Ariel Siman Tov – 209449821, Tal Klein - 209234103

Data generation with GANs (DL EX 4)

Introduction

The goal of this assignment is to train and compare a Generative Adversarial Network (GAN) and a Conditional GAN (cGAN) on the "Adult" dataset, then evaluate the similarity between real and synthetic samples as well as the usefulness of the synthetic data as a substitute for real data.

We followed these stages in our work:

1. Load and preprocess the "Adult" dataset (in ARFF format).
2. Perform an 80%/20% train-test split, preserving label ratios, repeated with three different random seeds.
3. Train both a standard GAN and a conditional GAN (cGAN).
4. Visualize and evaluate the performance using two key metrics:
   - **Detection** (how easily a classifier can distinguish real vs. synthetic data),
   - **Efficacy** (how well a model trained on synthetic data performs on real data).

Dataset and Preprocessing

Dataset: The Adult dataset (also known as the Census Income dataset) contains demographic information such as age, workclass, education, marital status, occupation, race, gender, hours-per-week, native country, etc., along with a binary label "income" (>50K or ≤50K).

Steps:

1. **Train-Test Split**: The data was split into 80% for training and 20% for testing, ensuring we maintained the class distribution. This step was repeated for three different random seeds {12,21,27}.
2. **Handling Categorical Variables**: Used one-hot encoding (OneHotEncoder) on all categorical features. (We expanded our data representation from 14 features to 108, excluding the target column.)
3. **Handling Numerical Variables**: Scaled continuous features using MinMaxScaler with a range of $[-1,+1]$.
4. **Label Encoding**: Converted the binary label into 0 for "≤50K" and 1 for ">50K".
5. **Final Shape**: After preprocessing, the dataset was represented as a NumPy array X_processed for y_encoded for labels.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

Models and Architectures

We implemented two main architectures: a standard GAN and a conditional GAN (cGAN). For both:

- **Generator**: Takes random noise (and optionally conditional labels in cGAN) as input and outputs synthetic samples of the same dimensionality as the real data.
- **Discriminator**: Attempts to distinguish between real and synthetic samples (and also uses labels in cGAN). Below is a high-level overview of each architecture:

3.1 Standard GAN

- **Generator**:
  - Input: Random noise vector of size nz.
  - Architecture: Several linear (fully connected) layers with LeakyReLU activations, ending with a tanh layer to produce output in $[-1, +1]$.
  - Output: Synthetic sample x' in the same dimension as the real data.
- **Discriminator**:
  - Input: A sample (real or generated).
  - Architecture: Multiple linear layers with LeakyReLU activations and Dropout for regularization, ending with a sigmoid output for binary classification (real vs. fake).
  - Output: Probability that the input is a real sample.

3.2 Conditional GAN (cGAN)

- **Generator**:
  - Input: Random noise vector (nz) **concatenated** with a label embedding (y) specifying the target income class. The label embedding dimension equals the number of classes (which is 2 for "Adult" data).
  - Architecture: Similar to the standard GAN generator but with additional embedding and concatenation of class information.
  - Output: Synthetic sample x' (conditioned on the desired label).
- **Discriminator**:
  - Input: A sample (real or generated) **plus** its corresponding label embedding.
  - Architecture: Similar to the standard GAN discriminator but expanded to accept concatenated inputs of features + label embedding.
  - Output: Probability that the input sample is real, given the label.

3.3 GAN/cGAN with AutoEncoder - We trained a separate autoencoder architecture, then use the encoder component to create an embedding of the real sample before it is sent to the discriminator. We tried this improvement for both models.

- **Encoder:**
  - Input: Raw input data of dimension data_dim.
  - Architecture: A sequential network with six linear layers, each followed by batch normalization, ReLU activation, and dropout. The final layer outputs a compressed latent representation of size latent_dim without activation or normalization.
  - Output: A latent representation of the input data.
- **Decoder:**
  - Input: Latent representation (latent_dim dimension).
  - Architecture: A mirrored structure of the encoder, using linear layers with batch normalization, ReLU activation, and dropout. The final output layer applies a Tanh activation function, assuming the input data is scaled to [-1, 1].
  - Output: Reconstructed data with the same dimension as the original input.
- **AE training:**
  - Loss Function: Smooth L1 loss to minimize reconstruction error.
  - Optimizer: Adam optimizer with a learning rate of 1e-3.
  - Training Process: The model iteratively minimizes reconstruction error using a batch-wise update, logging loss at each step.

Usage of the Autoencoder (AE) in GAN Training:

- **Optional Encoding Step:** The code includes a section where the real data (data_row) could be encoded using an autoencoder before training the GAN.

```
# Encode real data to get real embeddings
with torch.no_grad():
  data_row = autoencoder.encode(data_row)
```

- **Purpose of the Autoencoder:** If enabled, the autoencoder would transform real data into a lower-dimensional latent space, making the GAN operate on learned feature representations instead of raw data. With AE, the GAN generator might learn to produce meaningful embeddings that are later decoded into the original space, whereas without AE, it directly synthesizes data from random noise in the original feature space.

Using the autoencoder may improve stability and reduce the complexity of modeling high-dimensional data, potentially leading to better quality synthetic samples. Later in this work, we will analyze the impact of adding AE to our model architecture.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

Training Setup

4.1 Hyperparameters

- **Batch Size**: 128
- **Noise Dimension (nz)**: 128
- **Learning Rate**: 0.0001
- **Weight Decay**: 0.00005
- **Optimizer**: Adam (both Generator and Discriminator)
- **Loss Function**: Binary Cross-Entropy (BCELoss)
- **Number of Epochs**: both Standard GAN and cGAN: 50 epochs

4.2 Training Procedure

1. **GAN**:

   Discriminator Update:

   - Sample a batch of real data from the training set.
   - Generate a batch of fake data (via the Generator).
   - Compute discriminator loss for real vs. fake samples, backpropagate, and update the discriminator weights.

   Generator Update:

   - Generate fake data.
   - Pass it through the discriminator and compute the generator loss, which seeks to fool the discriminator into classifying fake data as real.
   - Backpropagate and update generator weights.

2. **cGAN**:
   - The same structure as above, but the generator and discriminator both receive label information as additional input.

