APPENDIX

A. Implementation Details

For MNIST and f-MNIST, we use standard, fully connected networks for both generators and discriminators. On the other hand, we adopt a Deep Convolutional Generative Adversarial Network (DCGAN) structure for EMNIST [40], CIFAR-10, and CelebA datasets. All GAN models have identical generator and discriminator architectures for the respective dataset (see details below).

For evaluation, we train the privGAN models with an Adam (β=5) optimizer for both the generator and discriminator. For differentially private GANs (i.e., DPGAN, DP-FedProx GAN, DP-FedAvg GAN, and DP-FedSGD GAN), we use Differentially Private Stochastic Gradient Descent (DP-SGD) optimizer for the discriminator. For the generator, we use Adam (β=5) optimizer in DPGAN and SGD optimizer in DP-FedProx GAN, DP-FedAvg GAN, and DP-FedSGD GAN. We set a 0.0002 learning rate for all optimizers. The batch size adopted for privGAN is 256. For DP-GAN, the batch size is varied from 16 to 64 in order to meet the specified privacy parameter ϵ. For DP-FedProx GAN and DP-FedAvg GAN, the batch size is set to 32 for CIFAR-10 and CelebA and to 64 for other datasets. For each round in DP-FedSGD GAN, we randomly select a batch size of qD^k for the k-th party, where q signifies the sampling probability (0.1) and D^k refers to the local data size of the k-th party. The default value for DP-FedSGD GAN is set at ϵ=98.01.

While evaluating the different adversarial attacks on PrivGAN, we train PrivGAN for 500 epochs using the same optimizer and hyperparameters. Similarly, we train all differentially private GAN models with a fixed privacy budget ϵ, setting the noise scale z to achieve the specified ϵ. For WB and TVD attacks, we use 10% of the training set to train models, following the approach in [5].

To evaluate the MC attack, we follow the methodology in [26], [35]. We use 10% of the training set for training the attack model and evaluate the model on the residual 90% of the training set. The test set is utilized solely to calculate the principal components for all datasets. In federated training (i.e., DP-FedProx GAN, DP-FedAvg GAN, and DP-FedSGD GAN), the central server generates 100,000 synthetic samples. In local training (i.e., PrivGAN and DPGAN), we sample 100,000 synthetic samples from each party.

In order to ensure the results are representative, we performed each experiment five times and reported the average outcomes for both utility and adversarial assessments.

B. Complexity Analysis

In DP-FedProx GAN and DP-FedAvg GAN, the time complexity for each party is $O(nBT))$, where n represents the number of steps for local discriminator and generator updates, B represents the batch size, and T represents the total number of rounds. In the context of privGAN and DPGAN, the time complexity for each party can be expressed as $O(DE)$, where D is the upper bound of local data size, and E corresponds to the number of epochs for the local model training.

◇ **MNIST and f-MNIST**

Generator Layers

- Dense(units= 256, input size= 100)
- LeakyReLU(α = 0.2)
- Dense(units= 512)
- LeakyReLU(α = 0.2)
- Dense(units= 1024)
- LeakyReLU(α = 0.2)
- Dense(units= 784, activation = 'tanh')

Discriminator Layers

- Dense(units= 1024)
- LeakyReLU(α = 0.2)
- Dense(units= 512)
- LeakyReLU(α = 0.2)
- Dense(units= 256)
- LeakyReLU(α = 0.2)
- Dense(units= 1, activation = 'sigmoid')

Privacy Discriminator Layers in privGAN

- Dense(units= 1024)
- LeakyReLU(α = 0.2)
- Dense(units= 512)
- LeakyReLU(α = 0.2)
- Dense(units= 256)

- LeakyReLU(α = 0.2)
- Dense(units = # generators, activation ='softmax')

◇ **CIFAR-10**

Generator Layers

- Dense(units= 2048, input size= 100, target shape= (2, 2, 512))
- Conv2DTranspose(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2DTranspose(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2DTranspose(filters= 64, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2DTranspose(filters= 3, kernel size= 5, strides= 2, activation = 'tanh')

Discriminator Layers

- Conv2D(filters= 64, kernel size= 5, strides= 2)
- Reshape(target shape= (2, 2, 512))
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Dense(units= 1, activation = 'sigmoid')

Privacy Discriminator Layers in privGAN

- Conv2D(filters= 64, kernel size= 5, strides= 2)
- Reshape(target shape= (2, 2, 512))
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Dense(units = # generators, activation = 'softmax')

◇ **CelebA**

Generator Layers

- Dense(units= 2048, input size= 100, target shape= (2, 2, 512))
- Conv2DTranspose(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2DTranspose(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2DTranspose(filters= 64, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2DTranspose(filters= 3, kernel size= 5, strides= 3, activation = 'tanh')

Discriminator Layers

- Conv2D(filters= 64, kernel size= 5, strides= 2)
- Reshape(target shape= (2, 2, 512))
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Dense(units= 1, activation = 'sigmoid')

Privacy Discriminator Layers in privGAN

- Conv2D(filters= 64, kernel size= 5, strides= 2)

