Assignment 4

Nehoray chalfon - 325833531

Ori Sinvani - 325770824

1. Introduction

Generative Adversarial Networks (GANs) have emerged as a powerful framework for data synthesis, particularly in the domain of computer vision. However, adapting GANs to tabular data presents distinct challenges. Unlike image pixels, tabular data consists of heterogeneous columns—a mix of discrete categories, continuous values, and highly skewed distributions with frequent "point masses" (single values occurring with high probability). In this assignment, we implemented and evaluated two generative architectures—a standard GAN and a Conditional GAN (cGAN)—on the "Adult" Census Income dataset. Our objective was to generate synthetic tabular data that satisfies two conflicting criteria: realism (indistinguishability from real data, measured by the "Detection" metric) and utility (effectiveness for downstream machine learning tasks, measured by the "Efficacy" metric).

2. Methodology & Architecture Design

2.1. Preprocessing Decisions

To prepare the Adult dataset for the generative models, we implemented the following preprocessing pipeline:

- **Missing Values:** Missing values were imputed using the mode for categorical variables and the median for continuous variables.
- **Log-Transformation:** Exploratory analysis revealed that fnlwgt features were extremely right-skewed. To facilitate stable training, we applied a log-transformation ($\boldsymbol{log}(\boldsymbol{X}+\mathbf{1})$) to this column to compress its dynamic range. The capital-gain and capital-loss features were handled separately using the Mixture Model approach described below, as they exhibit zero-inflation rather than simple skewness.
- **Categorical Encoding:** All categorical features (workclass, education, marital-status, occupation, relationship, race, sex, native-country) were one-hot encoded. Crucially, we treated education-num as a categorical variable (with 16 distinct classes) rather than a continuous one. Early experiments showed that treating it as continuous resulted in the generator producing unrealistic fractional values (e.g., "9.5 years of education") which were easily detectable.

- **Normalization:** All continuous features were scaled to the range $[\mathbf{0}, \mathbf{1}]$ using Min-Max scaling.

2.2. GAN Architecture: The Mixture Model Approach

A standard GAN generator typically produces output using sigmoid or tanh activations, resulting in smooth, continuous probability distributions. However, the Adult dataset contains features with significant "point masses." For example, 91.7% of the dataset has a capital-gain of exactly 0, and 46.8% has hours-per-week of exactly 40. A standard GAN struggles to reproduce these sharp spikes, often generating values close to zero (e.g., 0.001) but rarely exactly zero. This discrepancy allows a discriminator or downstream classifier to trivially distinguish real from synthetic data.

To address this, we designed a custom Mixture Model Generator. For features identified as "Zero-Inflated" (capital-gain, capital-loss) or "Peak-Inflated" (hours-per-week), the generator outputs three distinct components:

1. **Probability ($\boldsymbol{P}$):** A sigmoid output predicting the probability that the value belongs to the specific mode (e.g., exactly 0 or 40).
2. **Continuous Value ($\boldsymbol{V}$):** A continuous value generated for the non-mode cases.
3. **Mode Sampling:** During the generation process, we sample from a Bernoulli distribution parameterized by $\boldsymbol{P}$. If the mode is selected, the output is forced to the exact discrete value (0 or 40); otherwise, the continuous value $\boldsymbol{V}$ is used.

The Generator network is a 3-layer Multi-Layer Perceptron (MLP) with hidden dimensions of 256 → 512 → 512, using a latent noise dimension of 128. We incorporated Layer Normalization after each hidden layer and Residual Skip Connections to improve gradient flow. The generator uses Gumbel-Softmax for differentiable sampling of categorical features, enabling end-to-end training. The Discriminator employs Spectral Normalization on all layers to stabilize training and prevent mode collapse.

2.3. Conditional GAN (cGAN) Architecture

The Conditional GAN extends the architecture described above by conditioning the generation process on the target label (income: <=50K vs. >50K). We utilized concatenation-based conditioning:

- **Generator:** The one-hot encoded label is concatenated to the random noise vector $\boldsymbol{Z}$ before being fed into the network.
- **Discriminator:** The one-hot encoded label is concatenated to the input data sample (real or synthetic).