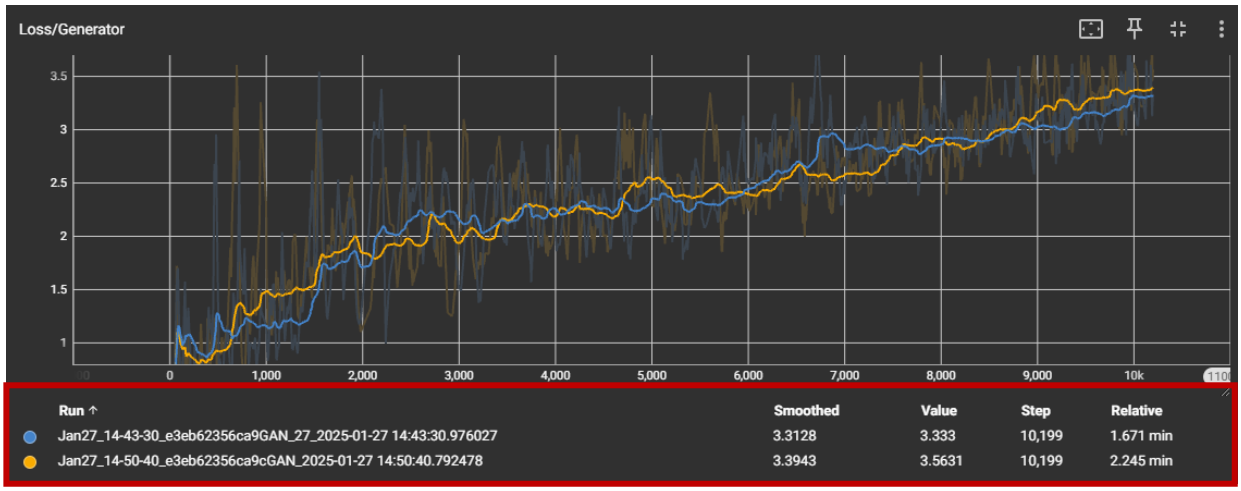
4.3 Stabilization Techniques

- **LeakyReLU**: Used in generator and discriminator to avoid dying ReLU problems.
- **Dropout**: Added discriminator layers to mitigate overfitting and improve training stability.
- **Label Embeddings**: For the cGAN, use nn.Embedding to embed the label into a dense vector space.

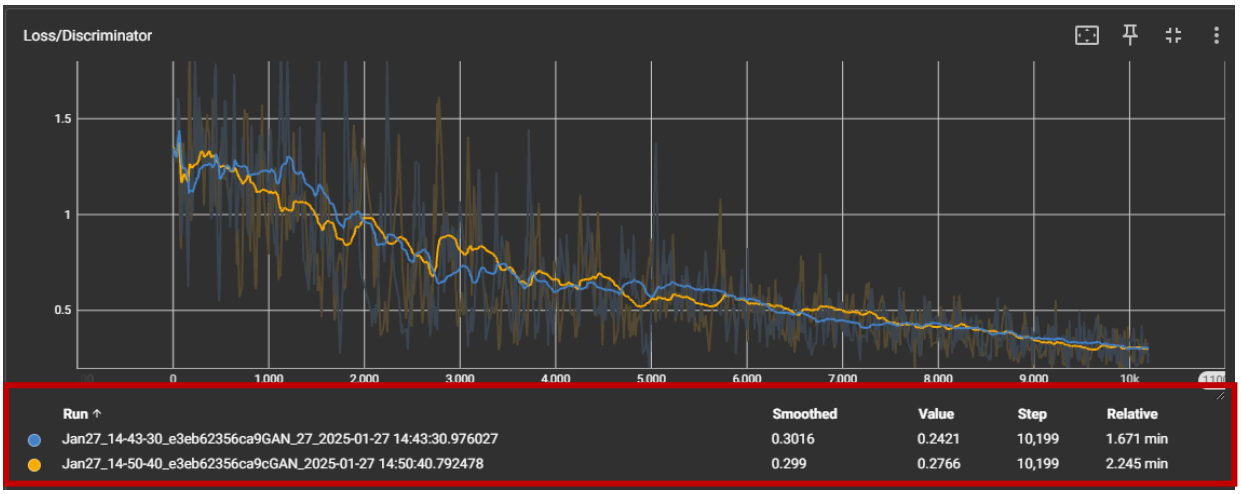
Ariel Siman Tov – 209449821, Tal Klein - 209234103

A Comparison between the loss values of GAN vs cGAN:

Generator loss



Discriminator loss

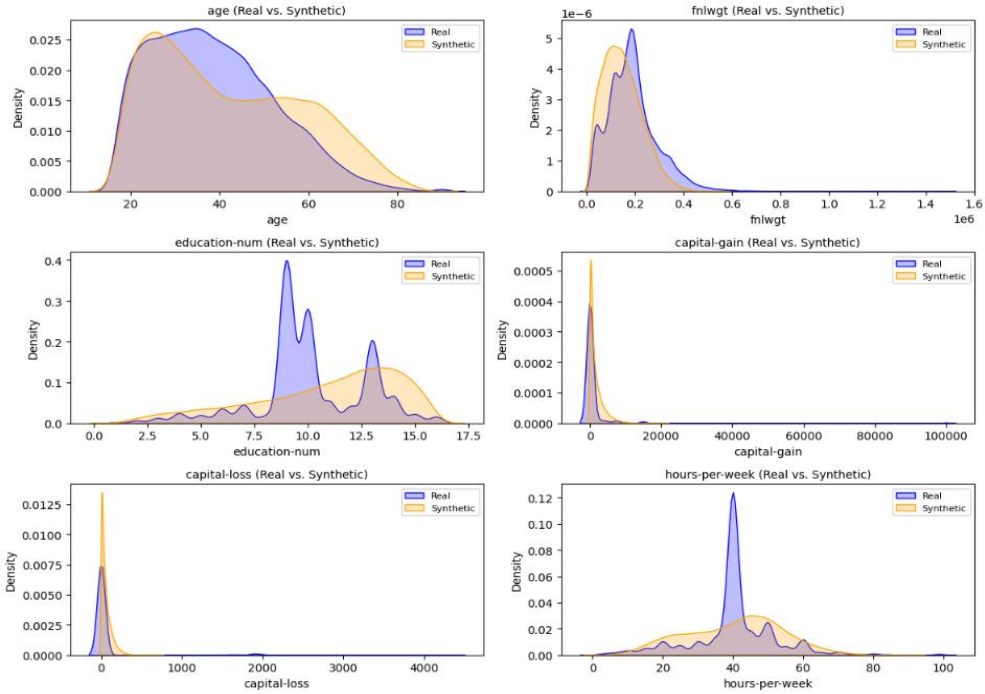


We can see that in both models the generator's loss increases gradually, while the discriminator's loss decreases, indicating that the discriminator becomes better at distinguishing real from fake samples as training progresses. The increasing generator loss suggests it struggles to produce convincing samples as the discriminator improves. The smoothed trends indicate some stability over time, with neither loss collapsing, suggesting that the GAN avoids major training instabilities.

The GAN model exhibits slightly improved loss values compared to the cGAN model.

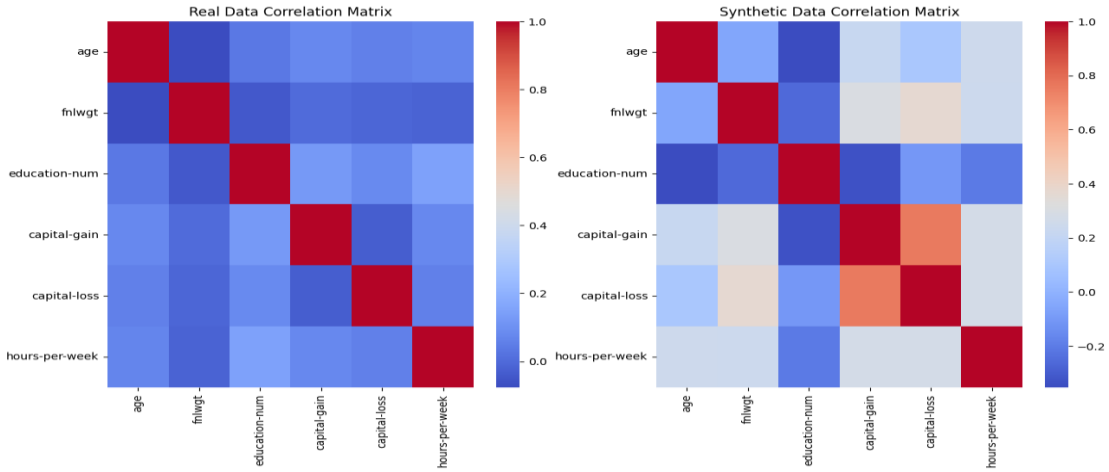
Feature analysis cGAN

Comparing the feature distributions of the synthetic data vs the original data – numeric features



These plots reveal differences between real and synthetic data distributions across several features. For features like age, fnlwgt, capital-gain and capital-loss, the synthetic data captures the overall trends but deviates in certain peaks and variances, indicating an incomplete replication of the real data's nuances. For education-num and hours-per-week, the synthetic data shows less accuracy in replicating the distribution of the original data, with noticeable deviations in key peaks and overall patterns.

