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: April 9, 2025

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

Abstract

asdf

Contents

List of Figures

List of Tables

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 sucessfully, 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 Gradiants & Unstable Gradiants
- Imbalance between Generator- and Discriminator Model

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 addrese mode collapse and inter-class diversity MADGANs explicitly enfocre mode seperation by 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.

Set 1: Training and Analysis of GANs The first set trains and analyses GANs, explicitly MADGANs and *Conditional GANs* (cGANs). Here, the quality of the resulting images during training will be scored by the *Fréchet Inception Distance* (FID) [HRU+18] and the *Inception Score* (IS) [SGZ+16].

Set 2: Generating and Classifying Unlabeled Images The second set uses the afore trained generative models to create images. Images without labels—images originating from MADGANs—will be classified using auxiliary classifiers trained with traditional data augmentation techniques.

Set 3: Training and Evaluating Classifiers The third and most significant set of experiments trains classifiers using the generated data. For this, stratified classifiers with differing numbers of real and fake images are trained and evaluated on the respective validation set. Their classification performance will be assessed using standard metrics. [1]

All of the above described is executed on the following datasets:

- MNIST [LCB10]
- Fashion MNIST [XRV17]
- CIFAR10 [Kri09]

Aim of the Thesis This thesis evaluates the effectiveness of Multi-Agent Diverse GANs for Generative Data Augmentation. First, the quality of their generated samples is compared to those produced by a Conditional GAN. Next, both sets of generated images are used to augment training datasets for classifiers, which are then assessed on their respective test sets. Classifiers trained on cGAN-augmented data and those trained with traditional augmentation techniques — such as flipping, rotation, and noise addition — serve as baselines for comparison. By doing so, this study examines the impact of MADGAN-based augmentation on classifier performance, highlighting its advantages and limitations relative to conventional methods and cGAN-based augmentation.

[1] The set of metrics used to assess the quality of the resulting classifiers is defined in chapter Experiments Setup 5.

2 Related Work

The effectiveness of deep learning models is instrinsically 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 specializied to the training data, failing to perfom accurately on previously unencountered data [Yin19].

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

Traditional Data Augmentation

Traditional data augmentation on images typically involves applying various tranformations to existing data. For image based data, augmentations can take a variaty 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 collegues 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 iamge data augmentation [HFM22]. Their architecture specifically utilizes *Convolutional Neural Network* (CNNs) [LBD+89] in both, the generator and the discriminator. The use of CNNs allows DCGANs to learn hierarchical features from the input images and capture the spatial relationship and structure inherent in images. This leads to the generation of more realistic and coherent synthetic images. A study from Zhau et al. [ZCWD23] applied DCGANs, along their adjusted version of those 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 convoltional layers, the generators role is soley 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 to be beneficial to expand data for a supervised classification task.

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 conditining 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 sucesses 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 (3.7) 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. Convolutional layers employ learnable filters to detect local patterns in the two-dimensional information - two dimensional in the case of images specifically. 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 TODO: here is a nice place for a ref to [DV18] 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 is written to the subsequent layer.

The stride determines how far the filter is moved 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 X in the Appendix (7). 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.

Pooling Layers A pooling layer compresses the data along the spatial axes to reduce its dimensionality. Similar to the convolutional layer, a pooling layer uses a filter that moves by the stride value. However, instead of summing the covered elements, the pooling operation applies the Max operator, selecting the maximum value within the filter's region.

For 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.

Max pooling helps retain the most important features, providing some invariance to small translations or distortions in the input, which is crucial for tasks like object recognition in convolutional neural networks (CNNs).

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 $[1 \times 1 \times 10]$[3]. By this stage, all spatial information has been transformed, leaving a quasi-one-dimensional vector containing the ten class scores for the CIFAR-10 dataset.

In a FC layer, each input is connected to each output, meaning every neuron in the

[3]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 $[1 \times 4]$.

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.

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) is an integreal part of convolutional networks. These layers normalize the inputs to subsequent layers and theirby stability the distribution of activations throughout the training process. This reduces the internal covariate shift, allowing for higher learning rates and faster convergence. By normalization of activation, batchnorm helps prevent gradients from vanishing or exploding. Additionally, it can provide regularization benefits and eliminate the need for Dropout, in some cases. With the afore mentioned benefits, batchnorm layers are particularily beneficial for deep learning networks with many layers.

Typical Activation Functions for 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.

- **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 and CIFAR10 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 three models use a 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 specifical model architectures can be found in chapter 4.