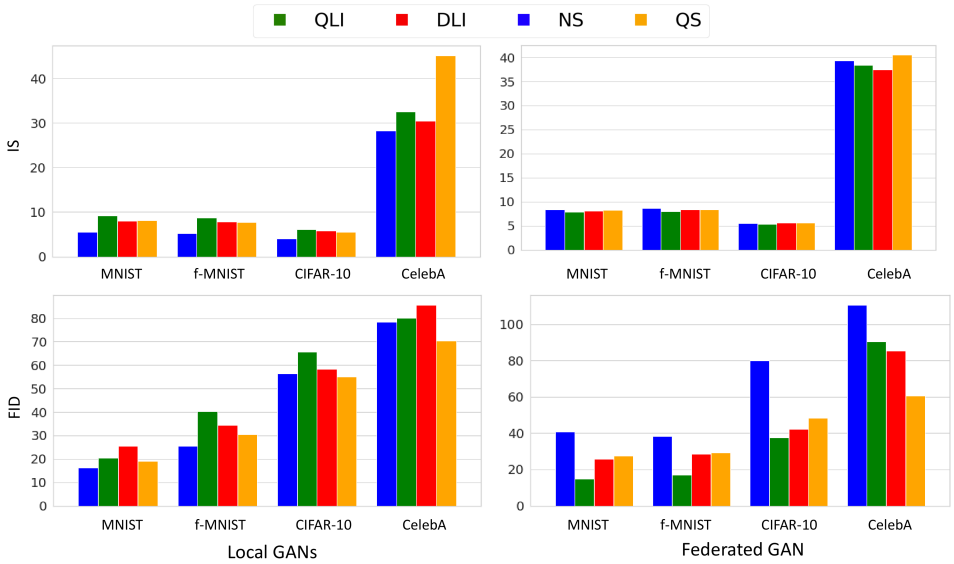

Fig. 9: Inception Score with Non-Private Local GANs and Federated GAN: The Figure shows the inception score and FID score for different distributions for each dataset in local GANs and federated GAN with $K = 10$.

- Reshape(target shape= (2, 2, 512))
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Conv2D(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU(α = 0.2)
- Dense(units = # generators, activation = 'softmax')

◇ **EMNIST**

Generator Layers

- Dense(units= 1024, input size= 100)
- Dense(units= 7 * 7 * 256)
- Reshape(target shape= (7, 7, 256))
- Conv2D(filters= 64, kernel size= 4, strides= 2)
- BatchNormalization()
- LeakyReLU(α = 0.01)
- Conv2D(filters= 32, kernel size= 4, strides= 2)
- BatchNormalization()
- LeakyReLU(α = 0.01)
- Conv2D(filters= 1, kernel size= 4)

Discriminator Layers

- Conv2D(filters= 64, kernel size= 4, strides= 2)
- LeakyReLU(α = 0.01)
- Conv2D(filters= 128, kernel size= 4, strides= 2)
- LeakyReLU(α = 0.01)
- Flatten()
- Dense(units= 1024)
- LeakyReLU(α = 0.01)
- Dense(units= 1, activation = 'sigmoid')

C. Additional Experiments on Utility and Membership Inference Attacks

For all following experiments, we use the default parameter values listed in Section 6 unless otherwise stated.