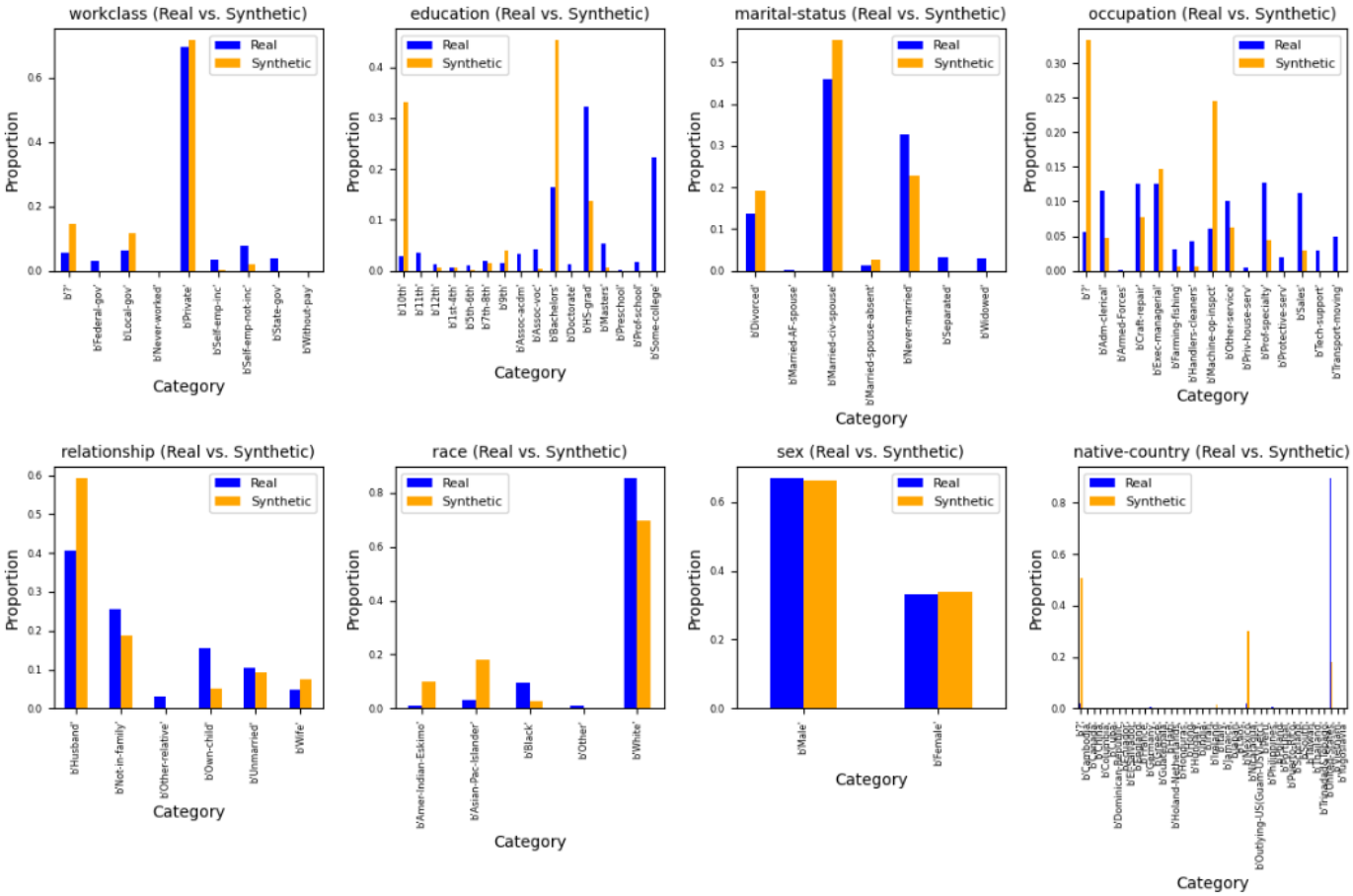
Comparing correlation matrices



The correlation matrices for real and synthetic datasets reveal significant differences in how relationships between numerical variables are captured. In the real data, weak correlations are evident between variables such as capital-gain and capital-loss, indicating that these variables are not linked or influenced by similar

factors. In contrast, the synthetic data struggles to replicate these relationships accurately. While some correlations are preserved, such as between age and hours-per-week, other relationships, like those involving capital-gain and capital-loss, are represented wrongly.

Comparing the feature distributions of the synthetic data vs the original data – categorial features



For attributes like workclass, the synthetic data tends to overrepresent common categories such as "Private" or "local-gov", while underrepresenting less frequent ones like "Self-employed." This discrepancy may be due to the synthetic model's difficulty in balancing rare and frequent classes.

Similarly, in education, the synthetic dataset fails to accurately replicate rare categories such as "Doctorate" or some-college", indicating that the model struggles to handle infrequent data points, likely due to insufficient examples during training.

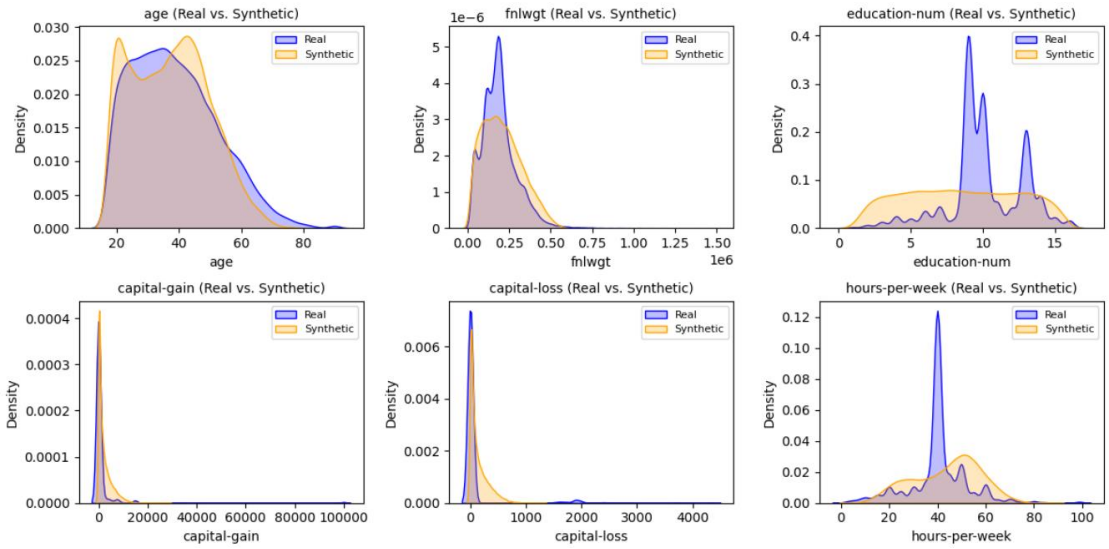
For attribute like sex the synthetic data aligns relatively well with the real data, likely because of its binary nature, which is simpler to replicate.

