Hochschule der Medien
Fakultät für Druck und Medien
Computer Science and Media

Generative Data Augmentation

Multi-Agent Diverse Generative Adversarial Networks for Generative Data Augmentation

Dissertation submitted for the degree of Master of Science

Topic: Generative Data Aufmentation

Author: Nicolas Reinhart - nr063@hdm-stuttgart.de
MatNr. 44100

Version of Date: May 15, 2025

1. Advisor: Prof. Dr.-Ing. Johannes Maucher
2. Advisor: Prof. Dr.-Ing. Oliver Kretzschmar

Abstract

asdf

Contents

List of Figures 6

List of Tables 7

List of Abbreviations 8

1 Introduction and Motivation 9

2 Related Work 11

3 Theoretical Background 13
3.1 Image Classification Models . . . 13
3.1.1 Neural Networks for Classification . . . 13
3.1.2 Classification Models for augmented Training . . . 16
3.1.3 Classification Model Performance Metrics . . . 16
3.2 Data Augmentation - DA . . . 17
3.2.1 Traditional Data Augmentation - TDA . . . 17
3.2.2 Generative Data Augmentation - GDA . . . 18
3.3 Generative Adversarial Network - GAN . . . 20
3.3.1 Mathematical Formulation . . . 20
3.3.2 Training Process . . . 21
3.3.3 Challenges in GAN Training . . . 21
3.4 Deep Convolutional Generative Adversarial Network - DCGAN . . . 22
3.4.1 Architectural Adjustments . . . 22
3.5 Conditional Generative Adversarial Network - cGAN . . . 23
3.5.1 Mathematical Formulation . . . 23
3.5.2 Architectural Adjustments . . . 24
3.6 Multi-Agent Diverse Generative Adversarial Network - MADGAN . . . 24
3.6.1 Mathematical Formulation . . . 25
3.6.2 Architectural Adjustments . . . 27
3.7 Adapting MAD-GAN for Conditional Generation with Explicit Diversity - CMADGAN . . . 27
3.7.1 Mathematical Formulation . . . 28
3.7.2 Architectural Adjustments . . . 29
3.8 Image Scores . . . 30
3.8.1 Inception Score - IS . . . 30
3.8.2 Fréchet Inception Distance - FID . . . 31
3.8.3 InceptionV3 Model . . . 32

4 Experiments Setup 34
4.1 Preliminary Remarks . . . 34
4.1.1 Scope Limitation Regarding Standard CIFAR-10 . . . 34
4.1.2 Used Datasets . . . 35
4.1.3 GAN: Architecture, Training and Data Augmentation . . . 35
4.1.4 Stratified Classifiers as measure for augmentation Quality . . . 36
4.1.5 Labeling unconditioned data . . . 36
4.1.6 Utilization of InceptionV3 for FID and IS . . . 36
4.2 Experimental Workflow . . . 37

4.3 Comparison of Classifier Performance . . . 38
4.4 Hardware and Software Environment . . . 38
4.4.1 Hardware . . . 38
4.4.2 Software . . . 39

5 Experiments Results **40**
5.1 Key Research Questions . . . 40
5.2 Key Research Question Answers . . . 41
5.2.1 Question 1 . . . 42
5.2.2 Question 2 . . . 44
5.2.3 Question 3 . . . 53
5.2.4 Question 4 . . . 62
5.2.5 Question 5 . . . 70

6 Remarks **72**

7 Outlook **73**

8 Conclusion **74**

List of References **75**

Appendix **1**
8.1 Network Architectures . . . 1
8.1.1 Classifiers . . . 1
8.1.2 Generator Model Architectures . . . 2
8.1.3 Discriminator Model Architectures . . . 3
8.2 FID and Inception Scores from MADGAN Architectures . . . 5
8.2.1 MADGAN MNIST . . . 5
8.2.2 MADGAN Fashion-MNIST . . . 6
8.2.3 cMADGAN MNIST . . . 7
8.2.4 cMADGAN Fashion-MNIST . . . 8
8.2.5 DCGAN MNIST . . . 9
8.2.6 DCGAN Fashion-MNIST . . . 9
8.2.7 Conditional MNIST . . . 9
8.2.8 Conditional Fashion-MNIST . . . 9
8.3 Stratified Classifier Performances and Graphs . . . 10
8.3.1 Dataset: MNIST, Architecture: MADGAN . . . 10
8.3.2 Dataset: MNIST, Architecture: cMADGAN . . . 14
8.3.3 Dataset: FASHION, Architecture: MADGAN . . . 18
8.3.4 Dataset: FASHION, Architecture: cMADGAN . . . 22
8.3.5 Dataset: MNIST, Architecture: DCGAN . . . 26
8.3.6 Dataset: MNIST, Architecture: cGAN . . . 27
8.3.7 Dataset: FASHION, Architecture: DCGAN . . . 28
8.3.8 Dataset: FASHION, Architecture: cGAN . . . 29
8.3.9 Dataset: MNIST, Architecture: TDA . . . 30
8.3.10 Dataset: FASHION, Architecture: TDA . . . 31
8.4 Other Graphs and Figures . . . 32
8.4.1 DCGAN MNIST, Mode Collapse . . . 32
8.4.2 Convolutional Filtering . . . 32

List of Figures

1 Exemplary use of traditional augmentation techniques from the categories *geometric* (first row), *photometric* (second row), and *noise-corruption* (third row). The image shown is an image from the CIFAR10 dataset, assigned to the class *airplane*. 19

2 Visualization of the vanilla GAN architecture. The figure shows the noise vector flowing into the generator. The generators output, as well as the real sample from P_{data}, flow into the discriminator. 20

3 Visualization of the MADGAN architecture. The figure shows the $k+1$ outputs of the discriminator and the k generators, with tied weights in the middle of the network. 25

4 Figure taken from the original paper [GKN+18]. The visualization shows how different generators G_1 and G_2 are pushed to different modes M_1 and M_2 26

17 Depiction of the CNN architecture used to classify unlabeled images from the MNIST GDA experiments and judge the effectiveness of said GDA. (Image created with [Gav20]) 1

18 Depiction of the CNN architecture used to classify unlabeled images from the Fashion-MNIST GDA experiments and judge the effectiveness of said GDA. (Image created with [Gav20]) 1

19 Depiction of the CNN architecture used to classify unlabeled images from the CIFAR10 GDA experiments and judge the effectiveness of said GDA. (Image created with [Gav20]) 1

20 Depiction of the generator used in the deep convolutional GAN dependent experiments. Used to train a generator and create fake image data based on the MNIST dataset. 2

21 Depiction of the generator used in the deep convolutional GAN dependent experiments. Used to train a generator and create fake image data based on the Fashion-MNIST dataset. 2

22 Depiction of the generator used in the Conditional GAN dependent experiments. Used to train a generator and create fake image data based on the MNIST and Fashion-MNIST datasets. 2

23 Depiction of the generators used in the MADGAN dependent experiments. Used to train a generator and create fake image data based on the MNIST and Fashion-MNIST datasets. 3

24 Depiction of the generators used in the cMADGAN dependent experiments. Used to train a generator and create fake image data based on the MNIST and Fashion-MNIST datasets. 3

25 Depiction of the discriminator used in the deep convolutional GAN dependent experiments. Used to train a generator based on the MNIST dataset. 3

26 Depiction of the discriminator used in the deep convolutional GAN dependent experiments. Used to train a generator based on the Fashion-MNIST dataset. 4

27 Depiction of the discriminator used in the Conditional GAN dependent experiments. Used to train a generator based on the MNIST and Fashion-MNIST datasets. 4

28 Depiction of the discriminator used in the MADGAN dependent experiments. Used to train a generator based on the MNIST and Fashion-MNIST datasets. 4
29 Depiction of the discriminator used in the cMADGAN dependent experiments. Used to train a generator based on the MNIST and Fashion-MNIST datasets. 4
30 A histogram chart depicting the class distribution of the generated data with the DCGAN generator trained on the **MNIST** dataset. The labels result from an auxiliary classifier as mentioned in 4.1.5. 32
31 Depiction of the concept of convolutional filtering [Mau25]. 33

List of Tables

1 Description of the used datasets for benchmarking. 35
2 FID and IS results for GAN models on MNIST, comparing single-generator (DCGAN, cGAN) and multi-generator (MADGAN, cMADGAN; K=3-10) approaches. Baseline created from training datasets. Baseline created using images from the training set. 42
3 FID and IS results for GAN models on Fashion-MNIST, comparing single-generator (DCGAN, cGAN) and multi-generator (MADGAN, cMADGAN; K=3-10) approaches. Baseline created using images from the training set. 43
4 Final F1 Scores after 50 epochs. Augmentation technique: MADGAN . 45
5 Final F1 Scores after 50 epochs. Augmentation technique: TDA 45
6 Final F1 Scores after 50 epochs. Augmentation technique: MADGAN . 47
7 Final F1 Scores after 50 epochs. Augmentation technique: TDA 47
8 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=7) 49
9 Final F1 Scores after 50 epochs. Augmentation technique: TDA 49
10 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10) 51
11 Final F1 Scores after 50 epochs. Augmentation technique: TDA 51
12 Final F1 Scores after 50 epochs. Augmentation technique: MADGAN (K=10) 54
13 Final F1 Scores after 50 epochs. Augmentation technique: cGAN . . . 54
14 Final F1 Scores after 50 epochs. Augmentation technique: MADGAN (K=10) 56
15 Final F1 Scores after 50 epochs. Augmentation technique: cGAN . . . 56
16 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=7) 58
17 Final F1 Scores after 50 epochs. Augmentation technique: cGAN . . . 58
18 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10) 60
19 Final F1 Scores after 50 epochs. Augmentation technique: cGAN . . . 60
20 Final F1 Scores after 50 epochs. Augmentation technique: MADGAN (K=10) 63
21 Final F1 Scores after 50 epochs. Augmentation technique: cMADGAN (K=5) 63
22 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10) 65
23 Final F1 Scores after 50 epochs. Augmentation tech.: cMADGAN (K=10) 65
24 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=7) 67
25 Final F1 Scores after 50 epochs. Augmentation tech.: cMADGAN (K=5) 67
26 Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10) 69
27 Final F1 Scores after 50 epochs. Augmentation tech.: cMADGAN (K=7) 69

28 Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the MADGAN model on FID and Inception Score (IS ± Std. Dev) for the **MNIST** dataset. Results for each generator index are presented alongside baseline metrics. 5
29 Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the MADGAN model on FID and Inception Score (IS ± Std. Dev) for the **Fashion-MNIST** dataset. Results for each generator index are presented alongside baseline metrics. 6
30 Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the cMADGAN model on FID and Inception Score (IS ± Std. Dev) for the **MNIST** dataset. Results for each generator index are presented alongside baseline metrics. 7
31 Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the cMADGAN model on FID and Inception Score (IS ± Std. Dev) for the **Fashion-MNIST** dataset. Results for each generator index are presented alongside baseline metrics. 8
32 FID and Inception Score (Mean ± Std. Dev) for a single DCGAN generator ($N = 1$) trained on the **MNIST** dataset. Baseline scores are included for reference. 9
33 FID and Inception Score (Mean ± Std. Dev) for a single DCGAN generator ($N = 1$) trained on the **Fashion-MNIST** dataset. Baseline scores are included for reference. 9
34 FID and Inception Score (Mean ± Std. Dev) for a single Conditional GAN generator ($N = 1$) trained on the **MNIST** dataset. Baseline scores are included for reference. 9
35 FID and Inception Score (Mean ± Std. Dev) for a single Conditional GAN generator ($N = 1$) trained on the **Fashion-MNIST** dataset. Baseline scores are included for reference. 9

1 Introduction and Motivation

Generative Adversarial Networks (GANs) [GPAM+14] and their variants revolutionized the field of computer vision in the year of 2014, enabling advacements in multiple areas of generating data. From *Text to Image Synthesis* [RAY+16], *Image Translation* [IZZE18], *Super Resolution* [LTH+17], *Image Inpainting* [PKD+16], *Style Transfer* [WWR+23] to *Data Augmentation* [SK19], GANs have been used in a variety of applications.

The idea of using GANs for *Generative Data Augmentation* (GDA) has already been applied successfully, e.g.: in computer vision [JLR25], [BNI+23] or for creating music [JLY20]. Especially the former survey *A Comprehensive Survey of Image Generation Models Based on Deep Learning* has, along *Variational Auto Encoders* (VAEs), a dedicated focus on GANs. Despite these achievements, in practice, GANs suffer from several challenges, complicating the training and inference process:

- Mode Collapse
- Lack of inter-class Diversity
- Failure to Converge
- Vanishing Gradients & Unstable Gradients
- Imbalance between Generator- and Discriminator Model

[1]

This thesis investigates the potential of using GANs — specifically *Multi-Agent Diverse Generative Adversarial Networks* (MADGANs) [GKN+18] — for Generative Data Augmentation. MADGANs aim to aid the first two of the afore mentioned in particular: Mode Collapse and Loss of inter-class Diversity. They, along other modifications, *propose to modify the objective function of the discriminator, in which, along with finding the real and the fake samples, the discriminator also has to correctly predict the generator that generated the given fake sample.* [GKN+18]. The goal of this adjustment of the discriminator is, that the discriminator has to push the generators towards distinct identifiable modes. While various strategies have been proposed to address mode collapse and inter-class diversity, MADGANs explicitly enforce mode separation by the introduction of multiple generators and the adjusted discriminator objective. This makes them particularly promising for GDA, as diverse samples and clear distinction of modes is crucial for training robust classifiers. In their paper, they experimentally show, that their architectural adjustment of GANs is generally capable of giving providing assistance for the first two of the mentioned problems.

The experiments in this work are structured into three major parts.

[1] A more detailed depiction of the problems during training of GANs are depicted in 3.3.3

Set 1: Training and Analysis of GANs The first set trains multiple variations of GANs, explicitly *Deep Convolutional GANs* (DCGANs), *Conditional GANs* (cGANs) and the afore introduced MADGANs, in addition to an adapted conditionalized version called *Conditional Multi-Agent Diverse GANs* (cMADGANs) 3.7.

Set 2: Generating and Classifying Unlabeled Images The second set uses the afore trained generative models to create images. Specifically, the goal for this stage is to generate at least 6000 images per class, in the respective datasets. Images without labels, originating from MADGANs and DCGANs, will be classified using auxiliary classifiers trained with traditional data augmentation techniques. When all samples are classified, the quality of the resulting images is scored by the *Fréchet Inception Distance* (FID) [HRU+18] and the *Inception Score* (IS) [SGZ+16].

Set 3: Training and Evaluating Classifiers The third and most important set of experiments trains classifiers using the generated and labeled samples. For this, stratified classifiers with different ratios of real to fake images are trained and evaluated on the respective validation set. These experiments are split into two scenarios: Replacement- and Expansion Scenarios 4.2. Their classification performance will be assessed using the *F1 Score*.

All of the above described is executed on the well-known benchmark datasets:

- MNIST [LCB10]
- Fashion MNIST [XRV17]

Aim of the Thesis This thesis evaluates the effectiveness of Multi-Agent Diverse GANs (MADGAN) for Generative Data Augmentation. First, the quality of generated samples is compared to those produced by a Deep Convolutional- and Conditional GAN. Further, a conditional adaptation of the MADGAN architecture, called cMADGAN, is introduced and tested alongside already established methods. The quality of the resulting images is then evaluated utilizing the *InceptionV3* model to calculate the Inception Score and FID of those images. Next, the sets of generated images are used to replace and expand training datasets for stratified classifiers. The performances of the different classifiers are assessed by the F1 score. These sets of experiments compare against traditional augmentations techniques, such as flipping, rotating and adding noise to the images, which serve as a minimum baseline.
With this experimental succession, this thesis studies the effectiveness of the MADGAN-based data augmentation for subsequent classifiers, together with its conditionalized counterpart and comparing their performances against established traditional and generative augmentation techniques.

2 Related Work

The effectiveness of deep learning models is intrinsically linked to the availability of large and diverse datasets for training. Models with deep and complex architectures require extensive exposure to a wide range of data to learn underlying patterns and generalize well to unseen instances. Insufficient training data can lead to a phenomena called *overfitting*, where a model becomes too specialized to the training data, failing to perform accurately on previously unencountered data [Yin19].

Data augmentation artificially expands the amounts and diversity of training datasets by creating modified versions of existing data or by generating entirely new instances. To mitigate the problem of data scarcity and improve generalization capabilities of deep learning models, data augmentation techniques became indispensable.

Traditional Data Augmentation

Traditional data augmentation on images typically involves applying various transformations to existing data. For image based data, augmentations can take a variety of forms such as [2] : *Geometric Augmentation*, *Photometric Augmentation*, *Noise-Corruption Augmentation* 3.2.1.

The sucess of the above mentioned augmentation techniques is established in many papers [PW17], [KSH12a], [Yin19], [SK19], [WZZ+13].

Generative Data Augmentation using Deep Convolutional GANs

The basic GAN framework introduced by Goodfellow and colleagues offers a high degree of flexibility and can be adapted for specific augmentation tasks. It can be applied to generate music [DHYY17], speech [LMWN22], text [YZWY17], images [GPAM+14] or other instances of data, e.g. tabular data [XSCIV19].

Especially for image data, *Deep Convolutional GANs* (DCGANs) [RMC16] represent a significant advacement in applying GANs to image data augmentation [HFM22]. Their architecture specifically utilizes *Convolutional Neural Network* (CNNs) [LBD+89] in both, the generator and the discriminator. The use of CNNs allow DCGANs to learn hierarchical features from the input images effectively and capture the spatial relationship and structure inherent in the training data. This leads to the generation of more realistic and coherent synthetic images. A study from Zhau et al. [ZCWD23] applied DCGANs, along their adjusted versions on multiple dataset, including *Fashion MNIST* and *Cifar10*. With their experimental setup, they achieved consistent significant improvements over multiple datasets using the DCGAN-architechture, compared to their baseline.

[2] More categories of traditional data augmentation techniques exists, such as Occlusion-Based, Composition-Based, Domain-Specific or Adversarial Augmentation. For the purpose of this work, soley the afore mentioned are discussed in greater detail.

Inherently in the vanilla version of GANs or the DCGANs realization of using convolutional layers, the generator's role is solely to learn the underlying data distribution of the training samples and produce instances of close resemblance to instances from the training data. This, however, results in unlabeled samples, not beneficial to expand data for a supervised classification task out of the box.

Generative Data Augmentation using Conditional GANs

The introduction of *Conditional Generative Adversarial Networks* (cGANs) [MO14] allows to condition the generative process by additional information, such as class labels or other modalities. The conditioning acts on both the generator and the discriminator, which means that both models have access to the same conditional information. The generator combines the random vector input and the conditioning information into a joint hidden representation. The discriminator, on the other hand, evaluates the created data from generator, given context of the conditioning information, i.e., the class label passed. This approach enables the generator to create data that adheres to specific inputs, like creating specific digits from the MNIST dataset 1. Multiple papers were able to utilize the advantages of cGANs to, e.g., unify class distributions for a stratified classifier training or generatively increase the number of images and augmenting the training data[JPB22][ZCWD23][RCF25][WM21].

Generative Data Augmentation using MADGANs

Regardless of the mentioned successes using GANs (DCGANs or cGANs) for GDA 2 2, GANs in general have proven to be notoriously hard to train. *Among them, mode collapse stands out as one of the most daunting ones.* [DCLK20], which limits the GANs ability to generate diverse samples, able to be assigned to all classes trained on. Another prominent problem with GANs is the lack of inter-class diversity between generated samples.

MADGANs [GKN+18] emphasis on diversity, achieved through its multi-agent architecture and the modified discriminator objective function, directly addresses these limitations. By encouraging multiple generators to specialize in different modes of the data distribution, MADGAN aims to generate a more comprehensive and diverse set of synthetic samples compared to traditional GANs and potentially other generative data augmentation techniques that might be susceptible to mode collapse. The ability of MADGAN to disentangle different modalities i.e., classes, as suggested by experiments involving diverse-class datasets, indicates its potential to generate augmented data that effectively covers both intra-class and inter-class variations. This comprehensive coverage is crucial for training robust image classifiers that can generalize well to a wide range of real-world scenarios.

3 Theoretical Background

This chapter serves as a reference for the theoretical background necessary to understand the insights gained in the following experimental chapters. Section 3.1 discusses classification models used to train on the extended dataset resulting from the generative augmentation process. In it, *Neural Networks* (NNs) for image classification are introduced and the baselines for later comparisons are examined. Sections 3.2 and 3.2.2 establish the foundation for data augmentation and generative data augmentation. Following sections 3.3, 3.4, 3.5, and 3.6 provide theoretical knowledge necessary to understand the GAN architectures and their differences. The narrative follows their increasing complexity, starting from vanilla GANs, moving through deep convolutional GANs and conditional GANs, before diving into the background of multi-agent diverse GANs. The final section (3.8) explains the theory behind the Inception Score and Fréchet Inception Distance, concluding with an examination of the state-of-the-art *InceptionV3* model used to compute them.

3.1 Image Classification Models

3.1.1 Neural Networks for Classification

Convolutional Neural Networks (CNNs) have become the dominant architecture for image classification tasks due to their inherent ability to automatically learn hierarchical features from raw pixel data. At their core, CNNs are build up by a sequence of convolutional-, pooling- and fully connected layers to extract hierarchical features, and funneling the information, typically into the N classes defined by the training data. That is, in case of classification tasks.

Convolutional layers employ learnable filters to detect local patterns in the two-dimensional information. The two-dimensionality is inherent to image data. Pooling layers reduce the spatial dimensions to small translations. Fully connected layers then map the extracted information into class probabilities, utilizing the *Softmax* activation function. The afore mentioned layers are discussed in greater detail, in the following subsections.

Convolutional Layers These layers compute the output from the local regions of the input. Let $(r \times c)$ be the two-dimensional input, e.g., a grayscale image, where r represents the X-coordinate and c the Y-coordinate of a pixel. Thus, $r \cdot c$ denotes the size of the image. Furthermore, let $(a \times b)$ be a filter with kernel size $a \cdot b$, where the filter is smaller than the input. This filter is moved from the top-left to the bottom-right over the input.

In each iteration, the dot product between the respective coefficients of the input region and the coefficients of the filter is computed. This dot product is then processed

by the activation function g, which determines how much of the feature is present. If the activation function is ReLU, for example, only positive values are retained, meaning negative responses are set to zero. The result are written to the subsequent layer.

The stride determines how far the filter moves after each operation. For a stride of $s = 1$, the filter can be placed in $(r - a + 1)$ positions along the height and $(c - b + 1)$ positions along the width, leading to an output size of $(r - a + 1) \times (c - b + 1)$. In general, the output size in the two-dimensional case is given by:

$$\left(\frac{r - a + s}{s}\right) \times \left(\frac{c - b + s}{s}\right).$$

An image of this process can be found in Figure 31, in the Appendix. This image shows an input of size $[10 \times 10]$ and a filter of size $[3 \times 3]$. With a stride of $s = 1$, the resulting layer has a size of $[8 \times 8]$ (computed as $(10 - 3 + 1) \times (10 - 3 + 1)$).

The stride of the filter can also be greater than 1. Additionally, there is the option to apply padding to the image. There are different ways to implement padding. When padding of size p is applied, the output size for a square input and filter is calculated as follows:

$$\textbf{Output size} = \left\lfloor \frac{r - a + 2p}{s} \right\rfloor + 1$$

where $\lfloor \cdot \rfloor$ denotes the floor function, which ensures that the output size is an integer. A well known source for the context of convolutional arithmetic is the paper *A guide to convolutional arithmetic for deep learning*, by Dumoulin et al. 2018 [DV18].

Pooling Layers A pooling layer compress data along its spatial axes, removing dimensionality. Similar to convolutional layers, pooling layers apply a filter operation, that moves by a given stride. Instead of summing elements covered by the filter, typically, the filter applies one of the following operations: Max-, Average-, Global Max- or Global Average Pooling. [3]

To give an example, starting with an input of size $[32 \times 32 \times 10]$, applying a pooling operation with a $[2 \times 2]$ filter and a stride of 2 results in an output size of $[16 \times 16 \times 10]$. This operation reduces the spatial dimensions by half while keeping the depth unchanged.

Fully-Connected Layers Fully-Connected (FC), also called *Dense* layer, typically computes the scores for the respective classes. In the case of ten classes, the result is a volume of size $[1 \times 1 \times 10]$[4]. By this stage, all spatial information has been transformed, leaving a quasi-one-dimensional vector.

[3] Other types of pooling involve e.g., L2-norm-, stochastic- or learnable pooling. However, these are less common than the afore mentioned types.

[4] Typically, the output from the layer before the FC one is *flattened* into a one-dimensional vector, preserving all information but removing spatial structure. For example, a $[2 \times 2]$ layer would become a vector of size $[1 \times 4]$.

In a FC layer, each input is connected to each output, meaning every neuron in the previous layer is connected to each neuron in the FC layer. The output is the weighted sum of all inputs, followed by an activation function, leading to the final classification scores that represent the likelihood of the input belonging to each class. The spatial dimensions are collapsed into a single vector of class scores, which are then used for classification. The class probabilities are a result of applying the Soft-Max Function 3.1.1.

Batch Normalization Layers With their introduction in *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* by Ioffe et al. [IS15], Batch Normalization (batchnorm) established as an integral part of convolutional networks and neural networks in general. This type of layer normalizes its inputs to subsequent layers and thereby stabilizing the distribution of activations throughout the training process. This reduces the internal covariate shift, allowing for higher learning rates and faster convergence. By normalizing activations, batchnorm helps prevent gradients from vanishing or exploding. However, it does not prevent it entirely. Additionally, it can provide regularization benefits and eliminate the need for Dropout, in some cases. With the afore mentioned benefits, batchnorm layers are particularly beneficial for deep learning networks with many layers.

Typical Activation Functions in CNNs

- **ReLU (Rectified Linear Unit)**:

$$g(x) = \max(0, x) \tag{1}$$

  ReLU is the most widely used activation function in CNNs. It introduces non-linearity while maintaining efficiency by outputting zero for negative values and passing positive values unchanged.

- **Leaky ReLU**:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases} \tag{2}$$

  A variant of ReLU, Leaky ReLU allows small negative values to flow through, addressing the *dying ReLU* problem where neurons can become inactive [LLYSYSGEK20].

- **Sigmoid (Logistic)**:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

  The sigmoid function squashes values between 0 and 1, commonly used for binary classification tasks. However, it can suffer from vanishing gradients for very large or small inputs.

- **Tanh (Hyperbolic Tangent)**:

$$g(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{4}$$

Tanh outputs values between -1 and 1 and is similar to the sigmoid, but with a wider output range, making it more effective in many scenarios compared to sigmoid.

- **Softmax**:

$$g(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5}$$

Softmax is typically used in the output layer of CNNs for multi-class classification. It converts logits into probabilities, ensuring that the sum of the outputs is 1.

- **Cross-Entropy Loss:**

$$g(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{6}$$

Cross-entropy is a common loss function for classification tasks. It measures the dissimilarity between the true label distribution y and the predicted probability distribution $\hat{y}$. Lower values indicate a better alignment between the predicted and true classes. For consistency, the cross-entropy function is denoted with g here, although it is more commonly represented in the literature as $H(y, \hat{y})$ or $H(p, q)$, where p refers to the true label distribution and q to the output of the discriminative model.

3.1.2 Classification Models for augmented Training

The classification models, on which the data augmentation is tested, are simple CNN classifiers consisting of the described layers. For each dataset MNIST, Fashion-MNIST 1, a dedicated classifier architecture is created. The main differences between these architectures is the number of blocks, made of two-dimensional convolutional, batchnorm and pooling layers. All models use the ReLU function for activation of the convolutional layers and the Softmax function for the activation at their respective output layer, resulting in probability distribution over the space of classes. More on the specific model architectures and used metrics for evaluation can be found in chapter 4.1.

3.1.3 Classification Model Performance Metrics

For the assessment of classification performance, several standard metrics are commonly used. *Accuracy* gives a measure for the overall proportion of correctly predicted instances, but can be misleading in imbalanced datasets. *Precision* quantifies the proportion of correctly predicted positive instances among all predicted instances, whereas

recall [5] captures the proportion of correctly predicted positives out of all actual positive samples.

The *F1 score* is the harmonic mean of precision and recall and serves as a balanced metric when both false positives and false negative are of concern. This is especially useful for imbalanced datasets. Mathematically, the formulation of the F1 score is as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

Even when working with stratified data, where class distributions are balanced, the F1 score remains valuable by providing a single, informative measure that accounts for the trade-off between precision and recall. Due to its sensitivity to both types of classification errors, the F1 score is used as the primary evaluation metric throughout this work, in the context of classifications.

3.2 Data Augmentation

In this chapter, data augmentation (DA) techniques in the context of images are discussed is greater detail. Starting with traditional augmentations e.g., rotating or cropping an image, ending on generative augmentations for which generative models are used to expand the training data of subsequent classification models.

3.2.1 Traditional Data Augmentation

The need for data augmentation to make classification algorithms more resilient has existed for decades. Early papers mentioning the augmentation of data for classification tasks date back to the 1960s [NS67] (1966). For the context of deep learning, however, the augmentation of images was popularized by Krizhevsky et al. in 2012 [KSH12b]. This paper introduced the *AlexNet*, a deep CNN used to classify images from the *ImageNet* dataset [DDS+09], containing 1000 classes. This paper also referenced the earlier work of Simard et al. from 2003 [SSP03].

Generally, traditional data augmentation (TDA) techniques can be described as applying transformations to the initial training data. These transformations preserve the respective labels y and only act on the data X. [6] Applied transformations could either be used to enlarge the training data or to alter existing data i.e., altering the variability

[5] Recall is also known as sensitivity.

[6] It is to point out, that there are also transformations to be applied to the labels. This can include operations like random flipping of the labels. Such alterations also include techniques such as smoothing the labels, e.g., adjusting labels to not be strictly 0 or 1, but rather 0.1 or 0.9.

in it.

Augmentations are categorized based on the type of transformations applied:

- *Geometric Augmentation*: This category modifies the shape, position, and perspective: Rotation, Scaling, Flipping, Cropping, Shearing, Perspective Transform.
- *Photometric Augmentation*: Alters pixel values while keeping the spatial structure: Brightness, Contrast, Hue Shift, Blurring
- *Noise-Corruption Augmentation*: Imitates real-world degradations and distortions caused by cameras and sensors: Gaussian Noise, Speckle Noise, Salt-and-Pepper Noise.

Mathematically, let X be an original data sample drawn from the dataset distribution $P(X)$. Traditional data augmentation applies a transformation function $f : X \mapsto \tilde{X}$, where f is a function sampled from a predefined set of augmentation operations $\mathcal{F}$. The augmented data sample $\tilde{X}$ is then given by:

$$\tilde{X} = f(X), \quad f \sim \mathcal{F}.$$

Since TDA modifies already existing samples, the distribution of augmented samples $P_{\tilde{X}}$ should ideally remain close to the original data distribution:

$$P_{\tilde{X}}(X) \approx P(X).$$

In the context of data augmentation pipelines, this can be generalized as:

$$\text{TDA} : (X, f) \mapsto \tilde{X}, \quad f \in \mathcal{F}.$$

When applying the augmentations shown in Figure 1, it is mandatory to consider domain-specific knowledge and constraints. For example, flipping images from the MNIST dataset to train a generative model may result in an image where a horizontally flipped *9* appears. This is in the domain of Arabic numerals semantically incorrect. Further, flipping a 9 along horizontal and vertical axes would result in a 6. Conversely, when classifying on an airplane, which can have varying shapes, colors, three-dimensional orientations in space, and images taken through a dusted lens, applying all of the above augmentations could be beneficial.

3.2.2 Generative Data Augmentation

Differing from the previously mentioned TDA 3.2.1, generative data augmentation (GDA) techniques do not focus on altering existing data instances. Rather, it focuses

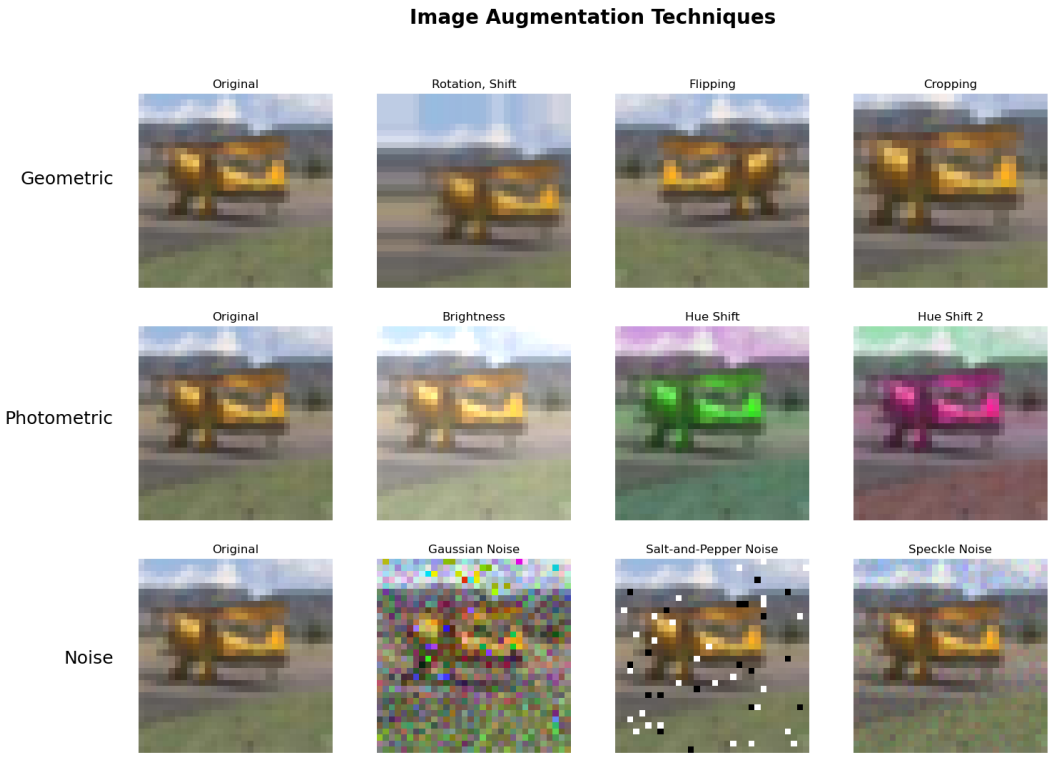


Figure 1: Exemplary use of traditional augmentation techniques from the categories *geometric* (first row), *photometric* (second row), and *noise-corruption* (third row). The image shown is an image from the CIFAR10 dataset, assigned to the class *airplane*.

on creating entirely new samples that match the underlying data distribution of the training data. These generated instances may or may not include labels.

The goal is to train a generative model G that produces instances X_1, for example, from a noise vector z [7], such that the distribution of the generated data approximates the true distribution $P(X)$ of the original dataset. In this context, G can be viewed as a function:

$$G : z \mapsto X_1, \quad X_1 \sim P_G(X_{fake}) \approx P(X),$$

where $P_G(X)$ is the learned distribution of the generative model, aiming to approximate the real data distribution $P(X)$.

In the case of *conditional* generative data augmentation, additional information such as class labels y is incorporated into the generation process. This allows the model to generate samples corresponding to specific categories within the data. The conditional generative model G then follows:

$$G : (z, y) \mapsto X_{fake}, \quad X_{fake} \sim P_G(X_{fake} \mid y) \approx P(X \mid y),$$

where $P_G(Xfake \mid y)$ represents the learned conditional distribution, aiming to approx-

[7] A noise vector may serve as input for a Generative Adversarial Network. See: 3.3

imate the real class-conditioned data distribution $P(X \mid y)$. This can enable targeted data generation for specific categories, enhancing data diversity while maintaining class consistency.

3.3 Generative Adversarial Network

Generative Adversarial Network (GANs) have first been introduced by Goodfellow et al. in 2014 [GPAM+14] GANs are a type of generative models designed to learn the underlying data distribution of their training data and generate new, realistic instances. The core idea of the framework is an adversarial training process between two neural networks (NNs): the *Generator* G and *Discriminator* D, are competing against one another in a minimax game [Neu28]. The following figure (2) shows a visualization of the vanilla GAN architecture.

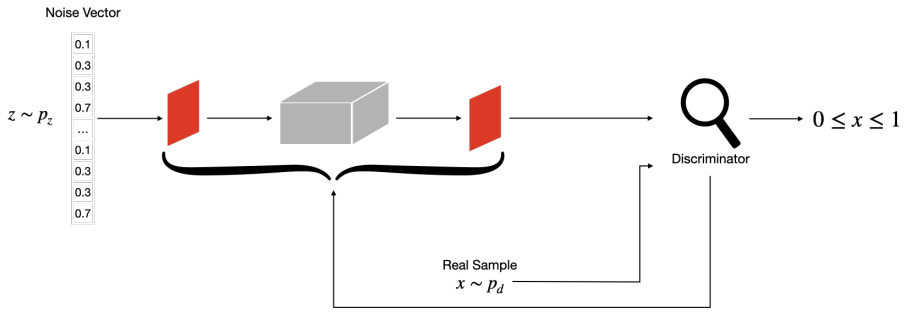


Figure 2: Visualization of the vanilla GAN architecture. The figure shows the noise vector flowing into the generator. The generators output, as well as the real sample from P_{data}, flow into the discriminator.

3.3.1 Mathematical Formulation

Let $X \sim P_{data}$ be samples drawn from the real data distribution, and let $z \sim P_z$ be random noise sampled from a known prior (e.g., a Gaussian or uniform distribution). The generator G is a function $G : \mathbb{R}^d \to \mathbb{R}^n$ that maps a noise vector z to a synthetic data instance $\tilde{X}$, attempting to approximate P_{data}:$\tilde{X} = G(z), \quad z \sim P_z$.