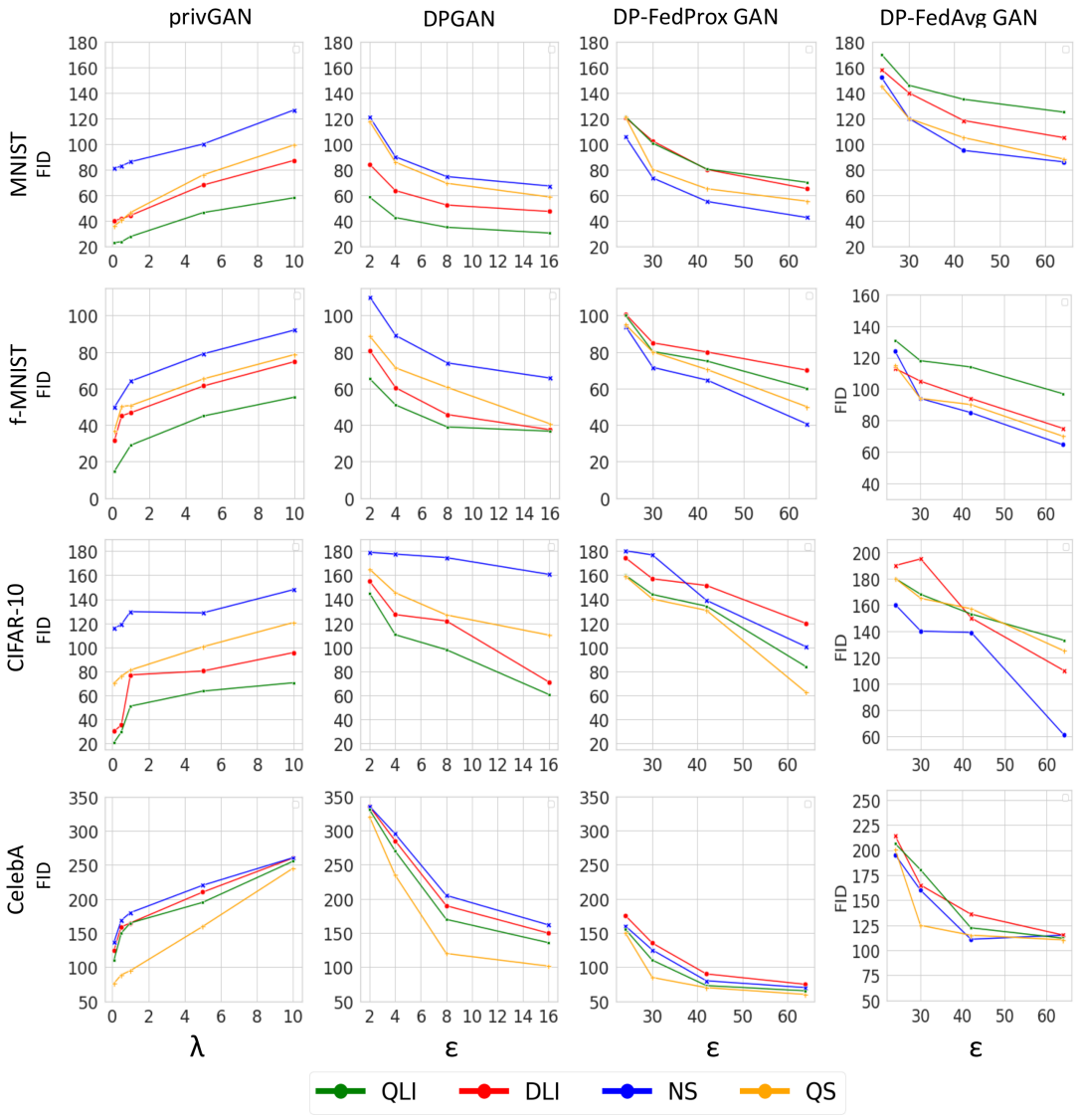

Fig. 10: Varying Privacy Parameters: The Figure illustrates FID Score for various privacy parameters in privGAN, DPGAN, DP-FedProx GAN, and DP-FedAvg GAN.

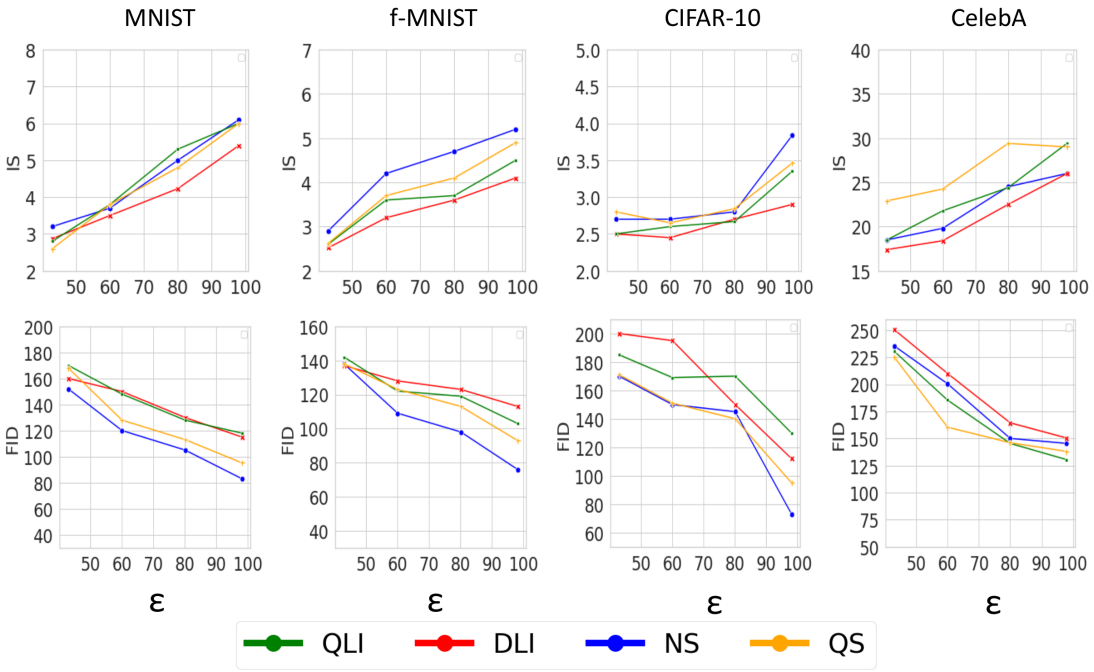

Fig. 11: Varying Privacy Parameters in DP-FedSGD GAN: The Figure represents inception and FID Score at various privacy parameters in DP-FedSGD GAN. DP-FedSGD GAN has the worst utility in terms of IS and FID compared to the DP-FedProx GAN and DP-FedAvg GAN, as shown in the Fig 3 and 10.