3.2 Data Augmentation

In this chapter, DA techiques in the context on 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 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 1970s [NS67]. For the context of deep learning, however, the augmentation of images was popularized by Krizhevsky et al. in 2012 [KSH12b], with the introduction of *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 speaking, traditional data augmentation techniques can be described as enlarging the initial training data by applying transformations that preserve the respective labels of individual instances. These techniques solely focus on modifying already existing data without creating entirely new instances (see: 3.2.2). 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 does not create entirely new data points but modifies existing ones, 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, which, in the domain of Arabic numerals, is semantically incorrect. Conversely, when classifying 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.

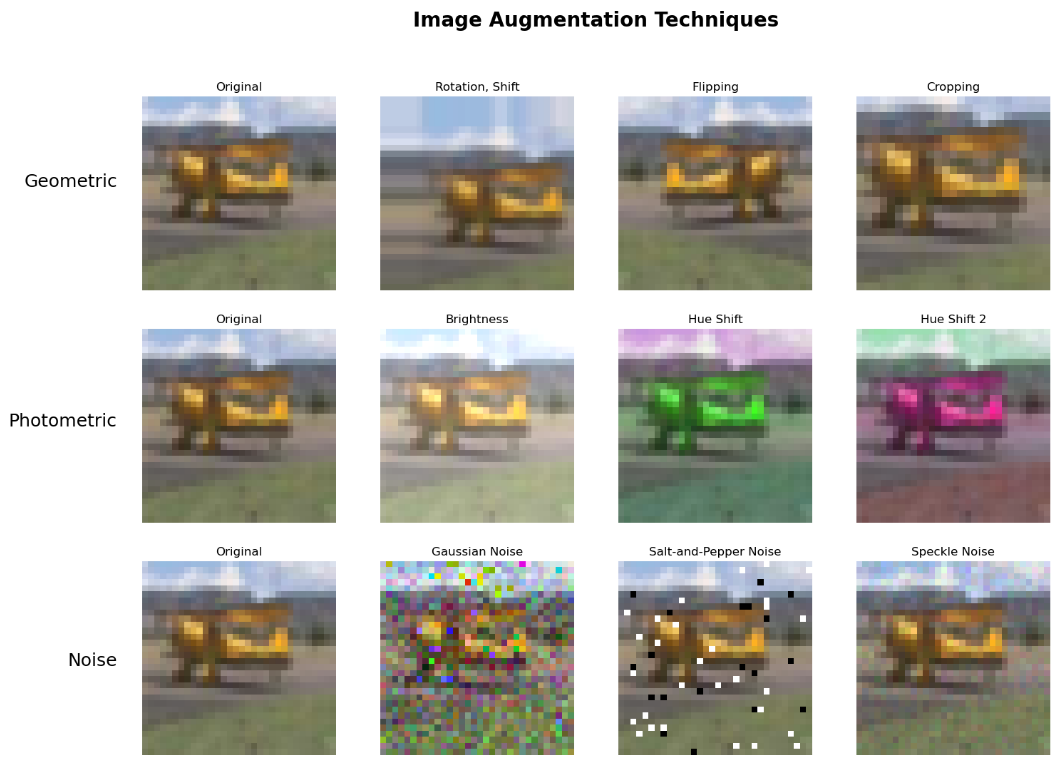

Figure 1: Exemplary use of traditional augmentation techniques from the categories *geometric* (first row), *photometric* (second row), and *noise-corruption* (third row). The image shown is an image from the CIFAR10 dataset, assigned to the class "airplane".

3.2.2 Generative Data Augmentation

Differing from the previously mentioned TDA 3.2.1, GDA does not focus on altering existing data instances but rather 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, 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:

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

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:

$$G : (z, y) \mapsto X_1, \quad X_1 \sim P_G(X \mid y) \approx P(X \mid y),$$

where $P_G(X \mid y)$ represents the learned conditional distribution, aiming to approximate the real class-conditioned data distribution $P(X \mid y)$. This enables 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 at 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 NNs: the textitGenerator G and the *Discriminator* D, competing against one another in a minimax game [Neu28]. The following figure (2) shows a visualization of the vanilla gan architecture.

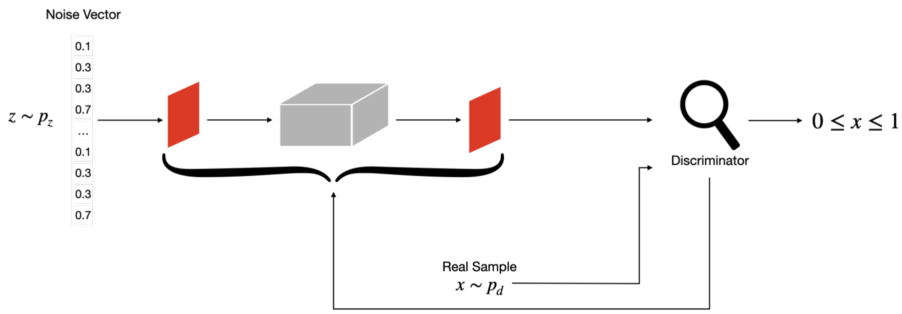

Figure 2: Visualization of the vanilla GAN architecture. The figure shows the noise vector, flowing into the discriminator, as well as the real sample from P_d

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 rather than 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:

$$\min_G \max_D \mathbb{E}_{X\sim P_{data}}[\log D(X)] + \mathbb{E}_{z\sim P_z}[\log(1 - D(G(z)))]. \tag{7}$$

Here, the discriminator D aims to maximize the probability of correctly classifying real and fake samples, 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., $P_G \approx P_{data}$.