The discriminator D is a function $D : \mathbb{R}^n \to [0, 1]$ that outputs the probability that a given sample is real or generated. It is trained to distinguish between real samples $X \sim P_{data}$ and generated samples $\tilde{X} \sim P_G$, where P_G is the implicit distribution induced by G.

The training objective is formulated as the following minimax game:

$$\mathcal{L}_{\text{adv}} = \min_G \max_D \mathbb{E}_{X \sim P_{data}}[\log D(X)] + \mathbb{E}_{z \sim P_z}[\log(1 - D(G(z)))]. \tag{9}$$

Here, the discriminator D aims to maximize the probability of correctly differentiating real and fake samples $\hat{x}$, while the generator G aims to generate samples that fool D, minimizing $\log(1 - D(G(z)))$. In an ideal scenario, the game converges to a Nash equilibrium where G produces samples indistinguishable from real data, i.e., $P_G \approx P_{data}$. In this scenario, the discriminators output oscillates around 0.5, continuously unable to differentiate real and fake samples.

3.3.2 Training Process

GAN training follows an alternating optimization approach. Typically, alternating training the generator and the discriminator:

1. Update D: Given a batch of real samples from P_{data} and fake samples generated by G, update D to maximize its ability to discriminate real from fake data.
2. Update G: Generate new fake samples and update G to minimize $\log(1 - D(G(z)))$, effectively pushing G to generate more realistic samples.
3. Repeat the process iteratively, typically using stochastic gradient descent (SGD) or Adam optimization.

3.3.3 Challenges in GAN Training

Following, challenges that can occur during the training of GANs are discussed. These have already been mentioned in the introductory section 1. Following, they are described in greater detail.

Mode Collapse Mode collapse occurs when the generator produces only a small subset of the data distribution, leading to a lack of diversity. Instead of generating varied diverse samples, it repeatedly produces similar ones that fool the discriminator. This happens when the generator finds an easy *shortcut* rather than learning the full distribution over all classes. More formally, G collapses many values of z to the same value of x [GPAM+14]. A common technique to mitigate this issue is minibatch discrimination [SGZ+16]. In several studies, experiments have been conducted to enhance diversity of GANs ([CFB+24], [HBB21], [HBB22])

Lack of Inter-Class Diversity Even if mode collapse is avoided, GANs may struggle to generate samples that represent all data classes equally. This is a common issue in class-conditional GANs, where samples across different classes may overlap or lack distinct features. Causes include imbalanced datasets, poor class conditioning, or weak discriminator feedback [OOS17].

Failure to Converge Unlike traditional neural networks, GANs follow an adversarial training process, making optimization highly unstable. The loss functions of both the generator and discriminator change dynamically, often leading to non-convergent behavior. Methods like Wasserstein GANs (WGAN) [ACB17] and spectral normalization [MKKY18] improve stability and help achieve better convergence.

Vanishing & unstable Gradients When the discriminator becomes too strong, it perfectly distinguishes real from fake samples, leading to vanishing gradients for the generator. This prevents meaningful updates and stalling progress. On the other hand, unstable gradients cause erratic updates, preventing smooth learning. Alternative loss functions (e.g., LSGANs [MLX+17]) and spectral normalization can help stabilize the training.

Imbalance between Generator and Discriminator A well-balanced GAN requires both models to improve at a similar pace. If the discriminator overpowers the generator, gradient updates to the generator can vanish. If it's too weak, the generator receives poor feedback and produces low-quality outputs [GPAM+14]. Balancing techniques include adaptive learning rates, gradient penalties, and label smoothing [RMC16]. Another interesting alternative, to tackle the problem of imbalance is introduced in the paper *Progressive Growing of GANs for Improved Quality, Stability, and Variation* [KALL18].

3.4 Deep Convolutional Generative Adversarial Network

Deep Convolutional Generative Adversarial Networks (DCGANs) were introduced by Radford et al. in 2015 [RMC16] as an improvement over vanilla GANs. While the fundamental adversarial framework remains the same (see 3.3 Mathematical Formulation, 3.3 Training Process), DCGANs leverage deep convolutional neural networks to enhance stability and generate higher-quality images.

3.4.1 Architectural Adjustments

To improve training stability and image quality, DCGANs implement the use of convolutional layers.

- **Convolutional Architecture:** Fully connected layers in both G and D are replaced with deep convolutional layers, enabling better spatial feature extraction.
- **Strided Convolutions:** In the discriminator, pooling layers are removed in favor of strided convolutions, reducing the risk of information loss.

- **Transposed Convolutions:** The generator employs transposed convolutions (also known as fractionally-strided convolutions) instead of upsampling layers to improve the quality of generated images.
- **Batch Normalization:** Applied to both G and D, batchnorm helps stabilize training by reducing internal covariate shift 3.1.1. Batchnorm is omitted in the generators final layer to allow unrestricted output variability and in the discriminators input layer to preserve the original data distribution.
- **LeakyReLU Activation:** The discriminator uses LeakyReLU instead of standard ReLU to prevent dying neurons and allow gradients to flow through negative inputs 3.1.1.
- **No Fully Connected Layers:** Fully connected layers are removed to maintain spatial coherence in generated images, as they discard spatial information by flattening feature maps. Instead, convolutional layers preserve local structures, enabling more realistic image synthesis 3.1.1.

3.5 Conditional Generative Adversarial Network

Conditional Generative Adversarial Networks (cGANs), introduced by Mirza and Osindero in 2014 [MO14], extend the vanilla GAN framework by incorporating additional information y, such as class labels, into both the generator and discriminator. This allows cGANs to generate samples conditioned on specific attributes, enabling controlled generation.

3.5.1 Mathematical Formulation

The core idea of cGANs is to condition both the generator G and the discriminator D on auxiliary information y. Instead of generating data solely from a noise vector z, the generator now takes y as an additional input:

$$\tilde{X} = G(z, y) \tag{10}$$

Similarly, the discriminator receives both the real or generated sample and the corresponding condition:

$$D(X, y) \quad \text{and} \quad D(G(z, y), y) \tag{11}$$

The adversarial objective function for cGANs extends the standard GAN loss to incorporate this conditional dependency:

$$\mathcal{L}_{\text{adv}} = \min_G \max_D \mathbb{E}_{X,y\sim P_{data}}[\log D(X, y)] + \mathbb{E}_{z\sim P_z, y\sim P_y}[\log(1 - D(G(z, y), y))] \tag{12}$$

This objective encourages the generator to produce samples that are not only realistic but also consistent with the provided condition, while the discriminator is trained to discern real from synthetic data based on their respective conditions.

3.5.2 Architectural Adjustments

To implement cGANs, architectural adjustments are necessary compared to vanilla GANs:

- **Input Conditioning:** Both the generator and discriminator must receive the conditional information y as input. This is typically achieved by concatenating the condition y with the noise vector z in the generator's input and with the input image X in the discriminators input.
- **Embedding Conditional Information:** For categorical conditions e.g., class labels, the condition y is often embedded into a lower-dimensional vector before concatenation. This embedding allows the network to learn meaningful representations of the conditions.
- **Concatenation, Addition, or Multiplication:** The conditional information can be incorporated through concatenation, addition, or element-wise multiplication at various layers within the generator and discriminator, depending on the specific application and architecture.
- **Preservation of Conditional Information:** Care must be taken, that the conditional information is preserved throughout the network. This means, that the information must traverse the network, all the way to the output layer.

These architectural modifications ensure that the generator and discriminator can effectively utilize the conditional information to generate and discriminate samples based on the given conditions.

3.6 Multi-Agent Diverse Generative Adversarial Network

MADGAN is proposed as a generalized framework for the GAN architecture [GKN+18]. The framework employs multiple generators, one discriminator and an adjusted objective for the discriminator. The adjusted objective aims to enforce the identifications of the generator creating given fake images $G_i\hat{x}$. These changes specifically aim to ease the first two mentioned problems of GANs, namely *Mode Collapse* and *Lack of Inter-Class Diversity* 3.3.3. This chapter delves into the specifics of this framework: integration of multi-agent systems with diversity-promoting techniques, within the GAN framework. The following subsections will detail the architecture, objective function, and training procedure of the MADGAN framework.

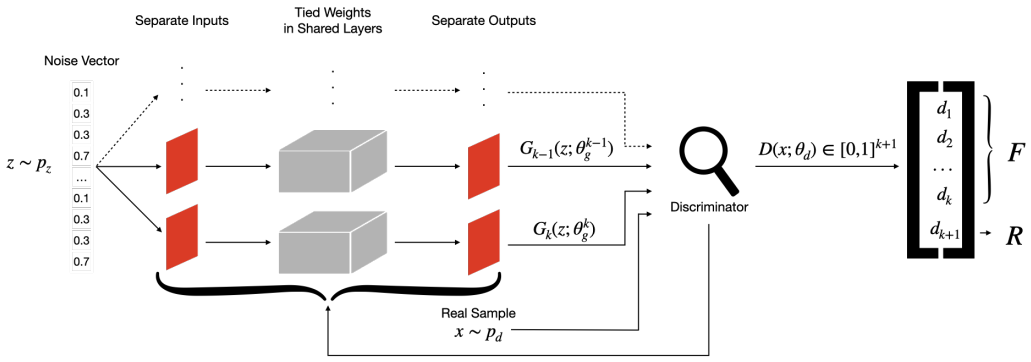


Figure 3: Visualization of the MADGAN architecture. The figure shows the $k+1$ outputs of the discriminator and the k generators, with tied weights in the middle of the network.

3.6.1 Mathematical Formulation

As afore mentioned, the MADGAN architecture employs multiple generators and one discriminator. The goal for the K generators is to generate samples from different high probability regions of the data P_{data}. In order to guide the generators into their respective direction, the objective of the discriminator has been modified. The objective no longer only has to differentiate between real and fake images, but also identify the generator that produced a given fake sample. Intuitively, the discriminator thereby forces the generators into mostly disjoint regions within the real data distribution. Inspired by the formulation for the discriminator in the paper *Improved Techniques for Training GANs* [SGZ+16], their discriminator model has an output of $k+1$, utilizing the *Softmax* activation function (3.1.1), where k sets the number of used generators. The output at $k+1$ represents the probability that the given samples belongs to the real data distribution P_{data}, whereas the scores at $j \in \{1, ..., k\}$ describe the probability of said sample to originate from either of the generators. The above figure 3 shows a visualization of the architecture. Thus, when learning the discriminators parameters θ_d, the cross-entropy between the softmax outputs and the *Dirac delta distribution* (Ddd) $\delta \in \{0,1\}^{k+1}$ is optimized. Here, $\delta(j) = 1$, if the sample belongs to the j-th generator, else $\delta(k+1) = 1$ if the sample originates from P_{data}. Following this definition, the Ddd δ can be understood as a one-hot encoding indicating whether a given sample is real of, and if not, from which generator it originates [8]. The objective function for the optimization of θ_d, with θ_g frozen is therefore:

$$\max_{\theta_d} \mathbb{E}_{x \sim p} H(\delta, D(x; \theta_d)) \tag{13}$$

[8] It is important to point out, that the Dirac delta distribution is actually continuous. Their usage of the Ddd reminds of the Kronecker delta function, which $\delta(i,j) = 1$ for $i = j; 0$ for $i \neq j$.

.

$H(...)$ here represents the negative cross-entropy function. Important to point out here is that, *Intuitively, in order to correctly identify the generator that produced a given fake sample, the discriminator must learn to push different generators towards different identifiable modes.* [GKN+18], page 4. Figure 2 shows a visualization of the generators being pushed to different modes. The objective for the generators however, remains semantically the same as in the vanilla GAN 9. The difference here is that the objective function is generalized with an indexing i for the number of generators. That is, for the i-th generator, the objective function is to minimize:

$$\mathcal{L}_{Gen_i} = \mathbf{E}_{x\sim P_{data}}[\log D(X;\theta_d)] + \mathbf{E}_{z\sim P_z}\log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_x)) \tag{14}$$

Enforcing Diverse Modes: The researchers provide a theorem demonstrating that the k generators collectively form a mixture model. In it, each generator represents a mixture component to the global optimum of $-(k+1)\log(k+1) + 1\log k$. This is achieved, when $p_{data} = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$. Significant to note is that, for $k = 1$, the case with one generator, one obtains the exact Jensen-Shannon divergence based objective, as shown in the vanilla GAN paper by Goodfellow et al. [GPAM+14]. Following their proof, the objective function for all k generators is to minimize:

$$\mathcal{L}_{Gen_{[0...K]}} = \mathbf{E}_{x\sim P_{data}}[\log D_{k+1}(X)] + \sum_{i=1}^{k}\mathbf{E}_{x\sim p_{g_i}}\log(1 - D_{k+1}(x)) \tag{15}$$

The exact formulation of their theorem, proof, and propositions can be found in Ghosh et al. [GKN+18], following formulas (3) through (9).

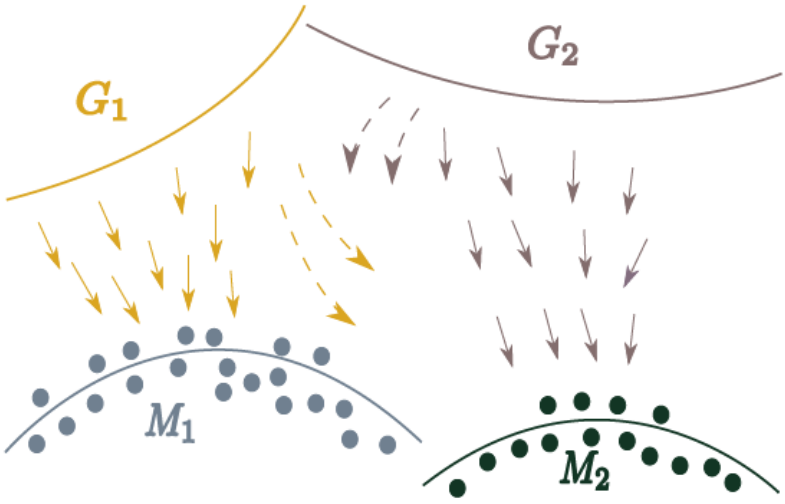


Figure 4: Figure taken from the original paper [GKN+18]. The visualization shows how different generators G_1 and G_2 are pushed to different modes M_1 and M_2

3.6.2 Architectural Adjustments

The architecture changes can be summarized into three mayor changes applied to the vanilla GAN (3.3) architecture:

Multiple Generators: The MADGAN architecture employs multiple generators instead of a single one. Following the intuition behind the adjusted discriminator objective, the respective GANs distribute into different, mostly disjoint regions in the real data distribution.

Modified Discriminator Objective: The discriminator in this architecture outputs $k+1$ instead of 1 utilizing the softmax function 3.1.1. One output for every generator indicating whether an image originates from either one of the generator and one output implying if the discriminator came to the conclusion that the current image is fake or not. As afore described, this encourages the discriminator to push the generators into identifiable distinct outputs.

Parameter Sharing among Generators: Generators in the MADGAN architecture share most of their layers between the k generators. This reduces redundant compute in the initial feature extraction layers capturing high-frequency structures common in many datasets. Especially in single-view data, this is recommended. For multi-view data it is not recommended, to allow each generator to capture mode-specific features.

3.7 Adapting MADGAN for Conditional Generation with Explicit Diversity

The finial goal of this research includes exploring methods for generating diverse samples in the mentioned datasets 1 to be used for data augmentations. While the multi-agent framework by Ghosh et al. presents a potential approach, experiments revealed significant challenges in achieving stable training and satisfactory results in terms of image quality with the original method. Especially the CIFAR10 datasets showed to be challenging, which is the only colored dataset used.

Consequently, an alternative framework, termed Conditional Multi-Agent Diverse GAN (cMADGAN), was conceptualized. The adjustment aims to retain the multi-generator diversity principle while simplifying the task of the discriminator. The discriminator model is rolled back to its original objective to differentiate between real and fake images. However, initial experiments indicated that this approach also faced convergence difficulties when applied to the original CIFAR10 dataset. This specific is explained in more detail in the experiments section 4.1.1.

The following sections detail the cMADGAN framework as implemented. This approach utilizes multiple generators $(G_1, \ldots, G_K)$, a single conditional discriminator (D) performing standard real/fake classification, and incorporates an explicit diversity loss between generator outputs alongside the adversarial objective.

3.7.1 Mathematical Formulation

Let z be a latent vector sampled from a prior distribution $p(z)$, typically a standard normal distribution. Let c be the condition variable (e.g., class label) sampled from a distribution $p(c)$. The framework also employs K generator networks $G_1, \ldots, G_K$ and a single discriminator network D. Each generator G_i maps an input pair (z, c) to the data space, producing a fake sample $\hat{x}_i = G_i(z, c)$. The discriminator D takes an input pair (x, c) (where x can be a real sample from the true data distribution $p_{\text{data}}(x|c)$ or a fake sample $\hat{x}_i$) and outputs a scalar probability estimating the likelihood that x is real given the condition c.

The training involves optimizing two competing objectives, $\mathcal{L}_D$ for the discriminator and $\mathcal{L}_G$ for the generators:

Discriminator Loss: The discriminator is trained to distinguish real samples from fake samples generated by *any* of the K generators, given the condition c. Using the binary cross-entropy (BCE) loss, the objective is:

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{x\sim p_{\text{data}}(x|c), c\sim p(c)}[\log D(x, c)] \\ &- \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{z\sim p(z), c\sim p(c)}[\log(1 - D(G_i(z, c), c))]\end{aligned} \tag{16}$$

In practice, techniques like label smoothing are applied to stabilize training. In its current implementation, it is set to 0.9 for real and 0.1 for fake samples.

Generator Loss: The generators are trained collectively to fool the discriminator and simultaneously produce diverse outputs. The combined loss function includes an adversarial term and an explicit diversity term:

$$\mathcal{L}_G = \mathcal{L}_{\text{adv}} - \lambda_{\text{div}}\mathcal{L}_{\text{sim}} \tag{17}$$

where λ_{div} is a hyperparameter controlling the weight of the diversity component.

The components are defined as:

- **Adversarial Loss ($\mathcal{L}_{\text{adv}}$):** Encourages generators to produce samples that the discriminator classifies as real. The non-saturating formulation based on maximizing the probability of fake samples being classified as real is commonly used:

$$\mathcal{L}_{\text{adv}} = -\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{z\sim p(z), c\sim p(c)}[\log D(G_i(z, c), c)] \tag{18}$$

- **Similarity Loss ($\mathcal{L}_{\text{sim}}$):** This term quantifies the similarity between outputs of different generator pairs for the same input (z, c). The goal during optimization ($\min \mathcal{L}_G$) is to maximize diversity, which corresponds to minimizing the similarity

term $\mathcal{L}_{\text{sim}}$ when $\lambda_{\text{div}} > 0$ due to the negative sign in Eq. 17. Cosine similarity between the flattened outputs is used:

$$\text{CosSim}(a, b) = \frac{\text{vec}(a) \cdot \text{vec}(b)}{\|\text{vec}(a)\|_2 \|\text{vec}(b)\|_2} \tag{19}$$

where $\text{vec}(\cdot)$ denotes flattening the image tensor into a vector. The similarity loss is the average pairwise cosine similarity:

$$\mathcal{L}_{\text{sim}} = \frac{1}{N_p} \sum_{i=1}^{K} \sum_{j=i+1}^{K} \mathbb{E}_{z \sim p(z), c \sim p(c)} [\text{CosSim}(G_i(z, c), G_j(z, c))] \tag{20}$$

where $N_p = K(K-1)/2$ is the number of unique generator pairs. Minimizing the combined $\mathcal{L}_G$ in Eq. 17 thus encourages fooling the discriminator (minimizing $\mathcal{L}_{\text{adv}}$) while discouraging similarity between generator outputs (by minimizing $-\mathcal{L}_{\text{sim}}$).

3.7.2 Architectural Adjustments

The cMADGAN architecture makes specific adjustments compared to both standard conditional GANs and the original MADGAN:

- **Multiple Generators:** As the original MADGAN framework proposed, the cMADGAN employs K distinct generator networks. As suggested in the original paper, the generators do not share their weights, due to the selection of datasets (cf. section 4.1 [GKN+18]). Each generator takes both the latent vector z and the condition c as input.
- **Conditional Discriminator:** Unlike the original MADGAN discriminator, the task of the cMADGAN discriminator performs the standard conditional GAN task 3.5.2. D receives an image x (real or fake) and its corresponding condition c, and outputs a single probability indicating whether the image is real given the condition.
- **Conditioning Mechanism:** Class conditioning c is integrated into both generator and discriminator networks. There are multiple different ways to encode information 3.5.2.
  - In the generators, this embedding is concatenated with the latent vector z before being processed by the main network layers (e.g., transposed convolutions).
  - In the discriminator, the condition embedding might be concatenated with intermediate feature maps or projected via learned linear layers and combined with image features before the final classification output, following common practices for conditional discriminators [MO14].

- **Explicit Diversity Enforcement:** Diversity is not solely an emergent property of competition but is explicitly encouraged via the $\mathcal{L}_{\text{sim}}$ term in the generator objective (17), directly penalizing high similarity between the outputs of different generators for the same input (z, c).

The specific convolutional layers, normalization techniques, and activation functions within the generator and discriminator networks follow standard deep convolutional GAN (DCGAN) [RMC16] principles and other relevant architectural patterns suitable. This cMADGAN structure provides a framework for conditional generation that leverages multiple generators for diversity.

3.8 Image Scores

To quantitatively evaluate the quality and diversity of images generated by generative models, several metrics have been proposed in the literature. These image scores aim to provide an objective measure to compare generative models independently of human evaluation. This section introduces widely-used metrics for evaluating generative models: *Inception Score* (IS), *Fréchet Inception Distance* (FID), and the underlying InceptionV3 model employed for these metrics.

3.8.1 Inception Score

The Inception Score (IS) [SGZ+16] is one of the earliest and most commonly used metrics for evaluating generative models, especially GANs. It leverages a pretrained InceptionV3 classifier to assess two main criteria of generated images:

- **Image Quality (Clarity)**: Each generated image should be classified into a specific class with high confidence. This corresponds to a low-entropy conditional label distribution $p(y|x)$.
- **Diversity**: Across the entire set of generated images, the distribution of predicted classes should be diverse and cover many labels. This corresponds to a high-entropy marginal label distribution $p(y)$.

Mathematically, the Inception Score is computed over a set of generated images x drawn from the generator's distribution p_g:

$$IS = \exp\left(\mathbb{E}_{x\sim p_g}\left[D_{\text{KL}}(p(y|x)\|p(y))\right]\right) \tag{21}$$

where D_{KL} denotes the Kullback-Leibler divergence between the conditional class distribution $p(y|x)$ for a specific image x and the marginal class distribution $p(y)$ estimated over all generated images.

Interpretation: A higher IS indicates that the model generates images that are confidently classified into diverse classes.
Limitations of IS:

- Does not directly compare generated images with real images.
- Insensitive to intra-class diversity (e.g., generating only one type of dog within the dog class).
- Sensitive to the choice and specific training of the pretrained classifier.

3.8.2 Fréchet Inception Distance

The Fréchet Inception Distance (FID) [HRU+18] improves upon IS by directly comparing the feature distributions of real and generated images. FID embeds both real (r) and generated (g) images into a lower-dimensional feature space using a pretrained InceptionV3 network (typically using the activations from the final average pooling layer, often referred to as 'pool3'). It then models the distributions of these embeddings as multivariate Gaussian.

Let (μ_r, Σ_r) and (μ_g, Σ_g) denote the means and covariance matrices of the real and generated image Inception feature embeddings, respectively. The FID score is computed as the Fréchet distance (also known as Wasserstein-2 distance) between these two Gaussian distributions:

$$FID = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right) \tag{22}$$

where $\|\cdot\|_2^2$ is the squared Euclidean norm, $\mathrm{Tr}(\cdot)$ is the trace of a matrix, and $(\Sigma_r\Sigma_g)^{1/2}$ is the matrix square root of the product of the covariance matrices.
Interpretation: Lower FID indicates that the distribution of generated image features is more similar to the distribution of real image features, suggesting better quality and diversity. A score of 0 indicates identical distributions.
Advantages over IS:

- Directly compares the generated image distribution to the real image distribution.
- More sensitive to intra-class mode collapse (lack of diversity within classes) as this affects Σ_g.
- Generally found to correlate better with human judgment of image quality than IS.

Limitations of FID:

- Assumes Gaussian distribution of features, which might be an approximation.

- Sensitive to the choice of feature extractor network and the specific layer used.
- Sensitive to both image quality (influencing feature representations) and diversity (influencing the mean and covariance of features).
- Requires a sufficiently large number of samples (both real and generated) for stable estimation of moments (μ, Σ).

3.8.3 InceptionV3 for Image Evaluation

Both IS and FID commonly rely on the InceptionV3 model [SVI+16], a deep convolutional neural network pretrained on the large-scale ImageNet dataset [DDS+09]. The InceptionV3 network serves two primary roles in evaluating generative models:

1. **Feature Extractor (for FID)**: Intermediate activations, typically from the final average pooling layer *pool3*, are used as a semantic feature representation of the images. The statistics (mean and covariance) of these features are then compared between real and generated image sets.
2. **Classifier (for IS)**: The output layer provides class probabilities for input images. These probabilities are used to calculate the entropy terms required for the Inception Score.

Properties of InceptionV3 relevant for image scoring:

- Trained on the diverse ImageNet dataset, providing a rich semantic feature space capable of capturing complex visual patterns.
- Uses fixed, non-trainable weights during evaluation, ensuring standardization and comparability of scores across different studies, provided the same implementation is used.
- The choice of layer for feature extraction (primarily for FID) impacts sensitivity; earlier layers capture lower-level features like texture, while later layers capture more abstract semantic information. The *pool3* layer is commonly chosen as a balance.

Limitations of using InceptionV3:

- Domain Gap: The features learned on ImageNet (mostly natural images centered on objects) might not be optimally suited for evaluating images from significantly different domains, such as medical scans, satellite imagery, or abstract art.
- Sensitivity to ImageNet Classes: Performance, particularly for IS, can be less meaningful if the generated images depict objects or scenes vastly different from the 1000 ImageNet classes.

- Potential Bias: Potential biases present within the ImageNet dataset itself (e.g., representation bias) may be inherited by the features, potentially impacting evaluation fairness or relevance for specific target domains.

4 Experiments Setup

4.1 Preliminary Remarks

Before presenting the specific experiments and corresponding results in Chapter 5, this section outlines essential preliminary remarks. These remarks cover common configurations, definitions, and methodological aspects that apply across the subsequent experimental evaluations, providing necessary context and avoiding repetition later.

4.1.1 Scope Limitation Regarding Standard CIFAR-10

The CIFAR-10 dataset, with its 32x32 pixel color images across 10 classes, represents a significant setup in complexity compared to MNIST or Fashion-MNIST and is a common benchmark in generative modeling. Consequently, initial plans involved evaluating the performance of multi-generator GAN approaches, including the original MADGAN [GKN+18] and the adapted cMADGAN (Section 3.7), on this standard dataset.

However, preliminary investigations encountered substantial difficulties in achieving stable training and generating samples of sufficient quality and diversity using these frameworks directly on standard CIFAR-10. Extensive efforts were undertaken to address these challenges, spanning a range of common techniques and modifications found in GAN literature. These included, but were not limited to:

- Testing multiple generator and discriminator architectures with varying depths and capacities.
- Experimenting with different normalization layers (e.g., Batch Normalization, Spectral Normalization).
- Adjusting hyperparameters related to the Adam optimizer, particularly the learning rates for the generator(s) and discriminator, including various decay schedules and relative magnitudes.
- Implementing techniques designed to combat mode collapse and improve sample diversity, such as Mini-batch Discrimination.
- Tuning framework-specific hyperparameters like the latent dimension size and the diversity of weight (λ_{div}) for cMADGAN.
- Employing standard stabilization methods like label smoothing.

Despite these comprehensive attempts, persistent issues such as training instabilities (e.g., oscillating losses, vanishing or exploding gradients) or consistently poor quantitative results (e.g., low IS, high FID relative to benchmarks or simpler datasets) indicated

that the models did not converge to a satisfactory performance level on the standard CIFAR-10 dataset within the practical constraints of this study.

Given that the primary focus of this thesis is to investigate the comparative effects of different GAN-based data augmentation strategies (including MADGAN and cMADGAN), a pragmatic decision was made to exclude the standard CIFAR-10 dataset from the main comparative experiments presented in the subsequent results chapters. This allows the analysis to focus on datasets (MNIST, Fashion-MNIST) where the generative models achieved more stable and interpretable performance, enabling a clearer evaluation of the core research questions related to data augmentation effectiveness. The challenges encountered with standard CIFAR-10, while informative about the limitations of these specific multi-generator approaches on more complex data, are not subjected to further detailed analysis herein.

4.1.2 Used Datasets

Dataset	N-Samples	Image Size	Description
MNIST	70,000	$28 \times 28 \times 1$	Grayscale images of handwritten digits (0–9). A classic benchmark for basic image classification tasks.
Fashion-MNIST	70,000	$28 \times 28 \times 1$	Grayscale images of clothing items. Provides more complexity and variability than MNIST, better reflecting real-world tasks.

Table 1: Description of the used datasets for benchmarking.

4.1.3 GAN: Architecture, Training and Data Augmentation

Architecture of the GANs In all experiments involving GAN models and their derivatives, the same network architecture has been employed for the *MNIST* and *Fashion MNIST* datasets. The architectures for the generators and their corresponding discriminator can be seen in the appendix (8.1.2, 8.1.3). In the beginning of an experiment, the used architectures will be linked accordingly.

Training For training all GAN-based models, including DCGAN, cGAN, MADGAN and cMADGAN, the *Adam* optimizer has been utilized. The learning rate follows an exponentially decaying schedule throughout the training process.

Data Augmentation To increase the diversity of the training data for both generator and discriminator models, several traditional augmentation techniques have been applied. These include horizontal flips, brightness and contrast adjustments, and the addition of Gaussian noise. Horizontal flips are applied with a probability of 50%, except for the *MNIST* dataset, where flips are omitted due to the semantic relevance of digit orientation. Brightness and contrast adjustments are always applied within a

uniform range of $[-0.1, 0.1]$. The augmentation process adds Gaussian noise, by sampling from a normal distribution with mean 0 and standard deviation 0.05. Finally, the augmented images are clipped to the valid value range of $[-1, 1]$.

4.1.4 Stratified Classifiers as measure for augmentation Quality

To definitively evaluate the quality of the GDA, the fake images are used to replace and expand the underlying original datasets and train classification models on them. For this, the training datasets are specifically created to contain different ratios of real to fake images. To avoid biasing one class in the datasets over another, the datasets are stratified with respect to the number of samples per class. [9]

4.1.5 Labeling unconditioned data

Due to the fact, that multiple experiments use unconditioned GANs were executed (5.1), many images have been created with no corresponding label to them. To classify the unlabeled data, simple CNN classifiers, with adequate TDA, were utilized. The applied augmentations techniques are the as follows: horizontal- and vertical shift by 0.1 relative to the absolute size of the image, rotation of up to 15 degree and vertical flipping. As afore mentioned, flipping images along the vertical axes is omitted for the MNIST dataset, due to semantical invalidity. Graphical depictions of the classifier used can be found in the appendix (8.1.1). The auxiliary classifiers were optimized for their weighted sum of accuracy and precision on the held-out test set of the respective dataset.

4.1.6 Utilization of InceptionV3 for FID and IS

It is crucial to note that the significant differences between the ImageNet domain (high-resolution, color, 1000 object classes) and datasets commonly used in GAN research like MNIST, Fashion-MNIST (low-resolution, grayscale), or CIFAR-10 (low-resolution, color, 10 simpler classes) represent a substantial domain gap 3.8.3. This gap may limit the effectiveness or absolute interpretability of InceptionV3-based scores for these specific datasets. Furthermore, due to the sensitivity of these scores to implementation details (e.g., image resizing methods, specific InceptionV3 weight versions), a direct comparison of scores calculated here to those from external literature is generally unreliable unless the evaluation methodology is verified to be identical. Therefore, within this thesis, IS and FID scores are primarily utilized for relative comparisons between the different models and experiments conducted herein, rather than for absolute benchmarking against potentially disparate external results. This context warrants careful consideration when analyzing the experimental outcomes presented later.

[9] Out of the used datasets, only the MNIST dataset is not originally stratified.

4.2 Experimental Workflow

The evaluation of each generative model adhered to a consistent experimental workflow, outlined below:

1. **GAN Training:** The specific generative model (e.g., DCGAN, cGAN, MADGAN, cMADGAN) was trained on the target dataset. Model performance during training was monitored using predefined metrics such as FID and IS.

2. **Synthetic Sample Generation (Per Generator):** After training, the individual generators G_i / $G_{i,j}$ within the trained model (where $i = 1$ for single-generator models like DCGAN/cGAN, and $i = 1...K, j = 0...(K-1)$ for multi-generator models MADGAN/cMADGAN with $K \in \{3, 5, 7, 10\}$) are used to create a distinct set of synthetic images, denoted as $S_{\text{fake},i}$. For each class in the original dataset, at least 6000 images are generated by each generator G_i, resulting in K separate datasets of synthetic samples for each trained multi-generator model.

   - *Labeling Unconditional Samples:* For samples generated by unconditional models or generators (DCGAN, MADGAN generators), class labels are assigned to the images within each respective set $S_{\text{fake},i}$ using the pre-trained classifiers detailed in Section 4.1.5.

3. **Downstream Classifier Training (GDA Evaluation - Per Generator):** The effectiveness of Generative Data Augmentation (GDA) was evaluated separately for each generator G_i / $G_{i,j}$ of a trained GAN-based model, using its corresponding synthetic sample set $S_{\text{fake},i}$ / $S_{\text{fake},i,j}$. For single-generator models, this step was performed once using $S_{\text{fake},1}$. Using a fixed classifier architecture specific to each dataset (8.1.1) trained for 50 epochs. The classifiers are trained under two distinct augmentation scenarios for each sample set $S_{\text{fake},i}$:

   - *Replacement Scenario:* This assessed the utility of synthetic data from generator G_i as a substitute for real data. Training commenced with a baseline classifier using 5000 real images per class. In subsequent steps, the number of real images per class was decreased by 1000 while the number of synthetic images per class (drawn from $S_{\text{fake},i}$) was increased by 1000, maintaining a constant dataset size of 5000 images per class. This process continued until the final classifier was trained solely on 5000 synthetic images per class from $S_{\text{fake},i}$.

   - *Expansion Scenario:* This evaluated synthetic data from generator G_i as a supplement to real data. Training started with the same baseline (5000 real images per class). The real dataset was then augmented by adding synthetic images from $S_{\text{fake},i}$ in increments of 1000 per class per step, reaching a maximum of 5000 synthetic images per class. The final classifier in this scenario

was trained on a combined dataset of 5000 real and 5000 synthetic images per class from $S_{\text{fake},i}$.

Note that for each trained MADGAN or cMADGAN model with K generators, the full set of Replacement and Expansion classifier trainings were performed K times, once for each generator's synthetic dataset.

4. **Downstream Classifier Evaluation:** The performance of all trained classifiers (baseline and those from the replacement and expansion scenarios for each generator set $S_{\text{fake},i}$) was evaluated using predefined classification metrics 3.1.3. Results for multi-generator models may be presented as averages across the K generators or by selecting representative examples of specific generators with specified ratios between real and fake images.

In total, this setup results in 44 (datasets (2) * augmentation types (4) * experiment setup (2) * n-generators trained ([3, 5, 7, 10])) experiment sets. Taking the different ratios of real to fake images in the expansion / replacement scenarios into account, a total of 1.166 separate classifiers were trained to evaluate the potential of multi-agent architecture for GDA.

4.3 Comparison of Classifier Performance

Due to the fact that the MADGAN and cMADGAN architectures apply a multi-agent strategy i.e., training multiple generators for one model, only the best run is selected for direct comparison to others. For example, when comparing MADGAN to cMADGAN in 5.2.4, the best performing subset of the respective GAN architecture is used for comparison. The rest of the experiments not discussed explicitly or only mentioned can be seen in the appendix 8.3.

4.4 Hardware and Software Environment

4.4.1 Hardware

All models trained in the context of this thesis were trained utilizing the *Deeplearning Cluster* provided by the *Hochschule der Medien - Stuttgart*. The cluster provides 8 nodes that with dedicated graphics cards supporting the *CUDA* framework. The specific machines deployed in the cluster can be seen here Deeplearn cluster Documentation[10].

[10]Link to the documentation for print versions: `https://deeplearn.pages.mi.hdm-stuttgart.de/docs/`.

4.4.2 Software

The code for the experiments was developed with the programming language python, using version 3.9. Models are based on the tensorflow ecosystem. The exact packages and their respective version can be found here Aanconda environment[11].

[11] Link to the environment file for print versions: `https://github.com/N10100010/mad_gan_thesis/blob/main/code/server_env.yml`.

5 Experiments Results

Motivation The primary motivation for investigating multi-generator GAN architectures for Generative Data Augmentation (GDA) stems from a suggestion by Ian Goodfellow on the Lex Fridman Podcast [FG19]. He proposed leveraging the diversity inherent in multiple generative models trained on the same data to potentially improve downstream classifiers:

> So one thing I think is worth trying [...] is, what if you trained a whole lot of different generative models on the same training set, create samples from all of them and then train a classifier on that. Because each of the generative models might generalize in a slightly different way, they might capture different axes of variation, that one individual model wouldn't and then the classifier can capture all of those ideas, by training on all of their data.

