communicated by:
Juniorprofessur für Mathematische Bild- und Signalverarbeitung

Prof. Dr. Benjamin Berkels

The present work was submitted to the
Aachen Institute for Advanced Study in Computational Engineering Science

Master's Thesis

Model-based analysis of the image generation quality of adversarial latent autoencoders for industrial machine vision

Ruslan Yermakov

Mar 2023

1st Examiner: Prof. Dr. Benjamin Berkels
2nd Examiner: Prof. Dr.-Ing. Robert Schmitt
Supervisor: Dominik Wolfschläger

Abstract

In industrial machine vision applications, generative models have an advantage over discriminative models because they enable interpretable feature extraction and overcome the limitations of non-transparent and inexplicable decisions associated with the latter. The state-of-the-art style-based generative adversarial networks (StyleGANs) generate high-quality images mapped from a latent feature vector space by learning the approximation of the high-dimensional training data distribution. Thereby, learned representations can be identified by interpreting the latent space and used to control the properties of the synthesized images. StyleGANs in combination with an embedding algorithm - adversarial latent autoencoder (ALAE) - enable the assessment of the properties of embedded images through their latent space representations. However, in order to achieve the best possible approximation of the training data distribution, it is necessary to optimize the network's design parameters based on its effectiveness to learn the quality characteristics of industrial machine vision data. This requires a quantitative evaluation of the quality of generated, and reconstructed images in comparison to the quality of original images.

This work presents an evaluation framework to assess the capabilities of generative ALAE models to learn the features and characteristics of the original data consistently and reliably. Feature consistency is proposed as an evaluation criterium to estimate the performance of the generative models. The quality of images generated by ALAE is quantitatively evaluated, with a focus on determining how the latent space size affects the outcome. Latent space size systematically varied during training on industrially relevant datasets with representative features. Based on the application-specific and human-interpretable features, the image quality of multiple ALAEs with varying latent space dimensionalities is compared using statistical tests to select the most favorable latent space dimension.

Note: The condensed version of the MSc thesis is included in Proceedings Volume 12438, AI and Optical Data Sciences IV, SPIE OPTO 2023, and can be found at [39]. If you use the current MSc Thesis, please cite the mentioned publication[1].

[1]See the publication - Dominik Wolfschläger, Ruslan Yermakov, Benjamin Montavon, Benjamin Berkels, Robert H. Schmitt, "Identifying the advantageous latent space dimensionality for StyleGANs used in industrial machine vision applications," Proc. SPIE 12438, AI and Optical Data Sciences IV, 124380R (15 March 2023); https://doi.org/10.1117/12.2646326

I Contents

II Acronyms

Abbreviation	Description
VAE	Variational Autoencoder
GAN	Generative Adversarial Network
PGGAN	Progressively-Growing Generative Adversarial Network
StyleGAN	Style-based Generative Adversarial Network
ALAE	Adversarial Latent Autoencoder
StyleALAE	Style-based Adversarial Latent Autoencoder
FID	Frechet Inception Distance
IS	Inception Score
JSD	Jensen-Shannon Distance

III List of Figures

IV List of Tables

1 Introduction

German Federal Government has named Industry 4.0 to be the central focus of its digital agenda in its efforts to support Germany's manufacturing industry by replacing the current labor-intensive processes with autonomous, interconnected, and sustainable smart factories. The Industry 4.0 framework combines multiple cutting-edge technological fields, such as data connectivity (internet of things, sensors, blockchain), human-machine interaction (robotics, automated guided systems, augmented reality), intelligence (advanced analytics, artificial intelligence, knowledge work automation), advanced production methods (renewable energy, 3D printing).

According to the McKinsey report "Capturing the true value of Industry 4.0" [12], the companies who successfully implement digital transformation reap significant benefits across the whole production value chain - from decreased production inefficiencies to higher customer satisfaction and reduced environmental impact. Upon successful implementation in real-life manufacturing systems, newly-introduced technologies can yield 10-20% cost-quality improvement, 15-30% labor productivity increase, and 10-30% throughput increase, among others. On top of the quantitative results, digital transformation is seen to improve workplace safety, employee satisfaction, decision-making process, as well as open opportunities for employee upskilling and cross-functional collaboration. Moreover, while Industry 4.0 adoption was already growing before COVID-19, the pandemic has made the transition even more critical, thereby further accelerating the manufacturing industry's digitalization and automation.

The key attribute of smart manufacturing is the automation of production pipelines. This study contributes to the establishment of Industry 4.0 by employing computer vision advancements in Artificial Intelligence to automatically inspect the produced goods in the predictive quality assurance pipeline in smart manufacturing.

1.1 Motivation

In predictive maintenance and quality assurance, one of the main tasks is to automatically detect defects in certain objects with high accuracy. Deep learning methods achieve state-of-the-art performance on this task by first learning feature representation of each sample in a black-box fashion and then correctly classifying whether there is a defect or not. The desired accuracy of correct predictions outperforms previously conventional image processing techniques. However, one of the main challenges for the broad adoption of deep learning techniques in industry is the lack of understanding of their decisions. Since neutral networks inherently rely on self-learned features, the lack of transparency and interpretability prevents the industry from adapting sophisticated deep learning methods. In real-world applications, a less capable but interpretable model is favorable to a black-box model with superior performance. Hence, it is crucial to have a transparent model with interpretable predictions to perform a task with high accuracy.

To achieve explainability and transparency in decision-making predictions without sacrificing their accuracy, generative models can be utilized to learn a meaningful feature representation for a given image, which can be further applied to downstream tasks (e.g., defect detection) or to estimate specific quantitative characteristics of data samples. Schmitt et al. demonstrated how a generative model in the form of StyleALAE could be

effectively applied to extract metrologically interpretable features for industrial machine vision applications [34]. An essential prerequisite to successfully performing quality control and metrology[1] tasks for generative models is to learn data features as consistently and accurately as possible to the features of original data samples. Hence, there is a strict need to determine whether a generative model has consistently captured features representative to quality characteristics. The current practice is to employ user studies in which humans conduct a visual examination. Such a costly and time-consuming approach has a high variance in its estimates, neither standardized nor validated, thus unreliable. However, there is no defined automatic evaluation strategy to assess the performance of generative models and verify how well they learned the human-interpretable features of real images. This study addresses the lack of an evaluation mechanism to verify whether the human-interpretable images were learned consistently from the training dataset (collection of real images) by a generative model.

One main generative modeling objective is to produce a condensed vector representation for a given high-dimensional input image, which is then analyzed to identify prominent image characteristics. The vector size is controlled by the predefined hyperparameter - latent space size. It is advantageous to have a limited number of features (i.e., small latent space size), as it is easier to interpret further. However, when the latent space size is too small, there is a risk of image features not being adequately learned. There is no established procedure to identify a model's optimal latent space size to produce an interpretable and disentangled vector representation. Thus, there is no available method to differentiate the models' performance with different latent space sizes.

1.2 Research questions

The thesis is set to address the following research questions:

- What are the available methods for feature consistency assessment and performance evaluation in deep generative modeling?
- How can the demand for feature consistency assessment be considered when evaluating the performance of generative models for industrial machine vision applications?
- Can a feature-consistent performance evaluation be used to identify an optimal value for the latent space size hyperparameter?

1.3 Structure of this thesis

This thesis is organized as follows. In Chapter 2 generative modeling field is introduced along with widely adapted methods to produce synthetic data. Related work, including the necessary concepts and terminology, is covered so that a reader can understand the thesis's setting and the generative modeling field in general. In recent years generative learning has been studied extensively, resulting in numerous advancements and rapid progress. However, it remains challenging to assess the generative model's performance reliably. The most popular qualitative (human evaluation) and quantitative (IS, FID, Precision and Recall) evaluation metrics are

[1] Metrology is the science of measurements. It refers to accurately and reliably measuring one or several specific parameters for a defined purpose.

also described. The chapter is concluded by indicating the lack of consistent and robust evaluation metrics to estimate the generative models' performance in real-world industrial applications.

Chapter 3 describes the experimental setting and provides details on how to train a specific class of generative models - adversarial latent autoencoders (ALAE). In Chapter 4, the evaluation framework is designed to assess the abilities of generative models to learn original dataset features consistently and reliably. It is tested in the simplified setting by verifying its effectiveness on a synthetic ellipse dataset. The newly developed evaluation approach is then applied on the real-world industrial dataset in Chapter 5. The evaluation framework is used to determine the optimal value for the latent space size hyperparameter of ALAE models that enables learning specific image features consistently and reliably. Chapter 6 concludes the thesis by analyzing the results and proposing avenues for future work.

2 Image Generation with Deep Generative Models

The goal of this chapter is to provide the necessary background for a reader to understand the main concepts and terminology to follow the material in the next sections easily. The field of generative modeling with Deep Learning is introduced at the beginning, including several prominent approaches and methods to produce synthetic data samples representing the original data set. Then there is an overview of the most common evaluation metrics in the field of generative learning. The limitations of widely-adapted metrics are highlighted, giving motivation for this thesis.

2.1 Generative modelling

Generative modeling is a class of models in the unsupervised learning field. The ultimate objective is to generate novel samples that resemble true data distribution using unlabeled training data. Training data is considered to be generated from some unknown distribution $p_{data}(x)$. The objective of generative modeling is to learn a model probability distribution $p_{model}(x)$ that approximates $p_{data}(x)$, so that new data points can be sampled from $p_{model}(x)$. Consequently, generative models are designed to address the probability density estimations of high-dimensional, complex distributions. Two main types of generative modeling accomplish this goal: explicit and implicit density estimations.

Autoregressive models [27] and variational autoencoders (VAEs) [22] are both approaches of explicit density models, which means that they explicitly define and represent $p_{model}(x)$. The prominent examples among autoregressive models are PixelRNN [27] and PixelCNN [28], which directly model the conditional distribution over pixels. The generated images are photo-realistic with a resolution larger than 256$\times$256, but the generation process is sequential and slow. In addition, the lack of latent representation can potentially limit the applicability of these models. Compared to autoregressive models, VAEs possess interpretable latent space and can generate new samples faster. However, produced results tend to be blurry and of lower quality.

Generative adversarial networks (GAN) [11] is a type of implicit density estimation that generates samples from $p_{model}(x)$ without explicitly defining it. Although generative models can produce different types of data (e.g., text, video, audio), this study is focused only on images, as the aim is to develop a model-based analysis of image generation quality for industrial machine vision.

2.1.1 Methods in Generative Modelling

Variational Autoencoders (VAEs)

Autoencoders are an unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data by encoding input image into a lower dimensional space z (often called latent space z). In order to capture meaningful factors of variation in input data sample, the latent vector z is by design smaller

than the input x, thus dimensionality reduction is achieved.

The traditional autoencoder architecture is provided in Figure 2.1. The encoder (or compression function) learns the mapping from the data x to a low-dimensional latent space z, while decoder (decompression function) learns the mapping back from latent vector z to a reconstructed observation of $\hat{x}$. Both encoder and decoder are represented as a neural network. Feature representation is learned by training the model with the objective to reconstruct original data from latent representation by means of reconstruction L_2 loss function $L_2(x, \hat{x}) = ||x - \hat{x}||^2$. Reconstruction loss forces the latent representation to encode as much "useful information" about the data inputs as possible. The dimensionality of latent space directly impacts the reconstruction quality. The smaller latent space will force a larger training bottleneck, as the bottleneck layer aims to simplify the input data and keep only essential information.

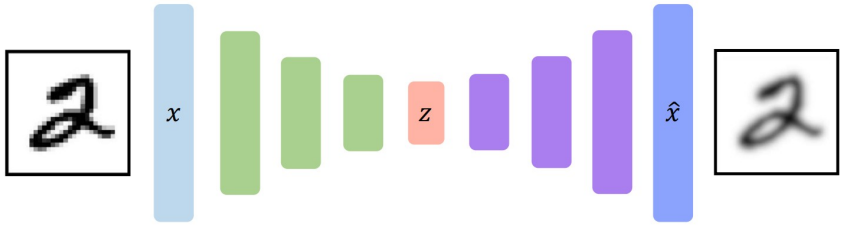

Figure 2.1: Autoencoder architecture. The model is trained to learn the reconstruction $\hat{x}$ of its original input x. The lower-dimensional space z in the middle is known as the latent space. It controls the compression rate of the high-dimensional input data and captures the prominent properties of input samples.

Autoencoders are a powerful representation learning technique. The bottleneck hidden layer z forces the network to learn a compressed latent representation that captures the prominent properties of input samples, including meaningful factors of variation in the training data. However, the traditional autoencoder architecture is not capable of generating new synthetic samples from the training data manifold. To add the generative capabilities, Variational Autoencoders (VAEs) architecture was proposed by Kingma et al. [22].

Variational Autoencoders is a probabilistic spin on autoencoders that allows sampling from the model to generate new data samples. The architecture is presented in Figure 2.2. The general assumption is that training data ${x^{(i)}}_{i=1}^{N}$ is generated from underlying latent distribution $p_\theta(z)$. VAEs are set to learn this probability distribution function's parameters instead of an arbitrary function as in traditional autoencoders. The simple Gaussian distribution is further imposed on the true prior distribution $p_\theta(z)$. The new synthetic data points are sampled from the true conditional distribution $p_\theta(x|z)$, which is represented as a neural network due to its ability to learn a complex mapping from latent codes to an image. Additionally to decode network modeling $p_\theta(x|z)$, an encoder network $q_\alpha(z|x)$ is defined that approximates $p_\theta(z|x)$. This allows to train the model and learn model parameters θ by maximizing the data likelihood lower bound. The loss function is

$$L(\theta, \phi) = E_{p_\theta(z|x)}[\log p_\theta(x|z)] - D_{KL}[(p_\phi(z|X)||p_\theta(z)], \tag{2.1}$$

where x represents the real data distribution and ϕ, θ are the parameters for the decoder and encoder of the VAE respectively.

A latent vector z is sampled from a known simple Gaussian distribution $z \sim \mathcal{N}(\mu, \sigma)$ to generate new data.

Different dimensions of latent space z encode different interpretable factors of variation. The trained decoder network (represented as a neural network) $p_\theta(x|z)$ transforms the latent vector into an image.

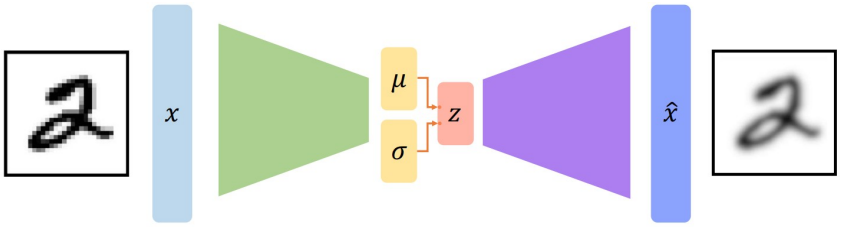

Figure 2.2: Variational Autoencoders architecture consists of encoder-decoder components as in traditional autoencoders. However, the encoder network encodes a distribution $p_\theta(z|x)$ for the latent code z given the input example x instead of directly encoding the latent code. The distribution $p_\theta(z|x)$ is usually represented as a Gaussian distribution $z \sim \mathcal{N}(\mu, \sigma)$.

Variational Autoencoders are unsupervised approaches that combine generative and representational properties by learning an encoder-generator map simultaneously. The model provides insightful data representations in its latent space and is capable of producing new samples. However, the generated images are unrealistic and blurry, which is a major drawback of VAEs, as suggested by Dosovitskiy et al. in [8].

Generative adversarial networks (GANs)

Due to the disadvantages of the explicit density estimation models in generative modeling, the primary focus of the thesis is devoted to the implicit density models, in particular, a family of generative adversarial networks (GANs). Originally introduced by Goodfellow et al. [11], GANs are a special type of architecture for training generative models. The idea is not to explicitly model the training data distribution but instead focus on sampling to generate new instances similar to the data points in a training set. The problem is that the true data distribution is unknown, so there is no direct way to sample from it. One way to address this challenge is to follow the adversarial approach.

The GANs architecture consists of two components: generator and discriminator. The generator $G_{\theta_g}(z)$ is a differentiable function, which is originally represented as a deep neural network parameterized with θ_g. The generator network receives random noise z (also called latent code) as an input and is trained to learn a transformation from this vector z to images $X_{fake} = G(z)$. The generator's goal is to produce images as close to the training distribution as possible. The discriminator $D_{\theta_d}(x)$ is also a differentiable function represented as a deep neural network parameterized with θ_d. It outputs the probability of the input x being real rather than fake. The discriminator network takes the fake images produced by the generator and real images from the training set as inputs and learns to classify fake data from real using traditional supervised learning techniques. GAN architecture is illustrated in Figure 2.3.

The training process is viewed as an adversarial competition between two players. The generator network, as player one, aims to fool the discriminator by generating real looking images, whereas the second player - the discriminator network - is trying to distinguish between real and fake images. These two networks are trained

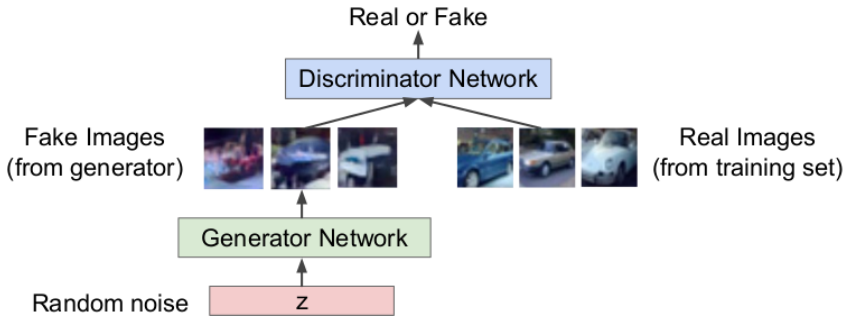

Figure 2.3: The GAN framework pits two adversaries against each other in a game. The generator network is trained to learn a transformation from latent vectors to fake images. The generator's goal is to produce images as close to the training distribution as possible. The discriminator network takes the fake images produced by the generator and real images from the training set as inputs and learns to classify fake data from real.

jointly in the minimax game. The original objective function is provided in the following formula:

$$min_{\theta_g} \ max_{\theta_d}[E_{x\sim p_{data}} log D_{\theta_d}(x) + E_{z\sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))] \tag{2.2}$$

where $D_{\theta_d}(x)$ is the discriminator output with a real data point x from training set as input, and $D_{\theta_d}(G_{\theta_g}(z))$ - discriminator output with a produced fake data point from generator $G_{\theta_g}(z)$ as input.

The discriminator D_{θ_d} aims to get the $D(x)$ close to 1 (real) and $D(G(z))$ close to 0 (fake), which means that discriminator network correctly classifies real images as real and generated images as fake. While the generator G_{θ_g} strives to have $D(G(z))$ close to 1, which means that the discriminator is tricked into thinking that generated sample $G(z)$ is real. Therefore the training process consists of alternating between gradient ascent on discriminator - trying to learn θ_d that maximizes the objective function, and gradient descent on generator - trying to learn parameters θ_g to minimize the same function.

By having these two deep neural networks competing against each other, the discriminator is trained to become as good as possible at distinguishing fake and real images. As the discriminator improves, the generator becomes better at producing new samples that are as close to real as possible. When the discriminator detects the difference between training and generated data distribution, the generator adjusts its parameters θ_g to eliminate the difference. It means that by backpropagating through discriminator and generator, the generator's parameters will be modified to make its outputs more confusing for the discriminator. In theory, Goodfellow et al. [11] have proved that with large enough models and unlimited data, the only solution is achieved when the generator is able to replicate the true data distribution exactly ($p_{data} = p_{model}$). The discriminator is guessing at random, unable to find a difference between the generated and training data distribution. After training the GAN model, the generator network learns the true data distribution and can produce new synthetic images of the exact nature of the original dataset.

Among the disadvantages of adversarial strategy is the non-convergence issue. GANs require finding the optimum objective function in a game with two players. Sometimes two players might undo each other's progress without reaching the optimum of the objective function. It leads to the equilibrium of the game not being reached, which is a general problem with the game-centric approach. Moreover, GANs can also suffer from mode collapse problems caused by generator producing only a few types of samples, which limits variation

in the generated images. Complete mode collapse is rare, but partial mode collapse is frequently observed. An example of partial mode collapse could be the scenario when multiple images containing different views of the same object are generated.

The challenges of traditional GANs architecture are alleviated by introducing specific extensions and advancements to the traditional model architecture, which are further covered in state-of-the-art GAN methods in this chapter.

2.1.2 Towards high resolution image generation

GANs have been proven to learn training data manifold in adversarial setting and generate artificial data samples. However, the original architecture by Goodfellow et al. [11] is operating only on low-resolution images (64×64) and has difficulties in generating high-resolution samples with high variability. In a high-resolution setting, the generator's task is more advanced than the discriminator's, as the generator has to learn all the coarse and fine-grained structures of the original dataset simultaneously from scratch. While the discriminator can easily detect flaws in samples produced by the generator, thus not providing a strong signal for model learning. This makes the mode collapse problem especially pronounced in high-resolution space, limiting the variability of the synthetic samples and prohibiting learning features of the original dataset accurately.

In contrast, machine vision tasks in industrial applications must operate on high-resolution images to perform decently well on different tasks. Thus, there is a need to investigate GANs-based architectures capable of generating high-resolution images and operating in a high-dimensional space.

Progressive growing of GANs for improved quality, stability, and variation (ProGAN)

Karras et al. [19] designed a new training methodology for GANs that resolves the high-resolution image generation issue, as displayed in Figure 2.4. The key idea is to progressively build the discriminator (D) and generator (G). The training starts with both networks having a low resolution of 4×4 pixels. With the training progress, new layers are incrementally added to the generator and discriminator, bringing in higher-resolution details until high-quality images of 1024×1024 are generated. The new training methodology indicates that learning a complex transformation from latent space to high-quality images is easier in steps. In this way, the generator is able first to learn the coarse large-scale structure of the images and then fine small-scale details as the training process evolves.

The progressive growing of GANs greatly stabilizes training since most of the training is carried out at low resolution. Karras et al. [19] explain this benefit with the intuition that increasing the resolution allows us "to ask simpler question" compared to the eventual goal of discovering a mapping from latent vectors to 1024×1024 images. Furthermore, the new architecture type reduces training time because the generator and discriminator are shallow and quick to evaluate when operating on low-resolution images. As a result, the progressive growing of GANs generates high-resolution images robustly and efficiently.

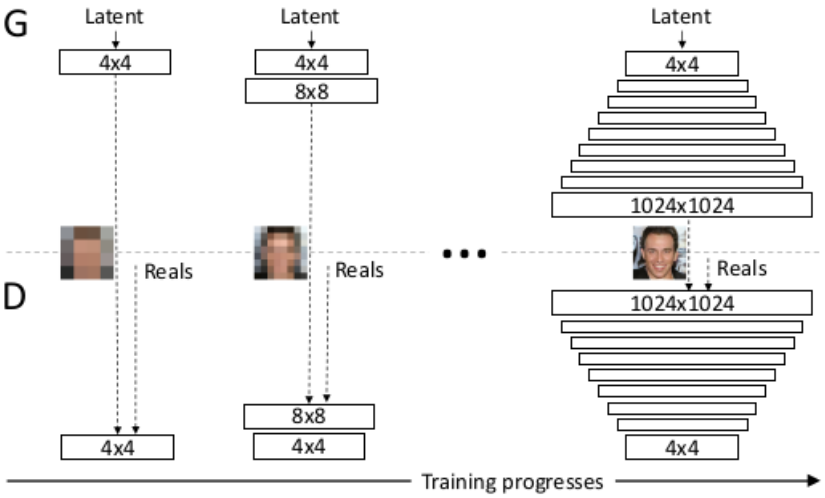

Figure 2.4: The progressive growing scheme of generator and discriminator. Training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, new layers are incrementally added to both G and D, thus increasing the spatial resolution of the generated images. Image from [19].

Style-Based Generator Architecture for Generative Adversarial Networks (StyleGAN) - state-of-the-art performance in image synthesis

The ProGAN model has achieved impressive results in image synthesis tasks in terms of image quality and resolution. However, the generator in GANs remains a black box. In the research paper [20] Karras et al. proposed an an alternative generator architecture for GANs (see Figure 2.5). The new architecture relies on a progressive GAN setup and provides novel ways to control the image synthesis process.

In contrast to the traditional architecture, the new generator has a non-linear mapping network $f : Z \rightarrow W$. This network is implemented as an 8-layer fully-connected neural network and acts as an embedding of the input latent code z (random noise) to the intermediate latent space W. Compared to traditional architecture, this modification provides a more linear, less entangled representation of different factors of variation to the generator, which further contributes to high-resolution image generation. Besides, Karras et al. observed that the generator does not benefit from having latent code as an input into the first convolution layer. Thus, the traditional input layer was removed and instead used a learned 4×4×512 constant tensor as a starting point for image generation.