The native-country attribute reveals the greatest divergence, with the synthetic data poorly representing many countries and overemphasizing the majority. This is likely due to the high imbalance and the large number of unique values, which make accurate replication challenging.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

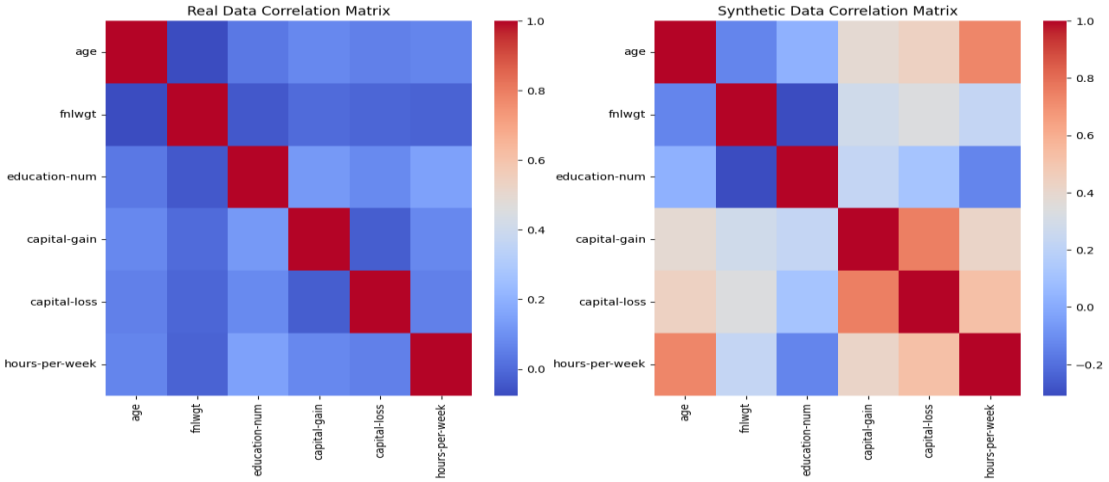
Feature analysis GAN

Comparing the feature distributions of the synthetic data vs the original data – numeric features



The plots show differences between real and synthetic data distributions. Both GAN and cGAN perform similarly on features like age, capital-gain, and capital-loss, with the GAN performing slightly better overall. However, both struggle with features like education-num and hours-per-week, and the GAN shows lower accuracy than the cGAN in replicating the fnlwgt distribution.

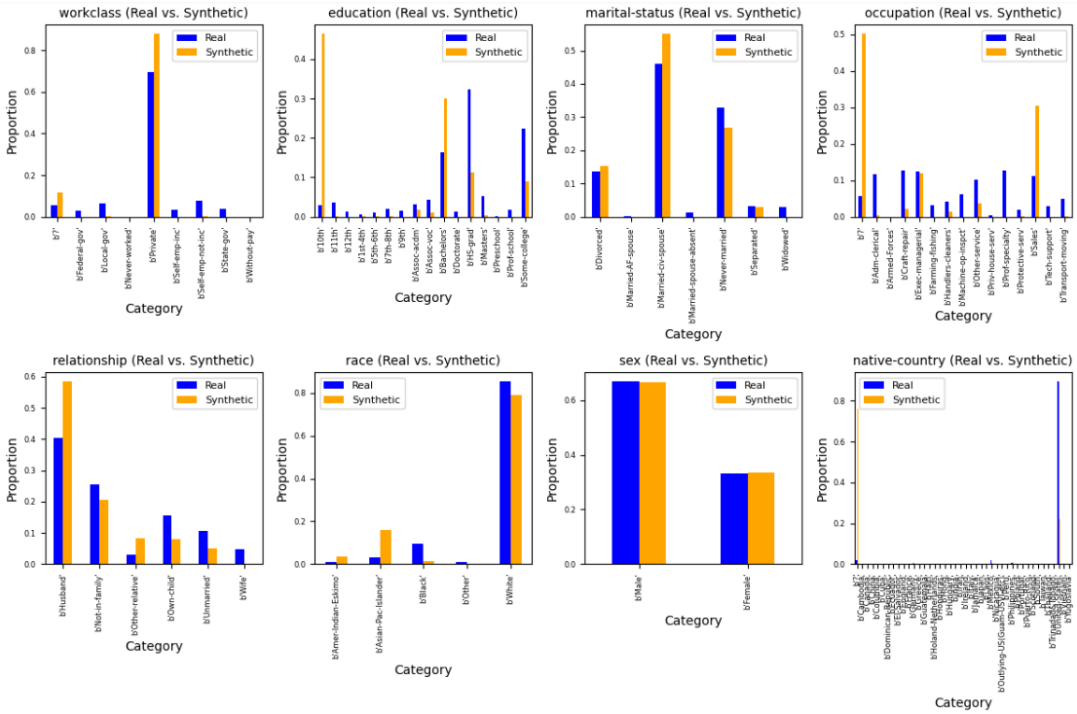
Comparing correlation matrices



The correlation matrices for real and synthetic datasets of the GAN model reveal even greater differences in capturing relationships between numerical variables compared to the cGAN.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

Comparing the feature distributions of the synthetic data vs the original data – categorial features

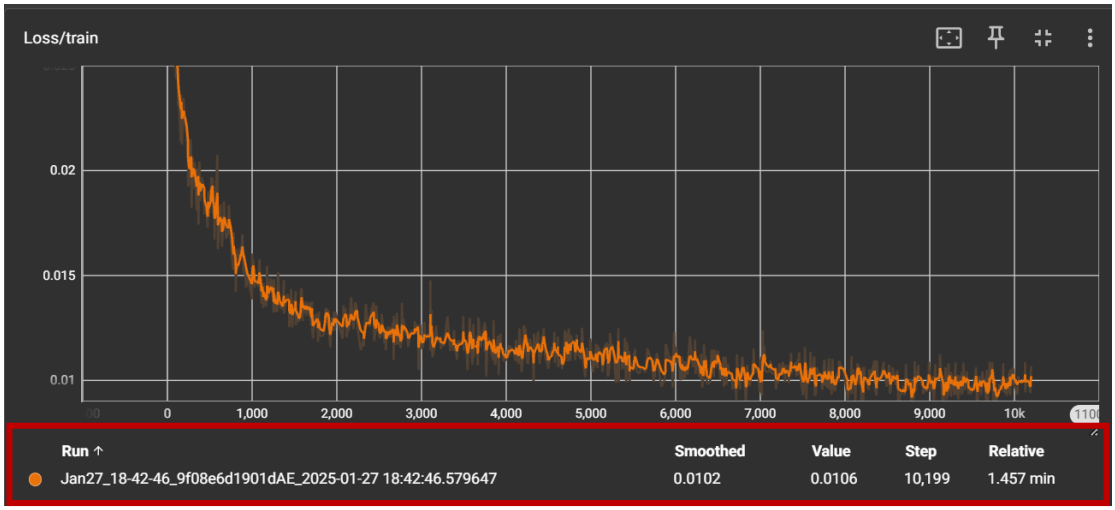


For categorical values, the GAN model appears to capture the original data slightly better; however, the overall results are comparable to the cGAN and exhibit similar trends.