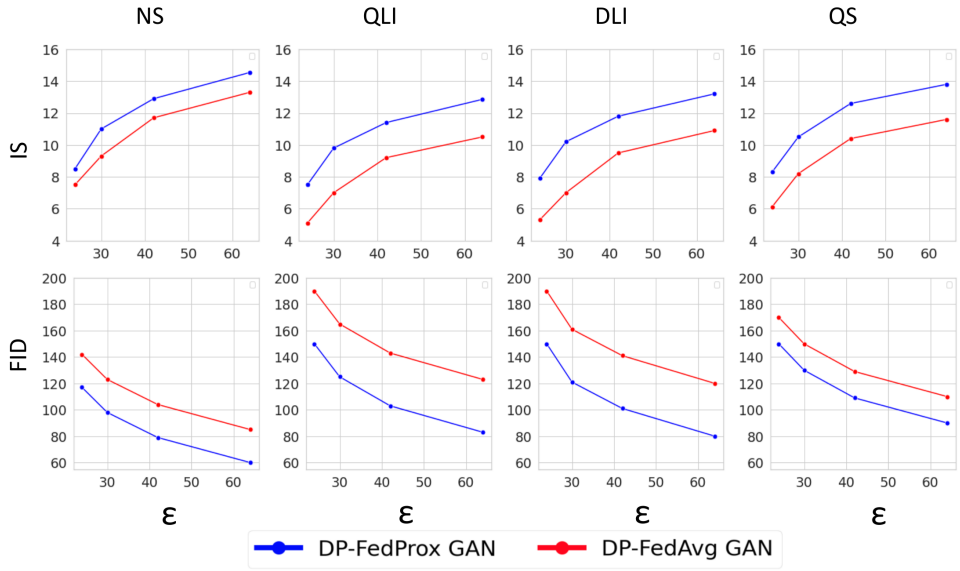

Fig. 12: Varying Privacy Parameters on EMNIST. The Figure depicts the inception and FID score at various privacy parameters in DP-FedProx GAN and DP-FedAvg GAN at K=500 on EMNIST.

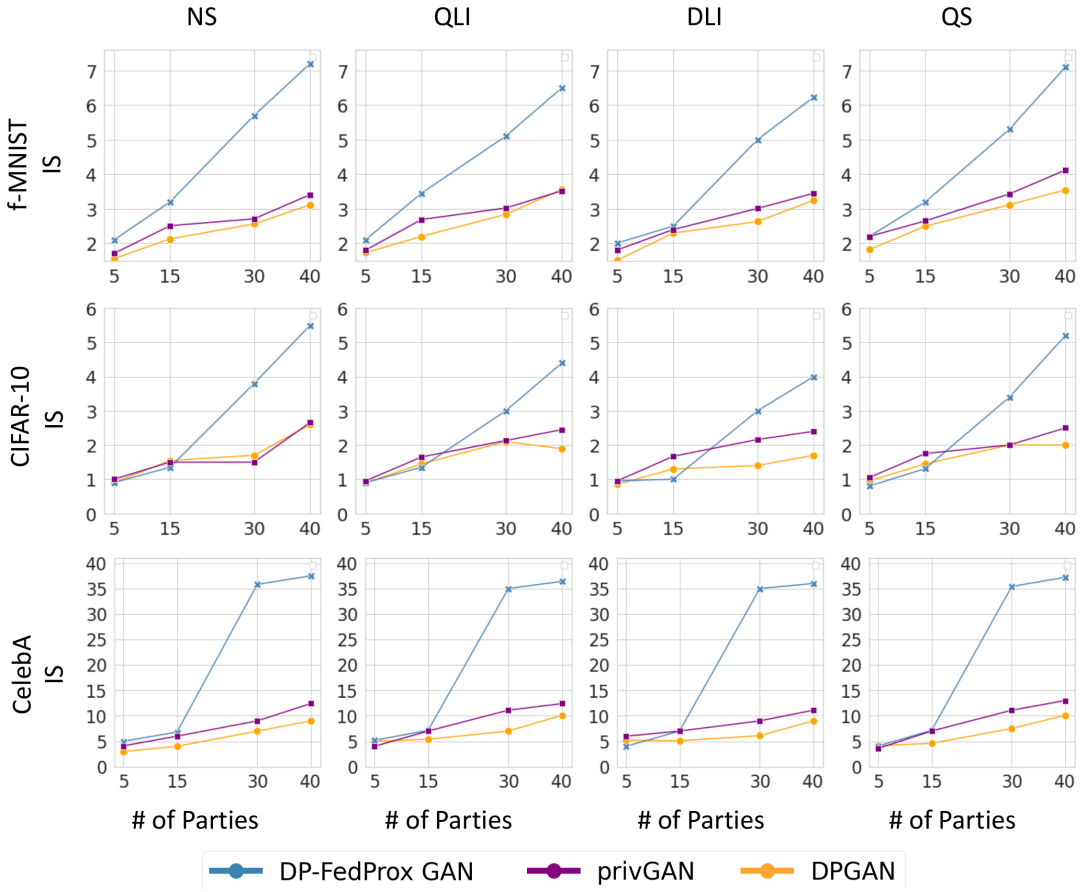

Fig. 13: Varying Number of Parties: The Figure depicts the inception score for various distributions at various numbers of parties in privGAN, DPGAN, and DP-FedProx GAN.