This conditioning allows the model to learn the specific feature distributions associated with each income class, rather than learning a global average of the entire population.

2.4. Loss Functions

For stability, both models were trained using WGAN-GP. Unlike the standard Jensen-Shannon divergence used in regular GANs, the used distance provides a smoother gradient for the generator even when the real and synthetic distributions dont overlap. We also incorporated auxiliary losses to enforce statistical alignment:

- **Correlation Loss:** Penalizes the $\boldsymbol{L_2}$ distance between the correlation matrices of the real and synthetic batches, ensuring the generator captures inter-feature dependencies.
- **Moment Loss:** Penalizes differences in the mean and standard deviation of continuous features.
- **Proportion Loss:** Explicitly penalizes the generator if the predicted probability of point masses (e.g., zeros) deviates from the real batch proportion.

Quantile Loss: Matches the quantiles of continuous feature distributions between real and synthetic data, ensuring the generator captures the full range of values.

Categorical Loss: Penalizes differences in categorical feature frequency distributions between real and synthetic batches.

3. Training Process

We trained the models using the following hyperparameters and configuration:

- **Optimizer:** Adam optimizer with a Two-Time-Scale Update Rule (TTUR).
  - Generator Learning Rate: 2×10^{-4} $\mathbf{2 * 10^{-4}}$
  - Discriminator Learning Rate: 1×10^{-4} $\mathbf{10^{-4}}$
- **Batch Size:** 256 samples.
- **Epochs:** 300.
- **Critic Iterations:** 5 discriminator updates for every 1 generator update.
- **Gradient Penalty is 10**.

The training process utilized an 80%/20% train-test split, maintaining label ratios. We monitored the Wasserstein distance and auxiliary losses to ensure convergence (see Figure 1 and 4 in Appendix).

4. Evaluation and Results

To ensure the robustness of our results, we evaluated the models across three seeds (42, 123, 456). The following results represent the average performance across these three experiments. For both GAN and cGAN, we generated synthetic datasets equal in size to the training set (~26,000 samples). For the cGAN, we maintained the same label distribution as the original training data (75.9% ≤50K, 24.1% >50K).

4.1. Quantitative Results

Metric	**GAN (Unconditional)**	**cGAN (Conditional)**	**Ideal Score**
Detection AUC	**0.8845** (± 0.0112)	**0.8694** (± 0.0039)	0.5 (Indistinguishable)
Efficacy Ratio	**0.5688** (± 0.0402)	**0.9508** (± 0.0028)	1.0 (High Utility)

4.2. Analysis of Detection Metric

The Detection metric measures the ability of a Random Forest classifier to distinguish

synthetic data from real data. A score of 0.5 indicates the data is indistinguishable, while 1.0 indicates trivial detection. We implemented this using 4-fold stratified cross-validation: the training data and synthetic data (each equal in size) are combined into a 50/50 dataset. A Random Forest classifier is trained on 3 folds and evaluated on the held-out fold. This process is repeated 4 times, and we report the average AUC.

Our unconditional GAN achieved a detection AUC of 0.8845, and the cGAN achieved 0.8694. These scores indicate that while the synthetic data is not perfectly indistinguishable, it successfully mimics the major statistical properties of the real data. In our preliminary experiments (prior to implementing the Mixture Model), the Detection AUC was 1.0 because the classifier could trivially identify synthetic samples by checking if capital-gain was exactly zero. The introduction of the Mixture Model solved this "trivial detection" problem(see peeks in figures 2 and 5 in the Appendix). The remaining distinguishability (0.87 vs 0.5) likely stems from subtle differences in the tails of complex continuous distributions (like fnlwgt or age) which are notoriously difficult for neural networks to model perfectly.

4.3. Analysis of Efficacy Metric

The Efficacy metric measures the utility of the synthetic data for training downstream classifiers. It is calculated as the ratio of the AUC of a model trained on synthetic data to the AUC of a model trained on real data. Specifically, we first train a Random Forest on the original training data and evaluate it on the held-out test set to obtain AUC_real. Then, we train another Random Forest on the synthetic data and evaluate it on the same test set to obtain AUC_synth. The efficacy ratio is computed as AUC_synth / AUC_real.