The learned latent vector w as the source of features and information is repeatedly injected into the generator at each convolution layer. In this way, the synthesis network adjusts the "style" of the image at each layer, thus enabling to control the strength of image features at different scales. This modification is opposite to the traditional GAN formulation, where the latent vector is used just once as input and then never again. The latent vector w from intermediate latent space W controls the generator through adaptive instance normalization (AdaIN), the technique initially used in style transfer [16]. The AdaIN uses a learned affine transformation (shown as "A" in Figure 2.5) that maps the latent vector w to a spatially invariant style $y = (y_s, y_b)$ where y_s

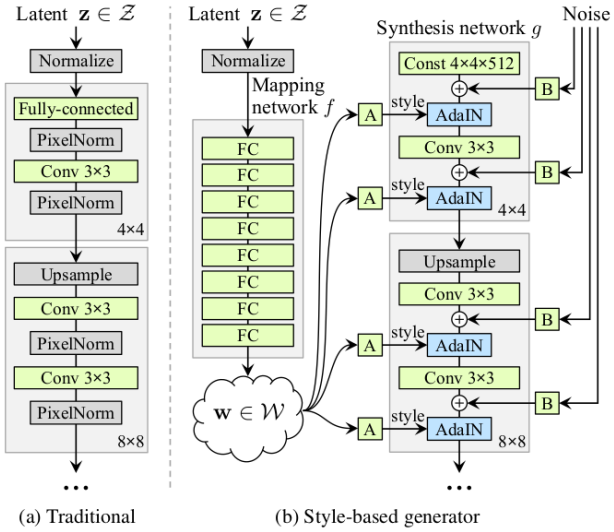

Figure 2.5: The StyleGAN generator architecture (b) compared to traditional architecture (a). Here "A" stands for a learned affine transformation, and "B" applies learned per-channel scaling factors to the noise input. Image from [20].

is a scale term and y_b - bias term. The AdaIN operation is defined as

$$AdaIN(x_i, y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \tag{2.3}$$

where each i-th sample of a batch x_i is normalized separately and then scaled and biased with a set of two corresponding scalars y_s and y_b that control the "style" of the generated image.

Karras et al. showed that changing a specific subset of styles can affect only certain aspects of an image. Thus the influence of each style is localized in the network. With the introduction of affine transformations and AdaIN operations, the generator can be viewed as a way to produce an image based on the collection of styles hence the name "style-based generator architecture".

The noise inputs are explicitly injected into the generator, represented as single-channel images consisting of Gaussian noise. The noise image is passed to all feature maps using learned per-feature scaling factors (shown as "B" in Figure 2.5). Karras et al. indicated that the noise only affects the stochastic aspects and does not impact the overall composition and high-level aspects (changing pose, identity, etc.). This modification allows the synthesis network to generate stochastic details. For example, in people's face generation, noise inputs will result in a few variations of the same image (e.g., minor changes like slightly different hairstyles, less/more freckles).

StyleGAN produces not only high-quality realistic images but also provides scale-specific control and understanding of the image synthesis process. The new generator architecture leads to better interpolation properties, better disentanglement of latent factors of variation, and automatic separation of high-level attributes (e.g., pose, identity) from stochastic variations (e.g., freckles, hair) in the generated images. StyleGAN model developed by Karras et al. is regarded as one of the best performing generative models. In order to objectively compare different models with the StyleGAN model, qualitative and quantitative evaluation metrics were used, which are covered in Section 2.2.

Adversarial Latent Autoencoder

The Adversarial Latent Autoencoder (ALAE) is an autoencoder-based neural architecture that combines representative properties of variational autoencoders with generative capabilities of GANs. In this way, ALAE is able to jointly learn disentangled representations (latent feature vectors) of the input image as well as generate new synthetic samples mimicking the training data samples.

The variational autoencoder approach assumes that a probability distribution should be fixed a priori for the latent space, and the autoencoder should learn to approximate it. This assumption enables learning meaningful representations but prevents from generating high-quality images comparable to GANs models. On the other hand, the StyleGAN study proved that the use of learned intermediate (and not fixed a priori) latent space has improved disentanglement qualities compared to imposed input space with known density distribution.

Based on this observation, a new autoencoder architecture was designed by Pidhorskyi et al. [29], in which the latent space distribution is learned from data to achieve more disentangled representations, and output data distribution is learned with an adversarial strategy (i.e. through discriminator). Figure 2.6 depicts the ALAE architecture as proposed in [29].

This architecture adopts the original GAN paradigm, but the generator G and the discriminator D components are further split into two networks. The generator is composed of 1) a deterministic mapping network F, which transforms the random input vector from latent space Z with known probability distribution to an intermediate latent space W; 2) a generator network G, which produces synthetic images from latent vectors in W including independent noisy input η with a known fixed distribution $p_\eta(\eta)$. The discriminator is made up of 1) an encoder network E, which expects either generated or real image as input and produces a latent vector in the same intermediate latent space W; 2) a discriminator network which yields the probability of a latent vector from W space originating from a real image. This is a key difference compared to the GAN framework - instead of identifying generated images and real images, the discriminator operates on a latent space and tries to differentiate real images from generated images based on their latent vectors W. Another advantage of operating on the latent space size instead of autoencoding the data space is that using reconstruction losses based on plain L_2 norm is no longer required since they are not optimal in image data space. The name Adversarial Latent Autoencoder (ALAE) also stems from this property.

Two types of adversarial latent autoencoders were designed: one based on the fully-connected neural network encoder (same as in vanilla GANs), another based on the StyleGAN generator, which is called StyleALAE. Since the later architecture leverages StyleGAN advancements, the generative capabilities are superior to the vanilla generator represented as a neural network. Therefore, in this study, the StyleALAE model will be used to get a representation for a given image and produce synthetic samples from the training data manifold.

StyleALAE is an autoencoder-based model capable of generating high-quality images comparable to the results of state-of-the-art StyleGAN-based models. At the same time, StyleALAE can produce reconstructions and manipulations for a given image, which by design is impossible to achieve with a generator-only type of architecture (i.e. StyleGAN). Additionally as shown in [29] the latent space w of StyleALAE results in less entangled representation compared to StyleGAN latent space w. This is a great advantage as the representation

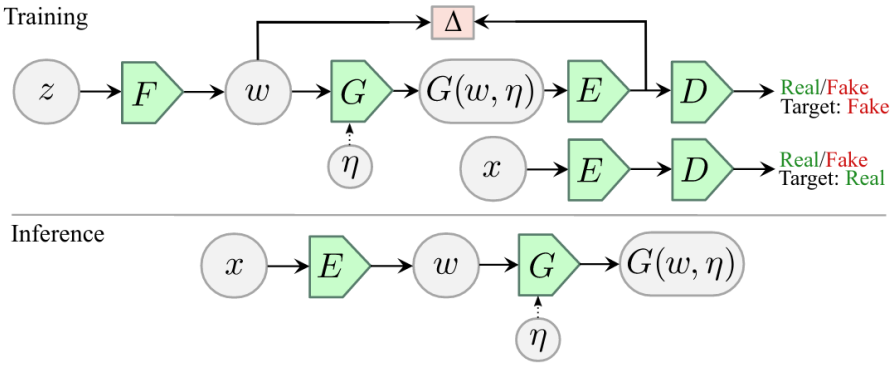

Figure 2.6: Adversarial Latent Autoencoder architecture. Figure adopted from [29].

feature vector $w \in W$ encodes meaningful information in a clear, compact form which is then necessary to find factors of variation in the feature space corresponding to certain characteristics of an input image.

2.2 Assessing Image-generation quality

In the methods of generative modeling section, it was mentioned that StyleGAN and StyleALAE models are regarded to be the best performing methods from the wide variety of designed GANs models and achieve state-of-the-art performance in the generation quality. However, it is not yet clear to the research community how to correctly measure the performance of generative models and compare different models with each other. The main challenge of evaluating GANs stems from the fact that the probability distribution p_{model} is intractable. Evaluation metrics are a critical research topic as they are necessary to identify modifications that result in better model performance and lead to better algorithms and their understanding. In the following subsections most prominent qualitative and quantitative approaches are presented, such as human evaluation, Inception Score (IS), Frechet Inception Distance (FID), and Precision, Recall.

Supervised learning vs. generative learning evaluation. Generally speaking, GANs evaluation is similar to evaluating other models by taking a model checkpoint (trained model) with weights frozen. Then compare the outputs by different models against a metric for a particular task. However, evaluating GANs is a challenging task. For example, in supervised learning, true labels are available for held-out test sets with which predicted labels are compared, and such metrics as accuracy, recall, and precision are calculated. However, in the GAN paradigm, the input in the form of random noise is transformed into a fake image. There is no generally-suitable way of measuring how realistic these generated images are. Given a random noise vector as input, there is no clear objective for what pixels a model is expected to generate. Additionally, there is no clear right or wrong output and no sense of "correctness". Borrowing the analogy from GANs research, the generative model is viewed as an artist learning how to paint masterpieces contrary to learning the exact brush strokes in a known painting as in supervised machine learning.

Using discriminator for evaluation. The discriminator component of GANs architecture is used to classify real versus fake images. So one might naturally think of using a discriminator to benchmark the performance

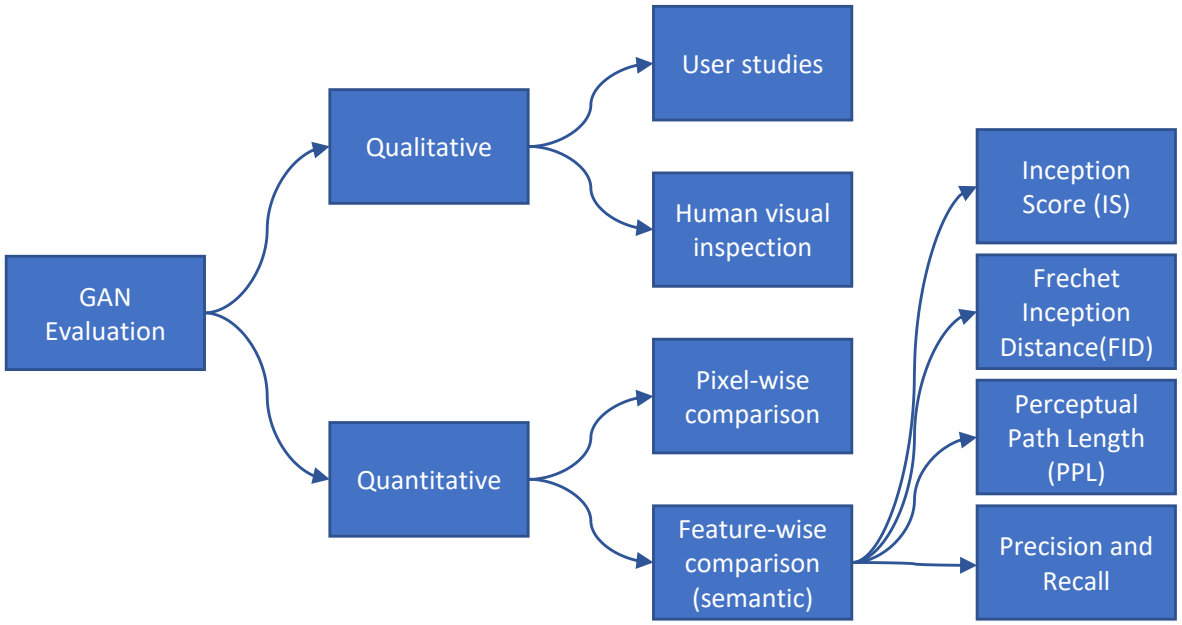

Figure 2.7: The flowchart demonstrates the GAN evaluation hierarchy and main types of metrics used to assess the the performance of generative models.

of different generators. However, there is some critical limitation that prohibits the application of this idea. The main shortcoming is that a discriminator is trained jointly with a generator, so it picks up on certain qualities (artifacts) of its particular generator. Thus, it overfits to discriminating real versus fake images for the current generator during training. As a result, a discriminator applied to another generator (from another training) most likely will have random predictions or predict either real or fake class all the time. Hence, no universal discriminators can be applied to multiple generators, not even mentioning the quantitative comparison of different generators.

Pixel difference as a measure of similarity. The simple approach to comparing sets of images is to look at the differences between their pixels, known as the pixel distance, but this method is mainly insufficient and unreliable. The idea is to subtract pixel values of fake images from real images, meaning values from 0 to 255 in one image from the other, and then get their absolute difference which can be summed over all real-fake image pairs. However, the pixel distance is not reliable. For example, if a fake image is shifted by only one pixel to the right, this would potentially have a colossal pixel distance from the unshifted image, even though the images are still really similar. Especially in the case of high-resolution images, the difference of shifting an image by 1 pixel is imperceptible, yet the pixel distance would change dramatically. Thus the approach based on pixel distance is unreliable and insufficient.

Evaluation in feature space. Alternative to the naive idea of comparing images on the pixel space is first to transform images to a feature space and then compare images in the feature space. Such higher-level semantic information would be less sensitive to small shifts and changes in the images, which further enables comparison using feature distance, leading to greater comparison reliability since it looks at higher-level information. Such an approach is quite common in the machine learning field. For instance, in Natural Language Processing (NLP) textual information is compared by first converting text into embeddings with the help of the BERT

model [7] and then applying cosine similarity to measure the difference between two feature vectors representing two texts.

Feature extraction. In order to compute the feature distance between real and generated images, the first step is to extract the features from those images. The typical approach in Deep Learning is to use a pre-trained model (i.e., weights of a pre-trained classifier on huge data sets) as a feature extractor. The weights of such a pre-trained model have essentially encoded many features useful for classification tasks. As a result, the input image can be converted into the feature vector by going through a forward pass of the pre-trained model. The most common choice for the feature extraction objective is to use the last layer before the fully connected neural network, where the actual predictions for classes are produced. At this pooling layer, it is regarded that the model has learned the most fine-grained feature information needed for classification and, thus, a good feature representation of the input image. It is also possible to use activations of earlier layers of a pre-trained network, which are better for getting more primitive information. Although, it has to be stated that the extracted features are abstract and do not necessarily correspond to features known to a human. A large neural network has learned such representation in a black-box fashion, completely hidden from a human.

2.2.1 Qualitative evaluation

As in other fields, human evaluation remains the gold benchmark for assessing the generative quality of synthetic samples. The most intuitive qualitative approach is a visual examination of samples by humans, as in [33] and [6]. It is essential to employ human judgments to evaluate the realism of generated samples for most generative modeling tasks. This evaluation approach possesses one major advantage - the ability to distinguish generated images from real ones, which allows for model comparison based on the quality of generated samples. Hence, human visual perception fulfills the desired metric property - discriminability.

This approach might play a decisive role in choosing a suitable model for some applications. For example, based on the results of human evaluations, VAEs [22] are considered to produce images of lower quality than GANs [11]. However, direct human evaluation is subjective, neither standardized nor validated, time-consuming, costly and has a high estimates variance. Furthermore, evaluating the generative models is not only about the judgments of how realistic-looking produced images are. As discussed in Section 2.1.1, GANs are prone to mode collapse problems when the generator is inclined to produce similar images (a few "modes" of true distribution). Therefore, the visual inspection measure will favor models with high-quality generated images, even though these models could be limited in variation or even represent memorized training samples.

To facilitate the reliability and effectiveness of human evaluation, Zhou et al. [40] proposed a benchmark for Human eYe Perceptual Evaluation (abbreviated as HYPE) of generative models in order to set a reliable human assessment of generative realism. In contrast to direct human evaluations, this approach is grounded in psychophysics research in perceptual psychology, reliable across different sets of randomly sampled outputs from a model, able to capture separability between models, and (4) efficient in cost and time. The improved human evaluation metric produces reliable and separable scores of model assessment, which have been shown to correlate with human judgements. Although HYPE measures the sample realism of GAN models in an inexpensive and widely accessible method, it remains a human benchmark. Hence, it has some limitations

inherent to human evaluation, namely, the inability to capture diversity, overfitting, entanglement, training stability, and computational and sample efficiency of the model. Thus, the authors conclude that this evaluation method does not substitute the existing automatic methods but is meant to complement them.

2.2.2 Evaluation metrics in GANs field

Unlike human evaluations, quantitative measures are less subjective and designed to estimate visual fidelity and other characteristics such as variation and overfitting. However, they might not correspond to human visual perception. Thus, automated metrics are often justified by indicating a correlation with human evaluation [15]. This section covers widely adapted sample-based automatic metrics such as Inception Score (IS), Frechet Inception Distance (FID), and Precision, Recall. These metrics operate on feature space to are "model agnostic", meaning they can be applied to any model since the generator is considered a black box to only sample images.

Inception Score

The Inception Score (IS), proposed by Salimans et al. [33], is an automatic evaluation measure used for assessing the generated samples produced by GANs. The objective is to simultaneously estimate two properties of synthetic images: (1) diversity of images within a class label (e.g., each image is a different breed of dog) and (2) classifiability of images, meaning that an object on the image has to be easily classified. A higher score means the GAN model is able to generate high-quality distinct images.

Before calculating the IS itself, we have to apply the pre-trained Inception model [37] to every generated image to get the conditional label distribution $p(y|x)$. The observation is that easily classifiable images (i.e., samples with good quality) are expected to have label distribution $p(y|x)$ with low entropy. Since the model is also expected to produce varied images, the marginal distribution $p(y)$ should have high entropy. These two requirements fulfill the goal of having generated images of high quality and diversity. By combining these two requirements, the metric is defined as:

$$IS(P_g) = \exp(\mathbb{E}_x[\mathbb{KL}(p(y|x) \,||\, p(y))])$$

where $p(y|x)$ is the conditional label distribution for image x and $p(y)$ is the marginal distribution with $p(y) \approx \frac{1}{N}\sum_{n=1}^{N} p(y|x_n = G(z_n))$. The score measures the average Kullback-Leibler (KL) divergence between the conditional label distribution $p(y|x)$ of images and the marginal distribution obtained from all images. The IS measure provides model evaluation based on the quality and diversity of generated images.

Salimans et al. [33] confirmed that the IS score correlates reasonably well with human perceptual judgments, thus making IS a trustworthy metric. Besides, Borji et al. [3] showed that the metric is computational and sample efficient, meaning that it is computed relatively fast and requires less than 10.000 samples.

However, IS measure has significant limitations and shortcomings. First, the Inception Score relies on the Inception Network - a pre-trained ImageNet model - to compute statistics on the generated images. Previous work by Barratt et al. [2], by Rosca et al. [31] and by Borji et al. [3] discredited IS for its application to non-ImageNet datasets. They argue that evaluating generative models trained on other datasets with the IS

gives misleading results. Barratt et al. [2] conclude that IS can be used as an evaluation metric of a generative model if it was trained on the same dataset or at least the same image classes as in ImageNet.

Moreover, Barratt et al. [2] showed that IS is unable to detect overfitting. A generator that memorized a subset of images from true distribution would perform well in the Inception Score metric. It is also indicated that the IS can not detect the mode-collapse problem. In addition, the Inception Score by design is developed to measure the "objectness" of images [33]. In other words, the metric checks the condition of generated images to contain clear objects, e.g., whether images are sharp rather than blurry. However, having a clear object in the image does not necessarily mean that the image will be realistic. In [41], Zhou et al. demonstrate that the Inception Score mainly acts as a diversity measurement, and there is no reliable evidence that it can reflect the true sample quality.

Frechet Inception Distance (FID)

As mentioned in Section 2.1, the objective of generator modeling is to learn a model probability distribution $p_{model}(x)$ that approximates real data distribution $p_{data}(x)$. Intuitively, one way to evaluate the performance of GANs is to compare the statistics of true data with those of generated data. Inception Score is designed to compute statistics only of produced images. To further enhance the IS robustness, Heusel et al. [14] designed a Frechet Inception Distance (FID) - metric to measure the similarity of generated images with real ones. Frechet Inception Distance (FID) is the most popular metric to evaluate the quality of synthesized images by generative models compared to real images. The metric name combines Frechet distance and the Inception-V3 Model [37].

Still, measuring the similarity is challenging since the true data and model distribution are not explicitly defined in GANs. As a solution, Heusel et al. [14] suggested embedding the images into a feature space given by the last pooling layer of the Inception-v3 model. Then, they assumed that the retrieved vision-relevant features follow a continuous multivariate Gaussian distribution, so the mean and covariance can be estimated for both the generated and the real data. Finally, the Frechet Distance (also known as Wasserstein-2 distance) is used to measure the difference between two Gaussian distributions of generated and real features and is given by:

$$FID(r, g) = ||\mu_r - \mu_g||_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}})$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariance of the sample embeddings from the data distribution and model distribution, respectively.

The FID metric calculation steps and outputs after each step are the following:

1. Sample (N >10.000) generated and real images to reduce noise and selection bias as well as to have a large support for the estimation of sample distribution.
   Output: set of real images, set of fake images.

2. Feature extraction technique: Inception-V3 Model takes every image as input and generates a feature vector (2048-length) representing the input image. Inception-V3 is a neural network trained on the

ImageNet dataset to classify images into dozens of target classes. FID uses the output from a previous layer — the layer right before the labels. This is often called the feature layer, as the output of this layer is a feature embedding of the input image. Research has shown that deep convolutional neural networks trained on complex tasks, like classifying many classes, build increasingly sophisticated representations of features going deeper into the network. For example, the first few layers may learn to detect different edges and curves, while the later layers may have neurons that fire in response to human faces.
Output: set of real features (embeddings), set of fake features (embeddings)

3. The real and fake embeddings are assumed to follow two multivariate normal distributions, as they can approximately model many modes in image features. For real and fake embeddings, fit a multivariate normal distribution, meaning find the mean vector and covariance matrix across N real and fake embeddings.
   Output: μ_{fake} (mean of fake embeddings), μ_{real} (mean of real embeddings), Σ_{fake} (covariance of fake embeddings), Σ_{real} (covariance of real embeddings).

4. Calculate the Frechet distance between two multivariate normal distributions. In the study "The Frechet distance between multivariate normal distributions" by Dowson and Landau (1982) [9], the Frechet distance between two multivariate normal distributions is defined as:

$$FID(r, g) = ||\mu_r - \mu_g||_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}$$

   where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariance of the sample embeddings from the data distribution and model distribution, respectively.

   FID looks at statistics (mean, covariance) of real multivariate normal distribution and the statistics (mean, covariance) of fake multivariate normal distribution - and then calculates how far apart those statistics are from each other. The closer those statistics are to each other, the closer the fake embeddings to the real embeddings.
   Output: FID score quantifying the feature distance between real and generated images.

In this way, the FID metric quantifies the quality of generated samples by calculating the distance between the generated and real feature distributions. The lower the FID value, the better.

Heusel et al. [14] demonstrated that FID is consistent with human visual perception and is more robust to noise than IS. The FID metric captures the image distortions (e.g., Gaussian noise, blur, black rectangles inpainting). With the increasing degree of such disturbances, the FID value monotonically increases, whereas the IS stays flat, decreases, or fluctuates. Moreover, the model with a mode collapse problem can have a perfect IS but will have a bad FID, meaning that, unlike IS, FID is able to detect mode dropping. In addition, FID fulfills other desired properties of a metric, such as discriminability, robustness, and computational efficiency [3].

Even though FID possesses several advantages compared to IS, it also possesses certain drawbacks and disadvantages. Like the IS, FID depends on the Inception-v3 Network - a deep convolutional neural network trained on the ImageNet dataset. The validity of applying these metrics to non-ImageNet datasets is questionable

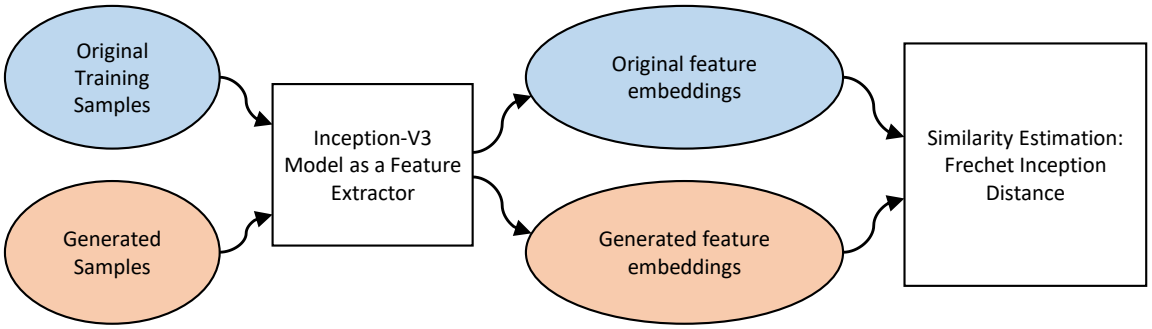

Figure 2.8: A schematic procedure to compute Frechet Inception Distance between original and generated samples.

[2], [31], [3]. The Inception-v3 model was pre-trained on the ImageNet dataset. Thus it does not necessarily output good feature embedding for images different from images in ImageNet. For example, suppose a generative model is trained on the MNIST dataset to generate handwritten digits. In that case, the Inception-v3 model might not be able to detect meaningful features since ImageNet is composed of natural photos, unlike grayscale handwritten digits.

Furthermore, the inability to detect overfitting is still present in the FID measure and IS. For example, the model which stores a subset of images from the training set ("memory GAN") would have a perfect score. Additionally, the assumption that vision-relevant features retrieved with the Inception Model follow the Gaussian distribution may often not be guaranteed. Lastly, the mean and covariance do not cover all aspects of the distribution. Other distribution properties like skew and kurtosis characterize the distribution and are useful to differentiate between different distributions. To compute FID, one needs to estimate the mean and covariance of the real and generated distributions by considering sample images from both sets. It leads to the inherent variance within the metric, as FID depends on the number of images chosen from both sides. However, choosing a bigger sample size and spending more time on FID computation will yield a better FID score [25]. The rule of thumb is to choose a big sample size of real and generated samples to obtain robust FID scores.

Perceptual Path Length (PPL)

Interpolation of latent space vectors might result in surprisingly non-linear changes in the image [24]. For instance, image features that are not present in either endpoint might appear in linear interpolation path. This is an indicator of entangled latent space where the factors of variation are not separated in latent vectors. To quantify the degree of latent space entanglement, perceptual path length (PPL) was introduced in [20]. Even though the authors indicated a correlation between PPL and image quality, it neither directly estimates the image fidelity nor measures how well a generative model learned certain original image characteristics. It is used to identify whether a latent space is sufficiently disentangled in order to find directions in latent vectors that correspond to individual factors of variation in images.

Precision and Recall

The authors also highlighted another limitation of IS and FID metrics - they are single-value metrics, which is why they do not provide direct insights into the quality and diversity of the generated images. For example, when approximating a distribution of face images, a model containing only male faces of high quality and a model containing both genders with blurry faces may have equal FID values. In the approach introduced in [32], precision can be intuitively viewed as the measure of the quality of generated images, whereas recall describes the proportion of the true distribution covered by the model distribution (quantifies the diversity in the fake images). These metrics are illustrated in Figure 2.9. Precision and recall scores can distinguish the mode-collapse (poor recall) and bad quality (poor precision) of generated samples.