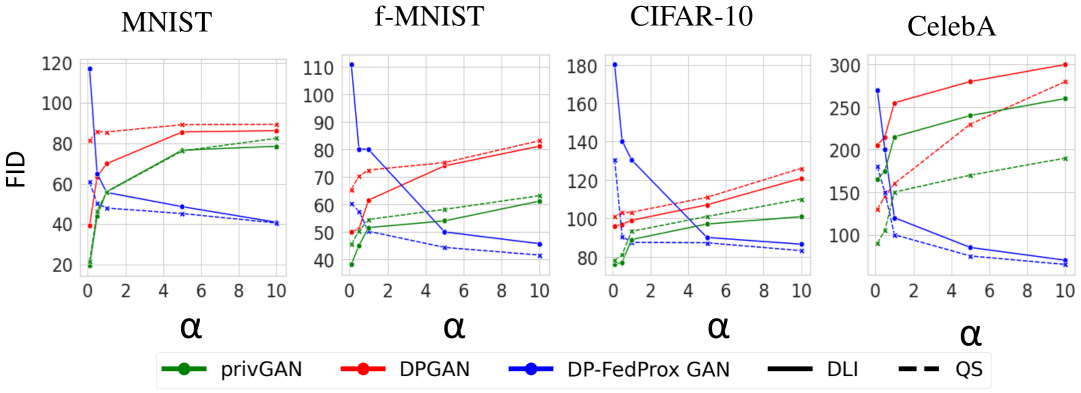

Fig. 14: Varying Concentration Parameters: The Figure illustrates the FID score for different distributions at various concentration parameters in privGAN, DPGAN, and DP-FedProx GAN.

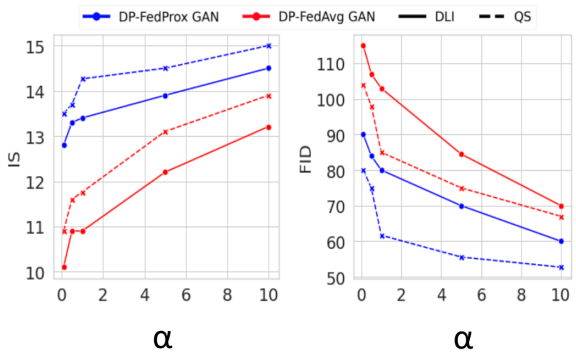

Fig. 15: Varying Concentration Parameter on EMNIST with 500 Parties (K=500) using DP-FedProx GAN and DP-FedAvg GAN .

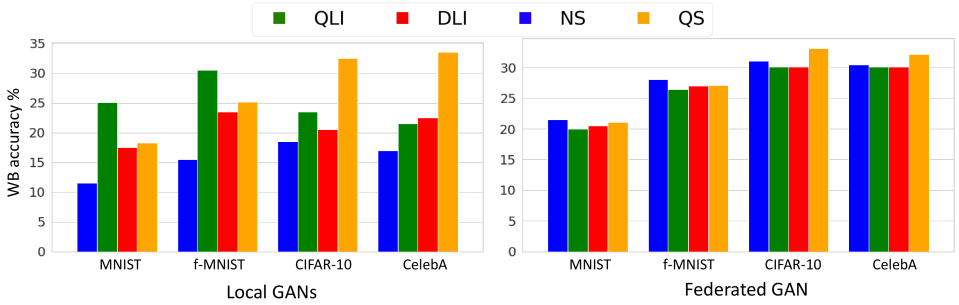

Fig. 16: White-Box accuracy for Non-Private Local GANs and Federated GAN: The Figure displays the results of a white-box attack using non-private local GANs and federated GAN with $K = 10$ for various distributions for each dataset.

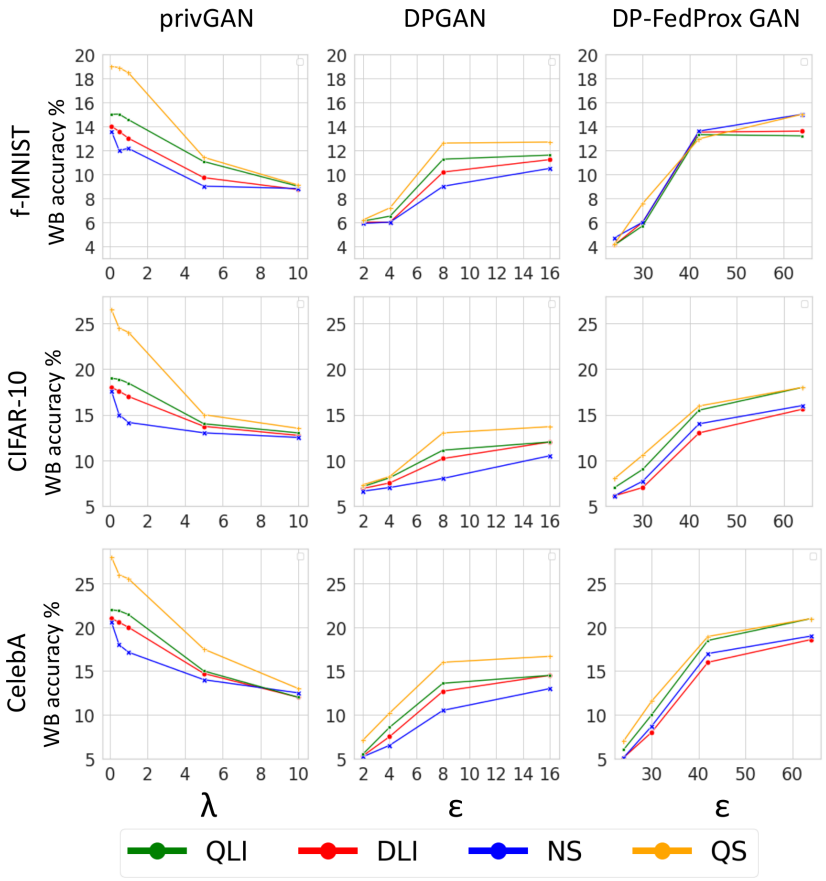

Fig. 17: Varying privacy parameters for White-Box Attack: The Figure shows white-box accuracy on discriminator(s) in privGAN, DPGAN, and DP-FedProx GAN for various privacy parameters.