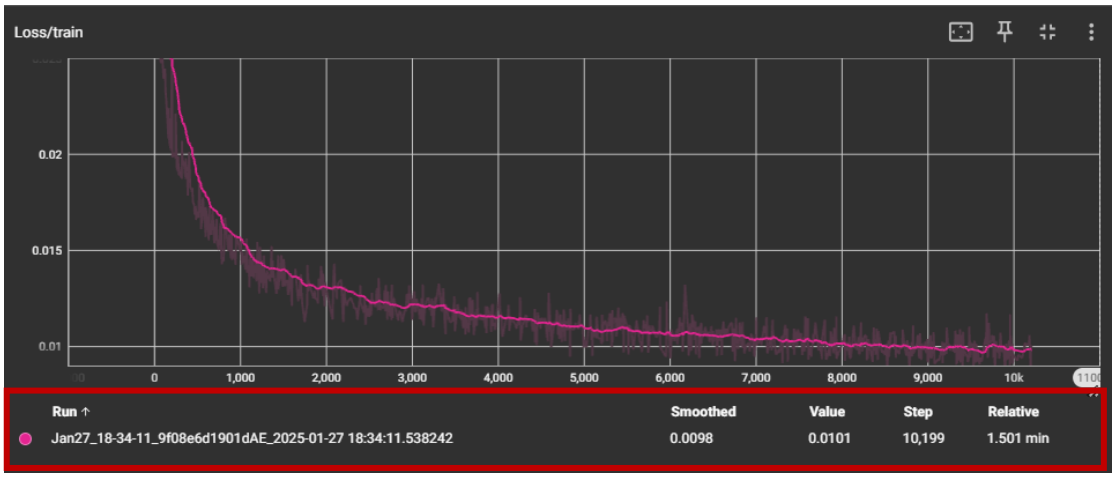
Ariel Siman Tov – 209449821, Tal Klein - 209234103

We have now conducted an experiment in which we incorporated an Autoencoder (AE) into the models' architecture:

Loss on train-set of cGAN with AE



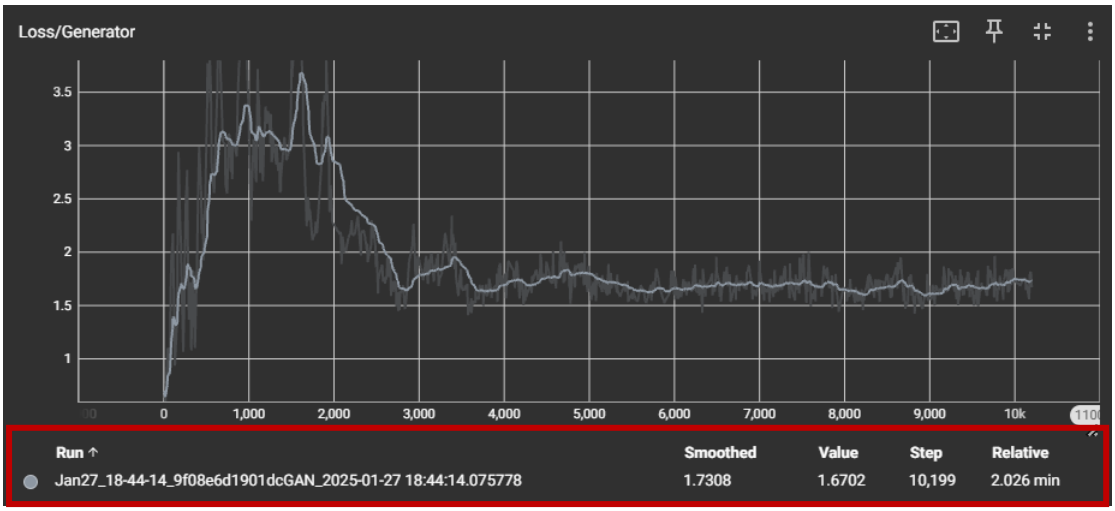
Loss on train-set of GAN with AE



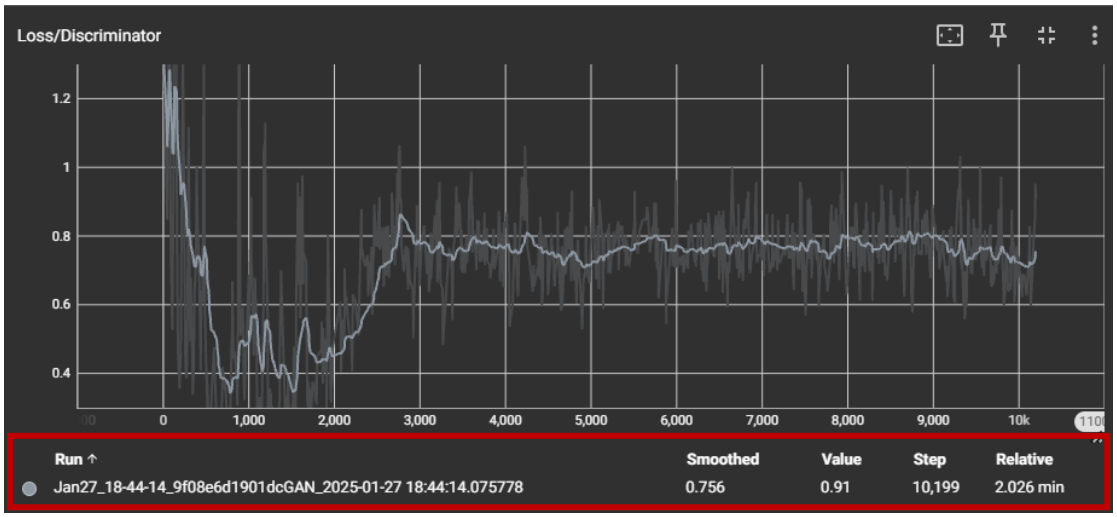
Ariel Siman Tov – 209449821, Tal Klein - 209234103