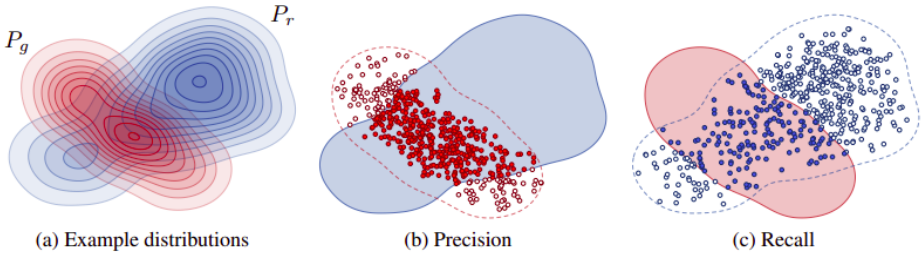

Figure 2.9: Definition of precision and recall scores. (a) Denote the distribution of real images with P_r (blue) and the distribution of generated images with P_g (red). (b) Precision is the probability that a random image from P_g falls within the distribution of P_r. (c) Recall is the probability that a random image from P_r falls within the support of P_g. Image adapted from [23]

2.3 Intermediate Conclusion

There are available methods in the generative modeling field used to produce high-quality samples and to learn a latent representation for input images in unsupervised settings. StyleALAE is the most prominent example that combines the generative power of state-of-the-art StyleGANs models and the representation properties of autoencoders. Schmitt et al. leveraged this architecture to extract metrologically interpretable features for industrial machine vision applications [34].

Currently, a plethora of evaluation metrics (see [3]) was developed to assess generative models. Nevertheless, there is no consensus as to which measures capture the strengths and limitations of generative models the best and should be used for fair model comparison. As described in the chapter, even widely adapted metrics such as Inception Score and Frechet Inception Distance have considerable shortcomings and limitations. Evaluation is an open area in generative modeling research. In response to the lack of robust and consistent evaluation metrics, we propose an evaluation framework consisting of ensembles of two evaluation techniques.

The developed evaluation framework described in Chapter 4 is primarily motivated by industrial machine vision application requirements. Even though fidelity and diversity are important evaluation criteria, we further frame another desired property, namely, the ability of GANs to capture the distribution of a particular feature from the original data set. In real-world use cases, the trained model should capture the distribution of a particular

feature from real data samples precisely and consistently because the generated data should possess the same quality characteristics as the original data. How a generative model learns a particular feature distribution can be used to further differentiate one model's performance from another. The objective of a developed evaluation procedure is to quantitatively measure how well generative models could learn the original features of the training dataset.

The feature vector leaned by the generator for a given image is of great interest since it provides valuable information to perform application-specific tasks in the industry. This produced feature representation has to be further analyzed and interpreted to identify specific patterns and properties of the input image. The smaller the latent space size, the easier it is to interpret. However, if it is too small, the model will fail to learn useful information. Hence a tradeoff has to be determined. With the evaluation procedure based on feature consistency estimation, it is also possible to estimate the optimal latent space size of a given generative model. The latent space size plays a crucial role in representative and generative capabilities. Thus there is a need to estimate the hyperparameter value robustly. The goal is to have a trained StyleALAE model with an optimal hyperparameter value that can consistently learn the desired features of original images in the training data set.

3 Training StyleALAE models for metrologically interpretable feature extraction

This study aims to quantitatively assess the capabilities of StyleALAE models to learn the interpretable features of the original dataset consistently. In this chapter, the experimental setup for training StyleALAE models is introduced and described in detail. The trained models (also called checkpoints) are then used for two main purposes 1) representation learning - get a vector representation by embedding a new input image into a latent space; 2) generative modeling - produce synthetic samples by learning the original data manifold in an unsupervised fashion.

3.1 Experimental setting

The official repository for working with StyleALAE models is publicly released by Pidhorskyi et al. at the following link[1]. The source code was written in Python programming language with the deep learning library PyTorch. The model's hyperparameters are configured using a YAML configuration file, which is displayed in Figure 3.1 as an example. The dataset section provides information about input data, including its size, resolution level, and location where the data is read. To initialize the model architecture for training, the following model hyperparameters are specified: learning rate for the Adam optimizer [21], how many iterations over the training dataset (epochs) per each resolution level, the dimensionality of latent space, the batch size for mini-batch learning and additional parameters referring to learning rate decay required for optimization. Most values are chosen either according to the training set or as the default values provided in the original paper [29].

A machine learning algorithm can output very different results depending on the architecture, hyperparameters, random initialization (i.e., a random seed for initial network weights), or the data set. Different random seed values may influence the final results, even with everything else being fixed. Hence, each training experiment is run with six random seeds (corresponding to six trained models, also called repetitions) to avoid reliance on random initialization and have robust relative model comparisons. Additionally, all the training parameters in experiments are constant to avoid their influence on end model performance and isolate the model performance to only depend on the latent space dimension.

The only varying hyperparameter in experiments is the latent space size. Arguably, the most important parameter when designing autoencoder-based architectures, which controls the compression rate and impacts both the output of autoencoders and input for the generative process. The encoder may overfit the data and thus suffer in accuracy if the latent space dimension is too big. On the contrary, if it is too small, the bottleneck will not have enough capacity to encode the high-dimensional information into a small vector size,

[1] https://github.com/podgorskiy/ALAE

```
NAME: textures_32
DATASET:
  PART_COUNT: 2
  SIZE: 18062
  FFHQ_SOURCE: /data/datasets/textures_all_256_tfrecords/textures_all_256_tfrecords-r%02d.tfrecords
  PATH: /data/datasets/textures_all_256_tfrecords/tfrecords/textures_all_256-r%02d.tfrecords.%03d
  MAX_RESOLUTION_LEVEL: 8
  SAMPLES_PATH: 'no_path'
MODEL:
  LATENT_SPACE_SIZE: 32
  LAYER_COUNT: 7
  MAX_CHANNEL_COUNT: 256
  START_CHANNEL_COUNT: 32
  DLATENT_AVG_BETA: 0.995
  MAPPING_LAYERS: 8
  CHANNELS: 1
OUTPUT_DIR: /code/results/training_artifacts/textures_ls_32
TRAIN:
  BASE_LEARNING_RATE: 0.002
  EPOCHS_PER_LOD: 6
  LEARNING_DECAY_RATE: 0.1
  LEARNING_DECAY_STEPS: []
  TRAIN_EPOCHS: 200
  #                   4    8   16    32    64    128    256
  LOD_2_BATCH_8GPU: [512, 256, 128,   64,   32,    32,    32,       32,        32]
  LOD_2_BATCH_4GPU: [512, 256, 128,   64,   32,    32,    32,       32,        16]
  LOD_2_BATCH_2GPU: [512, 256, 128,   64,   32,    32,    16]
  LOD_2_BATCH_1GPU: [512, 256, 128,   64,   32,    16,   16]

  LEARNING_RATES: [0.0015,  0.0015,   0.0015,   0.0015,  0.0015,   0.0015,     0.002,     0.003,    0.003]
```

Figure 3.1: An example of configuration file with hyperparameters and their values to train a StyleALAE model.

leading to information loss and incomplete reconstruction. Identifying a tradeoff between these extremes is currently based on heuristics, which do not explain why a particular choice of dimension may seem to work well.

4 Development of a model-based evaluation procedure to verify feature consistency of synthetic samples produced by trained StyleALAE models

In this chapter, an evaluation framework is designed and developed to justify whether and how well the trained StyleALAE models are able to learn different image features present in the original dataset. The StyleALAE models are fixed, meaning the model architecture and hyperparameters remain the same, except for the varying latent space size. Per each latent space size, six different StyleALAE models were trained by altering random seeds to avoid reliance on a single random result. The evaluation framework aims to identify the optimal latent space size, resulting in consistent original features being learned. The general evaluation pipeline consisting of multiple stages is illustrated in Figure 4.1.

The first step is to use trained StyleALAE models (see Chapter 3) to generate artificial data samples by learning to approximate the original data manifold. Considering just a few samples produced by a generative model will not yield meaningful results. The model suffers from a mode collapse problem (explained in Section 2.1.1) when it is able to produce a limited number (low variability in outputs) of realistic examples. To identify such negative properties of a trained generative model, a substantial amount of generated samples must be compared to the original samples. Generally speaking, the more data samples used in the evaluation, the more robust and reliable the results will be. The rule of thumb in the generative modeling field is to use at least 10k from the original and synthetic set. In order to avoid computationally heavy calculations in experiments, in the following experiments, the number of original images is set to 10k, and a model is set to generate the same amount to have a fair comparison.

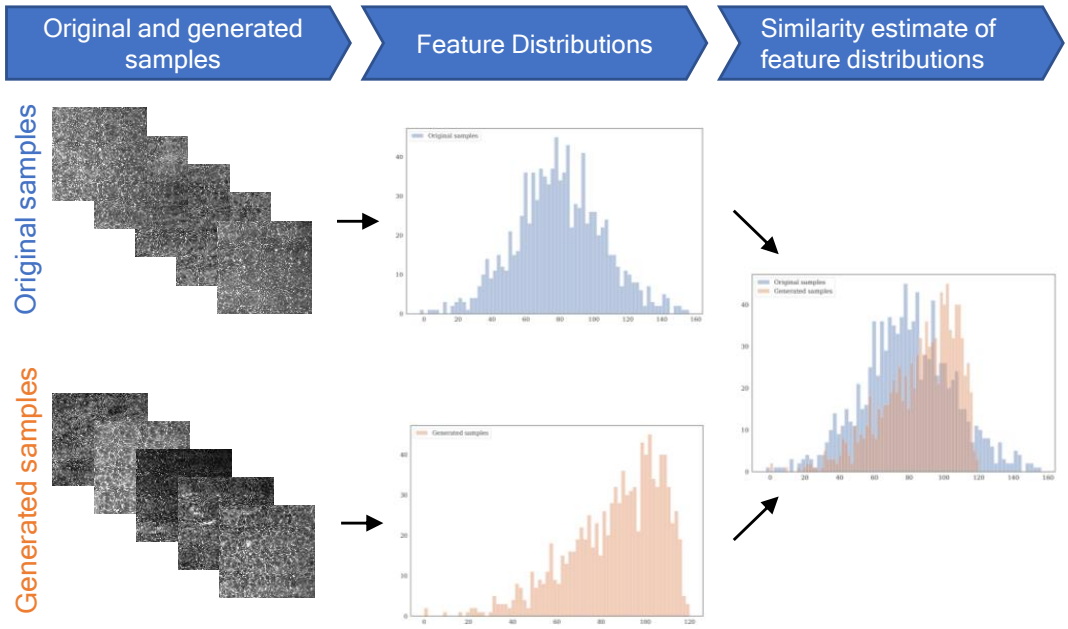

Figure 4.1: The Figure displays the main stages of the designed evaluation framework. The performance of a generative model is evaluated by first producing its synthetic samples and comparing them to original samples in a feature space. To this end, certain features must be extracted from samples, which are stacked together, resulting in a feature distribution for a particular dataset. The last stage estimates the similarity between two feature distributions of a generated and original dataset.

4.1 Synthetic ellipse dataset with known features

For general datasets, estimating feature consistency between original and generated sets might be challenging and ambiguous due to intangible feature definition and uncertain feature extraction methods. However, feature consistency can be efficiently estimated if a data manifold is constructed, so that image features are well-known and straightforward for a StyleALAE model to learn. To this end, a synthetic ellipse dataset is proposed to provide a set of meaningful, human-interpretable features. Ellipse denotes a simple geometric figure with known properties - location of the center ($x\ center$, $y\ center$), ellipse rotation $angle$, $major\ axis$ and $minor\ axis$ of an ellipse. By randomly varying the values of these five features, 10.000 gray-scale ellipses represented as one channel images were simulated. Some examples are displayed in Figure 4.2. The simulated artificial samples belong to a low-dimensional manifold embedded in $R^{64\times64}$. Intuitively, the axes of variation (ellipse characteristics) are represented by the coordinate system of this manifold. The assumption is that these factors of variation should be learned and captured by a well-trained generative model.

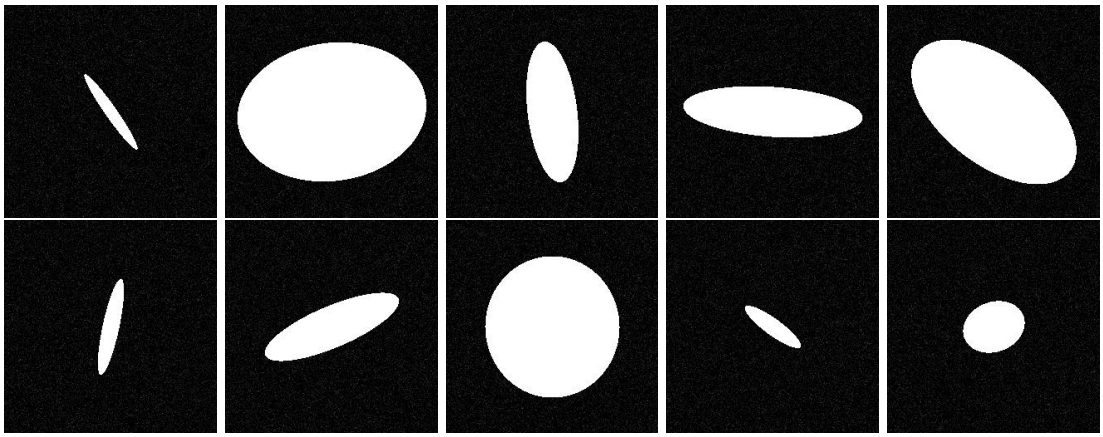

Figure 4.2: Examples of ellipses from simulated ellipse training dataset. This collection of samples will have the role of a real dataset which StyleALAE models will be trained to model and approximate.

4.2 Trained StyleALAE models

As described in Chapter 3, the following StyleALAE models were trained on the ellipse dataset introduced in Section 4.1.

Latent space size	L4	L8	L32	L128	L512
Number of repetitions	6	6	6	6	6

Table 4.1: Summary of trained StyleALAE models on ellipse dataset.

4.3 Ellipse feature extraction procedure

Following the strategy and concept of most common evaluation metrics in machine learning, the evaluation framework is designed to operate on image feature space rather than pixel space (see Section 2.2). In the previous step, 10.000 ellipse images were simulated to play the role of the original data set for training StyleALAE models. Then the trained models were used to produce 10.000 synthetic data samples during inference. The main advantage of the synthetic ellipse data set is that image features are known. Hence the current task is to extract robustly such features as the location of the center ($x\ center$, $y\ center$), ellipse rotation $angle$, $major\ axis$, and $minor\ axis$ of an ellipse. The step-by-step ellipse feature extraction method is described in Figure 4.3.

The first step in the ellipse feature extraction pipeline is to detect ellipse contours on the image. *Contour* is a curve joining all the points along the object boundary. The implementation of the algorithm introduced in [36] is provided by an OpenCV library. In order to improve accuracy and make contours extraction easier, the input grayscale image (pixel values varying in the interval from 0 to 255) has to be converted into a binary representation (pixel values either 0 or 255). The input images were binarized by performing an image segmentation task via straightforward thresholding. Noise is removed from the image by keeping only the most extensive object (with largest contours) on the image.

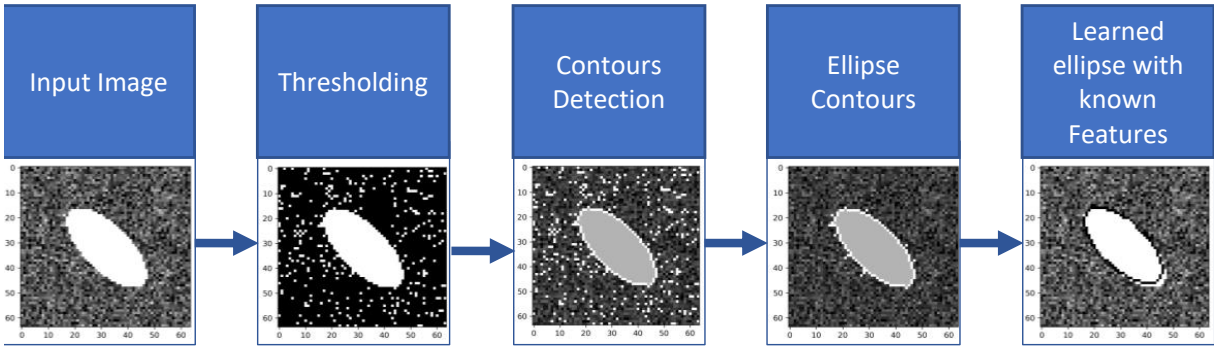

Figure 4.3: Ellipse feature extraction steps.

The output of the contours extraction step is the set of 2D points describing an ellipse in the image. This is a necessary input for the algorithm described by [10] to fit an ellipse (with known features) around a contour. The algorithm implementation in the OpenCV library is used to compute the ellipse that fits in a least-squares sense a set of 2D points best of all. As a result, the algorithm returns the rotated rectangle in which the ellipse is inscribed. The known features of a modeled ellipse are then used to describe an ellipse on the image.

As a result of feature extraction procedure, the example image illustrated on the Figure 4.3 has the following estimated features: (x $center$, y $center$) center location - (31.77, 32.11),$major$ $axis$, and $minor$ $axis$ - (18.04, 39.26), $angle$ - 136.35, which can be concatenated in a vector of length five in a form (31.77, 32.11, 18.04, 39.26, 136.35). The numeric values are measured in pixels. Extracting features for every image in the original set and synthetic set leads each to the 2D array of size $(10000, 5)$, where 10000 is the number of samples, and 5 is the dimensionality of the feature vector. Such ellipse feature representation allows qualitatively verifying feature consistency by visualizing the distribution of a particular feature from the original set versus the synthetic set.

In the form of histograms, the distribution comparisons were plotted for $angle$, and $major$ $axis$ features extracted from the original dataset and generated samples produced by models with different latent space sizes. Models with latent space sizes 4 and 8 learned $angle$ and $major$ $axis$ features worse than models with other latent sizes since the feature distributions are visually different from original feature distributions. However, it is unclear how to differentiate between the feature distributions of models with latent space sizes 32, 128, and 512, as their shapes are similar to the original feature distributions. Hence, there is a clear need for quantitative estimation of the difference between two feature distributions representing generated and original samples.

Figure 4.4: The left side of a figure displays the $angle$ distribution, while the right side - $major\ axis$ distribution. The five plots show the feature distribution of real images (blue color) and the distribution of features extracted from generated images (orange color) obtained by five model variations. From top to bottom, the model latent space sizes are 4, 8, 32, 128, and 512. The bin step of size two is chosen to plot histograms.

4.4 Feature distributions comparison via statistical hypothesis tests

Now that the distributions of each ellipse feature have been calculated and visualized for synthetic and original images, another question arises how to quantitatively compare them (last step as displayed in Figure 4.1). Having two distributions represented by two histograms, one can paraphrase the question into "Are these distributions consistent?" meaning verifying whether two histograms are consistent with having been sampled from the same parent distribution. There are multiple hypothesis testing statistical tests to address whether two collections of data points are consistent with being drawn from the same continuous distribution, as reported in [30].

Significance tests work by first defining the null hypothesis (things are happening as expected, e.g., samples come from the same distribution) and alternative hypothesis (unexpected event, e.g., samples do NOT come from the same distribution). Then a threshold has to be set up, known as significance level (alpha). Before conducting experiments, the confidence level of a test is specified in advance, equaling the standard value of 5%. Based on the sample realizations, the corresponding test statistic is calculated. The next step is calculating the p-value - the probability of obtaining a sample test statistic given that the null hypothesis is true. Then the p-value is compared against a predefined significance level. If the p-value is lower than the significance level, the null hypothesis is rejected, and there is evidence suggesting the alternative hypothesis. The null hypothesis cannot be rejected if the p-value is greater than the significance level.

Multiple statistical tests were adapted to determine whether two histograms have the same underlying distribution. According to Porter et al. [30], the most common tests are the following:

- Two-sample Kolmogorov-Smirnov (KS) test estimates the underlying distributions of two independent sets of data samples [18]. Given two sets of random variables observations of original and synthetic features, it outputs KS test statistics and the associated p-value. The null hypothesis is that two distributions are identical, whereas the alternative is that they are not identical ('two-sided' set).
- The Mann-Whitney U rank test [26]. Non-parametric test to check whether the null hypothesis - the distribution underlying sample x is the same as the distribution underlying sample y. The alternative hypothesis is that the distributions are not equal.
- Two-sample Cramér-von Mises test [1]. For two independent sets of samples, it verifies the null hypothesis whether samples come from the same continuous distribution, whereas the alternative hypothesis is that samples come from different distributions.

The statistical hypothesis test results are provided in Table 4.2 and Table 4.3.

By comparing the pre-defined significance level and the test p-value, the conclusion can be made whether the null hypothesis of the corresponding test can or can not be rejected. From the Table 4.3, the p-value values of every test are lower than the significance level of 5%. This implies sufficient evidence to reject the null hypothesis meaning the distribution of the generated features of any latent space size is not the same or similar to the distribution of the original features. The same conclusion is drawn from the table Table 4.2, with the

Latent space size	(a) KS test (statistic, p-value)	(b) MW test (statistic, p-value)	(c) CM test (statistic, p-value)
L4	(0.13221, 1.61e-31)	(49844441.5, 0.00031)	(20.37574, 1.17e-09)
L8	(0.04925, 6.16e-11)	(50325260.5, 0.00528)	(1.88771, 2.28e-05)
L32	(0.01888, 0.05651)	(48912275.0, 0.13996)	(0.30306, 0.13251)
L128	(0.02594, 0.00239)	(50345188.0, 0.04312)	(0.68854, 0.01357)
L512	(0.02898, 0.00044)	(48417087.0, 0.00141)	(1.22908, 0.00072)

Table 4.2: Statistical hypothesis tests results per one StyleALAE configuration (repetition 0) of latest space size in values [4, 8, 32, 18, 512] for the $angle$ ellipse feature. Comparison of models with different latent space size by leveraging (a) Two-sample Kolmogorov-Smirnov (KS) test; (b) The Mann-Whitney U rank (MW) test; (c) Two-sample Cramér-von Mises (CM) test.

Latent space size	(a) KS test (statistic, p-value)	(b) MW test (statistic, p-value)	(c) CM test (statistic, p-value)
L4	(0.19786, 0.18860)	(41135147.0, 4.70e-59)	(69.07173, 1.59e-08)
L8	(0.17585, 0.00011)	(45901476.0, 5.77e-08)	(55.13462, 9.61e-09)
L32	(0.12281, 5.99e-15)	(53263865.5, 3.68e-20)	(25.79365, 5.12e-09)
L128	(0.13502, 1.58e-14)	(52545248.5, 5.78e-13)	(29.92694, 5.85e-09)
L512	(0.11706, 5.99e-15)	(53094261.5, 2.81e-17)	(22.67795, 6.90e-09)

Table 4.3: Statistical hypothesis tests results per only one StyleALAE configuration (repetition 0) of latent space size in values [4, 8, 32, 18, 512] for the $y\ center$ ellipse feature. Comparison of models with different latent space size by leveraging (a) Two-sample Kolmogorov-Smirnov (KS) test; (b) The Mann-Whitney U rank (MW) test; (c) Two-sample Cramér-von Mises (CM) test.

only difference of KS test p-value of 0.05651 is larger than 0.05, suggesting that the distribution of generated features by L32 is similar to the distribution of original features.

The results for $y\ center$ and $angle$ features comparison from table Table 4.3 and Table 4.2 suggest that statistical tests can not be used to compare the performance of models with different latent space sizes. By definition, the absolute value of the p-value can not be used to compare different models' performances with each other. Hence determining whether the distribution of generated and original features is similar is not enough to quantify the difference between model performances.

4.5 Quantify difference in feature distributions with Jensen-Shannon distance

A divergence metric between the probability distributions has to be chosen in order to compare the probability distributions of the real features computed with the probability distributions of the generated features. For this task, the most suitable metric is the Jensen-Shannon distance since it estimates the similarity between two probability distributions and is especially well-suited for discrete probability distributions. Since it is a distance-based metric, the higher the score, the lower the similarity between two distributions.

The Jensen-Shannon distance JSD is computed as the square root of the *Jensen-Shannon divergence*, which is a smoothed and symmetric version of the Kullback-Leibler divergence $D(P\|Q)$. The formula for Jensen-Shannon distance calculation is defined as

$$JSD(P\|Q) = \sqrt{\frac{D(P\|M)}{2} + \frac{D(Q\|M)}{2}} \tag{4.1}$$

for probability distributions P, Q and where M is the pointwise mean of P, Q: $\frac{1}{2}(P + Q)$.

JSD expects the probability vectors of equal size describing two corresponding distributions as inputs. However, as of now, the ellipse features are represented in plain absolute pixel values, which can be viewed as realizations of a random variable. Therefore it is required first to convert observations of random variables into probability vectors to compute the JSD.

4.5.1 Binning approach to obtain probability vectors of synthetic and original features

The most straightforward method to transform feature values into probability vectors is the binning approach, similar to the histogram representation of the distribution in Figure 4.4. In order to construct a histogram, the interval covered by the data values is divided into equal sub-intervals, known as 'bins'. The number of bins and their size must be identical for both generated and original features to ensure a robust JSD calculation.

Nevertheless, choosing the location and the number of bins of equal size is not optimally determined and can be easily manipulated to change the JSD result. A significant problem with the histogram approach to obtain probability vectors from plain input features is that the choice of binning can have a disproportionate effect and lead to divergent conclusions regarding the underlying shape of the data distribution. In turn, this can lead to representations that have qualitatively different features. For instance, the left-most plot on the figure 4.5 represents a bimodal distribution. However, unimodal distribution with a long tail is illustrated on the right-most plot. Yet, these two histograms were produced by the same data points with the only difference that the right-most plot bins were moved by a small margin, which led to a dramatic change in data representation. Thus, choosing the number of bins impacts how the data is interpreted and density estimated. This example proves that poor choice of bin size for transforming plain input features into probability vectors will negatively affect the JSD estimation of similarity in two distributions.