- **GAN (0.57 - Poor Utility):** The unconditional GAN performs poorly on the efficacy metric, with AUC_real = 0.902 and AUC_synth = 0.513. Since it is not provided with labels during training, it is forced to learn the average distribution of the entire population. It blurs the critical distinctions between high-income and low-income features. Consequently, a classifier trained on this "muddy" data fails to learn sharp decision boundaries, resulting in performance only slightly better than random guessing.

- **cGAN (0.95 - High Utility):** The Conditional GAN demonstrates exceptional utility, with AUC_real = 0.902 and AUC_synth = 0.857, achieving roughly 95% of the performance of real data. By conditioning on the income label, the generator learns two distinct manifolds—one for high-income earners and one for low-income earners. It successfully captures class-specific correlations, such as the fact that high income is positively correlated with education-num and capital-gain. This result strongly suggests that conditional generation is strictly necessary when the goal is to produce tabular data for supervised learning tasks.

4.4. Visualizations

The visual analysis confirms our quantitative findings. The training loss plots (Figure 1 and 4) show the characteristic fluctuation of Wasserstein loss, indicating active adversarial learning. The feature distribution histograms (Figures 2 and 5) demonstrate that the Mixture Model successfully reproduces the sharp peaks at 0 for capital-gain and 40 for hours-per-week. Finally, the correlation analysis (Figure 3) confirms that the cGAN preserves the complex inter-feature relationships required for efficacy.

5. Conclusion

We developed and evaluated GAN and cGAN architectures. We identified that standard GANs are not good at handling the mixed data types and point masses inherent in the Adult dataset. By implementing a custom Mixture Model Generator with WGAN-GP, we significantly improved the realism of the generated data, reducing the detection AUC from a baseline of 1.0 to approximately 0.87. Furthermore, our experiments highlighted the critical importance of conditioning: while the unconditional GAN could generate statistically reasonable data, only the Conditional GAN could preserve the predictive relationships necessary for downstream tasks, achieving an efficacy ratio of 0.95.

Appendix:

Figure 1: GAN Training Losses (Seed 42)