loss values of cGAN with AE

Generator loss



Discriminator loss

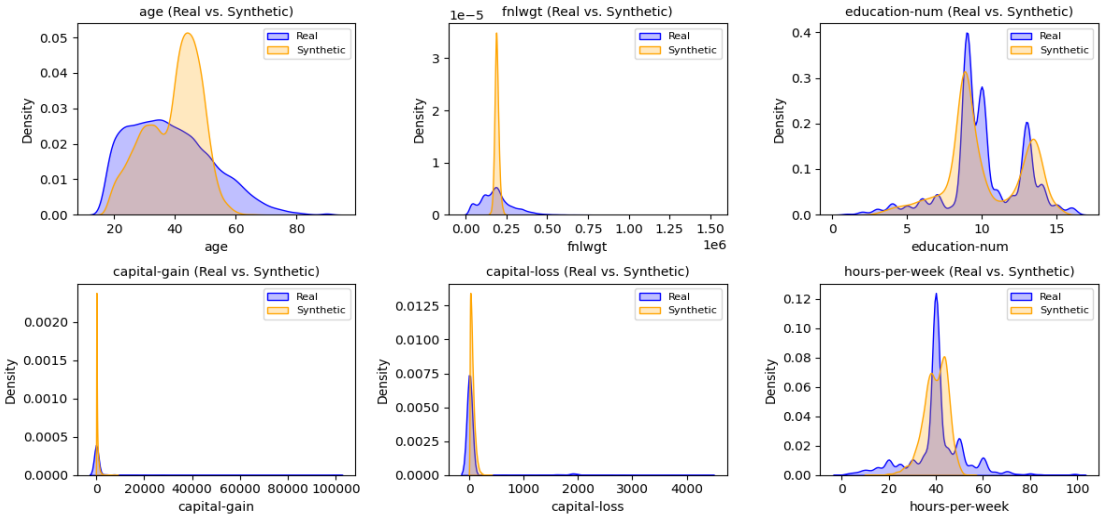


A notable change occurs around step 2,000-3,000, where both losses stabilize after initial volatility. The Generator's loss starts high (around 3.0-3.5) before settling at approximately 1.67, while the Discriminator's loss initially fluctuates between 0.4-1.2 before stabilizing around 0.9. The convergence of both losses to relatively stable values suggest the GAN reached a reasonable equilibrium in its training, though the continuing small oscillations indicate the ongoing adversarial nature of the training process.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

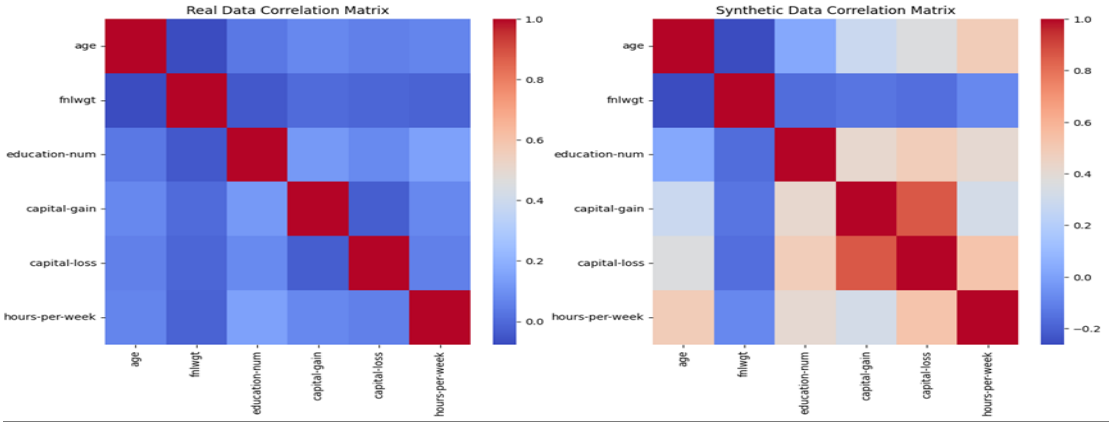
Feature analysis cGAN with AE

Comparing the feature distributions of the synthetic data vs the original data – numeric features



It seems that adding AE to the model architecture has some impact on the feature distributions of the synthetic data vs the original data. For age feature, the synthetic distribution roughly captures the overall shape but appears more concentrated around the middle ages. The financial-related features (fnlwgt, capital-gain, and capital-loss) show sharp peaks in both distributions, though the synthetic data doesn't perfectly match the real data's long tails. The education-num (likely years of education) shows discrete peaks that are reasonably well-matched between real and synthetic data, particularly around the 10–12-year mark which is likely to correspond to high school completion. The hours-per-week distribution shows a strong peak of around 40 hours (typical full-time work) in both datasets, though the synthetic data appears to smooth out some of the finer details present in the real data.

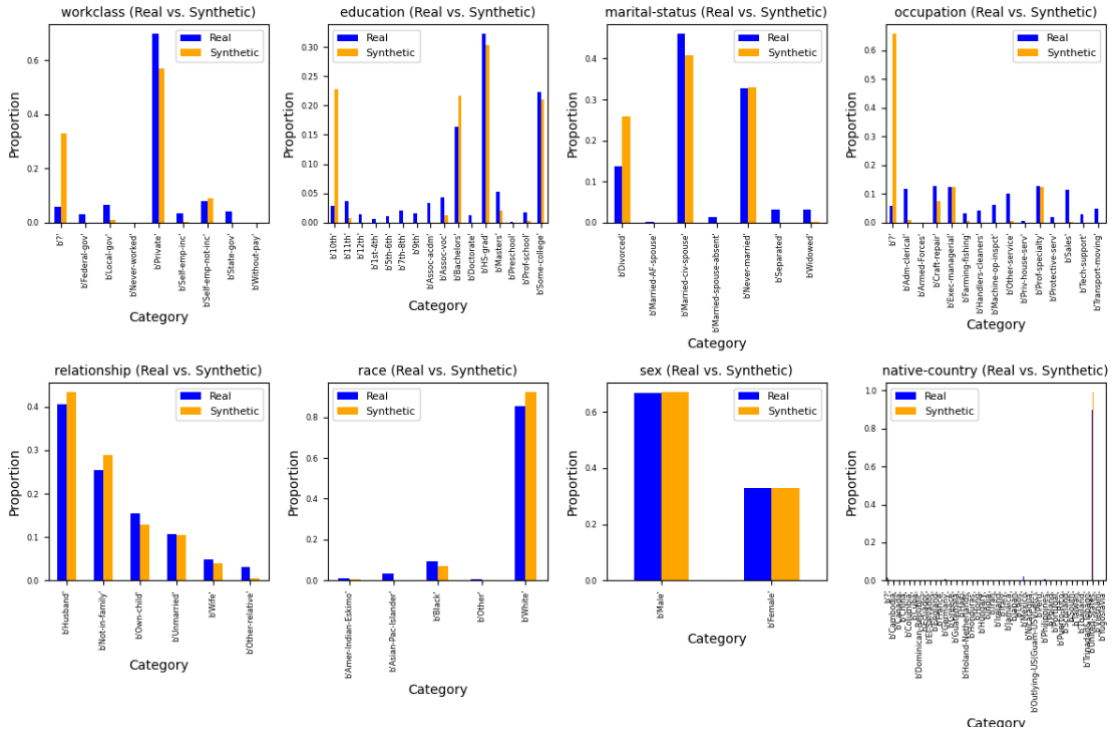
Comparing correlation matrices



Compared to the cGAN model without AE, in the cGAN with AE the synthetic data exhibits stronger correlations between variables, widening the gap between the correlation structures of the real and synthetic datasets.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

Comparing the feature distributions of the synthetic data vs the original data – categorial features