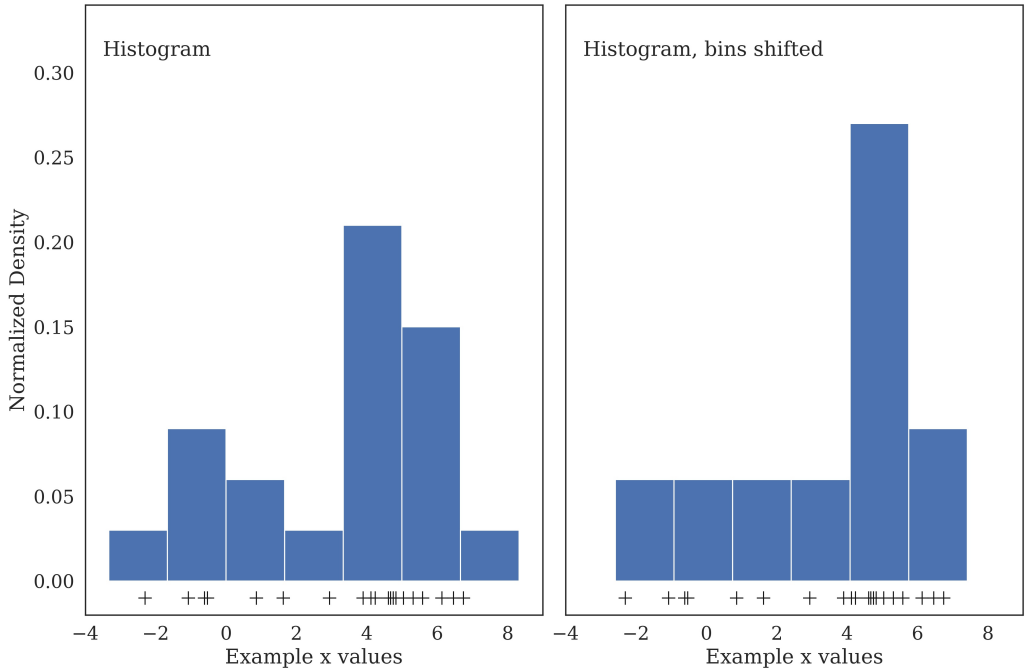

Figure 4.5: The figure displays a histogram over the same data in two plots, with the bins shifted to the right on the right plot. The bins are shifted to the right by a small margin, but the density estimation of the same data changes dramatically. This example demonstrates the binning approach's main limitation in transforming input feature values into probability vectors.

4.5.2 Kernel density estimation approach to obtain probability vectors of synthetic and original features

As seen in Section 4.5.1, histograms have the following shortcomings - they are not smooth data representations, depend on the start and end points of bins, and heavily depend on the size of the bins. Such caveats prevent the histogram approach from being used to create probability vectors. The kernel density estimators can mitigate the first two negative properties of histograms. Kernel density estimation (KDE) is a non-parametric estimator of the probability density function (PDF) of a random variable. The observations of a random variable (ellipse feature values) are fed into the algorithm to build the kernel density estimator. Then the learned KDE from both generated and original features are used to calculate the log-likelihood of test data points. The computed log-likelihoods serve as inputs to JSD to quantify the similarity between original and generated data distributions.

Given N independent realizations $x_i, i = 1..N$ from am unknown continuous probability density function f, the kernel density estimator at a point y is defined as

$$\hat{f}(x;h) = \frac{1}{N}\sum_{i=1}^{N} K(y - x_i; h) \tag{4.2}$$

where $K(x;h)$ is a kernel positive function which is controlled by the bandwidth parameter $h > 0$. The

Gaussian kernel is defined as

$$K(x;h) = \frac{1}{h\sqrt{2\pi}} \exp(-\frac{x^2}{2h^2})$$

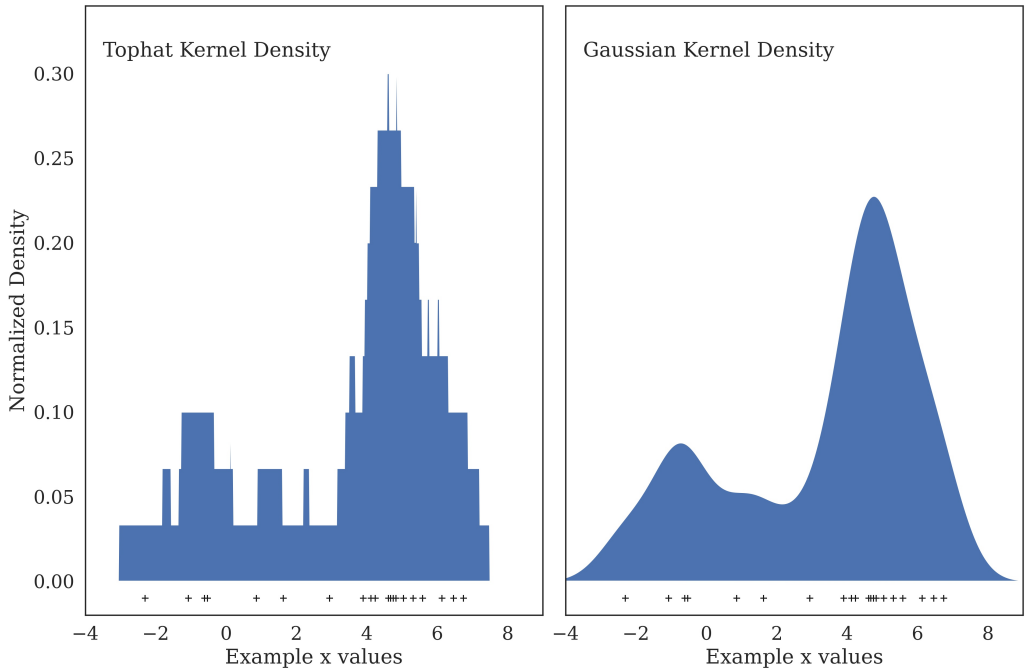

Figure 4.6: The figure displays density estimation results of the same data as in Figure 4.5 but modeled with a kernel density estimation approach with two different choices of the kernel function. This more closely reflects the actual data characteristics than the histogram approach.

There are still two primary hyperparameters of kernel density estimation that must be defined in advance - 1) the kernel function, which determines the shape of the distribution placed at each data point, and 2) the kernel bandwidth, which controls the size of the kernel at each data point. From the Formula 4.2, it can be seen that making the kernel too narrow does not provide much more information than the raw input data while making it too large oversmooths the data; thus, specific characteristics are lost. In this way, bandwidth controls the bias-variance trade-off in the density estimation. There are available methods designed to determine the optimal bandwidth hyperparameter, which is not the case for the histogram approach to get the optimal bin size.

Bandwidth selection methods

The kernel function does not significantly impact the shape of the resulting probability distribution. It is a common practice to use the Gaussian kernel, which has a smooth continuous shape. However, the bandwidth parameter strongly influences the estimate obtained from the KDE. Different bandwidths values might produce very different results for the same input data. Taking into account the importance of bandwidth, optimal parameter values have to be determined reliably. Three main methods described below are used to identify the

Ellipse feature name	(a) Cross-validation	(b) Scott's rule	(c) Diffusion method
x center	0.202	0.091	0.024
y center	0.22	0.092	0.025
major axis	0.179	2.238	0.174
minor axis	0.126	2.172	0.134
angle	0.737	8.271	3.039

Table 4.4: The optimal bandwidth values were obtained using three methods on the original ellipse features.

most suitable value for the bandwidth hyperparameter.

Cross-validation approach. In the machine learning field, hyperparameter tuning is often carried out empirically using a cross-validation strategy. Given a range of potential candidate values for the bandwidth parameter, the kernel density estimator is computed individually. Then the optimal bandwidth parameter is found, which maximizes the log-likelihood score of held-out data samples. Such a standard grid search technique checks different potential parameter values and chooses the best one based on the desired metric. This approach provides flexibility and can be used regardless of the underlying data distribution.

Scott's Rule as a common reference rule from statistics. Another set of approaches stems from statistics which rely on reference rules (called "rules of thumb") in which bandwidth is determined from theoretical forms given certain assumptions regarding the data distribution. According to the most common reference rule - Scott's Rule, the bandwidth is computed as $n^{-1/(d+4)}$, where n is the number of data points and d is the number of dimensions [35].

Diffusion method. In the study by Botev et al. [4], a new bandwidth selection method is proposed, which does not require the data normality assumption (unlike Scott's rule). The new method is nonparametric, as a preliminary normal model for the data is unnecessary. In several simulations, the proposed approach outperformed existing statistical methods in terms of accuracy and reliability.

Optimal bandwidth parameters obtained with three methods. By applying the three methods for optimal bandwidth selection, the following values shown in Table 4.4 were obtained for each ellipse feature extracted from original samples. Depending on the bandwidth value, the density function can take different forms. In Figure 4.7 visualisation for different bandwidth parameter is provided for the $angle$ ellipse feature. This example confirms that the large bandwidth value tends to oversmooth the density function (high bias) while the small hyperparameter overfits the dataset (high variance) and thus captures all the unnecessary details, including noise. For $angle$ ellipse feature, the choice of bandwidth by diffusion method recovers all the important peculiarities and data properties while maintaining smoothness.

4.5.3 Results of feature consistency estimation via Jensen-Shannon distance

The previous subsection explored methods for reliably converting plain feature values into probability vectors. Based on three KDE approaches and one binning approach, the JSD distance was calculated between original ellipse features and generated ellipse features per each latent space size. Since there are six StyleALAE models

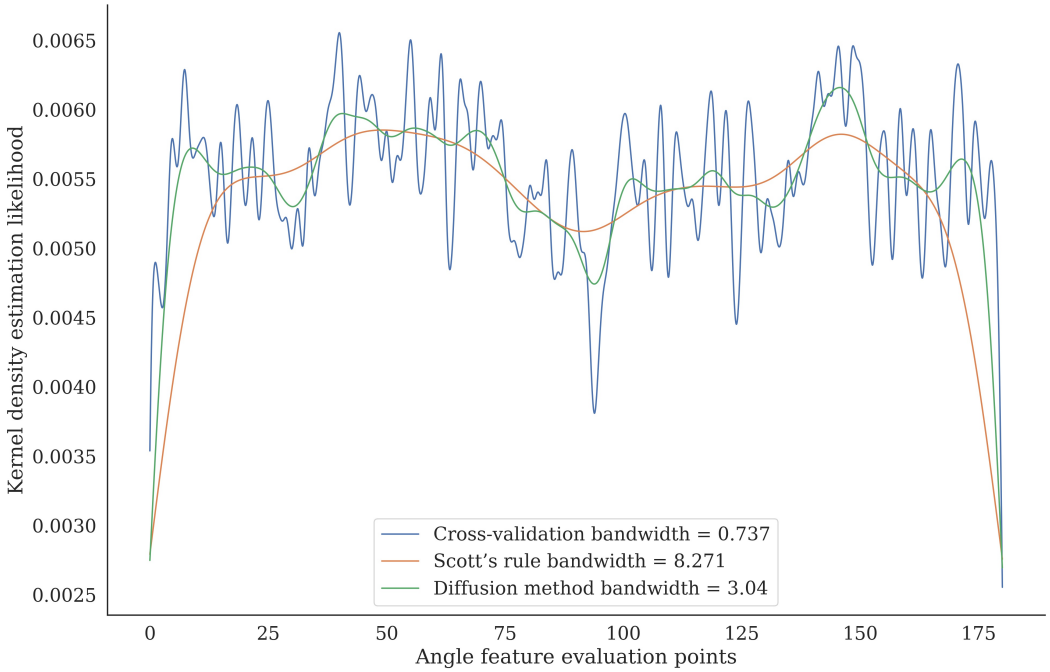

Figure 4.7: Kernel density estimation results for the angle ellipse feature with different bandwidth values obtained with three methods. The figure also illustrates the bandwidth parameter's effect on the density distribution — the bigger the values, the smoother the target density function.

trained per each latent space size, the JSD results were aggregated by calculating the mean and standard deviation. Intuitively, the mean values correspond to reliable similarity estimation, whereas standard deviation is an uncertainty measure. The JSD results obtained from the KDE cross-validation, KDE Scott's rule, KDE diffusion method as well as from histogram with bin size of 1 are presented in Figures 4.13, 4.25, 4.19, 4.31 respectively.

The JSD results for each latent space size are similar, independent of the chosen method to get probability vectors as JSD inputs. For every independently considered feature, the pattern of JSD results for different latent space sizes remains similar across four experiments. It empirically shows that JSD is a robust metric that can produce coherent results despite different approaches to computing probability vectors.

Ellipse feature $x\ center$. In all four experiments, latent space size $L4$ has the largest mean and standard deviation of JSD scores, followed by the mean and standard deviation of latent size $L8$. Visual inspection shows the curve of mean JSD scores and constant standard deviation plateaus for latent size $L32, L128, L512$ with comparable absolute mean scores values.

Ellipse feature $y\ center$. Latent space sizes $L4$ and $L8$ achieved the worst performance compared to other latent sizes since their means are larger than the rest. The JSD scores pattern for the $y\ center$ feature is similar to the $x\ center$ with only difference in JSD mean score jump for latent space size $L128$. The JSD mean values for $L32$ and $L512$ are within a small range, while the mean JSD scores of $L128$ is noticeably larger.

Ellipse feature $major\ axis$. Through qualitative observation of the JSD mean scores trend, the local minimum JSD mean is achieved by $L32$. The $L4$ and $L8$ mean scores are larger than the $L32$ mean score, but after passing the local minimum at latent $L32$, the mean scores are getting larger for $L128$ and $L512$.

Ellipse feature $minor\ axis$. In contrast to previous feature results, the JSD mean scores keep decreasing with increasing latent space size, as evidenced by the decrease in JSD means absolute values. The minimum JSD mean score is achieved by $L512$ among other latent space sizes.

Ellipse feature $angle$. As observed from the presented plots, latent sizes $L4$ and $L8$ captured the distribution of the $angle$ feature from the original dataset not as good as other latent sizes $L32, L128, L512$. The absolute mean values of $L4$ and $L8$ are considerably higher than in $L32, L128, L512$. After $L32$ latent size, the curve is (almost) constant indicating no performance difference between the last three latent space sizes.

Even though the plots provide results and insights into the model performance with different latent space values, it is still not possible to have a quantitative estimation of the best-performing model. In results analysis, formulations like "similar", "considerable", and "(significantly) larger" were used to compare JSD results, but there is no confidence in insights and conclusions achieved. Hence the open question remains: What is the optimal latent space size for the StyleALAE architecture to learn the features consistently?

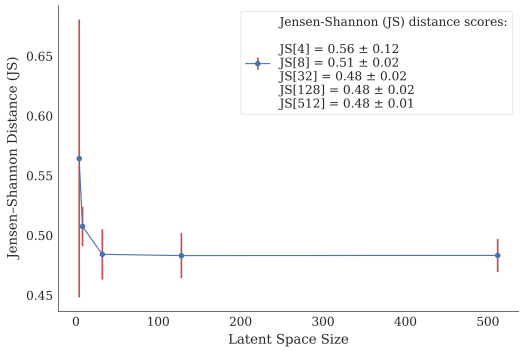

Figure 4.8: (a) Ellipse feature - x center

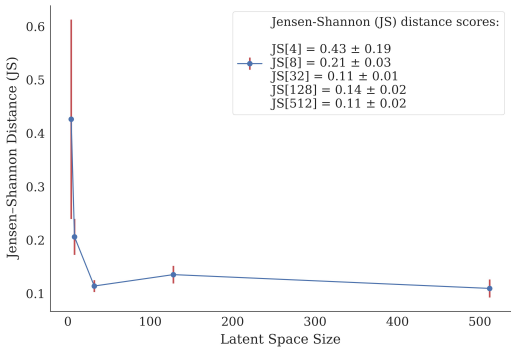

Figure 4.9: (b) Ellipse feature - y center

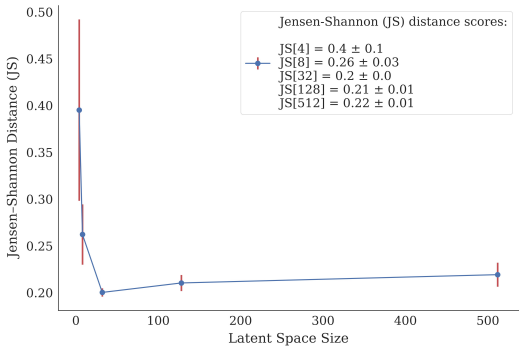

Figure 4.10: (c) Ellipse feature - major axis

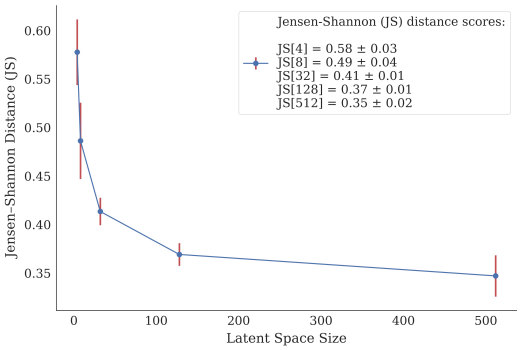

Figure 4.11: (d) Ellipse feature - minor axis

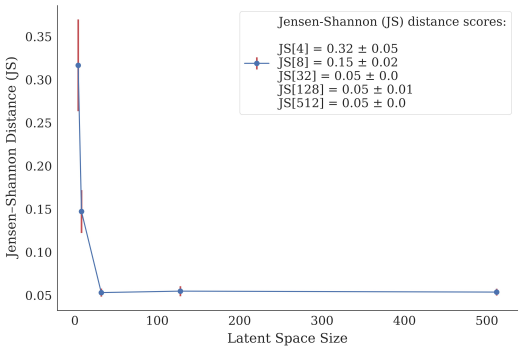

Figure 4.12: (e) Ellipse feature - angle

Figure 4.13: The average Jensen-Shannon distance is calculated with the kernel density estimation (**cross-validation**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. In total, six different models were trained per each latent space size from 4, 8, 32, 128, and 512. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

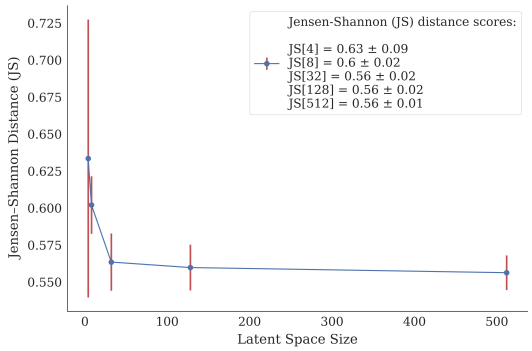

Figure 4.14: (a) Ellipse feature - x center

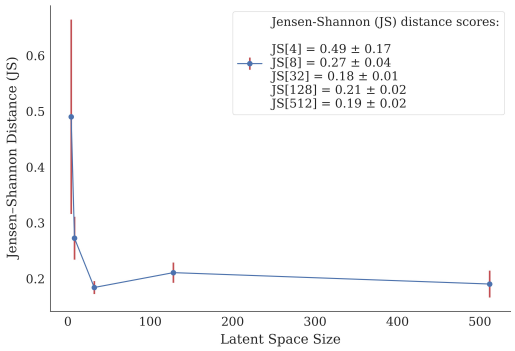

Figure 4.15: (b) Ellipse feature - y center

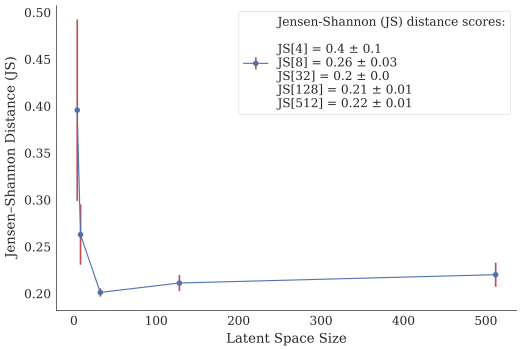

Figure 4.16: (c) Ellipse feature - major axis

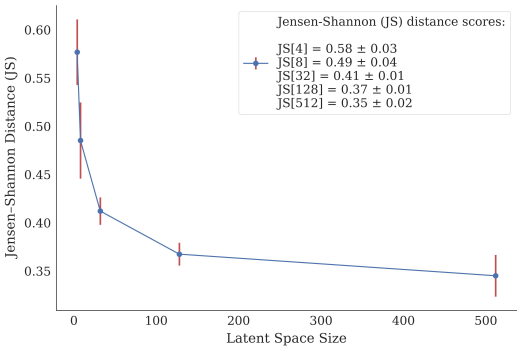

Figure 4.17: (d) Ellipse feature - minor axis

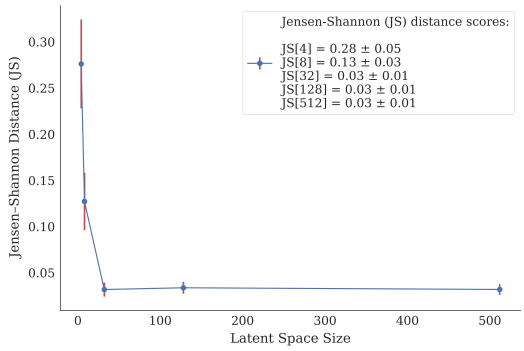

Figure 4.18: (e) Ellipse feature - angle

Figure 4.19: The average Jensen-Shannon distance is calculated with the kernel density estimation (**diffusion**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. In total, six different models were trained per each latent space size from 4, 8, 32, 128, and 512. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

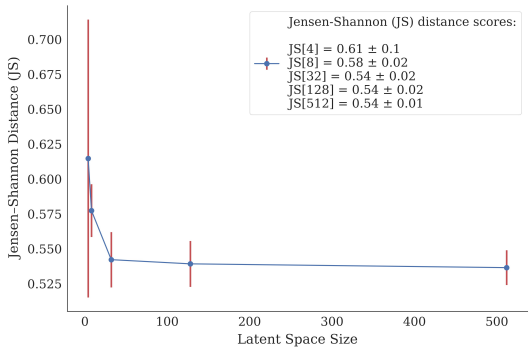

Figure 4.20: (a) Ellipse feature - x center

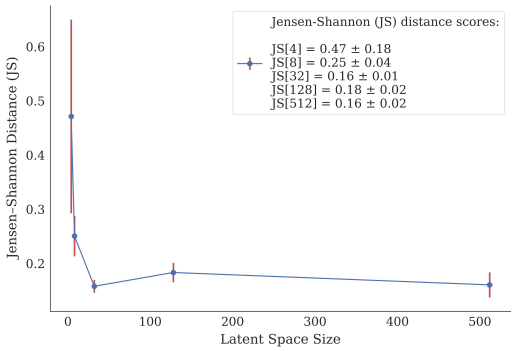

Figure 4.21: (b) Ellipse feature - y center

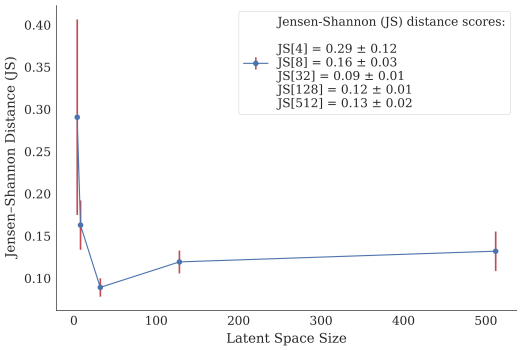

Figure 4.22: (c) Ellipse feature - major axis

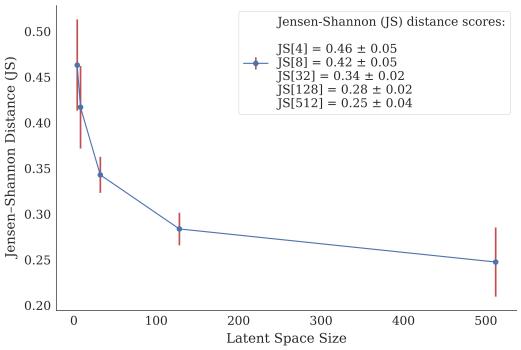

Figure 4.23: (d) Ellipse feature - minor axis

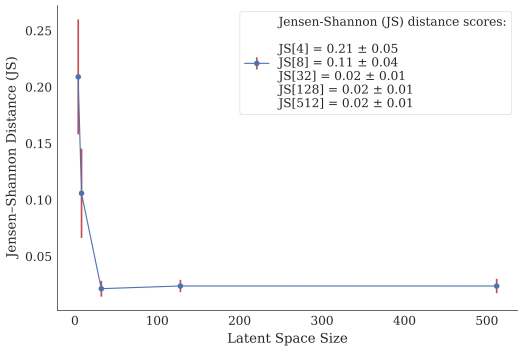

Figure 4.24: (e) Ellipse feature - angle

Figure 4.25: The average Jensen-Shannon distance is calculated with kernel density estimation (**Scott's rule**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. In total, six different models were trained per each latent space size from 4, 8, 32, 128, and 512. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

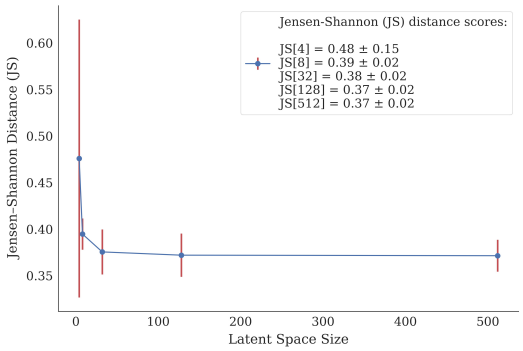

Figure 4.26: (a) Ellipse feature - x center

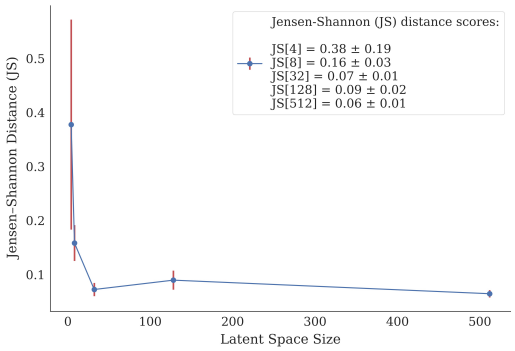

Figure 4.27: (b) Ellipse feature - y center

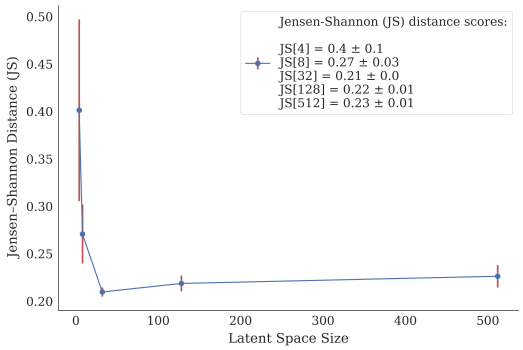

Figure 4.28: (c) Ellipse feature - major axis

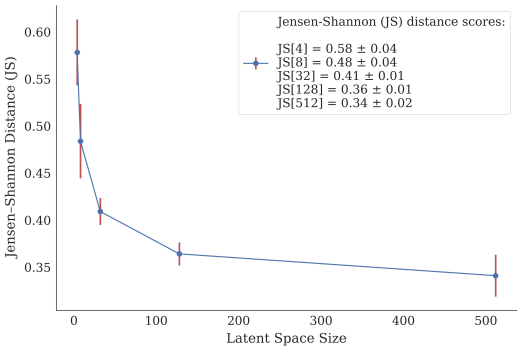

Figure 4.29: (d) Ellipse feature - minor axis

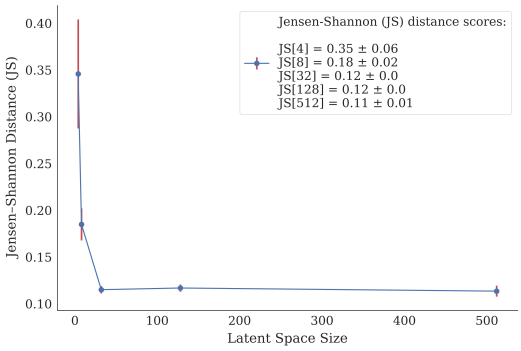

Figure 4.30: (e) Ellipse feature - angle

Figure 4.31: The average Jensen-Shannon distance is calculated with the histogram approach with (**bin step=1**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. In total, six different models were trained per each latent space size from 4, 8, 32, 128, and 512. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

4.6 Quantify difference in feature distributions with adapted Frechet inception distance

The concept of the original FID metric is described in Section 2.2.2 and calculation steps are explained. It is a widely adapted metric, yet it is claimed to be inapplicable to datasets other than ImageNet covering natural images. Since the Inception-v3 model was trained on the ImageNet dataset (real-world images), the model feature representation for toy ellipse dataset images might be useless as these images are not present in ImageNet. Hence the model is not able to get a reliable feature embedding for the input grayscale image of an ellipse.