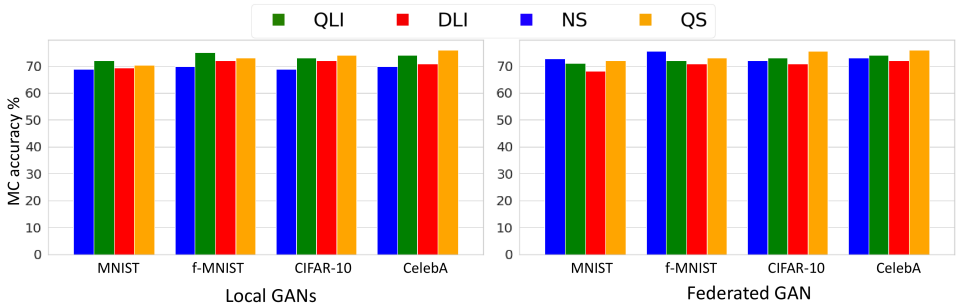

Fig. 18: Monte-Carlo accuracy for Non-Private Local GANs and Federated GAN: The Figure displays the results of a monte-carlo attack using non-private local GANs and federated GAN with $K = 10$ for various distributions for each dataset.

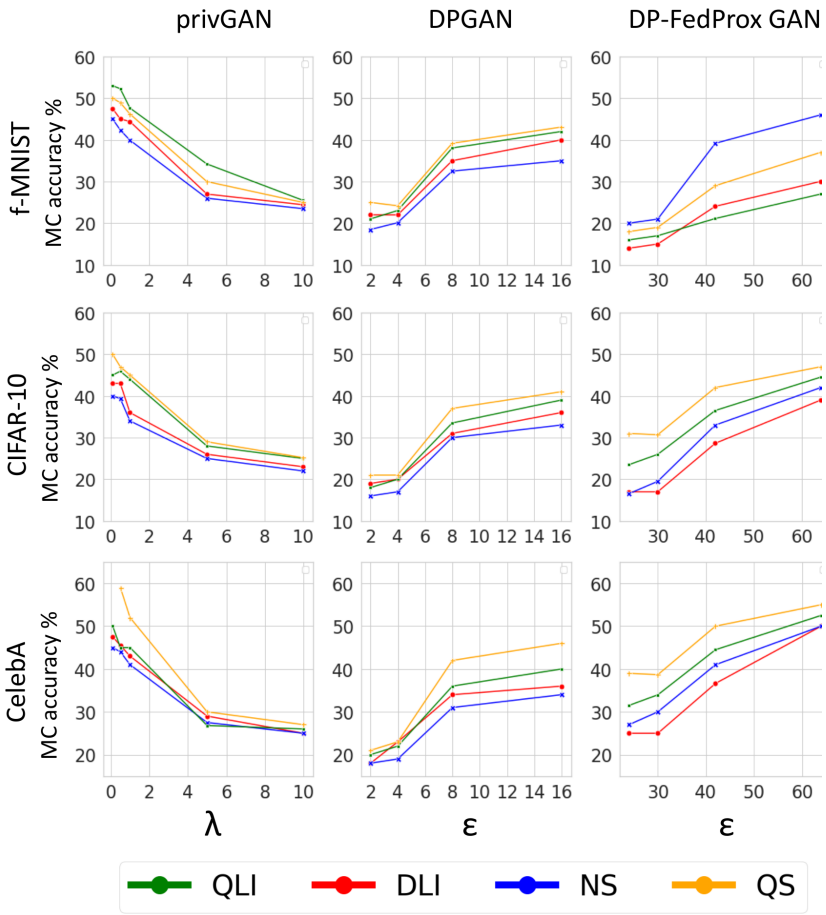

Fig. 19: Varying privacy parameters for Monte-Carlo Attack: The Figure shows Monte-Carlo Attack on the generator(s) in privGAN, DPGAN, and DP-FedProx GAN for various privacy parameters.