3.3.2 Training Process

GAN training follows an alternating optimization approach:

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. Here, 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 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. 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, 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 help stabilize 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, training halts. 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].

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 Procecss), 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

generator's final layer to allow unrestricted output variability and in the discriminator's 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:

$$\tilde{X} = G(z, y) \tag{8}$$

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

$$D(X, y) \quad \text{and} \quad D(G(z, y), y) \tag{9}$$

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

$$\min_G \max_D \mathbb{E}_{X,y\sim P_{data}}[\log D(X, y)] + \mathbb{E}_{z\sim P_z,y\sim P_y}[\log(1 - D(G(z, y), y))] \tag{10}$$

This formulation forces to generate samples that align with the given condition , while learns to discriminate between real and generated samples, considering 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 discriminator's 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. 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: integetration 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.

3.6.1 Mathematical Formulation

As afore mentioned, the MADGAN architecture employs multiple generators and one discriminator. The goal the for K generators is to generate samples from different

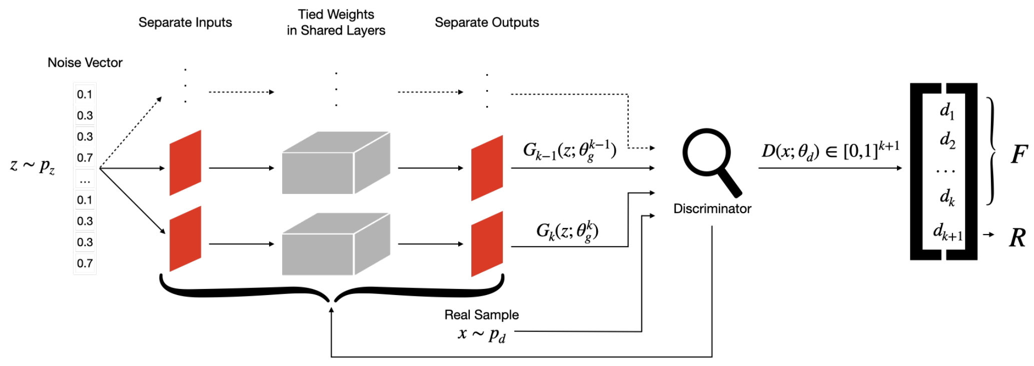

Figure 3: Visualization of the MADGAN architecture. The figure shows the $k+1$ outputs of the discriminator and the k generators, with tied weights in the middle of the network.

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, in which it no longer has to just differentiate between real and fake images, but also identify the generator that produced a given fake instance. 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 originating 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, $\delta(j) = 1$ is 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 [4] . The objective function for the optimization of θ_d, with θ_g freezed is therefore:

$$\max_{\theta_d} \mathbb{E}_{x \sim p} H(\delta, D(x; \theta_d)) \tag{11}$$

$H(...)$ here represents the negative cross-entropy function. Important to piont 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

[4] 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 $i \neq j$.

identifiable modes." [GKN+18], page 4. Figure 2 shows a visualization of the generators being pushed to diffenret modes. The objective for the generators however, remains semantically the same as in the vanilla GAN 7. 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:

$$\mathbf{E}_{x \sim P_{data}}[\log D(X; \theta_d)] + \mathbf{E}_{z \sim P_z} \log(1 - D_{k+1}(G_i(z; \theta_g^i); \theta_x)) \tag{12}$$

Enforing Diverse Modes: The researchers provide a theorem demonstrating that the k generators collectively form a mixture model. In it, each generator represents a mixture compotonent to the global optimimum of $-(k+1)\log(k+1) + 1\log k$. This is achieved, when $p_{data} = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$. It is to point out 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:

$$\mathbf{E}_{x \sim P_{data}}[\log D_{k+1}(X)] + \sum_{i=1}^{k} \mathbf{E}_{x \sim p_{g_i}} \log(1 - D_{k+1}(x)) \tag{13}$$