To keep the advantages of the FID metric and mitigate the above-mentioned shortcoming, the feature extraction step of the FID metric is replaced by ellipse features extracted from original ellipse samples and generated samples in Section 4.8. The rest of the steps are the same as in the original FID metric. In this way, the FID score is adjusted to the particular dataset, and the metric is called the "Adapted FID metric".

The better the extracted features, the more reliable and robust the evaluation method is. Hence, feature extraction is crucial in differentiating between samples from the original and synthetic datasets. Other alternatives could be the following:

- Features of Inception-v3 model from earlier layers. These features may represent low-level characteristics of the image, which might be a useful descriptor of an image. Such low-level descriptors are common to any image; thus, these features are independent of the trained dataset and do not overfit it.
- Stacked combination of known features. For example, for the toy ellipse dataset - features like $major\ and\ minor\ axis$, $angle$ and center location - $x\ center, y\ center$.
- Last layer activation of a trained neural network classifier for the dataset used in generative modeling. By building and training a model for the classification task, useful abstract features of images can be learned. Similar to the Inception-v3 model, the layer before the classification layer will be used for getting feature embeddings from the input image.

4.6.1 Transparent feature extraction for robust Frechet inception distance calculation

In the Deep learning field, earlier layers of a neural network are claimed to learn simple features of the image, e.g., vertical edges. However, using these features might not be beneficial since embedding vectors might be similar, and it would be challenging to differentiate between different vectors by using the Frechet distance. Additionally, training an auxiliary classifier requires a labeled dataset, which is not available for the toy ellipse dataset. Thus the stacked combination of known features such as $major\ and\ minor\ axis$, $angle$, center location - $x\ center, y\ center$ is used as the approach for feature extraction.

Consequently, the feature extraction step is replaced by the approach outlined in Section 4.8. Since the Frechet distance is calculated by considering the mean vectors and covariance matrices of original features and generated,

it is essential to perform values normalization in feature embeddings to avoid favoring a specific feature and consider all features equally. The Adapted FID score is calculated by keeping the remaining steps of the original FID metric. The expected outcome is a single score representing how similar generated images are to real images. The resulting method can compare different GAN models and different StyleALAE configurations.

4.6.2 Results of similarity estimation via the adapted Frechet inception distance calculation

Unlike the JSD-based similarity estimation method, which examines every feature individually, the adapted Frechet inception distance aggregates all features. Therefore, only one plot demonstrates the performance of StyleALAE models with different latent space sizes is displayed in Figure 4.32.

From the qualitative analysis of the produced plot and the calculated mean and standard deviation of the adjusted FID score per each latent space size, one can observe scores behavior similar to JSD-based similarity estimation. The latent space sizes $L4$ and $L8$ have the worst performance, with mean scores equal to 6.62 and 4.34, respectively. There is a rapid decrease in the JSD mean scores after $L8$, as the $L32$ JSD mean score becomes 1.89. The curve flattens after latent space size $L128$ with the mean score being the same as the $L512$ mean score. However, it is not feasible yet to examine whether the mean score of $L32$ is "similar" to $L128$ and $L512$ mean scores or significantly larger.

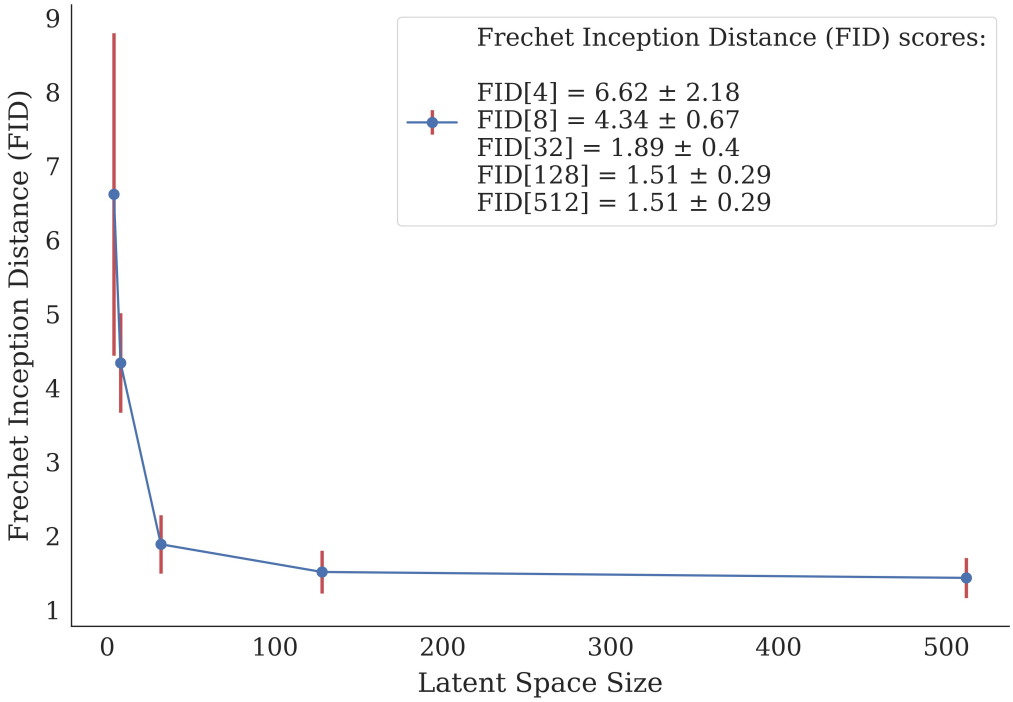

Figure 4.32: The average adjusted Frechet inception distance is calculated between the original feature distribution and synthetic feature distribution obtained from StyleALAE models with varying latent space sizes. In total, six different models were trained per each latent space size from 4, 8, 32, 128, and 512. The average values of the adjusted Frechet inception distance are annotated in the legend, including the standard deviation.

In the next section, an evaluation schema is designed and applied to the adjusted FID and JSD metric results. The aim is to determine whether latent space size performances in both metrics are identical or significantly different. It would be the last step in the evaluation framework to identify the optimal latent space size with some confidence.

4.7 Evaluation schema development to quantitatively compare mean and standard deviation of metric scores

In the previous two sections, a difference in feature distributions is quantified with the Jensen-Shannon distance and adapted FID metric. The JS distance was calculated per each latent space size, and the results per each feature were computed. The same trend in results is obtained with the adjusted FID metric approach, but only one plot aggregates each feature's contributions.

Yet there is still no clear answer to the question of which latent space size performs the best. Which model configuration with a latent space size could consistently learn the original feature distribution and outperform other model configurations with different latent space sizes? There is a clear need to design the evaluation schema to identify what value for latent space size is optional in the experimental setup with the ellipse dataset.

Since both approaches to similarity estimation between two distributions lead to the same output format results, the target evaluation schema is unified and can be applied to both cases. As an input, the evaluation schema expects the calculated distance result per each latent space size for each model repetition. The steps in the evaluation schema are demonstrated in Figure 4.33.

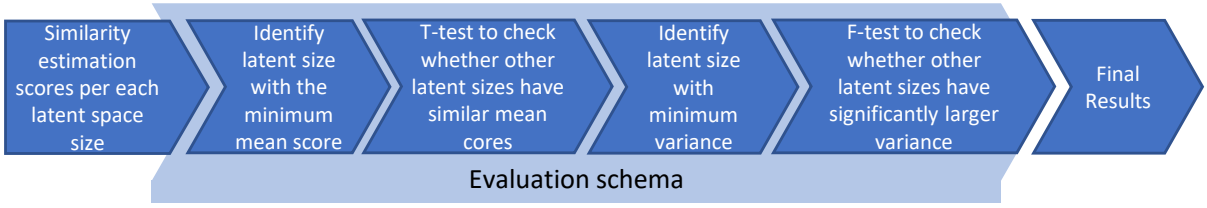

Figure 4.33: Diagram presents calculation steps in the evaluation schema.

Quantify the difference between the arithmetic means of the two sample sets.

Both JS and FID approaches are based on distance calculation. Intuitively, the smaller the distance, the smaller the difference between two distributions, and thus the better latent space size performance. Therefore the mean of scores is the primary goal to compare and verify how different the means per each latent space size are. It is tempting to identify the latent size with the minimum mean score and claim it to be the final result, but other latent space sizes might have mean values "comparable" to the minimum mean score. Hence there is a need to determine whether the means of other latent space sizes are significantly higher than the detected minimum.

The statistical T-test is leveraged to compare the means of two independent samples of scores. The null hypothesis is that two independent sets of values have the same (expected) average values. In contrast, the alternative hypothesis is left-tailed, meaning the mean of the distribution underlying the first set of samples is less than the mean of the distribution underlying the second set of samples. After the t-statistic is computed, the p-value is compared to the pre-defined significance level α of 5% since the p-value denotes the probability of observing such a t-statistic value, assuming the null hypothesis is true. Generally speaking, the p-value being larger than α indicates that our observation is not so unlikely to have occurred by chance. Thus, the null hypothesis of equal population means can not be rejected. If the p-value is smaller than α, then there is evidence for the alternative hypothesis rejecting the null hypothesis of equal population means.

By employing the T-test and comparing the mean score of other latent space sizes to the detected latent size with the minimum mean score, other latent space sizes are statistically proven to achieve good performance in learning features.

Quantify the difference between the variances of the two sample sets. However, comparing the means of scores per latent space size is insufficient. Observations with a large variance can easily pass the T-Test. This is the undesired behavior because certain features have to be learned by a model consistently. The illustrative scenario is displayed in Figure 4.34, where model 1 learned the feature A distribution consistently. Feature A distributions produced by model 1 are close to each other, and the aggregated mean is close to the mean of the original distribution. While model 2 has aggregated mean close to the mean of the original distribution as well, the distributions themselves are spread from each other, indicating the failure of model 2 to learn feature A consistently. The "big" variance in scores results can be considered as one of the factors signaling the instability in learning and hence failure in training.

The evaluation schema must be reliable and robust against such scenarios to avoid giving any preferences to latent space results with a huge variance. A huge variance is described as a variance that is significantly larger than variances of other latent space sizes. Therefore the variance of the scores per each latent space size has to be validated similarly to the means of the scores. Variance check is applied only to latent sizes which passed the T-test (mean scores are minimum).

To address the variance check, F-test is utilized to justify the null hypothesis that two population variances are equal. The alternative hypothesis is that the variance of one of the samples is larger than the variance of another set of samples. The latent space size with a minimum variance has to be detected, and the variances of other latent space sizes have to be checked pairwise with the minimum variance latent space size. The F-statistic and p-value are computed for each pair, and the p-value is compared to the pre-defined significance level. If the p-value is less than α, the null hypothesis can be rejected, meaning there is sufficient evidence to claim that the two population variances are not equal.

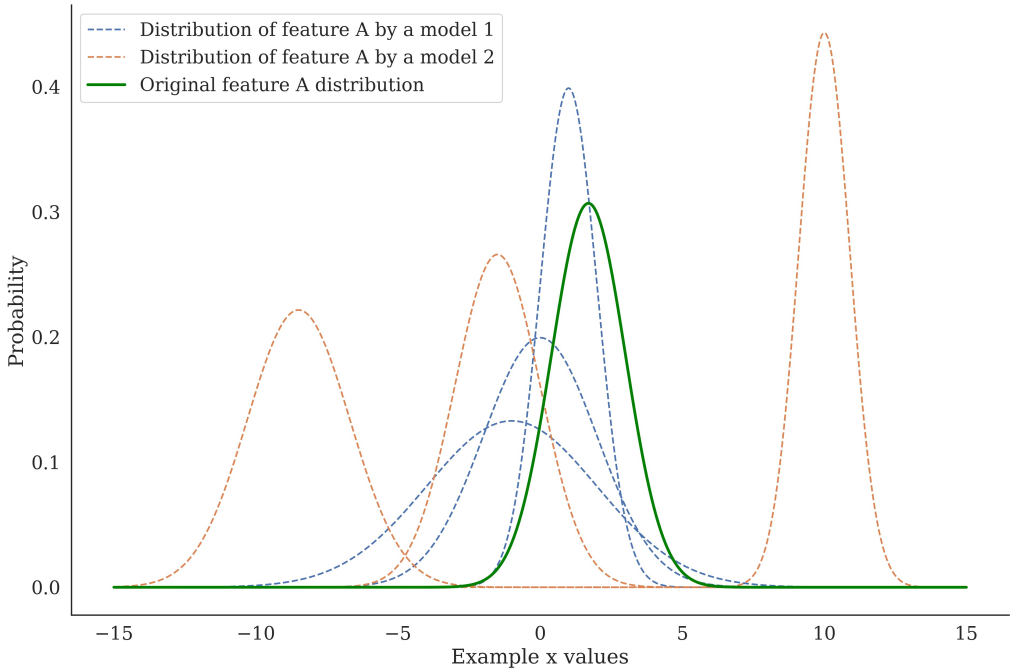

Figure 4.34: The figure displays an illustrative example of original feature distribution and feature distributions learned by two different models with three repetitions. The mean of generated features for model 1 and model 2 is similar. However, the feature distributions of model 1 are "grouped together" (small variance), whereas feature distributions of model 2 are spread from each other (big variance). This example illustrated that model 2 was not able to learn feature A consistently, while model 1 did.

4.8 Evaluation framework results

The aggregated distance-based scores per each latent space size, as well as the outcomes of every step in the evaluation schema are presented in Tables 4.5, 5.6, 5.4, 4.8. The first column indicates the mean and variance values calculated from JSD or adjusted FID scores of six StyleALAE models trained per each latent size. Then pairwise T-Test is performed to verify if the means of other latent sizes are statistically equal to the minimum mean score. If a model parameterized by latent size L passes the T-Test, it is proved to outperform other models that failed the test. Thus it leaned a particular original feature distribution better than others. Then follows the variance test to detect models with vast variance in distance-based scores. This is the last filtering step to avoid considering models with failed training. The advantage of applying evaluation schema to the JSD results is that each feature is considered independently. Thus certain weights can be assigned to features to indicate the importance of the feature in the evaluation procedure. For simplicity, no weights are assigned in this experiment, meaning each feature is treated equally.

Adjusted FID mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Final latent space size (L) scores
L4: 6.617±2.387	Failed	-	0
L8: 4.339±0.738	Failed	-	0
L32: 1.887±0.433	Failed	-	0
L128: 1.513±0.317	Passed	Passed	1
L512: 1.433±0.298	min Mean	min Variance	1

Table 4.5: Evaluation schema outcomes applied to adjusted FID metric scores calculated from ellipse features. The outcome suggests that either L128 or L512 are the best hyperparameter values and have to be chosen as optimal latent space sizes.

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Angle ellipse feature			
L4: 0.276±0.053	Failed	-	0
L8: 0.127±0.034	Failed	-	0
L32: 0.032±0.008	min Mean	Passed	1
L128: 0.034±0.007	Passed	Passed	1
L512: 0.032±0.006	Passed	min Variance	1
Minor axis ellipse feature			
L4: 0.577±0.037	Failed	-	0
L8: 0.485±0.043	Failed	-	0
L32: 0.412±0.016	Failed	-	0
L128: 0.368±0.013	Failed	-	0
L512: 0.345±0.024	min Mean	min Variance	1
Major axis ellipse feature			
L4: 0.396±0.106	Failed	-	0
L8: 0.263±0.035	Failed	-	0
L32: 0.201±0.005	min Mean	min Variance	1
L128: 0.211±0.009	Failed	-	0
L512: 0.22±0.014	Failed	-	0
X center location ellipse feature			
L4: 0.633±0.103	Passed	Failed	0
L8: 0.602±0.021	Failed	-	0
L32: 0.563±0.021	Passed	Passed	1
L128: 0.56±0.017	Passed	Passed	1
L512: 0.556±0.013	min Mean	min Variance	1
Y center location ellipse feature			
L4: 0.49±0.191	Failed	-	0
L8: 0.273±0.042	Failed	-	0
L32: 0.184±0.013	min Mean	min Variance	1
L128: 0.211±0.02	Failed	-	0
L512: 0.19±0.026	Passed	Passed	1
Final latent space size (L) scores: L4: 0, L8: 0, **L32: 4**, L128: 2, **L512: 4**			

Table 4.6: Evaluation schema applied to results of JSD metric scores calculated from ellipse features with KDE technique (diffusion method used to estimate bandwidth parameter). The outcome suggests that either L32 or L512 are the most optimal hyperparameters values.

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Angle ellipse feature			
L4: 0.317±0.058	Failed	-	0
L8: 0.147±0.027	Failed	-	0
L32: 0.053±0.005	min Mean	Passed	1
L128: 0.055±0.006	Passed	Passed	1
L512: 0.054±0.004	Passed	min Variance	1
Minor axis ellipse feature			
L4: 0.578±0.037	Failed	-	0
L8: 0.486±0.043	Failed	-	0
L32: 0.414±0.015	Failed	-	0
L128: 0.369±0.013	Failed	-	0
L512: 0.347±0.023	min Mean	min Variance	1
Major axis ellipse feature			
L4: 0.395±0.106	Failed	-	0
L8: 0.262±0.035	Failed	-	0
L32: 0.2±0.005	min Mean	min Variance	1
L128: 0.21±0.009	Failed	-	0
L512: 0.219±0.014	Failed	-	0
X center location ellipse feature			
L4: 0.564±0.127	Passed	Failed	0
L8: 0.508±0.018	Failed	-	0
L32: 0.484±0.023	Passed	Passed	1
L128: 0.483±0.021	min Mean	Passed	1
L512: 0.483±0.015	Passed	min Variance	1
Y center location ellipse feature			
L4: 0.426±0.204	Failed	-	0
L8: 0.206±0.037	Failed	-	0
L32: 0.114±0.012	Passed	min Variance	1
L128: 0.135±0.018	Failed	-	0
L512: 0.109±0.019	min Mean	Passed	1
Final latent space size (L) scores: L4: 0, L8: 0, **L32: 4**, L128: 2, **L512: 4**			

Table 4.7: Evaluation schema applied to results of JSD metric scores calculated from ellipse features with KDE technique (cross-validation method used to estimate bandwidth parameter). The outcome suggests that either L32 or L512 are the most optimal hyperparameters values.

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Angle ellipse feature			
L4: 0.209±0.056	Failed	-	0
L8: 0.106±0.043	Failed	-	0
L32: 0.021±0.008	min Mean	Passed	1
L128: 0.024±0.006	Passed	min Variance	1
L512: 0.024±0.007	Passed	Passed	1
Minor axis ellipse feature			
L4: 0.463±0.055	Failed	-	0
L8: 0.417±0.05	Failed	-	0
L32: 0.343±0.022	Failed	-	0
L128: 0.284±0.02	Failed	-	0
L512: 0.248±0.041	min Mean	min Variance	1
Major axis ellipse feature			
L4: 0.291±0.127	Failed	-	0
L8: 0.163±0.032	Failed	-	0
L32: 0.089±0.012	min Mean	min Variance	1
L128: 0.119±0.015	Failed	-	0
L512: 0.132±0.026	Failed	-	0
X center location ellipse feature			
L4: 0.615±0.109	Passed	Failed	0
L8: 0.577±0.021	Failed	-	0
L32: 0.542±0.022	Passed	Passed	1
L128: 0.539±0.018	Passed	Passed	1
L512: 0.537±0.014	min Mean	min Variance	1
Y center location ellipse feature			
L4: 0.471±0.196	Failed	-	0
L8: 0.251±0.041	Failed	-	0
L32: 0.158±0.013	min Mean	min Variance	1
L128: 0.183±0.02	Failed	-	0
L512: 0.16±0.025	Passed	Passed	1
Final latent space size (L) scores: L4: 0, L8: 0, **L32: 4**, L128: 2, **L512: 4**			

Table 4.8: Evaluation schema applied to results of JSD metric scores calculated from ellipse features with KDE technique (Scott's rule used to estimate bandwidth parameter). The outcome suggests that either L32 or L512 are the most optimal hyperparameters values.

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Angle ellipse feature			
L4: 0.346±0.064	Failed	-	0
L8: 0.185±0.019	Failed	-	0
L32: 0.115±0.004	Passed	Passed	1
L128: 0.117±0.004	Passed	min Variance	1
L512: 0.114±0.006	min Mean	Passed	1
Minor axis ellipse feature			
L4: 0.578±0.038	Failed	-	0
L8: 0.484±0.043	Failed	-	0
L32: 0.409±0.016	Failed	-	0
L128: 0.364±0.013	Failed	-	0
L512: 0.341±0.024	min Mean	min Variance	1
Major axis ellipse feature			
L4: 0.401±0.105	Failed	-	0
L8: 0.271±0.034	Failed	-	0
L32: 0.21±0.005	min Mean	min Variance	1
L128: 0.219±0.009	Failed	-	0
L512: 0.226±0.013	Failed	-	0
X center location ellipse feature			
L4: 0.476±0.164	Passed	Failed	0
L8: 0.395±0.018	Failed	-	0
L32: 0.376±0.027	Passed	Passed	1
L128: 0.372±0.025	Passed	Passed	1
L512: 0.371±0.019	min Mean	min Variance	1
Y center location ellipse feature			
L4: 0.378±0.213	Failed	-	0
L8: 0.158±0.036	Failed	-	0
L32: 0.072±0.013	Passed	Passed	1
L128: 0.09±0.019	Failed	-	0
L512: 0.064±0.008	min Mean	min Variance	1
Final latent space size (L) scores: L4: 0, L8: 0, **L32**: 4, L128: 2, **L512: 4**			

Table 4.9: Evaluation schema outcomes after applying to JSD metric results calculated from ellipse features with binning method. The outcome suggests that either L32 or L512 are the most optimal hyperparameters values.

4.9 Conclusions

In this chapter, a new evaluation framework is designed to estimate the capabilities of generative models in learning the original dataset features consistently. The primary motivation for the new approach stems mainly from the requirement in industrial machine vision real-world applications. In order to perform quality assurance (e.g., defect detection) or metrologically interpretable feature extraction, the deep generative models have to capture the original features distribution instead of just learning how to generate new samples.

The assessment procedure is applied to the artificially created ellipse dataset with known features to test its applicability robustly and whether meaningful insights can be observed. Hence, six StyleALAE models per each latent space size in 4, 8, 32, 128, 512 were trained on the ellipse dataset to avoid reliance on single model performance comparison. Then the feature extraction step is performed on original and generated images to transform input images into feature space. Afterward, different strategies of similarity estimation were examined to measure the difference between original and generated feature distributions, summarized in Figure 4.35. As the last step, the evaluation schema is applied to similarity estimation results to determine latent space size with the best performance and avoid models with latent sizes, which result in a high variance in final scores. The final results of JSD and adapted FID similarity estimation yield the optimal latent space size of 32 and 128, respectively.