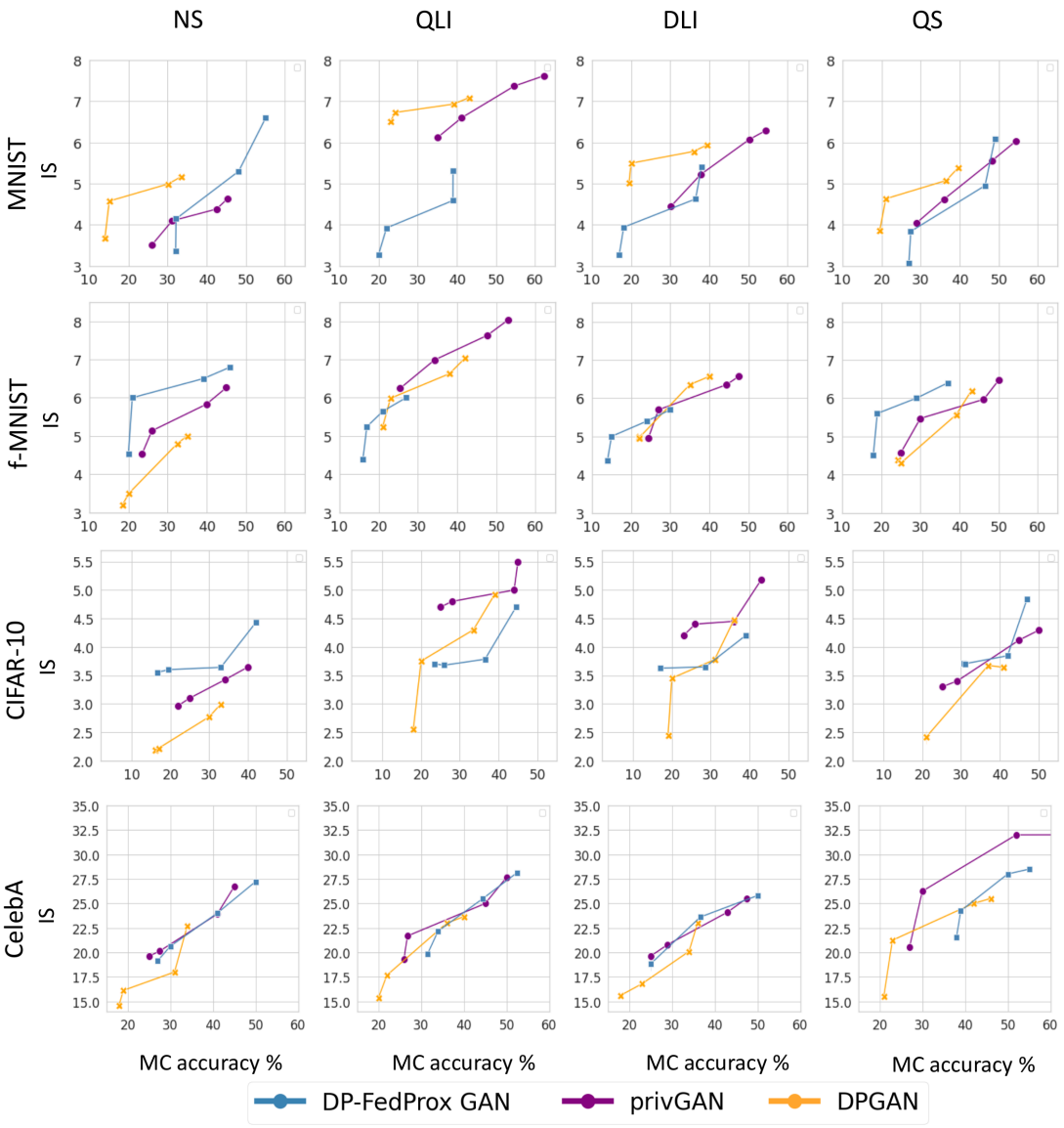

Fig. 20: Privacy vs. Utility: The Figure shows Monte-Carlo Attack vs Inception Score in privGAN, DPGAN, and DP-FedProx GAN for various privacy parameters.

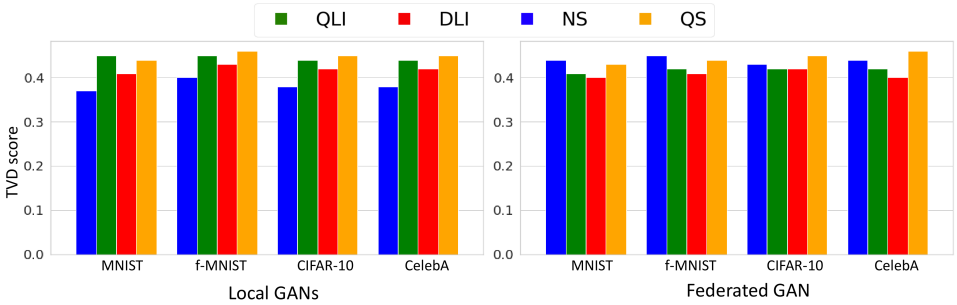

Fig. 21: TVD score for Non-Private Local GANs and Federated GAN: The Figure displays the results of a TVD attack score using non-private local GANs and federated GAN with $K = 10$ for various distributions for each dataset.