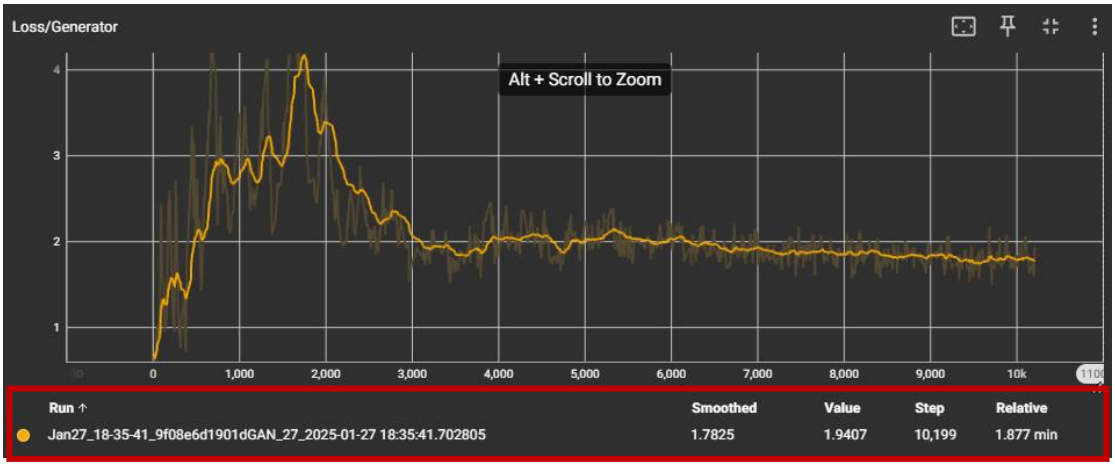


The synthetic data (orange) generally captures the major patterns and relative proportions of the real data (blue) across various demographic categories like workclass, education, marital status, and occupation. The matching is particularly good for simpler binary or few-category variables like sex (which shows almost perfect alignment) and marital status. However, for categories with many possible values (like occupation and native-country), there are some discrepancies in the proportions, though the overall patterns are preserved. The race category shows good alignment with the predominant categories while maintaining appropriate proportions for minority groups. This suggests the GAN has learned the broad demographic patterns well, though it may struggle somewhat with more granular or less frequent categories in the data.

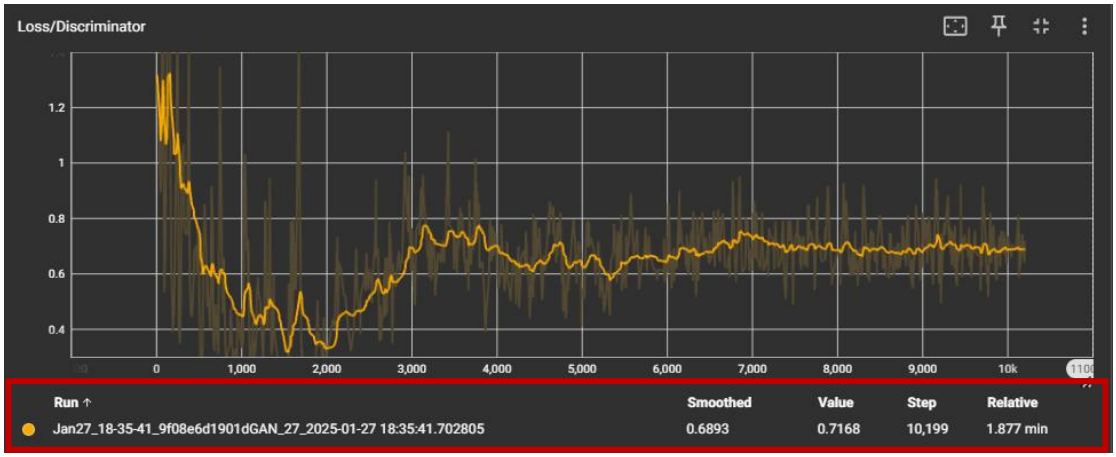
These results show a slight improvement compared to the cGAN without AE model, indicating better representation of categorical distributions.

loss values of GAN with AE

Generator loss



Discriminator loss



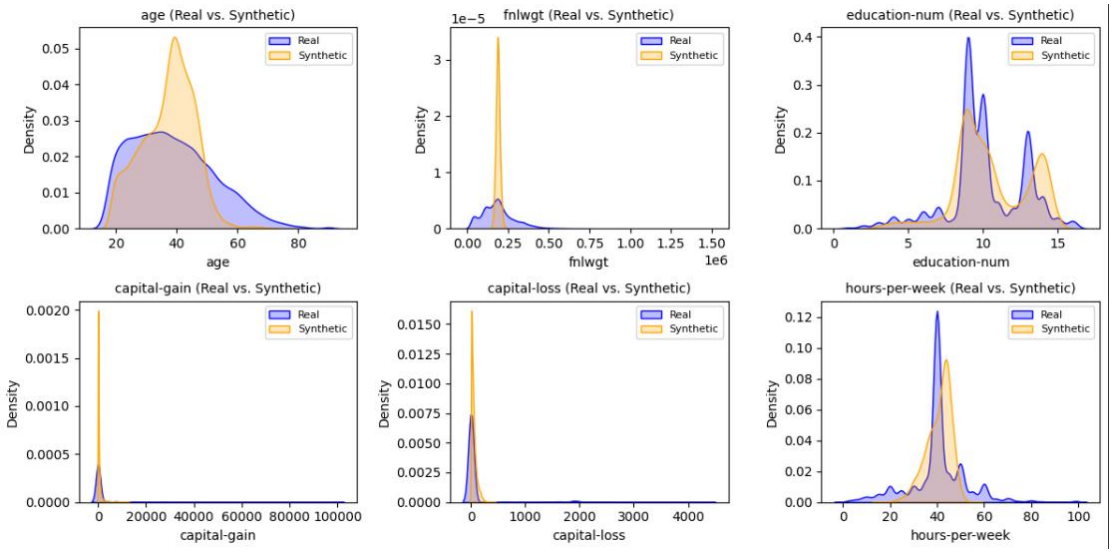
In this case the key transition also occurs around step 2,000-3,000, where both losses stabilize. The Generator's loss peaks at around 4.0 before settling to approximately 1.9, while the Discriminator's loss starts with high volatility (peaking around 1.2-1.4) before stabilizing around 0.7.

Compared to the previous plots, the overall pattern and convergence behavior is very similar, suggesting no fundamental change in the training dynamics. The differences are minor and within normal variation for GAN training. Both runs appear to have reached a stable equilibrium, and the similar patterns suggest consistent, reproducible training behavior, although the **cGAN with AE** model achieved the best results in terms of loss function performance.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

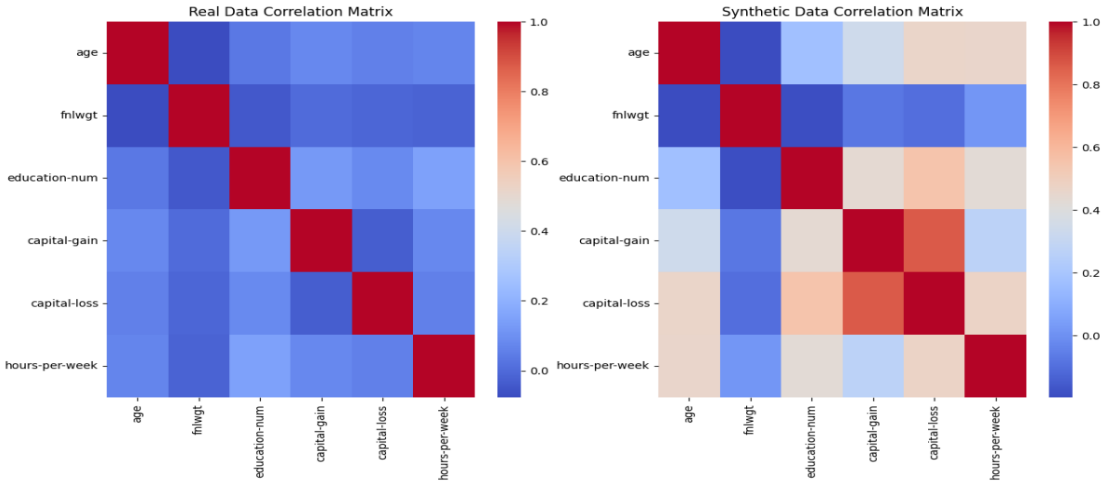
Feature analysis GAN with AE

Comparing the feature distributions of the synthetic data vs the original data – numeric features



Similar results to the model cGAN with AE model.