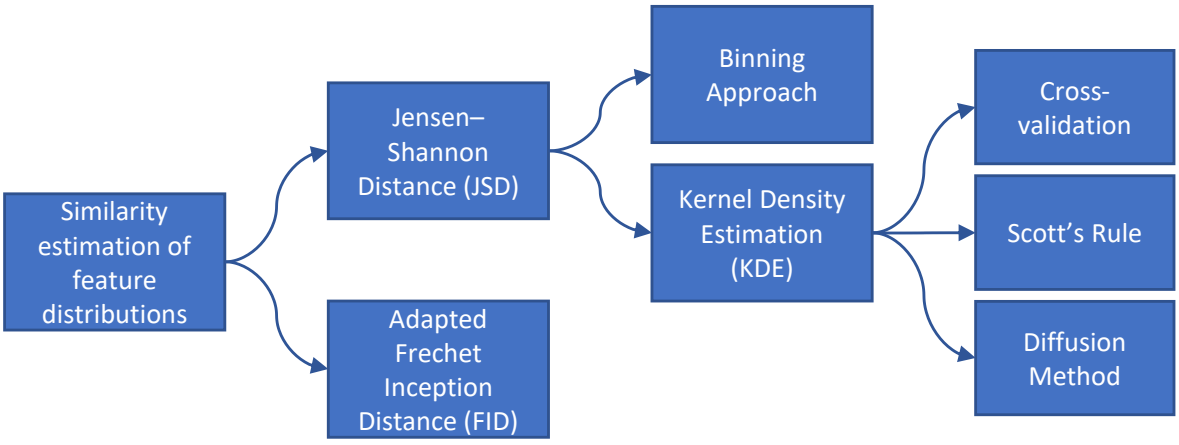

Figure 4.35: Hierarchy of similarity estimation calculation between original and generated feature distributions employed in the evaluation framework.

According to the evaluation schema results on the ellipse dataset, JSD has identical final results independent of the method used to calculate the metric, which points towards metric robustness. Nevertheless, the KDE approach is more reliable than the histogram approach due to the binning shortcomings outlined in Section 4.5.1. To select the optimal bandwidth hyperparameter for KDE, the diffusion method has yielded a robust bandwidth estimator approximating the target model density distribution more accurately than other statistical methods [4]. Last but not least, the combination of bandwidth selection methods can be leveraged instead of a single method. For example, a range of bandwidth candidates is determined with the help of diffusion and Scott's

rule methods. Then a cross-validation method can be further utilized to test different values in the specified range to potentially identify even a better bandwidth hyperparameter.

One of the main limitations of the original FID metric - reliance on features extracted from the Inception-v3 model is mitigated by utilizing known ellipse features. However, an assumption that a set of features representing given images follow multivariate normal distribution prohibits from considering all different aspects of distributions by focusing only on the main statistics like mean and covariance.

The similarity estimation based on JSD is advantageous since the image features are considered separately in the evaluation pipeline. Hence, it provides a tool to investigate the model behavior on the most prominent features for the application-specific tasks and identify potential weaknesses and limitations of a model. Since both approaches of similarity estimation between feature distributions are computationally efficient, it is fast to use the ensemble of metrics to determine the optimal latent space size.

In the next chapter, the developed evaluation framework is applied to the real-world use case by operating on textures dataset from the industrial setting.

5 Application of the developed evaluation framework on real-world industrial machine vision task

In the previous chapter, an evaluation framework consisting of multiple stages was developed and tested on the synthetic ellipse dataset. It appears to reliably differentiate the models' performance of varying latent space sizes and verify the feature consistency of generated samples versus original samples. In this chapter, the same stages of the evaluation procedure are applied to a real-world dataset from an industrial setting to validate the procedure.

One of the most crucial tasks in quality assurance at manufacturing is defect detection which identifies whether the current product passes the quality assessment or has certain defects and flaws. In particular, consider the aluminum die casting surfaces, which inherently can be viewed as texture. Most of them pass the quality criteria, but some certain exhibit defects. As demonstrated in [34], generative modeling architectures such as StyleALAE can be leveraged to interpret the extracted features, which is needed to perform the downstream classification task for defect detection. Since latent space acts as a compressed representation of relevant properties of the input image, it is still ambiguous what the optimal dimensionality for this space should be to encode the necessary information effectively. What lacking is the mechanism to verify whether the human-interpretable features were learned consistently from the original dataset by a model, identify the optimal latent space size to represent the input image efficiently, and differentiate between the models' performance with different latent space sizes. Therefore the developed evaluation framework in chapter 2 is applied to this real-world scenario to clear the uncertainty.

5.1 Textures dataset

The industrial dataset in the following experiments consists of 1,000 images of aluminum die casting components. Images show a predefined region of the surface of manufactured components. All samples display the same component variant and were acquired at the production line by the same camera. The resolution is 256×256 pixels. Some examples are illustrated in Figure 5.1. Even though in the previous section it was argued that at least 10.000 original samples are needed to have a robust comparison with generated images, in a real-world application, it is barely the case. Getting a clean industrial dataset is a complicated and labor-intensive process. In order to approximate real-life situations, the dataset of a 1.000 sample size is used.

The texture surfaces are different in terms of visual appearance and possess such salient features as coarseness, degree of illumination, roughness, presence of specific patterns and/or defects (i.e., whether there is motion blur). By analyzing the texture features, insights about the mold that produces the components can be collected since these characteristics depend on its state.

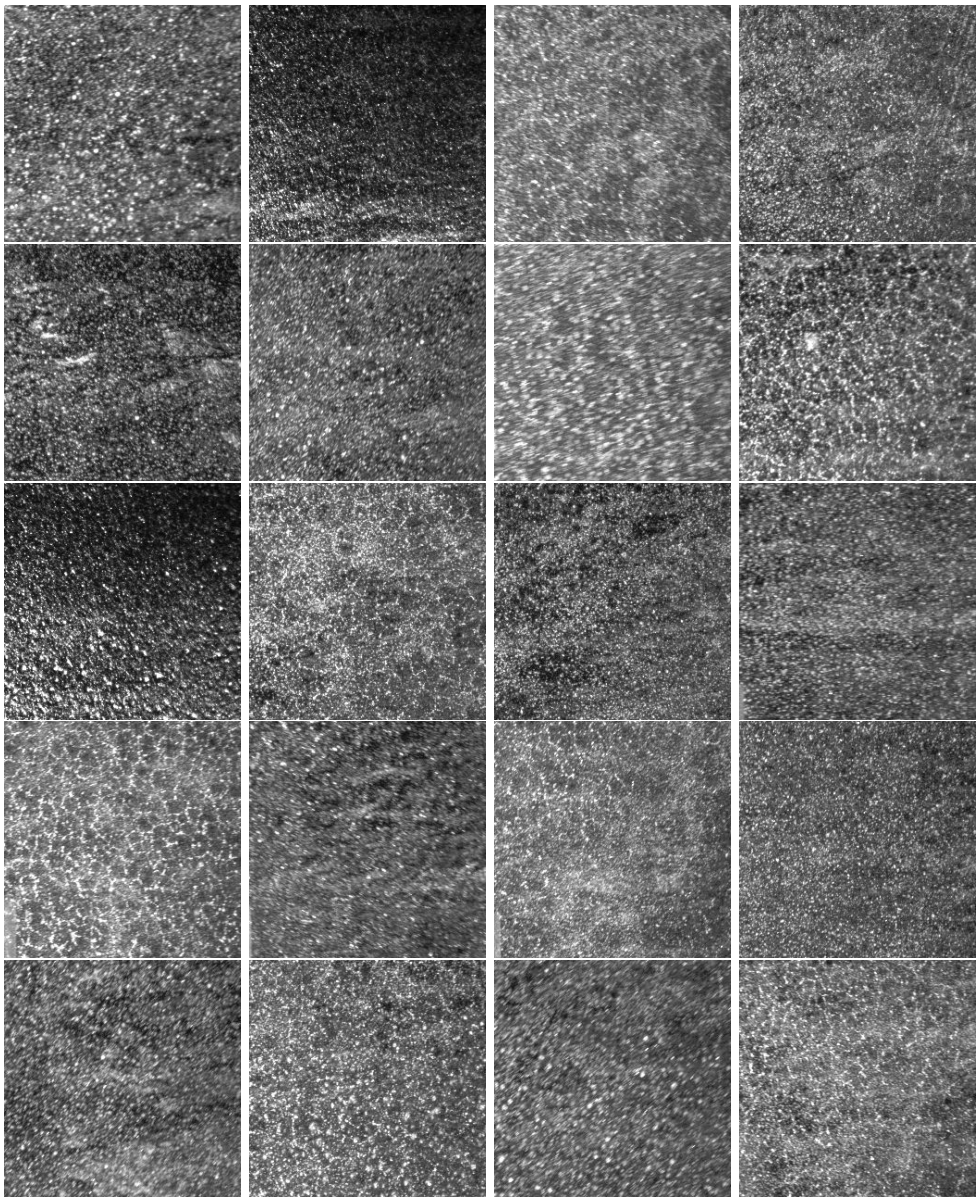

Figure 5.1: Examples of real textures from the training dataset.

5.2 Trained StyleALAE models

As described in Chapter 3, the following StyleALAE models were trained on the textures dataset introduced in Section 5.1.

Latent space size	L8	L16	L32	L64
Number of repetitions	6	6	6	6

Table 5.1: Summary of trained StyleALAE models on texture dataset.

5.3 Haralick features as texture representations

Among the plethora of available texture feature extraction methods proposed in literature [17], in this study, the statistical approach introduced by Haralick et al. is utilized [13]. The method produces texture descriptors by computing statistical properties of the spatial distribution of gray values from an input texture image. The target feature representations are computed from a Gray Level Co-occurrence Matrix (GLCM). This matrix determines a relative frequency $P(i, j|d, \theta)$ for two gray-level values i and j by calculating the number of times the gray-level j appears at a distance d and an angle θ from gray-level i. Distances d and angles θ define the offset over an image and are the parameters needed to compute GLCM. The standard practice is to set distance value d to $1, 2, 3, 4$ values and angle θ to $0, \pi/2$ values.

Once the matrices have been created, certain statistical measures can be extracted from this matrix, as covered in [13]. According to the [5], the main six measures out of the original fourteen are adequate to describe a texture, providing necessary characteristics. These are energy (measures the sum of squared elements in the GLCM), homogeneity (determines the closeness of the distribution of elements in the GLCM to the GLCM diagonal), contrast (estimates the local variations in the gray-level co-occurrence matrix), dissimilarity (assesses the image's range of gray levels), entropy (measures the disorder degree in the image), correlation (determines the joint probability occurrence of the specified pixel pairs).

The Haralick feature extraction method provides a comprehensive set of features describing the texture surface, which is necessary to perform texture comparisons in the evaluation case. Besides being good texture descriptors, Haralick features are easy to compute and result in interpretable texture descriptors, making them a suitable feature representation of a texture.

5.4 Application of evaluation procedure based on Jensen-Shannon Distance

One way to estimate the similarity between the feature distributions is relying on the JSD. The distance-based metric is computed with two approaches: binning (bin step size equals 1) and kernel density estimators obtained from three different bandwidth hyperparameters estimated with cross-validation, Scott's rule and diffusion methods.

The advantage of the JS-based evaluation procedure is that each sample feature is considered separately, which enables tracking the similarity of generated samples versus original samples and detecting specific sample characteristics contributing to the low similarity measure. Since Haralick features are used as texture descriptors, the feature representation of a texture is of length 48 (6 statistics $\times$ 4 distance values $\times$ 2 angle values). However, leveraging all the 48 features will make the evaluation procedure cumbersome and challenging to follow in this study. That is why the feature values have been aggregated over the distances per each angle value and statistic. After the aggregation, the feature vector size is 12, yielding average statistic (e.g. 'energy') per angle value θ. The distance parameter d controls the scale at which the matrix is computed but goes to a maximum of 4 pixels by design. Therefore the calculated results with different distance d values would not

Haralick statistic	Pixel distance d	Angle θ
Energy	1	0
Homogeneity	2	$\pi/2$
Contrast	3	
Dissimilarity	4	
Angular second moment (asm)		
Correlation		

Table 5.2: Pixel distances and angles used for computation of Haralick statistics. The feature values are aggregated over distance d for each statistic and angle value θ.

be very different. The angle parameter specifies the direction in the image for matrix calculation. Different texture qualities in the same image might happen if the values are uniform in one direction and in the other are changing a lot. Hence the results have been aggregated over distance per each angle to have as representative feature representation as possible.

After Haralick features have been extracted from original texture samples and generated samples obtained by trained models in Section 5.2, the Jensen-Shannon Distance is calculated between feature distributions. The JSD results per each latent space size are visually presented in Figures 5.8, 4.13, 5.22, 5.29.

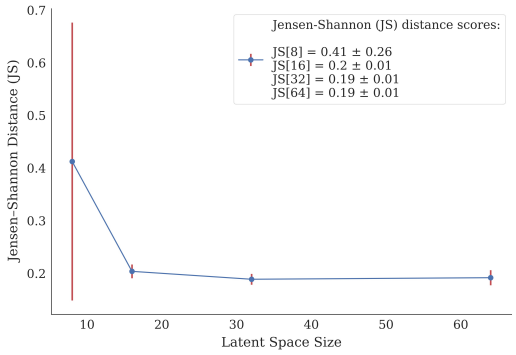

Figure 5.2: (a) Haralick feature - asm.

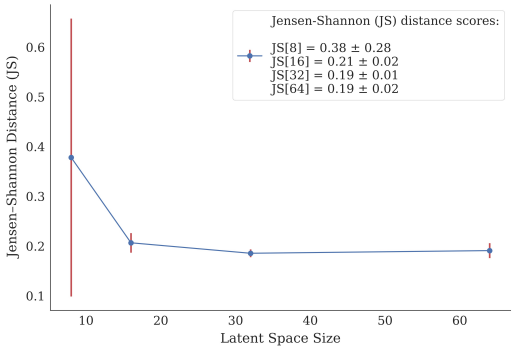

Figure 5.3: (b) Haralick feature - homogeneity.

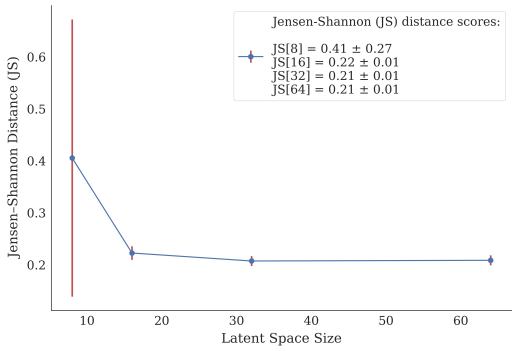

Figure 5.4: (c) Haralick feature - contrast.

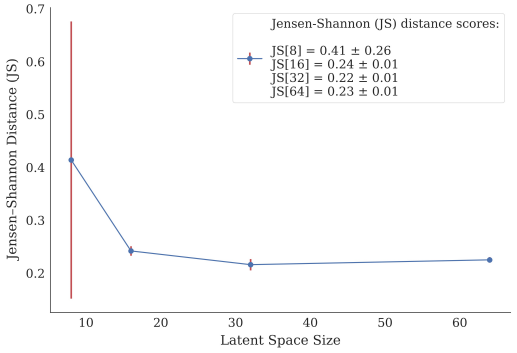

Figure 5.5: (d) Haralick feature - dissimilarity.

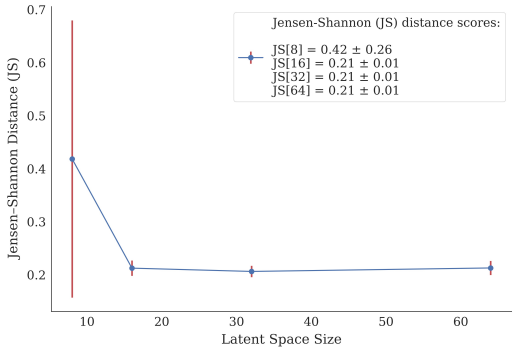

Figure 5.6: (e) Haralick feature - energy.

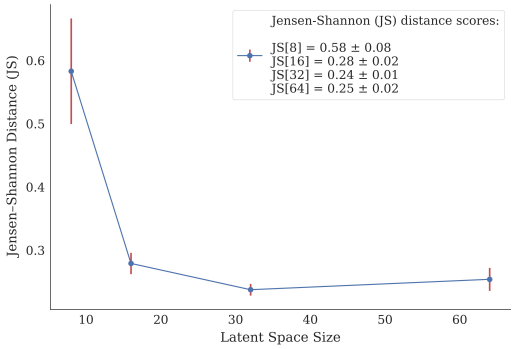

Figure 5.7: (e) Haralick feature - correlation.

Figure 5.8: The average Jensen-Shannon distance calculated with the histogram approach (**bin step=1**) between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal 0. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

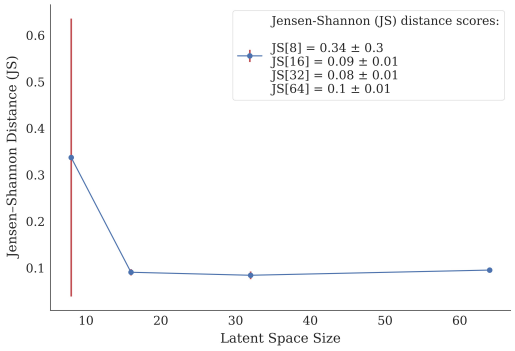

Figure 5.9: (a) Haralick feature - asm.

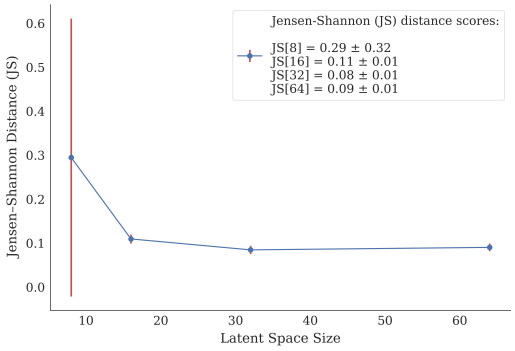

Figure 5.10: (b) Haralick feature - homogeneity.

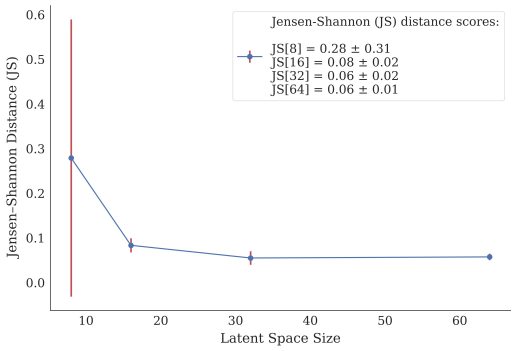

Figure 5.11: (c) Haralick feature - contrast.

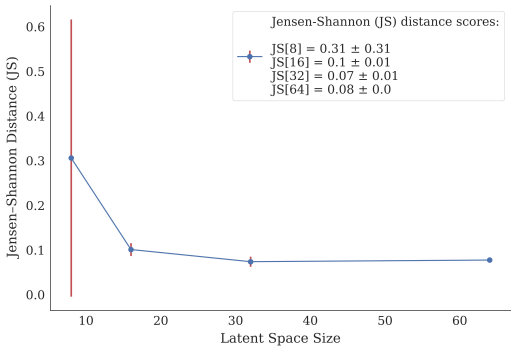

Figure 5.12: (d) Haralick feature - dissimilarity.

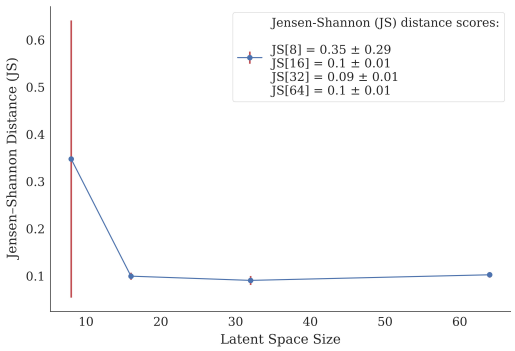

Figure 5.13: (e) Haralick feature - energy.

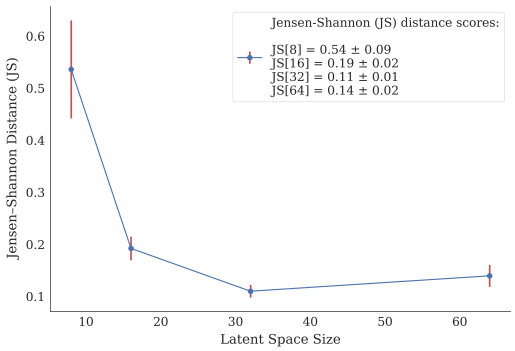

Figure 5.14: (e) Haralick feature - correlation.

Figure 5.15: The average Jensen-Shannon distance calculated with kernel density estimation (**cross validation**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal 0. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

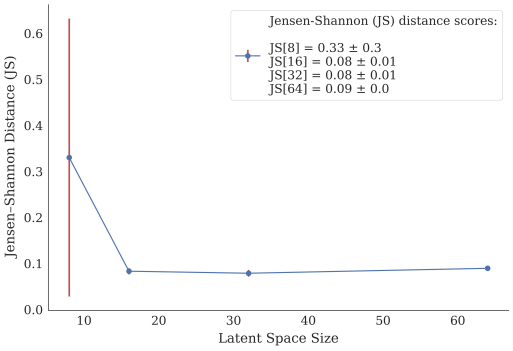

Figure 5.16: (a) Haralick feature - asm.

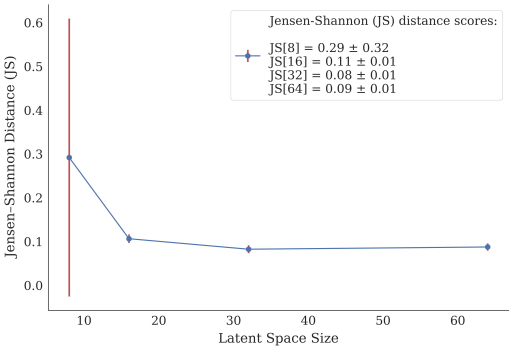

Figure 5.17: (b) Haralick feature - homogeneity.

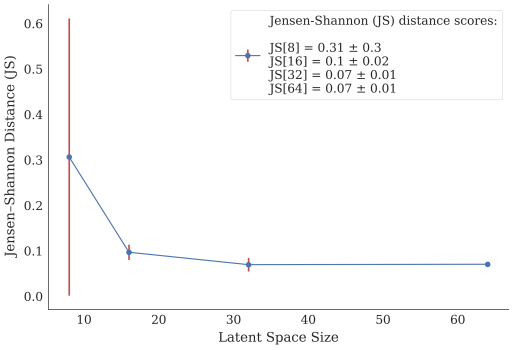

Figure 5.18: (c) Haralick feature - contrast.

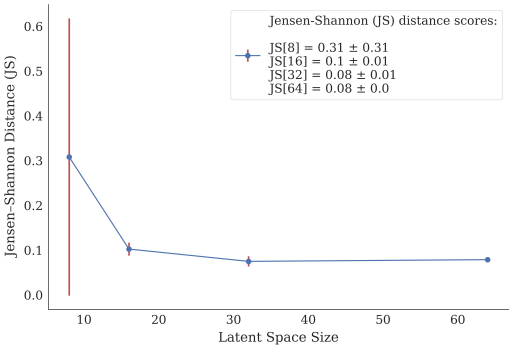

Figure 5.19: (d) Haralick feature - dissimilarity.

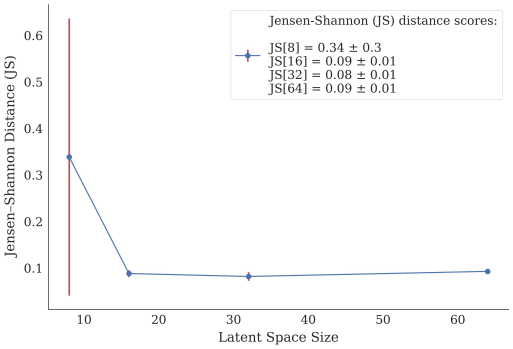

Figure 5.20: (e) Haralick feature - energy.

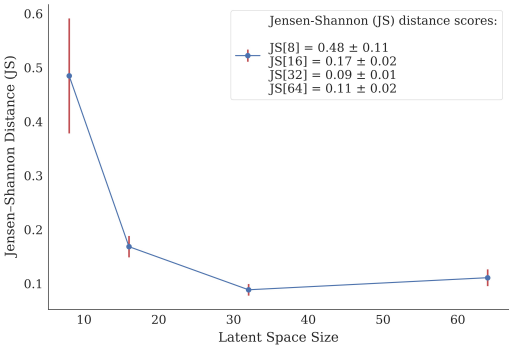

Figure 5.21: (e) Haralick feature - correlation.

Figure 5.22: The average Jensen-Shannon distance calculated with kernel density estimation (**Scott's rule**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal 0. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

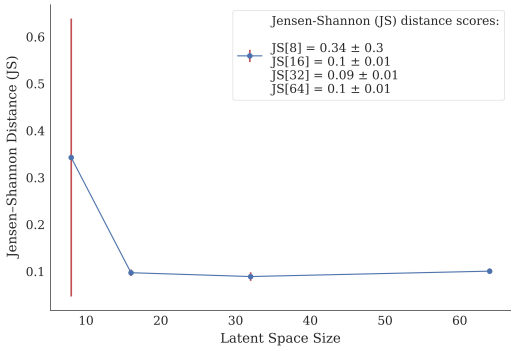

Figure 5.23: (a) Haralick feature - asm.

