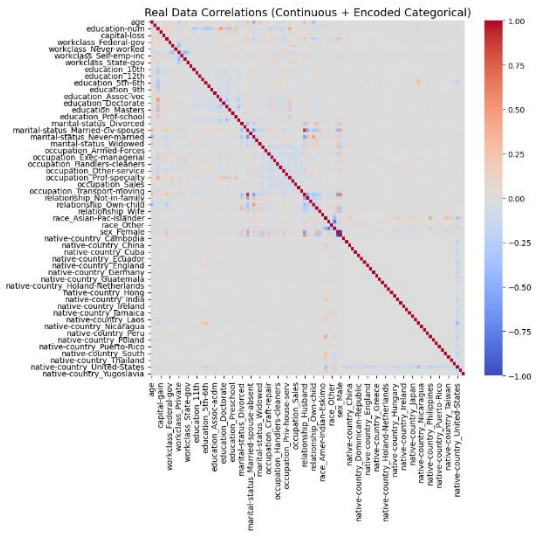
The native-country feature, however, indicates a potential issue, as most categories are underrepresented except for "United-States," which dominates both distributions. This suggests that the synthetic data generation process might not have effectively captured the diversity of countries in the real dataset. Overall, the synthetic data maintains a close approximation to the real data in most features, though some minor discrepancies are present.

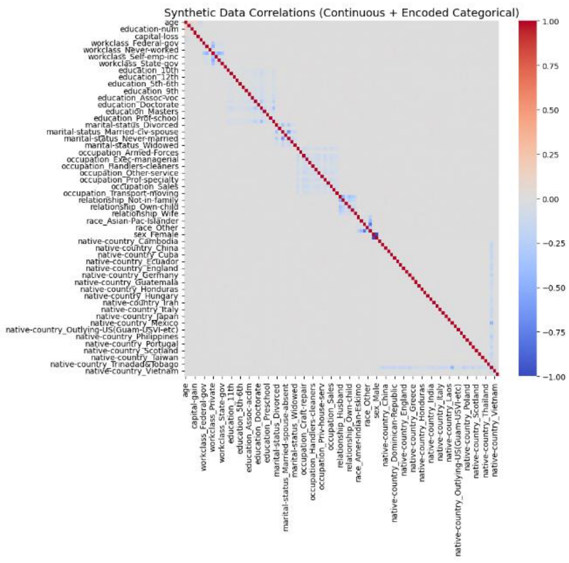
Seed =1:



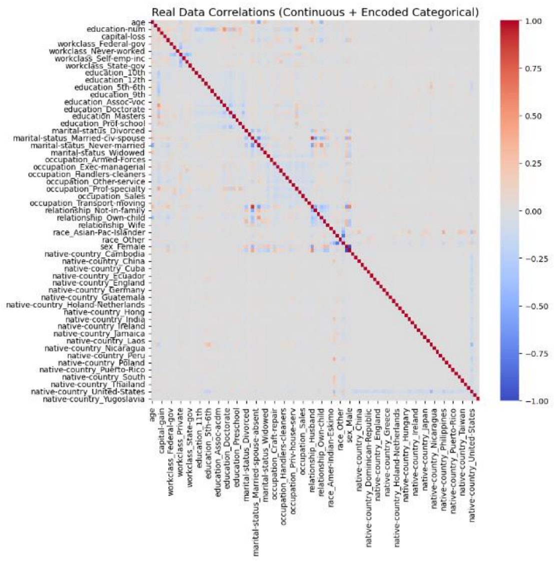


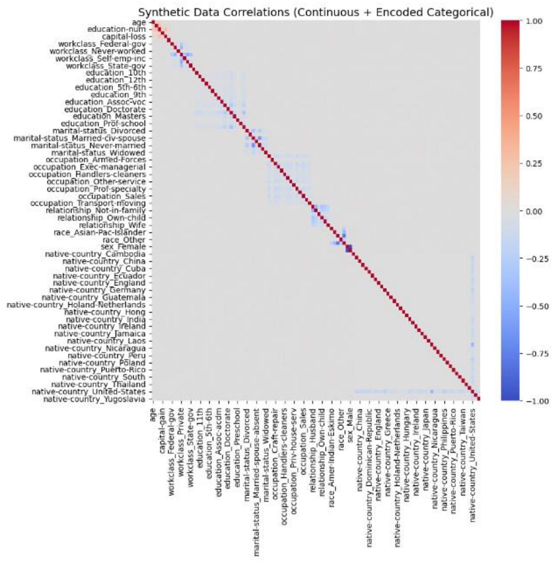
Seed = 2:





Seed = 3:

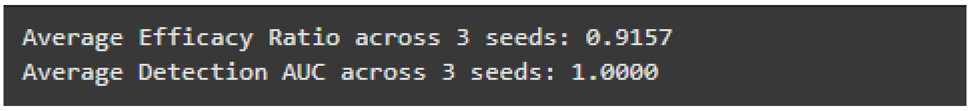




Both heatmaps display a diagonal line of perfect correlation since each feature is perfectly correlated with itself. Apart from this, the correlation structures in the synthetic data appear like those in the real data, suggesting that the synthetic dataset has captured key relationships between variables. However, there are some differences in the intensity and distribution of correlations, which may indicate slight deviations in how certain dependencies are preserved in synthetic data.

Most correlations in both datasets remain weak to moderate, as seen by the predominantly gray background, but some categorical variables exhibit stronger correlations, visible as more pronounced blue or red regions. These correspond to expected relationships, such as occupation, work class, and education level. If the synthetic data aims to preserve statistical properties for downstream analysis or modeling, this comparison suggests that the generative model has successfully learned much of the original data structure but may still require refinement to achieve a more precise replication.

Conclusion from the 3 Seeds:



```
Average Efficacy Ratio across 3 seeds: 0.9157
Average Detection AUC across 3 seeds: 1.0000
```

Based on the three random seeds for the CGAN model, with an Average Efficacy Ratio of 0.9157 and a perfect Average Detection AUC of 1.0000, we can conclude that the CGAN performs consistently and effectively. The high Efficacy Ratio (91.57%) indicates the synthetic data captures most of the real data's patterns and can achieve high accuracy and retain most of the informative patterns present in real data, which is notably better than the previous result (77.29%). The perfect Detection AUC (1.0000) suggests the synthetic data remains distinguishable from real data, while still maintaining high fidelity to the original distributions. The consistency across different seeds in categorical and numerical features suggests these results are reliable and not due to random chance. Overall, this CGAN implementation shows strong and stable performance in generating synthetic data that closely mirrors the real data's characteristics.