[FG19, 50:37]
Goodfellow's concept resonates strongly with the principles of Multi-Agent Diverse GANs (MADGANs) [GKN+18]. The MADGAN architecture, with its explicit diversity-promoting objective and use of multiple generators, provides a suitable framework for realizing this augmentation strategy. Therefore, the work by Ghosh et al. laid the conceptual groundwork for this thesis.

5.1 Key Research Questions

This chapter investigates the following questions regarding MADGANs for data augmentation:

- **Question 1**: How do the FID- and Inception Score compare between the generative methods?
- **Question 2**: Does Generative Data Augmentation (GDA) with MADGANs enhance downstream classifier performance more effectively than Traditional Data Augmentation (TDA)?
- **Question 3**: How does the performance enhancement achieved with MADGAN-based GDA compare to that of GDA using standard GANs or conditional GANs?
- **Question 4**: How does the performance enhancement achieved with MADGAN-based GDA compare to that of cMADGAN-GDA?
- **Question 5**: What is the impact of varying the number of MADGAN generators on downstream classifier performance?

5.2 Key Research Question Answers

As mentioned in the subchapter about the experimental workflow, four sets of multi-agent generators models were trained for the MADGAN and cMADGAN architectures (4.2). The four sets differ by K, the number of generators trained, with $K \in [3, 5, 7, 10]$. Onward, the generator of a discussion is identified explicitly by a suffix referencing the generator of a trained set $G_{i,j}$ with $i = K, j = 0...(K-1)$ (e.g., $\text{MADGAN}_{5,0}$ for the first generator in the architecture trained with five generators). The set of generators, however, is referenced via the notation: MADGAN $K = 5$ (e.g., if a discussion talks about the average performance of the generators 0...4). Throughout this chapter, the experiment will be viewed with respect to afore mentioned differentiation between expansion and replacement scenarios 3. The comparisons of FID- and Inception Score are excluded from this differentiation. Experiment not directly discussed in greater detail can be seen in the appendix 8.3. The generated graphs contain all respective runs of a given experimental setup for which, best, worst, average, median, and baseline are explicitly highlighted. Each gray line in a graph, that is not highlighted, represents every other combination of the setup. Especially for the multi-agent generator setups, this results in many graphs, up to 60 per figure in case of 10 generators. Therefore, a colorful highlighting of specific graphs is omitted as this would not benefit the overall readability of the respective graphs. This, however, removes the potential for interesting insights.

5.2.1 Comparison of FID- and Inception Scores

In order to compare the FID-Score and the means and standard deviations of the IS, 10000 generated samples are chosen by random selection. These fake images are drawn from the entirety of the generated data for a given generator, regardless of their assigned class. The comparison is based on the respective datasets used (MNIST, Fashion-MNIST) for data generation. Baselines have been created by probing the respective training datasets for the same amount of 10000 samples.

MNIST Dataset

Generator Type	Generators trained	FID	IS	IS-std
Baseline	-	−0.004	2.549	0.036
DCGAN	1	122.097	**2.611**	0.056
cGAN	1	28.721	2.553	0.022
MADGAN	K=3 (avg)	23.177	2.511	0.04
MADGAN	K=5 (avg)	22.656	2.471	0.044
MADGAN	K=7 (avg)	21.599	2.533	0.051
MADGAN	K=10 (avg)	**20.973**	2.474	0.037
cMADGAN	K=3 (avg)	25.578	2.398	0.036
cMADGAN	K=5 (avg)	29.071	2.35	0.039
cMADGAN	K=7 (avg)	30.645	2.354	0.039
cMADGAN	K=10 (avg)	110.553	2.062	0.018

Table 2: FID and IS results for GAN models on MNIST, comparing single-generator (DCGAN, cGAN) and multi-generator (MADGAN, cMADGAN; K=3-10) approaches. Baseline created from training datasets. Baseline created using images from the training set.

Interpretation: Table 2 presents the Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for various GAN models evaluated on the MNIST dataset. The results reveal a trade-off between the two metrics across different architectures.

The baseline DCGAN achieved the highest IS (2.611) but performed poorly in terms of FID (122.097). Introducing conditioning via cGAN significantly improved the FID to 28.721 while maintaining a high IS (2.553). The results for the DCGAN point to an eventual mode collapse. A histogram of the resulting labels from this data generation can confirm the mode collapse[12].

For unconditional models, the multi-generator MADGAN framework consistently yielded better FID scores than the baselines. Furthermore, MADGAN's FID improved monotonically as the number of generators increased, achieving the best overall FID of 20.973 with 10 generators. However, its IS scores (2.5) were slightly lower than the

[12] A histogram of the resulting labels originating from the generation process of the DCGAN on MNIST can be found here 30.

single-generator baselines.

The conditional multi-generator adaptation, cMADGAN, showed a more complex relationship with the number of generators (K). It performed very poorly for K=3 (FID 110.553) but improved substantially at K=5, achieving a competitive FID (25.578) - better than cGAN - albeit with a lower IS (2.398). Contrary to MADGAN, increasing generators beyond K=5 resulted in worse FID scores for cMADGAN (29.071 for K=7, 30.645 for K=10). Consequently, for K>=5, the unconditional MADGAN consistently outperformed cMADGAN in FID within these experiments.

In summary, on MNIST, standard conditioning (cGAN) greatly enhances baseline FID. The unconditional MADGAN framework effectively improves FID further, benefiting from more generators. The conditional cMADGAN variant demonstrates potential (peaking at K=5) but exhibits non-monotonic FID performance with increasing generator count in this setup, suggesting a more complex optimization landscape compared to its unconditional counterpart.

Fashion-MNIST Dataset

Generator Type	Generators trained	FID	IS	IS-std
Baseline	-	−0.004	4.723	0.056
DCGAN	1	25.56	4.21	0.099
cGAN	1	123.349	3.573	0.117
MADGAN	K=3 (avg)	26.202	4.496	0.099
MADGAN	K=5 (avg)	24.218	4.497	0.098
MADGAN	K=7 (avg)	23.875	4.523	0.094
MADGAN	K=10 (avg)	**21.587**	4.534	0.097
cMADGAN	K=3 (avg)	25.555	**4.623**	0.105
cMADGAN	K=5 (avg)	160.082	3.346	0.038
cMADGAN	K=7 (avg)	154.115	2.929	0.034
cMADGAN	K=10 (avg)	159.067	3.317	0.035

Table 3: FID and IS results for GAN models on Fashion-MNIST, comparing single-generator (DCGAN, cGAN) and multi-generator (MADGAN, cMADGAN; K=3-10) approaches. Baseline created using images from the training set.

The evaluation metrics for the various GAN models on the Fashion-MNIST dataset, presented in Table 3, reveal distinct performance patterns. The baseline DCGAN provided a reasonable starting point with an FID of 25.56 and an IS of 4.21. However, unlike observations on MNIST, standard conditioning via cGAN proved detrimental on this dataset within this setup, resulting in significantly degraded FID (123.35) and IS (3.57) compared to the unconditional DCGAN.

In contrast, the unconditional MADGAN framework demonstrated robust and consistently improving performance. Its FID, initially comparable to DCGAN at K=3 (26.20), improved monotonically as the number of generators increased, achieving the

table's best FID of 21.59 at K=10. Notably, MADGAN also maintained high IS scores (around 4.5), which showed a slight tendency to increase with more generators.

The conditional adaptation, cMADGAN, exhibited highly sensitive and divergent behavior on Fashion-MNIST compared to MNIST. Only the $K = 3$ configuration yielded strong results, achieving a competitive FID (25.56) while registering the table's highest IS (4.62). However, increasing the generator count further to $K = 5$, 7, or 10 led to a dramatic performance collapse, characterized by extremely high FID scores (approximately 154–160) and substantially lower IS scores (around 2.9–3.3).

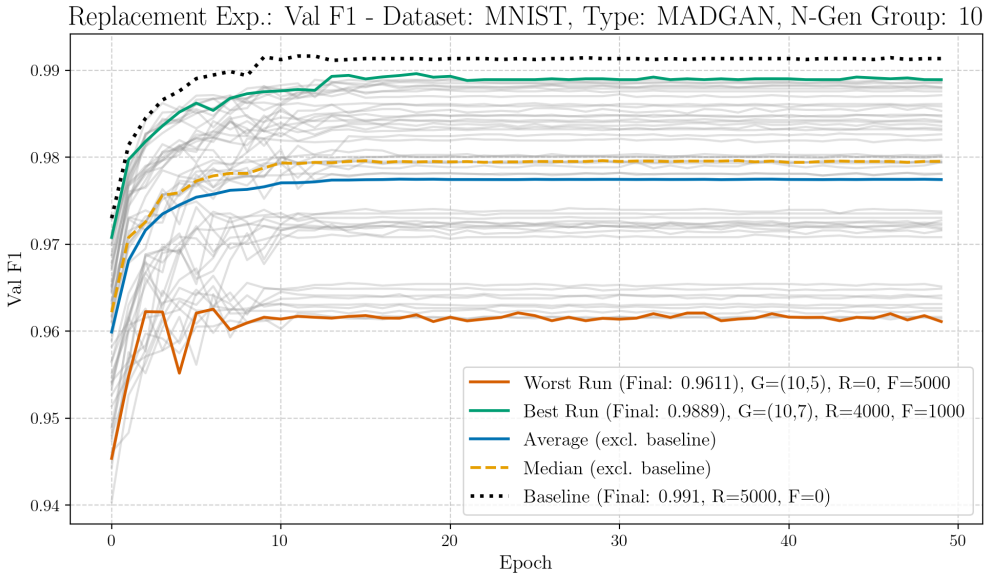
Overall, while cMADGAN K=3 achieved the best IS score, MADGAN K=10 attained the best FID score and offered a strong combination of both metrics. In conclusion, for Fashion-MNIST under these experimental conditions, standard conditioning failed to provide benefits, whereas the unconditional MADGAN framework scaled effectively, improving FID and IS with more generators. The conditional cMADGAN approach was only viable with a small number of generators (K=3) and did not exhibit the positive scaling or peak performance characteristics observed on MNIST.

5.2.2 Question 2: Effectiveness of MADGAN GDA vs. TDA

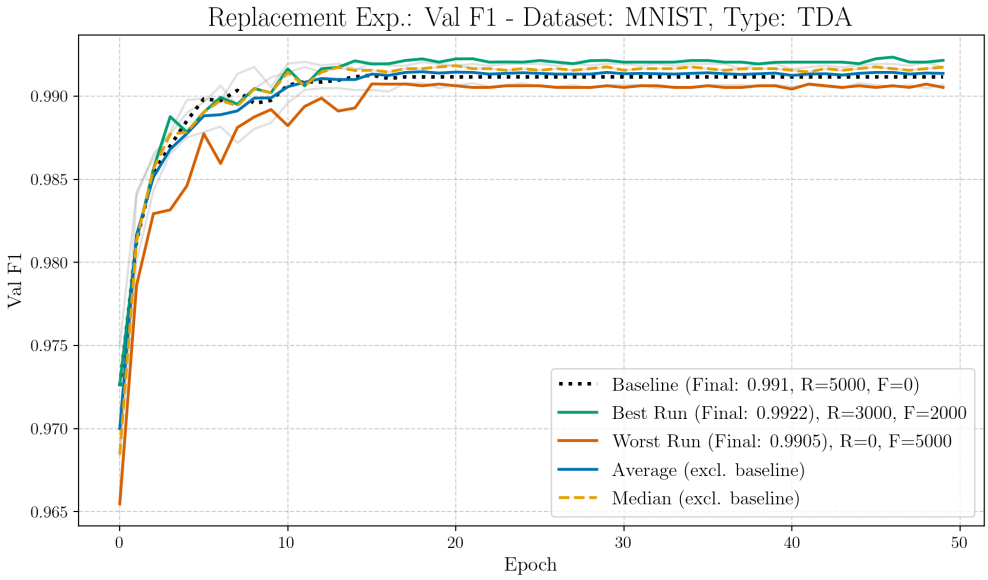
As mentioned in the beginning of this chapter, four sets of generators with different numbers for K were trained, with $K \in [3, 5, 7, 10]$, for the multi-agent architectures. For comparison against other GDA methods, the best performing set (e.g., MADGAN $K = 3$) or the best performing generator (e.g., $G_{3,2}$) is used. All other experiments are present in the appendix 8.3. From here onward, the differentiation between replacement and expansion experiments will be used. The experiments data is displayed with two consecutive graphs and tables. The graphs show the trajectory of validation F1 scores over the trained epochs. The tables below summarize their performance.

TODO: preamble the results of the following experiments

Replacement Experiment, Dataset: MNIST



(a) F1 Score on MNIST over 50 epochs. Augmentation technique: MADGAN (K=10)



(b) F1 Score on MNIST over 50 epochs. Augmentation Technique: TDA

Run Type	Experiment	Val F1
best	$G_{10,7}$, R:4000, F:1000	0.9889
worst	$G_{10,5}$, R:0, F:5000	0.9611
median	G (K=10)	0.9795
average	G (K=10)	0.9774

Table 4: Final F1 Scores after 50 epochs. Augmentation technique: MADGAN

Run Type	Metric	Val F1
best	TDA (R:3000, F:2000)	0.9922
worst	TDA (R:0, F:5000)	0.9905
median	TDA	0.9917
average	TDA	0.9914

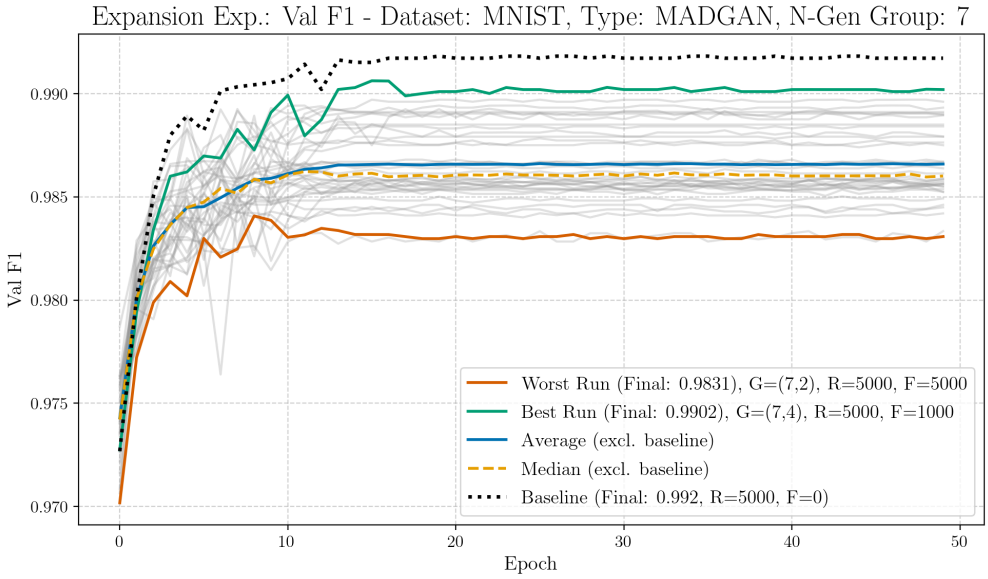
Table 5: Final F1 Scores after 50 epochs. Augmentation technique: TDA

All graphs in the corresponding figure show rapid convergence and stable training 15 epochs. After 20 epochs, only small fluctuations in the result can be seen for both TDA and MADGAN GDA. With a small spread of 0.0017, all ratios for the TDA resulted in a F1 score, on the validation set of over 0.99. This allows the conclusion that replacing training images for a classifier with modified images shows minimal negative impact on their performance, measured no the validation set using the F1 score. It is to point out, that the best, median and avg measured surpassed the baseline, leading to the fact that the augmentation improved the classifiers' performance on average.

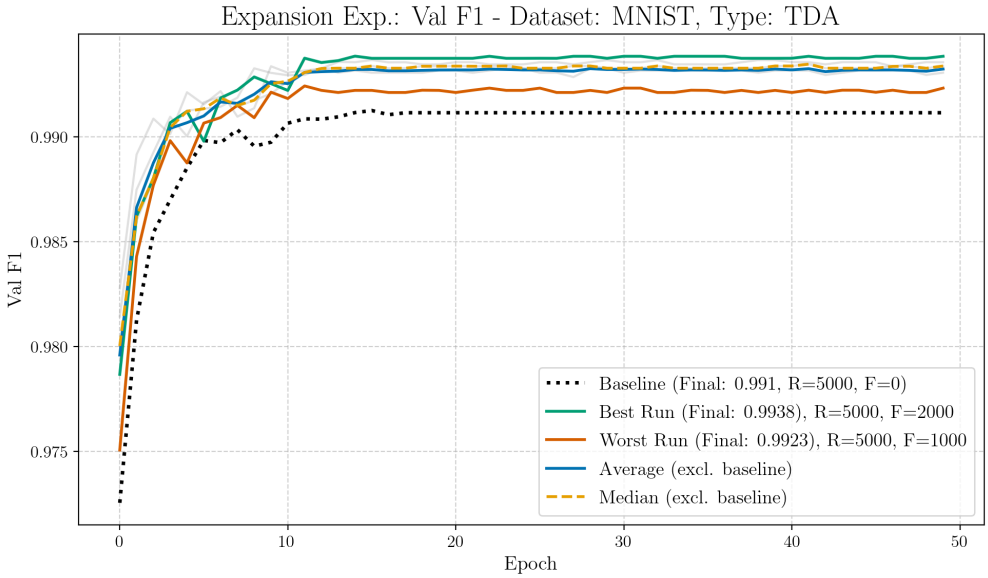
The best performing setting on MNIST for MADGAN is with $K = 10$. Showing a similar fast convergence compared to TDA, the MADGAN-augmented classifiers are mostly converged after the 15th epoch. In contrast to the TDA classifiers however, the performance using MADGAN images resulted in a significantly wider spread of classifier performances, of 0.0278. It is important to again mention, that Figure 5a shows performances across all ten generators, over all replacement ratios. This results in a total of 60 classifiers trained. Generally, the graphs convergence to lower F1 scores, compared to the classifiers utilizing TDA. Here, it can be concluded, that the average and median performance suffered from replacing the real images with generated images. The summary table quantifies this fact: the average final F1 score across ratios and generators is 0.9775, which is significantly lower than the average of the traditional augmentation. While the best setup ($G_{10,7}, R : 4000, F : 1000$) reached a good score: 0.9889; it is lower than the average performance across the different ratios in the TDA experiment. Even the worst performing classifier in the TDA experiment is better than the best score for GDA in this case.

In direct comparison, the TDA consistently outperforms GDA (with its best resulting setting, MADGAN (K=10)) in this replacement scenario. Replacing the real data with synthetic lead to a noticeable degradation of the classifiers' performance on the validation set. Taking the results from question 1 into account (5.2.1), even the MADGAN setup resulting in the best FID score are not as effective as traditional augmented real data, when substituting the real samples in the training set.

Expansion Experiment, Dataset: MNIST



(a) F1 Score on MNIST over 50 epochs. Augmentation technique: MADGAN (K=7)



(b) F1 Score on MNIST over 50 epochs. Augmentation Technique: TDA

Run Type	Experiment	Val F1
best	$G_{7,4}$, R:5000, F:1000	0.9902
worst	$G_{7,2}$, R:5000, F:5000	0.9831
median	G (K=7)	0.9860
average	G (K=7)	0.9866

Table 6: Final F1 Scores after 50 epochs. Augmentation technique: MADGAN

Run Type	Experiment	Performance
best	TDA (R:5000, F:2000)	0.9938
worst	TDA (R:5000, F: 1000)	0.9923
median	TDA	0.9934
average	TDA	0.9932

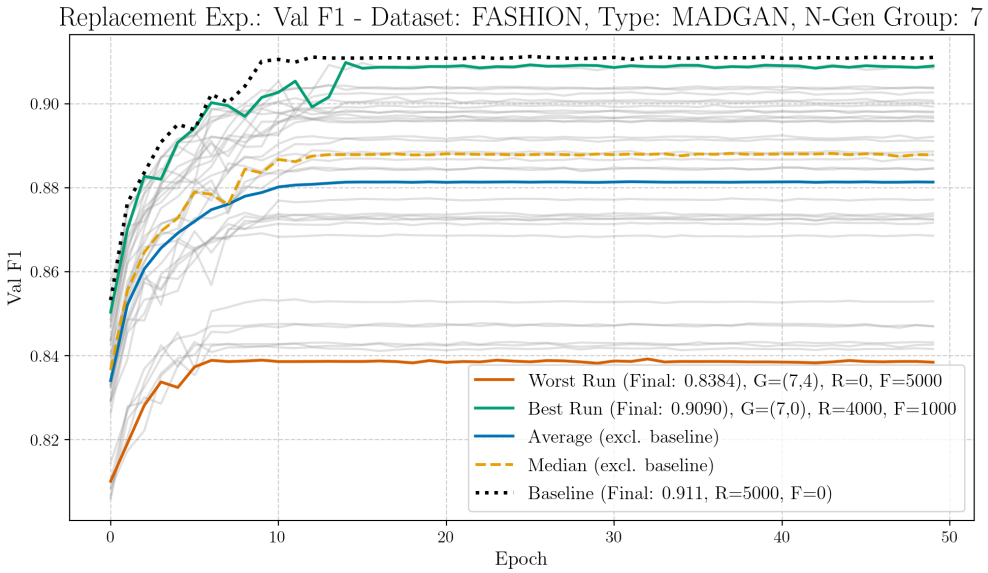
Table 7: Final F1 Scores after 50 epochs. Augmentation technique: TDA

The results using TDA (Table 7) show consistently high performance. The average final F1 score across all expansion levels (adding 0 to 5000 augmented samples per class) is 0.9932, with minimal variation between the best (0.9938, achieved when adding 2000 augmented samples) and worst (0.9923) cases. This indicates that expanding the dataset with traditionally augmented samples maintains, and perhaps very slightly improves, the already high baseline performance on MNIST.

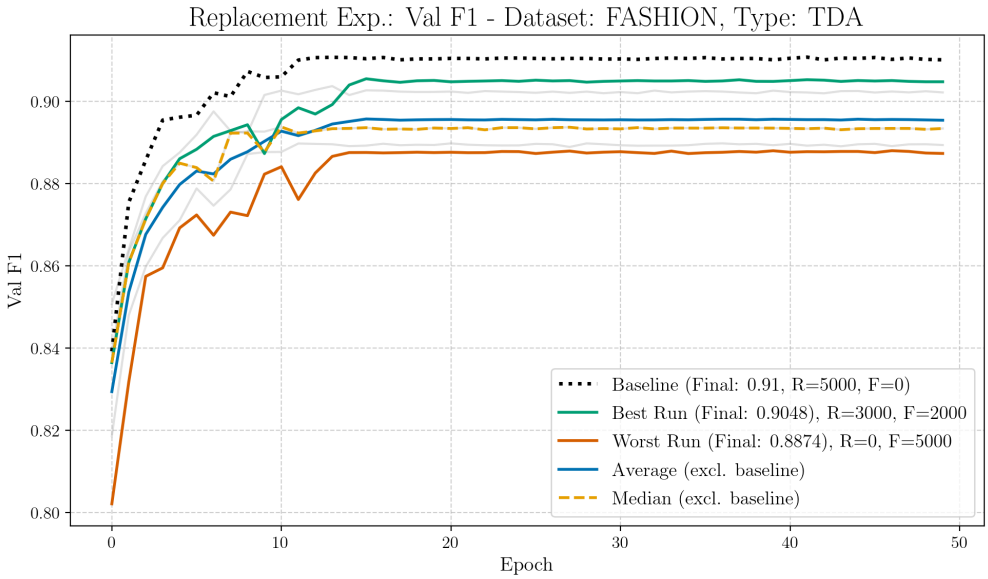
In contrast, using GDA with samples generated by the MADGAN (K=7) model did not yield performance improvements over the baseline trained only on real data. To be noted explicitly, none of the expansion experiments using MADGAN GDA surpassed the baseline performance level (see: 8.3.1). The summary statistics in Table 6, which cover results across all 7 generators and all expansion ratios, confirm this. The average final F1 score is 0.9866, noticeably lower than the TDA results. The best-performing run across all generators and expansion levels only reached 0.9902 (using generator $G_{7,4}$ when adding 1000 synthetic samples), which is below even the worst TDA result. Performance tended to decrease as more synthetic data was added, with the lowest score (0.9831) occurring when the maximum of 5000 synthetic samples per class were added (using generator $G_{7,2}$).

Comparing the two augmentation strategies in the expansion scenario, TDA is clearly superior on MNIST in this setup. Expanding the dataset with traditionally augmented data maintains excellent performance, whereas expanding with MADGAN-generated synthetic data fails to improve over the real-data baseline and leads to lower overall performance. This suggests that the synthetic samples from MADGAN (K=7), despite the model potentially having good generative scores, dilute rather than enhance the training data quality when added to the real MNIST dataset for this downstream classification task.

Replacement Experiment, Dataset: Fashion-MNIST



(a) F1 Score on FASHION over 50 epochs. Augmentation tech.: MADGAN (K=7)



(b) F1 Score on FASHION over 50 epochs. Augmentation Technique: TDA

Run Type	Experiment	Val F1
best	$G_{7,0}$, R:4000, F:1000	0.9090
worst	$G_{7,4}$, R:0, F:5000	0.8384
median	G (K=7)	0.8879
average	G (K=7)	0.8813

Table 8: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=7)

Run Type	Experiment	Val F1
best	TDA (R:3000, F:2000)	0.9048
worst	TDA (R:0, F:5000)	0.8874
median	TDA	0.8934
average	TDA	0.8955

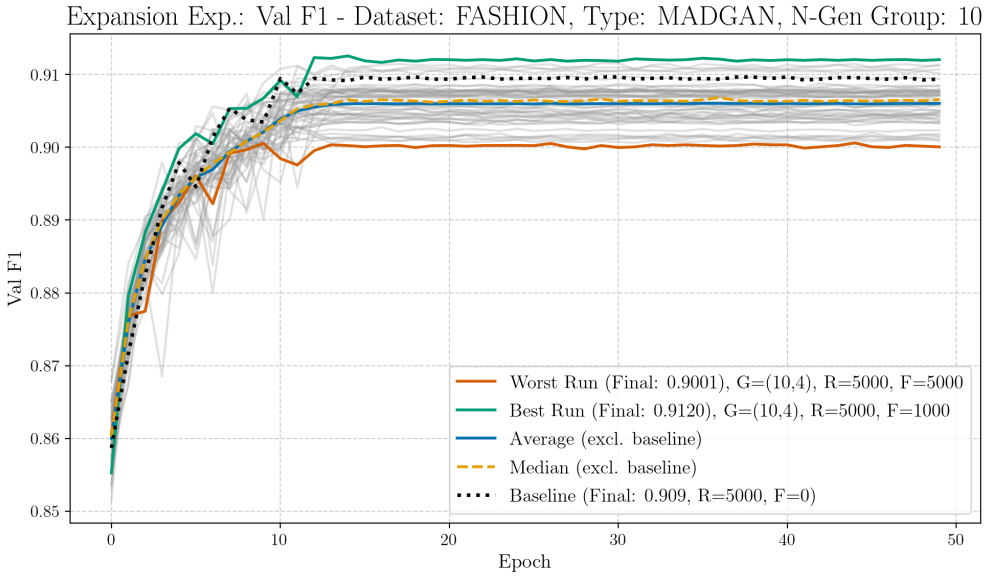
Table 9: Final F1 Scores after 50 epochs. Augmentation technique: TDA

For Traditional Data Augmentation (TDA), as shown in Table 9, the classifier performance remains reasonably strong and relatively stable even as real data is substituted with traditionally augmented samples. Across all replacement ratios, the average final F1 score was 0.8955. Performance experienced only a moderate decline with increasing replacement, peaking at 0.9048 with a mix of 3000 real and 2000 augmented samples per class, and bottoming at 0.8874 when relying solely on 5000 augmented samples. The narrow performance range underscores the consistency offered by TDA.

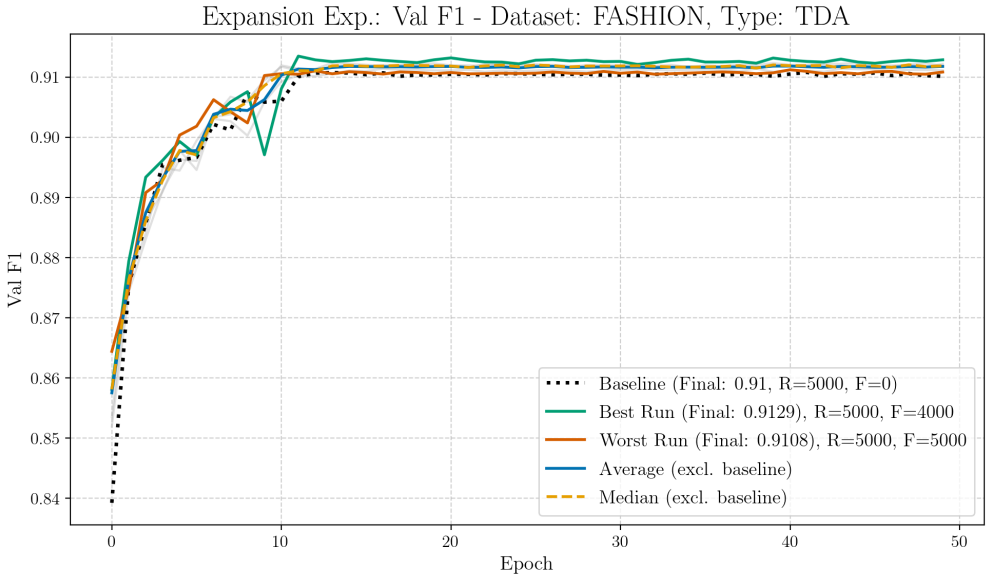
The MADGAN (K=7) GDA results, detailed in Table 8, present a more varied picture. Notably, MADGAN GDA achieved a peak F1 score of 0.9090 (from generator $G_{7,0}$ with 4000 real and 1000 synthetic samples), which slightly surpasses the best performance seen with TDA. However, this potential for high performance is coupled with greater variability. The worst-performing scenario for MADGAN GDA (generator $G_{7,4}$ using only its 5000 generated samples) yielded an F1 score of 0.8384. While this is a significant drop, it's a marked improvement over previously indicated much lower worst-case scenarios. This brings MADGAN's average F1 score to 0.8813, which is closer to, but still below, the TDA average. The median F1 score for MADGAN GDA (0.8879) is also slightly lower than TDA's median (0.8934).

Directly comparing the replacement scenario on Fashion-MNIST, TDA continues to offer a more consistently reliable performance. While MADGAN GDA, with its best generator and limited replacement, demonstrates the capacity to achieve a slightly higher peak F1 score than TDA, TDA maintains better average, median, and notably, worst-case F1 scores. The synthetic data from some MADGAN generators, especially when used as a complete replacement for real data, leads to a more substantial performance degradation than observed with TDA, making TDA the more robust strategy overall for this replacement task, despite MADGAN's higher performance ceiling under optimal GDA conditions.

Expansion Experiment, Dataset: Fashion-MNIST



(a) F1 Score on FASHION over 50 epochs. Augmentation tech.: MADGAN (K=10)



(b) F1 Score on FASHION over 50 epochs. Augmentation Technique: TDA

Run Type	Experiment	Val F1
best	$G_{10,4}$, R:5000, F:1000	0.9120
worst	$G_{10,4}$, R:5000, F:5000	0.9001
median	G (K=10)	0.9066
average	(K=10)	0.9060

Table 10: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10)

Run Type	Experiment	Val F1
best	TDA (R:5000, F:4000)	0.9129
worst	TDA (R:5000, F:5000)	0.9108
median	TDA	0.9119
average	TDA	0.9118

Table 11: Final F1 Scores after 50 epochs. Augmentation technique: TDA

Above results show, that data expansion on Fashion-MNIST is beneficial for both augmentation techniques. Unlike observations on MNIST, for which MADGAN GDA did not surpass the respective baseline. Utilizing TDA (Table 11) to add samples improves the classifiers performance over the baseline, achieving a peak F1 score of 0.9129 when adding 4000 augmented samples to each class. The F1 performance remains high, despite varying ratios or real to fake images. The setting achieved an average F1 score of 0.9118 and even the worst case (adding 5000 samples) scored 0.9108.

On the Fashion-MNIST dataset, data expansion proved advantageous for both augmentation strategies, a notable contrast to earlier MNIST findings where MADGAN GDA struggled to exceed baseline performance. With Traditional Data Augmentation (TDA), as detailed in 11, incorporating additional augmented samples demonstrably boosted classifier F1 scores over the R:5000/F:0 baseline. This enhancement culminated in a peak F1 score of 0.9129 with the addition of 4000 augmented samples per class. TDA maintained remarkably high and stable performance throughout the expansion, evidenced by an average F1 score of 0.9118 and a strong worst-case score of 0.9108 even when 5000 augmented samples were added.

Generative Data Augmentation using MADGAN (K=10) also improved performance over the presumed baseline, as indicated by the summary statistics in 10. Its best-case F1 score reached 0.9120, achieved by generator $G_{10,4}$ with an addition of 1000 synthetic samples per class, closely rivaling TDA's peak and underscoring GDA's effectiveness in this scenario. While this peak is promising, MADGAN GDA exhibited slightly less consistency than TDA. Its average F1 score (0.9060) and median (0.9066) were marginally lower than TDA's equivalents. Furthermore, performance showed a more pronounced decline when maximally expanded with 5000 synthetic samples, dropping to 0.9001 in the worst run—a score notably below TDA's worst case. Nevertheless, the variability across MADGAN's generators and expansion ratios was considerably less extreme than what was observed in the Fashion-MNIST replacement experiments.

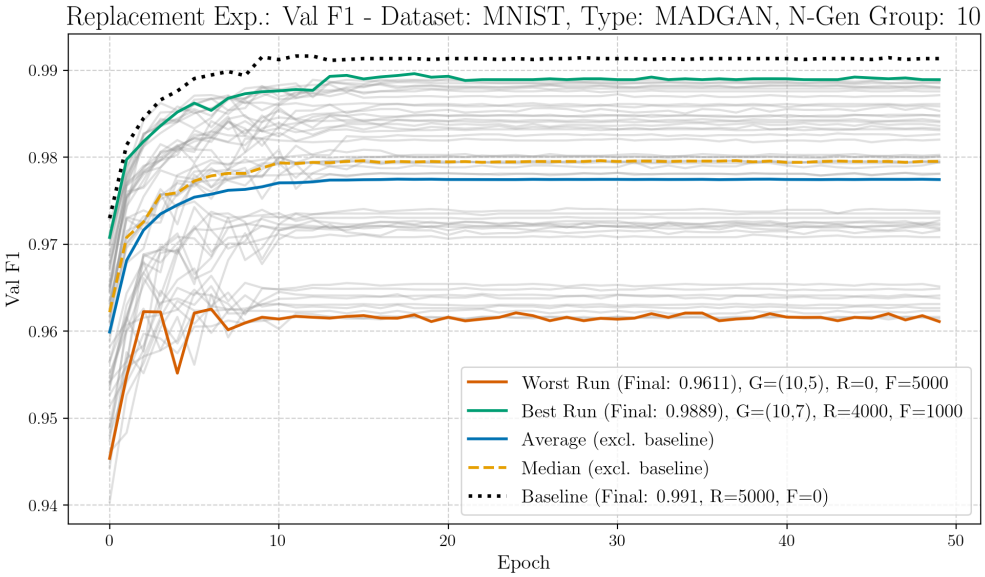
Ultimately, for data expansion on Fashion-MNIST, both TDA and MADGAN GDA (K=10) offered tangible benefits over relying solely on the original real data. TDA, however, maintained a slight overall advantage, delivering marginally higher peak performance (0.9129 for TDA vs. 0.9120 for MADGAN GDA) and superior stability, particularly evident when large volumes of augmented data were incorporated. While MADGAN GDA proved highly competitive by nearly matching TDA's peak, its slightly lower average scores and more significant performance drop under maximum expansion conditions indicate that TDA remains the more robust expansion technique in this specific setup, though MADGAN GDA is clearly a viable alternative.

5.2.3 MADGAN GDA vs. Deep Convolutional/Conditional GAN GDA

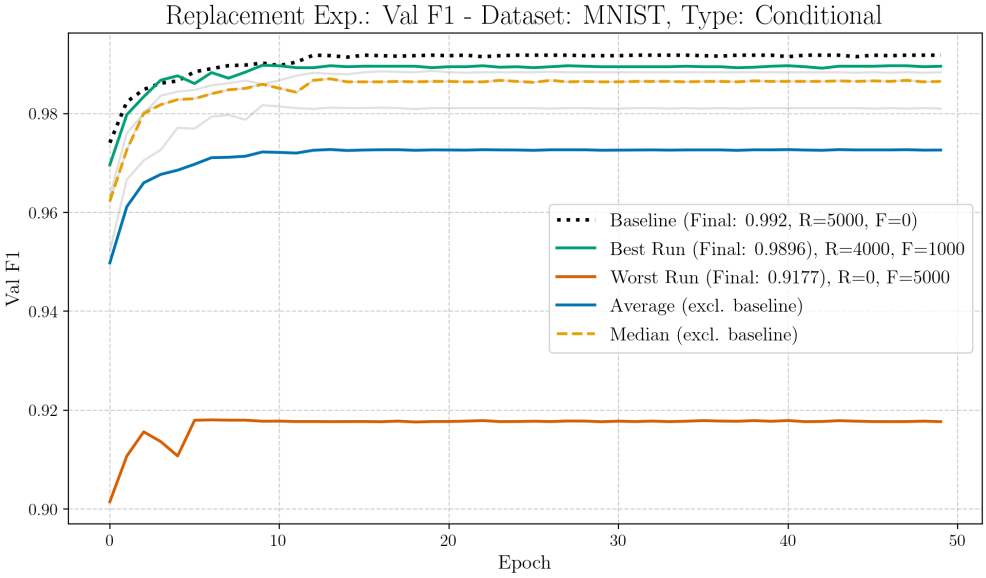
In the foregoing chapter (5.2.2), the most successful setting for the MADGAN architecture has been set. MADGAN with K=10 for Replacement-, with K=7 for the Expansion scenario on MNIST and for Fashion-MNIST K=7 for Replacement- and K=10 for Expansion proved to perform best under given circumstances. Following, the same settings for the respective dataset and the corresponding experimental setup (replacement, expansion) will be used to compare against the best performing architecture of deep convolutional and conditional GANs. For both, the Replacement and Expansion scenario, the cGAN surpassed the performance of the DCGAN. Thus, the cGAN is selected for direct comparison. The DCGAN will only be described and linked were fit.