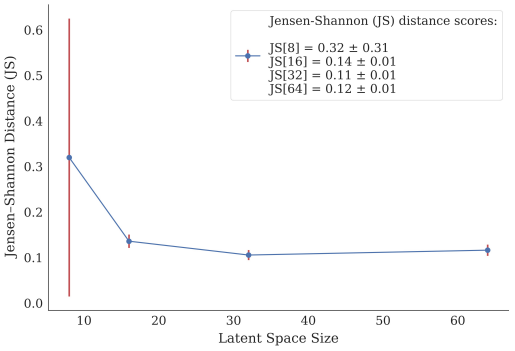

Figure 5.24: (b) Haralick feature - homogeneity.

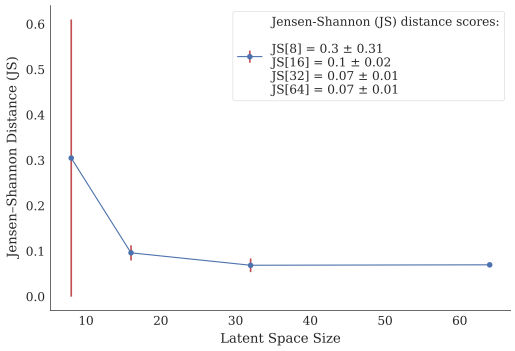

Figure 5.25: (c) Haralick feature - contrast.

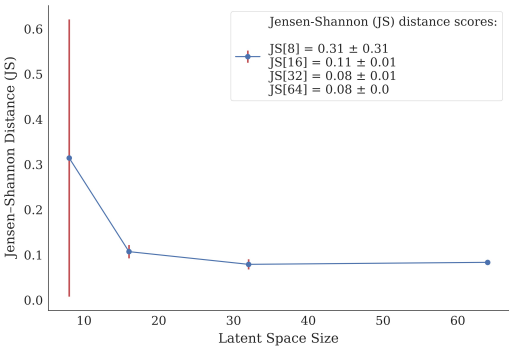

Figure 5.26: (d) Haralick feature - dissimilarity.

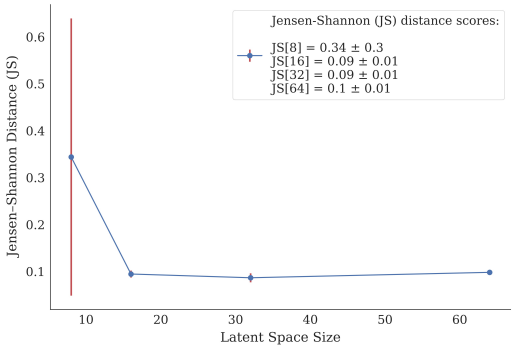

Figure 5.27: (e) Haralick feature - energy.

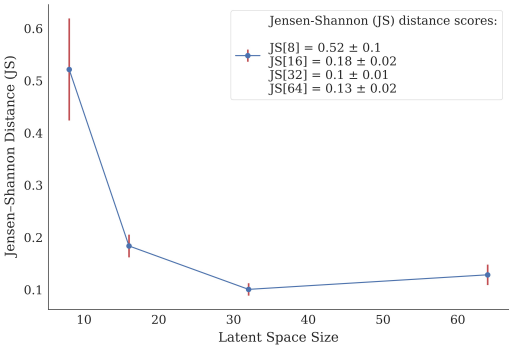

Figure 5.28: (e) Haralick feature - correlation.

Figure 5.29: The average Jensen-Shannon distance calculated with kernel density estimation (**diffusion**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal 0. In total, six different models were trained per each latent space size from 8, 16, 32, and 64.. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

5.5 Application of evaluation procedure based on adapted FID metric

In this section, the adapted FID metric is utilized to quantify the similarity between Haralick feature distributions. Normalized Haralick features are stacked together to get a single vector representation of the input image. The result is obtained by following instructions outlined in Section 4.6 and displayed in Figure 5.30.

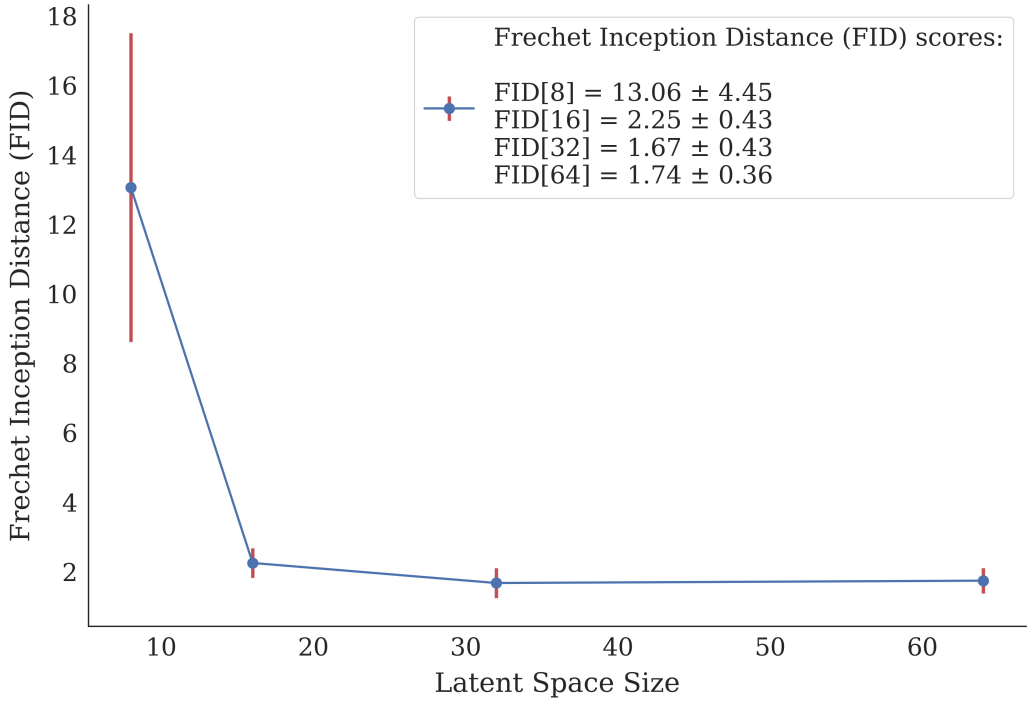

Figure 5.30: The adjusted Frechet inception distance is calculated between the original feature distribution and synthetic feature distribution obtained from StyleALAE models with varying latent space sizes. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the adjusted Frechet inception distance are annotated in the legend, including the standard deviation.

A similar elbow pattern is observed in adapted FID metric results as in JSD-based outcomes. The model with latent space size L8 has the worst performance, evidenced by a big mean score, whereas models with latent space size L16, L32, and L64 have comparable mean and variance scores. Nevertheless, it is still unclear which latent space size is optimal, motivating the need for the evaluation schema.

5.6 Outcomes of evaluation schema

By applying the same steps as in the designed evaluation schema in section 4.7, first, the metric means are being tested to identify the latent sizes with mean scores being minimum (or comparable to a minimum). After this, the variance of scores is checked to eliminate latent sizes with a vast variance (for reasoning, see 4.7).

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Current feature name being used for scoring: asm with angle parameter θ set to 0			
L8: 0.412±0.289	Passed	Failed	0
L16: 0.204±0.014	Failed	-	0
L32: 0.189±0.011	min Mean	min Variance	1
L64: 0.192±0.016	Passed	Passed	1
Current feature name being used for scoring: homogeneity with angle parameter θ set to 0			
L8: 0.378±0.306	Passed	Failed	0
L16: 0.207±0.022	Failed	-	0
L32: 0.185±0.009	min Mean	min Variance	1
L64: 0.191±0.016	Passed	Passed	1
Current feature name being used for scoring: contrast with angle parameter θ set to 0			
L8: 0.406±0.292	Passed	Failed	0
L16: 0.223±0.014	Failed	-	0
L32: 0.207±0.01	min Mean	min Variance	1
L64: 0.209±0.011	Passed	Passed	1
Current feature name being used for scoring: dissimilarity with angle parameter θ set to 0			
L8: 0.414±0.287	Passed	Failed	0
L16: 0.242±0.01	Failed	-	0
L32: 0.216±0.012	min Mean	Passed	1
L64: 0.225±0.006	Passed	min Variance	1
Current feature name being used for scoring: energy with angle parameter θ set to 0			
L8: 0.418±0.286	Passed	Failed	0
L16: 0.213±0.016	Passed	Passed	1
L32: 0.206±0.012	min Mean	min Variance	1
L64: 0.213±0.015	Passed	Passed	1
Current feature name being used for scoring: correlation with angle parameter θ set to 0			
L8: 0.583±0.091	Failed	Failed	0
L16: 0.279±0.018	Failed	Failed	0
L32: 0.238±0.011	min Mean	min Variance	1
L64: 0.254±0.02	Passed	Passed	1
Final latent space size (L) scores: L8: 0, L16: 1, **L32: 6**, **L64: 6**			

Table 5.3: Evaluation schema applied to results of JSD metric calculated from texture features with histogram technique (bin step = 1).

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Current feature name being used for scoring: asm with angle parameter θ set to 0			
L8: 0.337±0.327	Passed	Failed	0
L16: 0.091±0.007	Passed	min Variance	1
L32: 0.084±0.009	min Mean	Passed	1
L64: 0.095±0.006	Failed	-	0
Current feature name being used for scoring: homogeneity with angle parameter θ set to 0			
L8: 0.295±0.346	Passed	Failed	0
L16: 0.109±0.011	Failed	-	0
L32: 0.085±0.01	min Mean	Passed	1
L64: 0.09±0.01	Passed	min Variance	1
Current feature name being used for scoring: contrast with angle parameter θ set to 0			
L8: 0.279±0.34	Passed	Failed	0
L16: 0.083±0.017	Failed	-	0
L32: 0.055±0.017	min Mean	Passed	1
L64: 0.057±0.008	Passed	min Variance	1
Current feature name being used for scoring: dissimilarity with angle parameter θ set to 0			
L8: 0.306±0.34	Passed	Failed	0
L16: 0.101±0.016	Failed	-	0
L32: 0.074±0.013	min Mean	Failed	0
L64: 0.077±0.004	Passed	min Variance	1
Current feature name being used for scoring: energy with angle parameter θ set to 0			
L8: 0.348±0.322	Passed	Failed	0
L16: 0.1±0.008	Passed	min Variance	1
L32: 0.091±0.01	min Mean	Passed	1
L64: 0.102±0.006	Failed	-	0
Current feature name being used for scoring: correlation with angle parameter θ set to 0			
L8: 0.536±0.103	Failed	Failed	0
L16: 0.192±0.025	Failed	Failed	0
L32: 0.11±0.013	min Mean	min Variance	1
L64: 0.14±0.023	Failed	Failed	0
Final latent space size (L) scores: L8: 0, L16: 2, **L32: 5**, L64: 3			

Table 5.4: Evaluation schema applied to results of JSD metric calculated from texture features with KDE technique (cross-validation method used to estimate bandwidth parameter).

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Current feature name being used for scoring: asm with angle parameter θ set to 0			
L8: 0.33±0.331	Passed	Failed	0
L16: 0.083±0.007	Passed	min Variance	1
L32: 0.079±0.008	min Mean	Passed	1
L64: 0.09±0.005	Failed	-	0
Current feature name being used for scoring: homogeneity with angle parameter θ set to 0			
L8: 0.292±0.348	Passed	Failed	0
L16: 0.107±0.011	Failed	-	0
L32: 0.083±0.01	min Mean	Passed	1
L64: 0.088±0.009	Passed	min Variance	1
Current feature name being used for scoring: contrast with angle parameter θ set to 0			
L8: 0.307±0.334	Passed	Failed	0
L16: 0.097±0.018	Failed	-	0
L32: 0.07±0.016	min Mean	Failed	0
L64: 0.071±0.006	Passed	min Variance	1
Current feature name being used for scoring: dissimilarity with angle parameter θ set to 0			
L8: 0.308±0.339	Passed	Failed	0
L16: 0.103±0.016	Failed	-	0
L32: 0.075±0.012	min Mean	Failed	0
L64: 0.079±0.004	Passed	min Variance	1
Current feature name being used for scoring: energy with angle parameter θ set to 0			
L8: 0.339±0.326	Passed	Failed	0
L16: 0.088±0.008	Passed	min Variance	1
L32: 0.082±0.01	min Mean	Passed	1
L64: 0.093±0.006	Failed	-	0
Current feature name being used for scoring: correlation with angle parameter θ set to 0			
L8: 0.485±0.117	Failed	Failed	0
L16: 0.169±0.022	Failed	Failed	0
L32: 0.089±0.012	min Mean	min Variance	1
L64: 0.111±0.017	Failed	Failed	0
Final latent space size (L) scores: L8: 0, L16: 2, **L32: 4**, L64: 3			

Table 5.5: Evaluation schema applied to results of JSD metric calculated from texture features with KDE technique (Scott's rule used to estimate bandwidth parameter).

JSD mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Latent space size (L) scores
Current feature name being used for scoring: asm with angle parameter θ set to 0			
L8: 0.343±0.324	Passed	Failed	0
L16: 0.097±0.008	Passed	min Variance	1
L32: 0.089±0.01	min Mean	Passed	1
L64: 0.101±0.007	Failed	-	0
Current feature name being used for scoring: homogeneity with angle parameter θ set to 0			
L8: 0.32±0.334	Passed	Failed	0
L16: 0.136±0.016	Failed	-	0
L32: 0.106±0.012	min Mean	min Variance	1
L64: 0.116±0.013	Passed	Passed	1
Current feature name being used for scoring: contrast with angle parameter θ set to 0			
L8: 0.305±0.334	Passed	Failed	0
L16: 0.096±0.018	Failed	-	0
L32: 0.069±0.016	min Mean	Failed	0
L64: 0.07±0.006	Passed	min Variance	1
Current feature name being used for scoring: dissimilarity with angle parameter θ set to 0			
L8: 0.314±0.336	Passed	Failed	0
L16: 0.107±0.016	Failed	-	0
L32: 0.079±0.012	min Mean	Failed	0
L64: 0.083±0.004	Passed	min Variance	1
Current feature name being used for scoring: energy with angle parameter θ set to 0			
L8: 0.344±0.324	Passed	Failed	0
L16: 0.095±0.008	Passed	min Variance	1
L32: 0.087±0.01	min Mean	Passed	1
L64: 0.098±0.006	Failed	-	0
Current feature name being used for scoring: correlation with angle parameter θ set to 0			
L8: 0.522±0.107	Failed	-	0
L16: 0.184±0.024	Failed	-	0
L32: 0.101±0.013	min Mean	min Variance	1
L64: 0.128±0.021	Failed	-	0
Final latent space size (L) scores: L8: 0, L16: 2, **L32: 4**, L64: 3			

Table 5.6: Evaluation schema applied to results of JSD metric calculated from texture features with KDE technique (diffusion method used to estimate bandwidth parameter).

Adjusted FID mean and std results	Result of pairwise T-test between min mean score and other mean scores	Result of pairwise F-test between min variance score and other variance scores	Final latent space size (L) scores
L8: 13.061±4.871	Failed	-	0
L16: 2.25±0.467	Failed	-	0
L32: 1.673±0.473	min Mean	Passed	1
L64: 1.74±0.398	Passed	min Variance	1

Table 5.7: Evaluation schema applied to results of adjusted FID metric calculated based on Haralick features extracted from original and generated samples.

5.7 Evaluation framework results

As seen from Table 5.8, the final outcomes of applying the evaluation framework to the real-world industrial dataset are the same for different methods used for evaluation similarity estimation. In the case of equal scores for some latent space sizes, the smaller latent space size is preferred since it contributes to learning compact, disentangled latent space. As a result, the latent space of 32 is identified as the optimal hyperparameter value by different methods through a feature-consistent performance evaluation.

Method	JSD with binning (bin step=1)	JSD with KDE (cross-validation)	JSD with KDE (Scott's rule)	JSD with KDE (diffusion)	Adapted FID metric
Detected optimal latent size	L32, L64	L32	L32	L32	L32, L64

Table 5.8: Final results of evaluation framework applied to textures dataset.

Similarly to intermediate conclusions in Section 4.9, kernel density estimation with diffusion method as bandwidth technique is the recommended approach to calculate the JSD as a similarity estimate between distributions. It appears to output more robust and reliable compared to alternatives, as explained in Section 4.5.2.

The designed evaluation framework operates on the feature space of data samples. Therefore, it strongly depends on the feature extraction procedure and selected features to differentiate between original and generated samples. As a potential improvement, Tamura features can be utilized to represent textures [38] as a complement to Haralick features [13]. The evaluation framework yielded meaningful results for the sample sizes of 1.000. Nevertheless, it is recommended to use a larger sample size to have large support for the estimation of sample distribution and avoid selection bias.

In the evaluation schema part, the T-Test and F-Test are used to reliably estimate the difference between the mean and variance of distance scores. Following the standard practice in statistics, the statistical level α is set to 5%, which might not fit the experiments with six scores obtained by six trained models. Hence, the current value can be adjusted depending on the application and task-specific requirements.

6 Conclusion

In industrial machine vision applications, generative deep learning methods have been employed to get accurate predictions for practical tasks (e.g., defect detection) while providing transparency in their predictions. The explainability property of generative models is the key advantage over discriminative deep learning methods, further facilitating their adaption in the industry. The state-of-the-art generative model used for such purposes is StyleALAE - a prominent architecture from the adversarial latent autoencoders field. It combines the generative abilities of style-based generative adversarial networks (StyleGANs) and the representational properties of autoencoders. Consequently, StyleALAE is able to assess the properties of embedded images employing their latent space representations and to synthesize high-quality images by learning a mapping from a disentangled latent space onto the image manifold. Thereby, learned representations can be identified by interpreting the latent space and used to control the properties of the synthesized industrial machine vision data. However, a trade-off between the dimensionality of the StyleALAE's latent space and the quality of generated images must be found. While a smaller latent space is easier to interpret, a higher degree of compression must be achieved during training to learn all quality characteristics properly.

In this thesis, the evaluation framework was developed to assess the capabilities of generative models (in the form of StyleALAE) to learn the features and characteristics of the original data manifold consistently and reliably. The primary motivation for the approach is inspired by the requirement in the industry. In order to successfully perform downstream tasks such as quality control, generative models have to capture original feature distributions (corresponding to quality characteristics) of real data as precisely as possible. For practical reasons, it is desired to keep the number of variables that contain feature information as small as possible. For this reason, the minimum latent space size that still provides all relevant information is the advantageous latent space size. The optimal hyperparameter value can be further identified through a feature-consistent performance evaluation.

The evaluation procedure was first designed on a synthetic ellipse dataset with known human-interpretable features, where the optimal latent space size of StyleALAE architecture was identified. Then the evaluation procedure was applied to the real-world industrial machine vision dataset consisting of images of aluminum die casting surfaces. With dataset-specific features, the image quality of multiple StyleALAEs trained with different latent space dimensionalities was compared using statistical tests to identify an advantageous latent space dimension. It was found that even for slightly different extracted features and for different methods used in the evaluation scheme, the developed evaluation procedure suggested the same optimal latent space size, indicating the evaluation framework's robustness.

Unlike other automatic metrics available in the generative field, the proposed evaluation framework can indicate the mode collapse issue of generative models. In case of limited variability of generated samples, the shape of feature distributions will be different from feature distributions obtained from original samples, which the evaluation procedure will automatically capture. Furthermore, the introduced procedure is model agnostic and only requires original and generated samples. Hence the performance in learning original features from the training dataset of any generative model can be estimated. In addition, the evaluation framework is

computationally efficient, linearly scales with the increased number of samples and is easy to use out of the box, which make it a convenient quantitative measure to assess generative models performance in terms of feature consistency criterion.

The introduced evaluation framework provides functionality to identify weaknesses of a generative model by examining the model behavior on particular features of interest. Future work might involve designing a special experimental setting in which generative models can be tested on how they learn specific image characteristics (e.g., high-frequency components). If a generative model fails to learn specific features, this could indicate potential modifications and extensions of the generative model to improve the preservation of application-specific features.

The objective of representation learning with generative models is to create interpretable and navigable latent spaces. The smaller the latent space is, the easier it is to analyze it for downstream tasks further. The optimal latent space hyperparameter value contributes significantly towards learning disentangled feature representations but does not necessarily guarantee it. One of the potential avenues of future research is to investigate latent space's disentangled properties, which could become another evaluation criterion used to assess generative models.

Different automatic evaluation metrics in generative modeling have different objectives, strengths, and shortcomings. There is no one-fits-all evaluation method. Therefore, the evaluation of generative models must be made considering the intended application. The introduced evaluation framework is initially designed for quality control and metrologically interpretable feature extraction tasks, and it is not meant to replace existing automatic methods but to complement them.

References

[1] T. W. Anderson. "On the Distribution of the Two-Sample Cramer-von Mises Criterion". In: *The Annals of Mathematical Statistics* 33.3 (1962), pp. 1148–1159. DOI: 10.1214/aoms/1177704477.

[2] S. T. Barratt and R. Sharma. "A Note on the Inception Score". In: *ArXiv* abs/1801.01973 (2018).

[3] A. Borji. "Pros and cons of GAN evaluation measures". In: *Computer Vision and Image Understanding* 179 (2019), pp. 41–65. DOI: 10.1016/j.cviu.2018.10.009.

[4] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. "Kernel density estimation via diffusion". In: *The Annals of Statistics* 38.5 (2010), pp. 2916–2957. DOI: 10.1214/10-AOS799.

[5] R. W. Conners and C. A. Harlow. "A theoretical comparison of texture algorithms". In: *IEEE transactions on pattern analysis and machine intelligence* 3 (1980), pp. 204–222.

[6] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 1486–1494.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019. DOI: 10.18653/v1/N19-1423.

[8] A. Dosovitskiy and T. Brox. "Generating Images with Perceptual Similarity Metrics based on Deep Networks". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[9] D. C. Dowson and B. V. Landau. "The Fréchet distance between multivariate normal distributions". In: *Journal of Multivariate Analysis* 12.3 (1982), pp. 450–455.

[10] A. W. Fitzgibbon and R. B. Fisher. "A Buyer's Guide to Conic Fitting". In: *Proceedings of the 6th British Conference on Machine Vision (Vol. 2)*. BMVC '95. Birmingham, United Kingdom: BMVA Press, 1995, pp. 513–522. ISBN: 0952189828.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680.

[12] E. Gregolinska, R. Khanam, F. Lefort, and P. Prashanth. *Capturing the true value of Industry 4.0*. 2022.

[13] R. M. Haralick, K. Shanmugam, and I. Dinstein. "Textural Features for Image Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3.6 (1973), pp. 610–621. DOI: 10.1109/TSMC.1973.4309314.

[14] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 6626–6637.

[15] G. Huang, Y. Yuan, Q. Xu, C. Guo, Y. Sun, F. Wu, and K. Weinberger. "An empirical study on evaluation metrics of generative adversarial networks". In: *arXiv preprint arXiv:1806.07755* (2018). DOI: 10.48550/arXiv.1806.07755.

[16] X. Huang and S. Belongie. "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization". In: *International Conference on Learning Representations*. 2017.

[17] A. Humeau-Heurtier. "Texture feature extraction methods: a survey". In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 8975–9000. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2890743.

[18] F. J. M. Jr. "The Kolmogorov-Smirnov Test for Goodness of Fit". In: *Journal of the American Statistical Association* 46.253 (1951), pp. 68–78. DOI: 10.1080/01621459.1951.10500769.

[19] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[20] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*. Computer Vision Foundation / IEEE, 2019, pp. 4401–4410. DOI: 10.1109/CVPR.2019.00453.

[21] D. Kingma and J. Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014). DOI: 10.48550/arXiv.1412.6980.

[22] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR*. Ed. by Y. Bengio and Y. LeCun. 2014.

[23] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. "Improved Precision and Recall Metric for Assessing Generative Models". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 3927–3936.

[24] S. Laine. *Feature-Based Metrics for Exploring the Latent Space of Generative Models*. 2018.

[25] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. "Are GANs Created Equal? A Large-Scale Study". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 700–709.

[26] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. DOI: 10.1214/aoms/1177730491.

[27] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *Proceedings of The 33rd International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1747–1756.

[28] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. "Conditional Image Generation with PixelCNN Decoders". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS 2016. Curran Associates Inc., 2016, pp. 4797–4805. ISBN: 9781510838819.

[29] S. Pidhorskyi, D. A. Adjeroh, and G. Doretto. "Adversarial latent autoencoders". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14104–14113.

[30] F. C. Porter. "Testing Consistency of Two Histograms". In: *arXiv: Data Analysis, Statistics and Probability* (2008).

[31] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. "Variational Approaches for Auto-Encoding Generative Adversarial Networks". In: *ArXiv* abs/1706.04987 (2017).

[32] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. "Assessing Generative Models via Precision and Recall". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 5228–5237.

[33] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 2234–2242.

[34] R. H. Schmitt, D. Wolfschläger, E. Masliankova, and B. Montavon. "Metrologically interpretable feature extraction for industrial machine vision using generative deep learning". In: *CIRP Annals* 71.1 (2022), pp. 433–436. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2022.03.016.

[35] D. W. Scott. "Multivariate density estimation. Theory, practice, and visualization." In: *Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics.* (1992). DOI: 10.1002/9780470316849.

[36] S. Suzuki and K. be. "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7.

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[38] H. Tamura, S. Mori, and T. Yamawaki. "Textural Features Corresponding to Visual Perception". In: *IEEE Transactions on Systems, Man, and Cybernetics* 8.6 (June 1978). Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, pp. 460–473. ISSN: 2168-2909. DOI: 10.1109/TSMC.1978.4309999.

[39] D. Wolfschläger, R. Yermakov, B. Montavon, B. Berkels, and R. H. Schmitt. "Identifying the advantageous latent space dimensionality for StyleGANs used in industrial machine vision applications". In: *AI and Optical Data Sciences IV*. Ed. by B. Jalali and K.-i. Kitayama. Vol. 12438. International Society for Optics and Photonics. SPIE, 2023, 124380R. DOI: 10.1117/12.2646326.