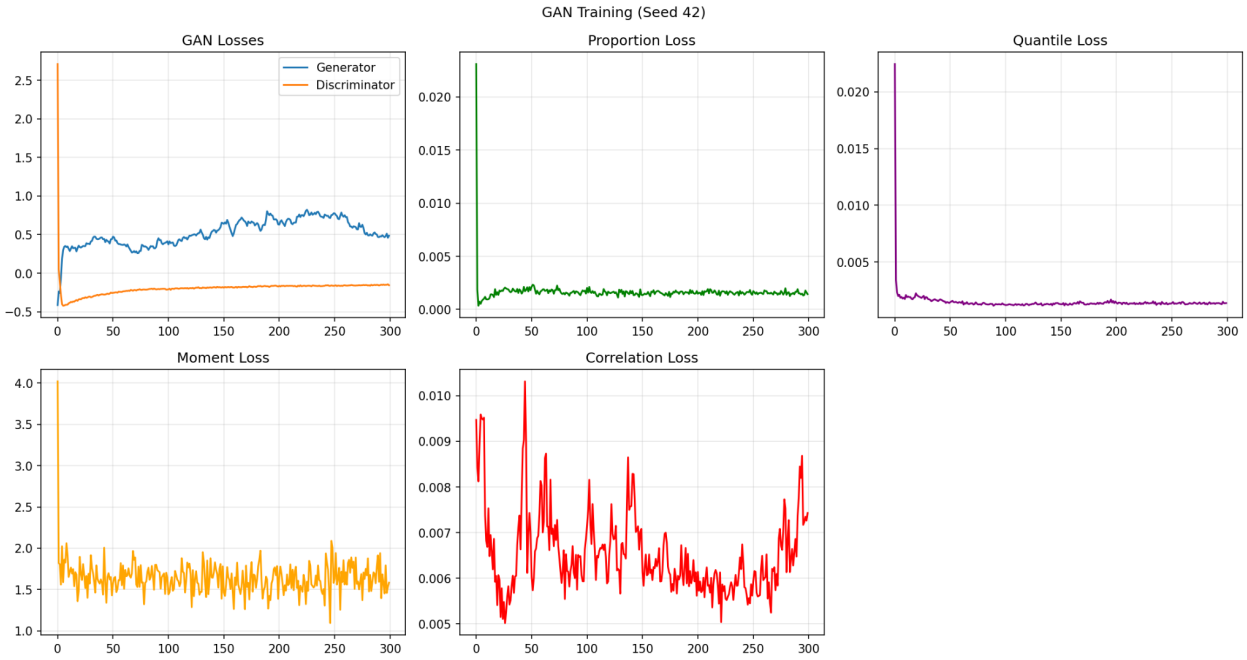

Figure 2: GAN Feature Distributions (Seed 42)