TODO: preamble the results of the following experiments

Replacement Experiment, Dataset: MNIST



(a) F1 Score on MNIST over 50 epochs. Augmentation technique: MADGAN (K=10)



(b) F1 Score on MNIST over 50 epochs. Augmentation Technique: cGAN

Run Type	Experiment	Val F1
best	$G_{10,7}$, R:4000, F:1000	0.9889
worst	$G_{10,5}$, R:0, F:5000	0.9611
median	G (K=10)	0.9795
average	G (K=10)	0.9774

Table 12: Final F1 Scores after 50 epochs. Augmentation technique: MADGAN (K=10)

Run Type	Experiment	Val F1
best	Conditional (R:4000, F:1000)	0.9896
worst	Conditional (R:0, F:5000)	0.9177
median	Conditional	0.9865
average	Conditional	0.9726

Table 13: Final F1 Scores after 50 epochs. Augmentation technique: cGAN

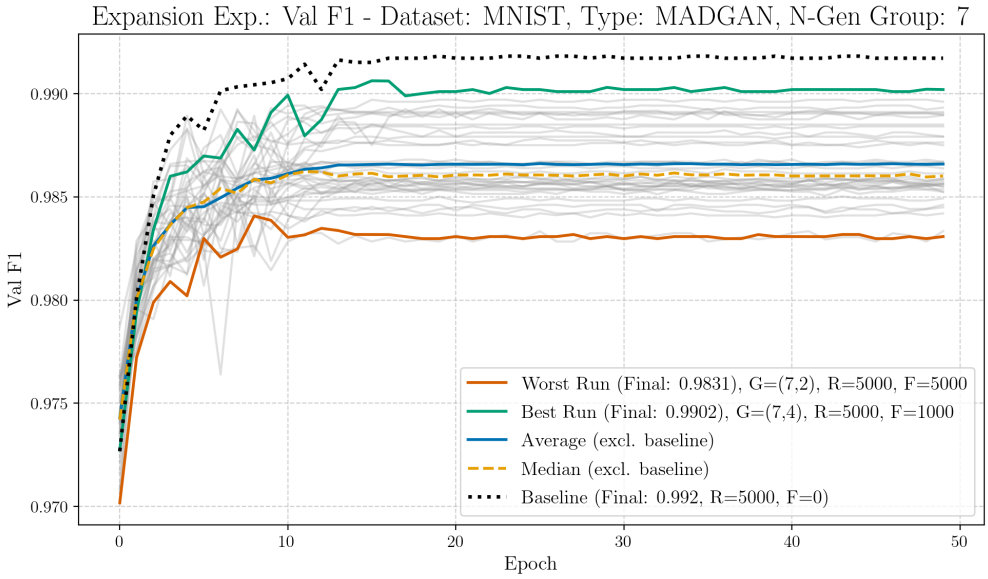
Both MADGAN (K=10) and cGAN demonstrate strong peak performance when replacing only a small amount of real data. As shown in Tables 12 and 13, the best F1 scores achieved are very similar, with cGAN reaching 0.9896 and MADGAN K=10 reaching 0.9889, both occurring when 1000 real samples per class were replaced with synthetic ones (R:4000, F:1000).

However, the two methods exhibit different robustness levels as more real data is replaced. The cGAN performance degrades significantly when relying entirely on synthetic data (R:0, F:5000), with the F1 score dropping to 0.9177. In contrast, the MADGAN (K=10) framework shows greater resilience in the purely synthetic scenario; even the worst-performing generator ($G_{10,5}$) at full replacement achieved an F1 score of 0.9611, considerably higher than cGAN's worst score.

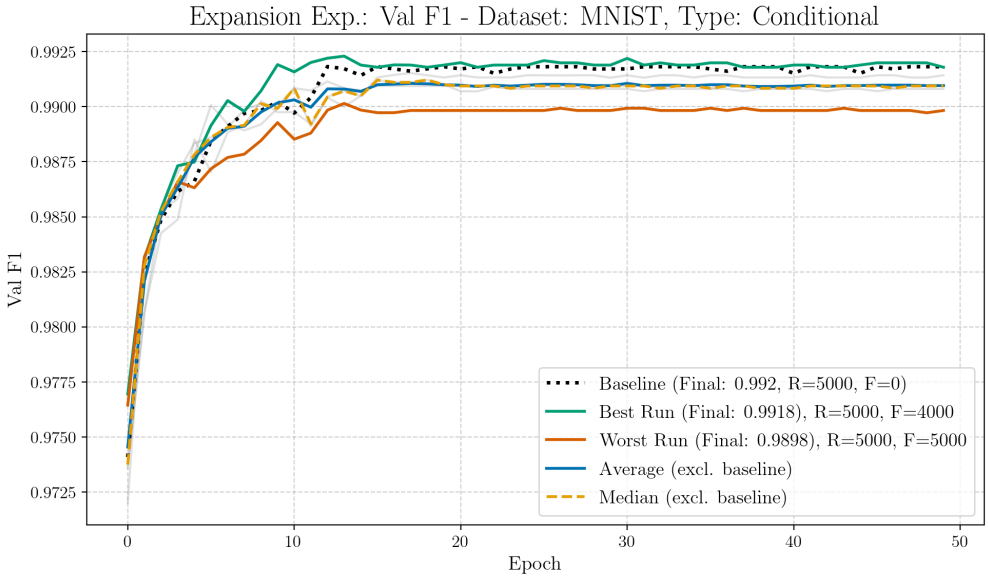
This difference in worst-case performance impacts the overall statistics. While cGAN achieves a higher median F1 score (0.9865) compared to MADGAN K=10 (0.9795), suggesting better typical performance across intermediate replacement ratios, MADGAN K=10 achieves a slightly higher average F1 score (0.9774 vs. 0.9726 for cGAN). This is because MADGAN's performance does not drop as drastically as cGAN's in the full replacement scenario. Consequently, cGAN exhibits a larger overall performance range (spread of 0.072) compared to MADGAN K=10 (spread of 0.028) in this replacement experiment.

In conclusion, when replacing real MNIST data, both cGAN and MADGAN (K=10) GDA achieve similar high peak performance with limited replacement. However, MADGAN (K=10) demonstrates superior robustness when large amounts of real data are substituted, particularly in the scenario using only synthetic data. While cGAN performs slightly better across median replacement ratios, its sharp decline in the full synthetic setting makes MADGAN (K=10) appear more consistent overall in this specific replacement task, yielding a slightly better average F1 score.

Expansion Experiment, Dataset: MNIST



(a) F1 Score on MNIST over 50 epochs. Augmentation technique: MADGAN (K=7)



(b) F1 Score on MNIST over 50 epochs. Augmentation Technique: cGAN

Run Type	Experiment	Val F1
best	$G_{7,4}$, R:5000, F:1000	0.9902
worst	$G_{7,2}$, R:5000, F:5000	0.9831
median	G (K=7)	0.9860
average	G (K=7)	0.9866

Table 14: Final F1 Scores after 50 epochs. Augmentation technique: MADGAN (K=10)

Run Type	Experiment	Val F1
best	Conditional (R:5000, F:4000)	0.9918
worst	Conditional (R:5000, F:5000)	0.9898
median	Conditional	0.9909
average	Conditional	0.9910

Table 15: Final F1 Scores after 50 epochs. Augmentation technique: cGAN

The results clearly indicate that cGAN provides superior GDA performance compared to MADGAN (K=7) in this expansion scenario on MNIST. Examining the summary statistics, cGAN outperforms MADGAN K=7 across all metrics. The best F1 score achieved with cGAN (0.9918) is higher than MADGAN's best (0.9902). More significantly, cGAN maintains high performance even when the maximum number of synthetic samples are added (worst case F1 0.9898), whereas MADGAN's worst-case performance drops considerably lower (0.9831).

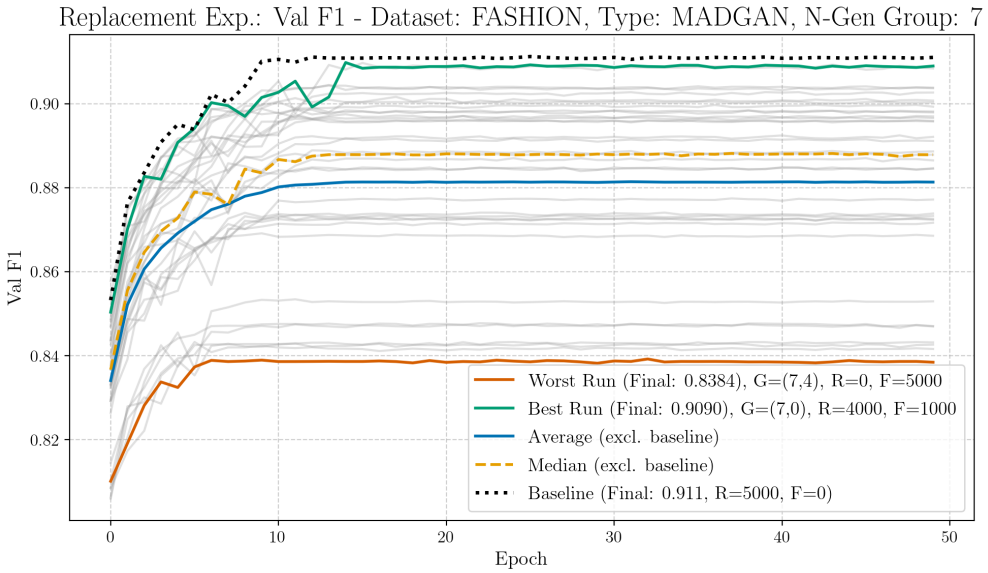
This difference in robustness is reflected in the average and median scores. cGAN achieves an average F1 of 0.9910 and a median of 0.9909, both higher than MADGAN's average (0.9866) and median (0.9860). Furthermore, cGAN exhibits much lower variability across the different expansion ratios compared to the variability observed across MADGAN's generators and expansion ratios (cGAN range 0.002 vs. MADGAN range 0.007).

Consistent with the earlier comparison against TDA, neither GDA method appears to significantly surpass their respective baseline performance achieved with 5000 real images per class alone. However, cGAN maintains performance very close to this baseline level, while adding MADGAN (K=7) samples tends to result in slightly lower F1 scores.

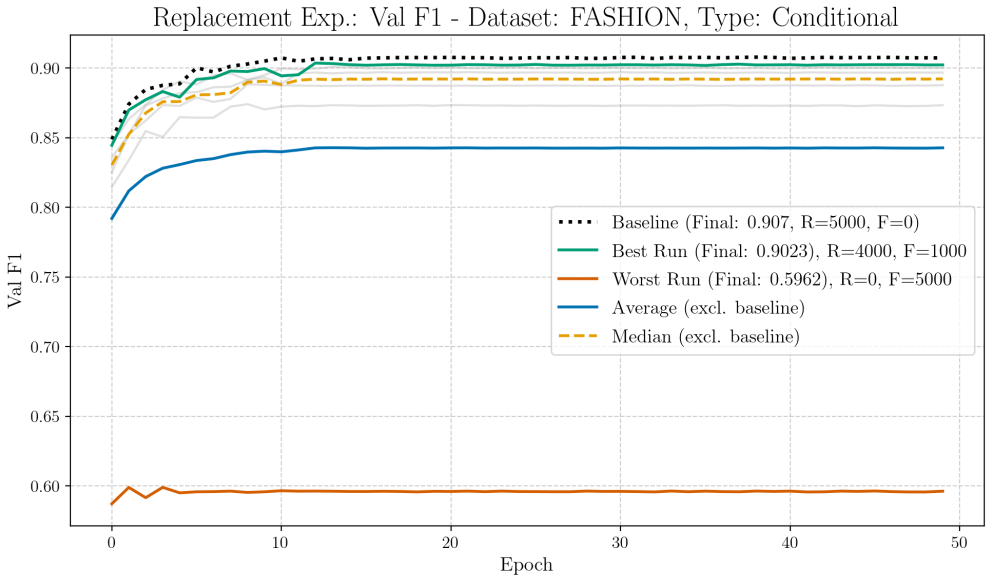
Interestingly, taking the performance of the DCGAN into account (8.3.5), it is clear, that its peak performance is in line with the highest score of the best MADGAN setup (K=7). The difference in peak F1 score is only 0.0005 ($0.9889 - 0.9884$). Furthermore, the spread of performance is significantly smaller for the DCGAN.

In conclusion, for the task of expanding the MNIST training set, cGAN GDA is demonstrably more effective and stable than MADGAN (K=7) GDA. It achieves higher peak performance, maintains significantly better performance when large amounts of synthetic data are added, and exhibits less variability compared to the multi-generator approach in this configuration.

Replacement Experiment, Dataset: Fashion-MNIST



(a) F1 Score on FASHION over 50 epochs. Augmentation tech.: MADGAN (K=7)



(b) F1 Score on FASHION over 50 epochs. Augmentation Technique: cGAN

Run Type	Experiment	Val F1
best	$G_{7,2}$, R:4000, F:1000	0.9079
worst	$G_{7,0}$, R:0, F:5000	0.3419
median	G = (K=7)	0.8927
average	G = (K=7)	0.7993

Table 16: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=7)

Run Type	Experiment	Val F1
best	Conditional (R:4000, F:1000)	0.9023
worst	Conditional (R:0, F:5000)	0.5962
median	Conditional	0.8923
average	Conditional	0.8427

Table 17: Final F1 Scores after 50 epochs. Augmentation technique: cGAN

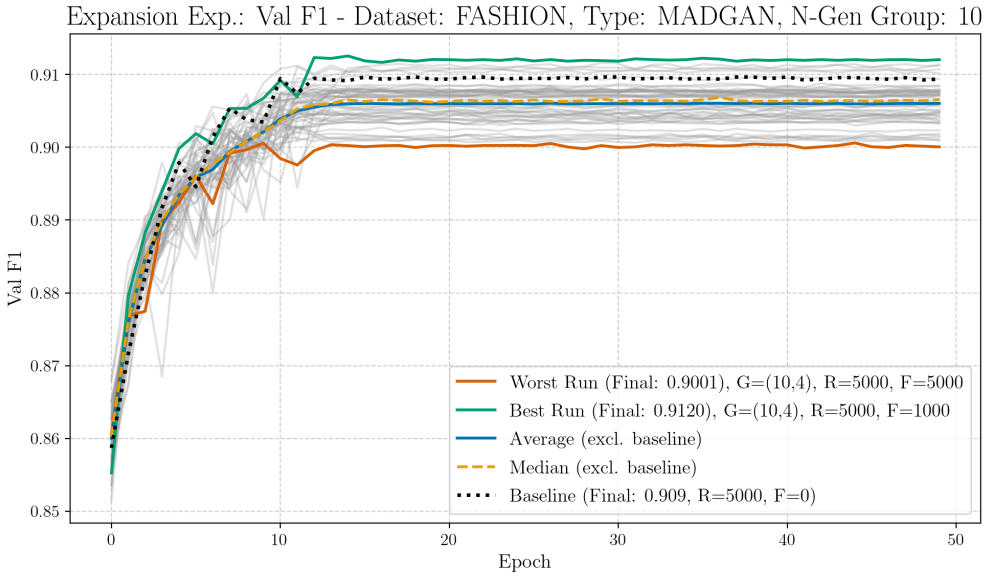
Examining the peak performances, MADGAN (K=7) achieved a slightly higher best F1 score (0.9079 with generator $G_{7,2}$ at R:4000, F:1000) compared to cGAN's best (0.9023 at the same replacement ratio). This suggests that, under optimal conditions (specific generator and limited replacement), MADGAN GDA can potentially offer a marginal advantage.

However, the performance when relying more heavily on synthetic data reveals significant differences in robustness. When all real data is replaced (R:0, F:5000), cGAN's F1 score drops to 0.5962. While this is a substantial decrease, it is considerably better than the absolute worst-case for MADGAN (K=7), where using only synthetic data from its poorest performing generator ($G_{7,0}$) resulted in an F1 score of just 0.3419.

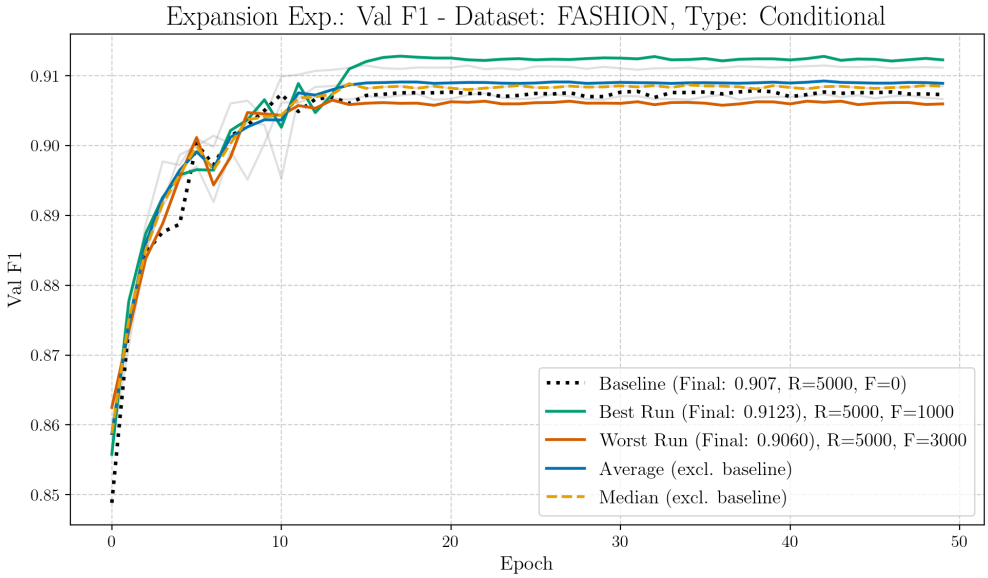
This disparity in worst-case performance heavily influences the average scores. The cGAN achieves an average F1 score of 0.8427 across all replacement ratios, which is notably higher than MADGAN K=7's average of 0.7993. Interestingly, the median F1 scores are very similar (0.8923 for cGAN and 0.8927 for MADGAN K=7), indicating that the typical performance of both methods (excluding extreme outliers) is quite comparable. Nevertheless, MADGAN (K=7) exhibits a much larger overall spread in performance (range 0.566) compared to cGAN (range 0.306), primarily due to its extremely low worst-case scores.

In conclusion, for the replacement experiment on Fashion-MNIST, MADGAN (K=7) GDA can achieve a marginally higher peak F1 score than cGAN GDA with limited data replacement. That said, cGAN demonstrates greater robustness on average and provides a significantly better worst-case performance when real data is fully substituted. The similar median scores suggest comparable typical performance, but MADGAN K=7 is less reliable overall due to the high variability among its individual generators, with some performing very poorly when used as the sole source of training data.

Expansion Experiment, Dataset: Fashion-MNIST



(a) F1 Score on FASHION over 50 epochs. Augmentation tech.: MADGAN (K=10)



(b) F1 Score on FASHION over 50 epochs. Augmentation Technique: cGAN

Run Type	Experiment	Val F1
best	$G_{10,4}$, R:5000, F:1000	0.9120
worst	$G_{10,4}$, R:5000, F:5000	0.9001
median	G (K=10)	0.9066
average	(K=10)	0.9060

Table 18: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10)

Run Type	Experiment	Val F1
best	Conditional (R:5000, F:1000)	0.9123
worst	Conditional (R:5000, F:3000)	0.9060
median	Conditional	0.9085
average	Conditional	0.9089

Table 19: Final F1 Scores after 50 epochs. Augmentation technique: cGAN

Both GDA approaches demonstrate effectiveness in the expansion scenario on Fashion-MNIST, achieving high F1 scores that suggest an improvement over a baseline trained on real data alone. The peak performances are very competitive: cGAN achieved a best F1 score of 0.9123 (when adding 1000 synthetic samples), marginally edging out MADGAN K=10's best F1 score of 0.9120 (also achieved when adding 1000 synthetic samples from its best generator, $G_{10,4}$). These top scores are notably close to the peak performance observed with TDA in previous comparisons.

However, cGAN exhibits slightly better overall consistency and robustness as more synthetic data is added. The average F1 score for cGAN across all expansion levels is 0.9089, and its median is 0.9085. These are slightly higher than MADGAN K=10's average (0.9060) and median (0.9066). More critically, cGAN's worst-case performance (F1 0.9060, when adding 3000 synthetic samples) is substantially better than MADGAN K=10's worst-case (F1 0.9001, occurring for generator $G_{10,4}$ when adding 5000 synthetic samples). This indicates that cGAN's performance degrades less and remains more stable even with larger amounts of synthetic data compared to MADGAN K=10. Consequently, cGAN shows a tighter performance spread (range 0.0063) than MADGAN K=10 (range 0.0119).

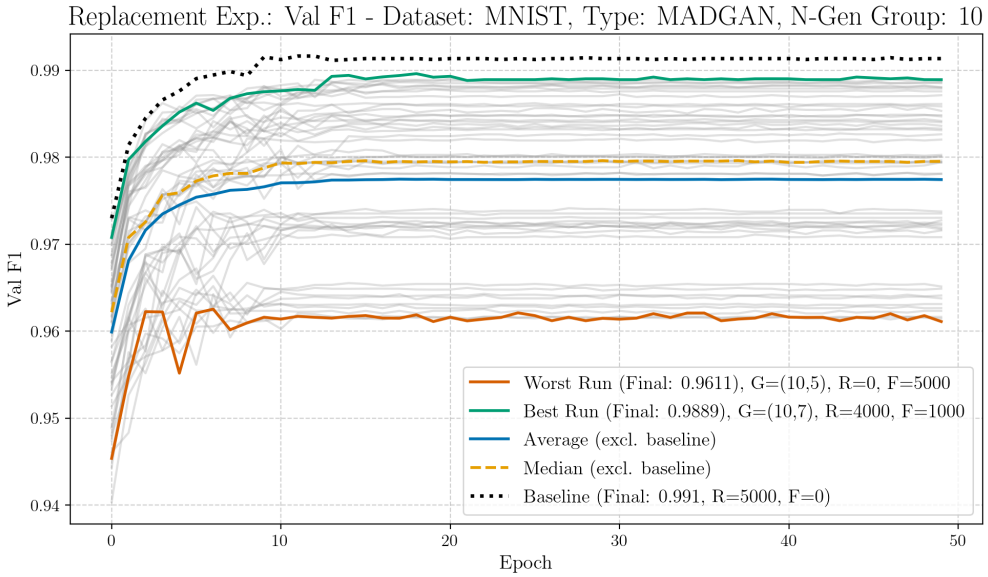
In conclusion, for expanding the Fashion-MNIST dataset, both cGAN and MADGAN (K=10) GDA can provide performance benefits, achieving F1 scores competitive with traditional augmentation methods at their peak. However, cGAN demonstrates a slight advantage in overall performance, offering higher average and median scores, and particularly showing greater stability and less degradation when larger volumes of synthetic data are incorporated.

5.2.4 MADGAN GDA vs. cMADGAN GDA Performance

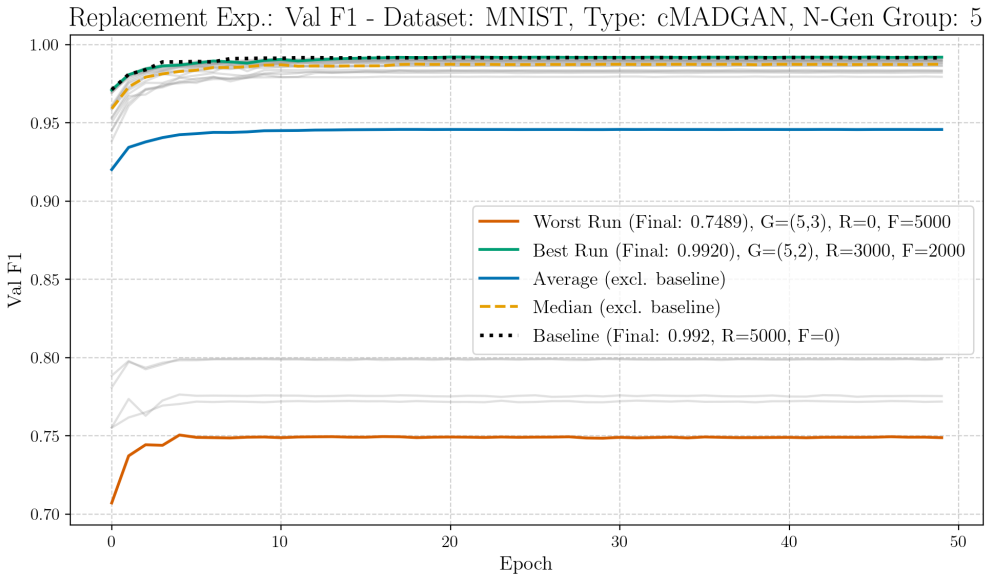
Ultimately, the final direct comparison sets MADGAN and the conditional adaptation cMADGAN side-by-side. As established before, the best selection of MADGAN experiments (K=10 for Replacement-, K=7 for Expansion scenario on MNIST and K=7 for Replacement-, K=10 for Expansion scenario on Fashion-MNIST) is directly compared against the best performing setting of cMADGAN. Orienting on the afore comparison against MADGAN vs. DCGAN/cGAN (5.2.3). Other references to cMADGAN experiments from the appendix are mentioned were reasonable.

TODO: preamble the results of the following experiments

Replacement Experiment, Dataset: MNIST



(a) F1 Score on MNIST over 50 epochs. Augmentation technique: MADGAN (K=10)



(b) F1 Score on MNIST over 50 epochs. Augmentation technique: cMADGAN (K=5)

Run Type	Experiment	Val F1
best	$G_{10,7}$, R:4000, F:1000	0.9889
worst	$G_{10,5}$, R:0, F:5000	0.9611
median	G (K=10)	0.9795
average	G (K=10)	0.9774

Table 20: Final F1 Scores after 50 epochs. Augmentation technique: MADGAN (K=10)

Run Type	Experiment	Val F1
best	$G_{5,2}$, R:3000, F:2000	0.9920
worst	$G_{5,3}$, R:0, F:5000	0.7489
median	G (K=5)	0.9874
average	G (K=5)	0.9458

Table 21: Final F1 Scores after 50 epochs. Augmentation technique: cMADGAN (K=5)

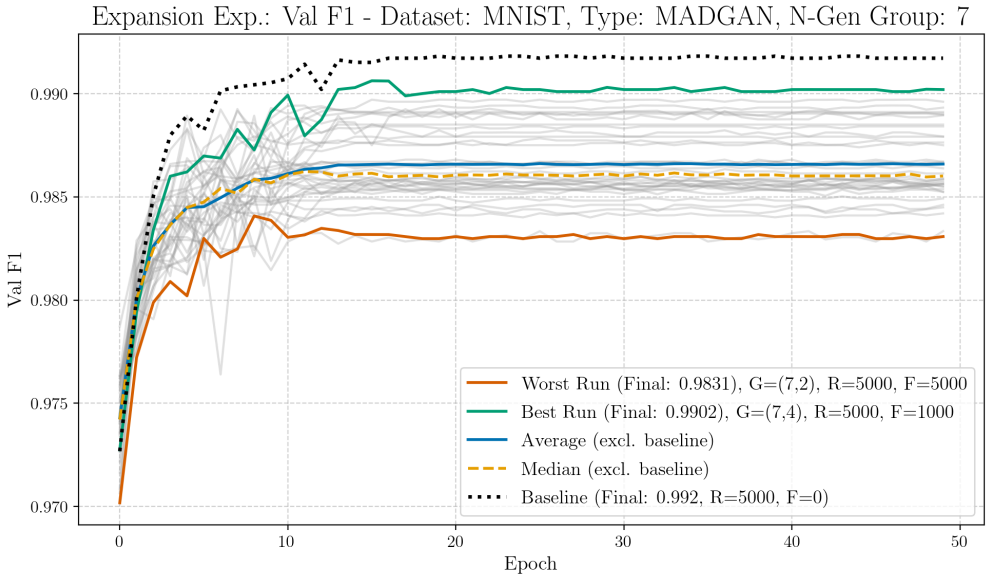
The comparison reveals a fascinating trade-off between peak performance potential and consistency. The cMADGAN (K=5) model achieved the highest individual F1 score in this set of experiments, reaching 0.9920 (with generator $G_{5,2}$ when replacing 2000 real samples with 2000 synthetic ones per class from the R:5000/F:0 baseline). This peak performance is notably high, surpassing the best score from MADGAN K=10 (0.9889) and even rivaling the top scores seen with Traditional Data Augmentation (TDA) in previous comparisons. This suggests that individual generators within the cMADGAN K=5 framework can produce highly effective synthetic data for replacement.

However, cMADGAN K=5 exhibits extreme variability. Its worst-performing run (generator $G_{5,3}$ when using only synthetic data, R:0, F:5000) resulted in a drastically low F1 score of 0.7489. This drop for at least one of its generators signifies a lack of reliability. It is to note, that four other generators resulted with similar lower scores, between 0.75 and 0.8. In contrast, MADGAN K=10, while not reaching the same peak as cMADGAN K=5, demonstrated much greater consistency. Its worst-case F1 score was 0.9611, indicating that even its poorest performing generator at full replacement maintained a high level of performance.

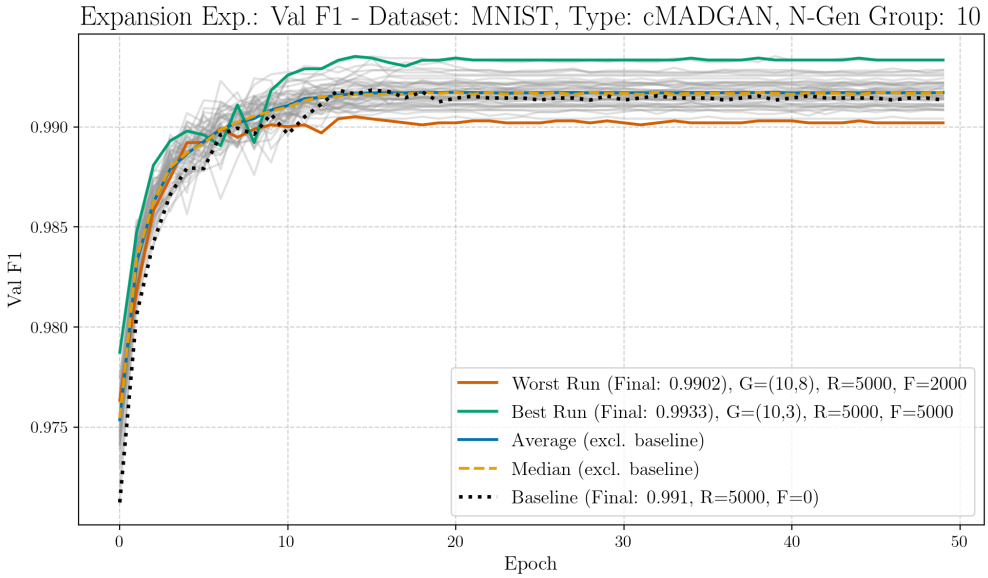
This difference in variability influences the average and median scores. MADGAN K=10 achieved a substantially better average F1 score (0.9774) compared to cMADGAN K=5 (0.9458), as the latter's average was significantly pulled down by its low outlier(s). Conversely, cMADGAN K=5 reported a higher median F1 score (0.9874) than MADGAN K=10 (0.9795), suggesting that the typical performance of a cMADGAN K=5 generator (excluding the worst cases) is very strong and even surpasses the typical MADGAN K=10 generator. The overall performance spread for cMADGAN K=5 (range 0.2431) is vastly larger than that of MADGAN K=10 (range 0.0278).

To conclude, for the replacement task on MNIST, cMADGAN (K=5) offers the potential for superior peak GDA performance from its best generators, achieving results competitive with the best augmentation methods. However, it suffers from significant inconsistency, with some generators producing very poor quality synthetic data leading to a low average performance. MADGAN (K=10) provides more reliable and stable GDA performance, with a better average F1 score and a much more dependable worst-case outcome, though its peak performance is slightly lower than the best of cMADGAN (K=5). Overall, the MADGANs reliability, proved itself, almost regardless of the number of generators used 8.3.1. cMADGAN, on the other hand, showed to be less reliable overall, consistently resulting in a couple of bad generators, in terms of their augmentation performance 8.3.2.

Expansion Experiment, Dataset: MNIST



(a) F1 Score on MNIST over 50 epochs. Augmentation tech.: MADGAN (K=7)



(b) F1 Score on MNIST over 50 epochs. Augmentation tech.: cMADGAN (K=10)

Run Type	Experiment	Val F1
best	$G_{7,4}$, R:5000, F:1000	0.9902
worst	$G_{7,2}$, R:5000, F:5000	0.9831
median	G (K=7)	0.9860
average	G (K=7)	0.9866

Table 22: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10)

Run Type	Experiment	Val F1
best	$G_{10,3}$, R:5000, F:5000	0.9933
worst	$G_{10,8}$, R:5000, F:2000	0.9902
median	G (K=10)	0.9917
average	G (K=10)	0.9917

Table 23: Final F1 Scores after 50 epochs. Augmentation tech.: cMADGAN (K=10)

The results compellingly demonstrate the superiority of cMADGAN (K=10) over MADGAN (K=7) for data expansion on MNIST. Across all summary statistics, cMADGAN (K=10) achieves markedly better and more consistent F1 scores.

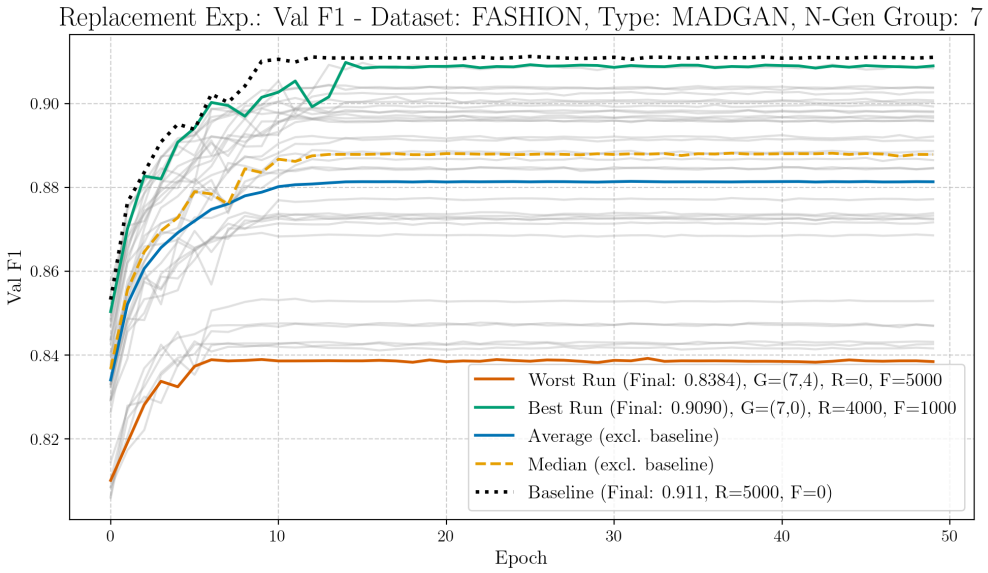
Specifically, the best F1 score attained by cMADGAN (K=10) is 0.9933 (generator $G_{10,3}$ when adding 5000 synthetic samples), which is not only significantly higher than MADGAN K=7's best of 0.9902 but is also highly competitive with the peak performance observed using Traditional Data Augmentation (TDA, previously around 0.9938). Remarkably, this peak for cMADGAN K=10 occurs at the maximum level of data expansion (F:5000), indicating a strong positive contribution from the synthetic data.

The robustness of cMADGAN K=10 is further highlighted by its worst-case performance. Its lowest F1 score across all its generators and expansion ratios is 0.9902. Strikingly, this worst score for cMADGAN K=10 is identical to the "best" score achieved by MADGAN K=7. In contrast, MADGAN K=7's performance drops to 0.9831 in its worst-case scenario.