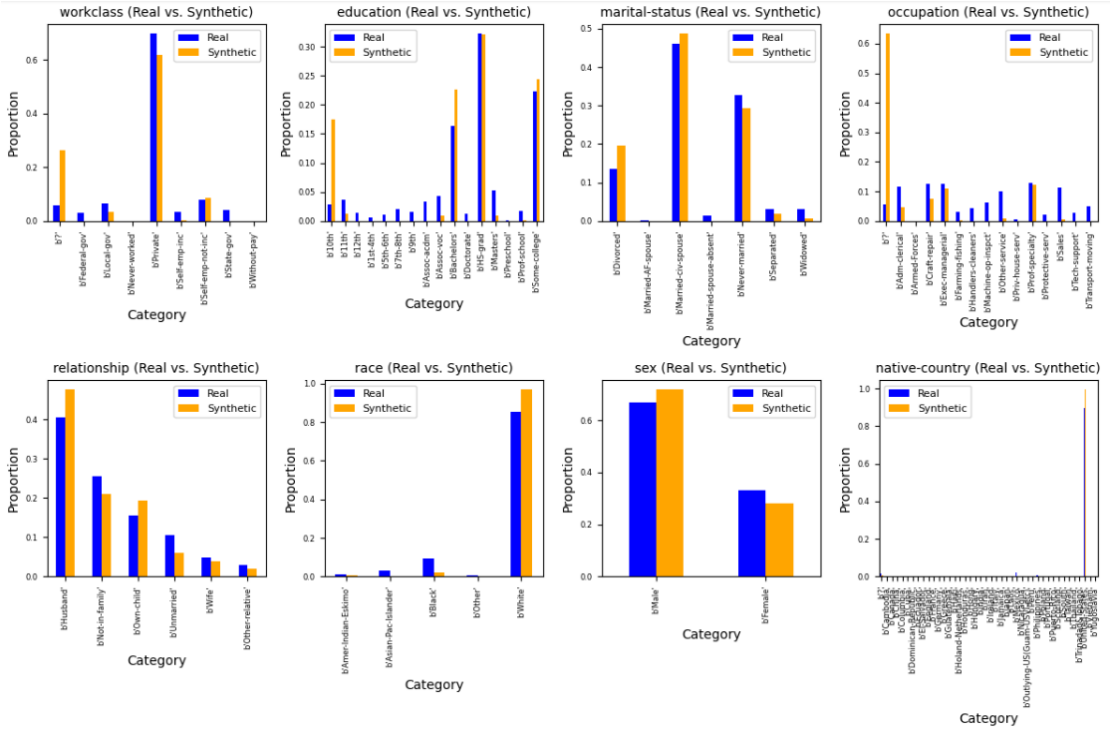
Comparing correlation matrices



Similar results to the model cGAN with AE model.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

Comparing the feature distributions of the synthetic data vs the original data – categorial features



Slightly less good results compared to the cGAN with AE model.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

Summary of results:

Model	Generator loss	Discriminator loss	Number of Epochs	Batch Size
GAN	3.333	0.2421	50	128
cGAN	3.561	0.2766		
cGAN with AE	**1.67**	**0.91**		
GAN with AE	1.94	0.7168		

The results show that adding an Autoencoder (AE) significantly impacts both the generator and discriminator losses. Without AE, the cGAN has higher generator and discriminator losses compared to the GAN. However, with AE, both models achieve considerably lower generator losses, with the cGAN showing the lowest at 1.67. The discriminator losses increase when AE is added, with the cGAN showing the highest discriminator loss (0.91). This suggests that incorporating AE helps improve the generator's performance while making the discriminator's task more challenging.

Evaluation: To evaluate the results, we employed two key metrics: Detection and Efficacy.

Detection Metric - The Detection metric evaluates how well the synthetic data mimics the real data.

- Combines real and synthetic data in a 50/50 mix.
- Trains a Random Forest on three folds of each data type (real/synthetic), then tests on the remaining fold (again, containing 50% real, 50% synthetic).
- The average AUC (Area Under the ROC Curve) measures how well the Random Forest can distinguish real from synthetic. Lower AUC indicates better similarity.

Efficacy Metric - the Efficacy metric measures the utility of synthetic data as a substitute for real data in training predictive models. It consists of:

- Training a Random Forest on real training data and evaluating on real test data to obtain auc_real_efficacy.
- Training a Random Forest on synthetic training data and evaluating on real test data to obtain auc_synth_efficacy.
- Computing the **Efficacy Ratio**: $\text{Efficacy Ratio} = (\text{auc synth efficacy}) \backslash (\text{auc real efficacy})$

  a ratio close to 1 indicates the synthetic data closely matches the utility of real data.

Ariel Siman Tov – 209449821, Tal Klein - 209234103

cGAN efficiency results

	average_auc_detection	auc_real_efficacy	auc_synth_efficacy	ratio_efficacy
12	1.0	0.902217	0.568879	0.630534
21	1.0	0.895185	0.411128	0.459266
27	1.0	0.901219	0.533555	0.592037
average	1.0	0.899540	0.504521	0.560612

GAN efficiency results

	average_auc_detection	auc_real_efficacy	auc_synth_efficacy	ratio_efficacy
12	1.0	0.902217	0.493467	0.546950
21	1.0	0.895185	0.494624	0.552539
27	1.0	0.901219	0.342685	0.380246
average	1.0	0.899540	0.443592	0.493245

cGAN with AE efficiency results

	average_auc_detection	auc_real_efficacy	auc_synth_efficacy	ratio_efficacy
12	0.999999	0.902217	0.510756	0.566112
21	1.000000	0.895185	0.625043	0.698228
27	1.000000	0.901219	0.550163	0.610466
average	1.000000	0.899540	0.561987	0.624935

GAN with AE efficiency results

	average_auc_detection	auc_real_efficacy	auc_synth_efficacy	ratio_efficacy
12	1.0	0.902217	0.451021	0.499903
21	1.0	0.895185	0.535645	0.598362
27	1.0	0.901219	0.544878	0.604601
average	1.0	0.899540	0.510514	0.567622

Key findings:

- **GAN and cGAN:** Both achieve similar detection performance, indicating bad abilities to mimic real data, because the average AUC Detection is 1.

  Both show moderate efficacy ratios, demonstrating that synthetic data is useful but not as reliable as the real data for training.
- **GAN with AE and cGAN with AE:** Adding the Autoencoder (AE) doesn't improve the detection performance. Adding the AE significantly boosts the efficacy ratio. In particular, **cGAN with AE** achieves the highest average efficacy ratio (0.62), indicating it produces the most effective synthetic data for model training.