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

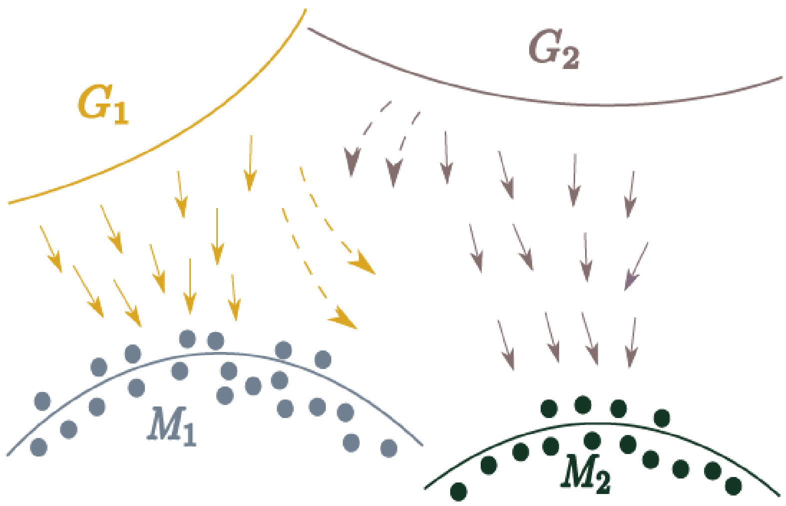

Figure 4: Figre taken from the original paper [GKN+18]. The visualization shows how different generators G_1 and G_2 are pushed to different modes M_1 and M_2

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 wheter an image originates from either one of the generator and one output implying if the discriminator cam to the conclusion that the current image is fake or not. As afore described, this ecurages the discriminator to push the generators into identifiable distinct outptus.
Parameter Sharing among Generators: Generators in the MADGAN architecture share most of their layers between the k generators. This reducese 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 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.

3.7.1 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 (Objectiveness)**: 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 different labels. This corresponds to a high-entropy marginal label distribution $p(y)$.

Mathematically, the Inception Score is computed as:

$$IS = \exp\left(\mathbb{E}_{x \sim p_g}\left[D_{\mathrm{KL}}(p(y|x) \| p(y))\right]\right) \tag{14}$$

where D_{KL} denotes the Kullback-Leibler divergence between the conditional class distribution and the marginal class distribution.

Limitations of IS:

- Does not directly compare generated images with real images.
- Insensitive to intra-class diversity.
- Sensitive to the choice of the pretrained classifier.

3.7.2 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 and generated images into a lower-dimensional feature space (obtained from a pretrained InceptionV3 network) and models these embeddings as multivariate Gaussians.

Let (μ_r, Σ_r) and (μ_g, Σ_g) denote the means and covariances of the real and generated image features, respectively. The FID score is computed as:

$$FID = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right) \tag{15}$$

Interpretation:

- Lower FID indicates better quality and diversity.
- Sensitive to both image quality and intra-class diversity.

Advantages over IS:

- Considers real image distribution.
- Sensitive to intra-class mode collapse.

Limitations:

- Assumes Gaussian distribution of features.
- Sensitive to the choice of feature extractor.

3.7.3 InceptionV3 for Image Evaluation

Both IS and FID rely on the InceptionV3 model [SVI+16], a deep convolutional neural network trained on ImageNet. The InceptionV3 network serves two primary roles in generative image evaluation:

1. **Feature Extractor**: Intermediate activations (typically from the pool3 layer) are used as a semantic representation of the images for computing FID.

2. **Classifier**: The output class probabilities of InceptionV3 are used to compute IS.

Properties of InceptionV3 relevant for image scoring:

- Trained on large-scale ImageNet dataset — providing a rich semantic feature space.
- Fixed, non-trainable weights during evaluation — ensuring standardization.
- Layer choice affects sensitivity — earlier layers capture texture, later layers capture semantics.

Limitations of using InceptionV3:

- Domain gap between ImageNet images and domain-specific generative tasks (e.g., medical images, industrial images).
- Performance can degrade if generated images differ significantly from ImageNet classes.

4 Preliminary Remarks

This chapter outlines general remarks relevant to the subsequent experiment chapters (5, 6).

4.1 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. For experiments on the *CIFAR10* dataset, however, deeper architectures have been used for both the generator and discriminator networks to account for the increased complexity of the data.

Training For training all GAN-based models, including DCGAN, cGAN, and MADGAN, 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 $[-0.1, 0.1]$. Gaussian noise is added 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 $[-1, 1]$.

4.2 Labeling unconditioned data

4.3 Calculation of FID and IS

5 Experiments Setup

6 Experiments Results

Motivation

7 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.

8 Outlook

repair networks [TFNL22]

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

9 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 theoratival 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.

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

[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

[Kri09] Krizhevsky, Alex: *Learning Multiple Layers of Features from Tiny Images*. Technical Report, University of Toronto. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. Version: 2009

[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)

[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

[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

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.

$Signiture :$ $Place, Date :$