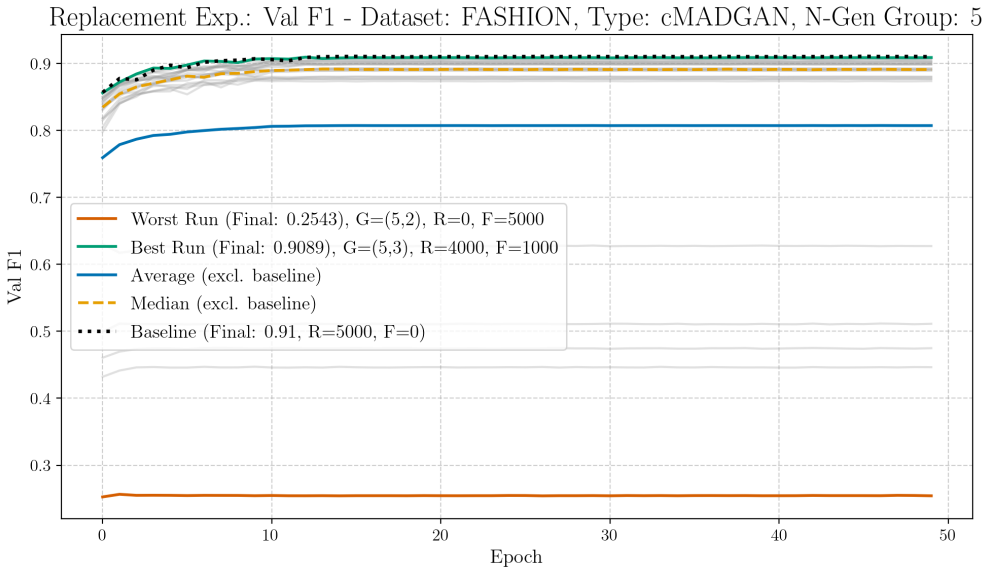
Consequently, the average (0.9917) and median (0.9917) F1 scores for cMADGAN K=10 are substantially higher than those for MADGAN K=7 (average 0.9866, median 0.9860). Moreover, cMADGAN K=10 exhibits a much tighter performance cluster, with a very small spread between its best and worst scores (range 0.0031), indicating high consistency across its generators and different levels of augmentation. This contrasts with MADGAN K=7's larger spread (range 0.0071).

In conclusion, for the MNIST expansion task, cMADGAN (K=10) GDA proves to be a significantly more effective and reliable augmentation technique than MADGAN (K=7) GDA. It not only achieves higher peak performance that rivals traditional methods but also maintains excellent scores with minimal variability even when large volumes of synthetic data are introduced, showcasing its strong potential for enhancing classifier training. Graphs in the appendix further strengthen the claims made 8.3.2.

Replacement Experiment, Dataset: Fashion-MNIST



(a) F1 Score on FASHION over 50 epochs. Augmentation tech.: MADGAN (K=7)



(b) F1 Score on FASHION over 50 epochs. Augmentation tech.: cMADGAN (K=5)

Run Type	Experiment	Val F1
best	$G_{7,0}$, R:4000, F:1000	0.9090
worst	$G_{7,4}$, R:0, F:5000	0.8384
median	G (K=7)	0.8879
average	G (K=7)	0.8813

Table 24: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=7)

Run Type	Experiment	Val F1
best	$G_{5,3}$, R:4000, F:1000	0.9089
worst	$G_{5,2}$, R:0, F:5000	0.2543
median	G (K=5)	0.8909
average	G (K=5)	0.8072

Table 25: Final F1 Scores after 50 epochs. Augmentation tech.: cMADGAN (K=5)

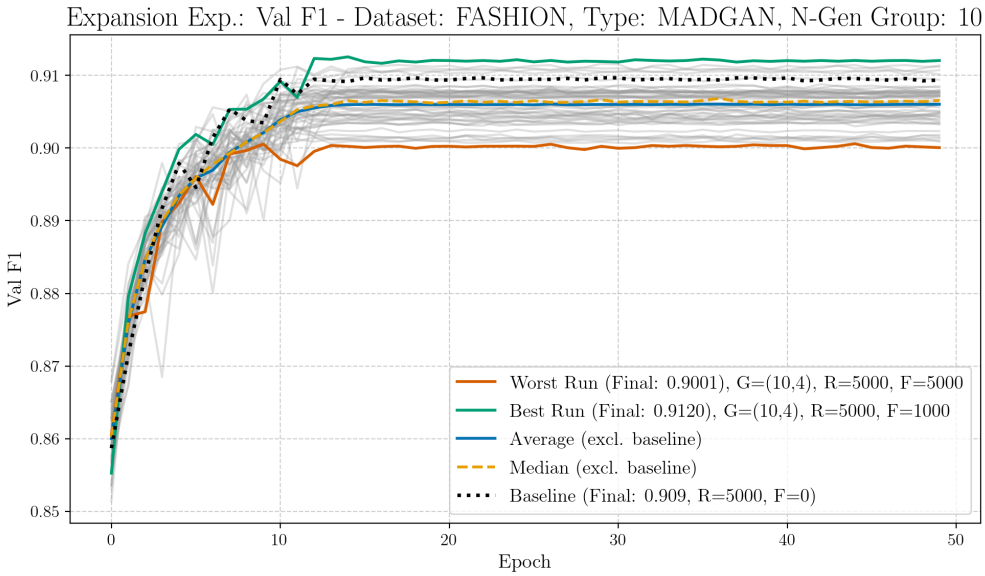
Both multi-generator approaches demonstrate the capability to achieve high peak F1 scores when a limited amount of real data is replaced. cMADGAN (K=5) achieved a slightly lower best F1 score of 0.9089 (generator $G_{5,3}$ with R:4000, F:1000), marginally falling short of MADGAN K=7's best of 0.9090 (generator $G_{7,0}$ with R:4000, F:1000). Notably, these peak performances from both models are competitive with, and even slightly exceed, the best scores previously observed with Traditional Data Augmentation (TDA) on this dataset.

However, this high peak potential is starkly contrasted by extreme inconsistency and poor performance when relying heavily on synthetic data, particularly when all real data is substituted. cMADGAN (K=5) exhibited the lowest worst-case F1 score, dropping to a critically low 0.2543 (generator $G_{5,2}$ with R:0, F:5000). While MADGAN K=7's worst-case (0.8384 from generator $G_{7,4}$ with R:0, F:5000) was also degraded, it remained significantly above cMADGAN K=5's absolute floor.

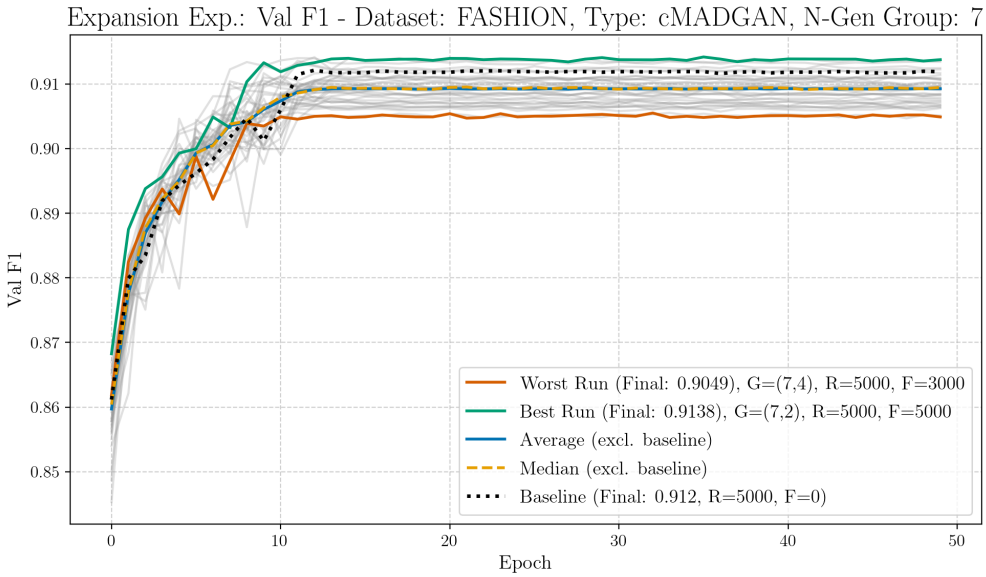
These extreme low scores significantly impact the overall assessment. Despite its lower absolute worst score, cMADGAN (K=5) managed a slightly higher median F1 score (0.8909) compared to MADGAN K=7 (0.8879). This suggests that, aside from its most extreme outliers, the bulk of cMADGAN K=5's generator/ratio combinations might perform marginally better than those of MADGAN K=7. Conversely, MADGAN K=7 shows a slightly better average F1 score (0.8813) than cMADGAN K=5 (0.8072), indicating its typical performance, discounting the severe negative outliers from both, might be marginally more consistent. Both models display very large performance spreads, with cMADGAN K=5 having a significantly wider range due to its lower floor.

Taken together, for the Fashion-MNIST replacement task, both MADGAN (K=7) and cMADGAN (K=5) can produce synthetic data from their best generators that lead to excellent, TDA-competitive classifier performance with limited replacement. Nonetheless, both frameworks suffer from extreme variability, cMADGAN more so than MADGAN. Both contain individual generators that produce very poor-quality data for this task, leading to unusable classifiers when real data is fully replaced. While cMADGAN K=5 achieved a slightly higher median, its worst-case was more severe, making both models unreliable for extensive data replacement compared to more stable traditional methods, despite their promising best-case results. Ultimately, the results show, that the presented GDA methods must be strictly controlled and carefully monitored to use them. Acting upon average results might mask single bad performances, at least in the replacement scenarios.

Expansion Experiment, Dataset: Fashion-MNIST



(a) F1 Score on FASHION over 50 epochs. Augmentation tech.: MADGAN (K=10)



(b) F1 Score on FASHION over 50 epochs. Augmentation tech.: cMADGAN (K=7)

Run Type	Experiment	Val F1
best	$G_{10,4}$, R:5000, F:1000	0.9120
worst	$G_{10,4}$, R:5000, F:5000	0.9001
median	G (K=10)	0.9066
average	(K=10)	0.9060

Table 26: Final F1 Scores after 50 epochs. Augmentation tech.: MADGAN (K=10)

Run Type	Experiment	Val F1
best	$G_{7,2}$, R:5000, F:5000	0.9138
worst	$G_{7,4}$, R:5000, F:3000	0.9049
median	G (K=7)	0.9095
average	G (K=7)	0.9093

Table 27: Final F1 Scores after 50 epochs. Augmentation tech.: cMADGAN (K=7)

The results are indicating a clear advantage for cMADGAN (K=7) over MADGAN (K=10) in the expansion scenario on Fashion-MNIST. Here, cMADGAN (K=7) is demonstrating superior performance across all metrics aggregated in the table 27, compared to 26.

cMADGAN (K=7) achieved a remarkable peak F1 score of 0.9138 (with generator $G_{7,2}$ when adding the maximum of 5000 synthetic samples per class). This is the highest F1 score recorded among all GDA methods on Fashion-MNIST in these expansion experiments and is highly competitive with the best scores from Traditional Data Augmentation (TDA, previously 0.9129). The fact that this peak occurs at maximum data expansion is particularly noteworthy. Furthermore, cMADGAN K=7's worst-case F1 score (0.9049) is impressively high, indicating strong performance even from its least effective generator/ratio combination.

In comparison, MADGAN (K=10) achieved a best F1 score of 0.9120 (generator $G_{10,4}$ with R:5000, F:1000), slightly below cMADGAN K=7's peak. More significantly, MADGAN K=10's worst-case F1 score dropped to 0.9001, which, while still good, is notably lower than cMADGAN K=7's worst case.

This superiority of cMADGAN K=7 is further reflected in its average (0.9093) and median (0.9095) F1 scores, both of which are higher than MADGAN K=10's average (0.9060) and median (0.9066). Additionally, cMADGAN (K=7) exhibited a tighter performance distribution, with a smaller spread between its best and worst scores (0.0089) compared to MADGAN K=10 (0.0119), signifying greater consistency.

In conclusion, for the Fashion-MNIST expansion task, cMADGAN (K=7) emerges as the more effective and reliable GDA technique compared to MADGAN (K=10). It not only achieves a higher peak F1 score that rivals TDA but also delivers better average performance and greater robustness, maintaining strong results. This is especially true when substantially expanding the dataset with generated samples. The MADGAN architecture classifiers, on the other hand, performed worst with the highest amount of augmented samples.

5.2.5 Effect of increasing the Generators

The following paragraphs conclude the experimental section and answers question 5 (5.1). Above experiments are used to conclude the effect of adjusting **K** (the number of generators in multi-agent GAN architectures). First, the sections focus the MADGAN- and secondly the cMADGAN architecture. To answer this question, all experiments, including those present in the appendix are taken into consideration.

MADGAN Across the datasets (MNIST, Fashion-MNIST), increasing K positively correlated with an increase of image quality and diversity, judged by the average FID and IS. Especially for the Fashion-MNIST dataset, the MADGAN with $K = 10$ performed the best. This setting achieved the lowest FID and the highest IS, excluding

the baseline 3. The results on MNIST, however, show that this trend does not fully generalize. The MADGAN with $K = 10$ does result in the lowest FID. The IS, on the other hand, is the third highest from four (2). Analyzing the scores for the single generator evaluations, the lowest FID score results from the ninth generator from the $K = 10$ experiment. The highest IS occurred in the setting with $K = 7$ (28). For MNIST the lowest FID and highest IS are created by the generator $G_{10,4}$.

Taking the Replacement and Expansion scenarios into consideration, the MADGAN architecture performed best with a value, close to the number of classes in the dataset. For the Replacement scenario on MNIST, $K = 10$ and for the Expansion scenario $K = 7$ achieved the highest F1 scores. In accordance to these results, the outcome on Fashion-MNIST was the same, with $K = 7$ and $K = 10$ performing best on the two scenarios respectively.

Overall, it can be concluded that the increase of K generally improved the performance of the MADGANs, especially for the Fashion-MNIST dataset. The effect is, however, less consistent on the MNIST dataset.

cMADGAN The same cannot be said about the cMADGAN architecture. In fact, the opposite is the case here. On MNIST, the best FID and IS can be found with $K = 3$. Here, the lowest FID score is achieved by generator $G_{3,2}$, with 22.753 and the highest IS achieved by generator $G_{3,0}$, with 2.494 (30). The averaged performance of the architecture with growing K showed a monotonic increase of the FID, pointing at a negative correlation between the number of generators and the quality of the generated images, judged by the FID. The results for the Fashion-MNIST dataset are consistent with the conclusions drawn from the MNIST dataset. The best performing setup is again found in the setting with the lowest generator count of $K = 3$. In both metrics, the FID and the IS, generator $G_{3,2}$ achieved the best results (FID: 22.753, IS: 5.049). Again pointing to the fact that smaller values for K benefit the cMADGAN architecture.

For both datasets, the opposite results as in the above paragraph must be concluded in terms of the FID and IS. Generally, the cMADGAN architecture performance decreases with an increase of K for these metrics.

Considering the Replacement and Expansion scenarios, however, points to an inconsistency with the afore mentioned results. $K = 5$ and $K = 7$ resulted in the best Replacement and Expansion scenarios on both datasets, favoring a mid to high value for K wrt the number of classes.

6 Remarks

Is Weight Sharing Mandatory in MAD-GAN?

No, weight sharing is not strictly mandatory to implement the general idea of having multiple generators aiming for diversity and fooling a single discriminator. You could implement K completely independent generator networks.

However, the weight-sharing strategy proposed by Ghosh et al. is a key architectural feature and contribution of their specific MAD-GAN formulation. In their paper, they propose sharing the parameters of the initial, deeper layers across all generators and only having separate (unshared) parameters for the final few layers of each generator.

So, while not theoretically mandatory for any multi-generator setup, it's central to the design and efficiency benefits highlighted in the original MAD-GAN paper. Deviating significantly from it means you are implementing a different variant of a multi-generator diverse GAN.

Advantages of Weight Sharing (as proposed in MAD-GAN):

Parameter Efficiency: This is a major advantage. Instead of storing and training K full generator networks, you store one large shared network base and K smaller "heads" (the final unshared layers). This drastically reduces the total number of parameters. Lower Memory Footprint: Requires less GPU memory, making it feasible to train more generators or use larger base networks. Faster Training (Potentially): Fewer parameters typically mean fewer computations per training step, potentially leading to faster epochs, although the overall training dynamic is complex. Improved Training Stability and Sample Efficiency: The shared base learns common low-level and mid-level features more effectively because it receives gradient updates influenced by the learning objectives of all generators. It essentially gets a stronger, more averaged learning signal for shared features. This shared learning can prevent individual generators from collapsing early or diverging drastically, potentially leading to more stable training. Generators benefit from features learned via the experiences of other generators through the shared base. Knowledge Transfer: Useful features learned by the shared base (e.g., basic shapes, textures common to the dataset) are immediately available to all generators, promoting faster learning of complex structures. Implicit Regularization: Sharing weights constrains the generators, acting as a form of regularization that might prevent individual generators from overfitting noise or specific data points too quickly. Disadvantages of Weight Sharing:

Limited Diversity Potential: Because a significant portion of the network is shared, the fundamental feature extraction process is common to all generators. The diversity is primarily introduced by the final few unshared layers. This might impose a ceiling on the maximum achievable diversity compared to completely independent generators, which could potentially learn radically different internal representations from scratch. Optimization Complexity: Training the shared parameters involves backpropagating

gradients derived from multiple generator heads, each with its adversarial loss and influenced by the diversity loss relative to others. Balancing these potentially conflicting gradient signals flowing into the shared base can be tricky and might require careful tuning of learning rates and the diversity weight (lambda). Poor tuning could lead to suboptimal learning in the shared base. Architectural Constraint: The fixed structure (shared base + separate heads) might not be the optimal architecture for every problem or dataset. Completely independent generators offer more flexibility in exploring different architectures for each one. Potential Bottleneck: If the shared base fails to learn good representations or gets stuck in a poor local minimum, it negatively impacts all generators simultaneously. With independent generators, there's a chance that at least one might find a better solution path independently. In summary, the weight-sharing strategy in the original MAD-GAN paper is a clever design choice offering significant efficiency and stability benefits, making it practical to train multiple generators. However, this comes at the cost of potentially limiting the absolute maximum diversity achievable and introducing specific optimization challenges compared to using fully independent generator networks. The choice depends on the trade-off between efficiency, stability, and the desired level/type of diversity for a specific application.

[ZCWD23] TODO: cite this again. they managed to apply dc gans to cifar10

7 Outlook

repair networks [TFNL22]

Measure the resulting diversity between the generatoed samples using the MS-SSIM scores [OOS17]; they did that as well

check the diversity of modes in the generated images. following the experiments shown figure 5

8 Conclusion

Data generated by a GAN, may it be a cGAN or a MADGAN, may not fully capturethe distribution characteristics of its training data. Though, generated images do visually appear realistic, they may only partially reflect the statistical characteristics of the original data. This can lead to synthetic images that appear *good* to a human inspector, but may contain amounts of noice that may interfere with a subsequent classifier.

from an information theorativa l standpoint, a generative model G trained on data X, distilling knowledge into a classifier C should not offer more information that what was already present in X.

Future research could focus on directly evaluating the impact of using MAD-GAN generated samples for augmenting various image classification datasets across different domains and comparing the resulting performance gains with those achieved by traditional and other generative augmentation techniques. Exploring methods to exert more control over the types of variations generated by MAD-GAN to specifically target weaknesses or improve the robustness of classifiers against particular types of noise or adversarial attacks would also be a valuable direction. Additionally, investigating the computational efficiency and scalability of training MAD-GAN for very large and complex datasets in the context of practical data augmentation pipelines would be crucial for its wider adoption. Finally, exploring the applicability of the MAD-GAN framework to generate diverse augmented data for other computer vision tasks beyond image classification, as well as for other data modalities such as natural language processing or audio processing, could further broaden its impact. The work by Ghosh et al. on Multi-Agent Diverse GANs represents a promising step towards leveraging the power of generative models for more effective and robust data augmentation in image classification and beyond.

after all, we tested an augmentation technique and maybe it should be treated as such. Augmentations should be drawn from a number of different techniques. Braughtly speaking, across different domains, types of classifiers, etc. there might not be a single best kind of augmentation technique. Ultimately, this is an optimization problem in and of itself and can and should be treated as such.

List of References

[ACB17] Arjovsky, Martin ; Chintala, Soumith ; Bottou, Léon: *Wasserstein GAN*. https://arxiv.org/abs/1701.07875. Version: 2017

[BNI+23] Biswas, Angona ; Nasim, MD Abdullah A. ; Imran, Al ; Sejuty, Anika T. ; Fairooz, Fabliha ; Puppala, Sai ; Talukder, Sajedul: *Generative Adversarial Networks for Data Augmentation*. https://arxiv.org/abs/2306.02019. Version: 2023

[CFB+24] Chang, A. ; Fontaine, M. C. ; Booth, S. ; Matarić, M. J. ; Nikolaidis, S.: Quality-Diversity Generative Sampling for Learning with Synthetic Data. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38 (2024), Nr. 18, S. 19805–19812

[DCLK20] Durall, Ricard ; Chatzimichailidis, Avraam ; Labus, Peter ; Keuper, Janis: *Combating Mode Collapse in GAN training: An Empirical Analysis using Hessian Eigenvalues*. https://arxiv.org/abs/2012.09673. Version: 2020

[DDS+09] Deng, Jia ; Dong, Wei ; Socher, Richard ; Li, Li-Jia ; Li, Kai ; Fei-Fei, Li: ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, S. 248–255

[DHYY17] Dong, Hao-Wen ; Hsiao, Wen-Yi ; Yang, Li-Chia ; Yang, Yi-Hsuan: *MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment*. https://arxiv.org/abs/1709.06298. Version: 2017

[DV18] Dumoulin, Vincent ; Visin, Francesco: *A guide to convolution arithmetic for deep learning*. https://arxiv.org/abs/1603.07285. Version: 2018

[FG19] Fridman, Lex ; Goodfellow, Ian: *Ian Goodfellow: Generative Adversarial Networks (GANs) | Lex Fridman Podcast #19*. Audio podcast episode, *Lex Fridman Podcast*, YouTube. https://www.youtube.com/watch?v=Z6rxFNMGdn0&t=3037s. Version: apr 2019. – Accessed: 2025-03-14

[Gav20] Gavrikov, Paul: *visualkeras*. https://github.com/paulgavrikov/visualkeras, 2020

[GKN+18] Ghosh, Arnab ; Kulharia, Viveka ; Namboodiri, Vinay ; Torr, Philip H. S. ; Dokania, Puneet K.: *Multi-Agent Diverse Generative Adversarial Networks*. https://arxiv.org/abs/1704.02906. Version: 2018

[GPAM+14] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: *Generative Adversarial Networks*. https://arxiv.org/abs/1406.2661. Version: 2014

[HBB21] Humayun, A. I. ; Balestriero, R. ; Baraniuk, R.: MaGNET: Uniform Sampling from Deep Generative Network Manifolds without Retraining. In: *arXiv preprint* arXiv:2110.08009 (2021)

[HBB22] Humayun, A. I. ; Balestriero, R. ; Baraniuk, R.: Polarity Sampling: Quality and Diversity Control of Pre-Trained Generative Networks via Singular Values. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, S. 10641–10650

[HFM22] Huang, Y. ; Fields, K. G. ; Ma, Y.: A Tutorial on Generative Adversarial Networks with Application to Classification of Imbalanced Data. In: *Statistical Analysis and Data Mining* 15 (2022), Nr. 5, S. 543–552. http://dx.doi.org/10.1002/sam.11570. – DOI 10.1002/sam.11570

[HRU+18] Heusel, Martin ; Ramsauer, Hubert ; Unterthiner, Thomas ; Nessler, Bernhard ; Hochreiter, Sepp: *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. https://arxiv.org/abs/1706.08500. Version: 2018

[IS15] Ioffe, Sergey ; Szegedy, Christian: *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. https://arxiv.org/abs/1502.03167. Version: 2015

[IZZE18] Isola, Phillip ; Zhu, Jun-Yan ; Zhou, Tinghui ; Efros, Alexei A.: *Image-to-Image Translation with Conditional Adversarial Networks*. https://arxiv.org/abs/1611.07004. Version: 2018

[JLR25] Jun Li, Wei Z. Chenyang Zhang Z. Chenyang Zhang ; Ren, Yawei: A Comprehensive Survey of Image Generation Models Based on Deep Learning. In: *Annals of Data Science* 12 (2025), February, 141–170. http://dx.doi.org/10.1007/s40745-024-00544-1. – DOI 10.1007/s40745–024–00544–1

[JLY20] Ji, Shulei ; Luo, Jing ; Yang, Xinyu: *A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions*. https://arxiv.org/abs/2011.06801. Version: 2020

[JPB22] Jeong, Jason ; Patel, B. ; Banerjee, I.: GAN augmentation for multiclass image classification using hemorrhage detection as a case-study. In: *Journal of Medical Imaging (Bellingham, Wash.)* 9 (2022), Nr. 3, S. 035504. http://dx.doi.org/10.1117/1.JMI.9.3.035504. – DOI 10.1117/1.JMI.9.3.035504

[KALL18] Karras, Tero ; Aila, Timo ; Laine, Samuli ; Lehtinen, Jaakko: *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. https://arxiv.org/abs/1710.10196. Version: 2018

[KSH12a] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E. ; Pereira, F. (Hrsg.) ; Burges, C.J. (Hrsg.) ;

Bottou, L. (Hrsg.) ; Weinberger, K.Q. (Hrsg.): *ImageNet Classification with Deep Convolutional Neural Networks*. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. Version: 2012

[KSH12b] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Communications of the ACM* 60 (2012), Nr. 6, S. 84–90. http://dx.doi.org/10.1145/3065386. – DOI 10.1145/3065386

[LBD+89] LeCun, Y. ; Boser, B. ; Denker, J. S. ; Henderson, D. ; Howard, R. E. ; Hubbard, W. ; Jackel, L. D.: Backpropagation Applied to Handwritten Zip Code Recognition. In: *Neural Computation* 1 (1989), Nr. 4, S. 541–551. http://dx.doi.org/10.1162/neco.1989.1.4.541. – DOI 10.1162/neco.1989.1.4.541

[LCB10] LeCun, Yann ; Cortes, Corinna ; Burges, CJ: MNIST handwritten digit database. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010)

[LLYSYSGEK20] Lu Lu, Lu L. ; Yeonjong Shin, Yeonjong S. ; Yanhui Su, Yanhui S. ; George Em Karniadakis, George Em K.: Dying ReLU and Initialization: Theory and Numerical Examples. In: *Communications in Computational Physics* 28 (2020), Januar, Nr. 5, 1671–1706. http://dx.doi.org/10.4208/cicp.oa-2020-0165. – DOI 10.4208/cicp.oa–2020–0165. – ISSN 1815–2406

[LMWN22] Li, Xiaomin ; Metsis, Vangelis ; Wang, Huangyingrui ; Ngu, Anne Hee H.: *TTS-GAN: A Transformer-based Time-Series Generative Adversarial Network*. https://arxiv.org/abs/2202.02691. Version: 2022

[LTH+17] Ledig, Christian ; Theis, Lucas ; Huszar, Ferenc ; Caballero, Jose ; Cunningham, Andrew ; Acosta, Alejandro ; Aitken, Andrew ; Tejani, Alykhan ; Totz, Johannes ; Wang, Zehan ; Shi, Wenzhe: *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. https://arxiv.org/abs/1609.04802. Version: 2017

[Mau25] Maucher, Johannes: *Markdown Convolutional Neural Network*. https://gitlab.mi.hdm-stuttgart.de/maucher/KI/-/blob/master/nb/N03ConvolutionNeuralNetworks.md, 2025. – Version: Mai 2025

[MKKY18] Miyato, Takeru ; Kataoka, Toshiki ; Koyama, Masanori ; Yoshida, Yuichi: *Spectral Normalization for Generative Adversarial Networks*. https://arxiv.org/abs/1802.05957. Version: 2018

[MLX+17] Mao, Xudong ; Li, Qing ; Xie, Haoran ; Lau, Raymond Y. K. ; Wang, Zhen ; Smolley, Stephen P.: *Least Squares Generative Adversarial Networks*. https://arxiv.org/abs/1611.04076. Version: 2017

[MO14] Mirza, Mehdi ; Osindero, Simon: *Conditional Generative Adversarial Nets*. https://arxiv.org/abs/1411.1784. Version: 2014

[Neu28] Neumann, John von: Zur Theorie der Gesellschaftsspiele. In: *Mathematische Annalen* 100 (1928), Nr. 1, S. 295–320

[NS67] Nagy, George ; Shelton, Henry: Self-Corrective Character Recognition System. In: *IBM Journal of Research and Development* 11 (1967), Nr. 6, S. 612–628. http://dx.doi.org/10.1147/rd.116.0612. – DOI 10.1147/rd.116.0612

[OOS17] Odena, Augustus ; Olah, Christopher ; Shlens, Jonathon: Conditional image synthesis with auxiliary classifier GANs. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, JMLR.org, 2017 (ICML'17), S. 2642–2651

[PKD+16] Pathak, Deepak ; Krahenbuhl, Philipp ; Donahue, Jeff ; Darrell, Trevor ; Efros, Alexei A.: *Context Encoders: Feature Learning by Inpainting*. https://arxiv.org/abs/1604.07379. Version: 2016

[PW17] Perez, Luis ; Wang, Jason: *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. https://arxiv.org/abs/1712.04621. Version: 2017

[RAY+16] Reed, Scott ; Akata, Zeynep ; Yan, Xinchen ; Logeswaran, Lajanugen ; Schiele, Bernt ; Lee, Honglak: *Generative Adversarial Text to Image Synthesis*. https://arxiv.org/abs/1605.05396. Version: 2016. – arXiv:1605.05396

[RCF25] Ribas, Lucas C. ; Casaca, Wallace ; Fares, Ricardo T.: Conditional Generative Adversarial Networks and Deep Learning Data Augmentation: A Multi-Perspective Data-Driven Survey Across Multiple Application Fields and Classification Architectures. In: *AI* 6 (2025), Nr. 2. http://dx.doi.org/10.3390/ai6020032. – DOI 10.3390/ai6020032. – ISSN 2673–2688

[RMC16] Radford, Alec ; Metz, Luke ; Chintala, Soumith: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016

[SGZ+16] Salimans, Tim ; Goodfellow, Ian ; Zaremba, Wojciech ; Cheung, Vicki ; Radford, Alec ; Chen, Xi: *Improved Techniques for Training GANs*. https://arxiv.org/abs/1606.03498. Version: 2016

[SK19] Shorten, Connor ; Khoshgoftaar, Taghi M.: A survey on Image Data Augmentation for Deep Learning. In: *Journal of Big Data* 6 (2019), July, Nr. 1, 60. http://dx.doi.org/10.1186/s40537-019-0197-0. – DOI 10.1186/s40537–019–0197–0. – ISSN 2196–1115

[SSP03] Simard, Patrice Y. ; Steinkraus, Dave ; Platt, John C.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: *Seventh International Conference on Document Analysis and Recognition*, 2003, S. 958–963

[SVI+16] Szegedy, Christian ; Vanhoucke, Vincent ; Ioffe, Sergey ; Shlens, Jon ; Wojna, Zbigniew: Rethinking the Inception Architecture for Computer Vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 2818–2826

[TFNL22] Tanno, Ryutaro ; F.Pradier, Melanie ; Nori, Aditya ; Li, Yingzhen: Repairing Neural Networks by Leaving the Right Past Behind. (2022), 07, S. 19. http://dx.doi.org/10.48550/arXiv.2207.04806. – DOI 10.48550/arXiv.2207.04806

[WM21] Wickramaratne, S. D. ; Mahmud, M. S.: Conditional-GAN Based Data Augmentation for Deep Learning Task Classifier Improvement Using fNIRS Data. In: *Frontiers in Big Data* 4 (2021), S. 659146. http://dx.doi.org/10.3389/fdata.2021.659146. – DOI 10.3389/fdata.2021.659146

[WWR+23] Wang, Hanyu ; Wu, Pengxiang ; Rosa, Kevin D. ; Wang, Chen ; Shrivastava, Abhinav: *Multimodality-guided Image Style Transfer using Cross-modal GAN Inversion*. https://arxiv.org/abs/2312.01671. Version: 2023

[WZZ+13] Wan, Li ; Zeiler, Matthew ; Zhang, Sixin ; Le Cun, Yann ; Fergus, Rob ; Dasgupta, Sanjoy (Hrsg.) ; McAllester, David (Hrsg.): *Regularization of Neural Networks using DropConnect*. https://proceedings.mlr.press/v28/wan13.html. Version: 17–19 Jun 2013 (Proceedings of Machine Learning Research)

[XRV17] Xiao, Han ; Rasul, Kashif ; Vollgraf, Roland: *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017

[XSCIV19] Xu, Lei ; Skoularidou, Maria ; Cuesta-Infante, Alfredo ; Veeramachaneni, Kalyan: *Modeling Tabular data using Conditional GAN*. https://arxiv.org/abs/1907.00503. Version: 2019

[Yin19] Ying, Xue: An Overview of Overfitting and its Solutions. In: *Journal of Physics: Conference Series* 1168 (2019), feb, Nr. 2, 022022. http://dx.doi.org/10.1088/1742-6596/1168/2/022022. – DOI 10.1088/1742–6596/1168/2/022022

[YZWY17] Yu, Lantao ; Zhang, Weinan ; Wang, Jun ; Yu, Yong: *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. https://arxiv.org/abs/1609.05473. Version: 2017

[ZCWD23] Zhao, Gaochang ; Cai, Zhao ; Wang, Xin ; Dang, Xiaohu: GAN Data Augmentation Methods in Rock Classification. In: *Applied Sciences* 13 (2023), Nr. 9, S. 5316. http://dx.doi.org/10.3390/app13095316. – DOI 10.3390/app13095316

Appendix

8.1 Network Architectures

The following section includes graphical representations of NN referenced throughout the thesis.

8.1.1 Classifiers

The graphical representations of the network architectures are created with the tool *visual keras*, by Paul Gavrikov ([Gav20]).

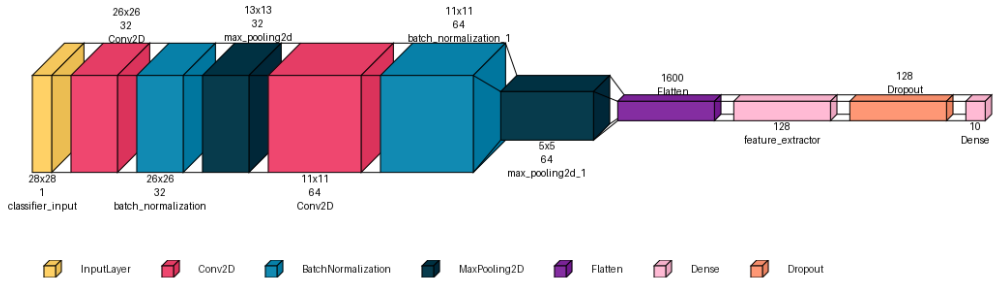


Figure 17: Depiction of the CNN architecture used to classify unlabeled images from the MNIST GDA experiments and judge the effectiveness of said GDA. (Image created with [Gav20])

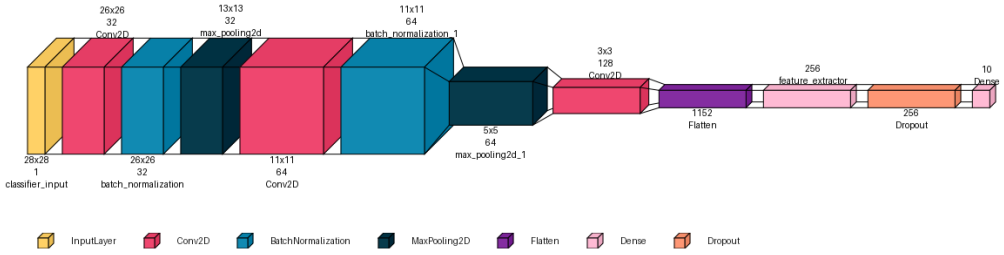


Figure 18: Depiction of the CNN architecture used to classify unlabeled images from the Fashion-MNIST GDA experiments and judge the effectiveness of said GDA. (Image created with [Gav20])

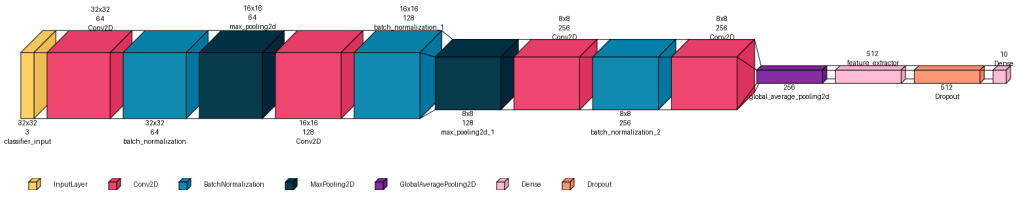


Figure 19: Depiction of the CNN architecture used to classify unlabeled images from the CIFAR10 GDA experiments and judge the effectiveness of said GDA. (Image created with [Gav20])

8.1.2 Generator Model Architectures

The graphical representations of the network architectures are created with the tool *visual keras*, by Paul Gavrikov ([Gav20]).

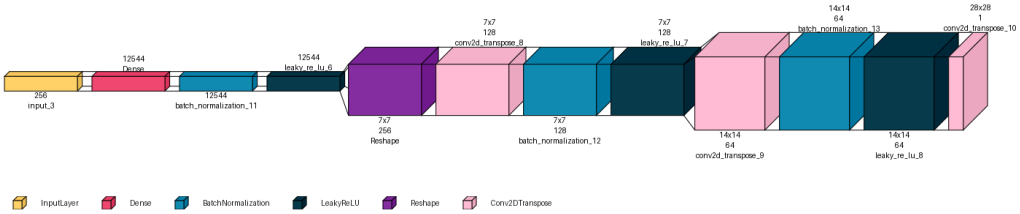


Figure 20: Depiction of the generator used in the deep convolutional GAN dependent experiments. Used to train a generator and create fake image data based on the MNIST dataset.