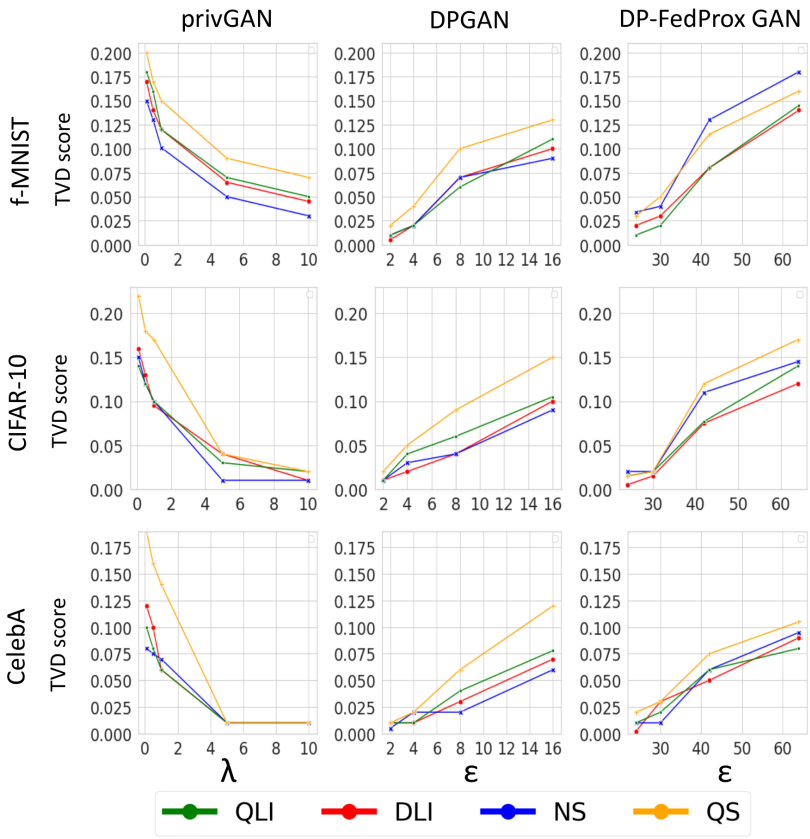

Fig. 22: Varying privacy parameters for TVD Attack: For different privacy parameters, the Figure displays the TVD attack score on the discriminator(s) in privGAN, DPGAN, and DP-FedProx GAN.

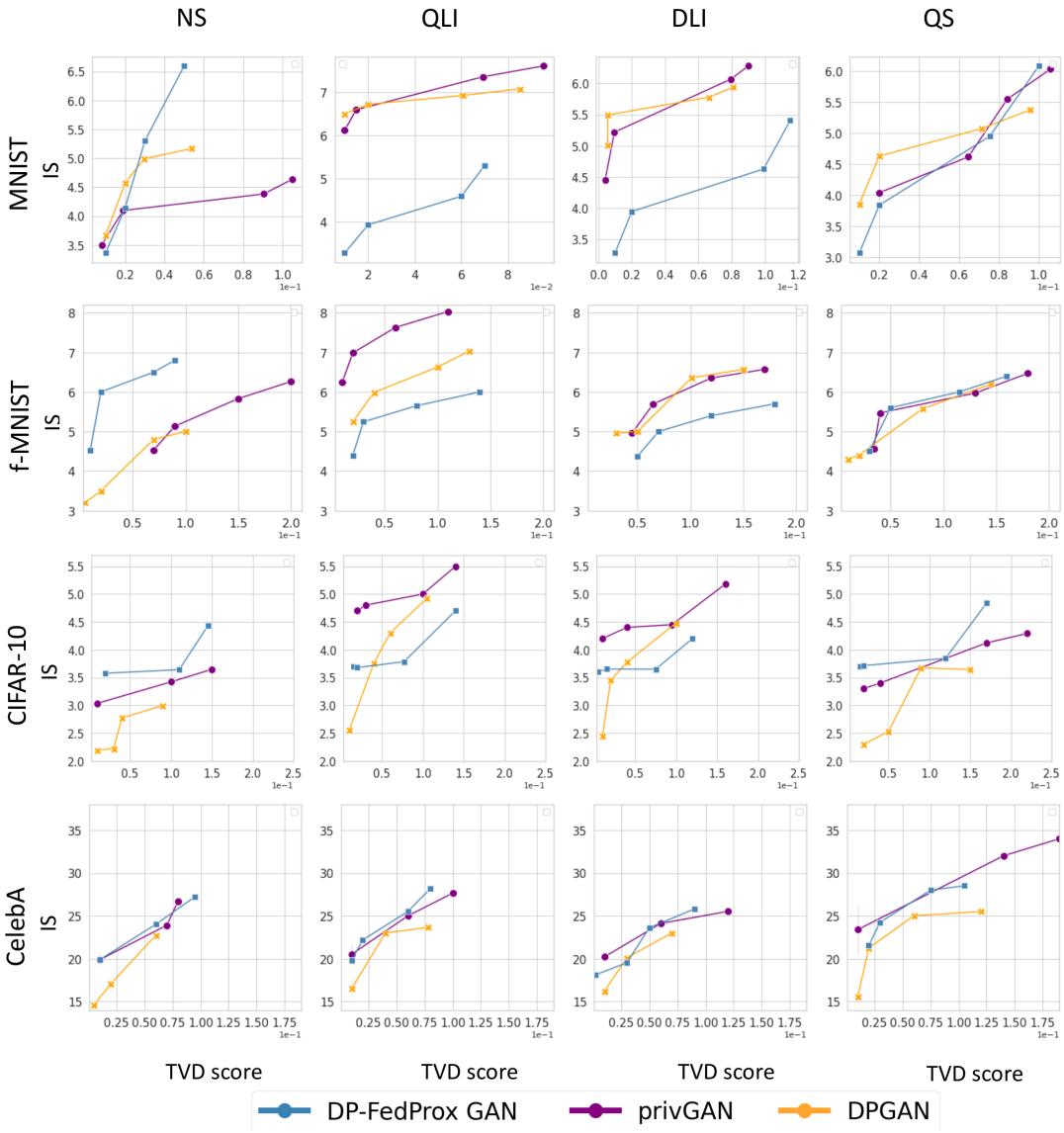

Fig. 23: Privacy vs. Utility: The Figure shows TVD Attack vs Inception Score in privGAN, DPGAN, and DP-FedProx GAN for various privacy parameters.