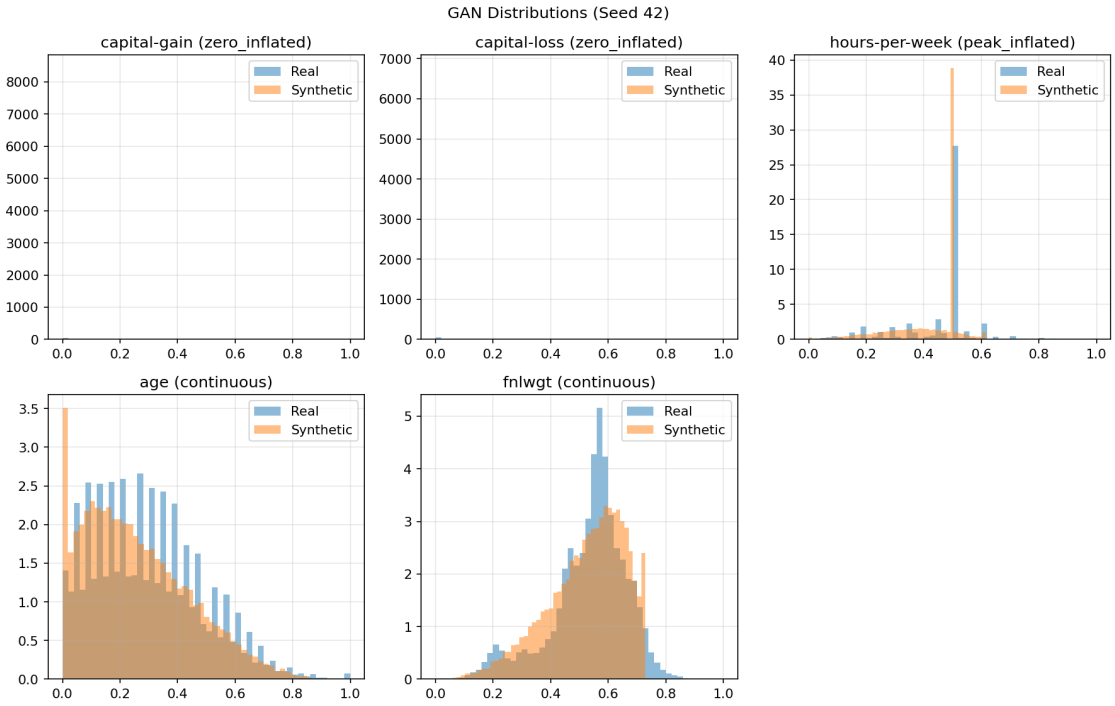

Figure 3: Correlation Matrix Difference

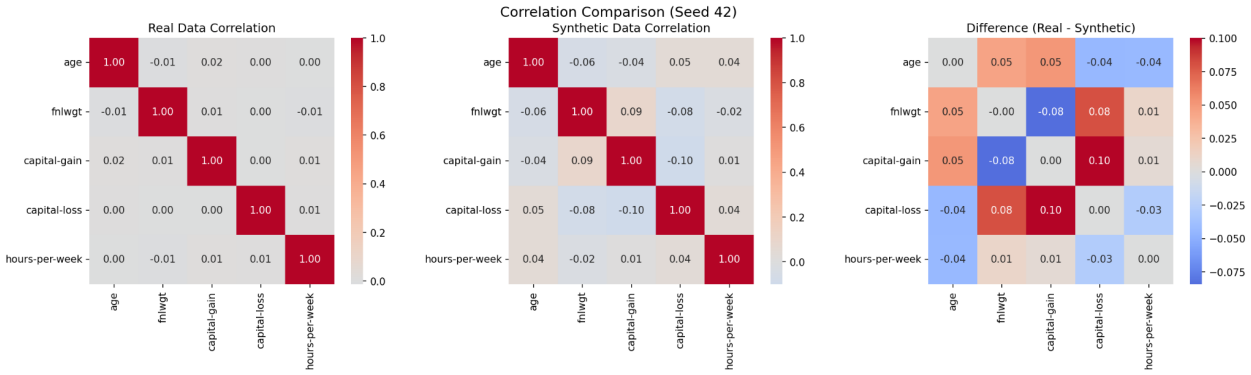

Figure 4: CGAN Training Losses (Seed 42)

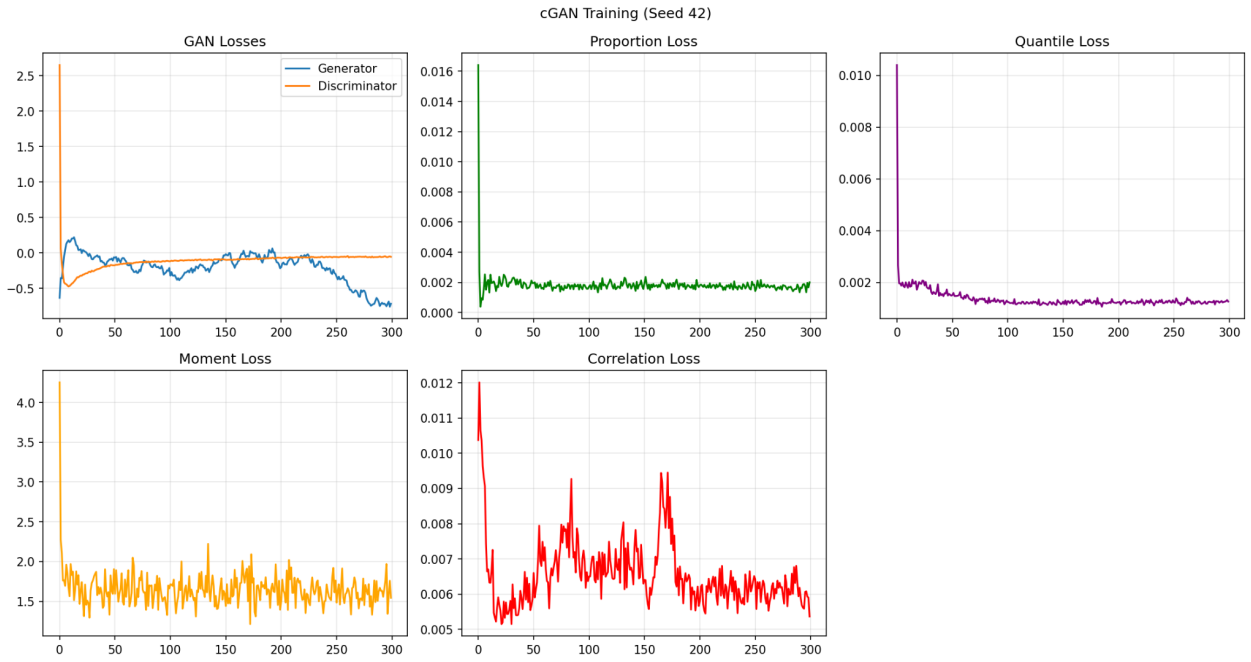

Figure 5: CGAN Feature Distributions (Seed 42)