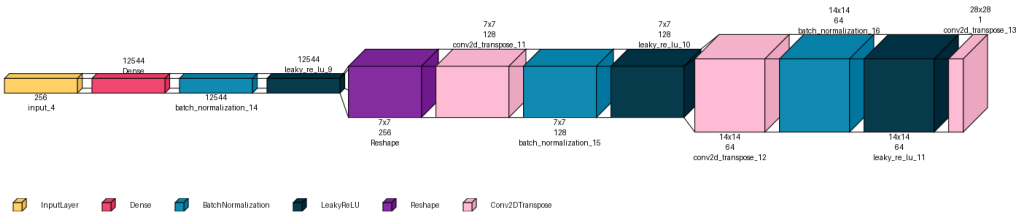


Figure 21: Depiction of the generator used in the deep convolutional GAN dependent experiments. Used to train a generator and create fake image data based on the Fashion-MNIST dataset.

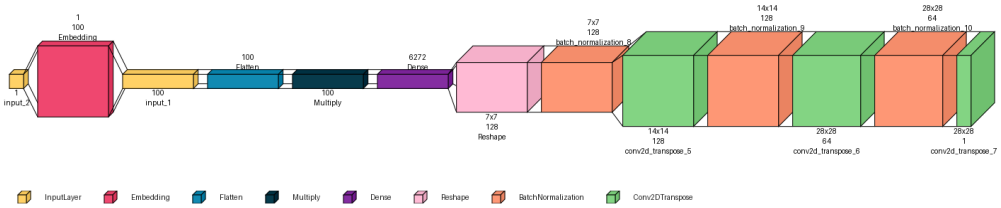


Figure 22: Depiction of the generator used in the Conditional GAN dependent experiments. Used to train a generator and create fake image data based on the MNIST and Fashion-MNIST datasets.

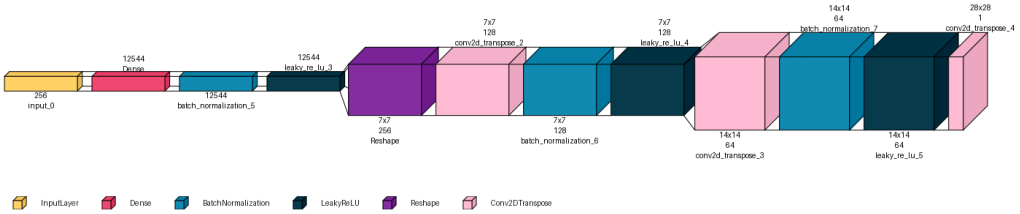


Figure 23: Depiction of the generators used in the MADGAN dependent experiments. Used to train a generator and create fake image data based on the MNIST and Fashion-

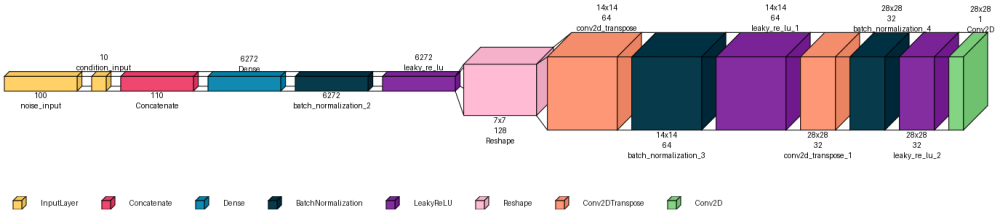


Figure 24: Depiction of the generators used in the cMADGAN dependent experiments. Used to train a generator and create fake image data based on the MNIST and Fashion-MNIST datasets.

8.1.3 Discriminator Model Architectures

The graphical representations of the network architectures are created with the tool *visual keras*, by Paul Gavrikov ([Gav20]).

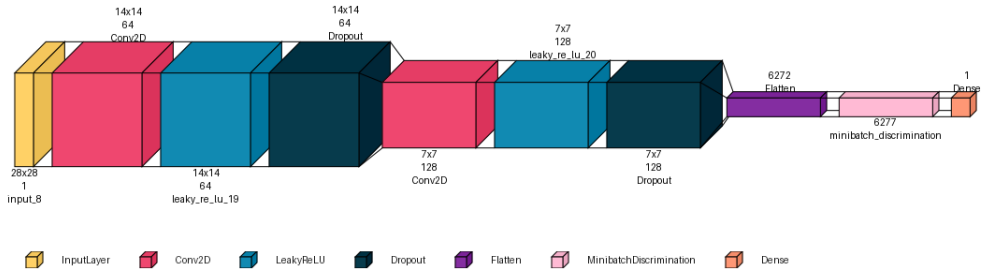


Figure 25: Depiction of the discriminator used in the deep convolutional GAN dependent experiments. Used to train a generator based on the MNIST dataset.

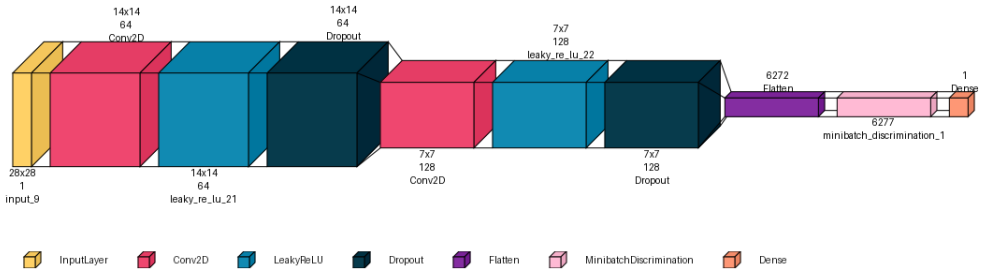


Figure 26: Depiction of the discriminator used in the deep convolutional GAN dependent experiments. Used to train a generator based on the Fashion-MNIST dataset.

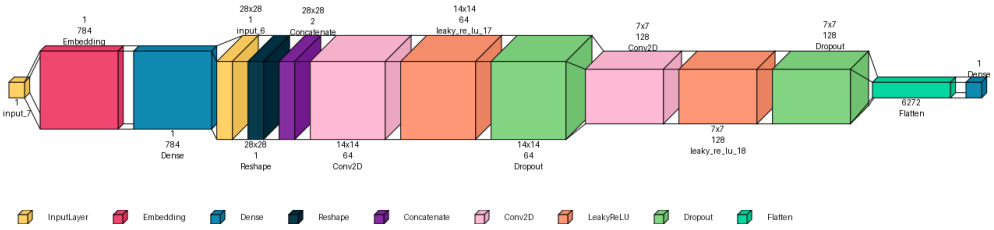


Figure 27: Depiction of the discriminator used in the Conditional GAN dependent experiments. Used to train a generator based on the MNIST and Fashion-MNIST datasets.

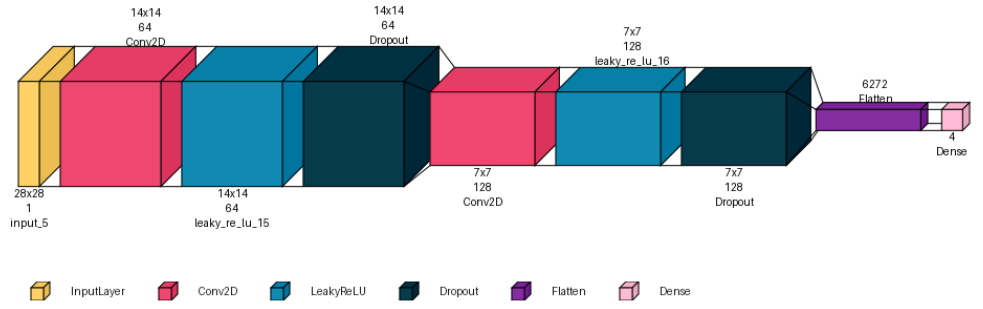


Figure 28: Depiction of the discriminator used in the MADGAN dependent experiments. Used to train a generator based on the MNIST and Fashion-MNIST datasets.

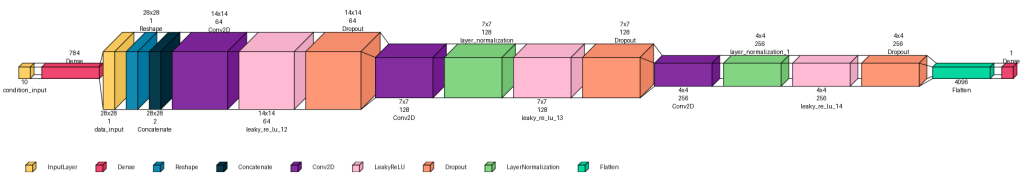


Figure 29: Depiction of the discriminator used in the cMADGAN dependent experiments. Used to train a generator based on the MNIST and Fashion-MNIST datasets.

8.2 FID and Inception Scores from MADGAN Architectures

8.2.1 MADGAN MNIST

N Generators	Index	FID	IS	IS-std
Baseline		−0.004	2.549	0.036
3	0	23.597	2.519	0.050
	1	22.682	2.506	0.027
	2	23.252	2.509	0.045
5	0	23.304	2.467	0.030
	1	22.020	2.446	0.040
	2	23.076	2.497	0.046
	3	22.265	2.467	0.060
	4	22.614	2.479	0.045
7	0	22.487	2.513	0.031
	1	21.076	**2.548**	0.087
	2	20.941	2.530	0.056
	3	22.250	2.520	0.035
	4	21.557	2.530	0.051
	5	21.297	2.545	0.035
	6	21.582	2.541	0.059
10	0	21.338	2.456	0.045
	1	20.872	2.462	0.040
	2	20.680	2.477	0.028
	3	21.315	2.470	0.030
	4	21.158	2.484	0.027
	5	20.655	2.473	0.037
	6	21.090	2.452	0.039
	7	20.689	2.508	0.048
	8	21.471	2.480	0.044
	9	**20.459**	2.477	0.036

Table 28: Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the MADGAN model on FID and Inception Score (IS ± Std. Dev) for the **MNIST** dataset. Results for each generator index are presented alongside baseline metrics.

8.2.2 MADGAN Fashion-MNIST

N Generators	Index	FID	IS	IS-std
Baseline		−0.004	4.723	0.056
3	0	26.086	4.463	0.103
	1	25.884	4.487	0.099
	2	26.637	4.538	0.094
5	0	23.393	4.467	0.119
	1	24.756	4.520	0.107
	2	25.663	4.506	0.123
	3	23.614	4.487	0.082
	4	23.665	4.504	0.061
7	0	22.370	4.558	0.111
	1	21.838	4.536	0.087
	2	21.923	4.508	0.101
	3	25.403	4.517	0.112
	4	31.493	4.372	0.087
	5	22.158	4.595	0.075
	6	21.942	4.577	0.087
10	0	22.454	4.534	0.117
	1	21.914	4.524	0.121
	2	21.327	4.494	0.089
	3	21.630	4.455	0.063
	4	**20.851**	**4.575**	0.097
	5	21.268	4.561	0.048
	6	22.324	4.566	0.127
	7	21.633	4.542	0.084
	8	21.503	4.533	0.127
	9	20.963	4.552	0.100

Table 29: Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the MADGAN model on FID and Inception Score (IS ± Std. Dev) for the **Fashion-MNIST** dataset. Results for each generator index are presented alongside baseline metrics.

8.2.3 cMADGAN MNIST

N Generators	Index	FID	IS	IS-std
Baseline		−0.004	2.549	0.036
3	0	30.105	**2.494**	0.034
	1	23.875	2.317	0.041
	2	**22.753**	2.384	0.032
5	0	37.638	2.460	0.039
	1	26.614	2.378	0.052
	2	26.739	2.356	0.025
	3	27.642	2.309	0.034
	4	26.722	2.246	0.043
a	0	23.141	2.418	0.037
	1	28.071	2.363	0.036
	2	33.490	2.340	0.033
7	3	28.333	2.347	0.031
	4	35.599	2.296	0.050
	5	36.075	2.389	0.047
	6	29.807	2.326	0.036
10	0	108.079	2.076	0.011
	1	82.478	2.250	0.022
	2	153.369	2.124	0.023
	3	126.574	1.846	0.005
	4	110.012	2.005	0.013
	5	130.054	1.861	0.009
	6	103.016	2.010	0.011
	7	74.926	2.383	0.035
	8	110.265	2.086	0.014
	9	106.762	1.975	0.033

Table 30: Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the cMADGAN model on FID and Inception Score (IS ± Std. Dev) for the **MNIST** dataset. Results for each generator index are presented alongside baseline metrics.

8.2.4 cMADGAN Fashion-MNIST

N Generators	Index	FID	IS	IS-std
Baseline		−0.004	4.723	0.056
3	0	25.874	4.750	0.132
	1	27.235	4.068	0.090
	2	**23.556**	**5.049**	0.093
5	0	170.120	2.938	0.020
	1	167.446	3.430	0.052
	2	221.934	3.325	0.032
	3	187.293	3.269	0.029
	4	53.616	3.770	0.057
7	0	156.601	2.392	0.016
	1	172.054	3.325	0.033
	2	171.505	2.754	0.024
	3	97.514	3.273	0.058
	4	155.474	3.157	0.063
	5	151.494	2.623	0.018
	6	174.161	2.976	0.028
10	0	147.670	3.650	0.037
	1	167.407	2.816	0.026
	2	109.349	3.914	0.068
	3	160.305	3.674	0.040
	4	181.778	2.961	0.023
	5	154.633	3.297	0.022
	6	166.511	2.759	0.034
	7	156.135	3.173	0.027
	8	155.758	3.853	0.036
	9	191.126	3.073	0.034

Table 31: Effect of varying the number of generators ($N = 3, 5, 7, 10$) in the cMADGAN model on FID and Inception Score (IS ± Std. Dev) for the **Fashion-MNIST** dataset. Results for each generator index are presented alongside baseline metrics.

8.2.5 DCGAN MNIST

N Generators	FID	IS	IS-std
Baseline	−0.004	2.549	0.036
1	122.097	2.611	0.056

Table 32: FID and Inception Score (Mean ± Std. Dev) for a single DCGAN generator ($N = 1$) trained on the **MNIST** dataset. Baseline scores are included for reference.

8.2.6 DCGAN Fashion-MNIST

N Generators	FID	IS	IS-std
Baseline	−0.004	4.723	0.056
1	123.349	3.573	0.117

Table 33: FID and Inception Score (Mean ± Std. Dev) for a single DCGAN generator ($N = 1$) trained on the **Fashion-MNIST** dataset. Baseline scores are included for reference.

8.2.7 Conditional MNIST

N Generators	FID	IS	IS-std
Baseline	−0.004	2.549	0.036
1	28.721	2.553	0.022

Table 34: FID and Inception Score (Mean ± Std. Dev) for a single Conditional GAN generator ($N = 1$) trained on the **MNIST** dataset. Baseline scores are included for reference.

8.2.8 Conditional Fashion-MNIST

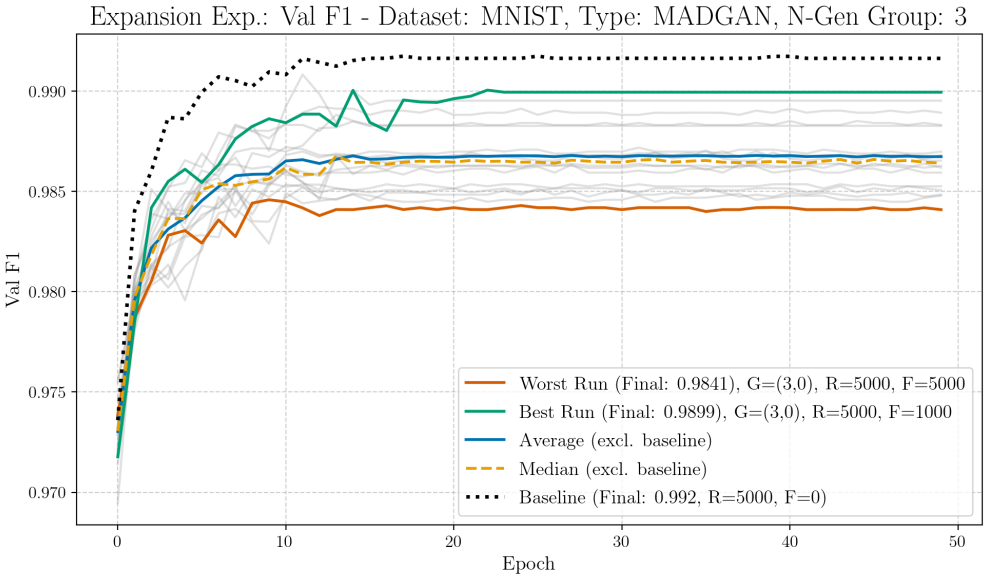
N Generators	FID	IS	IS-std
Baseline	−0.004	4.723	0.056
1	25.560	4.210	0.099

Table 35: FID and Inception Score (Mean ± Std. Dev) for a single Conditional GAN generator ($N = 1$) trained on the **Fashion-MNIST** dataset. Baseline scores are included for reference.

8.3 Stratified Classifier Performances and Graphs

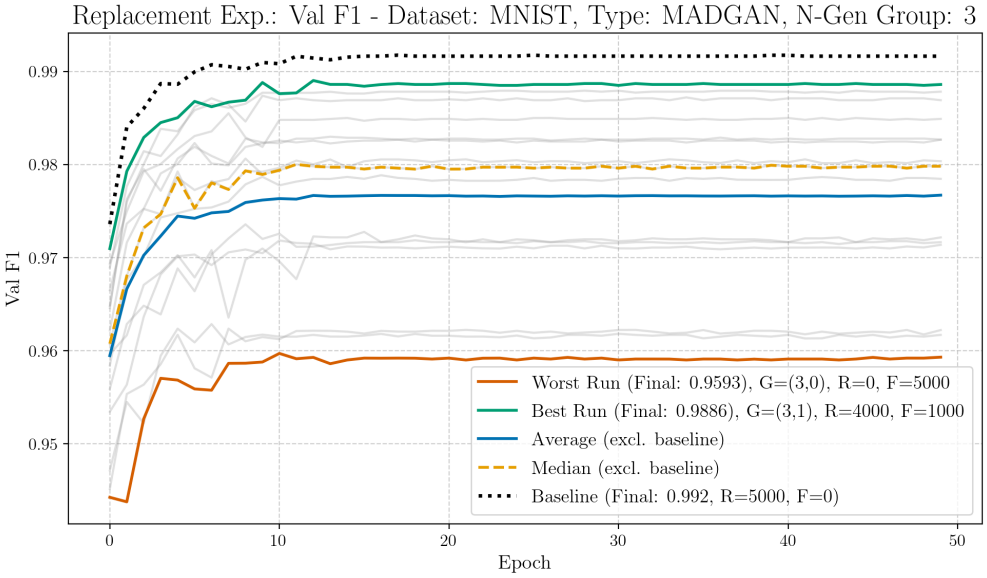
8.3.1 Dataset: MNIST, Architecture: MADGAN

Expansion Experiment: K=3



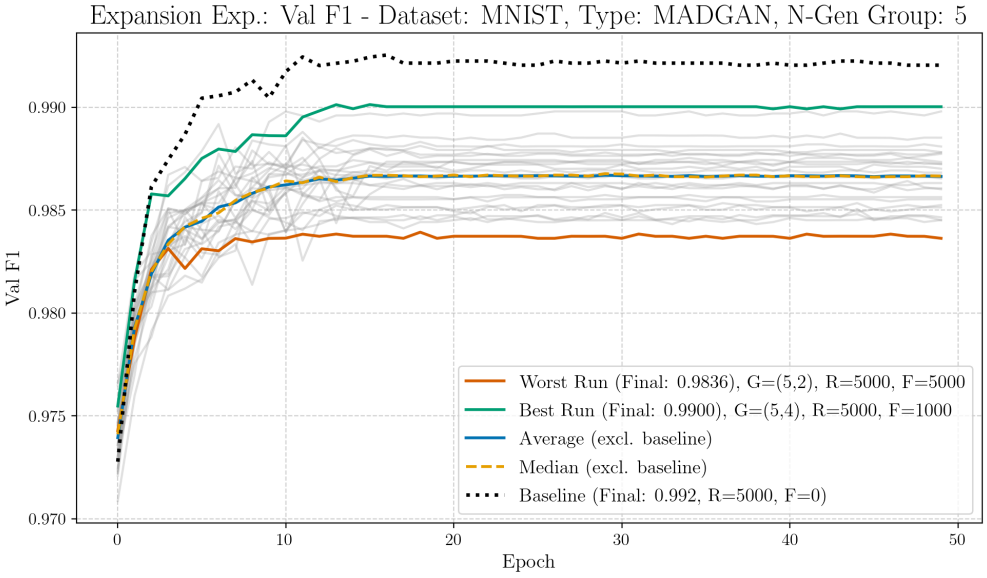
Run Type	Experiment	Val F1
best	$G_{3,0}$, R:5000, F:1000	0.9899
worst	$G_{3,0}$, R:5000, F:5000	0.9841
median	G (K=3)	0.9864
average	G (K=3)	0.9867

Replacement Experiment: K=3



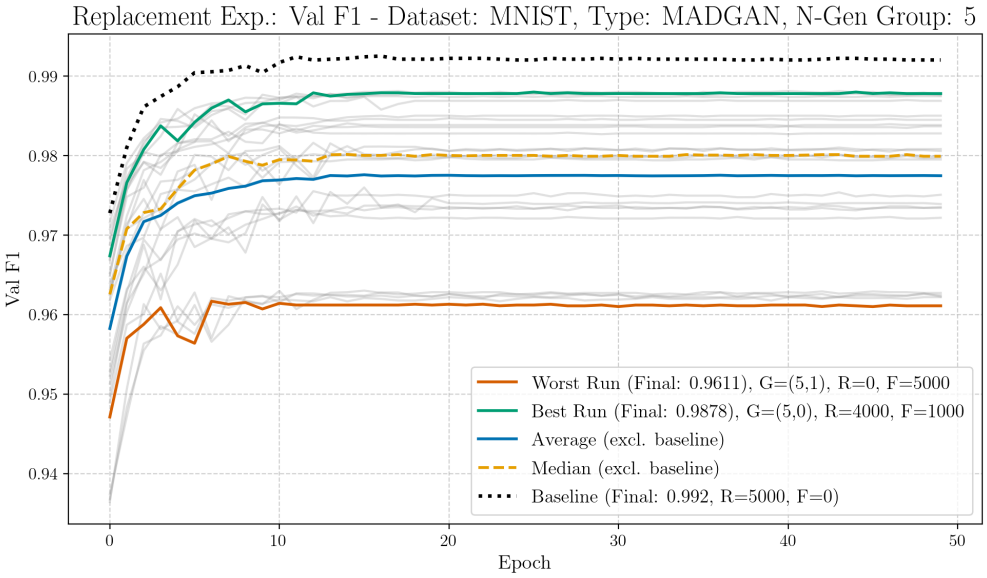
Run Type	Experiment	Val F1
best	$G_{3,1}$, R:4000, F:1000	0.9886
worst	$G_{3,0}$, R:0, F:5000	0.9593
median	G (K=3)	0.9798
average	G (K=3)	0.9767

Expansion Experiment: K=5



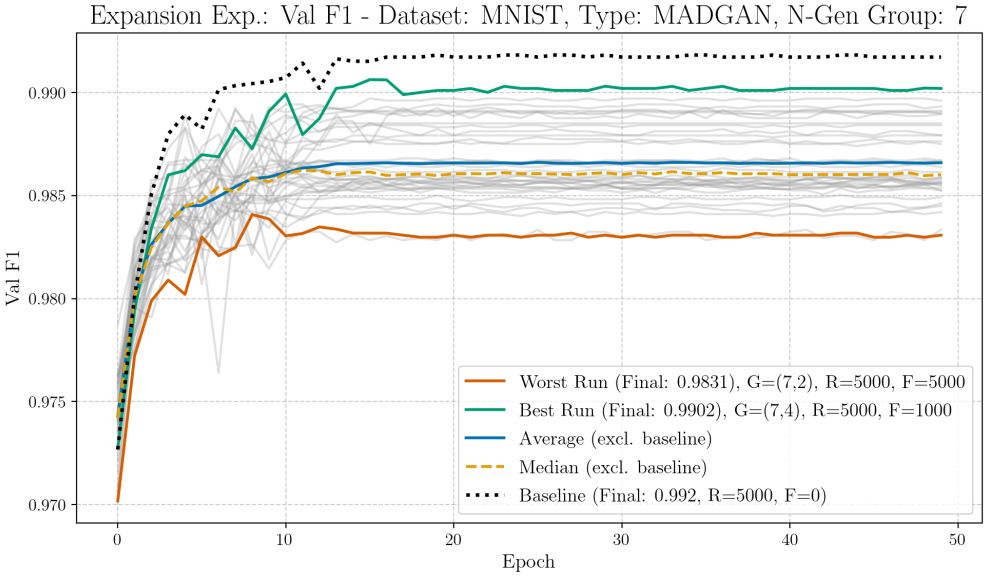
Run Type	Experiment	Val F1
best	$G_{5,4}$, R:5000, F:1000	0.9900
worst	$G_{5,2}$, R:5000, F:5000	0.9836
median	G (K=5)	0.9867
average	G (K=5)	0.9866

Replacement Experiment: K=5



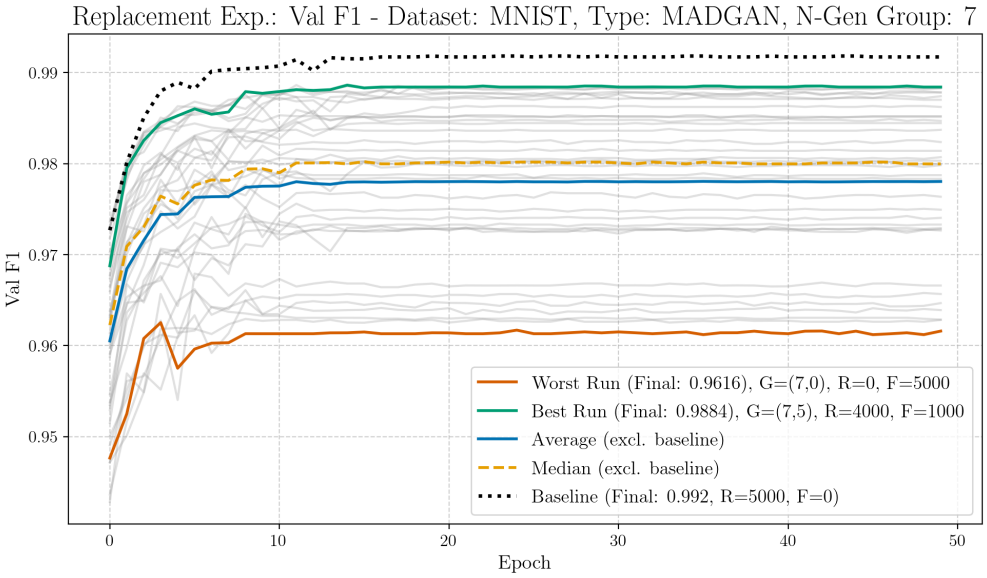
Run Type	Experiment	Val F1
best	$G_{5,0}$, R:4000, F:1000	0.9878
worst	$G_{5,1}$, R:0, F:5000	0.9611
median	G (K=5)	0.9799
average	G (K=5)	0.9775

Expansion Experiment: K=7



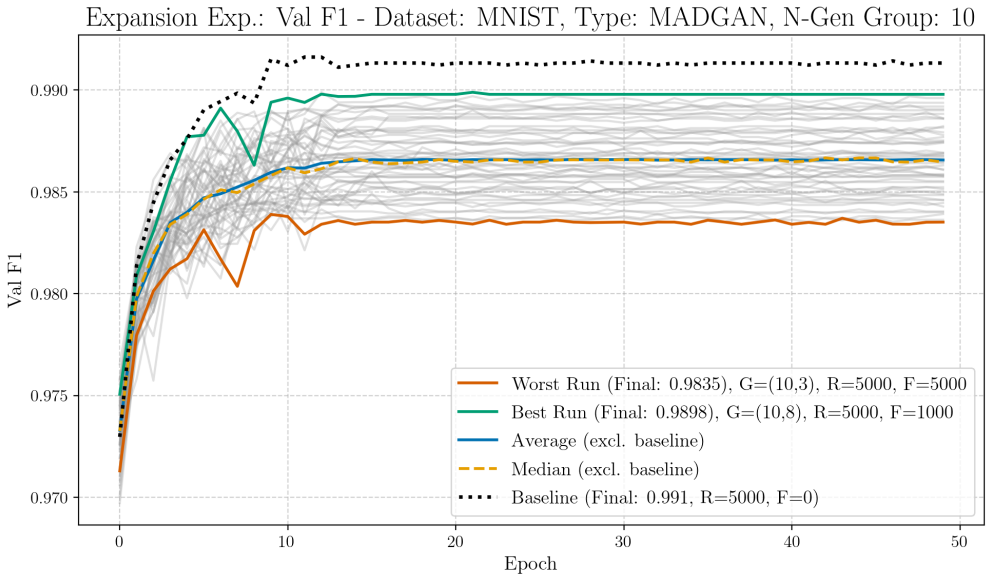
Run Type	Experiment	Val F1
best	$G_{7,4}$, R:5000, F:1000	0.9902
worst	$G_{7,2}$, R:5000, F:5000	0.9831
median	G (K=7)	0.9860
average	G (K=7)	0.9866

Replacement Experiment: K=7



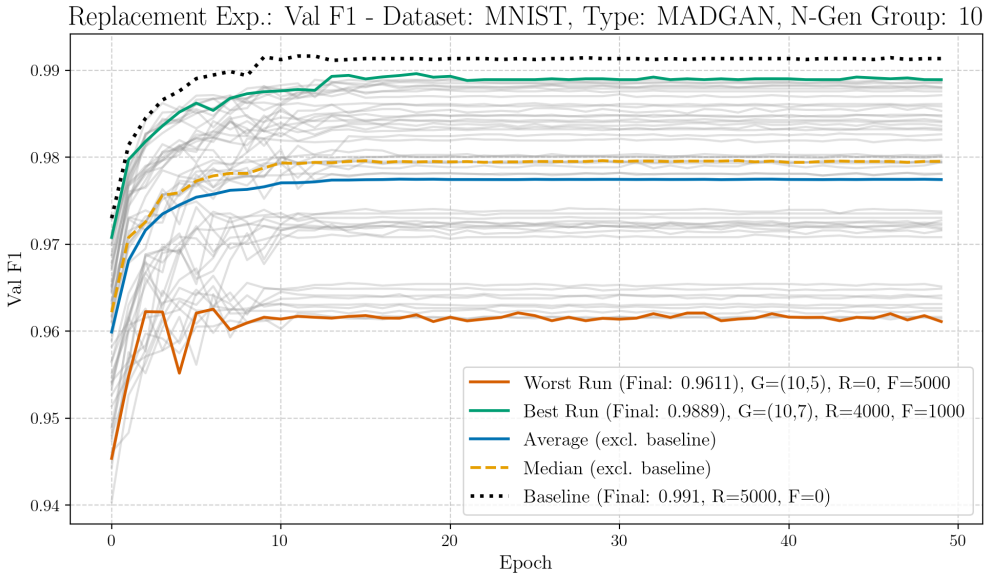
Run Type	Experiment	Val F1
best	$G_{7,5}$, R:4000, F:1000	0.9884
worst	$G_{7,0}$, R:0, F:5000	0.9616
median	G (K=7)	0.9800
average	G (K=7)	0.9780

Expansion Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,8}$, R:5000, F:1000	0.9898
worst	$G_{10,3}$, R:5000, F:5000	0.9835
median	G (K=10)	0.9865
average	G (K=10)	0.9866

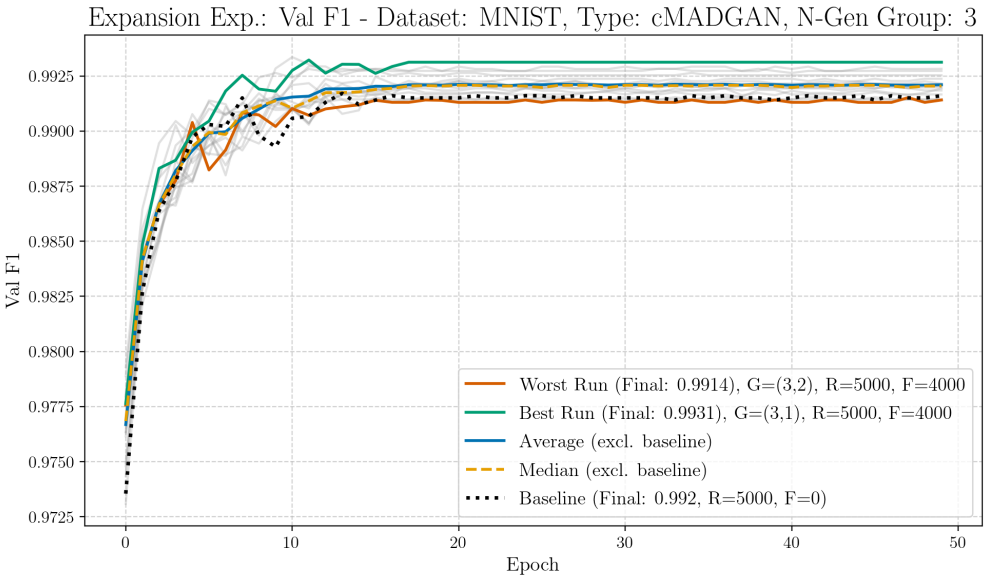
Replacement Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,7}$, R:4000, F:1000	0.9889
worst	$G_{10,5}$, R:0, F:5000	0.9611
median	G (K=10)	0.9795
average	G (K=10)	0.9774

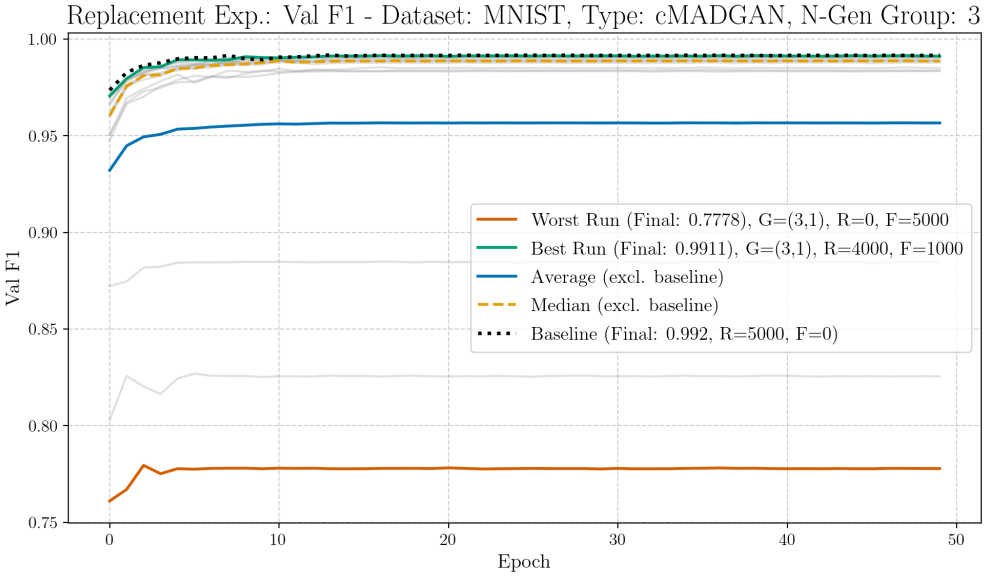
8.3.2 Dataset: MNIST, Architecture: cMADGAN

Expansion Experiment: K=3



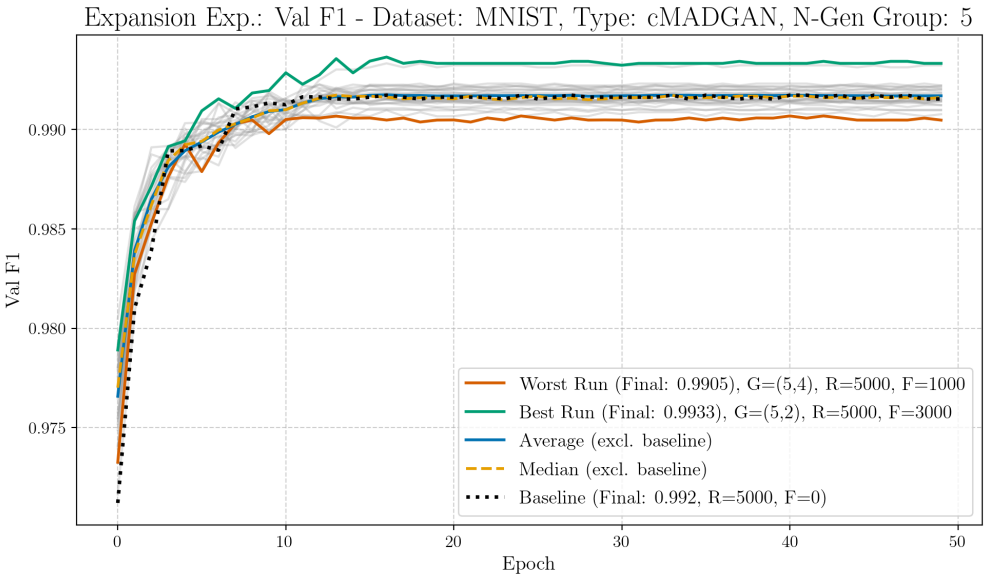
Run Type	Experiment	Val F1
best	$G_{3,1}$, R:5000, F:4000	0.9931
worst	$G_{3,2}$, R:5000, F:4000	0.9914
median	G (K=3)	0.9920
average	G (K=3)	0.9921