[40] S. Zhou, M. Gordon, R. Krishna, A. Narcomey, L. F. Fei-Fei, and M. Bernstein. "HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 3449–3461.

[41] Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, and Y. Yu. "Activation Maximization Generative Adversarial Nets". In: *International Conference on Learning Representations*. 2018.

A Distribution comparison of ellipse features between original and generated samples

The figures demonstrating the distribution comparison of ellipse features of real images with the images generated by StyleALAE models of latent space sizes 4, 8, 32, 128 and 512 are shown in Figure A.2 and A.1.

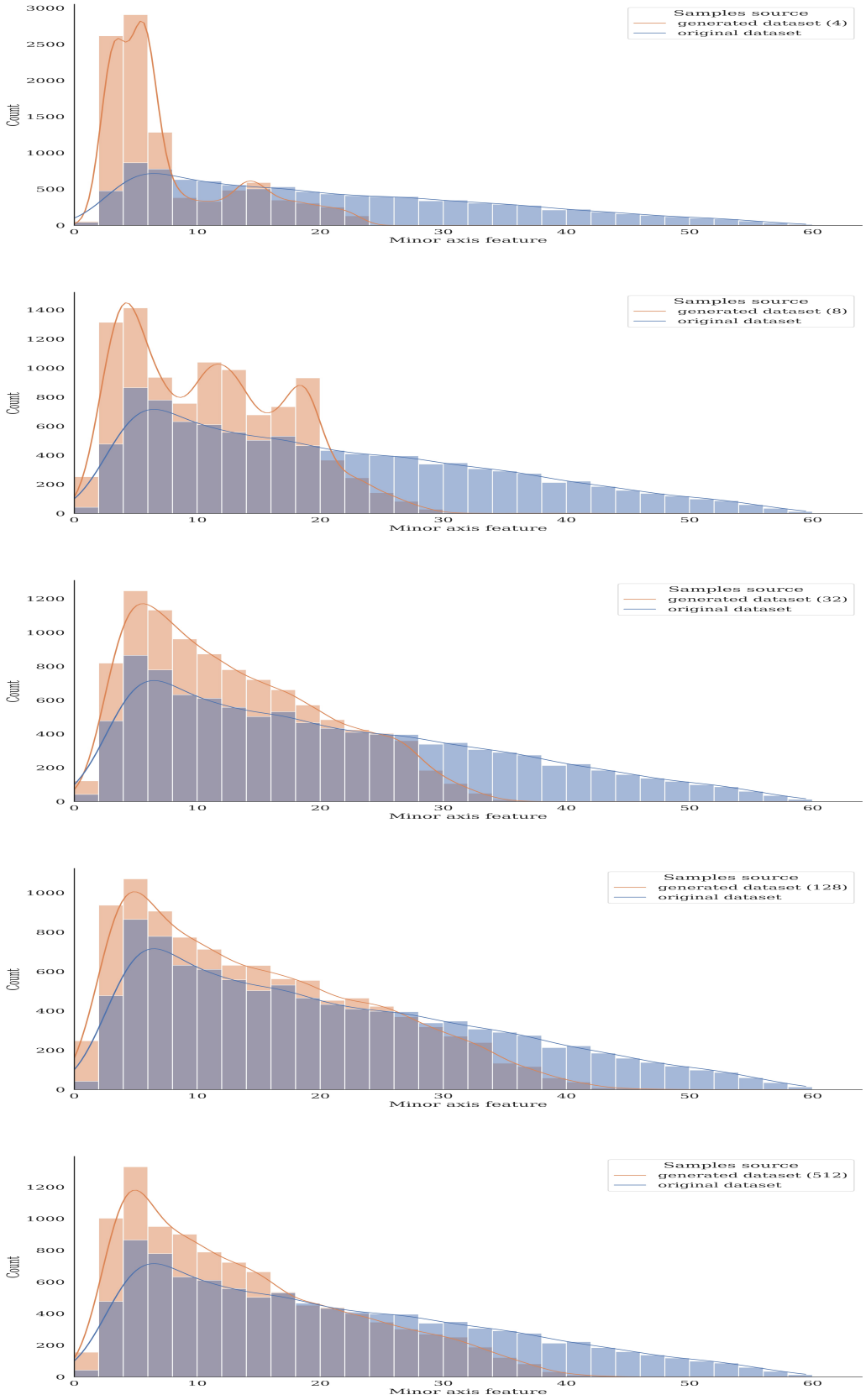

Figure A.1: The figure displays the $minor\ axis$ distributions. The five plots show the feature distribution of real images (blue color) and the distribution of features extracted from generated images (orange color) obtained by five model variations. From top to bottom, the model latent space sizes are 4, 8, 32, 128, and 512. The bin step of size two is chosen to plot histograms.

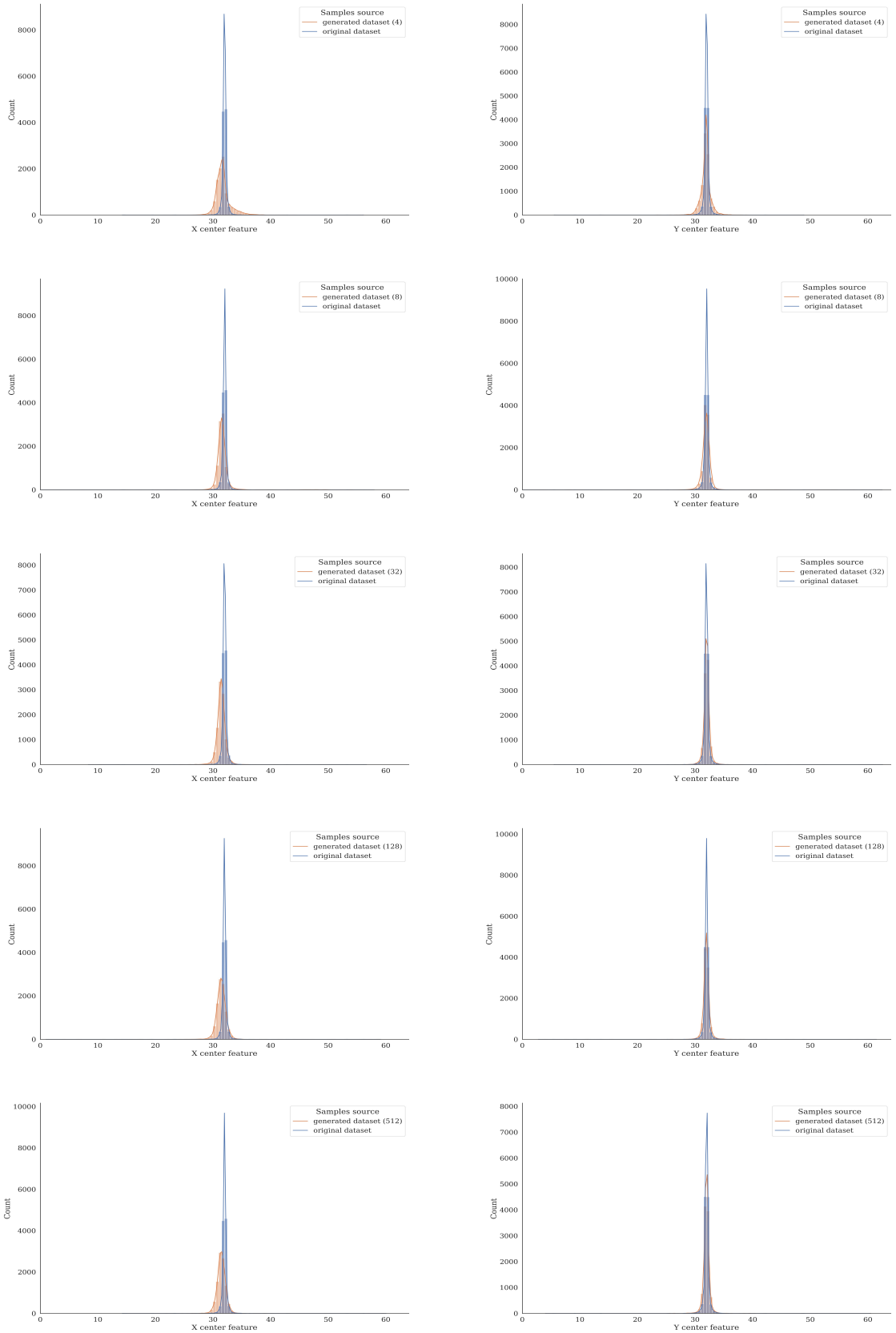

Figure A.2: The left side of a figure displays the x *center* distribution, while the right side - y *center* distribution. The five plots show the feature distribution of real images (blue color) and the distribution of features extracted from generated images (orange color) obtained by five model variations. From top to bottom, the model latent space sizes are 4, 8, 32, 128, and 512. The bin step of size two is chosen to plot histograms.

B Similarity estimation with JSD metric calculated from Haralick feature distributions of original and generated samples

Haralick features with the angle value θ set to $\pi/2$ have been extracted from original and generated texture samples. The JSD results per each latent space size are visually presented in Figures B.7, B.14, B.21, B.28.

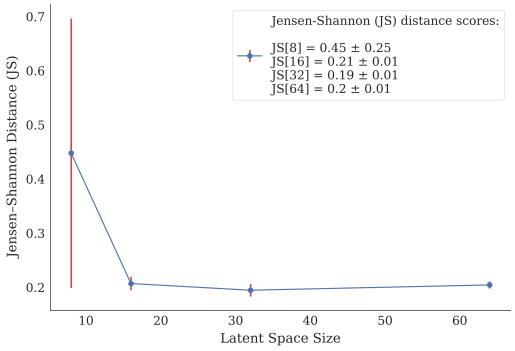

Figure B.1: (a) Haralick feature - asm.

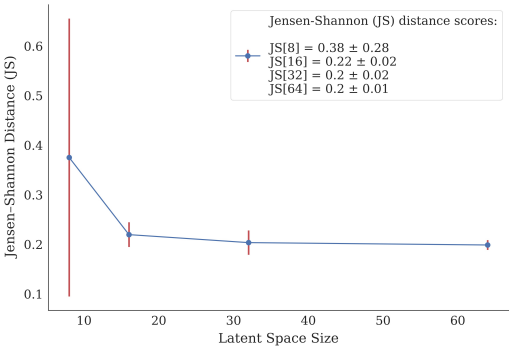

Figure B.2: (b) Haralick feature - homogeneity.

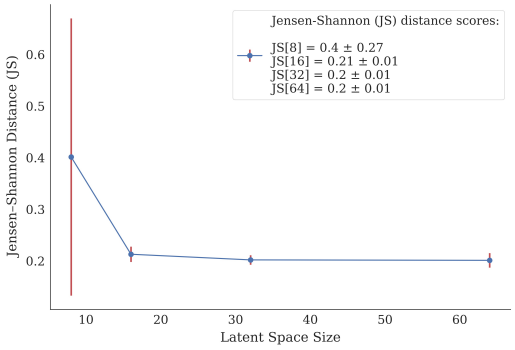

Figure B.3: (c) Haralick feature - contrast.

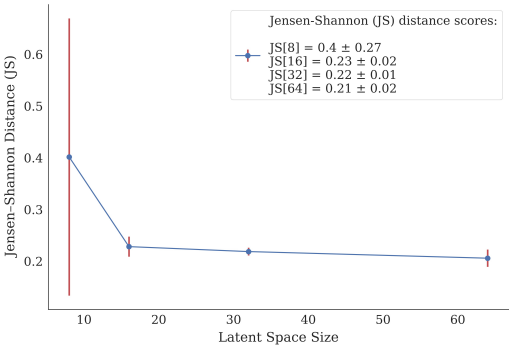

Figure B.4: (d) Haralick feature - dissimilarity.

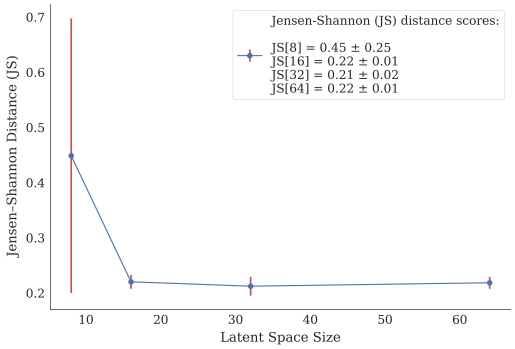

Figure B.5: (e) Haralick feature - energy.

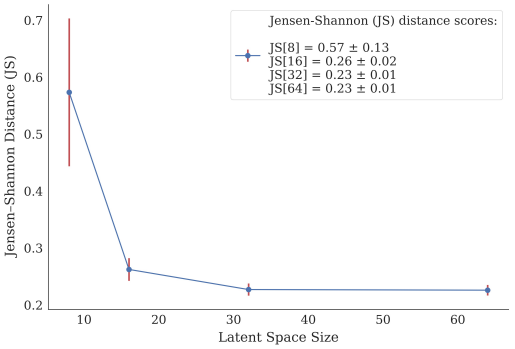

Figure B.6: (e) Haralick feature - correlation.

Figure B.7: The average Jensen-Shannon distance calculated with the histogram approach (**bin step=1**) between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal $\pi/2$. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

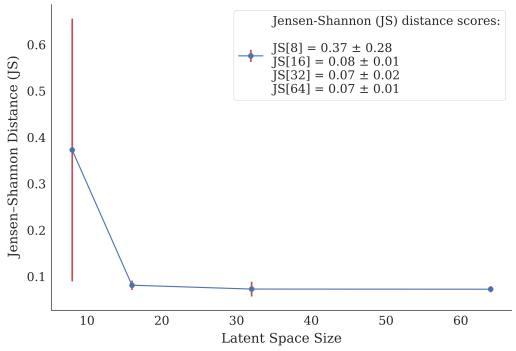

Figure B.8: (a) Haralick feature - asm.

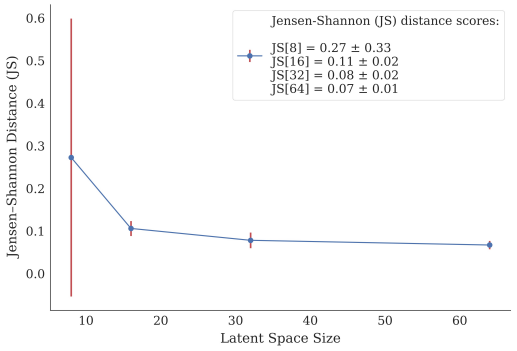

Figure B.9: (b) Haralick feature - homogeneity.

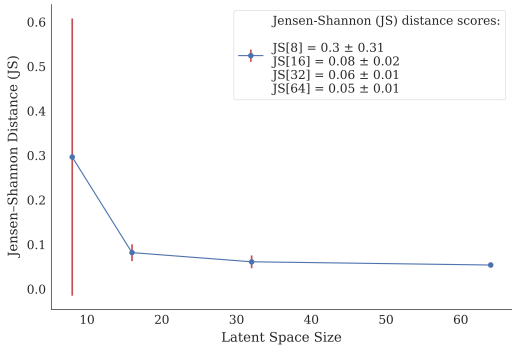

Figure B.10: (c) Haralick feature - contrast.

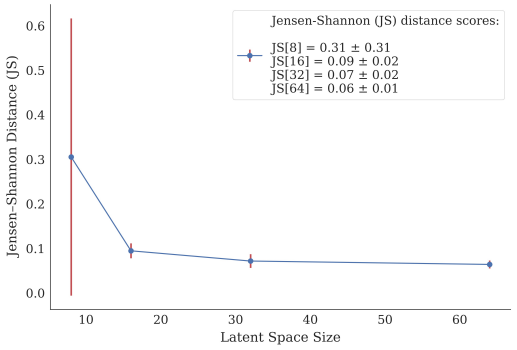

Figure B.11: (d) Haralick feature - dissimilarity.

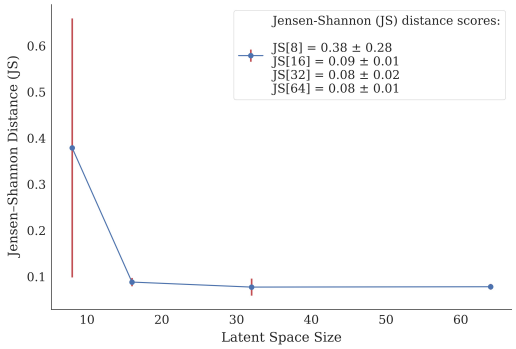

Figure B.12: (e) Haralick feature - energy.

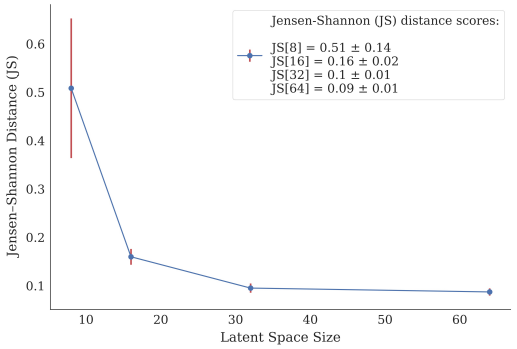

Figure B.13: (e) Haralick feature - correlation.

Figure B.14: The average Jensen-Shannon distance calculated with kernel density estimation (**cross-validation**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal $\pi/2$. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

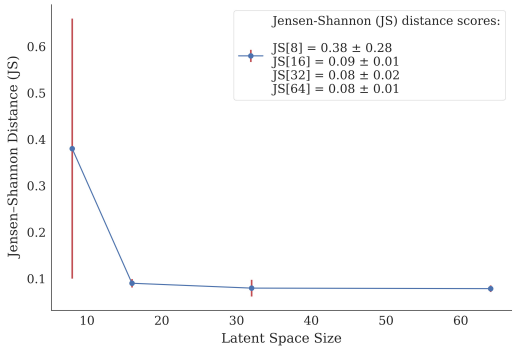

Figure B.15: (a) Haralick feature - asm.

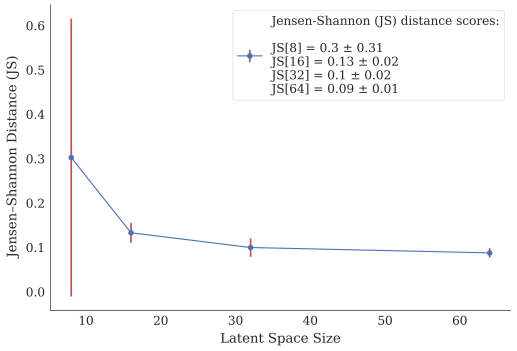

Figure B.16: (b) Haralick feature - homogeneity.

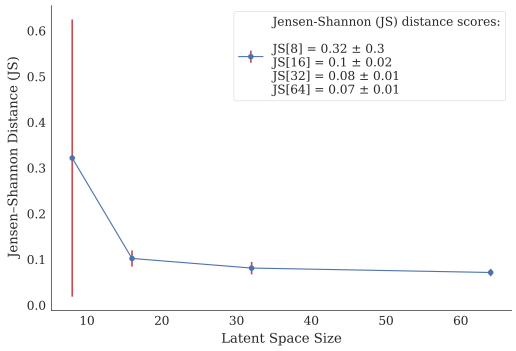

Figure B.17: (c) Haralick feature - contrast.

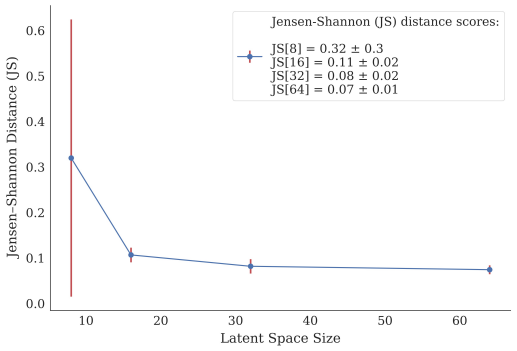

Figure B.18: (d) Haralick feature - dissimilarity.

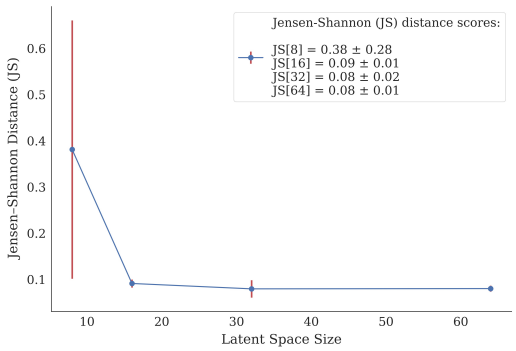

Figure B.19: (e) Haralick feature - energy.

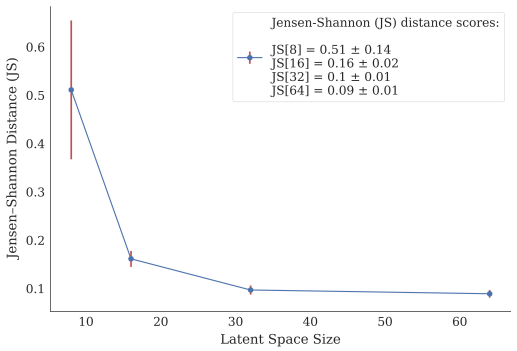

Figure B.20: (e) Haralick feature - correlation.

Figure B.21: The average Jensen-Shannon distance calculated with kernel density estimation (**Scott's rule**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal $\pi/2$. In total, six different models were trained per each latent space size from 8, 16, 32, and 64. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.

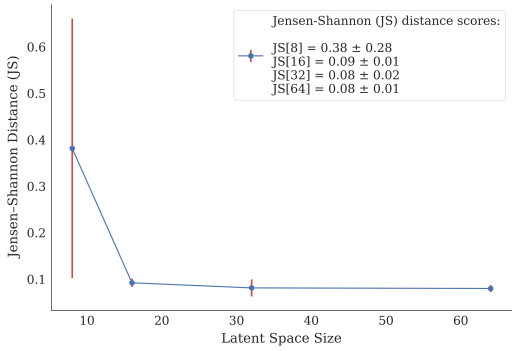

Figure B.22: (a) Haralick feature - asm.

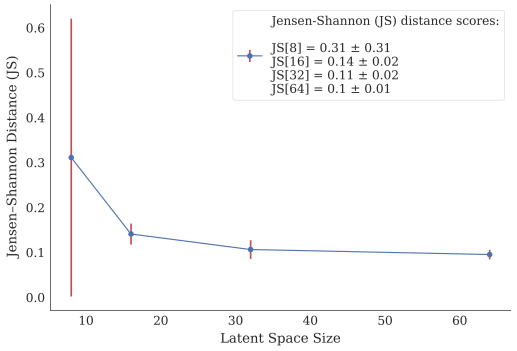

Figure B.23: (b) Haralick feature - homogeneity.

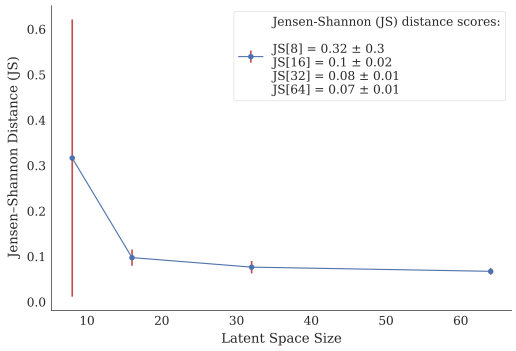

Figure B.24: (c) Haralick feature - contrast.

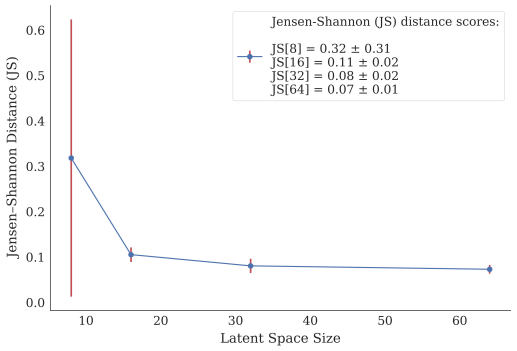

Figure B.25: (d) Haralick feature - dissimilarity.

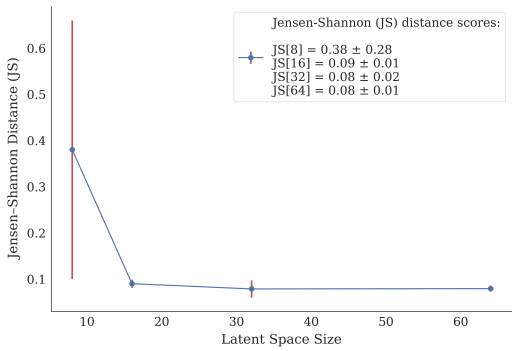

Figure B.26: (e) Haralick feature - energy.

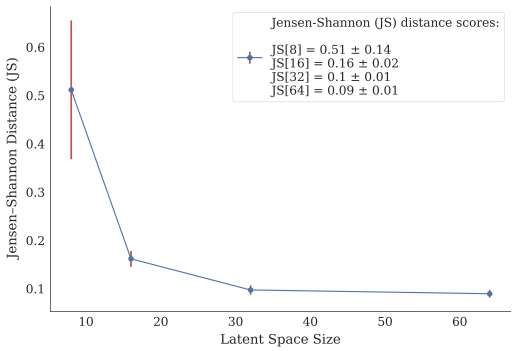

Figure B.27: (e) Haralick feature - correlation.

Figure B.28: The average Jensen-Shannon distance calculated with kernel density estimation (**diffusion**) method between the distributions of features of real images and the distributions of features of generated images by models with varying latent space sizes. Haralick features are aggregated over the distance parameter d for the angle value θ equal $\pi/2$. In total, six different models were trained per each latent space size from 8, 16, 32, and 64.. The average values of the Jensen-Shannon distances are annotated in the legend, including the standard deviation.