Replacement Experiment: K=3



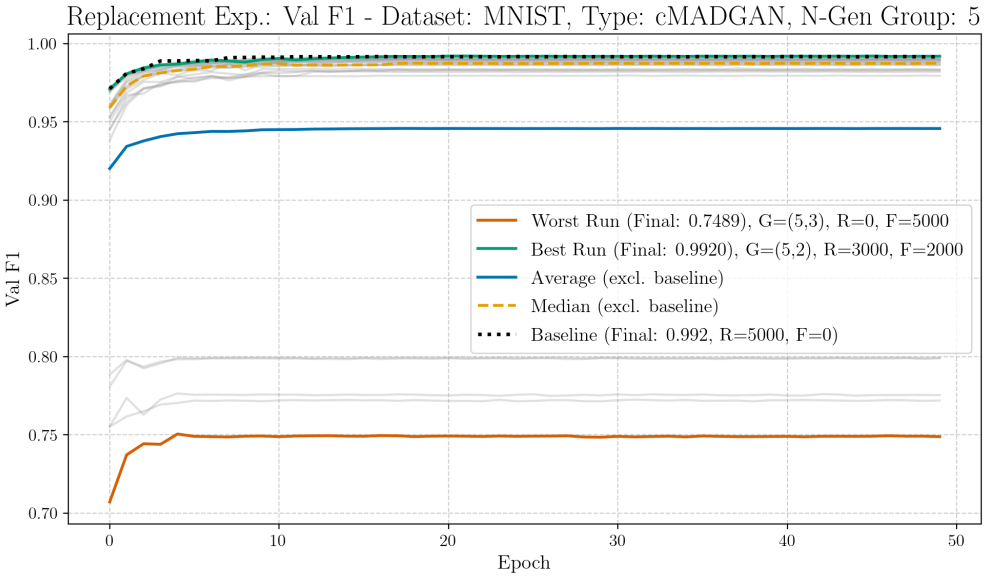
Run Type	Experiment	Val F1
best	$G_{3,1}$, R:4000, F:1000	0.9911
worst	$G_{3,1}$, R:0, F:5000	0.7778
median	G (K=3)	0.9886
average	G (K=3)	0.9566

Expansion Experiment: K=5



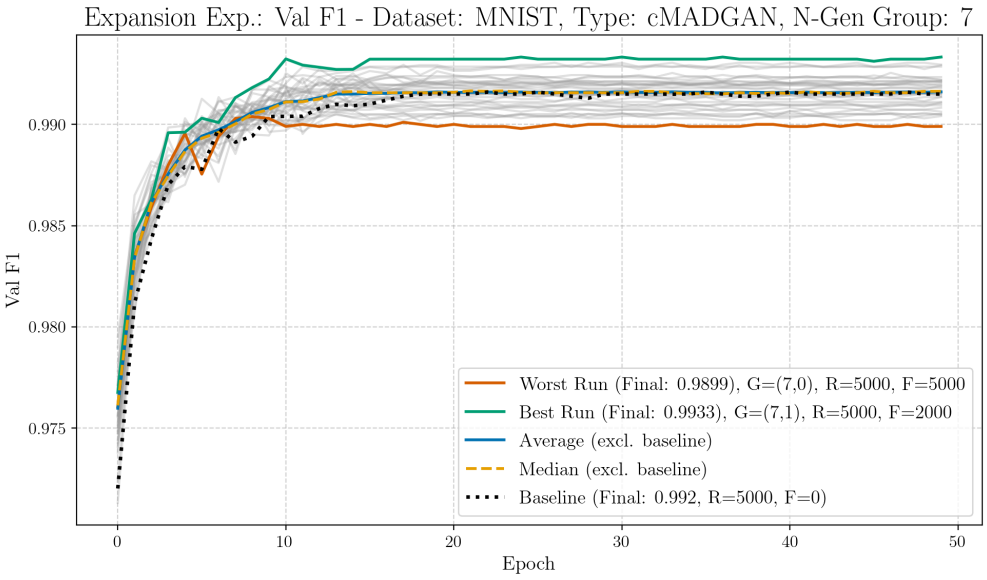
Run Type	Experiment	Val F1
best	$G_{5,2}$, R:5000, F:3000	0.9933
worst	$G_{5,4}$, R:5000, F:1000	0.9905
median	G (K=5)	0.9916
average	G (K=5)	0.9917

Replacement Experiment: K=5



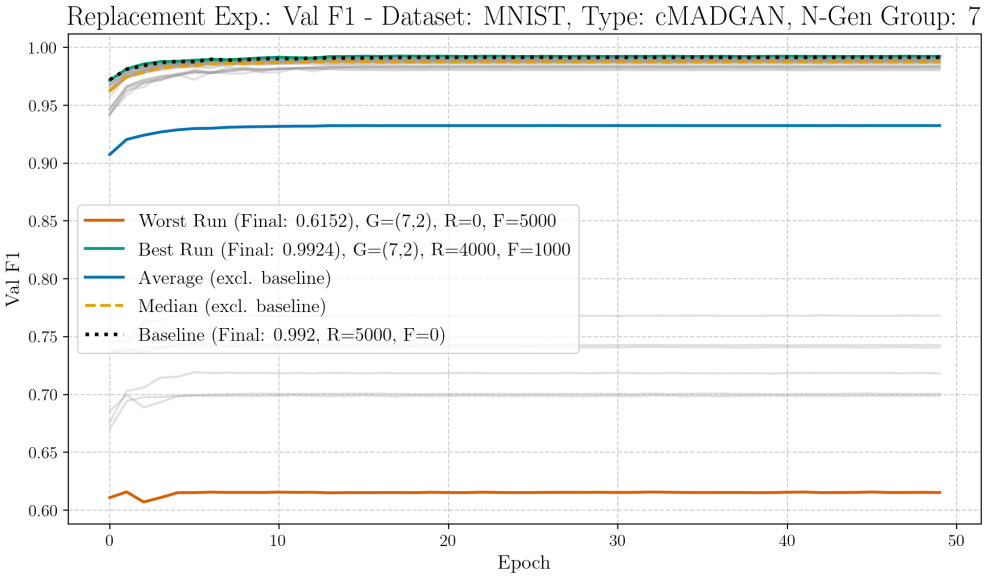
Run Type	Experiment	Val F1
best	$G_{5,2}$, R:3000, F:2000	0.9920
worst	$G_{5,3}$, R:0, F:5000	0.7489
median	G (K=5)	0.9874
average	G (K=5)	0.9458

Expansion Experiment: K=7



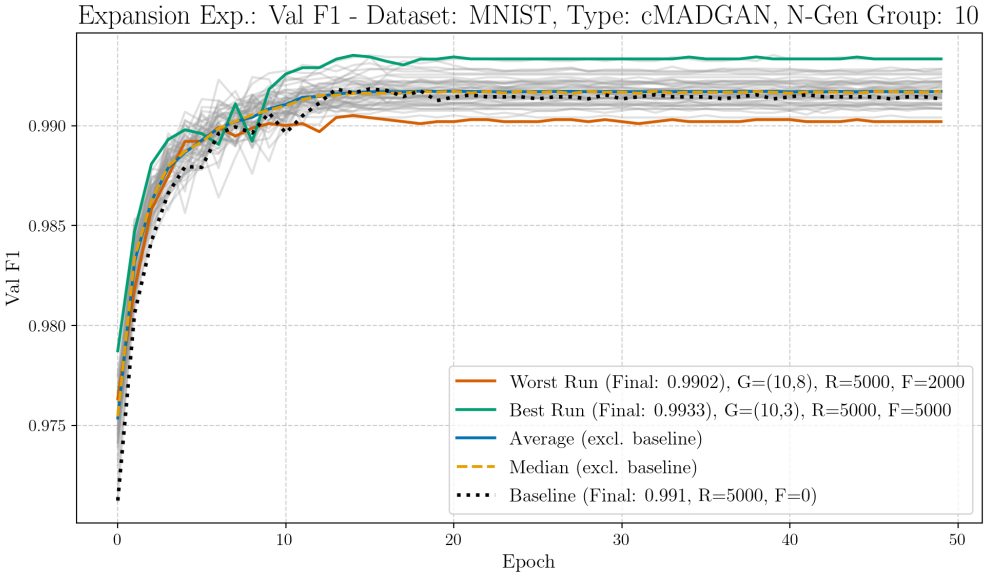
Run Type	Experiment	Val F1
best	$G_{7,1}$, R:5000, F:2000	0.9933
worst	$G_{7,0}$, R:5000, F:5000	0.9899
median	G (K=7)	0.9916
average	G (K=7)	0.9916

Replacement Experiment: K=7



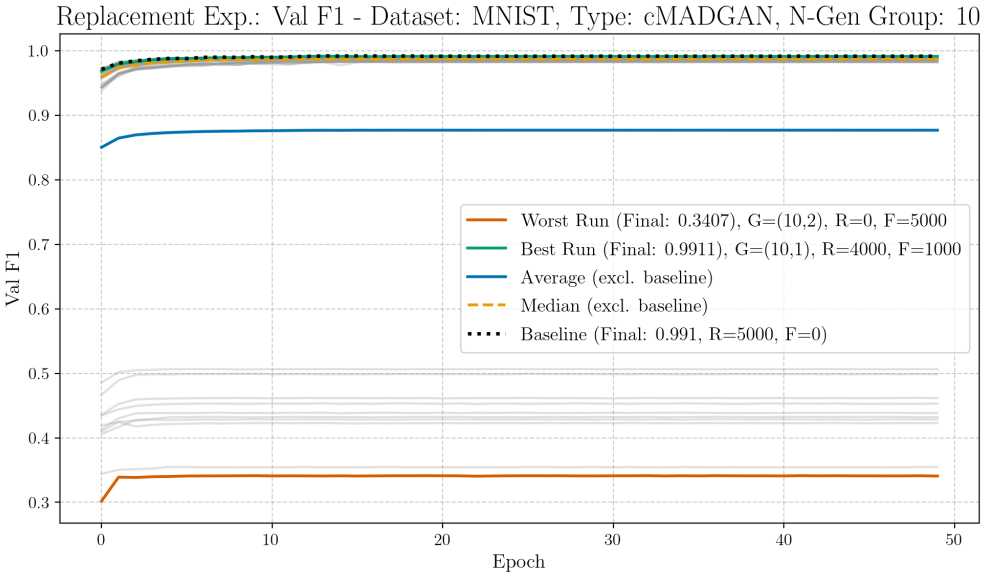
Run Type	Experiment	Val F1
best	$G_{7,2}$, R:4000, F:1000	0.9924
worst	$G_{7,2}$, R:0, F:5000	0.6152
median	G (K=7)	0.9874
average	G (K=7)	0.9325

Expansion Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,3}$, R:5000, F:5000	0.9933
worst	$G_{10,8}$, R:5000, F:2000	0.9902
median	G (K=10)	0.9917
average	G (K=10)	0.9917

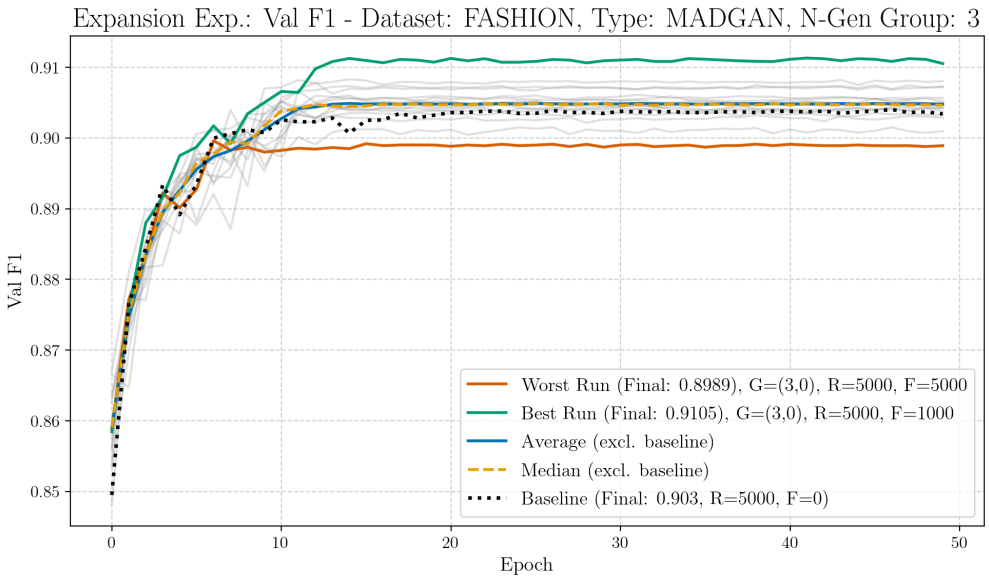
Replacement Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,1}$, R:4000, F:1000	0.9911
worst	$G_{10,2}$, R:0, F:5000	0.3407
median	G (K=10)	0.9872
average	G (K=10)	0.8768

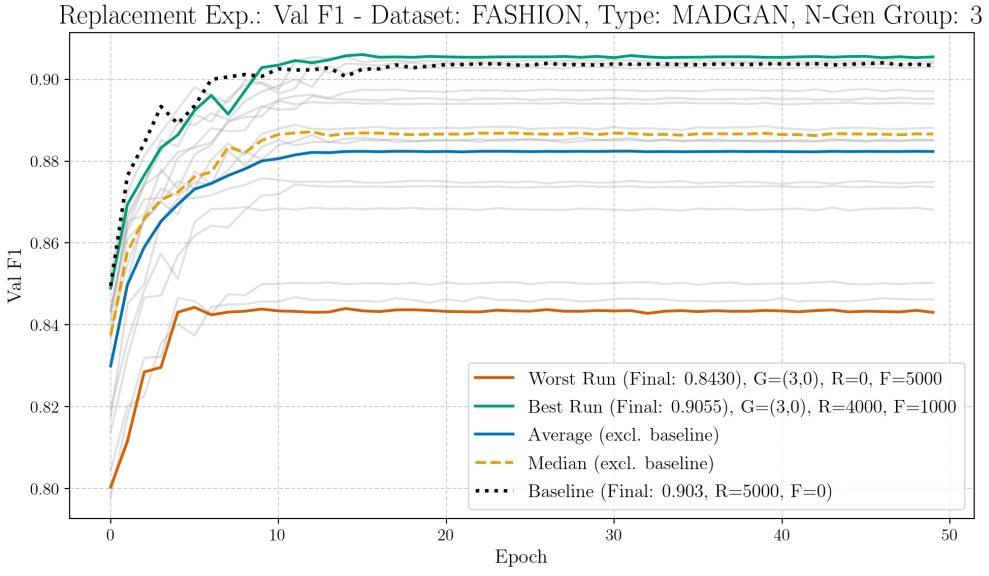
8.3.3 Dataset: FASHION, Architecture: MADGAN

Expansion Experiment: K=3



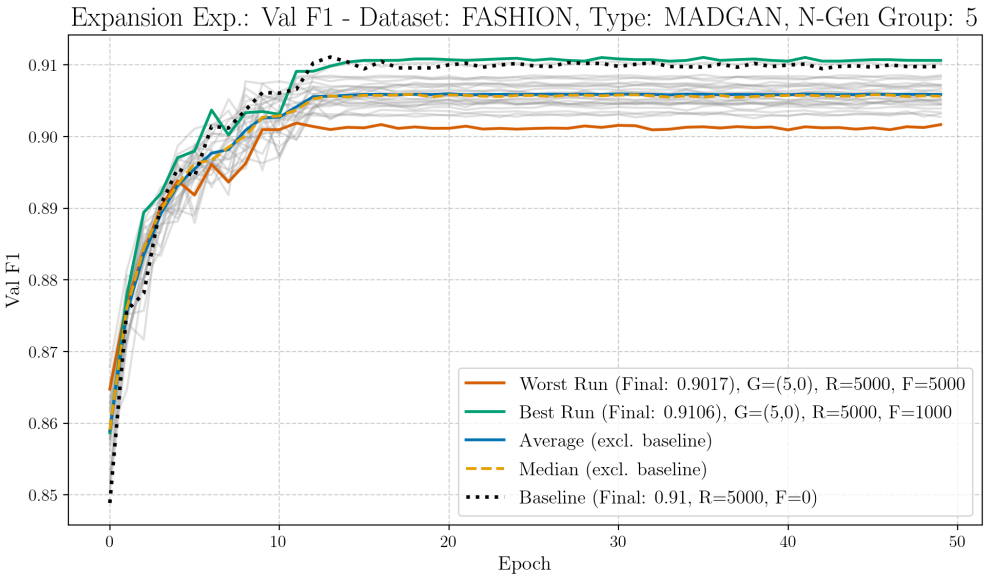
Run Type	Experiment	Val F1
best	$G_{3,0}$, R:5000, F:1000	0.9105
worst	$G_{3,0}$, R:5000, F:5000	0.8989
median	G (K=3)	0.9046
average	G (K=3)	0.9048

Replacement Experiment: K=3



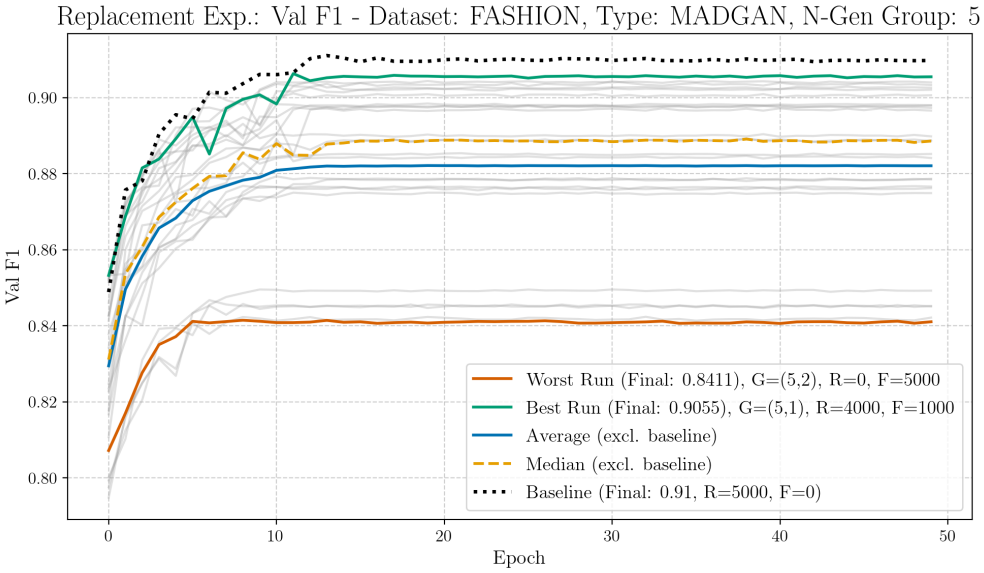
Run Type	Experiment	Val F1
best	$G_{3,0}$, R:4000, F:1000	0.9055
worst	$G_{3,0}$, R:0, F:5000	0.8430
median	G (K=3)	0.8866
average	G (K=3)	0.8824

Expansion Experiment: K=5



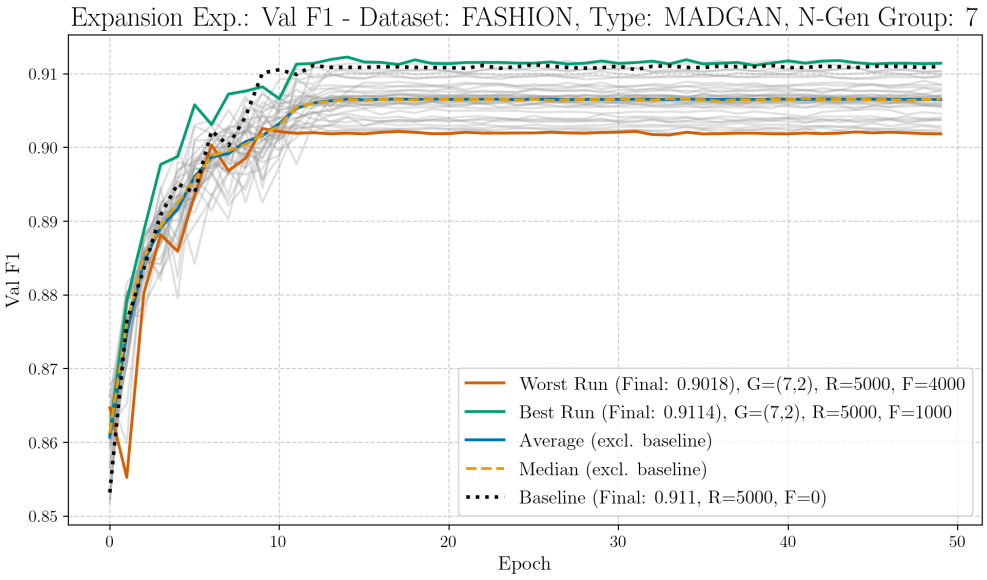
Run Type	Experiment	Val F1
best	$G_{5,0}$, R:5000, F:1000	0.9106
worst	$G_{5,0}$, R:5000, F:5000	0.9017
median	G (K=5)	0.9056
average	G (K=5)	0.9059

Replacement Experiment: K=5



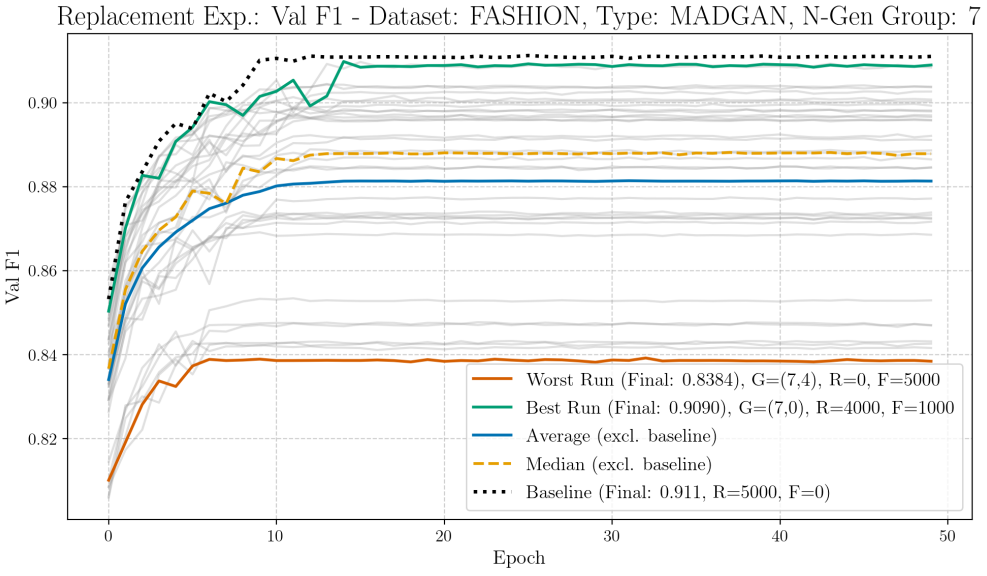
Run Type	Experiment	Val F1
best	$G_{5,1}$, R:4000, F:1000	0.9055
worst	$G_{5,2}$, R:0, F:5000	0.8411
median	G (K=5)	0.8886
average	G (K=5)	0.8821

Expansion Experiment: K=7



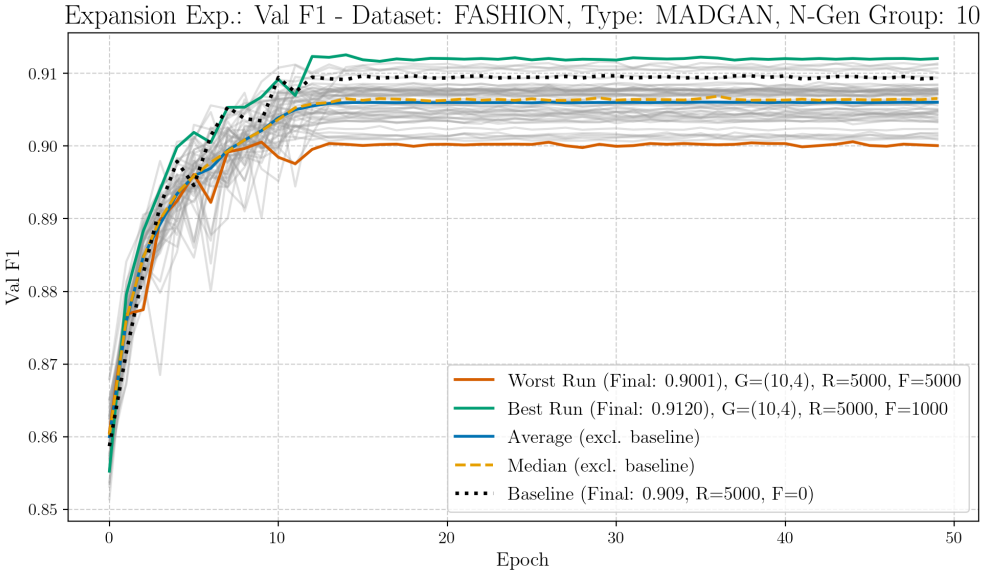
Run Type	Experiment	Val F1
best	$G_{7,2}$, R:5000, F:1000	0.9114
worst	$G_{7,2}$, R:5000, F:4000	0.9018
median	G (K=7)	0.9066
average	G (K=7)	0.9065

Replacement Experiment: K=7



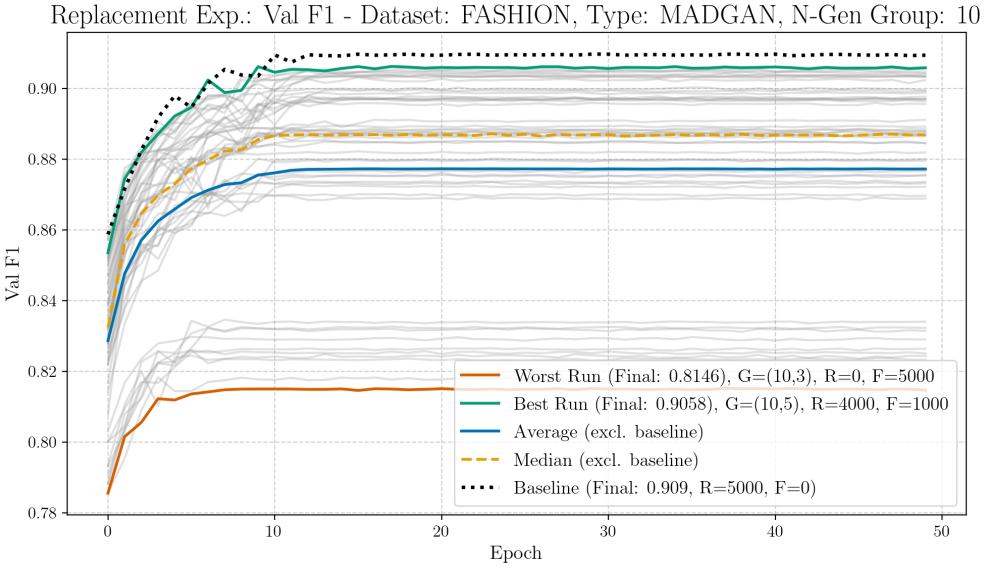
Run Type	Experiment	Val F1
best	$G_{7,0}$, R:4000, F:1000	0.9090
worst	$G_{7,4}$, R:0, F:5000	0.8384
median	G (K=7)	0.8879
average	G (K=7)	0.8813

Expansion Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,4}$, R:5000, F:1000	0.9120
worst	$G_{10,4}$, R:5000, F:5000	0.9001
median	G (K=10)	0.9066
average	G (K=10)	0.9060

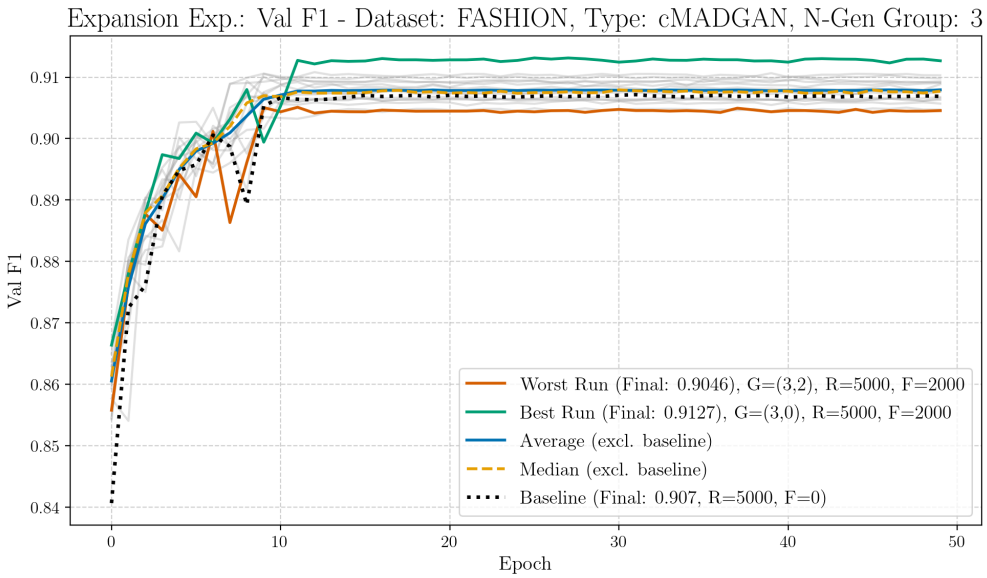
Replacement Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,5}$, R:4000, F:1000	0.9058
worst	$G_{10,3}$, R:0, F:5000	0.8146
median	G (K=10)	0.8869
average	G (K=10)	0.8772

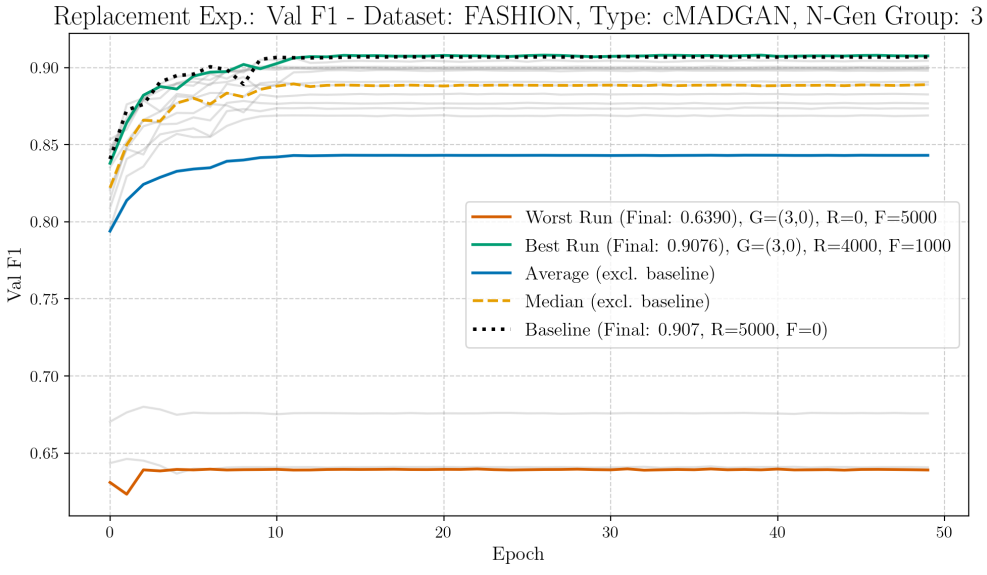
8.3.4 Dataset: FASHION, Architecture: cMADGAN

Expansion Experiment: K=3



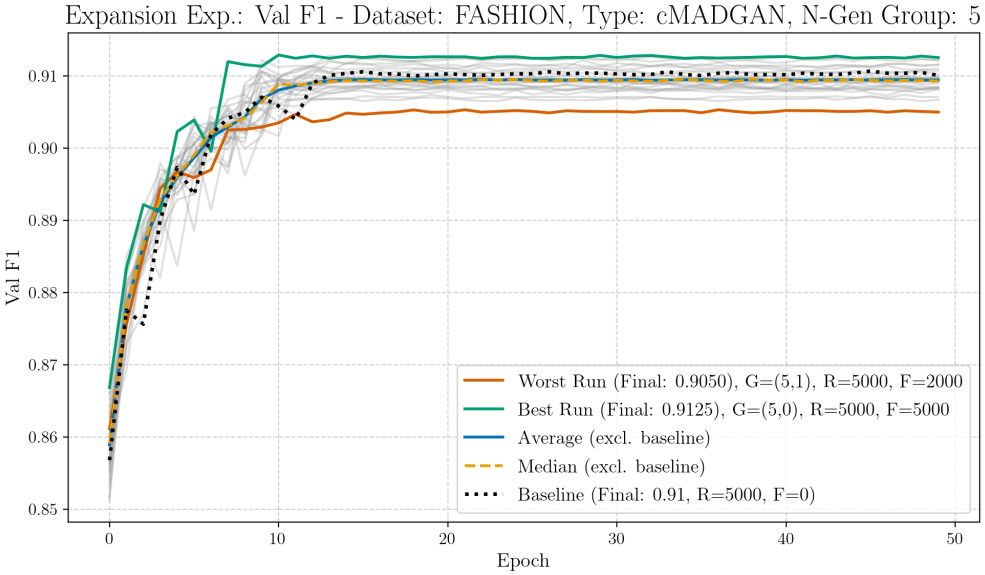
Run Type	Experiment	Val F1
best	$G_{3,0}$, R:5000, F:2000	0.9127
worst	$G_{3,2}$, R:5000, F:2000	0.9046
median	G (K=3)	0.9077
average	G (K=3)	0.9079

Replacement Experiment: K=3



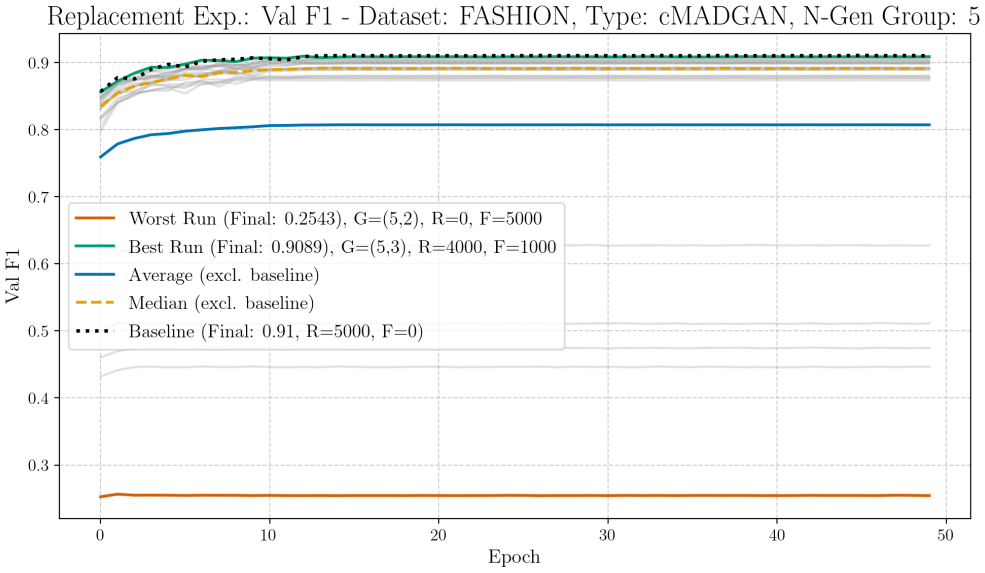
Run Type	Experiment	Val F1
best	$G_{3,0}$, R:4000, F:1000	0.9076
worst	$G_{3,0}$, R:0, F:5000	0.6390
median	G (K=3)	0.8890
average	G (K=3)	0.8431

Expansion Experiment: K=5



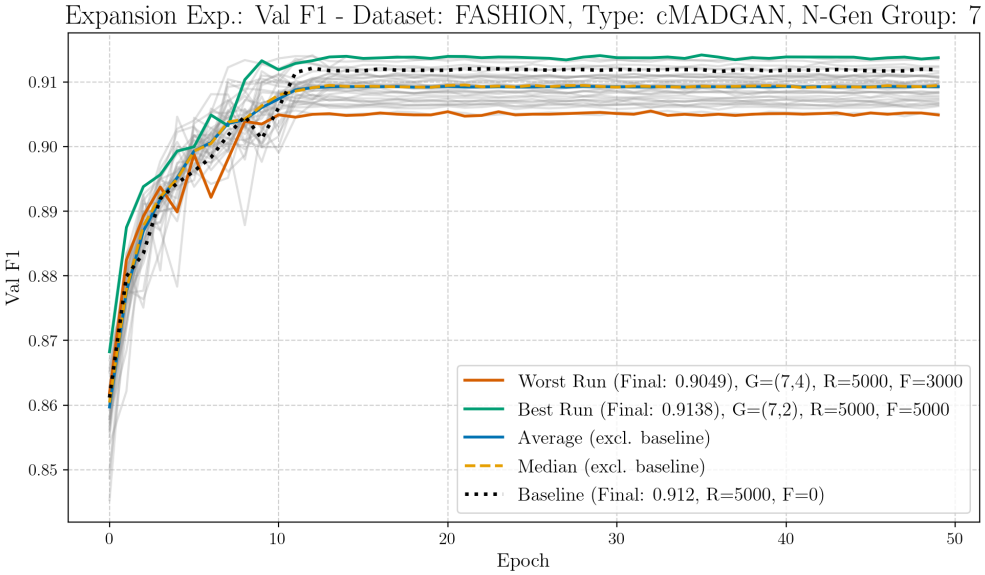
Run Type	Experiment	Val F1
best	$G_{5,0}$, R:5000, F:5000	0.9125
worst	$G_{5,1}$, R:5000, F:2000	0.9050
median	G (K=5)	0.9093
average	G (K=5)	0.9094

Replacement Experiment: K=5



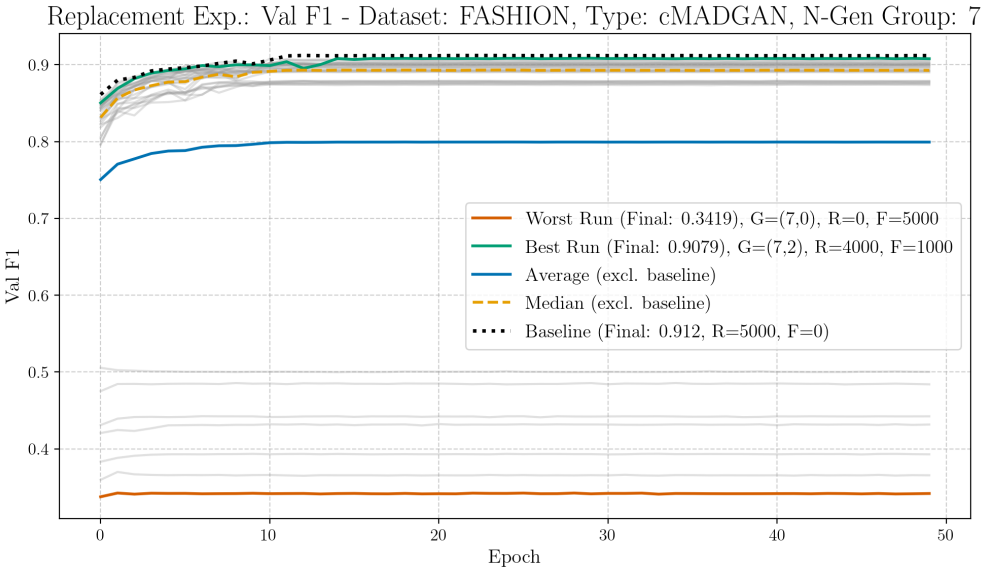
Run Type	Experiment	Val F1
best	$G_{5,3}$, R:4000, F:1000	0.9089
worst	$G_{5,2}$, R:0, F:5000	0.2543
median	G (K=5)	0.8909
average	G (K=5)	0.8072

Expansion Experiment: K=7



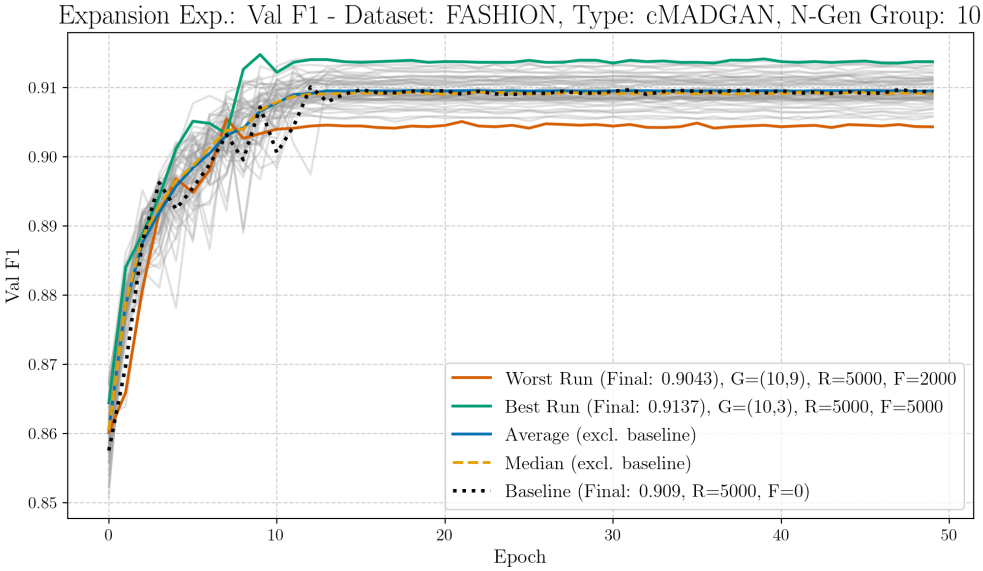
Run Type	Experiment	Val F1
best	$G_{7,2}$, R:5000, F:5000	0.9138
worst	$G_{7,4}$, R:5000, F:3000	0.9049
median	G (K=7)	0.9095
average	G (K=7)	0.9093

Replacement Experiment: K=7



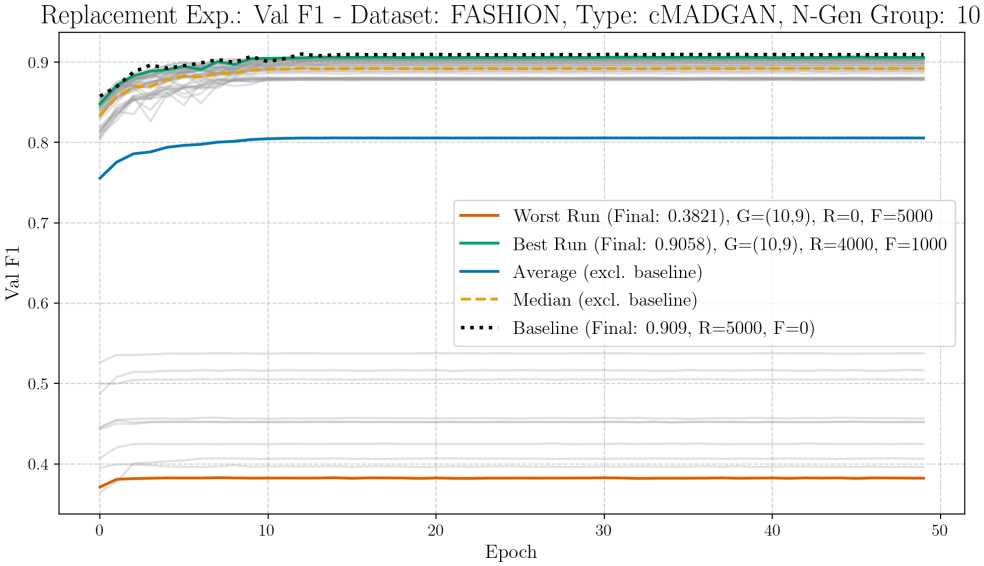
Run Type	Experiment	Val F1
best	$G_{7,2}$, R:4000, F:1000	0.9079
worst	$G_{7,0}$, R:0, F:5000	0.3419
median	G (K=7)	0.8927
average	G (K=7)	0.7993

Expansion Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,3}$, R:5000, F:5000	0.9137
worst	$G_{10,9}$, R:5000, F:2000	0.9043
median	G (K=10)	0.9091
average	G (K=10)	0.9095

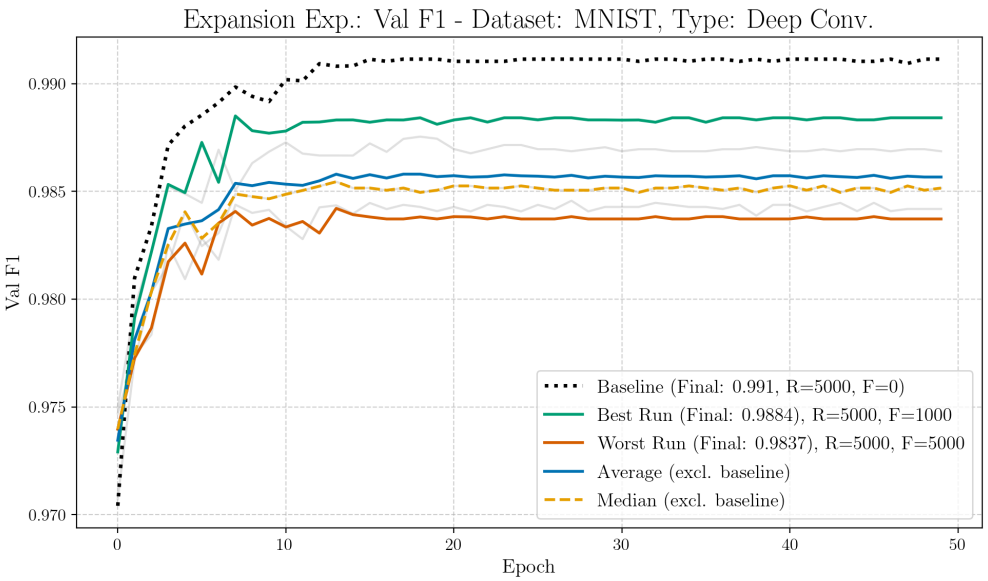
Replacement Experiment: K=10



Run Type	Experiment	Val F1
best	$G_{10,9}$, R:4000, F:1000	0.9058
worst	$G_{10,9}$, R:0, F:5000	0.3821
median	G (K=10)	0.8920
average	G (K=10)	0.8056

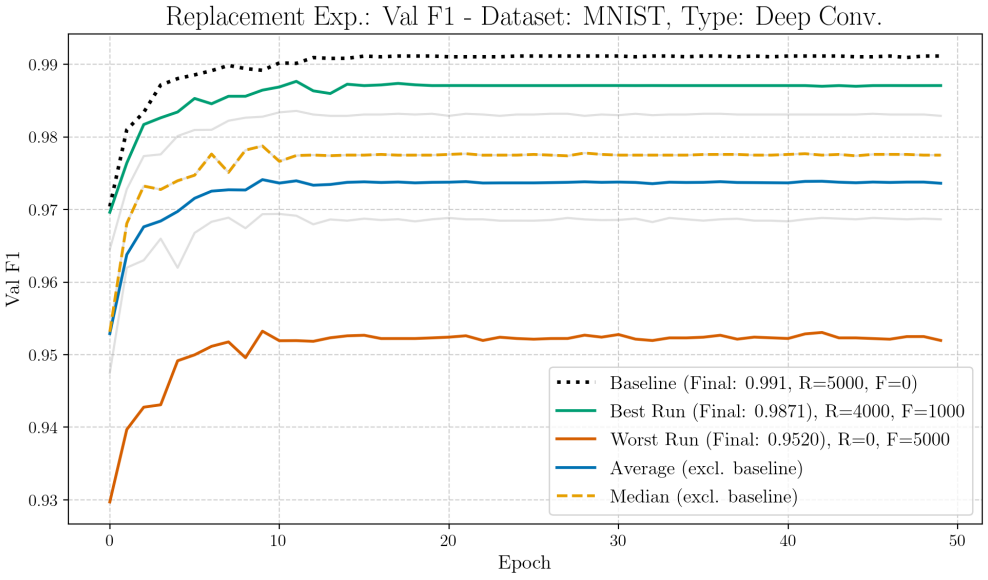
8.3.5 Dataset: MNIST, Architecture: DCGAN

Expansion Experiment:



Run Type	Experiment	Val F1
best	DC (R:5000, F:1000)	0.9884
worst	DC (R:5000, F:5000)	0.9837
median	DC	0.9852
average	DC	0.9857

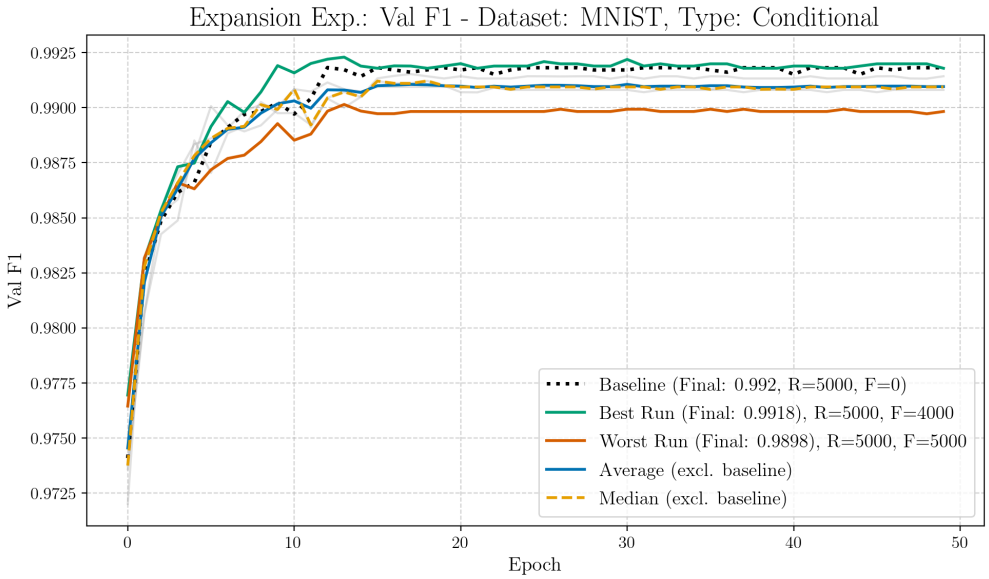
Replacement Experiment:



Run Type	Experiment	Val F1
best	DC (R:4000:, F:1000)	0.9871
worst	DC (R:0, F:5000)	0.9520
median	DC	0.9775
average	DC	0.9736

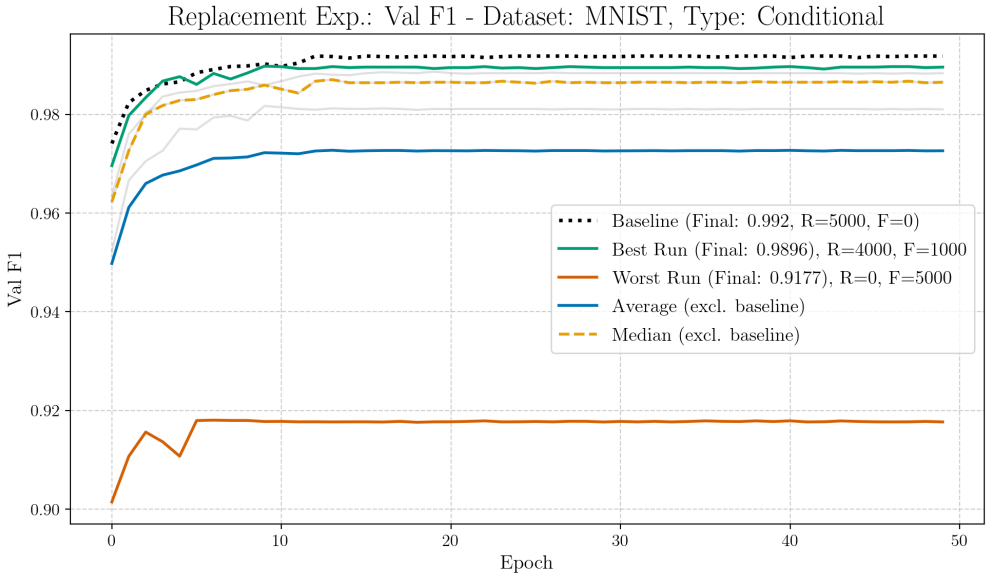
8.3.6 Dataset: MNIST, Architecture: cGAN

Expansion Experiment:



Run Type	Experiment	Val F1
best	Conditional (R:5000, F:4000)	0.9918
worst	Conditional (R:5000, F:5000)	0.9898
median	Conditional	0.9909
average	Conditional	0.9910

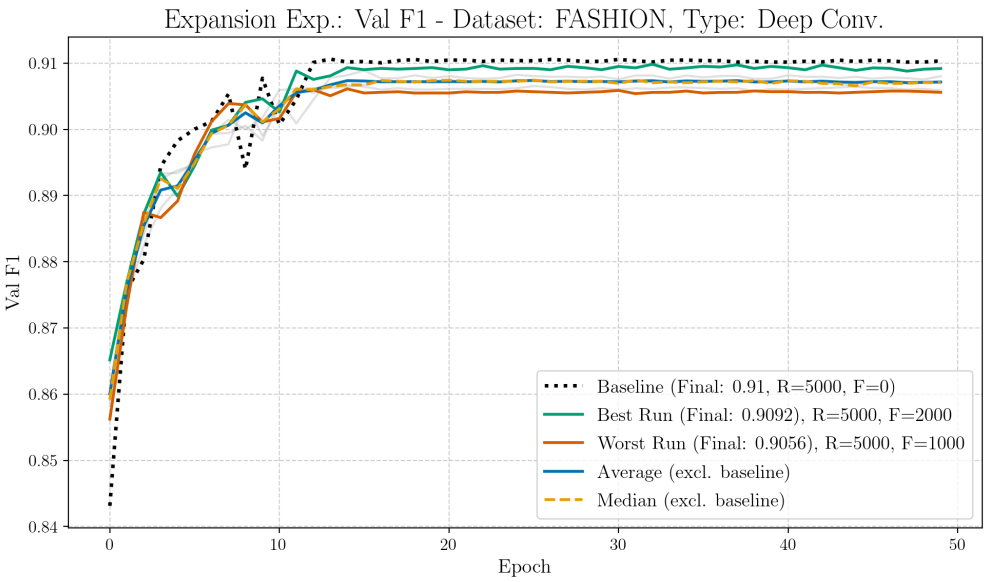
Replacement Experiment:



Run Type	Experiment	Val F1
best	Conditional (R:4000, F:1000)	0.9896
worst	Conditional (R:0, F:5000)	0.9177
median	Conditional	0.9865
average	Conditional	0.9726

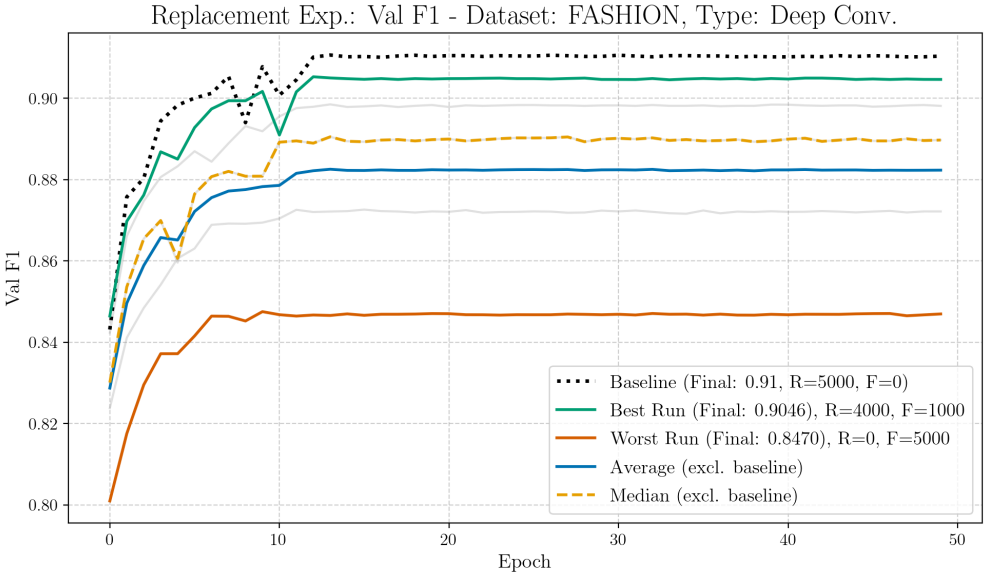
8.3.7 Dataset: FASHION, Architecture: DCGAN

Expansion Experiment:



Run Type	Experiment	Performance
best	DC (R:5000, F:2000)	0.9092
worst	DC (R:5000, F:1000)	0.9056
median	DC	0.9071
average	DC	0.9072

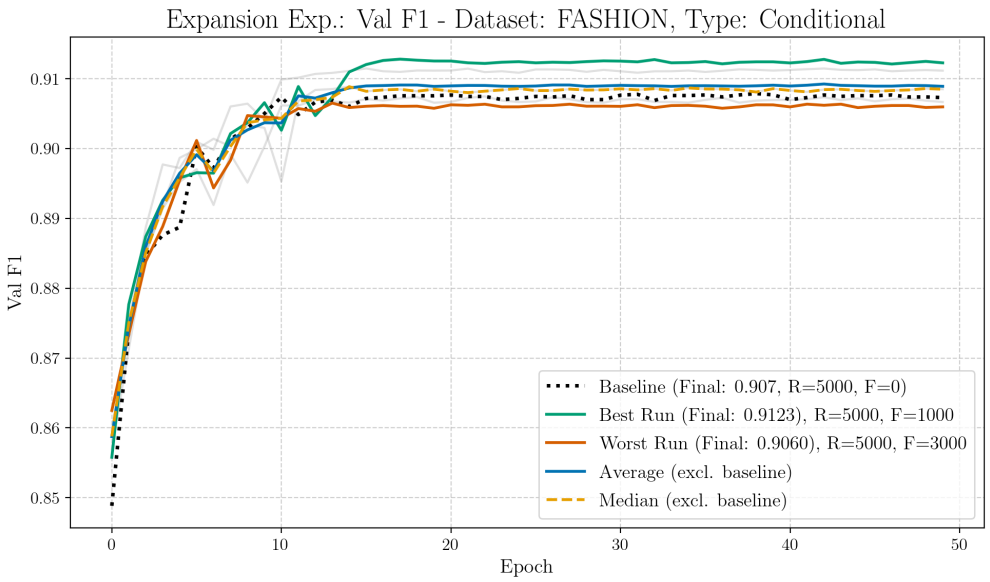
Replacement Experiment:



Run Type	Experiment	Performance
best	DC (R:5000, F:1000)	0.9046
worst	DC (R:0, F:5000)	0.8470
median	DC	0.8897
average	DC	0.8823

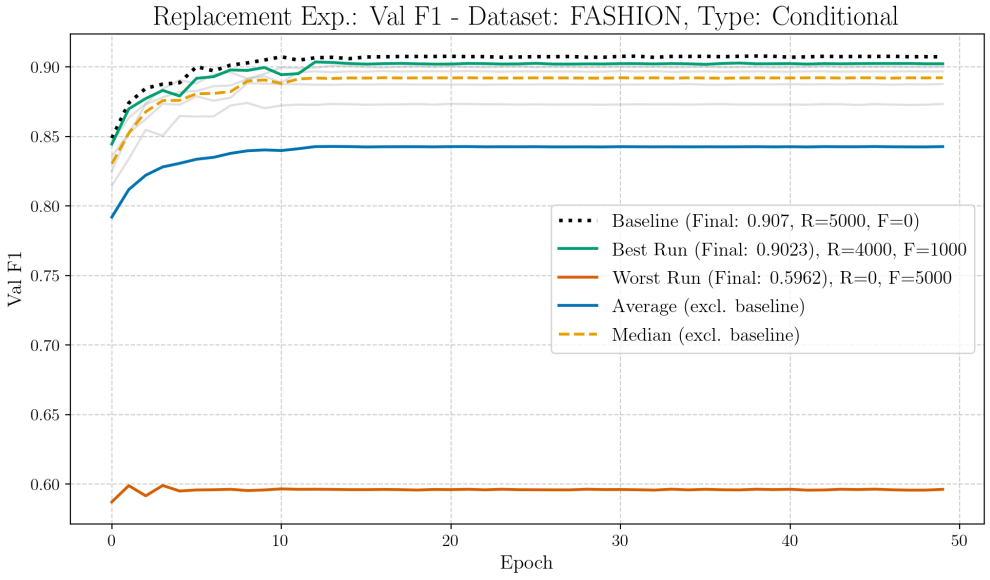
8.3.8 Dataset: FASHION, Architecture: cGAN

Expansion Experiment:



Run Type	Experiment	Val F1
best	Conditional (R:5000, F:1000)	0.9123
worst	Conditional (R:5000, F:3000)	0.9060
median	Conditional	0.9085
average	Conditional	0.9089

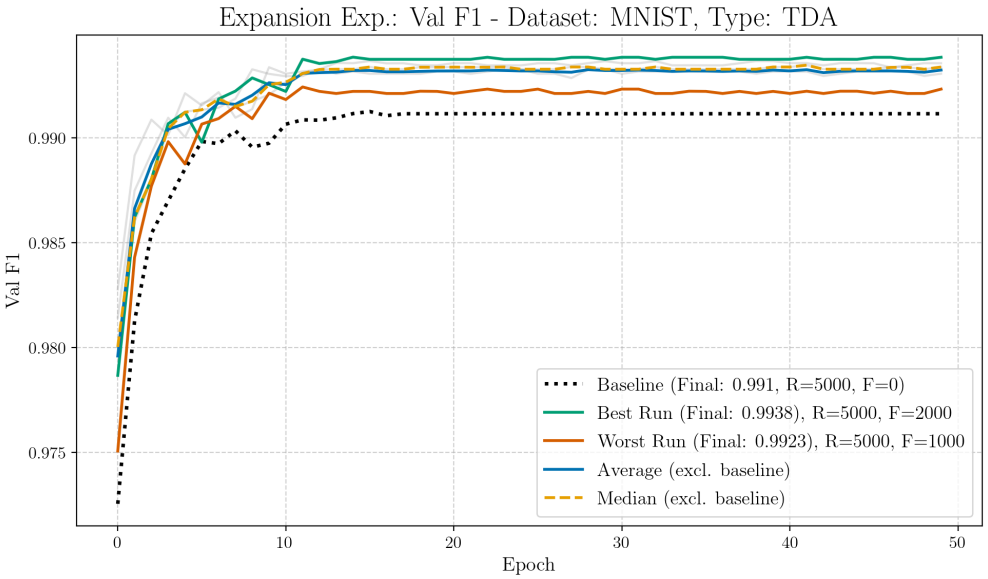
Replacement Experiment:



Run Type	Experiment	Val F1
best	Conditional (R:4000, F:1000)	0.9023
worst	Conditional (R:0, F:5000)	0.5962
median	Conditional	0.8923
average	Conditional	0.8427

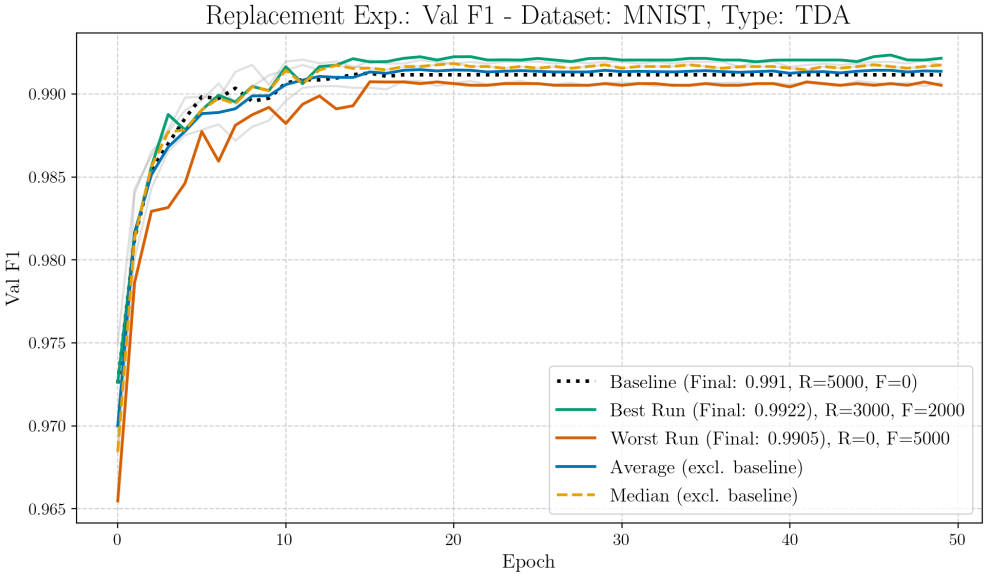
8.3.9 Dataset: MNIST, Architecture: TDA

Expansion Experiment:



Run Type	Metric	Val F1
best	TDA (R:5000, F:2000)	0.9938
worst	TDA (R:5000, F: 1000)	0.9923
median	TDA	0.9934
average	TDA	0.9932

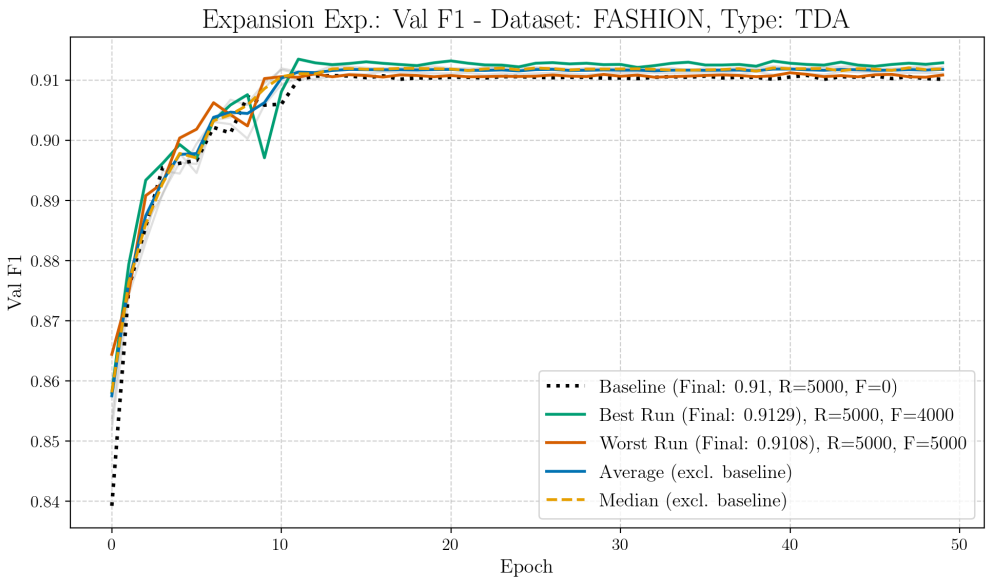
Replacement Experiment:



Run Type	Experiment	Performance
best	TDA (R:3000, F:2000)	0.9922
worst	TDA (R:0, F:5000)	0.9905
median	TDA	0.9917
average	TDA	0.9914

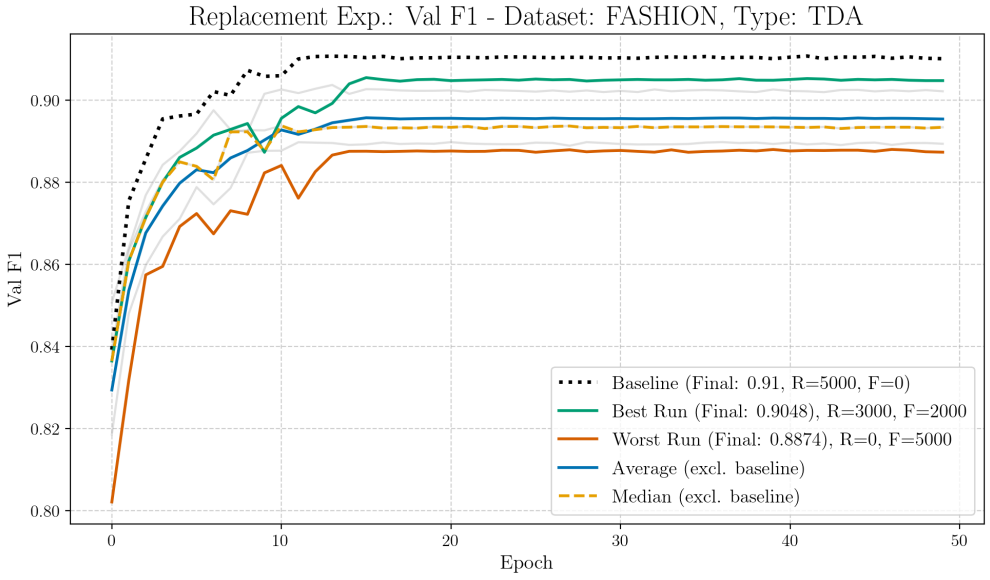
8.3.10 Dataset: FASHION, Architecture: TDA

Expansion Experiment:



Run Type	Experiment	Val F1
best	TDA (R:5000, F:4000)	0.9129
worst	TDA (R:5000, F:5000)	0.9108
median	TDA	0.9119
average	TDA	0.9118

Replacement Experiment:



Run Type	Experiment	Val F1
best	TDA (R:3000, F:2000)	0.9048
worst	TDA (R:0, F:5000)	0.8874
median	TDA	0.8934
average	TDA	0.8955

8.4 Other Graphs and Figures

8.4.1 DCGAN MNIST, Mode Collapse

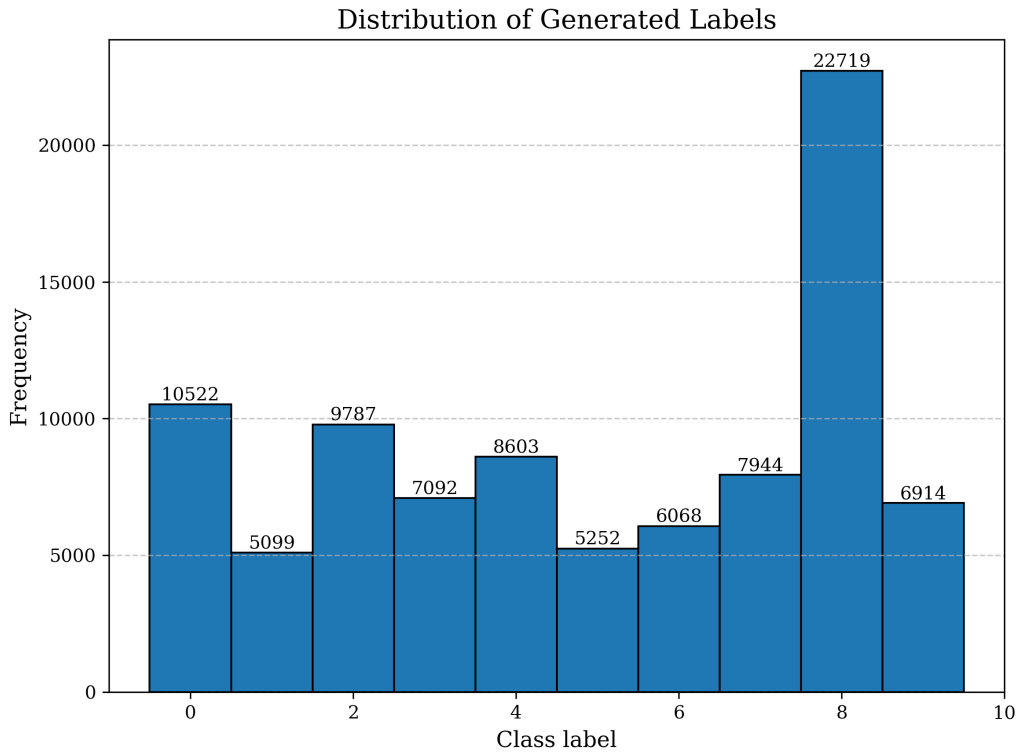


Figure 30: A histogram chart depicting the class distribution of the generated data with the DCGAN generator trained on the **MNIST** dataset. The labels result from an auxiliary classifier as mentioned in 4.1.5.

8.4.2 Convolutional Filtering

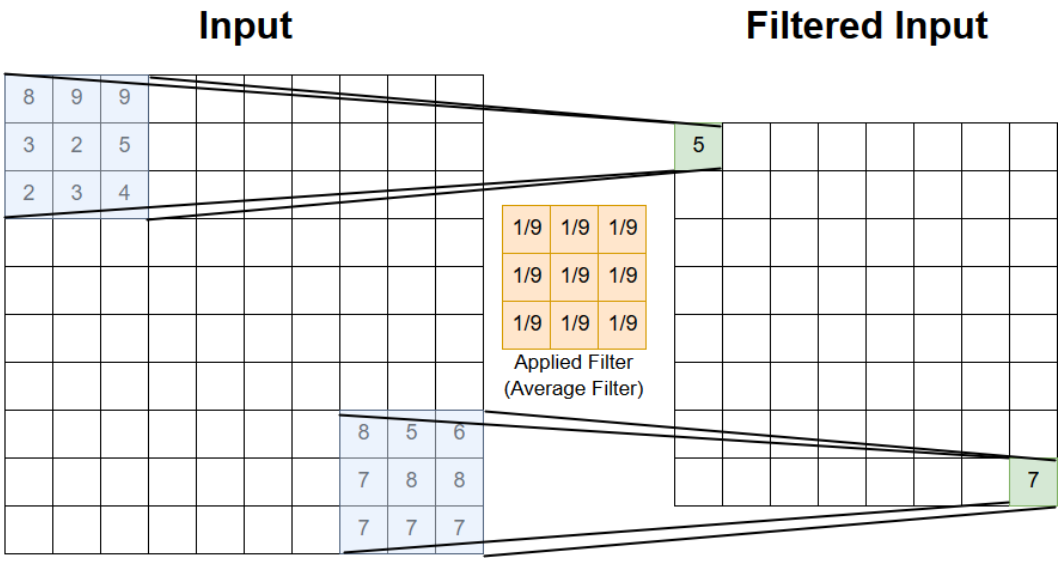


Figure 31: Depiction of the concept of convolutional filtering [Mau25].

Declaration of Academic Integrity

Generative Data Augmentation

Multi-Agent Diverse Generative Adversarial Networks for Generative Data Augmentation.

I hereby declare that I have written this thesis independently. I have properly cited all passages that are taken verbatim or in essence from published or unpublished works of others. All sources and aids used in the preparation of this thesis have been fully acknowledged. Furthermore, this thesis has not been submitted, in whole or in substantial part, to any other examination authority for academic credit.

Signiture :

Place, *Date* :